

Octonions

Angeliki Koutsoukou-Argyaki

February 23, 2021

Abstract

We develop the basic theory of Octonions, including various identities and properties of the octonions and of the octonionic product, a description of 7D isometries and representations of orthogonal transformations. To this end we first develop the theory of the vector cross product in 7 dimensions. The development of the theory of Octonions is inspired by that of the theory of Quaternions by Lawrence Paulson. However, we do not work within the type class *real_algebra_1* because the octonionic product is not associative.

Contents

1	Vector Cross Product in 7 Dimensions	3
1.1	Elementary auxiliary lemmas.	3
1.2	The definition of the 7D cross product and related lemmas . .	4
1.3	Continuity	11
2	Theory of Octonions	11
2.1	Basic definitions	11
2.2	Addition and Subtraction: An Abelian Group	12
2.3	A Normed Vector Space	13
2.4	The Octonionic product and related properties and lemmas .	17
2.5	Embedding of the Reals into the Octonions	20
2.6	”Expansion” into the traditional notation	24
2.7	Conjugate of an octonion and related properties.	25
2.8	Linearity and continuity of the components.	29
	2.8.1 Octonionic-specific theorems about sums.	31
	2.8.2 Bound results for real and imaginary components of limits.	31
2.9	Octonions for describing 7D isometries	33
	2.9.1 The HI operator	33
	2.9.2 The Hv operator	34
	2.9.3 Related basic identities	35
2.10	Representing orthogonal transformations as conjugation or congruence with an octonion.	35

Acknowledgements. A.K.-A. was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK. Many thanks to Manuel Eberl, Wenda Li and Lawrence Paulson for their suggestions and improvements.

1 Vector Cross Product in 7 Dimensions

```
theory Cross-Product-7
  imports HOL-Analysis.Multivariate-Analysis
begin
```

1.1 Elementary auxiliary lemmas.

```
lemma exhaust-7:
  fixes x :: 7
  shows x = 1 ∨ x = 2 ∨ x = 3 ∨ x = 4 ∨ x = 5 ∨ x = 6 ∨ x = 7
proof (induct x)
  case (of-int z)
  then have 0 ≤ z and z < 7 by simp-all
  then have z = 0 ∨ z = 1 ∨ z = 2 ∨ z = 3 ∨ z = 4 ∨ z = 5 ∨ z = 6 ∨ z = 7
  by arith
  then show ?case by auto
qed
```

```
lemma forall-7: (∀ i::7. P i) ⟷ P 1 ∧ P 2 ∧ P 3 ∧ P 4 ∧ P 5 ∧ P 6 ∧ P 7
  by (metis exhaust-7)
```

```
lemma vector-7 [simp]:
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$1 = x1
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$2 = x2
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$3 = x3
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$4 = x4
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$5 = x5
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$6 = x6
  (vector [x1,x2,x3,x4,x5,x6,x7] :: ('a::zero) ^ 7)$7 = x7
  unfolding vector-def by auto
```

```
lemma forall-vector-7:
  (∀ v::'a::zero ^ 7. P v) ⟷ (∀ x1 x2 x3 x4 x5 x6 x7. P(vector[x1, x2, x3, x4, x5, x6,
x7]))
  apply auto
  apply (erule-tac x=v$1 in allE)
  apply (erule-tac x=v$2 in allE)
  apply (erule-tac x=v$3 in allE)
  apply (erule-tac x=v$4 in allE)
  apply (erule-tac x=v$5 in allE)
  apply (erule-tac x=v$6 in allE)
```

```

apply (erule-tac x=v$7 in alle)
apply (subgoal-tac vector [v$1, v$2, v$3, v$4, v$5, v$6, v$7] = v)
apply simp
apply (vector vector-def)
apply (simp add: forall-7)
done

```

```

lemma UNIV-7: UNIV = {1::7, 2::7, 3::7, 4::7, 5::7, 6::7, 7::7}
using exhaust-7 by auto

```

```

lemma sum-7: sum f (UNIV::7 set) = f 1 + f 2 + f 3 + f 4 + f 5 + f 6 + f 7
unfolding UNIV-7 by (simp add: ac-simps)

```

```

lemma not-equal-vector7 :
  fixes x::real^7 and y::real^7
  assumes x = vector[x1,x2,x3,x4,x5,x6,x7] and y = vector [y1,y2,y3,y4,y5,y6,y7]
and x$1 ≠ y$1 ∨ x$2 ≠ y$2 ∨ x$3 ≠ y$3 ∨ x$4 ≠ y$4 ∨ x$5 ≠ y$5 ∨ x$6 ≠
y$6 ∨ x$7 ≠ y$7
shows x ≠ y using assms by blast

```

```

lemma equal-vector7:
  fixes x::real^7 and y::real^7
  assumes x = vector[x1,x2,x3,x4,x5,x6,x7] and y = vector [y1,y2,y3,y4,y5,y6,y7]
and x = y
shows x$1 = y$1 ∧ x$2 = y$2 ∧ x$3 = y$3 ∧ x$4 = y$4 ∧ x$5 = y$5 ∧ x$6 =
y$6 ∧ x$7 = y$7
using assms by blast

```

```

lemma numeral-4-eq-4: 4 = Suc( Suc (Suc (Suc 0)))
by (simp add: eval-nat-numeral)
lemma numeral-5-eq-5: 5 = Suc(Suc( Suc (Suc (Suc 0))))
by (simp add: eval-nat-numeral)
lemma numeral-6-eq-6: 6 = Suc( Suc(Suc( Suc (Suc (Suc 0))))))
by (simp add: eval-nat-numeral)
lemma numeral-7-eq-7: 7 = Suc(Suc( Suc(Suc( Suc (Suc (Suc 0))))))
by (simp add: eval-nat-numeral)

```

1.2 The definition of the 7D cross product and related lemmas

```

context includes no-Set-Product-syntax
begin

```

Note: in total there exist 480 equivalent multiplication tables for the definition, the following is based on the one most widely used:

```

definition cross7 :: [real^7, real^7] ⇒ real^7 (infixr ×7 80)
  where a ×7 b ≡
  vector [a$2 * b$4 - a$4 * b$2 + a$3 * b$7 - a$7 * b$3 + a$5 * b$6 - a$6
* b$5 ,

```

```

    a$3 * b$5 - a$5 * b$3 + a$4 * b$1 - a$1 * b$4 + a$6 * b$7 - a$7
* b$6 ,
    a$4 * b$6 - a$6 * b$4 + a$5 * b$2 - a$2 * b$5 + a$7 * b$1 - a$1
* b$7 ,
    a$5 * b$7 - a$7 * b$5 + a$6 * b$3 - a$3 * b$6 + a$1 * b$2 - a$2
* b$1 ,
    a$6 * b$1 - a$1 * b$6 + a$7 * b$4 - a$4 * b$7 + a$2 * b$3 - a$3
* b$2 ,
    a$7 * b$2 - a$2 * b$7 + a$1 * b$5 - a$5 * b$1 + a$3 * b$4 - a$4
* b$3 ,
    a$1 * b$3 - a$3 * b$1 + a$2 * b$6 - a$6 * b$2 + a$4 * b$5 - a$5
* b$4 ]

```

end

bundle *cross7-syntax* **begin**

notation *cross7* (**infixr** \times_7 80)

no-notation *Product-Type.Times* (**infixr** \times_7 80)

end

bundle *no-cross7-syntax* **begin**

no-notation *cross7* (**infixr** \times_7 80)

notation *Product-Type.Times* (**infixr** \times_7 80)

end

unbundle *cross7-syntax*

lemmas *cross7-simps* = *cross7-def inner-vec-def sum-7 det-def vec-eq-iff vector-def algebra-simps*

lemma *dot-cross7-self*: $x \cdot (x \times_7 y) = 0$ $x \cdot (y \times_7 x) = 0$ $(x \times_7 y) \cdot y = 0$ $(y \times_7 x) \cdot x = 0$

by (*simp-all add: orthogonal-def cross7-simps*)

lemma *orthogonal-cross7*: *orthogonal* $(x \times_7 y)$ x *orthogonal* $(x \times_7 y)$ y
orthogonal y $(x \times_7 y)$ *orthogonal* $(x \times_7 y)$ x

by (*simp-all add: orthogonal-def dot-cross7-self*)

lemma *cross7-zero-left* [*simp*]: $0 \times_7 x = 0$

and *cross7-zero-right* [*simp*]: $x \times_7 0 = 0$

by (*simp-all add: cross7-simps*)

lemma *cross7-skew*: $(x \times_7 y) = -(y \times_7 x)$

by (*simp add: cross7-simps*)

lemma *cross7-refl* [*simp*]: $x \times_7 x = 0$

by (*simp add: cross7-simps*)

lemma *cross7-add-left*: $(x + y) \times_7 z = (x \times_7 z) + (y \times_7 z)$

and *cross7-add-right*: $x \times_7 (y + z) = (x \times_7 y) + (x \times_7 z)$
by (*simp-all add: cross7-simps*)

lemma *cross7-mult-left*: $(c *_R x) \times_7 y = c *_R (x \times_7 y)$
and *cross7-mult-right*: $x \times_7 (c *_R y) = c *_R (x \times_7 y)$
by (*simp-all add: cross7-simps*)

lemma *cross7-minus-left* [*simp*]: $(-x) \times_7 y = -(x \times_7 y)$
and *cross7-minus-right* [*simp*]: $x \times_7 -y = -(x \times_7 y)$
by (*simp-all add: cross7-simps*)

lemma *left-diff-distrib*: $(x - y) \times_7 z = x \times_7 z - y \times_7 z$
and *right-diff-distrib*: $x \times_7 (y - z) = x \times_7 y - x \times_7 z$
by (*simp-all add: cross7-simps*)

hide-fact (**open**) *left-diff-distrib right-diff-distrib*

lemma *cross7-triple1*: $(x \times_7 y) \cdot z = (y \times_7 z) \cdot x$
and *cross7-triple2*: $(x \times_7 y) \cdot z = x \cdot (y \times_7 z)$
by (*simp-all add: cross7-def inner-vec-def sum-7 vec-eq-iff algebra-simps*)

lemma *scalar7-triple1*: $x \cdot (y \times_7 z) = y \cdot (z \times_7 x)$
and *scalar7-triple2*: $x \cdot (y \times_7 z) = z \cdot (x \times_7 y)$
by (*simp-all add: cross7-def inner-vec-def sum-7 vec-eq-iff algebra-simps*)

lemma *cross7-components*:
 $(x \times_7 y)\$1 = x\$2 * y\$4 - x\$4 * y\$2 + x\$3 * y\$7 - x\$7 * y\$3 + x\$5 * y\$6$
 $- x\$6 * y\5
 $(x \times_7 y)\$2 = x\$4 * y\$1 - x\$1 * y\$4 + x\$3 * y\$5 - x\$5 * y\$3 + x\$6 * y\$7$
 $- x\$7 * y\6
 $(x \times_7 y)\$3 = x\$5 * y\$2 - x\$2 * y\$5 + x\$4 * y\$6 - x\$6 * y\$4 + x\$7 * y\$1$
 $- x\$1 * y\7
 $(x \times_7 y)\$4 = x\$1 * y\$2 - x\$2 * y\$1 + x\$6 * y\$3 - x\$3 * y\$6 + x\$5 * y\$7$
 $- x\$7 * y\5
 $(x \times_7 y)\$5 = x\$6 * y\$1 - x\$1 * y\$6 + x\$2 * y\$3 - x\$3 * y\$2 + x\$7 * y\$4$
 $- x\$4 * y\7
 $(x \times_7 y)\$6 = x\$1 * y\$5 - x\$5 * y\$1 + x\$7 * y\$2 - x\$2 * y\$7 + x\$3 * y\$4$
 $- x\$4 * y\3
 $(x \times_7 y)\$7 = x\$1 * y\$3 - x\$3 * y\$1 + x\$4 * y\$5 - x\$5 * y\$4 + x\$2 * y\$6$
 $- x\$6 * y\2
by (*simp-all add: cross7-def inner-vec-def sum-7 vec-eq-iff algebra-simps*)

Nonassociativity of the 7D cross product showed using a counterexample

lemma *cross7-nonassociative*:

$\neg (\forall (c::real^7) (a::real^7) (b::real^7) . c \times_7 (a \times_7 b) = (c \times_7 a) \times_7 b)$

proof–

have *: $\exists (c::real^7) (a::real^7) (b::real^7) . c \times_7 (a \times_7 b) \neq (c \times_7 a) \times_7 b$

proof–

define *f::real^7* **where** $f = \text{vector}[0, 0, 0, 0, 0, 1, 1]$

```

define  $g::\text{real}^7$  where  $g = \text{vector}[0, 0, 0, 1, 0, 0, 0]$ 
define  $h::\text{real}^7$  where  $h = \text{vector}[1, 0, 1, 0, 0, 0, 0]$ 
define  $u$  where  $u = (g \times_7 h)$ 
define  $v$  where  $v = (f \times_7 g)$ 

have 1:  $u = \text{vector}[0, 1, 0, 0, 0, -1, 0]$ 
  unfolding  $\text{cross7-def } g\text{-def } h\text{-def } f\text{-def } u\text{-def}$  by  $\text{simp}$ 
have 3:  $f \times_7 u = \text{vector}[0, 1, 0, 0, 0, 1, -1]$ 
  unfolding  $\text{cross7-def } f\text{-def}$  using 1 by  $\text{simp}$ 

have 2:  $v = \text{vector}[0, 0, -1, 0, 1, 0, 0]$ 
  unfolding  $\text{cross7-def } g\text{-def } h\text{-def } f\text{-def } v\text{-def}$  by  $\text{simp}$ 
have 4:  $v \times_7 h = \text{vector}[0, -1, 0, 0, 0, -1, 1]$ 
  unfolding  $\text{cross7-def } h\text{-def}$  using 2 by  $\text{simp}$ 

define  $x::\text{real}^7$  where  $x = \text{vector}[0, 1, 0, 0, 0, 1, -1]$ 
define  $y::\text{real}^7$  where  $y = \text{vector}[0, -1, 0, 0, 0, -1, 1]$ 

  have *:  $x \neq y$  unfolding  $x\text{-def}$  by  $\text{simp}$ 
  have **:  $y \neq x$  unfolding  $y\text{-def}$  by  $\text{simp}$ 

  have ***:  $x \neq y$  using * ** by  $\text{auto}$ 
  have 5:  $f \times_7 u \neq v \times_7 h$ 
    unfolding  $x\text{-def } y\text{-def}$ 
    using ***
    by ( $\text{simp add: 3 4 } x\text{-def } y\text{-def}$ )

have 6:  $f \times_7 (g \times_7 h) \neq (f \times_7 g) \times_7 h$  using 5 by ( $\text{simp add: } u\text{-def } v\text{-def}$ )

  show ?thesis unfolding  $f\text{-def } g\text{-def } h\text{-def}$  using 6 by  $\text{blast}$ 
qed
show ?thesis using * by  $\text{blast}$ 
qed

```

The 7D cross product does not satisfy the Jacobi Identity (shown using a counterexample)

lemma cross7-not-Jacobi :

$$\neg (\forall (c::\text{real}^7) (a::\text{real}^7) (b::\text{real}^7) . (c \times_7 a) \times_7 b + (b \times_7 c) \times_7 a + (a \times_7 b) \times_7 c = 0)$$

proof–

$$\text{have } * : \exists (c::\text{real}^7) (a::\text{real}^7) (b::\text{real}^7) . (c \times_7 a) \times_7 b + (b \times_7 c) \times_7 a + (a \times_7 b) \times_7 c \neq 0$$

proof–

```

define  $f::\text{real}^7$  where  $f = \text{vector}[0, 0, 0, 0, 0, 1, 1]$ 
define  $g::\text{real}^7$  where  $g = \text{vector}[0, 0, 0, 1, 0, 0, 0]$ 
define  $h::\text{real}^7$  where  $h = \text{vector}[1, 0, 1, 0, 0, 0, 0]$ 
define  $u$  where  $u = (g \times_7 h)$ 

```

```

define v where  $v = (f \times_7 g)$ 
define w where  $w = (h \times_7 f)$ 

have 1:  $u = \text{vector}[0, 1, 0, 0, 0, -1, 0]$ 
  unfolding cross7-def g-def h-def f-def u-def by simp
have 3:  $u \times_7 f = \text{vector}[0, -1, 0, 0, 0, -1, 1]$ 
  unfolding cross7-def f-def using 1 by simp

have 2:  $v = \text{vector}[0, 0, -1, 0, 1, 0, 0]$ 
  unfolding cross7-def g-def h-def f-def v-def by simp
have 4:  $v \times_7 h = \text{vector}[0, -1, 0, 0, 0, -1, 1]$ 
  unfolding cross7-def h-def using 2 by simp
have 8:  $w = \text{vector}[1, 0, -1, -1, -1, 0, 0]$ 
  unfolding cross7-def h-def f-def w-def by simp
have 9:  $w \times_7 g = \text{vector}[0, -1, 0, 0, 0, -1, 1]$ 
  unfolding cross7-def h-def f-def w-def g-def apply simp done
have &:  $(u \times_7 f) \cdot (v \times_7 h) \cdot (w \times_7 g) = -3$  using 3 4 9 by simp
have &&:  $u \times_7 f + v \times_7 h + w \times_7 g \neq 0$  using &
  by (metis vector-add-component zero-index zero-neq-neg-numeral)

show ?thesis using && u-def v-def w-def by blast
qed

show ?thesis using * by blast
qed

```

The vector triple product property fulfilled for the 3D cross product does not hold for the 7D cross product. Shown below with a counterexample.

lemma *cross7-not-vectortriple*:

$\neg(\forall (c::\text{real}^7) (a::\text{real}^7) (b::\text{real}^7). (c \times_7 a) \times_7 b = (c \cdot b) *_R a - (c \cdot a) *_R b)$

proof–

have *: $\exists (c::\text{real}^7) (a::\text{real}^7) (b::\text{real}^7). (c \times_7 a) \times_7 b \neq (c \cdot b) *_R a - (c \cdot a) *_R b$

proof–

define *g*:: real^7 **where** $g = \text{vector}[0, 0, 0, 1, 0, 0, 0]$

define *h*:: real^7 **where** $h = \text{vector}[1, 0, 1, 0, 0, 0, 0]$

define *f*:: real^7 **where** $f = \text{vector}[0, 0, 0, 0, 0, 1, 1]$

define *u* **where** $u = (g \times_7 h)$

have 1: $u = \text{vector}[0, 1, 0, 0, 0, -1, 0]$

unfolding *cross7-def g-def h-def f-def u-def* **by** *simp*

have 2: $u \times_7 f = \text{vector}[0, -1, 0, 0, 0, -1, 1]$

unfolding *cross7-def f-def* **using** 1 **by** *simp*

have 3: $(g \cdot f) *_R h = 0$ **unfolding** *g-def f-def inner-vec-def*
by (*simp add: sum-7*)

have 4: $(g \cdot h) *_R f = 0$ **unfolding** *g-def h-def inner-vec-def*
by (*simp add: sum-7*)

have 5: $(g \cdot f) *_R h - (g \cdot h) *_R f = 0$

using 3 4 **by** *auto*

have 6: $u \times_7 f \neq 0$ **using 2**
 by (metis one-neq-zero vector-7(7) zero-index)

have 7: $(g \times_7 h) \times_7 f \neq 0$ **using 2 6 u-def by blast**
have 8: $(g \times_7 h) \times_7 f \neq (g \cdot f) *_R h - (g \cdot h) *_R f$
 using 5 7 by simp
 show ?thesis using 8 by auto
qed
 show ?thesis using * by auto
qed

lemma axis-nth-neq [simp]: $i \neq j \implies \text{axis } i \ x \ \$ \ j = 0$
 by (simp add: axis-def)

lemma cross7-basis:

$(\text{axis } 1 \ 1) \times_7 (\text{axis } 2 \ 1) = \text{axis } 4 \ 1 (\text{axis } 2 \ 1) \times_7 (\text{axis } 1 \ 1) = -(\text{axis } 4 \ 1)$
 $(\text{axis } 2 \ 1) \times_7 (\text{axis } 3 \ 1) = \text{axis } 5 \ 1 (\text{axis } 3 \ 1) \times_7 (\text{axis } 2 \ 1) = -(\text{axis } 5 \ 1)$
 $(\text{axis } 3 \ 1) \times_7 (\text{axis } 4 \ 1) = \text{axis } 6 \ 1 (\text{axis } 4 \ 1) \times_7 (\text{axis } 3 \ 1) = -(\text{axis } 6 \ 1)$
 $(\text{axis } 2 \ 1) \times_7 (\text{axis } 4 \ 1) = \text{axis } 1 \ 1 (\text{axis } 4 \ 1) \times_7 (\text{axis } 2 \ 1) = -(\text{axis } 1 \ 1)$
 $(\text{axis } 4 \ 1) \times_7 (\text{axis } 5 \ 1) = \text{axis } 7 \ 1 (\text{axis } 5 \ 1) \times_7 (\text{axis } 4 \ 1) = -(\text{axis } 7 \ 1)$
 $(\text{axis } 3 \ 1) \times_7 (\text{axis } 5 \ 1) = \text{axis } 2 \ 1 (\text{axis } 5 \ 1) \times_7 (\text{axis } 3 \ 1) = -(\text{axis } 2 \ 1)$
 $(\text{axis } 4 \ 1) \times_7 (\text{axis } 6 \ 1) = \text{axis } 3 \ 1 (\text{axis } 6 \ 1) \times_7 (\text{axis } 4 \ 1) = -(\text{axis } 3 \ 1)$
 $(\text{axis } 5 \ 1) \times_7 (\text{axis } 7 \ 1) = \text{axis } 4 \ 1 (\text{axis } 7 \ 1) \times_7 (\text{axis } 5 \ 1) = -(\text{axis } 4 \ 1)$
 $(\text{axis } 4 \ 1) \times_7 (\text{axis } 1 \ 1) = \text{axis } 2 \ 1 (\text{axis } 1 \ 1) \times_7 (\text{axis } 4 \ 1) = -(\text{axis } 2 \ 1)$
 $(\text{axis } 5 \ 1) \times_7 (\text{axis } 2 \ 1) = \text{axis } 3 \ 1 (\text{axis } 2 \ 1) \times_7 (\text{axis } 5 \ 1) = -(\text{axis } 3 \ 1)$
 $(\text{axis } 6 \ 1) \times_7 (\text{axis } 3 \ 1) = \text{axis } 4 \ 1 (\text{axis } 3 \ 1) \times_7 (\text{axis } 6 \ 1) = -(\text{axis } 4 \ 1)$
 $(\text{axis } 7 \ 1) \times_7 (\text{axis } 4 \ 1) = \text{axis } 5 \ 1 (\text{axis } 4 \ 1) \times_7 (\text{axis } 7 \ 1) = -(\text{axis } 5 \ 1)$
 $(\text{axis } 5 \ 1) \times_7 (\text{axis } 6 \ 1) = \text{axis } 1 \ 1 (\text{axis } 6 \ 1) \times_7 (\text{axis } 5 \ 1) = -(\text{axis } 1 \ 1)$
 $(\text{axis } 6 \ 1) \times_7 (\text{axis } 7 \ 1) = \text{axis } 2 \ 1 (\text{axis } 7 \ 1) \times_7 (\text{axis } 6 \ 1) = -(\text{axis } 2 \ 1)$
 $(\text{axis } 7 \ 1) \times_7 (\text{axis } 1 \ 1) = \text{axis } 3 \ 1 (\text{axis } 1 \ 1) \times_7 (\text{axis } 7 \ 1) = -(\text{axis } 3 \ 1)$
 $(\text{axis } 6 \ 1) \times_7 (\text{axis } 1 \ 1) = \text{axis } 5 \ 1 (\text{axis } 1 \ 1) \times_7 (\text{axis } 6 \ 1) = -(\text{axis } 5 \ 1)$
 $(\text{axis } 7 \ 1) \times_7 (\text{axis } 2 \ 1) = \text{axis } 6 \ 1 (\text{axis } 2 \ 1) \times_7 (\text{axis } 7 \ 1) = -(\text{axis } 6 \ 1)$
 $(\text{axis } 1 \ 1) \times_7 (\text{axis } 3 \ 1) = \text{axis } 7 \ 1 (\text{axis } 3 \ 1) \times_7 (\text{axis } 1 \ 1) = -(\text{axis } 7 \ 1)$
 $(\text{axis } 1 \ 1) \times_7 (\text{axis } 5 \ 1) = \text{axis } 6 \ 1 (\text{axis } 5 \ 1) \times_7 (\text{axis } 1 \ 1) = -(\text{axis } 6 \ 1)$
 $(\text{axis } 2 \ 1) \times_7 (\text{axis } 6 \ 1) = \text{axis } 7 \ 1 (\text{axis } 6 \ 1) \times_7 (\text{axis } 2 \ 1) = -(\text{axis } 7 \ 1)$
 $(\text{axis } 3 \ 1) \times_7 (\text{axis } 7 \ 1) = \text{axis } 1 \ 1 (\text{axis } 7 \ 1) \times_7 (\text{axis } 3 \ 1) = -(\text{axis } 1 \ 1)$
 by (simp-all add: vec-eq-iff forall-7 cross7-components)

lemma cross7-basis-zero:

$u=0 \implies (u \times_7 \text{axis } 1 \ 1 = 0) \wedge (u \times_7 \text{axis } 2 \ 1 = 0) \wedge (u \times_7 \text{axis } 3 \ 1 = 0)$
 $\wedge (u \times_7 \text{axis } 4 \ 1 = 0) \wedge (u \times_7 \text{axis } 5 \ 1 = 0) \wedge (u \times_7 \text{axis } 6 \ 1 = 0)$
 $\wedge (u \times_7 \text{axis } 7 \ 1 = 0)$
 by auto

lemma cross7-basis-nonzero:

$\neg (u \times_7 \text{axis } 1 \ 1 = 0) \vee \neg (u \times_7 \text{axis } 2 \ 1 = 0) \vee \neg (u \times_7 \text{axis } 3 \ 1 = 0)$
 $\vee \neg (u \times_7 \text{axis } 4 \ 1 = 0) \vee \neg (u \times_7 \text{axis } 5 \ 1 = 0) \vee \neg (u \times_7 \text{axis } 6 \ 1 = 0)$
 $\vee \neg (u \times_7 \text{axis } 7 \ 1 = 0) \implies u \neq 0$
 by auto

Pythagorean theorem/magnitude

lemma *norm-square-vec-eq*: $\text{norm } x \wedge 2 = (\sum_{i \in \text{UNIV}} x \$ i \wedge 2)$
by (*auto simp: norm-vec-def L2-set-def intro!: sum-nonneg*)

lemma *norm-cross7-dot-magnitude*: $(\text{norm } (x \times_7 y))^2 = (\text{norm } x)^2 * (\text{norm } y)^2 - (x \cdot y)^2$
unfolding *norm-square-vec-eq sum-7 cross7-components inner-vec-def real-norm-def inner-real-def*
by *algebra*

lemma *cross7-eq-0*: $x \times_7 y = 0 \longleftrightarrow \text{collinear } \{0, x, y\}$

proof –

have $x \times_7 y = 0 \longleftrightarrow \text{norm } (x \times_7 y) = 0$

by *simp*

also have $\dots \longleftrightarrow (\text{norm } x * \text{norm } y)^2 = (x \cdot y)^2$

using *norm-cross7-dot-magnitude [of x y]* **by** (*auto simp: power-mult-distrib*)

also have $\dots \longleftrightarrow |x \cdot y| = \text{norm } x * \text{norm } y$

using *power2-eq-iff*

by (*metis (mono-tags, hide-lams) abs-minus abs-norm-cancel abs-power2 norm-mult power-abs real-norm-def*)

also have $\dots \longleftrightarrow \text{collinear } \{0, x, y\}$

by (*rule norm-cauchy-schwarz-equal*)

finally show *?thesis* .

qed

lemma *cross7-eq-self*: $x \times_7 y = x \longleftrightarrow x = 0 \vee x \times_7 y = y \longleftrightarrow y = 0$

apply (*metis cross7-zero-left dot-cross7-self(1) inner-eq-zero-iff*)

apply (*metis cross7-zero-right dot-cross7-self(2) inner-eq-zero-iff*)

done

lemma *norm-and-cross7-eq-0*:

$x \cdot y = 0 \wedge x \times_7 y = 0 \longleftrightarrow x = 0 \vee y = 0$ (**is** *?lhs = ?rhs*)

proof

assume *?lhs*

then show *?rhs*

using *cross7-eq-0 norm-cauchy-schwarz-equal* **by** *force*

next

assume *?rhs*

then show *?lhs*

by *auto*

qed

lemma *bilinear-cross7*: *bilinear* (\times_7)

apply (*auto simp add: bilinear-def linear-def*)

apply *unfold-locales*

apply (*simp-all add: cross7-add-right cross7-mult-right cross7-add-left cross7-mult-left*)

done

1.3 Continuity

lemma *continuous-cross7*: $\llbracket \text{continuous } F f; \text{continuous } F g \rrbracket \implies \text{continuous } F (\lambda x. f x \times_7 g x)$
by (*subst continuous-componentwise*)(*auto intro!*: *continuous-intros simp: cross7-simps*)

lemma *continuous-on-cross*:
fixes $f :: 'a::t2\text{-space} \Rightarrow \text{real}^\gamma$
shows $\llbracket \text{continuous-on } S f; \text{continuous-on } S g \rrbracket \implies \text{continuous-on } S (\lambda x. f x \times_7 g x)$
by (*simp add: continuous-on-eq-continuous-within continuous-cross7*)

end

2 Theory of Octonions

theory *Octonions*
imports *Cross-Product-7*
begin

2.1 Basic definitions

As with the complex numbers, coinduction is convenient.

codatatype *octo* =
Octo (*Ree*: *real*) (*Im1*: *real*) (*Im2*: *real*) (*Im3*: *real*) (*Im4*: *real*)
(*Im5*: *real*) (*Im6*: *real*) (*Im7*: *real*)

lemma *octo-eqI* [*intro?*]:
 $\llbracket \text{Ree } x = \text{Ree } y; \text{Im1 } x = \text{Im1 } y; \text{Im2 } x = \text{Im2 } y; \text{Im3 } x = \text{Im3 } y;$
 $\text{Im4 } x = \text{Im4 } y; \text{Im5 } x = \text{Im5 } y; \text{Im6 } x = \text{Im6 } y; \text{Im7 } x = \text{Im7 } y \rrbracket \implies x = y$
by (*rule octo.expand*) *simp*

lemma *octo-eq-iff*:
 $x = y \longleftrightarrow \text{Ree } x = \text{Ree } y \wedge \text{Im1 } x = \text{Im1 } y \wedge \text{Im2 } x = \text{Im2 } y \wedge \text{Im3 } x = \text{Im3 } y$
 \wedge
 $\text{Im4 } x = \text{Im4 } y \wedge \text{Im5 } x = \text{Im5 } y \wedge \text{Im6 } x = \text{Im6 } y \wedge \text{Im7 } x = \text{Im7 } y$
by (*auto intro: octo.expand*)

context
begin

primcorec *octo-e0* :: *octo* (*e0*)
where $\text{Ree } e0 = 1 \mid \text{Im1 } e0 = 0 \mid \text{Im2 } e0 = 0 \mid \text{Im3 } e0 = 0$
 $\mid \text{Im4 } e0 = 0 \mid \text{Im5 } e0 = 0 \mid \text{Im6 } e0 = 0 \mid \text{Im7 } e0 = 0$

primcorec *octo-e1* :: *octo* (*e1*)
where $\text{Ree } e1 = 0 \mid \text{Im1 } e1 = 1 \mid \text{Im2 } e1 = 0 \mid \text{Im3 } e1 = 0$
 $\mid \text{Im4 } e1 = 0 \mid \text{Im5 } e1 = 0 \mid \text{Im6 } e1 = 0 \mid \text{Im7 } e1 = 0$

```

primcorec octo-e2 :: octo (e2)
  where Ree e2 = 0 | Im1 e2 = 0 | Im2 e2 = 1 | Im3 e2 = 0
    | Im4 e2 = 0 | Im5 e2 = 0 | Im6 e2 = 0 | Im7 e2 = 0

primcorec octo-e3 :: octo (e3)
  where Ree e3 = 0 | Im1 e3 = 0 | Im2 e3 = 0 | Im3 e3 = 1
    | Im4 e3 = 0 | Im5 e3 = 0 | Im6 e3 = 0 | Im7 e3 = 0

primcorec octo-e4 :: octo (e4)
  where Ree e4 = 0 | Im1 e4 = 0 | Im2 e4 = 0 | Im3 e4 = 0
    | Im4 e4 = 1 | Im5 e4 = 0 | Im6 e4 = 0 | Im7 e4 = 0

primcorec octo-e5 :: octo (e5)
  where Ree e5 = 0 | Im1 e5 = 0 | Im2 e5 = 0 | Im3 e5 = 0
    | Im4 e5 = 0 | Im5 e5 = 1 | Im6 e5 = 0 | Im7 e5 = 0

primcorec octo-e6 :: octo (e6)
  where Ree e6 = 0 | Im1 e6 = 0 | Im2 e6 = 0 | Im3 e6 = 0
    | Im4 e6 = 0 | Im5 e6 = 0 | Im6 e6 = 1 | Im7 e6 = 0

primcorec octo-e7 :: octo (e7)
  where Ree e7 = 0 | Im1 e7 = 0 | Im2 e7 = 0 | Im3 e7 = 0
    | Im4 e7 = 0 | Im5 e7 = 0 | Im6 e7 = 0 | Im7 e7 = 1
end

```

2.2 Addition and Subtraction: An Abelian Group

```

instantiation octo :: ab-group-add

```

```

begin

```

```

primcorec zero-octo
  where Ree 0 = 0 | Im1 0 = 0 | Im2 0 = 0 | Im3 0 = 0
    | Im4 0 = 0 | Im5 0 = 0 | Im6 0 = 0 | Im7 0 = 0

```

```

primcorec plus-octo
  where
    Ree (x + y) = Ree x + Ree y
    | Im1 (x + y) = Im1 x + Im1 y
    | Im2 (x + y) = Im2 x + Im2 y
    | Im3 (x + y) = Im3 x + Im3 y
    | Im4 (x + y) = Im4 x + Im4 y
    | Im5 (x + y) = Im5 x + Im5 y
    | Im6 (x + y) = Im6 x + Im6 y
    | Im7 (x + y) = Im7 x + Im7 y

```

```

primcorec uminus-octo
  where
    Ree (- x) = - Ree x

```

$| \text{Im1 } (-x) = - \text{Im1 } x$
 $| \text{Im2 } (-x) = - \text{Im2 } x$
 $| \text{Im3 } (-x) = - \text{Im3 } x$
 $| \text{Im4 } (-x) = - \text{Im4 } x$
 $| \text{Im5 } (-x) = - \text{Im5 } x$
 $| \text{Im6 } (-x) = - \text{Im6 } x$
 $| \text{Im7 } (-x) = - \text{Im7 } x$

primcorec *minus-octo*

where

$\text{Ree } (x - y) = \text{Ree } x - \text{Ree } y$
 $| \text{Im1 } (x - y) = \text{Im1 } x - \text{Im1 } y$
 $| \text{Im2 } (x - y) = \text{Im2 } x - \text{Im2 } y$
 $| \text{Im3 } (x - y) = \text{Im3 } x - \text{Im3 } y$
 $| \text{Im4 } (x - y) = \text{Im4 } x - \text{Im4 } y$
 $| \text{Im5 } (x - y) = \text{Im5 } x - \text{Im5 } y$
 $| \text{Im6 } (x - y) = \text{Im6 } x - \text{Im6 } y$
 $| \text{Im7 } (x - y) = \text{Im7 } x - \text{Im7 } y$

instance

by *standard (simp-all add: octo-eq-iff)*

end

lemma *octo-eq-0-iff*:

$x = 0 \iff \text{Ree } x^2 + \text{Im1 } x^2 + \text{Im2 } x^2 + \text{Im3 } x^2 +$
 $\text{Im4 } x^2 + \text{Im5 } x^2 + \text{Im6 } x^2 + \text{Im7 } x^2 = 0$

proof

assume $(\text{octo.Ree } x)^2 + (\text{Im1 } x)^2 + (\text{Im2 } x)^2 + (\text{Im3 } x)^2 + (\text{Im4 } x)^2 + (\text{Im5 } x)^2 + (\text{Im6 } x)^2 + (\text{Im7 } x)^2 = 0$

then have $\forall qa. qa - x = qa$

by *(simp add: add-nonneg-eq-0-iff minus-octo.ctr)*

then show $x = 0$

by *simp*

qed *auto*

2.3 A Normed Vector Space

instantiation *octo :: real-vector*

begin

primcorec *scaleR-octo*

where

$\text{Ree } (\text{scaleR } r x) = r * \text{Ree } x$
 $| \text{Im1 } (\text{scaleR } r x) = r * \text{Im1 } x$
 $| \text{Im2 } (\text{scaleR } r x) = r * \text{Im2 } x$
 $| \text{Im3 } (\text{scaleR } r x) = r * \text{Im3 } x$

| $Im4 (scaleR r x) = r * Im4 x$
| $Im5 (scaleR r x) = r * Im5 x$
| $Im6 (scaleR r x) = r * Im6 x$
| $Im7 (scaleR r x) = r * Im7 x$

instance

by *standard* (*auto simp: octo-eq-iff distrib-left distrib-right scaleR-add-right*)

end

instantiation *octo::one*

begin

primcorec *one-octo*

where

$Ree 1 = 1$ | $Im1 1 = 0$ | $Im2 1 = 0$ | $Im3 1 = 0$ |
 $Im4 1 = 0$ | $Im5 1 = 0$ | $Im6 1 = 0$ | $Im7 1 = 0$

instance by *standard*

end

fun *octo-proj*

where

$octo-proj x 0 = (Ree (x))$
| $octo-proj x (Suc 0) = (Im1(x))$
| $octo-proj x (Suc (Suc 0)) = (Im2 (x))$
| $octo-proj x (Suc (Suc (Suc 0))) = (Im3 (x))$
| $octo-proj x (Suc (Suc (Suc (Suc 0)))) = (Im4 (x))$
| $octo-proj x (Suc(Suc (Suc (Suc (Suc 0)))))) = (Im5 (x))$
| $octo-proj x (Suc(Suc (Suc (Suc (Suc (Suc 0)))))) = (Im6 (x))$
| $octo-proj x (Suc(Suc(Suc(Suc (Suc (Suc (Suc 0)))))) = (Im7 (x))$

lemma *octo-proj-add:*

assumes $i \leq 7$

shows $octo-proj (x+y) i = octo-proj x i + octo-proj y i$

proof –

consider $i = 0$ | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | $i = 6$ | $i = 7$

using *assms* **by** *force*

then show *?thesis*

by cases (*auto simp: numeral-2-eq-2 numeral-3-eq-3 numeral-4-eq-4 numeral-5-eq-5*

numeral-6-eq-6 numeral-7-eq-7 numeral-7-eq-7)

qed

instantiation *octo ::real-normed-vector*

begin

definition $norm\ x = \text{sqrt } ((\text{Ree } x)^2 + (\text{Im1 } x)^2 + (\text{Im2 } x)^2 + (\text{Im3 } x)^2 + (\text{Im4 } x)^2 + (\text{Im5 } x)^2 + (\text{Im6 } x)^2 + (\text{Im7 } x)^2)$ **for** $x::\text{octo}$

definition $\text{sgn } x = x /_R \text{ norm } x$ **for** $x :: \text{octo}$

definition $\text{dist } x\ y = \text{norm } (x - y)$ **for** $x\ y :: \text{octo}$

definition [code del]:

$(\text{uniformity} :: (\text{octo} \times \text{octo}) \text{ filter}) = (\text{INF } e \in \{0 < ..\}. \text{principal } \{(x, y). \text{dist } x\ y < e\})$

definition [code del]:

$\text{open } (U :: \text{octo set}) \longleftrightarrow (\forall x \in U. \text{eventually } (\lambda(x', y). x' = x \longrightarrow y \in U) \text{ uniformity})$

lemma norm-eq-L2 : $\text{norm } x = \text{L2-set } (\text{octo-proj } x) \{..7\}$

by ($\text{simp add: norm-octo-def L2-set-def eval-nat-numeral}$)

instance proof

fix $r :: \text{real}$ **and** $x\ y :: \text{octo}$ **and** $S :: \text{octo set}$

show $(\text{norm } x = 0) \longleftrightarrow (x = 0)$

by ($\text{simp add: norm-octo-def octo-eq-iff add-nonneg-eq-0-iff}$)

have $\text{eq: L2-set } (\text{octo-proj } (x + y)) \{..7\} = \text{L2-set } (\lambda i. \text{octo-proj } x\ i + \text{octo-proj } y\ i) \{..7\}$

by (rule L2-set-cong) ($\text{auto simp: octo-proj-add}$)

show $\text{norm } (x + y) \leq \text{norm } x + \text{norm } y$

by ($\text{simp add: norm-eq-L2 eq L2-set-triangle-ineq}$)

show $\text{norm } (\text{scaleR } r\ x) = |r| * \text{norm } x$

by ($\text{simp add: norm-octo-def octo-eq-iff}$)

$\text{power-mult-distrib distrib-left [symmetric] real-sqrt-mult}$)

qed ($\text{rule sgn-octo-def dist-octo-def open-octo-def uniformity-octo-def}$)

end

lemma norm-octo-squared :

$\text{norm } x ^ 2 = \text{Ree } x ^ 2 + \text{Im1 } x ^ 2 + \text{Im2 } x ^ 2 + \text{Im3 } x ^ 2 +$

$\text{Im4 } x ^ 2 + \text{Im5 } x ^ 2 + \text{Im6 } x ^ 2 + \text{Im7 } x ^ 2$

by ($\text{simp add: norm-octo-def}$)

instantiation $\text{octo} :: \text{real-inner}$

begin

definition inner-octo **where**

$\text{inner-octo } x\ y = \text{Ree } x * \text{Ree } y + \text{Im1 } x * \text{Im1 } y + \text{Im2 } x * \text{Im2 } y + \text{Im3 } x * \text{Im3 } y$

$+ \text{Im4 } x * \text{Im4 } y + \text{Im5 } x * \text{Im5 } y + \text{Im6 } x * \text{Im6 } y + \text{Im7 } x * \text{Im7 } y$ **for** $x\ y::\text{octo}$

instance

by *standard* (*auto simp: inner-octo-def algebra-simps norm-octo-def*
power2-eq-square octo-eq-iff add-nonneg-eq-0-iff)

end

lemma *octo-inner-1* [*simp*]: *inner 1 x = Re x*
and *octo-inner-1-right* [*simp*]: *inner x 1 = Re x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e1-left* [*simp*]: *inner e1 x = Im1 x*
and *octo-inner-e1-right* [*simp*]: *inner x e1 = Im1 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e2-left* [*simp*]: *inner e2 x = Im2 x*
and *octo-inner-e2-right* [*simp*]: *inner x e2 = Im2 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e3-left* [*simp*]: *inner e3 x = Im3 x*
and *octo-inner-e3-right* [*simp*]: *inner x e3 = Im3 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e4-left* [*simp*]: *inner e4 x = Im4 x*
and *octo-inner-e4-right* [*simp*]: *inner x e4 = Im4 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e5-left* [*simp*]: *inner e5 x = Im5 x*
and *octo-inner-e5-right* [*simp*]: *inner x e5 = Im5 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e6-left* [*simp*]: *inner e6 x = Im6 x*
and *octo-inner-e6-right* [*simp*]: *inner x e6 = Im6 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-inner-e7-left* [*simp*]: *inner e7 x = Im7 x*
and *octo-inner-e7-right* [*simp*]: *inner x e7 = Im7 x*
unfolding *inner-octo-def* **by** *simp-all*

lemma *octo-norm-pow-2-inner*: $(\text{norm } x)^2 = \text{inner } x x$ **for** $x :: \text{octo}$
by (*simp add: dot-square-norm*)

lemma *octo-norm-property*:
inner x y = (1/2) ((norm(x+y))^2 - (norm(x))^2 - (norm(y))^2)* **for** $x y$
 $:: \text{octo}$
by (*simp add: dot-norm norm-octo-def*)

2.4 The Octonionic product and related properties and lemmas

The multiplication is defined following one of the 480 equivalent multiplication tables in an analogy to the definition of the 7D cross product.

instantiation *octo* :: *times*
begin

definition *times-octo* :: [*octo*, *octo*] \Rightarrow *octo*

where

(*a* * *b*) = (let
t0 = *Ree a* * *Ree b* - *Im1 a* * *Im1 b* - *Im2 a* * *Im2 b* - *Im3 a* * *Im3 b*
- *Im4 a* * *Im4 b* - *Im5 a* * *Im5 b* - *Im6 a* * *Im6 b* - *Im7 a* * *Im7 b* ;
t1 = *Ree a* * *Im1 b* + *Im1 a* * *Ree b* + *Im2 a* * *Im4 b* + *Im3 a* * *Im7 b* -
Im4 a * *Im2 b* + *Im5 a* * *Im6 b* - *Im6 a* * *Im5 b* - *Im7 a* * *Im3 b* ;
t2 = *Ree a* * *Im2 b* - *Im1 a* * *Im4 b* + *Im2 a* * *Ree b* + *Im3 a* * *Im5 b*
+ *Im4 a* * *Im1 b* - *Im5 a* * *Im3 b* + *Im6 a* * *Im7 b* - *Im7 a* * *Im6 b* ;
t3 = *Ree a* * *Im3 b* - *Im1 a* * *Im7 b* - *Im2 a* * *Im5 b* + *Im3 a* * *Ree b* + *Im4 a*
* *Im6 b*
+ *Im5 a* * *Im2 b* - *Im6 a* * *Im4 b* + *Im7 a* * *Im1 b* ;
t4 = *Ree a* * *Im4 b* + *Im1 a* * *Im2 b* - *Im2 a* * *Im1 b* - *Im3 a* * *Im6 b* + *Im4 a*
* *Ree b*
+ *Im5 a* * *Im7 b* + *Im6 a* * *Im3 b* - *Im7 a* * *Im5 b* ;
t5 = *Ree a* * *Im5 b* - *Im1 a* * *Im6 b* + *Im2 a* * *Im3 b* - *Im3 a* * *Im2 b* - *Im4 a*
* *Im7 b*
+ *Im5 a* * *Ree b* + *Im6 a* * *Im1 b* + *Im7 a* * *Im4 b* ;
t6 = *Ree a* * *Im6 b* + *Im1 a* * *Im5 b* - *Im2 a* * *Im7 b* + *Im3 a* * *Im4 b* - *Im4 a*
* *Im3 b*
- *Im5 a* * *Im1 b* + *Im6 a* * *Ree b* + *Im7 a* * *Im2 b* ;
t7 = *Ree a* * *Im7 b* + *Im1 a* * *Im3 b* + *Im2 a* * *Im6 b* - *Im3 a* * *Im1 b* + *Im4 a*
* *Im5 b*
- *Im5 a* * *Im4 b* - *Im6 a* * *Im2 b* + *Im7 a* * *Ree b*
in *Octo t0 t1 t2 t3 t4 t5 t6 t7*)

instance by *standard*

end

instantiation *octo* :: *inverse*

begin

primcorec *inverse-octo*

where

Ree (inverse x) = *Ree x* / (*Ree x* ² + *Im1 x* ² + *Im2 x* ² + *Im3 x* ²
+ *Im4 x* ² + *Im5 x* ² + *Im6 x* ² + *Im7 x* ²)
| *Im1 (inverse x)* = - (*Im1 x*) / (*Ree x* ² + *Im1 x* ² + *Im2 x* ² + *Im3 x*
²
+ *Im4 x* ² + *Im5 x* ² + *Im6 x* ² + *Im7 x* ²)
| *Im2 (inverse x)* = - (*Im2 x*) / (*Ree x* ² + *Im1 x* ² + *Im2 x* ² + *Im3 x*

$$\begin{aligned}
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2}) \\
& | \text{Im}3 (\text{inverse } x) = - (\text{Im}3 x) / (\text{Ree } x^{\wedge 2} + \text{Im}1 x^{\wedge 2} + \text{Im}2 x^{\wedge 2} + \text{Im}3 x \\
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2}) \\
& | \text{Im}4 (\text{inverse } x) = - (\text{Im}4 x) / (\text{Ree } x^{\wedge 2} + \text{Im}1 x^{\wedge 2} + \text{Im}2 x^{\wedge 2} + \text{Im}3 x \\
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2}) \\
& | \text{Im}5 (\text{inverse } x) = - (\text{Im}5 x) / (\text{Ree } x^{\wedge 2} + \text{Im}1 x^{\wedge 2} + \text{Im}2 x^{\wedge 2} + \text{Im}3 x \\
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2}) \\
& | \text{Im}6 (\text{inverse } x) = - (\text{Im}6 x) / (\text{Ree } x^{\wedge 2} + \text{Im}1 x^{\wedge 2} + \text{Im}2 x^{\wedge 2} + \text{Im}3 x \\
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2}) \\
& | \text{Im}7 (\text{inverse } x) = - (\text{Im}7 x) / (\text{Ree } x^{\wedge 2} + \text{Im}1 x^{\wedge 2} + \text{Im}2 x^{\wedge 2} + \text{Im}3 x \\
& \wedge^2 \\
& + \text{Im}4 x^{\wedge 2} + \text{Im}5 x^{\wedge 2} + \text{Im}6 x^{\wedge 2} + \text{Im}7 x^{\wedge 2})
\end{aligned}$$

definition $x \text{ div } y = x * (\text{inverse } y)$ for $x y :: \text{octo}$

instance by standard

end

lemma *octo-mult-components:*

$$\begin{aligned}
& \text{Ree } (x * y) = \text{Ree } x * \text{Ree } y - \text{Im}1 x * \text{Im}1 y - \text{Im}2 x * \text{Im}2 y - \text{Im}3 x * \\
& \text{Im}3 y \\
& - \text{Im}4 x * \text{Im}4 y - \text{Im}5 x * \text{Im}5 y - \text{Im}6 x * \text{Im}6 y - \text{Im}7 x * \text{Im}7 y \\
& \text{Im}1 (x * y) = \text{Ree } x * \text{Im}1 y + \text{Im}1 x * \text{Ree } y + \text{Im}2 x * \text{Im}4 y + \text{Im}3 x * \\
& \text{Im}7 y - \\
& \text{Im}4 x * \text{Im}2 y + \text{Im}5 x * \text{Im}6 y - \text{Im}6 x * \text{Im}5 y - \text{Im}7 x * \text{Im}3 y \\
& \text{Im}2 (x * y) = \text{Ree } x * \text{Im}2 y - \text{Im}1 x * \text{Im}4 y + \text{Im}2 x * \text{Ree } y + \text{Im}3 x \\
& * \text{Im}5 y \\
& + \text{Im}4 x * \text{Im}1 y - \text{Im}5 x * \text{Im}3 y + \text{Im}6 x * \text{Im}7 y - \text{Im}7 x * \text{Im}6 y \\
& \text{Im}3 (x * y) = \text{Ree } x * \text{Im}3 y - \text{Im}1 x * \text{Im}7 y - \text{Im}2 x * \text{Im}5 y + \text{Im}3 x * \text{Ree } \\
& y + \text{Im}4 x * \text{Im}6 y \\
& + \text{Im}5 x * \text{Im}2 y - \text{Im}6 x * \text{Im}4 y + \text{Im}7 x * \text{Im}1 y \\
& \text{Im}4 (x * y) = \text{Ree } x * \text{Im}4 y + \text{Im}1 x * \text{Im}2 y - \text{Im}2 x * \text{Im}1 y - \text{Im}3 x * \\
& \text{Im}6 y + \text{Im}4 x * \text{Ree } y \\
& + \text{Im}5 x * \text{Im}7 y + \text{Im}6 x * \text{Im}3 y - \text{Im}7 x * \text{Im}5 y \\
& \text{Im}5 (x * y) = \text{Ree } x * \text{Im}5 y - \text{Im}1 x * \text{Im}6 y + \text{Im}2 x * \text{Im}3 y - \text{Im}3 x * \\
& \text{Im}2 y - \text{Im}4 x * \text{Im}7 y \\
& + \text{Im}5 x * \text{Ree } y + \text{Im}6 x * \text{Im}1 y + \text{Im}7 x * \text{Im}4 y \\
& \text{Im}6 (x * y) = \text{Ree } x * \text{Im}6 y + \text{Im}1 x * \text{Im}5 y - \text{Im}2 x * \text{Im}7 y + \text{Im}3 x * \\
& \text{Im}4 y - \text{Im}4 x * \text{Im}3 y \\
& - \text{Im}5 x * \text{Im}1 y + \text{Im}6 x * \text{Ree } y + \text{Im}7 x * \text{Im}2 y \\
& \text{Im}7 (x * y) = \text{Ree } x * \text{Im}7 y + \text{Im}1 x * \text{Im}3 y + \text{Im}2 x * \text{Im}6 y - \text{Im}3 x * \\
& \text{Im}1 y + \text{Im}4 x * \text{Im}5 y
\end{aligned}$$

$-Im5\ x * Im4\ y - Im6\ x * Im2\ y + Im7\ x * Re\ y$
unfolding *times-octo-def* **by** *auto*

lemma *octo-distrib-left* :

$a * (b + c) = a * b + a * c$ **for** $a\ b\ c :: octo$

unfolding *times-octo-def plus-octo-def minus-octo-def uminus-octo-def scaleR-octo-def*
by (*simp add: octo-eq-iff octo-mult-components algebra-simps*)

lemma *octo-distrib-right* :

$(b + c) * a = b * a + c * a$ **for** $a\ b\ c :: octo$

unfolding *times-octo-def plus-octo-def minus-octo-def uminus-octo-def scaleR-octo-def*
by (*simp add: octo-eq-iff octo-mult-components algebra-simps*)

lemma *multiplicative-norm-octo*: $norm\ (x * y) = norm\ x * norm\ y$ **for** $x\ y :: octo$

proof –

have $norm\ (x * y) ^ 2 = norm\ x ^ 2 * norm\ y ^ 2$

unfolding *norm-octo-squared octo-mult-components* **by** *algebra*

also have $... = (norm\ x * norm\ y) ^ 2$

by (*simp add: power-mult-distrib*)

finally show *?thesis* **by** *simp*

qed

lemma *mult-1-right-octo* [*simp*]: $x * 1 = (x :: octo)$

and *mult-1-left-octo* [*simp*]: $1 * x = (x :: octo)$

by (*simp-all add: times-octo-def*)

instance *octo* :: *power ..*

lemma *power2-eq-square-octo*: $x ^ 2 = (x * x :: octo)$

by (*simp add: numeral-2-eq-2 times-octo-def*)

lemma *octo-product-alternative-left*: $x * (x * y) = (x * x :: octo) * y$

unfolding *octo-eq-iff octo-mult-components* **by** *algebra*

lemma *octo-product-alternative-right*: $x * (y * y) = (x * y :: octo) * y$

unfolding *octo-eq-iff octo-mult-components* **by** *algebra*

lemma *octo-product-flexible*: $(x * y) * x = x * (y * x :: octo)$

unfolding *octo-eq-iff octo-mult-components* **by** *algebra*

lemma *octo-power-commutes*: $x ^ y * x = x * (x ^ y :: octo)$

by (*induction y*) (*simp-all add: octo-product-flexible*)

lemma *octo-product-noncommutative*: $\neg(\forall x\ y :: octo. (x * y = y * x))$

using *inverse-1 left-inverse mult-not-zero octo.sel(8) octo-e4.sel(2)*

octo-e4.sel(3) octo-e4.sel(5) octo-e5.sel(1) octo-e5.sel(2)

octo-e5.sel(3) octo-e5.sel(5) octo-e5.sel(6) octo-e5.sel(8) times-octo-def

by *smt*

lemma *octo-product-nonassociative* :

$\neg(\forall x y z::\text{octo}. x * (y * z) = (x * y) * z)$

proof–

define *x* **where** $x = \text{Octo } 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0$

define *y* **where** $y = \text{Octo } 1\ 3\ 0\ 0\ 0\ 1\ 0\ 0$

define *z* **where** $z = \text{Octo } 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0$

have $x * (y * z) \neq (x * y) * z$

by (*simp* *add: octo-eq-iff octo-mult-components x-def y-def z-def*)

thus *?thesis* **by** *blast*

qed

2.5 Embedding of the Reals into the Octonions

definition *octo-of-real* $:: \text{real} \Rightarrow \text{octo}$

where *octo-of-real* $r = \text{scaleR } r\ 1$

definition *octo-of-nat* $:: \text{nat} \Rightarrow \text{octo}$

where *octo-of-nat* $r = \text{scaleR } r\ 1$

definition *octo-of-int* $:: \text{int} \Rightarrow \text{octo}$

where *octo-of-int* $r = \text{scaleR } r\ 1$

lemma *octo-of-nat-sel* [*simp*]:

Ree (*octo-of-nat* x) = *of-nat* x *Im1* (*octo-of-nat* x) = 0 *Im2* (*octo-of-nat* x) = 0

Im3 (*octo-of-nat* x) = 0 *Im4* (*octo-of-nat* x) = 0 *Im5* (*octo-of-nat* x) = 0

Im6 (*octo-of-nat* x) = 0 *Im7* (*octo-of-nat* x) = 0

by (*simp-all* *add: octo-of-nat-def*)

lemma *octo-of-real-sel* [*simp*]:

Ree (*octo-of-real* x) = x *Im1* (*octo-of-real* x) = 0 *Im2* (*octo-of-real* x) = 0

Im3 (*octo-of-real* x) = 0 *Im4* (*octo-of-real* x) = 0 *Im5* (*octo-of-real* x) = 0

Im6 (*octo-of-real* x) = 0 *Im7* (*octo-of-real* x) = 0

by (*simp-all* *add: octo-of-real-def*)

lemma *octo-of-int-sel* [*simp*]:

Ree (*octo-of-int* x) = *of-int* x *Im1* (*octo-of-int* x) = 0 *Im2* (*octo-of-int* x) = 0

Im3 (*octo-of-int* x) = 0 *Im4* (*octo-of-int* x) = 0 *Im5* (*octo-of-int* x) = 0

Im6 (*octo-of-int* x) = 0 *Im7* (*octo-of-int* x) = 0

by (*simp-all* *add: octo-of-int-def*)

lemma *scaleR-conv-octo-of-real*: $\text{scaleR } r\ x = \text{octo-of-real } r * x$

by (*simp* *add: octo-eq-iff octo-mult-components octo-of-real-def*)

lemma *octo-of-real-0* [*simp*]: *octo-of-real* 0 = 0

by (*simp* *add: octo-of-real-def*)

lemma *octo-of-real-1* [*simp*]: *octo-of-real* 1 = 1

by (*simp* *add: octo-of-real-def*)

lemma *octo-of-real-add* [simp]: $\text{octo-of-real } (x + y) = \text{octo-of-real } x + \text{octo-of-real } y$
by (*simp add: octo-of-real-def scaleR-left-distrib*)

lemma *octo-of-real-minus* [simp]: $\text{octo-of-real } (- x) = - \text{octo-of-real } x$
by (*simp add: octo-of-real-def*)

lemma *octo-of-real-diff* [simp]: $\text{octo-of-real } (x - y) = \text{octo-of-real } x - \text{octo-of-real } y$
by (*simp add: octo-of-real-def scaleR-left-diff-distrib*)

lemma *octo-of-real-mult* [simp]: $\text{octo-of-real } (x * y) = \text{octo-of-real } x * \text{octo-of-real } y$
using *octo-of-real-def*
by (*metis scaleR-conv-octo-of-real scaleR-scaleR*)

lemma *octo-of-real-sum*[simp]: $\text{octo-of-real } (\text{sum } f s) = (\sum x \in s. \text{octo-of-real } (f x))$
by (*induct s rule: infinite-finite-induct*) *auto*

lemma *octo-of-real-power* [simp]:
 $\text{octo-of-real } (x \wedge y) = (\text{octo-of-real } x :: \text{octo}) \wedge y$
by (*induct y*)(*simp-all*)

lemma *octo-of-real-eq-iff* [simp]: $\text{octo-of-real } x = \text{octo-of-real } y \longleftrightarrow x = y$
using *octo-of-real-def*
by (*simp add: octo-of-real-def one-octo.code zero-octo.code*)

lemmas *octo-of-real-eq-0-iff* [simp] = *octo-of-real-eq-iff* [*of - 0, simplified*]
lemmas *octo-of-real-eq-1-iff* [simp] = *octo-of-real-eq-iff* [*of - 1, simplified*]

lemma *minus-octo-of-real-eq-octo-of-real-iff* [simp]: $- \text{octo-of-real } x = \text{octo-of-real } y \longleftrightarrow -x = y$
using *octo-of-real-eq-iff*[*of -x y*] **by** (*simp only: octo-of-real-minus*)

lemma *octo-of-real-eq-minus-of-real-iff* [simp]: $\text{octo-of-real } x = - \text{octo-of-real } y \longleftrightarrow x = -y$
using *octo-of-real-eq-iff*[*of x -y*] **by** (*simp only: octo-of-real-minus*)

lemma *octo-of-real-of-nat-eq* [simp]: $\text{octo-of-real } (\text{of-nat } x) = \text{octo-of-nat } x$
unfolding *octo-of-real-def*
by (*simp add: octo-of-nat-def*)

lemma *octo-of-real-of-int-eq* [simp]: $\text{octo-of-real } (\text{of-int } z) = \text{octo-of-int } z$
unfolding *octo-of-real-def*
by (*simp add: octo-of-int-def*)

lemma *octo-of-int-of-nat*: $\text{octo-of-int } (\text{of-nat } n) = \text{octo-of-nat } n$
by (*simp add: octo-eq-iff*)

lemma *octo-of-nat-add* [*simp*]: $octo-of-nat (a + b) = octo-of-nat a + octo-of-nat b$
and *octo-of-nat-mult* [*simp*]: $octo-of-nat (a * b) = octo-of-nat a * octo-of-nat b$
and *octo-of-nat-diff* [*simp*]: $b \leq a \implies octo-of-nat (a - b) = octo-of-nat a - octo-of-nat b$
and *octo-of-nat-0* [*simp*]: $octo-of-nat 0 = 0$
and *octo-of-nat-1* [*simp*]: $octo-of-nat 1 = 1$
and *octo-of-nat-Suc-0* [*simp*]: $octo-of-nat (Suc 0) = 1$
by (*simp-all add: octo-eq-iff octo-mult-components*)

lemma *octo-of-int-add* [*simp*]: $octo-of-int (a + b) = octo-of-int a + octo-of-int b$
and *octo-of-int-mult* [*simp*]: $octo-of-int (a * b) = octo-of-int a * octo-of-int b$
and *octo-of-int-diff* [*simp*]: $b \leq a \implies octo-of-int (a - b) = octo-of-int a - octo-of-int b$
and *octo-of-int-0* [*simp*]: $octo-of-int 0 = 0$
and *octo-of-int-1* [*simp*]: $octo-of-int 1 = 1$
by (*simp-all add: octo-eq-iff octo-mult-components*)

instance *octo* :: *numeral* ..

lemma *numeral-octo-conv-of-nat*: $numeral x = octo-of-nat (numeral x)$

proof (*induction x*)

case (*Bit0 x*)

have $numeral x + numeral x = octo-of-nat (numeral x + numeral x)$

unfolding *Bit0.IH octo-of-nat-add* ..

thus *?case* **by** (*simp add: numeral-Bit0*)

next

case (*Bit1 x*)

have $numeral x + numeral x + numeral num.One =$

$octo-of-nat (numeral x + numeral x + numeral num.One)$

unfolding *Bit1.IH octo-of-nat-add* **by** *simp*

thus *?case* **by** (*simp add: numeral-Bit1*)

qed *auto*

lemma *numeral-octo-sel* [*simp*]:

$Ree (numeral n) = numeral n$ $Im1 (numeral n) = 0$ $Im2 (numeral n) = 0$

$Im3 (numeral n) = 0$ $Im4 (numeral n) = 0$ $Im5 (numeral n) = 0$

$Im6 (numeral n) = 0$ $Im7 (numeral n) = 0$

by (*simp-all add: numeral-octo-conv-of-nat*)

lemma *octo-of-real-numeral* [*simp*]: $octo-of-real (numeral w) = numeral w$

by (*simp add: numeral-octo-conv-of-nat octo-of-real-def octo-of-nat-def*)

lemma *octo-of-real-neg-numeral* [*simp*]: $octo-of-real (- numeral w) = - numeral w$

by *simp*

lemma *octo-of-real-times-commute*: $octo-of-real r * q = q * octo-of-real r$

using *octo-of-real-def times-octo-def* **by** *simp*

lemma *octo-of-real-times-conv-scaleR*: $octo\text{-of-real } x * y = scaleR x y$
by (*simp add: octo-eq-iff octo-mult-components*)

lemma *octo-mult-scaleR-left*: $(r *_R x) * y = r *_R (x * y :: octo)$
by (*simp add: octo-eq-iff octo-mult-components algebra-simps*)

lemma *octo-mult-scaleR-right*: $x * (r *_R y) = r *_R (x * y :: octo)$
by (*simp add: octo-eq-iff octo-mult-components algebra-simps*)

lemma *scaleR-octo-of-real [simp]*: $scaleR r (octo\text{-of-real } s) = octo\text{-of-real } (r * s)$
by (*simp add: octo-of-real-def*)

lemma *octo-of-real-times-left-commute*: $octo\text{-of-real } r * (x * q) = x * (octo\text{-of-real } r * q)$
unfolding *octo-of-real-times-conv-scaleR* **by** (*simp add: octo-mult-scaleR-right*)

lemma *nonzero-octo-of-real-inverse*:
 $x \neq 0 \implies octo\text{-of-real } (inverse x) = inverse (octo\text{-of-real } x :: octo)$
by (*simp add: octo-eq-iff power2-eq-square divide-simps*)

lemma *octo-of-real-inverse [simp]*:
 $octo\text{-of-real } (inverse x) = inverse (octo\text{-of-real } x)$
by (*simp add: octo-eq-iff power2-eq-square divide-simps*)

lemma *nonzero-octo-of-real-divide*:
 $y \neq 0 \implies octo\text{-of-real } (x / y) = (octo\text{-of-real } x / octo\text{-of-real } y :: octo)$
by (*simp add: divide-inverse divide-octo-def*)

lemma *octo-of-real-divide [simp]*:
 $octo\text{-of-real } (x / y) = (octo\text{-of-real } x / octo\text{-of-real } y :: octo)$
using *divide-inverse divide-octo-def octo-of-real-def octo-of-real-inverse*
by (*metis octo-of-real-mult*)

lemma *octo-of-real-inverse-collapse [simp]*:
assumes $c \neq 0$
shows $octo\text{-of-real } c * octo\text{-of-real } (inverse c) = 1$
 $octo\text{-of-real } (inverse c) * octo\text{-of-real } c = 1$
using *assms* **by** (*simp-all add: octo-eq-iff octo-mult-components power2-eq-square*)

lemma *octo-divide-numeral*:
fixes $x :: octo$ **shows** $x / numeral y = x /_R numeral y$
using *octo-of-real-times-commute[of inverse (numeral y)]*
by (*simp add: scaleR-conv-octo-of-real divide-octo-def flip: octo-of-real-numeral*)

lemma *octo-divide-numeral-sel [simp]*:
 $Ree (x / numeral w) = Ree x / numeral w$
 $Im1 (x / numeral w) = Im1 x / numeral w$
 $Im2 (x / numeral w) = Im2 x / numeral w$

$Im3 (x / numeral w) = Im3 x / numeral w$
 $Im4 (x / numeral w) = Im4 x / numeral w$
 $Im5 (x / numeral w) = Im5 x / numeral w$
 $Im6 (x / numeral w) = Im6 x / numeral w$
 $Im7 (x / numeral w) = Im7 x / numeral w$
unfolding octo-divide-numeral by simp-all

lemma octo-norm-units [simp]:

$norm\ octo-e1 = 1$ $norm (e2::octo) = 1$ $norm (e3::octo) = 1$
 $norm (e4::octo) = 1$ $norm (e5::octo) = 1$ $norm (e6::octo) = 1$ $norm (e7::octo)$
 $= 1$
by (auto simp: norm-octo-def)

lemma e1-nz [simp]: $e1 \neq 0$

and e2-nz [simp]: $e2 \neq 0$

and e3-nz [simp]: $e3 \neq 0$

and e4-nz [simp]: $e4 \neq 0$

and e5-nz [simp]: $e5 \neq 0$

and e6-nz [simp]: $e6 \neq 0$

and e7-nz [simp]: $e7 \neq 0$

by (simp-all add: octo-eq-iff)

2.6 "Expansion" into the traditional notation

lemma octo-unfold:

$q = (Re\ q) *_R e0 + (Im1\ q) *_R e1 + (Im2\ q) *_R e2 + (Im3\ q) *_R e3$
 $+ (Im4\ q) *_R e4 + (Im5\ q) *_R e5 + (Im6\ q) *_R e6 + (Im7\ q) *_R e7$
by (simp add: octo-eq-iff)

lemma octo-trad: Octo x y z w u v q g =

$x *_R e0 + y *_R e1 + z *_R e2 + w *_R e3 + u *_R e4 + v *_R e5 + q *_R$
 $e6 + g *_R e7$
by (simp add: octo-eq-iff)

lemma octo-of-real-eq-Octo: octo-of-real a = Octo a 0 0 0 0 0 0 0

by (simp add: octo-eq-iff)

lemma e1-squared [simp]: $e1 \wedge 2 = -1$

and e2-squared [simp]: $e2 \wedge 2 = -1$

and e3-squared [simp]: $e3 \wedge 2 = -1$

and e4-squared [simp]: $e4 \wedge 2 = -1$

and e5-squared [simp]: $e5 \wedge 2 = -1$

and e6-squared [simp]: $e6 \wedge 2 = -1$

and e7-squared [simp]: $e7 \wedge 2 = -1$

by (simp-all add: octo-eq-iff power2-eq-square-octo octo-mult-components)

lemma inverse-e1 [simp]: $inverse\ e1 = -e1$

and inverse-e2 [simp]: $inverse\ e2 = -e2$

and inverse-e3 [simp]: $inverse\ e3 = -e3$

and *inverse-e4* [*simp*]: *inverse e4 = -e4*
and *inverse-e5* [*simp*]: *inverse e5 = -e5*
and *inverse-e6* [*simp*]: *inverse e6 = -e6*
and *inverse-e7* [*simp*]: *inverse e7 = -e7*
by (*simp-all add: octo-eq-iff*)

2.7 Conjugate of an octonion and related properties.

primcorec *cnj* :: *octo* \Rightarrow *octo*

where

$Ree (cnj z) = Ree z$
 $| Im1 (cnj z) = - Im1 z$
 $| Im2 (cnj z) = - Im2 z$
 $| Im3 (cnj z) = - Im3 z$
 $| Im4 (cnj z) = - Im4 z$
 $| Im5 (cnj z) = - Im5 z$
 $| Im6 (cnj z) = - Im6 z$
 $| Im7 (cnj z) = - Im7 z$

lemma *cnj-cancel-iff* [*simp*]: *cnj x = cnj y \longleftrightarrow x = y*

proof

show *cnj x = cnj y \implies x = y*
by (*simp add: octo-eq-iff*)

qed *auto*

lemma *cnj-cnj* [*simp*]:

$cnj(cnj q) = q$
by (*simp add: octo-eq-iff*)

lemma *cnj-of-real* [*simp*]: *cnj(octo-of-real x) = octo-of-real x*

using *octo-eq-iff*
by (*simp add: octo-of-real-eq-Octo*)

lemma *cnj-zero* [*simp*]: *cnj 0 = 0*

by (*simp add: octo-eq-iff*)

lemma *cnj-zero-iff* [*iff*]: *cnj z = 0 \longleftrightarrow z = 0*

using *cnj-cnj* **by** (*metis cnj-zero*)

lemma *cnj-one* [*simp*]: *cnj 1 = 1*

by (*simp add: octo-eq-iff*)

lemma *cnj-one-iff* [*simp*]: *cnj z = 1 \longleftrightarrow z = 1*

by (*simp add: octo-eq-iff*)

lemma *octo-norm-cnj* [*simp*]: *norm(cnj q) = norm q*

by (*simp add: norm-octo-def*)

lemma *cnj-add* [*simp*]: $cnj (x + y) = cnj x + cnj y$
using *octo-eq-iff inner-real-def of-real-0 of-real-inner-1* **by** *simp*

lemma *cnj-sum* [*simp*]: $cnj (sum f S) = (\sum x \in S. cnj (f x))$
by (*induct S rule: infinite-finite-induct*) **auto**

lemma *cnj-diff* [*simp*]: $cnj (x - y) = cnj x - cnj y$
using *octo-eq-iff*
by (*metis add.commute add-left-cancel cnj-add diff-add-cancel*)

lemma *cnj-minus* [*simp*]: $cnj (- x) = - cnj x$
using *octo-eq-iff cnj-cnj* **by** *auto*

lemma *cnj-inverse* [*simp*]: $cnj (inverse x) = inverse (cnj x)$ **for** $x y :: octo$
using *octo-eq-iff inner-real-def real-inner-1-right* **by** *auto*

lemma *cnj-mult* [*simp*]: $cnj (x * y) = cnj y * cnj x$ **for** $x y :: octo$
using *octo-eq-iff times-octo-def octo-mult-components cnj-cnj*
mult-not-zero nonzero-inverse-mult-distrib **by** *simp*

lemma *cnj-divide* [*simp*]: $cnj (x / y) = (inverse (cnj y)) * cnj x$
for $x y :: octo$
unfolding *divide-octo-def times-octo-def*
using *cnj-inverse cnj-mult octo-mult-components* **by** (*metis times-octo-def*)

lemma *cnj-power* [*simp*]: $cnj (x ^ y) = (cnj x) ^ y$ **for** $x :: octo$
by (*induction y*) (*simp-all add: octo-power-commutes*)

lemma *cnj-of-nat* [*simp*]: $cnj (octo-of-nat x) = octo-of-nat (cnj x)$
using *cnj-of-real octo-of-real-of-nat-eq* **by** *metis*

lemma *cnj-of-int* [*simp*]: $cnj (octo-of-int x) = octo-of-nat (cnj x)$
using *octo-of-real-def octo-of-real-of-int-eq octo-of-int-def octo-of-nat-def*
cnj-of-real **by** *auto*

lemma *cnj-numeral* [*simp*]: $cnj (numeral x) = numeral x$
by (*simp add: numeral-octo-conv-of-nat*)

lemma *cnj-neg-numeral* [*simp*]: $cnj (- numeral x) = - numeral x$
by (*simp add: numeral-octo-conv-of-nat*)

lemma *cnj-scaleR* [*simp*]: $cnj (scaleR r x) = scaleR r (cnj x)$
using *octo-eq-iff inner-real-def ln-one of-real-inner-1* **by** *simp*

lemma *cnj-units* [*simp*]: $cnj e1 = -e1 \quad cnj e2 = -e2 \quad cnj e3 = -e3$
 $cnj e4 = -e4 \quad cnj e5 = -e5 \quad cnj e6 = -e6 \quad cnj e7 = -e7$
by (*simp-all add: octo-eq-iff*)

lemma *cnj-eq-of-real*: $cnj q = octo-of-real x \iff q = octo-of-real x$

proof

show $\text{cnj } q = \text{octo-of-real } x \implies q = \text{octo-of-real } x$

by (*metis cnj-of-real cnj-cnj*)

qed *auto*

lemma *octo-trad-cnj* : $\text{cnj } q = (\text{Ree } q) *_R e0 - (\text{Im1 } q) *_R e1 - (\text{Im2 } q) *_R e2 -$
 $(\text{Im3 } q) *_R e3 -$
 $(\text{Im4 } q) *_R e4 - (\text{Im5 } q) *_R e5 - (\text{Im6 } q) *_R e6 - (\text{Im7 } q) *_R e7$ **for** $q::\text{octo}$
using *cnj-cnj octo-unfold octo-trad cnj-def Octonions.cnj.code* **by** *auto*

lemma *octonion-conjugate-property*:

$\text{cnj } x = -(1/6) *_R (x + (e1 *_R x) *_R e1 + (e2 *_R x) *_R e2 + (e3 *_R x) *_R e3 +$
 $(e4 *_R x) *_R e4 + (e5 *_R x) *_R e5 + (e6 *_R x) *_R e6 + (e7 *_R x) *_R e7)$

by (*simp add: octo-eq-iff octo-mult-components*)

lemma *octo-add-cnj*: $q + \text{cnj } q = 2 *_R (\text{Ree } q) *_R e0$ $\text{cnj } q + q = (2 *_R (\text{Ree } q) *_R$
 $e0)$

by (*simp-all add: octo-eq-iff*)

lemma *octo-add-cnj1*: $q + \text{cnj } q = \text{octo-of-real } (2 *_R (\text{Ree } q))$

$\text{cnj } q + q = \text{octo-of-real } (2 *_R (\text{Ree } q))$

by (*auto simp: octo-eq-iff octo-mult-components*)

lemma *octo-subtract-cnj*:

$q - \text{cnj } q = 2 *_R (\text{Im1 } q *_R e1 + \text{Im2 } q *_R e2 + \text{Im3 } q *_R e3 +$
 $\text{Im4 } q *_R e4 + \text{Im5 } q *_R e5 + \text{Im6 } q *_R e6 + \text{Im7 } q *_R e7)$

by (*simp add: octo-eq-iff*)

lemma *octo-mult-cnj-commute*: $\text{cnj } x * x = x * \text{cnj } x$

using *times-octo-def* **by** *auto*

lemma *octo-cnj-mult-conv-norm*: $\text{cnj } x * x = \text{octo-of-real } (\text{norm } x) ^ 2$

by (*simp add: octo-eq-iff octo-mult-components norm-octo-def power2-eq-square*
flip: octo-of-real-power)

lemma *octo-mult-cnj-conv-norm*: $x * \text{cnj } x = \text{octo-of-real } (\text{norm } x) ^ 2$

by (*simp add: octo-eq-iff octo-mult-components norm-octo-def power2-eq-square*
flip: octo-of-real-power)

lemma *octo-mult-cnj-conv-norm-aux*: $\text{octo-of-real } (\text{norm } x ^ 2) = x * \text{cnj } x$

using *octo-mult-cnj-conv-norm[of x]* **by** (*simp add: octo-mult-cnj-commute*)

lemma *octo-norm-cnj*: $\text{octo-of-real } (\text{inner } x y) = (1/2) *_R (x * (\text{cnj } y) + y *$
 $(\text{cnj } x))$

by (*simp add: octo-eq-iff octo-mult-components inner-octo-def*)

lemma *octo-inverse-cnj*: $\text{inverse } x = \text{cnj } x /_R (\text{norm } x ^ 2)$

by (*auto simp: octo-eq-iff norm-octo-def field-simps*)

lemma *inverse-octo-1*: $x \neq 0 \implies x * \text{inverse } x = (1 :: \text{octo})$
by (*simp* *add*: *octo-mult-scaleR-right octo-mult-cnj-conv-norm-aux [symmetric]*
divide-simps octo-inverse-cnj
del: *octo-of-real-power*)

lemma *inverse-octo-1-sym*: $x \neq 0 \implies \text{inverse } x * x = (1 :: \text{octo})$
by (*metis* *cnj-cnj cnj-inverse cnj-mult cnj-one cnj-zero inverse-octo-1*)

lemma *inverse-0-octo* [*simp*]: $\text{inverse } 0 = (0 :: \text{octo})$
by (*simp* *add*: *octo-eq-iff*)

lemma *inverse-octo-commutes*: $\text{inverse } x * x = x * (\text{inverse } x :: \text{octo})$
by (*cases* $x = 0$) (*simp-all* *add*: *inverse-octo-1 inverse-octo-1-sym*)

lemma *octo-inverse-mult*: $\text{inverse } (x * y) = \text{inverse } y * \text{inverse } x$ **for** $x y :: \text{octo}$
proof–
have $\text{inverse } (x * y) = (\text{cnj } y * \text{cnj } x) /_R (\text{norm } (x * y) \wedge 2)$
by (*simp* *add*: *octo-inverse-cnj*)
also have $\dots = (\text{cnj } y /_R \text{norm } y \wedge 2) * (\text{cnj } x /_R \text{norm } x \wedge 2)$
by (*simp* *add*: *octo-mult-scaleR-left octo-mult-scaleR-right multiplicative-norm-octo*
power2-eq-square)
also have $\dots = \text{inverse } y * \text{inverse } x$
by (*simp* *add*: *octo-inverse-cnj*)
finally show *?thesis* .

qed

lemma *octo-inverse-eq-cnj*: $\text{norm } q = 1 \implies \text{inverse } q = \text{cnj } q$ **for** $q :: \text{octo}$
by (*simp* *add*: *octo-inverse-cnj*)

lemma *octo-in-Reals-if-Re*: **fixes** $q :: \text{real}$ **shows** $\text{Ree}(\text{octo-of-real}(q)) = q$
by *simp*

lemma *octo-in-Reals-if-Re-con*: **assumes** $\text{Ree}(\text{octo-of-real } q) = q$
shows $q \in \text{Reals}$
by (*metis* *Reals-of-real inner-real-def mult.right-neutral of-real-inner-1*)

lemma *octo-in-Reals-if-cnj*: **fixes** $q :: \text{real}$ **shows** $\text{cnj}(\text{octo-of-real}(q)) = \text{octo-of-real } q$
by *simp*

lemma *octo-in-Reals-if-cnj-con*: **assumes** $\text{cnj}(\text{octo-of-real}(q)) = \text{octo-of-real } q$
shows $q \in \text{Reals}$
by (*metis* *Reals-of-real inner-real-def mult.right-neutral of-real-inner-1*)

lemma *norm-power2*: $\text{norm } q \wedge 2 = \text{Ree}(\text{cnj } q * q)$
by (*simp* *add*: *octo-mult-components norm-octo-def power2-eq-square*)

lemma *norm-power2-cnj*: $\text{norm } q \wedge 2 = \text{Ree}(q * \text{cnj } q)$
by (*simp* *add*: *octo-mult-components norm-octo-def power2-eq-square*)

lemma *octo-norm-imaginary*: $\text{Ree } x = 0 \implies x * x = -(\text{octo-of-real } (\text{norm } x))^2$
by (*simp add: octo-eq-iff octo-mult-components norm-octo-def power2-eq-square flip: octo-of-real-power octo-of-real-mult*)

2.8 Linearity and continuity of the components.

lemma *bounded-linear-Ree*: *bounded-linear Ree*
and *bounded-linear-Im1*: *bounded-linear Im1*
and *bounded-linear-Im2*: *bounded-linear Im2*
and *bounded-linear-Im3*: *bounded-linear Im3*
and *bounded-linear-Im4*: *bounded-linear Im4*
and *bounded-linear-Im5*: *bounded-linear Im5*
and *bounded-linear-Im6*: *bounded-linear Im6*
and *bounded-linear-Im7*: *bounded-linear Im7*
by (*simp-all add: bounded-linear-intro [where K=1] norm-octo-def real-le-rsqrt add.assoc*)

lemmas *Cauchy-Ree* = *bounded-linear.Cauchy [OF bounded-linear-Ree]*
lemmas *Cauchy-Im1* = *bounded-linear.Cauchy [OF bounded-linear-Im1]*
lemmas *Cauchy-Im2* = *bounded-linear.Cauchy [OF bounded-linear-Im2]*
lemmas *Cauchy-Im3* = *bounded-linear.Cauchy [OF bounded-linear-Im3]*
lemmas *Cauchy-Im4* = *bounded-linear.Cauchy [OF bounded-linear-Im4]*
lemmas *Cauchy-Im5* = *bounded-linear.Cauchy [OF bounded-linear-Im5]*
lemmas *Cauchy-Im6* = *bounded-linear.Cauchy [OF bounded-linear-Im6]*
lemmas *Cauchy-Im7* = *bounded-linear.Cauchy [OF bounded-linear-Im7]*

lemmas *tendsto-Re* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Ree]*
lemmas *tendsto-Im1* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im1]*
lemmas *tendsto-Im2* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im2]*
lemmas *tendsto-Im3* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im3]*
lemmas *tendsto-Im4* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im4]*
lemmas *tendsto-Im5* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im5]*
lemmas *tendsto-Im6* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im6]*
lemmas *tendsto-Im7* [*tendsto-intros*] = *bounded-linear.tendsto [OF bounded-linear-Im7]*

lemmas *isCont-Ree* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Ree]*
lemmas *isCont-Im1* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im1]*
lemmas *isCont-Im2* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im2]*
lemmas *isCont-Im3* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im3]*
lemmas *isCont-Im4* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im4]*
lemmas *isCont-Im5* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im5]*
lemmas *isCont-Im6* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im6]*
lemmas *isCont-Im7* [*simp*] = *bounded-linear.isCont [OF bounded-linear-Im7]*

lemmas *continuous-Ree* [*simp*] = *bounded-linear.continuous [OF bounded-linear-Ree]*
lemmas *continuous-Im1* [*simp*] = *bounded-linear.continuous [OF bounded-linear-Im1]*
lemmas *continuous-Im2* [*simp*] = *bounded-linear.continuous [OF bounded-linear-Im2]*
lemmas *continuous-Im3* [*simp*] = *bounded-linear.continuous [OF bounded-linear-Im3]*

lemmas *continuous-Im4* [simp] = *bounded-linear.continuous* [OF *bounded-linear-Im4*]
lemmas *continuous-Im5* [simp] = *bounded-linear.continuous* [OF *bounded-linear-Im5*]
lemmas *continuous-Im6* [simp] = *bounded-linear.continuous* [OF *bounded-linear-Im6*]
lemmas *continuous-Im7* [simp] = *bounded-linear.continuous* [OF *bounded-linear-Im7*]

lemmas *continuous-on-Ree* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Ree*]
lemmas *continuous-on-Im1* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im1*]
lemmas *continuous-on-Im2* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im2*]
lemmas *continuous-on-Im3* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im3*]
lemmas *continuous-on-Im4* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im4*]
lemmas *continuous-on-Im5* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im5*]
lemmas *continuous-on-Im6* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im6*]
lemmas *continuous-on-Im7* [continuous-intros] = *bounded-linear.continuous-on*[OF *bounded-linear-Im7*]

lemmas *has-derivative-Ree* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Ree*]
lemmas *has-derivative-Im1* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im1*]
lemmas *has-derivative-Im2* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im2*]
lemmas *has-derivative-Im3* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im3*]
lemmas *has-derivative-Im4* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im4*]
lemmas *has-derivative-Im5* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im5*]
lemmas *has-derivative-Im6* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im6*]
lemmas *has-derivative-Im7* [derivative-intros] = *bounded-linear.has-derivative*[OF *bounded-linear-Im7*]

lemmas *sums-Ree* = *bounded-linear.sums* [OF *bounded-linear-Ree*]
lemmas *sums-Im1* = *bounded-linear.sums* [OF *bounded-linear-Im1*]
lemmas *sums-Im2* = *bounded-linear.sums* [OF *bounded-linear-Im2*]
lemmas *sums-Im3* = *bounded-linear.sums* [OF *bounded-linear-Im3*]
lemmas *sums-Im4* = *bounded-linear.sums* [OF *bounded-linear-Im4*]
lemmas *sums-Im5* = *bounded-linear.sums* [OF *bounded-linear-Im5*]
lemmas *sums-Im6* = *bounded-linear.sums* [OF *bounded-linear-Im6*]
lemmas *sums-Im7* = *bounded-linear.sums* [OF *bounded-linear-Im7*]

2.8.1 Octonionic-specific theorems about sums.

lemma *Ree-sum* [simp]: $\text{Ree} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Ree}(f \ x)) \ S$
and *Im1-sum* [simp]: $\text{Im1} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im1} (f \ x)) \ S$
and *Im2-sum* [simp]: $\text{Im2} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im2} (f \ x)) \ S$
and *Im3-sum* [simp]: $\text{Im3} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im3} (f \ x)) \ S$
and *Im4-sum* [simp]: $\text{Im4} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im4} (f \ x)) \ S$
and *Im5-sum* [simp]: $\text{Im5} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im5} (f \ x)) \ S$
and *Im6-sum* [simp]: $\text{Im6} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im6} (f \ x)) \ S$
and *Im7-sum* [simp]: $\text{Im7} (\text{sum } f \ S) = \text{sum} (\lambda x. \ \text{Im7} (f \ x)) \ S$
by (*induct* S rule: *infinite-finite-induct*; *simp*) $+$

2.8.2 Bound results for real and imaginary components of limits.

lemma *Ree-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{octo.Ree} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Ree} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Re*])

lemma *Im1-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im1} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im1} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im1*])

lemma *Im2-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im2} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im2} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im2*])

lemma *Im3-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im3} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im3} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im3*])

lemma *Im4-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im4} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im4} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im4*])

lemma *Im5-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im5} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im5} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im5*])

lemma *Im6-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im6} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im6} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im6*])

lemma *Im7-tendsto-upperbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ \text{Im7} (f \ x) \leq b; \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow \text{Im7} \ \text{limit} \leq b$
by (*blast intro*: *tendsto-upperbound* [*OF tendsto-Im7*])

lemma *Ree-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \ \text{net}; \ \forall_F \ x \ \text{in} \ \text{net}. \ b \leq \text{octo.Ree} (f \ x); \ \text{net} \neq \text{bot} \rrbracket \Longrightarrow b \leq \text{Ree} \ \text{limit}$

by (blast intro: tendsto-lowerbound [OF tendsto-Re])

lemma *Im1-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im1 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im1 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im1])

lemma *Im2-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im2 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im2 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im2])

lemma *Im3-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im3 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im3 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im3])

lemma *Im4-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im4 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im4 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im4])

lemma *Im5-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im5 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im5 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im5])

lemma *Im6-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im6 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im6 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im6])

lemma *Im7-tendsto-lowerbound*:

$\llbracket (f \longrightarrow \text{limit}) \text{ net}; \forall_F x \text{ in net. } b \leq \text{Im7 } (f x); \text{ net} \neq \text{bot} \rrbracket \implies b \leq \text{Im7 limit}$
by (blast intro: tendsto-lowerbound [OF tendsto-Im7])

lemma *octo-of-real-continuous* [continuous-intros]:

continuous net f \implies *continuous net* ($\lambda x. \text{ octo-of-real } (f x)$)
by (auto simp: octo-of-real-def intro: continuous-intros)

lemma *octo-of-real-continuous-on* [continuous-intros]:

continuous-on S f \implies *continuous-on S* ($\lambda x. \text{ octo-of-real } (f x)$)
by (auto simp: octo-of-real-def intro: continuous-intros)

lemma *of-real-continuous-iff*: *continuous net* ($\lambda x. \text{ octo-of-real } (f x)$) \iff *continuous net f*

proof safe

assume *continuous net* ($\lambda x. \text{ octo-of-real } (f x)$)
hence *continuous net* ($\lambda x. \text{ Ree } (\text{ octo-of-real } (f x))$)
by (rule continuous-Ree)

thus *continuous net f* **by** *simp*
qed (auto intro: continuous-intros)

lemma *of-real-continuous-on-iff*:

continuous-on S ($\lambda x. \text{ octo-of-real } (f x)$) \iff *continuous-on S f*

proof *safe*

assume *continuous-on S* ($\lambda x. \text{octo-of-real } (f x)$)

hence *continuous-on S* ($\lambda x. \text{Ree } (\text{octo-of-real } (f x))$)

by (*rule continuous-on-Ree*)

thus *continuous-on S f* **by** *simp*

qed (*auto intro: continuous-intros*)

2.9 Octonions for describing 7D isometries

2.9.1 The *HIm* operator

definition *HIm* :: *octo* \Rightarrow *real⁷* **where**

HIm *q* \equiv *vector*[*Im1* *q*, *Im2* *q*, *Im3* *q*, *Im4* *q*, *Im5* *q*, *Im6* *q*, *Im7* *q*]

lemma *HIm-Octo*: *HIm* (*Octo* *w x y z u v q g*) = *vector*[*x,y,z*, *u*, *v*, *q*, *g*]

by (*simp add: HIm-def*)

lemma *him-eq*: *HIm* *a* = *HIm* *b* \longleftrightarrow *Im1* *a* = *Im1* *b* \wedge *Im2* *a* = *Im2* *b* \wedge *Im3* *a* = *Im3* *b*

\wedge *Im4* *a* = *Im4* *b* \wedge *Im5* *a* = *Im5* *b* \wedge *Im6* *a* = *Im6* *b* \wedge *Im7* *a* = *Im7* *b*

by (*metis HIm-def vector-7*)

lemma *him-of-real* [*simp*]: *HIm*(*octo-of-real* *a*) = 0

by (*simp add: octo-of-real-eq-Octo HIm-Octo vec-eq-iff vector-def*)

lemma *him-0* [*simp*]: *HIm* 0 = 0

by (*metis him-of-real octo-of-real-0*)

lemma *him-1* [*simp*]: *HIm* 1 = 0

using *HIm-def him-0* **by** *auto*

lemma *him-cnj*: *HIm*(*cnj* *q*) = - *HIm* *q*

by (*simp add: HIm-def vec-eq-iff vector-def*)

lemma *him-mult-left* [*simp*]: *HIm* (*a* *_R *q*) = *a* *_R *HIm* *q*

by (*simp add: HIm-def vec-eq-iff vector-def*)

lemma *him-mult-right* [*simp*]: *HIm* (*q* * *octo-of-real* *a*) = *HIm* *q* * *of-real* *a*

by (*metis Octonions.scaleR-conv-octo-of-real Real-Vector-Spaces.scaleR-conv-of-real him-mult-left octo-of-real-times-commute semiring-normalization-rules(7)*)

lemma *him-add* [*simp*]: *HIm* (*x* + *y*) = *HIm* *x* + *HIm* *y*

and *him-minus* [*simp*]: *HIm* (-*x*) = - *HIm* *x*

and *him-diff* [*simp*]: *HIm* (*x* - *y*) = *HIm* *x* - *HIm* *y*

by (*simp-all add: HIm-def vec-eq-iff vector-def*)

lemma *him-sum* [*simp*]: *HIm* (*sum* *f* *S*) = ($\sum_{x \in S} \text{HIm } (f x)$)

by (*induct S rule: infinite-finite-induct*) *auto*

lemma *linear-him*: *linear HIm*
by (*simp add: linearI*)

2.9.2 The Hv operator

definition $Hv :: \text{real}^7 \Rightarrow \text{octo}$ **where**

$$Hv\ v \equiv \text{Octo } 0\ (v\$1)\ (v\$2)\ (v\$3)\ (v\$4)\ (v\$5)\ (v\$6)\ (v\$7)$$

lemma *Hv-sel* [*simp*]:

$$\begin{aligned} \text{Ree } (Hv\ v) &= 0 & \text{Im1 } (Hv\ v) &= v\ \$\ 1 & \text{Im2 } (Hv\ v) &= v\ \$\ 2 & \text{Im3 } (Hv\ v) &= v\ \$\ 3 \\ \text{Im4 } (Hv\ v) &= v\ \$\ 4 & \text{Im5 } (Hv\ v) &= v\ \$\ 5 & \text{Im6 } (Hv\ v) &= v\ \$\ 6 & \text{Im7 } (Hv\ v) &= v\ \$\ 7 \end{aligned}$$

by (*simp-all add: Hv-def*)

lemma *hv-vec*: $Hv(\text{vec } r) = \text{Octo } 0\ r\ r\ r\ r\ r\ r\ r$
by (*simp add: Hv-def*)

lemma *hv-eq-zero* [*simp*]: $Hv\ v = 0 \iff v = 0$
by (*simp add: octo-eq-iff vec-eq-iff*) (*metis exhaust-7*)

lemma *hv-zero* [*simp*]: $Hv\ 0 = 0$
by *simp*

lemma *hv-vector* [*simp*]: $Hv(\text{vector}[x,y,z,u,v,q,g]) = \text{Octo } 0\ x\ y\ z\ u\ v\ q\ g$
by (*simp add: Hv-def*)

lemma *hv-basis*:

$$\begin{aligned} Hv(\text{axis } 1\ 1) &= e1 & Hv(\text{axis } 2\ 1) &= e2 & Hv(\text{axis } 3\ 1) &= e3 \\ Hv(\text{axis } 4\ 1) &= e4 & Hv(\text{axis } 5\ 1) &= e5 & Hv(\text{axis } 6\ 1) &= e6 & Hv(\text{axis } 7\ 1) &= e7 \end{aligned}$$

by (*simp-all add: octo-eq-iff*)

lemma *hv-add* [*simp*]: $Hv(x + y) = Hv\ x + Hv\ y$
by (*simp add: Hv-def octo-eq-iff*)

lemma *hv-minus* [*simp*]: $Hv(-x) = -Hv\ x$
by (*simp add: Hv-def octo-eq-iff*)

lemma *hv-diff* [*simp*]: $Hv(x - y) = Hv\ x - Hv\ y$
by (*simp add: Hv-def octo-eq-iff*)

lemma *hv-cmult* [*simp*]:

$$Hv(\text{scaleR } a\ x) = \text{scaleR } a\ (Hv\ x)$$

by (*simp add: Hv-def octo-eq-iff*)

lemma *hv-sum* [*simp*]: $Hv(\text{sum } f\ S) = (\sum x \in S. Hv\ (f\ x))$
by (*induct S rule: infinite-finite-induct*) *auto*

lemma *hv-inj*: $Hv\ x = Hv\ y \iff x = y$
by (*simp add: Hv-def octo-eq-iff vec-eq-iff*) (*metis (full-types) exhaust-7*)

lemma *linear-hv*: *linear Hv*
using *octo-of-real-def* **by** (*simp add: linearI*)

lemma *him-hv* [*simp*]: $HIm(Hv\ x) = x$
using *HIm-def hv-inj octo-eq-iff* **by** *fastforce*

lemma *cnj-hv* [*simp*]: $cnj(Hv\ v) = -Hv\ v$
by (*simp add: octo-eq-iff*)

lemma *hv-him*: $Hv(HIm\ q) = Octo\ 0\ (Im1\ q)\ (Im2\ q)\ (Im3\ q)\ (Im4\ q)\ (Im5\ q)\ (Im6\ q)\ (Im7\ q)$
by (*simp add: HIm-def*)

lemma *hv-him-eq*: $Hv(HIm\ q) = q \longleftrightarrow Ree\ q = 0$
by (*simp add: hv-him octo-eq-iff*)

lemma *dot-hv* [*simp*]: $Hv\ u \cdot Hv\ v = u \cdot v$
by (*simp add: Hv-def inner-octo-def inner-vec-def sum-7*)

lemma *norm-hv* [*simp*]: $norm\ (Hv\ v) = norm\ v$
by (*simp add: norm-eq*)

2.9.3 Related basic identities

lemma *mult-hv-eq-cross-dot*: $Hv\ x * Hv\ y = Hv(x \times_7 y) - octo-of-real\ (inner\ x\ y)$
by (*simp add: octo-eq-iff inner-octo-def cross7-components octo-mult-components inner-vec-def sum-7*)

lemma *octonion-identity1-cross7* :
 $Hv\ (x \times_7 y) = (1/2) *_{\mathbb{R}} (Hv\ x * Hv\ y - Hv\ y * Hv\ x)$
by (*simp add: octo-eq-iff octo-mult-components cross7-components*)

lemma *octonion-identity2-cross7*:
 $Hv\ (x \times_7 (y \times_7 z) + y \times_7 (z \times_7 x) + z \times_7 (x \times_7 y)) =$
 $-(3/2) *_{\mathbb{R}} ((Hv\ x * Hv\ y) * Hv\ z - Hv\ x * (Hv\ y * Hv\ z))$
unfolding *octo-eq-iff octo-mult-components cross7-components Hv-sel scaleR-octo.sel*
vector-add-component minus-octo.sel mult-zero-left mult-zero-right add-0-left
add-0-right diff-0 diff-0-right
by *algebra*

2.10 Representing orthogonal transformations as conjugation or congruence with an octonion.

lemma *HIm-nth* [*simp*]:
 $HIm\ x\ \$\ 1 = Im1\ x\ HIm\ x\ \$\ 2 = Im2\ x\ HIm\ x\ \$\ 3 = Im3\ x\ HIm\ x\ \$\ 4 = Im4\ x\ HIm\ x\ \$\ 5 = Im5\ x\ HIm\ x\ \$\ 6 = Im6\ x\ HIm\ x\ \$\ 7 = Im7\ x$
by (*simp-all add: HIm-def*)

lemma *orthogonal-transformation-octo-congruence:*
assumes $\text{norm } q = 1$
shows *orthogonal-transformation* $(\lambda x. \text{HIm}(\text{cnj } q * \text{Hv } x * q))$
proof –
have $\text{ng}: (\text{Ree } q)^2 + (\text{Im1 } q)^2 + (\text{Im2 } q)^2 + (\text{Im3 } q)^2 + (\text{Im4 } q)^2 + (\text{Im5 } q)^2 + (\text{Im6 } q)^2 + (\text{Im7 } q)^2 = 1$
using *assms norm-octo-def by auto*
have *Vector-Spaces.linear* $(*_R) (*_R) (\lambda x. \text{HIm} (\text{Octonions.cnj } q * \text{Hv } x * q))$
by *unfold-locales (simp-all add: octo-distrib-left octo-distrib-right octo-mult-scaleR-left octo-mult-scaleR-right)*
moreover have $\text{HIm} (\text{Octonions.cnj } q * \text{Hv } v * q) \cdot \text{HIm} (\text{Octonions.cnj } q * \text{Hv } w * q) =$
 $((\text{Ree } q)^2 + (\text{Im1 } q)^2 + (\text{Im2 } q)^2 + (\text{Im3 } q)^2 + (\text{Im4 } q)^2 + (\text{Im5 } q)^2 + (\text{Im6 } q)^2 +$
 $(\text{Im7 } q)^2) \wedge 2 * (v \cdot w)$ **for** $v \ w$
unfolding *octo-mult-components cnj.sel Hv-sel inner-vec-def sum-7 HIm-nth inner-real-def*
by *algebra*
ultimately show *?thesis*
by *(simp add: orthogonal-transformation-def linear-def ng)*
qed

lemma *orthogonal-transformation-octo-conjugation:*
assumes $q \neq 0$
shows *orthogonal-transformation* $(\lambda x. \text{HIm} (\text{inverse } q * \text{Hv } x * q))$
proof –
obtain $c \ d$ **where** $\text{eq}: q = \text{octo-of-real } c * d$ **and** $1: \text{norm } d = 1$
proof
show $1: q = \text{octo-of-real} (\text{norm } q) * (\text{inverse} (\text{octo-of-real} (\text{norm } q)) * q)$
using *assms norm-eq-zero right-inverse multiplicative-norm-octo*
by *(metis Octonions.scaleR-conv-octo-of-real octo-of-real-inverse scaleR-one scaleR-scaleR)*
show $\text{norm} (\text{inverse} (\text{octo-of-real} (\text{norm } q)) * q) = 1$
using *assms 1 norm-octo-def norm-mult inverse-octo-1 inverse-octo-1-sym nonzero-octo-of-real-inverse octo-inverse-eq-cnj cnj-of-real mult-cancel-left2 multiplicative-norm-octo norm-eq-zero norm-octo-squared norm-power2-cnj octo-mult-cnj-conv-norm power2-eq-square-octo*
by *metis*
qed
have $c \neq 0$
using *assms eq by (metis Octonions.scaleR-conv-octo-of-real scale-zero-left)*

then have $\text{HIm} (\text{Octonions.cnj } d * \text{Hv } x * d) =$
 $\text{HIm} (\text{inverse} (\text{octo-of-real } c * d) * \text{Hv } x * (\text{octo-of-real } c * d))$ **for** x
proof *(simp add: flip: octo-inverse-eq-cnj [OF 1] of-real-inverse)*
assume $c \neq 0$
then have $\text{inverse } d = \text{inverse } d * \text{inverse} (c *_R 1) * c *_R 1$
using *octo-of-real-def octo-of-real-inverse octo-of-real-inverse-collapse(1)*

```

      octo-of-real-times-commute octo-of-real-times-left-commute by force
then have  $inverse\ d * Hv\ x * d = inverse\ (c *_{R}\ 1 * d) * Hv\ x * c *_{R}\ (d * 1)$ 
by (metis (no-types) mult-1-right-octo octo-inverse-mult octo-mult-scaleR-left
      octo-mult-scaleR-right)
then show
   $HIm\ (inverse\ d * Hv\ x * d) = HIm\ (inverse\ (octo-of-real\ c * d) * Hv\ x * (octo-of-real\ c * d))$ 
using octo-mult-scaleR-right octo-of-real-def octo-of-real-times-commute by
presburger
qed

then show ?thesis
using orthogonal-transformation-octo-congruence [OF 1]
by (simp add: eq)
qed

unbundle no-cross7-syntax

bundle octonion-syntax
begin

notation octo-e0 (e0)
notation octo-e1 (e1)
notation octo-e2 (e2)
notation octo-e3 (e3)
notation octo-e4 (e4)
notation octo-e5 (e5)
notation octo-e6 (e6)
notation octo-e7 (e7)

end

bundle no-octonion-syntax
begin

no-notation octo-e0 (e0)
no-notation octo-e1 (e1)
no-notation octo-e2 (e2)
no-notation octo-e3 (e3)
no-notation octo-e4 (e4)
no-notation octo-e5 (e5)
no-notation octo-e6 (e6)
no-notation octo-e7 (e7)

end

unbundle no-octonion-syntax
hide-const (open) Octonions.cnj

```

end

References

- [1] J. C. Baez. The octonions. *Bull. Amer. Math. Soc.*, 39:145–205, 2002.
- [2] H.-D. E. et al. (eds.). *Numbers*. Springer, New York, 1991.
- [3] P. Lounesto. *Clifford Algebras and Spinors*. Cambridge University Press, 2 edition, 2001.