

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	6
1.1	Imports	6
1.2	Group Actions	7
1.3	Orbit and stabiliser	7
1.4	Stabiliser Theorems	7
1.5	Picking representatives from cosets	10
1.6	Orbit-Stabiliser Theorem	11
2	Rotational Symmetries of the Tetrahedron	13
2.1	Definition of the Tetrahedron and its Rotations	13
2.2	Properties of Rotations	14
2.3	Inversions	29
2.4	The Tetrahedral Group	30
2.5	Counting Orbits	32
2.6	Counting Stabilisers	33
2.7	Proving Finiteness	36
2.8	Order of the Group	37

theory *Left-Coset*

imports

HOL-Algebra.Coset

begin

definition

LCOSETS :: $[-, 'a \text{ set}] \Rightarrow ('a \text{ set})\text{set}$ (*lcosets*₁ - [81] 80)

where *lcosets*_G *H* = $(\bigcup_{a \in \text{carrier } G} \{a <\#_G H\})$

definition

LFactGroup :: $[('a, 'b) \text{ monoid-scheme}, 'a \text{ set}] \Rightarrow ('a \text{ set}) \text{ monoid}$ (**infixl** *LMod* 65)

— Actually defined for groups rather than monoids

where *LFactGroup* *G* *H* = $(\langle \text{carrier} = \text{lcosets}_G H, \text{mult} = \text{set-mult } G, \text{one} = H \rangle)$

lemma (**in** *group*) *lcos-self*: $[\langle x \in \text{carrier } G; \text{subgroup } H \text{ } G \rangle] \implies x \in x <\# H$

apply (*simp add: l-coset-def*)

apply (*blast intro: sym r-one subgroup.subset [THEN subsetD] subgroup.one-closed*)

done

Elements of a left coset are in the carrier

lemma (**in** *subgroup*) *elemcos-carrier*:

assumes *group* *G*

assumes *acarr*: $a \in \text{carrier } G$

and *a'*: $a' \in a <\# H$

shows $a' \in \text{carrier } G$

proof —

interpret *group* *G* **by** *fact*

from *subset* **and** *acarr*

have $a <\# H \subseteq \text{carrier } G$ **by** (*rule l-coset-subset-G*)

from *this* **and** *a'*

show $a' \in \text{carrier } G$

by *fast*

qed

Step one for lemma *rcos-module*

lemma (**in** *subgroup*) *lcos-module-imp*:

assumes *group* *G*

assumes *xcarr*: $x \in \text{carrier } G$

and *x'cos*: $x' \in x <\# H$

shows $(\text{inv } x \otimes x') \in H$
proof –
interpret *group* G **by** *fact*
from $x\text{carr } x'\text{cos}$
 have $x'\text{carr}: x' \in \text{carrier } G$
 by (*rule elemcos-carrier*[*OF is-group*])
from $x\text{carr}$
 have $ix\text{carr}: \text{inv } x \in \text{carrier } G$
 by *simp*
from $x'\text{cos}$
 have $\exists h \in H. x' = x \otimes h$
 unfolding *l-coset-def*
 by *fast*
from *this*
 obtain h
 where $hH: h \in H$
 and $x': x' = x \otimes h$
 by *auto*
from hH **and** *subset*
 have $hcarr: h \in \text{carrier } G$ **by** *fast*
note $\text{carr} = x\text{carr } x'\text{carr } hcarr$
from x' **and** carr
 have $(\text{inv } x) \otimes x' = (\text{inv } x) \otimes (x \otimes h)$ **by** *fast*
also from carr
 have $\dots = (x \otimes \text{inv } x) \otimes h$ **by** (*metis ixcarr l-inv m-assoc r-inv*)
also from carr
 have $\dots = h \otimes 1$ **by** *simp*
also from carr
 have $\dots = h$ **by** *simp*
finally
 have $(\text{inv } x) \otimes x' = h$ **by** *simp*
from hH *this*
 show $(\text{inv } x) \otimes x' \in H$ **by** *simp*
qed

Left cosets are subsets of the carrier.

lemma (*in subgroup*) *lcosets-carrier*:

assumes *group* G

assumes $XH: X \in \text{lcosets } H$

shows $X \subseteq \text{carrier } G$

proof –

interpret *group* G **by** *fact*

from XH **have** $\exists x \in \text{carrier } G. X = x <\# H$

unfolding *LCOSETS-def*

```

    by fast
  from this
    obtain x
      where xcarr: x ∈ carrier G
      and X: X = x <# H
    by fast
  from subset and xcarr
  show X ⊆ carrier G
  unfolding X
  by (rule l-coset-subset-G)
qed

```

```

lemma (in group) lcosets-part-G:
  assumes subgroup H G
  shows ⋃ (lcosets H) = carrier G
proof -
  interpret subgroup H G by fact
  show ?thesis
    apply (rule equalityI)
    apply (force simp add: LCOSETS-def l-coset-def)
    apply (auto simp add: LCOSETS-def intro: lcos-self assms)
  done
qed

```

```

lemma (in group) lcosets-subset-PowG:
  subgroup H G ⇒ lcosets H ⊆ Pow(carrier G)
apply (simp add: LCOSETS-def)
apply (blast dest: l-coset-subset-G subgroup.subset)
done

```

```

lemma (in group) lcos-disjoint:
  assumes subgroup H G
  assumes p: a ∈ lcosets H b ∈ lcosets H a ≠ b
  shows a ∩ b = {}
proof -
  interpret subgroup H G by fact
  from p show ?thesis
    apply (simp add: LCOSETS-def)
    apply (metis inf.cobounded1 inf.cobounded2 l-repr-independence sub-
      group-axioms subset-empty subset-iff)
  done
qed

```

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

lemma (in group) inj-on-f':
 $\llbracket H \subseteq \text{carrier } G; a \in \text{carrier } G \rrbracket \implies \text{inj-on } (\lambda y. y \otimes \text{inv } a) (a < \# H)$
apply (rule inj-onI)
apply (subgoal-tac $x \in \text{carrier } G \ \& \ y \in \text{carrier } G$)
prefer 2 **apply** (blast intro: l-coset-subset-G [THEN subsetD])
apply (simp add: subsetD)
done

lemma (in group) inj-on-f'':
 $\llbracket H \subseteq \text{carrier } G; a \in \text{carrier } G \rrbracket \implies \text{inj-on } (\lambda y. \text{inv } a \otimes y) (a < \# H)$
apply (rule inj-onI)
apply (subgoal-tac $x \in \text{carrier } G \ \& \ y \in \text{carrier } G$)
prefer 2 **apply** (blast intro: l-coset-subset-G [THEN subsetD])
apply (simp add: subsetD)
done

lemma (in group) inj-on-g':
 $\llbracket H \subseteq \text{carrier } G; a \in \text{carrier } G \rrbracket \implies \text{inj-on } (\lambda y. a \otimes y) H$
by (force simp add: inj-on-def subsetD)

lemma (in group) l-card-cosets-equal:
 $\llbracket c \in \text{lcosets } H; H \subseteq \text{carrier } G; \text{finite}(\text{carrier } G) \rrbracket$
 $\implies \text{card } H = \text{card } c$
apply (auto simp add: LCOSETS-def)
apply (rule card-bij-eq)
apply (rule inj-on-g', assumption+)
apply (force simp add: m-assoc subsetD l-coset-def)
apply (rule inj-on-f'', assumption+)

proof
fix a x
assume $H \subseteq \text{carrier } G$ **and** $\text{finite}(\text{carrier } G)$ **and** $a \in \text{carrier } G$ **and** $c = a < \# H$
assume $a : x \in (\otimes) (\text{inv } a) \ ' (a < \# H)$
have $a \otimes \text{inv } a = \mathbf{1}$ **by** (simp add: $\langle a \in \text{carrier } G \rangle$)
from a **have** $x \in \{y. \exists x \in \bigcup h \in H. \{a \otimes h\}. y = \text{inv } a \otimes x\}$ **unfolding**
l-coset-def image-def **by** simp
then have $(\exists x' \in \bigcup h \in H. \{a \otimes h\}. x = \text{inv } a \otimes x')$ **by** blast
then show $x \in H$
using $\langle H \subseteq \text{carrier } G \rangle \langle a \in \text{carrier } G \rangle$ *is-group*
by clarify (*metis inv-solve-left m-closed subsetD*)
next
show $\bigwedge a. H \subseteq \text{carrier } G \implies \text{finite}(\text{carrier } G) \implies a \in \text{carrier } G \implies c = a < \# H \implies \text{finite } H$
using *rev-finite-subset* **by** blast

```

next
  show  $\bigwedge a. H \subseteq \text{carrier } G \implies \text{finite } (\text{carrier } G) \implies a \in \text{carrier } G \implies c$ 
   $= a < \# H \implies \text{finite } (a < \# H)$ 
  by (simp add: l-coset-subset-G rev-finite-subset)
qed

theorem (in group) l-lagrange:
   $\llbracket \text{finite}(\text{carrier } G); \text{ subgroup } H \ G \rrbracket$ 
   $\implies \text{card}(\text{lcosets } H) * \text{card}(H) = \text{order}(G)$ 
apply (simp (no-asm-simp) add: order-def lcosets-part-G [symmetric])
apply (subst mult.commute)
apply (rule card-partition)
  apply (simp add: lcosets-subset-PowG [THEN finite-subset])
  apply (simp add: lcosets-part-G)
  apply (simp add: l-card-cosets-equal subgroup.subset)
apply (simp add: lcos-disjoint)
done

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 ⇒ 'b ⇒ 'b (infixl ∘ 51)  
  assumes id-act [simp]: 1 ∘ x = x  
  and compat-act:  
    g ∈ carrier G ∧ h ∈ carrier G ⟶ g ∘ (h ∘ x) = (g ⊗ h) ∘ 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 ⇒ 'b set where  
  orbit x = {y. (∃ g ∈ carrier G. y = g ∘ x)}
```

```
definition stabiliser :: 'b ⇒ 'a set  
  where stabiliser x = {g ∈ carrier G. g ∘ 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(rule subgroupI)
```

```
  show stabiliser x ⊆ carrier G using stabiliser-def by auto
```

```
next
```

```
  fix x
```

```
  from id-act have 1 ∘ x = x by simp
```

```
  then have 1 ∈ stabiliser x using stabiliser-def by auto
```

```
  then show stabiliser x ≠ {} by auto
```

```
next
```

```
  fix g x
```

```
  assume gStab:g ∈ stabiliser x
```

```
  then have g-car:g ∈ carrier G using stabiliser-def by simp
```

```
  then have invg-car:inv g ∈ carrier G using inv-closed by simp
```

```
  have g ∘ x = x using stabiliser-def gStab by simp
```

```
  then have inv g ∘ (g ∘ x) = inv g ∘ x by simp
```

then have $(\text{inv } g \otimes g) \odot x = \text{inv } g \odot x$ **using** *compat-act g-car invg-car*
by *simp*
then have $x = (\text{inv } g) \odot x$ **using** *g-car l-inv* **by** *simp*
then show $\text{inv } g \in \text{stabiliser } x$ **using** *invg-car stabiliser-def* **by** *simp*
next
fix $g \ h \ x$
assume $g\text{-stab}: g \in \text{stabiliser } x$ **and** $h\text{-stab}: h \in \text{stabiliser } x$
then have $g\text{-car}: g \in \text{carrier } G$ **and** $h\text{-car}: h \in \text{carrier } G$ **using** *stabiliser-def* **by** *auto*
then have $g \odot x = x \ h \odot x = x$
using *stabiliser-def g-stab h-stab* **by** *auto*
then have $g \odot (h \odot x) = x$ **by** *simp*
then have $(g \otimes h) \odot x = x$ **using** *compat-act g-car h-car* **by** *simp*
then show $(g \otimes h) \in \text{stabiliser } x$
using *g-stab h-stab stabiliser-def* **by** *auto*
qed

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 $g\text{-car}: g \in \text{carrier } G$ **and**

$h\text{-car}: h \in \text{carrier } G$

shows $(g \odot x) = (h \odot x) \iff ((\text{inv } g) \otimes h) \in \text{stabiliser } x$

proof

from $g\text{-car}$ **have** $\text{invg-car}: (\text{inv } g) \in \text{carrier } G$ **by** *auto*

show $(g \odot x) = (h \odot x) \implies \text{inv } g \otimes h \in \text{stabiliser } x$

proof –

assume $gh: (g \odot x) = (h \odot x)$

have $((\text{inv } g) \otimes h) \odot x = (\text{inv } g) \odot (h \odot x)$ **using** *assms compat-act*

by *simp*

moreover have $(\text{inv } g) \odot (h \odot x) = (\text{inv } g) \odot (g \odot x)$ **using** gh **by**

simp

moreover have $(\text{inv } g) \odot (g \odot x) = ((\text{inv } g) \otimes g) \odot x$ **using** *invg-car g-car compat-act* **by** *simp*

moreover have $((\text{inv } g) \otimes g) \odot x = x$ **using** *g-car* **by** *simp*

ultimately have $((\text{inv } g) \otimes h) \odot x = x$ **by** *simp*

then show *?thesis* **using** *stabiliser-def assms* **by** *simp*

qed

show $\text{inv } g \otimes h \in \text{stabiliser } x \implies g \odot x = h \odot x$

proof –

assume $gh\text{-stab}: \text{inv } g \otimes h \in \text{stabiliser } x$

with *stabiliser-def* **have** $x = ((\text{inv } g) \otimes h) \odot x$ **by** *simp*

then have $1 \odot x = ((inv\ g) \otimes h) \odot x$ **by** *simp*
then have $((inv\ g) \otimes g) \odot x = ((inv\ g) \otimes h) \odot x$ **using** *invg-car g-car*
by *simp*
then have $x = (inv\ g) \odot (h \odot x)$ **using** *compat-act g-car h-car* **by** *simp*
then have $g \odot x = (g \otimes (inv\ g)) \odot (h \odot x)$ **using** *compat-act g-car*
invg-car **by** *metis*
then have $g \odot x = h \odot x$ **using** *compat-act g-car id-act invg-car r-inv*
by *simp*
then show *?thesis* **by** *simp*
qed
qed

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 ∈ carrier G* **and**

h-car: h ∈ carrier G

shows $(g \odot x) = (h \odot x) \iff (g <\# \text{stabiliser } x) = (h <\# \text{stabiliser } x)$

proof

show $g \odot x = h \odot x \implies g <\# \text{stabiliser } x = h <\# \text{stabiliser } x$

proof –

assume $g \odot x = h \odot x$

then have *stab-elem: ((inv g) ⊗ h) ∈ stabiliser x*

using *assms stabiliser-subgroup-corollary* **by** *simp*

with *subgroup.lcos-module-rev[OF stabiliser-subgroup]* **have** $h \in g <\#$
(stabiliser x)

using *assms is-group* **by** *simp*

with *l-repr-independence* **have** $g <\# (\text{stabiliser } x) = h <\# (\text{stabiliser } x)$

using *assms stab-elem stabiliser-subgroup* **by** *auto*

then show *?thesis* **by** *simp*

qed

show $g <\# \text{stabiliser } x = h <\# \text{stabiliser } x \implies g \odot x = h \odot x$

proof –

assume $g <\# \text{stabiliser } x = h <\# \text{stabiliser } x$

with *subgroup.lcos-module-rev[OF stabiliser-subgroup]* **have** $h \in g <\#$
(stabiliser x)

using *assms is-group l-inv stabiliser-subgroup subgroup-def* **by** *metis*

with *subgroup.lcos-module-imp[OF stabiliser-subgroup]* **have** $((inv\ g) \otimes h) \in \text{stabiliser } x$

using *assms is-group* **by** *blast*

with *stabiliser-subgroup-corollary* **have** $g \odot x = h \odot x$ **using** *assms* **by**
simp
then show *?thesis* **by** *simp*
qed
qed

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

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

lemma *quotient-rep-ex* : $H \in (\text{carrier } (G \text{ LMod } (\text{stabiliser } x))) \Longrightarrow \text{rep } H \in H$

proof –

fix *H*
assume $H : H \in \text{carrier } (G \text{ LMod } \text{stabiliser } x)$
then obtain *g* **where** $g \in \text{carrier } G \ H = g <\# (\text{stabiliser } x)$
unfolding *LFactGroup-def LCOSETS-def* **by** *auto*
then have $(\text{SOME } x. x \in H) \in H$ **using** *lcos-self stabiliser-subgroup someI-ex* **by** *fast*
then show $\text{rep } H \in H$ **using** *H rep-def* **by** *auto*
qed

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 $H : H \in \text{carrier } (G \text{ LMod } \text{stabiliser } x)$ **and**
 $gH : g \in H$
shows $H = g <\# (\text{stabiliser } x)$
proof –
fix *h*
from *H* **obtain** *h* **where** $hG : h \in \text{carrier } G$ **and** $H2 : H = h <\# (\text{stabiliser } x)$
unfolding *LFactGroup-def LCOSETS-def* **by** *auto*
with *H gH* **have** $gh : g \in h <\# (\text{stabiliser } x)$ **by** *simp*
from *l-repr-independence* **have** $h <\# \text{stabiliser } x = g <\# \text{stabiliser } x$
using *hG gh stabiliser-subgroup* **by** *simp*

with $H2$ **have** $H = g <\#$ (*stabiliser* x) **by** *simp*
then show *?thesis* **by** *simp*
qed

1.6 Orbit-Stabiliser Theorem

We can now establish the bijection between $\text{orbit}(x)$ and the quotient group $G/(\text{stabiliser}(x))$

The idea for this bijection is from [1]

theorem *orbit-stabiliser-bij*:

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

proof (*rule* *bij-betw-imageI*)

show *inj-on* ($\lambda H. \text{rep } H \odot x$) (*carrier* ($G \text{ LMod } \text{stabiliser } x$))

proof(*rule* *inj-onI*)

fix $H H'$

assume $H:H \in \text{carrier } (G \text{ LMod } (\text{stabiliser } x))$

assume $H':H' \in \text{carrier } (G \text{ LMod } (\text{stabiliser } x))$

obtain $h h'$ **where** $h:h = \text{rep } H$ **and** $h': h' = \text{rep } H'$ **by** *simp*

assume *act-equal*: $(\text{rep } H) \odot x = (\text{rep } H') \odot x$

from $H h$ *quotient-rep-ex* **have** $hH: h \in H$ **by** *simp*

from $H' h'$ *quotient-rep-ex* **have** $hH': h' \in H'$ **by** *simp*

from *subgroup.lcosets-carrier*[*OF stabiliser-subgroup is-group*] H **have**
 $H \subseteq \text{carrier } G$

unfolding *LFactGroup-def* **by** *simp*

then have $hG: h \in \text{carrier } G$ **using** hH **by** *auto*

from *subgroup.lcosets-carrier*[*OF stabiliser-subgroup is-group*] H' **have**
 $H' \subseteq \text{carrier } G$

unfolding *LFactGroup-def* **by** *simp*

then have $h'G: h' \in \text{carrier } G$ **using** hH' **by** *auto*

have *hh'-equiv*: $h <\#$ (*stabiliser* x) = $h' <\#$ (*stabiliser* x)

using $hG h'G h h'$ *act-equal stabiliser-cosets-equivalent* **by** *simp*

from *hh'-equiv* **have** $H2:H = h <\#$ (*stabiliser* x)

using $H hH$ *rep-equivalent* **by** *blast*

moreover from *hh'-equiv* **have** $H3:H' = h <\#$ (*stabiliser* x)

using $H' hH'$ *rep-equivalent* **by** *blast*

then show $H = H'$ **using** $H2 H3$ **by** *simp*

qed

next

show ($\lambda H. \text{rep } H \odot x$) ' *carrier* ($G \text{ LMod } \text{stabiliser } x$) = *orbit* x

```

proof(auto)
  show  $\bigwedge H. H \in \text{carrier } (G \text{ LMod } \text{stabiliser } x) \implies \text{rep } H \odot x \in \text{orbit } x$ 
  proof –
    fix  $H$ 
    assume  $H:H \in \text{carrier } (G \text{ LMod } (\text{stabiliser } x))$ 
    obtain  $h$  where  $h:h = \text{rep } H$  by simp
    from  $H h$  quotient-rep-ex have  $hH: h \in H$  by simp
    have  $\text{stab-sub}: (\text{stabiliser } x) \subseteq \text{carrier } G$  using stabiliser-def by auto
    from subgroup.lcosets-carrier[OF stabiliser-subgroup is-group]  $H$  have
 $H \subseteq \text{carrier } G$ 
      unfolding LFactGroup-def by simp
      with  $hH$  have  $h \in \text{carrier } G$  by auto
      then show  $(\text{rep } H) \odot x \in \text{orbit } x$  using h orbit-def mem-Collect-eq
by blast
    qed
  show  $\bigwedge y. y \in \text{orbit } x \implies y \in (\lambda H. \text{rep } H \odot x) \text{ ' carrier } (G \text{ LMod } \text{stabiliser } x)$ 
  proof –
    fix  $y$ 
    assume  $y:y \in \text{orbit } x$ 
    obtain  $g$  where  $gG:g \in \text{carrier } G$  and  $y = g \odot x$  using y orbit-def
by auto
    obtain  $H$  where  $H:H = g <\# (\text{stabiliser } x)$  by auto
    with  $gG$  have  $H\text{-carr}:H \in \text{carrier } (G \text{ LMod } \text{stabiliser } x)$ 
      unfolding LFactGroup-def LCOSETS-def by auto
      then have  $\text{rep } H \in H$  using quotient-rep-ex by auto
      then obtain  $h$  where  $h\text{-stab}:h \in \text{stabiliser } x$  and  $gh:\text{rep } H = g \otimes h$ 
        unfolding H l-coset-def by auto
        have  $hG:h \in \text{carrier } G$  using h-stab stabiliser-def by auto
        from stabiliser-def h-stab have  $h \odot x = x$  by auto
        with  $\langle y = g \odot x \rangle$  have  $y = g \odot (h \odot x)$  by simp
        then have  $y = (g \otimes h) \odot x$  using gG hG compat-act by auto
        then have  $y = (\text{rep } H) \odot x$  using gh by simp
        then show  $y \in (\lambda H. \text{rep } H \odot x) \text{ ' carrier } (G \text{ LMod } \text{stabiliser } x)$ 
          using H-carr by simp
      qed
    qed
  qed

```

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 $\text{order } G = \text{card } (\text{orbit } x) * \text{card } (\text{stabiliser } x)$

```

proof –
  have card (carrier (G LMod (stabiliser x))) = card (orbit x)
    using bij-betw-same-card orbit-stabiliser-bij by auto
  moreover have card (carrier (G LMod (stabiliser x))) * card (stabiliser x) = order G
    using finite stabiliser-subgroup l-lagrange unfolding LFactGroup-def by simp
  ultimately show ?thesis by simp
qed
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  $\Rightarrow$  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 = ( $\lambda v. (case\ v\ of\ A \Rightarrow A\ | B \Rightarrow C\ | C \Rightarrow D\ | D \Rightarrow B)$ )
definition rotate-B :: Rotation where
  rotate-B = ( $\lambda v. (case\ v\ of\ A \Rightarrow D\ | B \Rightarrow B\ | C \Rightarrow A\ | D \Rightarrow C)$ )
definition rotate-C :: Rotation where
  rotate-C = ( $\lambda v. (case\ v\ of\ A \Rightarrow B\ | B \Rightarrow D\ | C \Rightarrow C\ | D \Rightarrow A)$ )
definition rotate-D :: Rotation where
  rotate-D = ( $\lambda v. (case\ v\ of\ A \Rightarrow C\ | B \Rightarrow A\ | C \Rightarrow B\ | D \Rightarrow D)$ )

```

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: $r \in \text{simple-rotations} \implies r \in \text{complex-rotations}$ |

comp: $r \in \text{simple-rotations} \implies s \in \text{complex-rotations} \implies (r \circ s) \in \text{complex-rotations}$

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 $r:r \in \text{simple-rotations}$

shows *inj* r

using r **unfolding** *simple-rotations-def*

by *safe*

(*rule injI*; *rename-tac a b*; *case-tac a*; *case-tac b*;

simp add: simple-rotations

)+

lemma *simple-rotations-surj*:

assumes $r:r \in \text{simple-rotations}$

shows *surj* r

using r **unfolding** *simple-rotations-def*

by *safe*

(*rename-tac a*; *case-tac a*;

auto simp add: simple-rotations Vertex.split

)+

lemma *simple-rotations-bij*:

assumes $r:r \in \text{simple-rotations}$

shows *bij* r

by (*simp add: r bij-def simple-rotations-surj simple-rotations-inj*)

lemma *complex-rotations-bij*: $r \in \text{complex-rotations} \implies \text{bij } r$

proof(*induction r rule: complex-rotations.induct*)

```

  case (simp r) then show ?case using simple-rotations-bij by simp
next
  case (comp r s) then show ?case using bij-comp simple-rotations-bij by
blast
qed

```

lemma *simple-rotation-bij-corollary*: $r \in \text{simple-rotations} \implies r x \neq r y \iff x \neq y$
using *bij-def simple-rotations-bij inj-eq* **by** *metis*

lemma *rotation-bij-corollary*: $r \in \text{complex-rotations} \implies r x \neq r y \iff x \neq y$
using *bij-def complex-rotations-bij inj-eq* **by** *metis*

lemma *complex-rotations-comp*:

$r \in \text{complex-rotations} \implies s \in \text{complex-rotations} \implies (r \circ s) \in \text{complex-rotations}$

apply(*induction arbitrary: s rule: complex-rotations.induct*)

apply(*auto simp add: comp-assoc complex-rotations.comp*)

done

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

lemma *simple-rotations-fix*:

assumes $r: r \in \text{simple-rotations}$

shows $\exists v. r v = v$

using r **unfolding** *simple-rotations-def*

by (*auto simp add: simple-rotations split: Vertex.split*)

lemma *simple-rotations-fix-unique*:

assumes $r: r \in \text{simple-rotations}$

shows $r \neq \text{id} \implies r v = v \implies r w = w \implies v = w$

using r **unfolding** *simple-rotations-def*

by *safe*

(*case-tac v; case-tac w;*

auto simp add: simple-rotations

)+

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

lemma *simple-rotations-cycle*:

assumes $r: r \in \text{simple-rotations}$

shows $r \neq \text{id} \implies r v = w \implies v \neq w \implies r w \neq v$

using r **unfolding** *simple-rotations-def*

by *safe*

```

(case-tac v;
 auto simp add: simple-rotations
)+

```

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]  $\implies$  ( $\forall e. e \in \{a,b,c,d\}$ )
apply(safe)
apply(case-tac a)
apply(auto simp add: distinct-def)
apply(metis Vertex.exhaust)+
done

```

```

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

```

proof –

```

  assume r:r  $\in$  complex-rotations and dist:distinct [a,b,c,d] and notf: $\forall e \in \{a,b,c\}. r e \neq f$ 

```

```

  then have 1:( $\forall v. v \in \{a,b,c,d\}$ ) using distinct-vertices by simp

```

```

  from complex-rotations-bij have  $\exists v. r v = f$  using r bij-pointE by metis

```

```

  then have  $\exists v \in \{a,b,c,d\}. r v = f$  using 1 by blast

```

```

  then show r d = f using notf by simp

```

qed

```

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

```

proof –

```

  assume r:r  $\in$  complex-rotations and dist:distinct [a,b,c,d] and notf: $\forall e \in \{a,b,c\}. r f \neq e$ 

```

```

  then have 1:( $\forall v. v \in \{a,b,c,d\}$ ) using distinct-vertices by simp

```

```

  have  $\exists v. r f = v$  by simp

```

```

  then have  $\exists v \in \{a,b,c,d\}. r f = v$  using 1 by blast

```

```

  then show r f = d using notf by simp

```

qed

```

lemma cycle-map: r  $\in$  complex-rotations  $\implies$  distinct[a,b,c,d]  $\implies$ 

```

```

  r a = b  $\implies$  r b = a  $\implies$  r c = d  $\implies$  r d = c  $\implies$   $\forall v w. r v = w \implies r w = v$ 

```

```

  using distinct-map' rotation-bij-corollary by fastforce

```

```

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

```

```

  using complex-rotations.simp distinct-map by simp

```


lemma *simple-distinct-map'*: $r \in \text{simple-rotations} \implies \text{distinct}[a,b,c,d] \implies$
 $(\forall e \in \{a,b,c\}. r f \neq e) \implies r f = d$
using *complex-rotations.simp distinct-map'* **by** *simp*

lemma *simple-distinct-ident*: $r \in \text{simple-rotations} \implies \text{distinct}[a,b,c,d] \implies$
 $(\forall e \in \{a,b,c\}. r e \neq e) \implies r d = d$
using *simple-rotations-fix simple-distinct-map'* **by** *metis*

lemma *id-decomp*:

assumes *distinct:distinct* $[(a::\text{Vertex}),b,c,d]$ **and** *ident*: $(\forall x \in \{a,b,c,d\}. r x = x)$

shows $r = \text{id}$

proof –

from *distinct-vertices* **have** $\forall e. e \in \text{set } [a,b,c,d]$ **using** *distinct* **by** *simp*
then have $\forall e. r e = e$ **using** *ident* **by** *auto*

then show $r = \text{id}$ **by** *auto*

qed

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*:

$r \in \text{complex-rotations} \implies ((\forall v. r v \neq v) \longrightarrow r v = w \longrightarrow r w = v) \wedge$
 $(r v = v \longrightarrow r w = w \longrightarrow v \neq w \longrightarrow r = \text{id})$

proof(*induction r arbitrary: v w rule: complex-rotations.induct*)

case (*simp r*)

assume $r:r \in \text{simple-rotations}$

show *?case*

proof

have $\exists v. r v = v$ **using** *simple-rotations-fix r* **by** *simp*

then have $\neg (\forall v. r v \neq v)$ **by** *simp*

then show $(\forall v. r v \neq v) \longrightarrow r v = w \longrightarrow r w = v$ **by** *blast*

show $r v = v \longrightarrow r w = w \longrightarrow v \neq w \longrightarrow r = \text{id}$ **using** *simple-rotations-fix-unique simp* **by** *blast*

qed

next

case (*comp r s*)

assume $r:r \in \text{simple-rotations}$

assume $s:s \in \text{complex-rotations}$

have *fix-unique*: $\forall v w. s v = v \longrightarrow s w = w \longrightarrow v \neq w \longrightarrow s = \text{id}$ **using**

comp **by** *blast*
show *?case*
proof
show $(\forall x. (r \circ s) x \neq x) \longrightarrow (r \circ s) v = w \longrightarrow (r \circ s) w = v$
proof (*rule impI*)
assume *nofixs*: $\forall x. (r \circ s) x \neq x$
assume $(r \circ s) v = w$
show $(r \circ s) w = v$
proof (*cases* $\forall x. s x \neq x$)
assume *nofixs*: $\forall x. s x \neq x$
then have *cycle*: $\forall x y. (s x = y \longrightarrow s y = x)$ **using** *comp.IH* **by**
blast
then show *?thesis*
proof (*cases* $r = id$)
assume *id*: $r = id$
then have $s v = w$ **using** $\langle (r \circ s) v = w \rangle$ **by** *simp*
then have $s w = v$ **using** *cycle* **by** *blast*
then show $(r \circ s) w = v$ **using** *id* **by** *simp*
next
assume *notid*: $r \neq id$
obtain *a* **where** $s v = a$ **and** $s a = v$ **and** $a \neq v$ **using** *comp.IH*
nofixs **by** *blast*
obtain *b* **where** $s w = b$ **and** $s b = w$ **and** $b \neq w$ **using** *comp.IH*
nofixs **by** *blast*
have $v \neq w$ **using** $\langle (r \circ s) v = w \rangle$ *nofixs* **by** *blast*
then have $a \neq b$ **using** *comp.hyps* *rotation-bij-corollary* $\langle s a = v \rangle$
 $\langle s b = w \rangle$ **by** *auto*
have $r a = w$ **using** $\langle s v = a \rangle$ $\langle (r \circ s) v = w \rangle$ **by** *auto*
then show *?thesis*
proof (*cases* $a = w$)
assume $a = w$
then have $r a = a$ **using** $\langle r a = w \rangle$ **by** *simp*
then have $s v = w$ **using** $\langle a = w \rangle$ $\langle s v = a \rangle$ **by** *simp*
then have $b = v$ **using** $\langle s b = w \rangle$ *rotation-bij-corollary* *comp.hyps*
by *blast*
then have $s w = v$ **using** $\langle s w = b \rangle$ **by** *simp*
then have $r v \neq v$ **using** *simple-rotations-fix-unique* *notid* $\langle r a = a \rangle$ $\langle a \neq v \rangle$
comp.hyps(1) **by** *auto*
obtain *c d* **where** $s c = d$ **and** $c \neq v$ **and** $c \neq w$
using $\langle a \neq v \rangle$ $\langle r a = w \rangle$ $\langle r v \neq v \rangle$ *comp.hyps*(1) *simple-rotation-bij-corollary* **by** *blast*
then have $d \neq v$ **and** $d \neq w$
using $\langle s w = v \rangle$ $\langle c \neq v \rangle$ $\langle s c = d \rangle$ $\langle s v = w \rangle$ *comp.hyps*(2)

rotation-bij-corollary **by** *auto*
then have $s\ d = c$ **using** $\langle s\ c = d \rangle$ *comp.IH nofixs* **by** *blast*
then have $c \neq d$ **using** *nofixs* **by** *auto*
then show *?thesis*
proof(*cases* $r\ v = c$)
assume $r\ v = c$
then have $r\ c \neq v$ **using** $\langle c \neq v \rangle$ *simple-rotations-cycle*
comp.hyps(1) **notid** **by** *simp*
have $r\ c \neq w$
using $\langle r\ a = a \rangle \langle c \neq w \rangle \langle r\ a = w \rangle$ *simple-rotation-bij-corollary*
comp.hyps(1) **by** *auto*
have $r\ c \neq c$ **using** $\langle a = w \rangle \langle c \neq w \rangle \langle r\ a = a \rangle$
comp.hyps(1) *simple-rotations-fix-unique* **notid** **by** *blast*
have *dist:distinct* $[v,w,c,d]$ **using** $\langle c \neq v \rangle \langle c \neq w \rangle \langle c \neq d \rangle \langle d$
 $\neq v \rangle \langle d \neq w \rangle \langle v \neq w \rangle$ **by** *simp*
then have $\forall v \in \{v,w,c\}. r\ c \neq v$ **using** $\langle r\ c \neq c \rangle \langle r\ c \neq v \rangle \langle r$
 $c \neq w \rangle$ **by** *auto*
then have $r\ c = d$ **using** *simple-distinct-map'* *comp.hyps(1)*
dist **by** *auto*
then have $(r \circ s)\ d = d$ **using** $\langle s\ d = c \rangle$ **by** *simp*
then have *False* **using** *nofixs* **by** *blast*
then show *?thesis* **by** *simp*
next
assume $r\ v \neq c$
then have $r\ v \neq w$
using $\langle r\ a = a \rangle \langle v \neq w \rangle \langle r\ a = w \rangle$ *simple-rotation-bij-corollary*
comp.hyps(1) **by** *auto*
then have $r\ v \neq v$ **using** $\langle a = w \rangle \langle r\ a = a \rangle$
comp.hyps(1) *simple-rotations-fix-unique* **notid** **by** *blast*
have *dist:distinct* $[w,c,v,d]$ **using** $\langle c \neq v \rangle \langle c \neq w \rangle \langle c \neq d \rangle \langle d$
 $\neq v \rangle \langle d \neq w \rangle \langle v \neq w \rangle$ **by** *simp*
then have $\forall x \in \{w,c,v\}. r\ v \neq x$ **using** $\langle r\ v \neq c \rangle \langle r\ v \neq v \rangle$
 $\langle r\ v \neq w \rangle$ **by** *auto*
then have $r\ v = d$ **using** *simple-distinct-map'* *comp.hyps(1)*
dist **by** *auto*
then have $r\ d \neq v$ **using** $\langle d \neq v \rangle$ *simple-rotations-cycle*
comp.hyps(1) **notid** **by** *simp*
have $r\ d \neq w$
using $\langle r\ a = a \rangle \langle d \neq w \rangle \langle r\ a = w \rangle$ *simple-rotation-bij-corollary*
comp.hyps(1) **by** *auto*
have $r\ d \neq d$ **using** $\langle a = w \rangle \langle d \neq w \rangle \langle r\ a = a \rangle$
comp.hyps(1) *simple-rotations-fix-unique* **notid** **by** *blast*
have *dist':distinct* $[w,v,d,c]$ **using** $\langle c \neq v \rangle \langle c \neq w \rangle \langle c \neq d \rangle \langle d$
 $\neq v \rangle \langle d \neq w \rangle \langle v \neq w \rangle$ **by** *simp*

then have $\forall v \in \{w, v, d\}. r d \neq v$ **using** $\langle r d \neq d \rangle \langle r d \neq w \rangle$
 $\langle r d \neq v \rangle$ **by auto**
then have $r d = c$ **using** *simple-distinct-map'* *comp.hyps(1)*
dist' **by auto**
then have $(r \circ s) c = c$ **using** $\langle s c = d \rangle$ **by simp**
then have *False* **using** *nofixrs* **by blast**
then show *?thesis* **by simp**
qed
next
assume $a \neq w$
then have $r a \neq a$ **using** $\langle r a = w \rangle$ **by simp**
have $b \neq v$ **using** $\langle a \neq w \rangle \langle s b = w \rangle \langle s v = a \rangle$ **by auto**
have $r w \neq w$ **using** $\langle a \neq w \rangle \langle r a = w \rangle$ *comp.hyps(1)* *simple-rotation-bij-corollary* **by auto**
from *nofixrs* **have** $s w \neq w$ **by simp**
then have $r v \neq w$ **using** $\langle a \neq v \rangle \langle r a = w \rangle$ *comp.hyps*
simple-rotation-bij-corollary **by blast**
have $s v \neq w$ **using** $\langle r a = w \rangle \langle r a \neq a \rangle \langle s v = a \rangle$ **by blast**
then show *?thesis*
proof (*cases* $r b = b$)
assume $r b = b$
then have $r b \neq a$ **using** $\langle a \neq b \rangle$ **by simp**
have $r w \neq a$ **using** $\langle r a = w \rangle \langle r w \neq w \rangle$ *comp.hyps(1)* *notid*
simple-rotations-cycle **by blast**
have *dist:distinct* $[a, b, w, v]$ **using** $\langle a \neq w \rangle \langle a \neq b \rangle \langle a \neq v \rangle \langle b$
 $\neq w \rangle \langle b \neq v \rangle \langle v \neq w \rangle$ **by simp**
then have $\forall x \in \{a, b, w\}. r x \neq a$ **using** $\langle r a \neq a \rangle \langle r b \neq a \rangle$
 $\langle r w \neq a \rangle$ **by auto**
then have $r v = a$ **using** *simple-distinct-map* *comp.hyps(1)* *dist*
by auto
then show *?thesis* **using** $\langle s a = v \rangle$ *nofixrs* *comp-apply* **bymetis**
next
assume $r b \neq b$
have *dist:distinct* $[w, a, b, v]$ **using** $\langle a \neq w \rangle \langle a \neq b \rangle \langle a \neq v \rangle \langle b$
 $\neq w \rangle \langle b \neq v \rangle \langle v \neq w \rangle$ **by simp**
then have $\forall x \in \{w, a, b\}. r x \neq x$ **using** $\langle r w \neq w \rangle \langle r a \neq a \rangle$
 $\langle r b \neq b \rangle$ **by auto**
then have $r v = v$ **using** *simple-distinct-ident* *comp.hyps(1)*
dist **by auto**
have $r w \neq a$ **using** $\langle a \neq w \rangle$ *simple-rotations-cycle* *comp.hyps(1)*
notid $\langle r a = w \rangle$ **by simp**
have $r w \neq v$ **using** $\langle r v = v \rangle \langle v \neq w \rangle$ *comp.hyps(1)*
simple-rotation-bij-corollary **by blast**
have *dist':distinct* $[a, v, w, b]$ **using** $\langle a \neq w \rangle \langle a \neq b \rangle \langle a \neq v \rangle \langle b$

$\neq w \rangle \langle b \neq v \rangle \langle v \neq w \rangle$ **by** *simp*
then have $\forall x \in \{a, v, w\}. r w \neq x$ **using** $\langle r w \neq a \rangle \langle r w \neq v \rangle$
 $\langle r w \neq w \rangle$ **by** *auto*
then have $r w = b$ **using** *simple-distinct-map' comp.hyps(1)*
dist' **by** *auto*
then show *?thesis* **using** $\langle s b = w \rangle$ *nofixrs comp-apply* **by** *metis*
qed
qed
qed
next
assume $\neg (\forall v. s v \neq v)$
then have *fix1*: $\exists v. s v = v$ **by** *blast*
then obtain a **where** $a: s a = a$ **by** *blast*
then show *?thesis*
proof (*cases* $r = id$)
assume $id: r = id$
then have $(r \circ s) a = a$ **using** a **by** *simp*
then have *False* **using** *nofixrs* **by** *auto*
then show *?thesis* **by** *simp*
next
assume *notid*: $r \neq id$
then have *fix1*: $\exists v. r v = v$ **using** *simple-rotations-fix comp.hyps*
by *simp*
then obtain b **where** $b: r b = b$ **by** *blast*
then show *?thesis*
proof (*cases* $a = b$)
assume $a = b$
then have $(r \circ s) a = a$ **using** $a b$ **by** *simp*
then have *False* **using** *nofixrs* **by** *blast*
then show *?thesis* **by** *simp*
next
assume $a \neq b$
have $r a \neq a$ **using** $\langle a \neq b \rangle$ *b comp.hyps(1) notid simple-rotations-fix-unique*
by *blast*
have $r a \neq b$ **using** $\langle a \neq b \rangle$ *b comp.hyps(1) simple-rotation-bij-corollary*
by *auto*
then obtain c **where** $r a = c$ **and** $a \neq c$ **and** $b \neq c$ **using** $\langle r a$
 $\neq a \rangle$ **by** *auto*
have $s b \neq a$ **using** $\langle a \neq b \rangle$ *a comp.hyps(2) rotation-bij-corollary*
by *blast*
have $s b \neq b$ **using** b *nofixrs comp-apply* **by** *metis*
then obtain d **where** $s b = d$ **and** $a \neq d$ **and** $b \neq d$ **using** $\langle s b$
 $\neq a \rangle$ **by** *auto*
have $r c \neq a$ **using** *simple-rotations-cycle* $\langle a \neq c \rangle \langle r a = c \rangle$

comp.hyps(1) notid by blast
have $r\ c \neq b$ **using** $\langle b \neq c \rangle$ *b comp.hyps(1) simple-rotation-bij-corollary*
by blast
have $r\ c \neq c$ **using** $\langle b \neq c \rangle$ *b comp.hyps(1) notid simple-rotations-fix-unique*
by blast
then show *?thesis*
proof (*cases c = d*)
assume $c = d$
then have $s\ c \neq c$ **using** $\langle b \neq c \rangle$ $\langle s\ b = d \rangle$ *rotation-bij-corollary*
s by auto
obtain e **where** $r\ c = e$ **and** $a \neq e$ **and** $b \neq e$ **and** $c \neq e$ **and**
 $d \neq e$
using $\langle r\ c \neq a \rangle$ $\langle r\ c \neq b \rangle$ $\langle r\ c \neq c \rangle$ $\langle c = d \rangle$ **by auto**
have $r\ e \neq b$ **using** $\langle b \neq e \rangle$ *b r simple-rotation-bij-corollary* **by**
blast
have $r\ e \neq c$ **using** $\langle a \neq e \rangle$ $\langle r\ a = c \rangle$ *r simple-rotation-bij-corollary*
by blast
have $r\ e \neq e$ **using** $\langle b \neq e \rangle$ *b notid r simple-rotations-fix-unique*
by blast
then have *dist:distinct* $[b,c,e,a]$ **using** $\langle a \neq b \rangle$ $\langle a \neq c \rangle$ $\langle a \neq e \rangle$
 $\langle b \neq c \rangle$ $\langle b \neq e \rangle$ $\langle c \neq e \rangle$ **by simp**
then have $\forall x \in \{b,c,e\}. r\ e \neq x$ **using** $\langle r\ e \neq b \rangle$ $\langle r\ e \neq c \rangle$ $\langle r\ e$
 $\neq e \rangle$ **by auto**
then have $r\ e = a$ **using** *simple-distinct-map'* *comp.hyps(1) dist*
by auto
have *dist:distinct* $[a,b,c,e]$ **using** $\langle a \neq b \rangle$ $\langle a \neq c \rangle$ $\langle a \neq e \rangle$ $\langle b \neq$
 $c \rangle$ $\langle b \neq e \rangle$ $\langle c \neq e \rangle$ **by simp**
then have $\forall x \in \{a,b,c\}. r\ c \neq x$ **using** $\langle r\ c \neq a \rangle$ $\langle r\ c \neq b \rangle$ $\langle r\ c$
 $\neq c \rangle$ **by auto**
then have $r\ c = e$ **using** *simple-distinct-map'* *comp.hyps(1) dist*
by auto
have $s\ e \neq a$ **using** $\langle a \neq e \rangle$ *a rotation-bij-corollary s* **by blast**
have $s\ e \neq c$ **using** $\langle b \neq e \rangle$ $\langle c = d \rangle$ $\langle s\ b = d \rangle$ *rotation-bij-corollary*
s by blast
have $s\ e \neq e$ **using** $\langle a \neq e \rangle$ $\langle s\ b \neq b \rangle$ *a fix-unique* **by fastforce**
then have *dist:distinct* $[a,c,e,b]$ **using** $\langle a \neq b \rangle$ $\langle a \neq c \rangle$ $\langle a \neq e \rangle$
 $\langle b \neq c \rangle$ $\langle b \neq e \rangle$ $\langle c \neq e \rangle$ **by simp**
then have $\forall x \in \{a,c,e\}. s\ e \neq x$ **using** $\langle s\ e \neq a \rangle$ $\langle s\ e \neq c \rangle$ $\langle s\ e$
 $\neq e \rangle$ **by auto**
then have $s\ e = b$ **using** *distinct-map'* *comp.hyps(2) dist* **by auto**
have $s\ c \neq a$ **using** $\langle a \neq c \rangle$ *a rotation-bij-corollary s* **by blast**
have $s\ c \neq b$ **using** $\langle c \neq e \rangle$ $\langle s\ e = b \rangle$ *rotation-bij-corollary s* **by**
blast

then have $dist:distinct [a,b,c,e]$ **using** $\langle a \neq b \rangle \langle a \neq c \rangle \langle a \neq e \rangle$
 $\langle b \neq c \rangle \langle b \neq e \rangle \langle c \neq e \rangle$ **by** *simp*
then have $\forall x \in \{a,b,c\}. s\ c \neq x$ **using** $\langle s\ c \neq a \rangle \langle s\ c \neq b \rangle \langle s\ c$
 $\neq c \rangle$ **by** *auto*
then have $s\ c = e$ **using** *distinct-map'* *comp.hyps(2)* *dist* **by**
auto

have $rsa:(r \circ s)\ a = c$ **using** $\langle r\ a = c \rangle$ *a* **by** *simp*
have $rsb:(r \circ s)\ b = e$ **using** $\langle c = d \rangle \langle r\ c = e \rangle \langle s\ b = d \rangle$ **by** *auto*

have $rsc:(r \circ s)\ c = a$ **using** $\langle r\ e = a \rangle \langle s\ c = e \rangle$ **by** *auto*
have $rse:(r \circ s)\ e = b$ **using** $\langle s\ e = b \rangle$ *b* **by** *simp*
then have $dist:distinct [a,c,b,e]$ **using** $\langle a \neq b \rangle \langle a \neq c \rangle \langle a \neq e \rangle$
 $\langle b \neq c \rangle \langle b \neq e \rangle \langle c \neq e \rangle$ **by** *simp*
have $comprs:r \circ s \in complex-rotations$ **using** *complex-rotations.comp*
r s **by** *simp*
show *?thesis* **using** *cycle-map[OF comprs dist rsa rsc rsb rse]* $\langle (r$
 $\circ s)\ v = w \rangle$ **by** *blast*
next
assume $c \neq d$
then have $dist:distinct [a,b,c,d]$ **using** $\langle a \neq b \rangle \langle a \neq c \rangle \langle a \neq d \rangle$
 $\langle b \neq c \rangle \langle b \neq d \rangle \langle c \neq d \rangle$ **by** *simp*
then have $\forall x \in \{a,b,c\}. r\ c \neq x$ **using** $\langle r\ c \neq a \rangle \langle r\ c \neq b \rangle \langle r\ c$
 $\neq c \rangle$ **by** *auto*
then have $r\ c = d$ **using** *simple-distinct-map'* *comp.hyps(1)* *dist*
by *auto*
have $r\ d \neq b$ **using** $\langle b \neq d \rangle$ *b* *comp.hyps(1)* *simple-rotation-bij-corollary*
by *blast*
have $r\ d \neq c$ **using** $\langle c \neq d \rangle \langle r\ c = d \rangle$ *comp.hyps(1)* *notid*
simple-rotations-cycle **by** *blast*
have $r\ d \neq d$ **using** $\langle c \neq d \rangle \langle r\ c = d \rangle$ *comp.hyps(1)* *sim-*
ple-rotation-bij-corollary **by** *auto*
have $dist:distinct [b,c,d,a]$ **using** $\langle a \neq b \rangle \langle a \neq c \rangle \langle a \neq d \rangle \langle b \neq$
 $c \rangle \langle b \neq d \rangle \langle c \neq d \rangle$ **by** *simp*
then have $\forall x \in \{b,c,d\}. r\ d \neq x$ **using** $\langle r\ d \neq b \rangle \langle r\ d \neq c \rangle \langle r$
 $d \neq d \rangle$ **by** *auto*
then have $r\ d = a$ **using** *simple-distinct-map'* *comp.hyps(1)* *dist*
by *auto*
have $s\ d \neq a$ **using** $\langle a \neq d \rangle$ *a* *comp.hyps(2)* *rotation-bij-corollary*
by *blast*
have $s\ d \neq c$ **using** *nofixrs* $\langle r\ c = d \rangle \langle c \neq d \rangle$ *comp-apply* **by**
metis
have $s\ d \neq d$ **using** $\langle b \neq d \rangle \langle s\ b = d \rangle$ *comp.hyps(2)* *rota-*
tion-bij-corollary **by** *auto*

```

      have dist:distinct [a,c,d,b] using ⟨a ≠ b⟩ ⟨a ≠ c⟩ ⟨a ≠ d⟩ ⟨b ≠
c⟩ ⟨b ≠ d⟩ ⟨c ≠ d⟩ by simp
      then have  $\forall x \in \{a,c,d\}. s\ d \neq x$  using ⟨s d ≠ a⟩ ⟨s d ≠ c⟩ ⟨s
d ≠ d⟩ by auto
      then have  $s\ d = b$  using distinct-map' comp.hyps(2) dist by
auto
      have  $s\ c \neq a$  using ⟨a ≠ c⟩ a comp.hyps(2) rotation-bij-corollary
by blast
      have  $s\ c \neq b$  using ⟨c ≠ d⟩ ⟨s d = b⟩ comp.hyps(2) rotation-bij-corollary by blast
      have  $s\ c \neq d$  using ⟨b ≠ c⟩ ⟨s b = d⟩ comp.hyps(2) rotation-bij-corollary by blast
      have dist:distinct [a,b,d,c] using ⟨a ≠ b⟩ ⟨a ≠ c⟩ ⟨a ≠ d⟩ ⟨b ≠
c⟩ ⟨b ≠ d⟩ ⟨c ≠ d⟩ by simp
      then have  $\forall x \in \{a,b,d\}. s\ c \neq x$  using ⟨s c ≠ a⟩ ⟨s c ≠ b⟩ ⟨s
c ≠ d⟩ by auto
      then have  $s\ c = c$  using distinct-map' comp.hyps(2) dist by
auto
      then have False using fix-unique ⟨s d ≠ d⟩ ⟨a ≠ c⟩ a by fastforce

      then show ?thesis by simp
      qed
    qed
  qed
  qed
  qed
next
show  $(r \circ s)\ v = v \longrightarrow (r \circ s)\ w = w \longrightarrow v \neq w \longrightarrow r \circ s = id$ 
proof(rule impI)+
  assume rsv:( $r \circ s$ )  $v = v$  and rsw:( $r \circ s$ )  $w = w$  and  $v \neq w$ 
  show  $r \circ s = id$ 
  proof(cases s = id)
    assume sid: $s = id$ 
    then have  $s\ v = v$  and  $s\ w = w$  by auto
    then have  $r = id$  using simple-rotations-fix-unique rsv rsw ⟨ $v \neq w$ ⟩
  r by auto
  with sid show ?thesis by simp
next
assume snotid: $s \neq id$ 
then show ?thesis
proof(cases r = id)
  assume rid: $r = id$ 
  then have  $s\ v = v$  and  $s\ w = w$  using rsv rsw by auto
  then have  $s = id$  using ⟨ $v \neq w$ ⟩ fix-unique by blast

```


with *rid* **show** *?thesis* **by** *simp*
next
assume *rnotid*: $r \neq id$
from *simple-rotations-fix-unique*[*OF comp.hyps(1) rnotid*] **have**
r-fix-forall: $\forall v w. r v = v \wedge r w = w \longrightarrow v = w$ **by** *blast*
from *comp.IH snotid* **have**
s-fix-forall: $\forall v w. s v = v \wedge s w = w \longrightarrow v = w$ **by** *blast*
have *fixes-two*: $\exists a b. (r \circ s) a = a \wedge (r \circ s) b = b \wedge a \neq b$ **using**
 $\langle v \neq w \rangle$ *rsv rsw* **by** *blast*
then show *?thesis*
proof (*cases* $\forall x. s x \neq x$)
assume *sfix'*: $\forall x. s x \neq x$
from *simple-rotations-fix* **obtain** *a* **where** $a:r a = a$ **using** *r* **by**
blast
from *sfix'* **have** $s a \neq a$ **by** *blast*
then have $(r \circ s) a \neq a$ **using** *a simple-rotation-bij-corollary r* **by**
fastforce
with *fixes-two* **obtain** *b* **where** $(r \circ s) b = b$ **and** $b \neq a$ **by** *blast*
with *fixes-two* **obtain** *c* **where** $(r \circ s) c = c$ **and** $c \neq a$ **and** $c \neq$
b
using $\langle (r \circ s) a \neq a \rangle$ **by** *blast*

have $s b \neq a$ **using** *a* $\langle (r \circ s) b = b \rangle$ *sfix'* **by** *force*
have $s c \neq a$ **using** *a* $\langle (r \circ s) c = c \rangle$ *sfix'* **by** *force*

then obtain *d* **where** $s d = a$ **and** $d \neq a$ **and** $d \neq b$ **and** $d \neq c$
using $\langle s a \neq a \rangle \langle s b \neq a \rangle \langle s c \neq a \rangle$ *complex-rotations-bij s*
bij-pointE **by** *metis*
have $(r \circ s) d = a$ **using** *a* $\langle s d = a \rangle$ **by** *simp*

have $r b \neq a$ **using** *a r simple-rotation-bij-corollary* $\langle b \neq a \rangle$ **by**
auto
have $r c \neq a$ **using** *a r simple-rotation-bij-corollary* $\langle c \neq a \rangle$ **by**
auto
have $r d \neq a$ **using** *a r simple-rotation-bij-corollary* $\langle d \neq a \rangle$ **by**
auto
have $r b \neq b$ **using** *a r simple-rotations-fix-unique rnotid* $\langle b \neq a \rangle$
by *blast*
have $r c \neq c$ **using** *a r simple-rotations-fix-unique rnotid* $\langle c \neq a \rangle$
by *blast*
have $r d \neq d$ **using** *a r simple-rotations-fix-unique rnotid* $\langle d \neq a \rangle$
by *blast*

then have *False* **using** *sfix'*

proof (*cases* $r\ b = c$)
assume $r\ b = c$
then have $r\ c \neq c$ **using** r *simple-rotation-bij-corollary* $\langle c \neq b \rangle$
by auto
then have $r\ c \neq b$ **using** r *rnotid simple-rotations-cycle* $\langle r\ b = c \rangle$ **by auto**
have $dist:distinct\ [a,b,c,d]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle d \neq b \rangle \langle c \neq b \rangle \langle b \neq a \rangle$ **by simp**
then have $\forall v \in \{a,b,c\}. r\ c \neq v$ **using** $\langle r\ c \neq c \rangle \langle r\ c \neq a \rangle \langle r\ c \neq b \rangle$ **by auto**
then have $r\ c = d$ **using** *simple-distinct-map'* r $dist$ **by auto**

have $r\ d \neq c$ **using** r *simple-rotation-bij-corollary* $\langle d \neq b \rangle \langle r\ b = c \rangle$ **by auto**
have $r\ d \neq d$ **using** $r\ a \langle d \neq a \rangle \langle r\ d \neq d \rangle$ **by simp**
have $dist':distinct\ [a,c,d,b]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle d \neq b \rangle \langle c \neq b \rangle \langle b \neq a \rangle$ **by simp**
then have $\forall v \in \{a,c,d\}. r\ d \neq v$ **using** $\langle r\ d \neq c \rangle \langle r\ d \neq a \rangle \langle r\ d \neq d \rangle$ **by auto**
then have $r\ d = b$ **using** *simple-distinct-map'* r $dist'$ **by auto**

then have $s\ b = d$ **using** $\langle (r \circ s)\ b = b \rangle$ r *simple-rotation-bij-corollary* **by auto**
have $s\ c = b$ **using** $\langle (r \circ s)\ c = c \rangle \langle r\ b = c \rangle$ r *simple-rotation-bij-corollary* **by auto**

then have $s\ b \neq c$ **using** $\langle s\ b = d \rangle \langle d \neq c \rangle$ **by simp**
then show *False* **using** $s\ sfix' \langle s\ c = b \rangle$ *comp(3)* **by blast**
next
assume $r\ b \neq c$
have $dist':distinct\ [a,b,c,d]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle d \neq b \rangle \langle c \neq b \rangle \langle b \neq a \rangle$ **by simp**
then have $\forall v \in \{a,b,c\}. r\ b \neq v$ **using** $\langle r\ b \neq a \rangle \langle r\ b \neq b \rangle \langle r\ b \neq c \rangle$ **by auto**
then have $r\ b = d$ **using** *simple-distinct-map'* r $dist'$ **by auto**

then have $r\ c \neq d$ **using** r *simple-rotation-bij-corollary* $\langle c \neq b \rangle$ **by auto**
have $dist'':distinct\ [a,c,d,b]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle d \neq b \rangle \langle c \neq b \rangle \langle b \neq a \rangle$ **by simp**
then have $\forall v \in \{a,c,d\}. r\ c \neq v$ **using** $\langle r\ c \neq a \rangle \langle r\ c \neq c \rangle \langle r\ c \neq d \rangle$ **by auto**
then have $r\ c = b$ **using** *simple-distinct-map'* r $dist''$ **by auto**

then have $r d \neq b$ **using** r *simple-rotation-bij-corollary* $\langle d \neq c \rangle$
by *auto*
have $dist''' : distinct [a, b, d, c]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle d \neq b \rangle \langle c \neq b \rangle \langle b \neq a \rangle$ **by** *simp*
then have $\forall v \in \{a, b, d\}. r d \neq v$ **using** $\langle r d \neq a \rangle \langle r d \neq b \rangle \langle r d \neq d \rangle$ **by** *auto*
then have $r d = c$ **using** *simple-distinct-map'* r $dist'''$ **by** *auto*

then have $s b = c$ **using** $\langle r c = b \rangle \langle (r \circ s) b = b \rangle$ r *simple-rotation-bij-corollary* **by** *auto*
have $s c = d$ **using** $\langle (r \circ s) c = c \rangle \langle r d = c \rangle$ r *simple-rotation-bij-corollary* **by** *auto*

then have $s c \neq b$ **using** $\langle d \neq b \rangle$ **by** *simp*
then have *False* **using** $comp(\mathcal{B})$ s $sfix'$ $\langle s b = c \rangle$ **by** *blast*
then show *?thesis* **by** *simp*
qed
then show *?thesis* **by** *simp*
next
assume $\neg (\forall x. s x \neq x)$
then have $\exists x. s x = x$ **by** *simp*
then obtain a **where** $a : s a = a$ **by** *blast*
from *simple-rotations-fix* **obtain** b **where** $b : r b = b$ **using** r **by**
blast
then show *?thesis*
proof (*cases* $a = b$)
assume $a \neq b$
with $a b$ **have** $r a \neq a$ **using** r *rnotid* *simple-rotations-fix-unique*
by *blast*
then have $(r \circ s) a \neq a$ **using** a **by** *simp*
have $s b \neq b$ **using** a $\langle a \neq b \rangle$ *s-fix-forall* **by** *blast*
then have $(r \circ s) b \neq b$ **using** b *simple-rotations-inj* r
complex-rotations.simp *rotation-bij-corollary* **by** *fastforce*
with *fixes-two* **obtain** c **where** $(r \circ s) c = c$ **and** $c \neq a$ **and** $c \neq b$ **using** $\langle (r \circ s) a \neq a \rangle$ **by** *blast*
from *fixes-two* **obtain** d **where** $(r \circ s) d = d$ **and** $d \neq a$ **and** $d \neq b$ **and** $d \neq c$
using $\langle (r \circ s) a \neq a \rangle \langle (r \circ s) b \neq b \rangle$ **by** *blast*

have $s c \neq a$ **using** a $\langle c \neq a \rangle$ *rotation-bij-corollary* s **by** *force*
have $s d \neq a$ **using** a $\langle d \neq a \rangle$ *rotation-bij-corollary* s **by** *force*

have $r a \neq c$ **using** $\langle s c \neq a \rangle \langle (r \circ s) c = c \rangle \langle c \neq a \rangle$ r
simple-rotation-bij-corollary **by** *auto*

have $r a \neq d$ **using** $\langle s d \neq a \rangle \langle (r \circ s) d = d \rangle \langle d \neq a \rangle r$
simple-rotation-bij-corollary **by** *auto*
have $r a \neq b$ **using** b *simple-rotation-bij-corollary* $\langle a \neq b \rangle r$ **by**
auto
have $dist:distinct [b,c,d,a]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle c \neq b \rangle \langle a \neq$
 $b \rangle \langle d \neq c \rangle \langle d \neq b \rangle$ **by** *simp*
then have $\forall v \in \{b,c,d\}. r a \neq v$ **using** $\langle r a \neq b \rangle \langle r a \neq c \rangle \langle r$
 $a \neq d \rangle$ **by** *auto*
then have $r a = a$ **using** *simple-distinct-map'* $r dist$ **by** *simp*
then have *False* **using** $\langle r a \neq a \rangle$ **by** *simp*
then show *?thesis* **by** *simp*
next
assume $a = b$
with $a b$ **have** $(r \circ s) a = a$ **by** *simp*
from *fixes-two* **obtain** c **where** $rsc:(r \circ s) c = c$ **and** $c \neq a$ **by**
blast
then have $r c \neq c$ **using** $b \langle a = b \rangle r$ *rnotid* *simple-rotations-fix-unique*
by *blast*
then have $s c \neq c$ **using** rsc **by** *auto*
then obtain d **where** $s c = d$ **and** $d \neq c$ **by** *blast*
then have $d \neq a$ **using** $a s$ *rotation-bij-corollary* **by** *blast*
have $s d \neq d$ **using** a **using** $\langle d \neq a \rangle s$ *fix-forall* **by** *blast*
have $r d = c$ **using** $rsc \langle s c = d \rangle$ **by** *simp*
then have $r c \neq d$ **using** $\langle d \neq c \rangle$ *simple-rotations-cycle* r *rnotid*
by *auto*
then obtain e **where** $r c = e$ **and** $e \neq d$ **by** *blast*
then have $e \neq a$ **using** $b \langle a = b \rangle$ *simple-rotation-bij-corollary* $\langle c$
 $\neq a \rangle r$ **by** *auto*
then have $e \neq c$ **using** $b \langle a = b \rangle \langle r c = e \rangle \langle r c \neq c \rangle$ **by** *blast*
then have $r e \neq c$ **using** $\langle r c = e \rangle$ *simple-rotations-cycle* r *rnotid*
by *auto*
have $r e \neq a$ **using** $b \langle a = b \rangle \langle e \neq a \rangle$ *simple-rotation-bij-corollary*
 r **by** *auto*
then have $r e \neq e$ **using** $\langle e \neq c \rangle \langle r c = e \rangle r$ *simple-rotation-bij-corollary*
by *blast*
have $dist:distinct [a,c,d,e]$ **using** $\langle c \neq a \rangle \langle d \neq a \rangle \langle d \neq c \rangle \langle e \neq$
 $a \rangle \langle e \neq c \rangle \langle e \neq d \rangle$ **by** *simp*
then have $\forall v \in \{a,c,d\}. r v \neq d$ **using** $\langle r b = b \rangle \langle a = b \rangle \langle r d$
 $= c \rangle \langle r c = e \rangle$ **by** *auto*
then have $r e = d$ **using** *simple-distinct-map* $r dist$ **by** *auto*

have $dist':distinct [a,c,e,d]$ **using** $dist$ **by** *auto*
have $s e \neq e$ **using** $\langle e \neq a \rangle a$ *s-fix-forall* **by** *blast*

```

      then have  $\forall v \in \{a, c, e\}. s v \neq e$  using  $\langle s a = a \rangle \langle s c = d \rangle$  dist
    by auto
      then have  $s d = e$  using distinct-map s dist' by auto
      then have  $\forall v \in \{a, c, d\}. s v \neq c$  using  $\langle s a = a \rangle \langle s c = d \rangle$ 
    dist by auto
      then have  $s e = c$  using distinct-map s dist by auto
      then have  $(r \circ s) d = d$  using  $\langle s d = e \rangle \langle r e = d \rangle$  by auto
      then have  $(r \circ s) e = e$  using  $\langle s e = c \rangle \langle r c = e \rangle$  by auto
      then show  $(r \circ s) = id$  using  $\langle (r \circ s) d = d \rangle \langle (r \circ s) a = a \rangle$ 
     $\langle (r \circ s) c = c \rangle$  dist id-decomp by auto
      qed
    qed
  qed
  qed
  qed
  qed
  qed
  qed

```

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

corollary *complex-rotations-fix*: $r \in \text{complex-rotations} \implies r a = a \implies r b = b \implies a \neq b \implies r = id$
 using *complex-rotations-invariants* by blast

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 $r: r \in \text{simple-rotations}$
 shows $r \circ r \circ r = id$
 using r unfolding *simple-rotations-def*
 by safe
 (rule ext; rename-tac a; case-tac a;
 simp add: *simple-rotations*
)+

lemma *simple-rotations-inverses*:
 assumes $r: r \in \text{simple-rotations}$
 shows $\exists y \in \text{complex-rotations}. y \circ r = id$
proof
 let $?y = r \circ r$
 from r show $y: ?y \in \text{complex-rotations}$ using *complex-rotations.intros* by simp

from *simple-rotations-rotate-id* **show** $?y \circ r = id$ **using** *r* **by** *auto*
qed

lemma *complex-rotations-inverses*:

$r \in \text{complex-rotations} \implies \exists y \in \text{complex-rotations}. y \circ r = id$

proof (*induction r rule: complex-rotations.induct*)

case (*simp r*) **then show** $?case$ **using** *simple-rotations-inverses* **by** *blast*
next

case (*comp r s*)

obtain r' **where** $r'\text{-comp}: r' \in \text{complex-rotations}$ **and** $r'\text{-inv}: r' \circ r = id$

using *simple-rotations-inverses comp.hyps* **by** *auto*

obtain y **where** $y\text{-comp}: y \in \text{complex-rotations}$ **and** $y\text{-inv}: y \circ s = id$ **using**
comp.IH **by** *blast*

from *complex-rotations-comp* **have** $yr': y \circ r' \in \text{complex-rotations}$ **using**
 $r'\text{-comp } y\text{-comp}$ **by** *simp*

have $r' \circ (r \circ s) = r' \circ r \circ s$ **using** *comp-assoc* **by** *metis*

then have $r' \circ (r \circ s) = s$ **using** $r'\text{-inv}$ **by** *simp*

then have $y \circ r' \circ (r \circ s) = id$ **using** $y\text{-inv } comp\text{-assoc}$ **by** *metis*

then show $?case$ **using** yr' **by** *metis*

qed

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**

$tetrahedral\text{-group} = (\text{carrier} = \text{complex-rotations}, \text{mult} = (\circ), \text{one} = id)$

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(*rule groupI*)

show $1_{tetrahedral\text{-group}} \in \text{carrier } tetrahedral\text{-group}$

by (*simp add: complex-rotations.intros(1) simple-rotations-def tetrahedral-group-def*)

next

fix x

assume $x \in \text{carrier } tetrahedral\text{-group}$

show $1_{tetrahedral\text{-group}} \otimes_{tetrahedral\text{-group}} x = x$

unfolding *id-comp tetrahedral-group-def monoid.select-convs(1) monoid.select-convs(2)*

..

next

fix $x y z$

```

assume  $x \in \text{carrier tetrahedral-group}$  and
   $y \in \text{carrier tetrahedral-group}$  and
   $z \in \text{carrier tetrahedral-group}$ 
then show  $x \otimes_{\text{tetrahedral-group}} y \otimes_{\text{tetrahedral-group}} z =$ 
   $x \otimes_{\text{tetrahedral-group}} (y \otimes_{\text{tetrahedral-group}} z)$ 
  unfolding tetrahedral-group-def monoid.select-convs(1) by auto
next
  fix  $x y$ 
  assume  $x \in \text{carrier tetrahedral-group}$  and
     $y \in \text{carrier tetrahedral-group}$ 
  then show  $x \otimes_{\text{tetrahedral-group}} y \in \text{carrier tetrahedral-group}$ 
    by (simp add: complex-rotations.intros(2) tetrahedral-group-def complex-rotations-comp)
next
  fix  $x$ 
  assume  $x \in \text{carrier tetrahedral-group}$ 
  then show  $\exists y \in \text{carrier tetrahedral-group}.$ 
     $y \otimes_{\text{tetrahedral-group}} x = \mathbf{1}_{\text{tetrahedral-group}}$ 
    using complex-rotations-inverses by (simp add: tetrahedral-group-def)
qed

```

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  $\Rightarrow$  Vertex  $\Rightarrow$  Vertex where apply-rotation  $r$ 
   $v = r v$ 

```

```

interpretation tetrahedral: orbit-stabiliser tetrahedral-group apply-rotation
  :: Rotation  $\Rightarrow$  Vertex  $\Rightarrow$  Vertex

```

```

proof intro-locales

```

```

  show Group.monoid tetrahedral-group using is-tetrahedral-group by (simp add: group.is-monoid)

```

```

  show group-axioms tetrahedral-group using is-tetrahedral-group by (simp add: group-def)

```

```

  show orbit-stabiliser-axioms tetrahedral-group apply-rotation

```

```

  proof

```

```

    fix  $x$ 

```

```

    show apply-rotation  $\mathbf{1}_{\text{tetrahedral-group}} x = x$  by (simp add: tetrahedral-group-def)

```

```

    next

```

```

      fix  $g h x$ 

```

```

      show  $g \in \text{carrier tetrahedral-group} \wedge h \in \text{carrier tetrahedral-group}$ 

```

```

         $\longrightarrow$  apply-rotation  $g$  (apply-rotation  $h x$ ) = apply-rotation ( $g$ 

```

```

 $\otimes_{\text{tetrahedral-group}} h$ )  $x$ 

```

```

        by (simp add: tetrahedral-group-def)

```

qed
qed

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

show *tetrahedral.orbit A* \subseteq *vertices* **unfolding** *vertices-def* **using** *Vertex.exhaust* **by** *blast*

have *id* \in *complex-rotations* **using** *complex-rotations.intros simple-rotations-def* **by** *auto*

then have *id* \in *carrier tetrahedral-group*

unfolding *tetrahedral-group-def partial-object.select-convs(1)*.

moreover have *apply-rotation id A = A* **by** *simp*

ultimately have *A:A* \in (*tetrahedral.orbit A*)

using *tetrahedral.orbit-def mem-Collect-eq* **by** *fastforce*

have *rotate-C* \in *simple-rotations*

using *simple-rotations-def insert-subset subset-insertI* **by** *blast*

then have *rotate-C* \in *complex-rotations* **using** *complex-rotations.intros(1)*

by *simp*

then have *rotate-C* \in *carrier tetrahedral-group*

unfolding *tetrahedral-group-def partial-object.select-convs(1)*.

moreover have *apply-rotation rotate-C A = B* **by** (*simp add: rotate-C-def*)

ultimately have *B:B* \in (*tetrahedral.orbit A*)

using *tetrahedral.orbit-def mem-Collect-eq* **by** *fastforce*

have *rotate-D* \in *simple-rotations*

using *simple-rotations-def insert-subset subset-insertI* **by** *blast*

then have *rotate-D* \in *complex-rotations* **using** *complex-rotations.intros(1)*

by *simp*

then have *rotate-D* \in *carrier tetrahedral-group*

unfolding *tetrahedral-group-def partial-object.select-convs(1)*.

moreover have *apply-rotation rotate-D A = C* **by** (*simp add: rotate-D-def*)

ultimately have *C:C* \in (*tetrahedral.orbit A*)

using *tetrahedral.orbit-def mem-Collect-eq* **by** *fastforce*

have *rotate-B* \in *simple-rotations*

using *simple-rotations-def insert-subset subset-insertI* **by** *blast*

then have *rotate-B* \in *complex-rotations* **using** *complex-rotations.intros(1)*

by *simp*

then have $rotate-B \in carrier\ tetrahedral-group$
unfolding $tetrahedral-group-def\ partial-object.select-convs(1)$.
moreover have $apply-rotation\ rotate-B\ A = D$ **by** $(simp\ add:\ rotate-B-def)$
ultimately have $D:D \in (tetrahedral.orbit\ A)$
using $tetrahedral.orbit-def\ mem-Collect-eq$ **by** $fastforce$

from $A\ B\ C\ D$ **show** $vertices \subseteq tetrahedral.orbit\ A$ **by** $(simp\ add:\ vertices-def\ subsetI)$

qed

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 –

from $card.empty\ card-insert-if$ **have** $card\ vertices = 4$ **unfolding** $vertices-def$ **by** $auto$

with $orbit-is-transitive$ **show** $card\ (tetrahedral.orbit\ A) = 4$ **by** $simp$

qed

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**

$stabiliser-A = \{id, rotate-A, rotate-A \circ rotate-A\}$

This lemma shows that our conjectured stabiliser is correct.

lemma $is-stabiliser: tetrahedral.stabiliser\ A = stabiliser-A$

proof

show $stabiliser-A \subseteq tetrahedral.stabiliser\ A$

proof –

have $id \in complex-rotations$ **using** $complex-rotations.intros\ simple-rotations-def$
by $auto$

then have $id \in carrier\ tetrahedral-group$

unfolding $tetrahedral-group-def\ partial-object.select-convs(1)$ **by** $simp$

moreover have $apply-rotation\ id\ A = A$ **by** $simp$

ultimately have $id:id \in (tetrahedral.stabiliser\ A)$

using $tetrahedral.stabiliser-def\ mem-Collect-eq$ **by** $fastforce$

have $rotate-A \in simple-rotations$

using $simple-rotations-def\ insert-subset\ subset-insertI$ **by** $blast$

then have $rotate-A \in complex-rotations$ **using** $complex-rotations.intros(1)$

by $simp$

then have $rotate-A \in carrier\ tetrahedral-group$
unfolding $tetrahedral-group-def\ partial-object.select-convs(1)$ **by** $simp$
moreover have $apply-rotation\ rotate-A\ A = A$ **by** ($simp\ add:\ rotate-A-def$)
ultimately have $A:rotate-A \in (tetrahedral.stabiliser\ A)$
using $tetrahedral.stabiliser-def\ mem-Collect-eq$ **by** $fastforce$

have $rotate-A \in simple-rotations$
using $simple-rotations-def\ insert-subset\ subset-insertI$ **by** $blast$
then have $rotate-A \circ rotate-A \in complex-rotations$ **using** $complex-rotations.intros$
by $simp$
then have $rotate-A \circ rotate-A \in carrier\ tetrahedral-group$
unfolding $tetrahedral-group-def\ partial-object.select-convs(1)$ **by** $simp$
moreover have $apply-rotation\ (rotate-A \circ rotate-A)\ A = A$ **by** ($simp\ add:\ rotate-A-def$)
ultimately have $AA:(rotate-A \circ rotate-A) \in (tetrahedral.stabiliser\ A)$
using $tetrahedral.stabiliser-def\ mem-Collect-eq$ **by** $fastforce$

from $id\ A\ AA$ **show** $stabiliser-A \subseteq tetrahedral.stabiliser\ A$
by ($simp\ add:\ stabiliser-A-def\ subsetI$)
qed
show $tetrahedral.stabiliser\ A \subseteq stabiliser-A$
proof
fix x
assume $a:x \in tetrahedral.stabiliser\ A$
with $tetrahedral.stabiliser-def$ **have** $apply-rotation\ x\ A = A$ **by** $simp$
with $apply-rotation.simps$ **have** $xA:x\ A = A$ **by** $simp$
from a **have** $x \in carrier\ tetrahedral-group$
using $subgroup.mem-carrier[of\ tetrahedral.stabiliser\ A]\ tetrahedral.stabiliser-subgroup$
by $auto$
then have $xC:x \in complex-rotations$
unfolding $tetrahedral-group-def\ partial-object.select-convs(1)$ **by** $simp$
have $x\ B \neq A$ **using** $xA\ xC\ rotation-bij-corollary$ **by** $fastforce$
then have $x \in complex-rotations \implies x\ A = A \implies x \in stabiliser-A$
proof ($cases\ x\ B,\ simp$)
assume $x\ B = B$
then have $x = id$ **using** $complex-rotations-fix\ xC\ xA$ **by** $simp$
then show $?thesis$ **using** $stabiliser-A-def$ **by** $auto$
next
assume $x\ B = C$
then have $x \neq id$ **by** $auto$
then have $x\ D \neq D$ **using** $complex-rotations-fix\ xC\ xA$ **by** $blast$
have $x\ D \neq C$ **using** $xC\ \langle x\ B = C \rangle\ rotation-bij-corollary$ **by** $fastforce$

have $x D \neq A$ **using** $x C x A$ *rotation-bij-corollary* **by** *fastforce*
then have $x D = B$ **using** $\langle x D \neq C \rangle \langle x D \neq D \rangle$ *Vertex.exhaust* **by**
blast

have $x C \neq A$ **using** $x C x A$ *rotation-bij-corollary* **by** *fastforce*
have $x C \neq B$ **using** $x C \langle x D = B \rangle$ *rotation-bij-corollary* **by** *fastforce*

have $x C \neq C$ **using** *complex-rotations-fix* $x C x A \langle x \neq id \rangle$ **by** *blast*
then have $x C = D$ **using** $\langle x C \neq A \rangle \langle x C \neq B \rangle$ *Vertex.exhaust* **by**
blast

have $\forall v. x v = rotate-A v$
using $x A \langle x B = C \rangle \langle x D = B \rangle \langle x C = D \rangle$ *Vertex.exhaust rotate-A-def*
Vertex.simps **by** *metis*
then have $x = rotate-A$ **by** *auto*
then show *?thesis* **using** *stabiliser-A-def* **by** *auto*
next
assume $x B = D$
then have $x \neq id$ **by** *auto*
then have $x C \neq C$ **using** *complex-rotations-fix* $x C x A$ **by** *blast*
have $x C \neq D$ **using** $x C \langle x B = D \rangle$ *rotation-bij-corollary* **by** *fastforce*

have $x C \neq A$ **using** $x C x A$ *rotation-bij-corollary* **by** *fastforce*
then have $x C = B$ **using** $\langle x C \neq D \rangle \langle x C \neq C \rangle$ *Vertex.exhaust* **by**
blast

have $x D \neq A$ **using** $x C x A$ *rotation-bij-corollary* **by** *fastforce*
have $x D \neq B$ **using** $x C \langle x C = B \rangle$ *rotation-bij-corollary* **by** *fastforce*

have $x D \neq D$ **using** *complex-rotations-fix* $x C x A \langle x \neq id \rangle$ **by** *blast*
then have $x D = C$ **using** $\langle x D \neq A \rangle \langle x D \neq B \rangle$ *Vertex.exhaust* **by**
blast

have $\forall v. x v = (rotate-A \circ rotate-A) v$
using $x A \langle x B = D \rangle \langle x C = B \rangle \langle x D = C \rangle$ *Vertex.exhaust rotate-A-def*
Vertex.simps comp-apply **by** *metis*
then have $x = rotate-A \circ rotate-A$ **by** *auto*
then show *?thesis* **using** *stabiliser-A-def* **by** *auto*
qed
then show $x \in stabiliser-A$ **using** $x A x C$ **by** *simp*
qed
qed

Using the previous result, we can now show that the cardinality of the sta-

biliser is 3.

lemma *card-stabiliser-help*: $\text{card stabiliser-}A = 3$

proof –

have *idA*: $\text{id} \neq \text{rotate-}A$

proof –

have $\text{id } B = B$ **by** *simp*

moreover have $\text{rotate-}A B = C$ **by** (*simp add: rotate-A-def*)

ultimately show $\text{id} \neq \text{rotate-}A$ **by force**

qed

have *idAA*: $\text{id} \neq \text{rotate-}A \circ \text{rotate-}A$

proof –

have $\text{id } B = B$ **by** *simp*

moreover have $(\text{rotate-}A \circ \text{rotate-}A) B = D$ **by** (*simp add: rotate-A-def*)

ultimately show $\text{id} \neq \text{rotate-}A \circ \text{rotate-}A$ **by force**

qed

have *AAA*: $\text{rotate-}A \neq \text{rotate-}A \circ \text{rotate-}A$

proof –

have $\text{rotate-}A B = C$ **by** (*simp add: rotate-A-def*)

moreover have $(\text{rotate-}A \circ \text{rotate-}A) B = D$ **by** (*simp add: rotate-A-def*)

ultimately show $\text{rotate-}A \neq \text{rotate-}A \circ \text{rotate-}A$ **by force**

qed

from *idA idAA AAA card.empty card-insert-if* **show**

$(\text{card stabiliser-}A) = 3$ **unfolding** *stabiliser-A-def* **by auto**

qed

lemma *card-stabiliser*: $\text{card } (\text{tetrahedral.stabiliser } A) = 3$

using *is-stabiliser card-stabiliser-help* **by simp**

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*: $(UNIV :: \text{Vertex set}) = \{A, B, C, D\}$

by(*auto, metis Vertex.exhaust*)

lemma *vertex-finite*: $\text{finite } (UNIV :: \text{Vertex set})$

by (*simp add: vertex-set*)

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

show *finite* (*UNIV* :: *Vertex set*) **by** (*simp add: vertex-set*)

qed

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

lemma *finite-carrier: finite (carrier tetrahedral-group)*

proof –

have *finite (UNIV :: (Vertex \Rightarrow Vertex) set)* **by** *simp*

moreover have *complex-rotations \subseteq (UNIV :: (Vertex \Rightarrow Vertex) set)*
by *simp*

ultimately show *finite (carrier tetrahedral-group)* **using** *finite-subset top-greatest* **by** *blast*

qed

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 –

have *card (tetrahedral.orbit A) * card (tetrahedral.stabiliser A) = 12*

using *card-stabiliser card-orbit* **by** *simp*

with *tetrahedral.orbit-stabiliser[OF finite-carrier]*

show *order tetrahedral-group = 12* **by** *simp*

qed

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].