

# Orbit-Stabiliser Theorem with Application to Rotational Symmetries

Jonas Rädle

September 13, 2023

## 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

<b>1</b>	<b>Orbit-Stabiliser Theorem</b>	<b>2</b>
1.1	Imports . . . . .	2
1.2	Group Actions . . . . .	2
1.3	Orbit and stabiliser . . . . .	2
1.4	Stabiliser Theorems . . . . .	3
1.5	Picking representatives from cosets . . . . .	5
1.6	Orbit-Stabiliser Theorem . . . . .	6
<b>2</b>	<b>Rotational Symmetries of the Tetrahedron</b>	<b>8</b>
2.1	Definition of the Tetrahedron and its Rotations . . . . .	9
2.2	Properties of Rotations . . . . .	9
2.3	Inversions . . . . .	25
2.4	The Tetrahedral Group . . . . .	26
2.5	Counting Orbits . . . . .	27
2.6	Counting Stabilisers . . . . .	29
2.7	Proving Finiteness . . . . .	32
2.8	Order of the Group . . . . .	32

# 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
    HOL-Algebra.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:
     $g \in \text{carrier } G \wedge h \in \text{carrier } G \longrightarrow g \odot (h \odot x) = (g \otimes h) \odot x$ 
```

## 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
```

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

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

## 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**(*rule subgroupI*)

**show** *stabiliser*  $x \subseteq carrier\ G$  **using** *stabiliser-def* **by** *auto*

**next**

**fix**  $x$

**from** *id-act* **have**  $1 \odot x = x$  **by** *simp*

**then** **have**  $1 \in stabiliser\ x$  **using** *stabiliser-def* **by** *auto*

**then** **show** *stabiliser*  $x \neq \{\}$  **by** *auto*

**next**

**fix**  $g\ x$

**assume** *gStab*:  $g \in stabiliser\ x$

**then** **have** *g-car*:  $g \in carrier\ G$  **using** *stabiliser-def* **by** *simp*

**then** **have** *inv-g-car*:  $inv\ g \in carrier\ G$  **using** *inv-closed* **by** *simp*

**have**  $g \odot x = x$  **using** *stabiliser-def gStab* **by** *simp*

**then** **have**  $inv\ g \odot (g \odot x) = inv\ g \odot x$  **by** *simp*

**then** **have**  $(inv\ g \otimes g) \odot x = inv\ g \odot x$  **using** *compat-act g-car inv-g-car*

**by** *simp*

**then** **have**  $x = (inv\ g) \odot x$  **using** *g-car l-inv* **by** *simp*

**then** **show**  $inv\ g \in stabiliser\ x$  **using** *inv-g-car stabiliser-def* **by** *simp*

**next**

**fix**  $g\ h\ x$

**assume** *g-stab*:  $g \in stabiliser\ x$  **and** *h-stab*:  $h \in stabiliser\ x$

**then** **have** *g-car*:  $g \in carrier\ G$  **and** *h-car*:  $h \in carrier\ G$  **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**  $(g \otimes h) \in stabiliser\ x$

**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**  $\text{inv}g\text{-car}: (\text{inv } g) \in \text{carrier } G$  **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**  $\text{inv}g\text{-car}$

$g\text{-car}$  **compat-act** **by** *simp*

**moreover have**  $((\text{inv } g) \otimes g) \odot x = x$  **using**  $g\text{-car}$  **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**  $1 \odot x = ((\text{inv } g) \otimes h) \odot x$  **by** *simp*

**then have**  $((\text{inv } g) \otimes g) \odot x = ((\text{inv } g) \otimes h) \odot x$  **using**  $\text{inv}g\text{-car}$   $g\text{-car}$

**by** *simp*

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

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

$\text{inv}g\text{-car}$  **by** *metis*

**then have**  $g \odot x = h \odot x$  **using** *compat-act*  $g\text{-car}$  *id-act*  $\text{inv}g\text{-car}$  *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**  $g\text{-car}: g \in \text{carrier } G$  **and**

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

**shows**  $(g \odot x) = (h \odot x) \longleftrightarrow (g <\# \text{ stabiliser } x) = (h <\# \text{ stabiliser } x)$   
**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*:  $((\text{inv } g) \otimes h) \in \text{ stabiliser } x$   
**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**  $g <\# (\text{ stabiliser } x) = h <\# (\text{ stabiliser } x)$   
*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**  $((\text{inv } g) \otimes$   
 $h) \in \text{ stabiliser } x$   
**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**  
 $(H \in \text{ carrier } (G \text{ LMod } (\text{ stabiliser } x))) \implies \text{ rep } H = (\text{ SOME } y. y \in H)$

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

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

**proof** –  
**fix**  $H$   
**assume**  $H:H \in \text{carrier } (G \text{ LMod stabiliser } x)$   
**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**  $H:H \in \text{carrier } (G \text{ LMod stabiliser } x)$  **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**  $H = g <\# (\text{stabiliser } x)$  **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*  $(\lambda H. \text{rep } H \odot x) (\text{carrier } (G \text{ LMod } (\text{stabiliser } x))) (\text{orbit } x)$   
**proof** *(rule bij-betw-imageI)*

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

**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: (rep H)  $\odot$  x = (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  $\subseteq$  carrier G
  unfolding LFactGroup-def by simp
then have hG: h  $\in$  carrier G using hH by auto
from subgroup.lcosets-carrier[OF stabiliser-subgroup is-group] H' have
H'  $\subseteq$  carrier G
  unfolding LFactGroup-def by simp
then have h'G: h'  $\in$  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. rep H  $\odot$  x) ' carrier (G LMod stabiliser x) = orbit x
proof(auto)
  show  $\bigwedge$ H. H  $\in$  carrier (G LMod stabiliser x)  $\implies$  rep H  $\odot$  x  $\in$  orbit x
  proof –
    fix H
    assume H:H  $\in$  carrier (G LMod (stabiliser x))
    obtain h where h:h = rep H by simp
    from H h quotient-rep-ex have hH: h  $\in$  H by simp
    have stab-sub: (stabiliser x)  $\subseteq$  carrier G using stabiliser-def by auto
    from subgroup.lcosets-carrier[OF stabiliser-subgroup is-group] H have
H  $\subseteq$  carrier G
      unfolding LFactGroup-def by simp
      with hH have h  $\in$  carrier G by auto
      then show (rep H)  $\odot$  x  $\in$  orbit x using h orbit-def mem-Collect-eq
by blast
    qed
  show  $\bigwedge$ y. y  $\in$  orbit x  $\implies$  y  $\in$  ( $\lambda$ H. rep H  $\odot$  x) ' carrier (G LMod
stabiliser x)
  proof –
    fix y
    assume y:y  $\in$  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  $H\text{-carr}:H \in \text{carrier } (G \text{ LMod stabiliser } x)$ 
    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  $h\text{-stab stabiliser-def}$  by auto
  from  $\text{stabiliser-def } h\text{-stab}$  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  $gG hG \text{ compat-act}$  by auto
  then have  $y = (\text{rep } H) \odot x$  using  $gh$  by simp
  then show  $y \in (\lambda H. \text{rep } H \odot x)$  ‘  $\text{carrier } (G \text{ LMod stabiliser } x)$ 
    using  $H\text{-carr}$  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 order  $G = \text{card } (\text{orbit } x) * \text{card } (\text{stabiliser } x)$ 
proof –
  have  $\text{card } (\text{carrier } (G \text{ LMod } (\text{stabiliser } x))) = \text{card } (\text{orbit } x)$ 
    using bij-betw-same-card orbit-stabiliser-bij by auto
  moreover have  $\text{card } (\text{carrier } (G \text{ LMod } (\text{stabiliser } x))) * \text{card } (\text{stabiliser } x) = \text{order } G$ 
    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* ⇒ *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* = (λv. (case v of A ⇒ A | B ⇒ C | C ⇒ D | D ⇒ B))

**definition** *rotate-B* :: *Rotation* **where**

*rotate-B* = (λv. (case v of A ⇒ D | B ⇒ B | C ⇒ A | D ⇒ C))

**definition** *rotate-C* :: *Rotation* **where**

*rotate-C* = (λv. (case v of A ⇒ B | B ⇒ D | C ⇒ C | D ⇒ A))

**definition** *rotate-D* :: *Rotation* **where**

*rotate-D* = (λv. (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*:  $r \in \text{simple-rotations} \implies r \in \text{complex-rotations}$  |

*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*:  $\text{distinct}[(a:\text{Vertex}),b,c,d] \implies (\forall e. e \in \{a,b,c,d\})$

**apply**(*safe*)

**apply**(*case-tac a*)

**apply**(*auto simp add: distinct-def*)

**apply**(*metis Vertex.exhaust*)+

**done**

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

**proof** –

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

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

**proof** –

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

**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*:  $r \in \text{complex-rotations} \implies \text{distinct}[a,b,c,d] \implies$   
 $r a = b \implies r b = a \implies r c = d \implies r d = c \implies \forall v w. r v = w \implies r w = v$

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

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

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

**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*

*blast*

**then show** *?thesis*

**proof** (*cases*  $r = \text{id}$ )

**assume**  $\text{id}:r = \text{id}$

**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**  $\text{notid}:r \neq \text{id}$   
**obtain**  $a$  **where**  $s v = a$  **and**  $s a = v$  **and**  $a \neq v$  **using**  $\text{comp.IH}$   
*nofixs* **by** *blast*  
**obtain**  $b$  **where**  $s w = b$  **and**  $s b = w$  **and**  $b \neq w$  **using**  $\text{comp.IH}$   
*nofixs* **by** *blast*  
**have**  $v \neq w$  **using**  $\langle (r \circ s) v = w \rangle$  *nofixrs* **by** *blast*  
**then have**  $a \neq b$  **using**  $\text{comp.hyps}$  *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*  $\text{comp.hyps}$   
**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*  $\text{notid}$   $\langle r a = a \rangle$   $\langle a \neq v \rangle$   
*comp.hyps(1)* **by** *auto*  
**obtain**  $c d$  **where**  $s c = d$  **and**  $c \neq v$  **and**  $c \neq w$   
**using**  $\langle a \neq v \rangle$   $\langle r a = w \rangle$   $\langle r v \neq v \rangle$   $\text{comp.hyps(1)}$  *simple-rotation-bij-corollary* **by** *blast*  
**then have**  $d \neq v$  **and**  $d \neq w$   
**using**  $\langle s w = v \rangle$   $\langle c \neq v \rangle$   $\langle s c = d \rangle$   $\langle s v = w \rangle$   $\text{comp.hyps(2)}$   
*rotation-bij-corollary* **by** *auto*  
**then have**  $s d = c$  **using**  $\langle s c = d \rangle$   $\text{comp.IH}$  *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*  
 $\text{comp.hyps(1)}$   $\text{notid}$  **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*  
 $\text{comp.hyps(1)}$  **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*  $\text{notid}$  **by** *blast*  
**have**  $\text{dist:distinct}$   $[v, w, c, d]$  **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'*  $\text{comp.hyps(1)}$

```

dist by auto
  then have  $(r \circ s) d = d$  using  $\langle s d = c \rangle$  by simp
  then have False using nofixrs 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 nofixs 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 ≠ a using ⟨a ≠ b⟩ by simp
  have r w ≠ a using ⟨r a = w⟩ ⟨r w ≠ w⟩ comp.hyps(1) notid
simple-rotations-cycle by blast
  have dist:distinct [a,b,w,v] using ⟨a ≠ w⟩ ⟨a ≠ b⟩ ⟨a ≠ v⟩ ⟨b
  ≠ w⟩ ⟨b ≠ v⟩ ⟨v ≠ w⟩ by simp
  then have ∀ x ∈ {a,b,w}. r x ≠ a using ⟨r a ≠ a⟩ ⟨r b ≠ a⟩
  ⟨r w ≠ a⟩ by auto
  then have r v = a using simple-distinct-map comp.hyps(1) dist
by auto
  then show ?thesis using ⟨s a = v⟩ nofixrs comp-apply by metis
next
  assume r b ≠ b
  have dist:distinct [w,a,b,v] using ⟨a ≠ w⟩ ⟨a ≠ b⟩ ⟨a ≠ v⟩ ⟨b
  ≠ w⟩ ⟨b ≠ v⟩ ⟨v ≠ w⟩ by simp
  then have ∀ x ∈ {w,a,b}. r x ≠ x using ⟨r w ≠ w⟩ ⟨r a ≠ a⟩
  ⟨r b ≠ b⟩ by auto
  then have r v = v using simple-distinct-ident comp.hyps(1)
dist by auto
  have r w ≠ a using ⟨a ≠ w⟩ simple-rotations-cycle comp.hyps(1)
notid ⟨r a = w⟩ by simp
  have r w ≠ v using ⟨r v = v⟩ ⟨v ≠ w⟩ comp.hyps(1)
simple-rotation-bij-corollary by blast
  have dist':distinct [a,v,w,b] using ⟨a ≠ w⟩ ⟨a ≠ b⟩ ⟨a ≠ v⟩ ⟨b
  ≠ w⟩ ⟨b ≠ v⟩ ⟨v ≠ w⟩ by simp
  then have ∀ x ∈ {a,v,w}. r w ≠ x using ⟨r w ≠ a⟩ ⟨r w ≠ v⟩
  ⟨r w ≠ w⟩ by auto
  then have r w = b using simple-distinct-map' comp.hyps(1)
dist' by auto
  then show ?thesis using ⟨s b = w⟩ nofixrs comp-apply by metis
qed
qed
qed
next
assume ¬ (∀ v. s v ≠ v)
then have fix1: ∃ 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 ∘ s) a = a using a by simp
  then have False using nofixrs by auto

```

**then show** *?thesis* **by** *simp*  
**next**  
**assume** *notid: r ≠ id*  
**then have** *fix1:∃ 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 ∘ s) a = a* **using** *a b* **by** *simp*  
**then have** *False* **using** *nofixrs* **by** *blast*  
**then show** *?thesis* **by** *simp*  
**next**  
**assume** *a ≠ b*  
**have** *r a ≠ a* **using** *⟨a ≠ b⟩ b comp.hyps(1) notid simple-rotations-fix-unique*  
**by** *blast*  
**have** *r a ≠ b* **using** *⟨a ≠ b⟩ b comp.hyps(1) simple-rotation-bij-corollary*  
**by** *auto*  
**then obtain** *c* **where** *r a = c* **and** *a ≠ c* **and** *b ≠ c* **using** *⟨r a ≠ a⟩* **by** *auto*  
**have** *s b ≠ a* **using** *⟨a ≠ b⟩ a comp.hyps(2) rotation-bij-corollary*  
**by** *blast*  
**have** *s b ≠ b* **using** *b nofixrs comp-apply* **by** *metis*  
**then obtain** *d* **where** *s b = d* **and** *a ≠ d* **and** *b ≠ d* **using** *⟨s b ≠ a⟩* **by** *auto*  
**have** *r c ≠ a* **using** *simple-rotations-cycle ⟨a ≠ c⟩ ⟨r a = c⟩ comp.hyps(1) notid* **by** *blast*  
**have** *r c ≠ b* **using** *⟨b ≠ c⟩ b comp.hyps(1) simple-rotation-bij-corollary*  
**by** *blast*  
**have** *r c ≠ c* **using** *⟨b ≠ c⟩ 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 ≠ c* **using** *⟨b ≠ c⟩ ⟨s b = d⟩ rotation-bij-corollary*  
*s* **by** *auto*  
**obtain** *e* **where** *r c = e* **and** *a ≠ e* **and** *b ≠ e* **and** *c ≠ e* **and** *d ≠ e*  
**using** *⟨r c ≠ a⟩ ⟨r c ≠ b⟩ ⟨r c ≠ c⟩ ⟨c = d⟩* **by** *auto*  
**have** *r e ≠ b* **using** *⟨b ≠ e⟩ b r simple-rotation-bij-corollary* **by** *blast*  
**have** *r e ≠ c* **using** *⟨a ≠ e⟩ ⟨r a = c⟩ r simple-rotation-bij-corollary*  
**by** *blast*  
**have** *r e ≠ e* **using** *⟨b ≠ e⟩ b notid r simple-rotations-fix-unique*

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

```

    then have False using fix-unique  $\langle s\ d \neq d \rangle$   $\langle a \neq c \rangle$  a by fastforce

    then show ?thesis by simp
  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  $\langle v \neq w \rangle$ 
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  $\langle v \neq w \rangle$  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**  $\langle r d \neq c \rangle \langle r d \neq a \rangle \langle r$

$d \neq d$  **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*(3)  $s$  *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**

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

**corollary** *complex-rotations-fix*:  $r \in \text{complex-rotations} \implies r a = a \implies r b = b \implies a \neq b \implies r = \text{id}$   
**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**  $r \circ r \circ r = \text{id}$   
**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**  $\exists y \in \text{complex-rotations}. y \circ r = \text{id}$

**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**  $?y \circ r = \text{id}$  **using**  $r$  **by** *auto*  
**qed**

**lemma** *complex-rotations-inverses*:

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

**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**  $r'\text{-inv}: r' \circ r = \text{id}$   
**using** *simple-rotations-inverses comp.hyps* **by** *auto*  
**obtain**  $y$  **where**  $y\text{-comp}: y \in \text{complex-rotations}$  **and**  $y\text{-inv}: y \circ s = \text{id}$  **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-inv comp-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-group* = (*carrier* = *complex-rotations*, *mult* = ( $\circ$ ), *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**  $1_{\text{tetrahedral-group}} \in \text{carrier tetrahedral-group}$

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

**next**

**fix** *x*

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

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

**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: tetrahe-
      dral-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  $A = \text{vertices}$ 

```

```

proof

```

```

  show tetrahedral.orbit  $A \subseteq \text{vertices}$  unfolding vertices-def using Ver-
  tex.exhaust by blast

```

```

  have  $id \in \text{complex-rotations}$  using complex-rotations.intros simple-rotations-def
  by auto

```

```

  then have  $id \in \text{carrier tetrahedral-group}$ 

```

```

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

```

```

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

```

```

    ultimately have  $A:A \in (\text{tetrahedral.orbit } A)$ 

```

**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**  $B: B \in (\textit{tetrahedral.orbit } A)$   
**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**  $C: C \in (\textit{tetrahedral.orbit } A)$   
**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**  $D: D \in (\textit{tetrahedral.orbit } A)$   
**using** *tetrahedral.orbit-def mem-Collect-eq* **by** *fastforce*

**from**  $A B C D$  **show**  $\textit{vertices} \subseteq \textit{tetrahedral.orbit } A$  **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*:  $\textit{card} (\textit{tetrahedral.orbit } A) = 4$

**proof** –

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

**with** *orbit-is-transitive* **show**  $\textit{card} (\textit{tetrahedral.orbit } A) = 4$  **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* ◦ *rotate-A*}

This lemma shows that our conjectured stabiliser is correct.

**lemma** *is-stabiliser*: *tetrahedral.stabiliser A* = *stabiliser-A*

**proof**

**show** *stabiliser-A* ⊆ *tetrahedral.stabiliser A*

**proof** –

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

**then have** *id* ∈ *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* ∈ (*tetrahedral.stabiliser A*)

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

**have** *rotate-A* ∈ *simple-rotations*

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

**then have** *rotate-A* ∈ *complex-rotations* **using** *complex-rotations.intros(1)*

**by** *simp*

**then have** *rotate-A* ∈ *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* ∈ (*tetrahedral.stabiliser A*)

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

**have** *rotate-A* ∈ *simple-rotations*

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

**then have** *rotate-A* ◦ *rotate-A* ∈ *complex-rotations* **using** *complex-rotations.intros*

**by** *simp*

**then have** *rotate-A* ◦ *rotate-A* ∈ *carrier tetrahedral-group*

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

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

**ultimately have** *AA: (rotate-A ◦ rotate-A)* ∈ (*tetrahedral.stabiliser A*)

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

**from** *id A AA* **show** *stabiliser-A* ⊆ *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 xC xA 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 xC xA rotation-bij-corollary by fastforce
  have x C  $\neq$  B using xC  $\langle$ x D = B $\rangle$  rotation-bij-corollary by fastforce

  have x C  $\neq$  C using complex-rotations-fix xC xA  $\langle$ 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  $\forall v. x v = \text{rotate-A } v$ 
  using xA  $\langle$ 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 xC xA* **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 xC xA*  $\langle x \neq id \rangle$  **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 stabiliser is 3.

**lemma** *card-stabiliser-help: card stabiliser-A = 3*

**proof** –

**have**  $id A : id \neq rotate-A$

**proof** –

**have**  $id B = B$  **by** *simp*

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

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

**qed**

**have**  $id A A : id \neq rotate-A \circ rotate-A$

**proof** –

**have**  $id B = B$  **by** *simp*

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

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

**qed**

**have**  $AAA : rotate-A \neq rotate-A \circ rotate-A$

**proof** –

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

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

```

    ultimately show rotate-A ≠ rotate-A ◦ rotate-A by force
  qed
  from idA idAA AAA card.empty card-insert-if show
    (card stabiliser-A) = 3 unfolding stabiliser-A-def by auto
  qed

```

```

lemma card-stabiliser: card (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 :: Vertex set) = {A, B, C, D}
  by(auto, metis Vertex.exhaust)

```

```

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

```

Next we need to 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 ⇒ Vertex) set) by simp
  moreover have complex-rotations ⊆ (UNIV :: (Vertex ⇒ 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](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](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](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](https://en.wikipedia.org/wiki/Group_action), 2017. [Online; accessed 18-July-2017].