Octonions

Angeliki Koutsoukou-Argyraki

March 3, 2024

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

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*⟩

lemma *forall-7*: (∀ *i::7. P i*) ⟷ *P 1 ∧ P 2 ∧ P 3∧ P 4 ∧ P 5 ∧ P 6∧ P 7*
  ⟨*proof*⟩

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*
  ⟨*proof*⟩

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*]))
  ⟨*proof*⟩

lemma *UNIV-7*: *UNIV* = {*1::7, 2::7, 3::7, 4::7, 5::7, 6::7, 7::7*}
  ⟨*proof*⟩

lemma *sum-7*: *sum f* (*UNIV::7 set*) = *f 1* + *f 2* + *f 3* + *f 4* + *f 5* + *f 6* + *f 7*
  ⟨*proof*⟩

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 $x \neq y$ ⟨*proof*⟩

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 $x\$1 = y\$1 \wedge x\$2 = y\$2 \wedge x\$3 = y\$3 \wedge x\$4 = y\$4 \wedge x\$5 = y\$5 \wedge x\$6 = y\$6 \wedge x\$7 = y\$7$
⟨*proof*⟩

lemma *numeral-4-eq-4*: $4 = Suc(\ Suc\ (Suc\ (Suc\ 0)))$
⟨*proof*⟩
lemma *numeral-5-eq-5*: $5 = Suc(Suc(\ Suc\ (Suc\ (Suc\ 0))))$
⟨*proof*⟩
lemma *numeral-6-eq-6*: $6 = Suc(\ Suc(Suc(\ Suc\ (Suc\ (Suc\ 0)))))$
⟨*proof*⟩
lemma *numeral-7-eq-7*: $7 = Suc(Suc(\ Suc(Suc(\ Suc\ (Suc\ (Suc\ 0))))))$
⟨*proof*⟩

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** $\times_7$ *80*)
where $a \times_7 b \equiv$
$vector\ [a\$2 * b\$4 - a\$4 * b\$2 + a\$3 * b\$7 - a\$7 * b\$3 + a\$5 * b\$6 - a\$6 * b\$5\ ,$
$a\$3 * b\$5 - a\$5 * b\$3 + a\$4 * b\$1 - a\$1 * b\$4 + a\$6 * b\$7 - a\$7 * b\$6\ ,$
$a\$4 * b\$6 - a\$6 * b\$4 + a\$5 * b\$2 - a\$2 * b\$5 + a\$7 * b\$1 - a\$1 * b\$7\ ,$
$a\$5 * b\$7 - a\$7 * b\$5 + a\$6 * b\$3 - a\$3 * b\$6 + a\$1 * b\$2 - a\$2 * b\$1\ ,$
$a\$6 * b\$1 - a\$1 * b\$6 + a\$7 * b\$4 - a\$4 * b\$7 + a\$2 * b\$3 - a\$3 * b\$2\ ,$
$a\$7 * b\$2 - a\$2 * b\$7 + a\$1 * b\$5 - a\$5 * b\$1 + a\$3 * b\$4 - a\$4 * b\$3\ ,$
$a\$1 * b\$3 - a\$3 * b\$1 + a\$2 * b\$6 - a\$6 * b\$2 + a\$4 * b\$5 - a\$5 * b\$4\ ]$

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*: $x \cdot (x \times_7 y) = 0$ $x \cdot (y \times_7 x) = 0$ $(x \times_7 y) \cdot y = 0$ $(y \times_7 x) \cdot y = 0$
⟨*proof*⟩

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
⟨*proof*⟩

lemma *cross7-zero-left* [*simp*]: $0 \times_7 x = 0$
and *cross7-zero-right* [*simp*]: $x \times_7 0 = 0$
⟨*proof*⟩

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

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

lemma *cross7-add-left*: $(x + y) \times_7 z = (x \times_7 z) + (y \times_7 z)$
and *cross7-add-right*: $x \times_7 (y + z) = (x \times_7 y) + (x \times_7 z)$
⟨*proof*⟩

lemma *cross7-mult-left*: $(c *_R x) \times_7 y = c *_R (x \times_7 y)$
and *cross7-mult-right*: $x \times_7 (c *_R y) = c *_R (x \times_7 y)$
⟨*proof*⟩

lemma *cross7-minus-left* [*simp*]: $(-x) \times_7 y = - (x \times_7 y)$
and *cross7-minus-right* [*simp*]: $x \times_7 -y = - (x \times_7 y)$
⟨*proof*⟩

lemma *left-diff-distrib*: $(x - y) \times_7 z = x \times_7 z - y \times_7 z$
and *right-diff-distrib*: $x \times_7 (y - z) = x \times_7 y - x \times_7 z$
⟨*proof*⟩

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*: $(x \times_7 y) \cdot z = x \cdot (y \times_7 z)$
⟨*proof*⟩

lemma *scalar7-triple1*: $x \cdot (y \times_7 z) = y \cdot (z \times_7 x)$
and *scalar7-triple2*: $x \cdot (y \times_7 z) = z \cdot (x \times_7 y)$
⟨*proof*⟩

lemma *cross7-components*:
$(x \times_7 y)\$1 = x\$2 * y\$4 - x\$4 * y\$2 + x\$3 * y\$7 - x\$7 * y\$3 + x\$5 * y\$6 - x\$6 * y\$5$
$(x \times_7 y)\$2 = x\$4 * y\$1 - x\$1 * y\$4 + x\$3 * y\$5 - x\$5 * y\$3 + x\$6 * y\$7 - x\$7 * y\$6$
$(x \times_7 y)\$3 = x\$5 * y\$2 - x\$2 * y\$5 + x\$4 * y\$6 - x\$6 * y\$4 + x\$7 * y\$1 - x\$1 * y\$7$
$(x \times_7 y)\$4 = x\$1 * y\$2 - x\$2 * y\$1 + x\$6 * y\$3 - x\$3 * y\$6 + x\$5 * y\$7 - x\$7 * y\$5$
$(x \times_7 y)\$5 = x\$6 * y\$1 - x\$1 * y\$6 + x\$2 * y\$3 - x\$3 * y\$2 + x\$7 * y\$4 - x\$4 * y\$7$
$(x \times_7 y)\$6 = x\$1 * y\$5 - x\$5 * y\$1 + x\$7 * y\$2 - x\$2 * y\$7 + x\$3 * y\$4 - x\$4 * y\$3$
$(x \times_7 y)\$7 = x\$1 * y\$3 - x\$3 * y\$1 + x\$4 * y\$5 - x\$5 * y\$4 + x\$2 * y\$6 - x\$6 * y\$2$
⟨*proof*⟩

Nonassociativity of the 7D cross product showed using a counterexample

lemma *cross7-nonassociative*:
$\neg (\forall (c::real^\frown 7) (a::real^\frown 7) (b::real^\frown 7) . c \times_7 (a \times_7 b) = (c \times_7 a) \times_7 b)$
⟨*proof*⟩

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

lemma *cross7-not-Jacobi*:
$\neg (\forall (c::real^\frown 7) (a::real^\frown 7) (b::real^\frown 7) . (c \times_7 a) \times_7 b + (b \times_7 c) \times_7 a + (a \times_7 b) \times_7 c = 0)$

⟨*proof*⟩

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*:
$\neg(\forall (c::real^\frown 7) (a::real^\frown 7) (b::real^\frown 7).(c \times_7 a) \times_7 b = (c \cdot b) *_R a - (c \cdot a) *_R b)$
⟨*proof*⟩

lemma *axis-nth-neq* [*simp*]: $i \neq j \Longrightarrow axis\ i\ x\ \$\ j = 0$
⟨*proof*⟩

lemma *cross7-basis*:
$(axis\ 1\ 1) \times_7 (axis\ 2\ 1) = axis\ 4\ 1$ $(axis\ 2\ 1) \times_7 (axis\ 1\ 1) = -(axis\ 4\ 1)$

$(axis\ 2\ 1) \times_7 (axis\ 3\ 1) = axis\ 5\ 1$ $(axis\ 3\ 1) \times_7 (axis\ 2\ 1) = -(axis\ 5\ 1)$
$(axis\ 3\ 1) \times_7 (axis\ 4\ 1) = axis\ 6\ 1$ $(axis\ 4\ 1) \times_7 (axis\ 3\ 1) = -(axis\ 6\ 1)$
$(axis\ 2\ 1) \times_7 (axis\ 4\ 1) = axis\ 1\ 1$ $(axis\ 4\ 1) \times_7 (axis\ 2\ 1) = -(axis\ 1\ 1)$
$(axis\ 4\ 1) \times_7 (axis\ 5\ 1) = axis\ 7\ 1$ $(axis\ 5\ 1) \times_7 (axis\ 4\ 1) = -(axis\ 7\ 1)$
$(axis\ 3\ 1) \times_7 (axis\ 5\ 1) = axis\ 2\ 1$ $(axis\ 5\ 1) \times_7 (axis\ 3\ 1) = -(axis\ 2\ 1)$
$(axis\ 4\ 1) \times_7 (axis\ 6\ 1) = axis\ 3\ 1$ $(axis\ 6\ 1) \times_7 (axis\ 4\ 1) = -(axis\ 3\ 1)$
$(axis\ 5\ 1) \times_7 (axis\ 7\ 1) = axis\ 4\ 1$ $(axis\ 7\ 1) \times_7 (axis\ 5\ 1) = -(axis\ 4\ 1)$
$(axis\ 4\ 1) \times_7 (axis\ 1\ 1) = axis\ 2\ 1$ $(axis\ 1\ 1) \times_7 (axis\ 4\ 1) = -(axis\ 2\ 1)$
$(axis\ 5\ 1) \times_7 (axis\ 2\ 1) = axis\ 3\ 1$ $(axis\ 2\ 1) \times_7 (axis\ 5\ 1) = -(axis\ 3\ 1)$
$(axis\ 6\ 1) \times_7 (axis\ 3\ 1) = axis\ 4\ 1$ $(axis\ 3\ 1) \times_7 (axis\ 6\ 1) = -(axis\ 4\ 1)$
$(axis\ 7\ 1) \times_7 (axis\ 4\ 1) = axis\ 5\ 1$ $(axis\ 4\ 1) \times_7 (axis\ 7\ 1) = -(axis\ 5\ 1)$
$(axis\ 5\ 1) \times_7 (axis\ 6\ 1) = axis\ 1\ 1$ $(axis\ 6\ 1) \times_7 (axis\ 5\ 1) = -(axis\ 1\ 1)$
$(axis\ 6\ 1) \times_7 (axis\ 7\ 1) = axis\ 2\ 1$ $(axis\ 7\ 1) \times_7 (axis\ 6\ 1) = -(axis\ 2\ 1)$
$(axis\ 7\ 1) \times_7 (axis\ 1\ 1) = axis\ 3\ 1$ $(axis\ 1\ 1) \times_7 (axis\ 7\ 1) = -(axis\ 3\ 1)$
$(axis\ 6\ 1) \times_7 (axis\ 1\ 1) = axis\ 5\ 1$ $(axis\ 1\ 1) \times_7 (axis\ 6\ 1) = -(axis\ 5\ 1)$
$(axis\ 7\ 1) \times_7 (axis\ 2\ 1) = axis\ 6\ 1$ $(axis\ 2\ 1) \times_7 (axis\ 7\ 1) = -(axis\ 6\ 1)$
$(axis\ 1\ 1) \times_7 (axis\ 3\ 1) = axis\ 7\ 1$ $(axis\ 3\ 1) \times_7 (axis\ 1\ 1) = -(axis\ 7\ 1)$
$(axis\ 1\ 1) \times_7 (axis\ 5\ 1) = axis\ 6\ 1$ $(axis\ 5\ 1) \times_7 (axis\ 1\ 1) = -(axis\ 6\ 1)$
$(axis\ 2\ 1) \times_7 (axis\ 6\ 1) = axis\ 7\ 1$ $(axis\ 6\ 1) \times_7 (axis\ 2\ 1) = -(axis\ 7\ 1)$
$(axis\ 3\ 1) \times_7 (axis\ 7\ 1) = axis\ 1\ 1$ $(axis\ 7\ 1) \times_7 (axis\ 3\ 1) = -(axis\ 1\ 1)$
⟨*proof*⟩

lemma *cross7-basis-zero*:
$u=0 \implies (u \times_7 axis\ 1\ 1 = 0) \wedge (u \times_7 axis\ 2\ 1 = 0) \wedge (u \times_7 axis\ 3\ 1 = 0)$
$\wedge (u \times_7 axis\ 4\ 1 = 0) \wedge (u \times_7 axis\ 5\ 1 = 0\ ) \wedge (u \times_7 axis\ 6\ 1 = 0\ )$
$\wedge (u \times_7 axis\ 7\ 1 = 0)$
⟨*proof*⟩

lemma *cross7-basis-nonzero*:
$\neg (u \times_7 axis\ 1\ 1 = 0) \vee \neg (u \times_7 axis\ 2\ 1 = 0) \vee \neg (u \times_7 axis\ 3\ 1 = 0)$
$\vee \neg\ (u \times_7 axis\ 4\ 1 = 0)\ \vee \neg\ (u \times_7 axis\ 5\ 1 = 0\ ) \vee \neg\ (u \times_7 axis\ 6\ 1 = 0\ )$
$\vee \neg\ (u \times_7 axis\ 7\ 1 = 0) \implies u \neq 0$
⟨*proof*⟩

Pythagorean theorem/magnitude

lemma *norm-square-vec-eq*: $norm\ x\ \hat{}\ 2 = (\sum i{\in}UNIV.\ x\ \$\ i\ \hat{}\ 2)$
⟨*proof*⟩

lemma *norm-cross7-dot-magnitude*: $(norm\ (x \times_7 y))^2 = (norm\ x)^2 * (norm\ y)^2 - (x \cdot y)^2$
⟨*proof*⟩

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

lemma *cross7-eq-self*: $x \times_7 y = x \longleftrightarrow x = 0$ $x \times_7 y = y \longleftrightarrow y = 0$
⟨*proof*⟩

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*⟩

lemma *bilinear-cross7*: *bilinear* $(\times_7)$
⟨*proof*⟩

1.3 Continuity

lemma *continuous-cross7*: ⟦*continuous F f*; *continuous F g*⟧ ⟹ *continuous F* ($\lambda x.$ $f\ x \times_7 g\ x$)
⟨*proof*⟩

lemma *continuous-on-cross*:
fixes *f* :: *'a::t2-space* ⇒ *real^7*
shows ⟦*continuous-on S f*; *continuous-on S g*⟧ ⟹ *continuous-on S* ($\lambda x.$ $f\ x \times_7$ $g\ x$)
⟨*proof*⟩

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?*]:
⟦*Ree x = Ree y*; *Im1 x = Im1 y*; *Im2 x = Im2 y*; *Im3 x = Im3 y*;
Im4 x = Im4 y;*Im5 x = Im5 y*; *Im6 x = Im6 y*; *Im7 x = Im7 y*⟧ ⟹ *x = y*
⟨*proof*⟩

lemma *octo-eq-iff*:
x = y ⟷ *Ree x = Ree y* ∧ *Im1 x = Im1 y* ∧ *Im2 x = Im2 y* ∧ *Im3 x = Im3 y* ∧
Im4 x = Im4 y ∧ *Im5 x = Im5 y* ∧ *Im6 x = Im6 y* ∧ *Im7 x = Im7 y*
⟨*proof*⟩

context
begin

primcorec *octo-e0* :: *octo* (*e0*)

where *Ree e0 = 1* | *Im1 e0 = 0* | *Im2 e0 = 0* | *Im3 e0 = 0*
| *Im4 e0 = 0* | *Im5 e0 = 0* | *Im6 e0 = 0* | *Im7 e0 = 0*

primcorec *octo-e1* :: *octo* (*e1*)
where *Ree e1 = 0* | *Im1 e1 = 1* | *Im2 e1 = 0* | *Im3 e1 = 0*
| *Im4 e1 = 0* | *Im5 e1 = 0* | *Im6 e1 = 0* | *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*
| *Im1* (− *x*) = − *Im1 x*
| *Im2* (− *x*) = − *Im2 x*
| *Im3* (− *x*) = − *Im3 x*
| *Im4* (− *x*) = − *Im4 x*
| *Im5* (− *x*) = − *Im5 x*
| *Im6* (− *x*) = − *Im6 x*
| *Im7* (− *x*) = − *Im7 x*

primcorec *minus-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*

instance
⟨*proof*⟩

end

lemma *octo-eq-0-iff*:
x = *0* ⟷ *Ree x* ^ *2* + *Im1 x* ^ *2* + *Im2 x* ^ *2* + *Im3 x* ^ *2* +
Im4 x ^ *2* + *Im5 x* ^ *2* + *Im6 x* ^ *2* + *Im7 x* ^ *2* = *0*
⟨*proof*⟩

2.3 A Normed Vector Space

instantiation *octo* :: *real-vector*

begin

primcorec *scaleR-octo*
where
Ree (*scaleR r x*) = *r* ∗ *Ree x*
| *Im1* (*scaleR r x*) = *r* ∗ *Im1 x*
| *Im2* (*scaleR r x*) = *r* ∗ *Im2 x*
| *Im3* (*scaleR r x*) = *r* ∗ *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
⟨*proof*⟩

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 ⟨*proof*⟩
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*⟩

instantiation *octo* ::*real-normed-vector*
begin

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

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

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

definition [*code del*]:
(*uniformity* :: (*octo* × *octo*) *filter*) = (*INF* $e \in \{0 <..\}$. *principal* {(*x*, *y*). *dist x y*

< *e*})

definition [*code del*]:
open (*U* :: *octo set*) ⟷ (∀ *x*∈*U*. *eventually* (λ(*x*′, *y*). *x*′ = *x* ⟶ *y* ∈ *U*) *uniformity*)

lemma *norm-eq-L2*: *norm x* = *L2-set* (*octo-proj x*) {..*7*}
⟨*proof*⟩

instance ⟨*proof*⟩

end

lemma *norm-octo-squared*:
norm x ^ *2* = *Ree x* ^ *2* + *Im1 x* ^ *2* + *Im2 x* ^ *2* + *Im3 x* ^ *2* +
Im4 x ^ *2* + *Im5 x* ^ *2* + *Im6 x* ^ *2* + *Im7 x* ^ *2*
⟨*proof*⟩

instantiation *octo* :: *real-inner*
begin

definition *inner-octo* **where**
inner-octo x y = *Ree x* ∗ *Ree y* + *Im1 x* ∗ *Im1 y* + *Im2 x* ∗ *Im2 y* + *Im3 x* ∗ *Im3 y*
+ *Im4 x* ∗ *Im4 y* + *Im5 x* ∗ *Im5 y* + *Im6 x* ∗ *Im6 y* + *Im7 x* ∗ *Im7 y* **for** *x y*::*octo*

instance
⟨*proof*⟩

end

lemma *octo-inner-1* [*simp*]: *inner 1 x* = *Ree x*
and *octo-inner-1-right* [*simp*]: *inner x 1* = *Ree x*
⟨*proof*⟩

lemma *octo-inner-e1-left* [*simp*]: *inner e1 x* = *Im1 x*
and *octo-inner-e1-right* [*simp*]: *inner x e1* = *Im1 x*
⟨*proof*⟩

lemma *octo-inner-e2-left* [*simp*]: *inner e2 x* = *Im2 x*
and *octo-inner-e2-right* [*simp*]: *inner x e2* = *Im2 x*
⟨*proof*⟩

lemma *octo-inner-e3-left* [*simp*]: *inner e3 x* = *Im3 x*
and *octo-inner-e3-right* [*simp*]: *inner x e3* = *Im3 x*
⟨*proof*⟩

lemma *octo-inner-e4-left* [*simp*]: *inner e4 x* = *Im4 x*

and *octo-inner-e4-right* [*simp*]: *inner x e4* = *Im4 x*
⟨*proof*⟩

lemma *octo-inner-e5-left* [*simp*]: *inner e5 x* = *Im5 x*
and *octo-inner-e5-right* [*simp*]: *inner x e5* = *Im5 x*
⟨*proof*⟩

lemma *octo-inner-e6-left* [*simp*]: *inner e6 x* = *Im6 x*
and *octo-inner-e6-right* [*simp*]: *inner x e6* = *Im6 x*
⟨*proof*⟩

lemma *octo-inner-e7-left* [*simp*]: *inner e7 x* = *Im7 x*
and *octo-inner-e7-right* [*simp*]: *inner x e7* = *Im7 x*
⟨*proof*⟩

lemma *octo-norm-pow-2-inner*: (*norm x*) ^ *2* = *inner x x* **for** *x*::*octo*
⟨*proof*⟩

lemma *octo-norm-property*:
inner x y = (*1/2*)∗ ((*norm*(*x*+*y*))^*2* − (*norm*(*x*))^*2* − (*norm*(*y*))^*2*) **for** *x y* ::*octo*
⟨*proof*⟩

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*] ⇒ *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 ⟨*proof*⟩

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* ^ *2*
+*Im4 x* ^ *2* + *Im5 x* ^ *2* + *Im6 x* ^ *2* + *Im7 x* ^*2*)
| *Im3* (*inverse x*) = − (*Im3 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*)
| *Im4* (*inverse x*) = − (*Im4 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*)
| *Im5* (*inverse x*) = − (*Im5 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*)
| *Im6* (*inverse x*) = − (*Im6 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*)
| *Im7* (*inverse x*) = − (*Im7 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*)

definition *x div y* = *x* ∗ (*inverse y*) **for** *x y* :: *octo*

instance ⟨*proof*⟩

end

lemma *octo-mult-components*:

Ree (*x* ∗ *y*) = *Ree x* ∗ *Ree y* − *Im1 x* ∗ *Im1 y* − *Im2 x* ∗ *Im2 y* − *Im3 x* ∗ *Im3 y*
− *Im4 x* ∗ *Im4 y* − *Im5 x* ∗ *Im5 y* − *Im6 x* ∗ *Im6 y*− *Im7 x* ∗ *Im7 y*
Im1 (*x* ∗ *y*) = *Ree x* ∗ *Im1 y* + *Im1 x* ∗ *Ree y* + *Im2 x* ∗ *Im4 y* +*Im3 x* ∗ *Im7 y* −
Im4 x ∗ *Im2 y* +*Im5 x* ∗ *Im6 y* − *Im6 x* ∗ *Im5 y* − *Im7 x* ∗ *Im3 y*
Im2 (*x* ∗ *y*) = *Ree x* ∗ *Im2 y* − *Im1 x* ∗ *Im4 y*+ *Im2 x* ∗ *Ree y* + *Im3 x* ∗ *Im5 y*
+ *Im4 x* ∗ *Im1 y* − *Im5 x* ∗ *Im3 y* + *Im6 x* ∗ *Im7 y* − *Im7 x* ∗*Im6 y*
Im3 (*x* ∗ *y*) = *Ree x* ∗ *Im3 y* −*Im1 x* ∗ *Im7 y* −*Im2 x* ∗*Im5 y* +*Im3 x* ∗ *Ree y* + *Im4 x* ∗ *Im6 y*
+ *Im5 x* ∗*Im2 y* − *Im6 x* ∗ *Im4 y* + *Im7 x* ∗ *Im1 y*
Im4 (*x* ∗*y*) = *Ree x* ∗ *Im4 y* + *Im1 x* ∗ *Im2 y* − *Im2 x* ∗ *Im1 y* −*Im3 x* ∗ *Im6 y* + *Im4 x* ∗ *Ree y*
+*Im5 x* ∗ *Im7 y* +*Im6 x* ∗ *Im3 y* −*Im7 x* ∗ *Im5 y*
Im5 (*x* ∗ *y*) = *Ree x* ∗ *Im5 y* − *Im1 x* ∗ *Im6 y* +*Im2 x* ∗ *Im3 y* −*Im3 x* ∗ *Im2 y* −*Im4 x* ∗ *Im7 y*
+*Im5 x* ∗ *Ree y* +*Im6 x* ∗ *Im1 y* + *Im7 x* ∗ *Im4 y*
Im6 (*x* ∗ *y*) = *Ree x* ∗ *Im6 y* + *Im1 x* ∗ *Im5 y* − *Im2 x* ∗ *Im7 y* +*Im3 x* ∗ *Im4 y* − *Im4 x* ∗ *Im3 y*
−*Im5 x* ∗ *Im1 y* + *Im6 x* ∗ *Ree y* + *Im7 x* ∗ *Im2 y*
Im7 (*x* ∗ *y*) = *Ree x* ∗ *Im7 y* + *Im1 x* ∗ *Im3 y* +*Im2 x* ∗ *Im6 y* − *Im3 x* ∗ *Im1 y* + *Im4 x* ∗ *Im5 y*
−*Im5 x* ∗ *Im4 y* − *Im6 x* ∗ *Im2 y* +*Im7 x* ∗ *Ree y*
⟨*proof*⟩

lemma *octo-distrib-left* :
a ∗ (*b* + *c*) = *a* ∗ *b* + *a* ∗ *c* **for** *a b c* ::*octo*
⟨*proof*⟩

lemma *octo-distrib-right* :
(*b* + *c*) ∗ *a* = *b* ∗ *a* + *c* ∗ *a* **for** *a b c* ::*octo*
⟨*proof*⟩

lemma *multiplicative-norm-octo*: *norm* (*x* ∗ *y*) = *norm x* ∗ *norm y* **for** *x y* ::*octo*
⟨*proof*⟩

lemma *mult-1-right-octo* [*simp*]: *x* ∗ *1* = (*x* :: *octo*)
and *mult-1-left-octo* [*simp*]: *1* ∗ *x* = (*x* :: *octo*)
⟨*proof*⟩

instance *octo* :: *power* ⟨*proof*⟩

lemma *power2-eq-square-octo*: *x* ^ *2* = (*x* ∗ *x* :: *octo*)
⟨*proof*⟩

lemma *octo-product-alternative-left*: *x* ∗ (*x* ∗ *y*) = (*x* ∗ *x* :: *octo*) ∗ *y*
⟨*proof*⟩

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

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

lemma *octo-power-commutes*: $x \mathbin{\hat{}} y * x = x * (x \mathbin{\hat{}} y :: octo)$
⟨*proof*⟩

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

⟨*proof*⟩

lemma *octo-product-nonassociative* :
$\neg(\forall\ x\ y\ z::octo.\ x * (y * z) = (x * y) * z)$
⟨*proof*⟩

2.5 Embedding of the Reals into the Octonions

definition *octo-of-real* :: *real* ⇒ *octo*
where *octo-of-real r* = *scaleR r 1*

definition *octo-of-nat* :: *nat* ⇒ *octo*
where *octo-of-nat r* = *scaleR r 1*

definition *octo-of-int* :: *int* ⇒ *octo*
where *octo-of-int r* = *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*
⟨*proof*⟩

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*
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *scaleR-conv-octo-of-real*: *scaleR r x* = *octo-of-real r* ∗ *x*
⟨*proof*⟩

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

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

lemma *octo-of-real-add* [*simp*]: *octo-of-real* (*x* + *y*) = *octo-of-real x* + *octo-of-real y*
⟨*proof*⟩

lemma *octo-of-real-minus* [*simp*]: *octo-of-real* (− *x*) = − *octo-of-real x*
⟨*proof*⟩

lemma *octo-of-real-diff* [*simp*]: *octo-of-real* (*x* − *y*) = *octo-of-real x* − *octo-of-real y*
⟨*proof*⟩

lemma *octo-of-real-mult* [*simp*]: *octo-of-real* (*x* ∗ *y*) = *octo-of-real x* ∗ *octo-of-real y*
⟨*proof*⟩

lemma *octo-of-real-sum*[*simp*]: *octo-of-real* (*sum f s*) = (∑ *x*∈*s*. *octo-of-real* (*f x*))
⟨*proof*⟩

lemma *octo-of-real-power* [*simp*]:
octo-of-real (*x* ^ *y*) = (*octo-of-real x* :: *octo*) ^ *y*
⟨*proof*⟩

lemma *octo-of-real-eq-iff* [*simp*]: *octo-of-real x* = *octo-of-real y* ⟷ *x* = *y*
⟨*proof*⟩

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*]: −*octo-of-real x* = *octo-of-real y* ⟷ −*x* = *y*
⟨*proof*⟩

lemma *octo-of-real-eq-minus-of-real-iff* [*simp*]: *octo-of-real x* = −*octo-of-real y* ⟷ *x* = −*y*
⟨*proof*⟩

lemma *octo-of-real-of-nat-eq* [*simp*]: *octo-of-real* (*of-nat x*) = *octo-of-nat x*
⟨*proof*⟩

lemma *octo-of-real-of-int-eq* [*simp*]: *octo-of-real* (*of-int z*) = *octo-of-int z*
⟨*proof*⟩

lemma *octo-of-int-of-nat*: *octo-of-int* (*of-nat n*) = *octo-of-nat n*

⟨*proof*⟩

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* ≤ *a* ⟹ *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*
⟨*proof*⟩

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* ≤ *a* ⟹ *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*
⟨*proof*⟩

instance *octo* :: *numeral* ⟨*proof*⟩

lemma *numeral-octo-conv-of-nat*: *numeral x* = *octo-of-nat* (*numeral x*)
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *octo-of-real-numeral* [*simp*]: *octo-of-real* (*numeral w*) = *numeral w*
⟨*proof*⟩

lemma *octo-of-real-neg-numeral* [*simp*]: *octo-of-real* (− *numeral w*) = − *numeral w*
⟨*proof*⟩

lemma *octo-of-real-times-commute*: *octo-of-real r* ∗ *q* = *q* ∗ *octo-of-real r*
⟨*proof*⟩

lemma *octo-of-real-times-conv-scaleR*: *octo-of-real x* ∗ *y* = *scaleR x y*
⟨*proof*⟩

lemma *octo-mult-scaleR-left*: ($r *_R x$) ∗ *y* = $r *_R$ (*x* ∗ *y* :: *octo*)
⟨*proof*⟩

lemma *octo-mult-scaleR-right*: *x* ∗ ($r *_R y$) = $r *_R$ (*x* ∗ *y* :: *octo*)
⟨*proof*⟩

lemma *scaleR-octo-of-real* [*simp*]: *scaleR r* (*octo-of-real s*) = *octo-of-real* (*r* ∗ *s*)
⟨*proof*⟩

lemma *octo-of-real-times-left-commute*: *octo-of-real r* ∗ (*x* ∗ *q*) = *x* ∗ (*octo-of-real r* ∗ *q*)
⟨*proof*⟩

lemma *nonzero-octo-of-real-inverse*:
x ≠ *0* ⟹ *octo-of-real* (*inverse x*) = *inverse* (*octo-of-real x* :: *octo*)
⟨*proof*⟩

lemma *octo-of-real-inverse* [*simp*]:
octo-of-real (*inverse x*) = *inverse* (*octo-of-real x*)
⟨*proof*⟩

lemma *nonzero-octo-of-real-divide*:
y ≠ *0* ⟹ *octo-of-real* (*x* / *y*) = (*octo-of-real x* / *octo-of-real y* ::*octo*)
⟨*proof*⟩

lemma *octo-of-real-divide* [*simp*]:
octo-of-real (*x* / *y*) = (*octo-of-real x* / *octo-of-real y* :: *octo*)
⟨*proof*⟩

lemma *octo-of-real-inverse-collapse* [*simp*]:
assumes *c* ≠ *0*
shows *octo-of-real c* ∗ *octo-of-real* (*inverse c*) = *1*
octo-of-real (*inverse c*) ∗ *octo-of-real c* = *1*
⟨*proof*⟩

lemma *octo-divide-numeral*:
fixes *x*::*octo* **shows** *x* / *numeral y* = *x* $/_R$ *numeral y*
⟨*proof*⟩

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*
⟨*proof*⟩

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*

⟨*proof*⟩

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*]: $e7 \neq 0$
  ⟨*proof*⟩

2.6 "Expansion" into the traditional notation

lemma *octo-unfold*:
$q = (Ree\ q) *_R e0 + (Im1\ q) *_R e1 + (Im2\ q) *_R e2 + (Im3\ q) *_R e3 + (Im4\ q) *_R e4 + (Im5\ q) *_R e5 + (Im6\ q) *_R e6 + (Im7\ q) *_R e7$
  ⟨*proof*⟩

lemma *octo-trad*: $Octo\ x\ y\ z\ w\ u\ v\ q\ g =$
$x *_R e0 + y *_R e1 + z *_R e2 + w *_R e3 + u *_R e4 + v *_R e5 + q *_R e6 + g *_R e7$
  ⟨*proof*⟩

lemma *octo-of-real-eq-Octo*: *octo-of-real a = Octo a 0 0 0 0 0 0 0*
  ⟨*proof*⟩

lemma *e1-squared* [*simp*]: $e1 \,\hat{}\, 2 = -1$
  and *e2-squared* [*simp*]: $e2 \,\hat{}\, 2 = -1$
  and *e3-squared* [*simp*]: $e3 \,\hat{}\, 2 = -1$
  and *e4-squared* [*simp*]: $e4 \,\hat{}\, 2 = -1$
  and *e5-squared* [*simp*]: $e5 \,\hat{}\, 2 = -1$
  and *e6-squared* [*simp*]: $e6 \,\hat{}\, 2 = -1$
  and *e7-squared* [*simp*]: $e7 \,\hat{}\, 2 = -1$
  ⟨*proof*⟩

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*]: $inverse\ e7 = -e7$
  ⟨*proof*⟩

2.7 Conjugate of an octonion and related properties.

primcorec *cnj* :: *octo* ⇒ *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* ⟷ *x* = *y*

⟨*proof*⟩

lemma *cnj-cnj* [*simp*]:
cnj(*cnj q*) = *q*
⟨*proof*⟩

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

lemma *cnj-zero* [*simp*]: *cnj 0* = *0*
⟨*proof*⟩

lemma *cnj-zero-iff* [*iff*]: *cnj z* = *0* ⟷ *z* = *0*
⟨*proof*⟩

lemma *cnj-one* [*simp*]: *cnj 1* = *1*
⟨*proof*⟩

lemma *cnj-one-iff* [*simp*]: *cnj z* = *1* ⟷ *z* = *1*
⟨*proof*⟩

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

lemma *cnj-add* [*simp*]: *cnj* (*x* + *y*) = *cnj x* + *cnj y*
⟨*proof*⟩

lemma *cnj-sum* [*simp*]: *cnj* (*sum f S*) = (∑ *x*∈*S*. *cnj* (*f x*))
⟨*proof*⟩

lemma *cnj-diff* [*simp*]: *cnj* (*x* − *y*) = *cnj x* − *cnj y*
⟨*proof*⟩

lemma *cnj-minus* [*simp*]: *cnj* (− *x*) = − *cnj x*
⟨*proof*⟩

lemma *cnj-inverse* [*simp*]: *cnj* (*inverse x*) = *inverse* (*cnj x*) **for** *x y* ::*octo*
⟨*proof*⟩

lemma *cnj-mult* [*simp*]: *cnj* (*x* ∗ *y*) = *cnj y* ∗ *cnj x* **for** *x y* ::*octo*

⟨*proof*⟩

lemma *cnj-divide* [*simp*]: *cnj* (*x* / *y*) = (*inverse* (*cnj y*)) ∗ *cnj x*
for *x y* ::*octo*
⟨*proof*⟩

lemma *cnj-power* [*simp*]: *cnj* (*x* ^ *y*) = (*cnj x*) ^ *y* **for** *x*::*octo*
⟨*proof*⟩

lemma *cnj-of-nat* [*simp*]: *cnj* (*octo-of-nat x*) = *octo-of-nat*(*x*)
⟨*proof*⟩

lemma *cnj-of-int* [*simp*]: *cnj* (*octo-of-int x*) = *octo-of-nat*(*x*)
⟨*proof*⟩

lemma *cnj-numeral* [*simp*]: *cnj* (*numeral x*) = *numeral x*
⟨*proof*⟩

lemma *cnj-neg-numeral* [*simp*]: *cnj* (− *numeral x*) = − *numeral x*
⟨*proof*⟩

lemma *cnj-scaleR* [*simp*]: *cnj* (*scaleR r x*) = *scaleR r* (*cnj x*)
⟨*proof*⟩

lemma *cnj-units* [*simp*]: *cnj e1* = −*e1 cnj e2* = −*e2 cnj e3* = −*e3*
cnj e4 = −*e4* *cnj e5* = −*e5* *cnj e6* = −*e6* *cnj e7* = −*e7*
⟨*proof*⟩

lemma *cnj-eq-of-real*: *cnj q* = *octo-of-real x* ⟷ *q* = *octo-of-real x*
⟨*proof*⟩

lemma *octo-trad-cnj* : *cnj q* = (*Ree 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* **for** *q*::*octo*
⟨*proof*⟩

lemma *octonion-conjugate-property*:
cnj x = −(*1/6*) $*_R$ (*x* + (*e1* ∗ *x*) ∗ *e1* + (*e2* ∗ *x*) ∗ *e2* + (*e3* ∗ *x*) ∗ *e3* +
(*e4* ∗ *x*) ∗ *e4* + (*e5* ∗ *x*) ∗ *e5* + (*e6* ∗ *x*) ∗ *e6* + (*e7* ∗ *x*) ∗ *e7*)
⟨*proof*⟩

lemma *octo-add-cnj*: *q* + *cnj q* = *2* $*_R$ (*Ree q*) $*_R$ *e0 cnj q* + *q* = (*2*$*_R$ (*Ree q*)$*_R$ *e0*)
⟨*proof*⟩

lemma *octo-add-cnj1*: *q* + *cnj q* = *octo-of-real* (*2*$*_R$ (*Ree q*))
cnj q + *q* = *octo-of-real* (*2*$*_R$ (*Ree q*))
⟨*proof*⟩

lemma *octo-subtract-cnj*:
$q - cnj\ q = 2 *_R (Im1\ q *_R e1 + Im2\ q *_R e2 + Im3\ q *_R e3 +$
$Im4\ q *_R e4 + Im5\ q *_R e5 + Im6\ q *_R e6 + Im7\ q *_R e7)$
⟨*proof*⟩

lemma *octo-mult-cnj-commute*: $cnj\ x * x = x * cnj\ x$
⟨*proof*⟩

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

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

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

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

lemma *octo-inverse-cnj*: $inverse\ x = cnj\ x\ /_R\ (norm\ x\ \widehat{}\ 2)$
⟨*proof*⟩

lemma *inverse-octo-1*: $x \neq 0 \Longrightarrow x * inverse\ x = (1 :: octo)$
⟨*proof*⟩

lemma *inverse-octo-1-sym*: $x \neq 0 \Longrightarrow inverse\ x * x = (1 :: octo)$
⟨*proof*⟩

lemma *inverse-0-octo* [*simp*]: $inverse\ 0 = (0 :: octo)$
⟨*proof*⟩

lemma *inverse-octo-commutes*: $inverse\ x * x = x * (inverse\ x :: octo)$
⟨*proof*⟩

lemma *octo-inverse-mult*: $inverse\ (x * y) = inverse\ y * inverse\ x$ **for** $x\ y{::}octo$
⟨*proof*⟩

lemma *octo-inverse-eq-cnj*: $norm\ q = 1 \Longrightarrow inverse\ q = cnj\ q$ **for** $q{::}octo$
⟨*proof*⟩

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

lemma *octo-in-Reals-if-Re-con*: **assumes** $Ree\ (octo\text{-}of\text{-}real\ q) = q$
shows $q \in Reals$
⟨*proof*⟩

lemma *octo-in-Reals-if-cnj*: **fixes** *q*:: *real* **shows** *cnj*(*octo-of-real*(*q*)) = *octo-of-real* *q*
⟨*proof*⟩

lemma *octo-in-Reals-if-cnj-con*: **assumes** *cnj*(*octo-of-real*(*q*)) = *octo-of-real q*
shows *q* ∈ *Reals*
⟨*proof*⟩

lemma *norm-power2*: *norm q* ^ *2* = *Ree* (*cnj q* ∗ *q*)
⟨*proof*⟩

lemma *norm-power2-cnj*: *norm q* ^ *2* = *Ree* (*q* ∗ *cnj q*)
⟨*proof*⟩

lemma *octo-norm-imaginary*: *Ree x* = *0* ⟹ *x* ∗ *x* = −(*octo-of-real* (*norm x*))^2
⟨*proof*⟩

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*
⟨*proof*⟩

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*]: *Ree* (*sum f S*) = *sum* ($\lambda x.$ *Ree*(*f x*)) *S*
and *Im1-sum* [*simp*]: *Im1* (*sum f S*) = *sum* ($\lambda x.$ *Im1* (*f x*)) *S*
and *Im2-sum* [*simp*]: *Im2* (*sum f S*) = *sum* ($\lambda x.$ *Im2* (*f x*)) *S*
and *Im3-sum* [*simp*]: *Im3* (*sum f S*) = *sum* ($\lambda x.$ *Im3* (*f x*)) *S*
and *Im4-sum* [*simp*]: *Im4* (*sum f S*) = *sum* ($\lambda x.$ *Im4* (*f x*)) *S*
and *Im5-sum* [*simp*]: *Im5* (*sum f S*) = *sum* ($\lambda x.$ *Im5* (*f x*)) *S*
and *Im6-sum* [*simp*]: *Im6* (*sum f S*) = *sum* ($\lambda x.$ *Im6* (*f x*)) *S*
and *Im7-sum* [*simp*]: *Im7* (*sum f S*) = *sum* ($\lambda x.$ *Im7* (*f x*)) *S*
⟨*proof*⟩

2.8.2 Bound results for real and imaginary components of limits.

lemma *Ree-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. octo.Ree* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Ree limit* ≤ *b*
⟨*proof*⟩

lemma *Im1-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. Im1* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Im1 limit* ≤ *b*
⟨*proof*⟩

lemma *Im2-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. Im2* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Im2 limit* ≤ *b*
⟨*proof*⟩

lemma *Im3-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. Im3* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Im3 limit* ≤ *b*
⟨*proof*⟩

lemma *Im4-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. Im4* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Im4 limit* ≤ *b*
⟨*proof*⟩

lemma *Im5-tendsto-upperbound*:
⟦(*f* ⟶ *limit*) *net*; $\forall_F$ *x in net. Im5* (*f x*) ≤ *b*; *net* ≠ *bot*⟧ ⟹ *Im5 limit* ≤ *b*
⟨*proof*⟩

lemma *Im6-tendsto-upperbound*:

⟦(f ⟶ limit) net; ∀$_F$ x in net. Im6 (f x) ≤ b; net ≠ bot⟧ ⟹ Im6 limit ≤ b
⟨proof⟩

lemma *Im7-tendsto-upperbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. Im7 (f x) ≤ b; net ≠ bot⟧ ⟹ Im7 limit ≤ b
⟨proof⟩

lemma *Ree-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ octo.Ree (f x); net ≠ bot⟧ ⟹ b ≤ Ree limit
⟨proof⟩

lemma *Im1-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im1 (f x); net ≠ bot⟧ ⟹ b ≤ Im1 limit
⟨proof⟩

lemma *Im2-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im2 (f x); net ≠ bot⟧ ⟹ b ≤ Im2 limit
⟨proof⟩

lemma *Im3-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im3 (f x); net ≠ bot⟧ ⟹ b ≤ Im3 limit
⟨proof⟩

lemma *Im4-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im4 (f x); net ≠ bot⟧ ⟹ b ≤ Im4 limit
⟨proof⟩

lemma *Im5-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im5 (f x); net ≠ bot⟧ ⟹ b ≤ Im5 limit
⟨proof⟩

lemma *Im6-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im6 (f x); net ≠ bot⟧ ⟹ b ≤ Im6 limit
⟨proof⟩

lemma *Im7-tendsto-lowerbound*:
⟦(f ⟶ limit) net; ∀$_F$ x in net. b ≤ Im7 (f x); net ≠ bot⟧ ⟹ b ≤ Im7 limit
⟨proof⟩

lemma *octo-of-real-continuous* [*continuous-intros*]:
continuous net f ⟹ continuous net (λx. octo-of-real (f x))
⟨proof⟩

lemma *octo-of-real-continuous-on* [*continuous-intros*]:
continuous-on S f ⟹ continuous-on S (λx. octo-of-real (f x))
⟨proof⟩

lemma *of-real-continuous-iff*: continuous net (λx. octo-of-real (f x)) ⟷ contin-

uous net f
⟨*proof*⟩

lemma *of-real-continuous-on-iff*:
continuous-on S (λx. *octo-of-real*(*f x*)) ⟷ *continuous-on S f*
⟨*proof*⟩

2.9 Octonions for describing 7D isometries

2.9.1 The *HIm* operator

definition *HIm* :: *octo* ⇒ *real*^*7* **where**
HIm q ≡ *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*]
⟨*proof*⟩

lemma *him-eq*: *HIm a* = *HIm 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*

⟨*proof*⟩

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

lemma *him-0* [*simp*]: *HIm 0* = *0*
⟨*proof*⟩

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

lemma *him-cnj*: *HIm*(*cnj q*) = − *HIm q*
⟨*proof*⟩

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

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

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*
⟨*proof*⟩

lemma *him-sum* [*simp*]: *HIm* (*sum f S*) = ($\sum$ *x*∈*S*. *HIm* (*f x*))
⟨*proof*⟩

lemma *linear-him*: *linear HIm*

⟨proof⟩

2.9.2 The *Hv* operator

definition *Hv* :: *real^7* ⇒ *octo* **where**
Hv v ≡ *Octo 0 (v$1) (v$2) (v$3) (v$4) (v$5) (v$6) (v$7)*

lemma *Hv-sel* [*simp*]:
Ree (Hv v) = 0 Im1 (Hv v) = v $ 1 Im2 (Hv v) = v $ 2 Im3 (Hv v) = v $ 3
Im4 (Hv v) = v $ 4 Im5 (Hv v) = v $ 5 Im6 (Hv v) = v $ 6 Im7 (Hv v) = v $ 7
⟨proof⟩

lemma *hv-vec*: *Hv(vec r) = Octo 0 r r r r r r r*
⟨proof⟩

lemma *hv-eq-zero* [*simp*]: *Hv v = 0 ⟷ v = 0*
⟨proof⟩

lemma *hv-zero* [*simp*]: *Hv 0 = 0*
⟨proof⟩

lemma *hv-vector* [*simp*]: *Hv(vector[x,y,z,u,v,q,g]) = Octo 0 x y z u v q g*
⟨proof⟩

lemma *hv-basis*:
Hv(axis 1 1) = e1 Hv(axis 2 1) = e2 Hv(axis 3 1) = e3
Hv(axis 4 1) = e4 Hv(axis 5 1) = e5 Hv(axis 6 1) = e6 Hv(axis 7 1) = e7
⟨proof⟩

lemma *hv-add* [*simp*]: *Hv(x + y) = Hv x + Hv y*
⟨proof⟩

lemma *hv-minus* [*simp*]: *Hv(−x) = −Hv x*
⟨proof⟩

lemma *hv-diff* [*simp*]: *Hv(x − y) = Hv x − Hv y*
⟨proof⟩

lemma *hv-cmult* [*simp*]:
Hv(scaleR a x) = scaleR a (Hv x)
⟨proof⟩

lemma *hv-sum* [*simp*]: *Hv (sum f S) = (∑x∈S. Hv (f x))*
⟨proof⟩

lemma *hv-inj*: *Hv x = Hv y ⟷ x = y*
⟨proof⟩

lemma *linear-hv*: *linear Hv*

⟨*proof*⟩

lemma *him-hv* [*simp*]: $HIm(Hv\ x) = x$
⟨*proof*⟩

lemma *cnj-hv* [*simp*]: $cnj(Hv\ v) = -Hv\ v$
⟨*proof*⟩

lemma *hv-him*: $Hv(HIm\ q) = Octo\ 0\ (Im1\ q)\ (Im2\ q)\ (Im3\ q)\ \ (Im4\ q)\ (Im5\ q)\ (Im6\ q)\ \ (Im7\ q)$
⟨*proof*⟩

lemma *hv-him-eq*: $Hv(HIm\ q) = q \longleftrightarrow Ree\ q = 0$
⟨*proof*⟩

lemma *dot-hv* [*simp*]: $Hv\ u \cdot Hv\ v = u \cdot v$
⟨*proof*⟩

lemma *norm-hv* [*simp*]: $norm\ (Hv\ v) = norm\ v$
⟨*proof*⟩

2.9.3 Related basic identities

lemma *mult-hv-eq-cross-dot*: $Hv\ x * Hv\ y = Hv(x\ \ \times_7\ y) - octo\text{-}of\text{-}real\ (inner\ x\ y)$
⟨*proof*⟩

lemma *octonion-identity1-cross7* :
$Hv\ (x \times_7\ y) = (1/2) *_R\ (Hv\ x * Hv\ y - Hv\ y * Hv\ x)$
⟨*proof*⟩

lemma *octonion-identity2-cross7*:
$Hv\ (x \times_7\ (y \times_7\ z) + y \times_7\ (z \times_7\ x) + z \times_7\ (x \times_7\ y)) =$
$-(3/2) *_R\ ((Hv\ x * Hv\ y) * Hv\ z - Hv\ x * (Hv\ y * Hv\ z))$
⟨*proof*⟩

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

lemma *HIm-nth* [*simp*]:
$HIm\ x\ \$\ 1 = Im1\ x\ HIm\ x\ \$\ 2 = Im2\ x\ HIm\ x\ \$\ 3 = Im3\ x\ \ HIm\ x\ \$\ 4 = Im4\ x$
$HIm\ x\ \$\ 5 = Im5\ x\ HIm\ x\ \$\ 6 = Im6\ x\ HIm\ x\ \$\ 7 = Im7\ x$
⟨*proof*⟩

lemma *orthogonal-transformation-octo-congruence*:
assumes $norm\ q = 1$
shows $orthogonal\text{-}transformation\ (\lambda x.\ HIm(cnj\ q * Hv\ x * q))$
⟨*proof*⟩

lemma *orthogonal-transformation-octo-conjugation*:
  assumes $q \neq 0$
  shows *orthogonal-transformation* (*λx. HIm* (*inverse q* ∗ *Hv x* ∗ *q*))
⟨*proof*⟩

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.