

Vector Calculus

Brian Huffman

August 27, 2008

Abstract

This development ...

Contents

| | | |
|----------|--|----------|
| 1 | Constructing Bounded Linear Operators | 3 |
| 2 | Frechet Derivative | 4 |
| 2.1 | Addition | 4 |
| 2.2 | Subtraction | 5 |
| 2.3 | Continuity | 5 |
| 2.4 | Composition | 5 |
| 2.5 | Product Rule | 5 |
| 2.6 | Powers | 6 |
| 2.7 | Inverse | 6 |
| 2.8 | Alternate definition | 6 |
| 3 | Inner Product Spaces | 6 |
| 3.1 | Real inner product spaces | 6 |
| 3.2 | Instances | 8 |
| 4 | Finite-Dimensional Vectors | 9 |
| 4.1 | Type definition | 9 |
| 4.2 | Vector arithmetic | 9 |
| 4.3 | Vector is a real vector space | 10 |
| 4.4 | Square root of sum of squares | 11 |
| 4.5 | Vectors are a real normed vector space | 12 |
| 4.6 | Vectors are an inner product space | 13 |
| 4.7 | Vector is a functor | 14 |
| 4.8 | Vector dot product | 15 |
| 4.9 | Single-component vectors | 15 |

| | | |
|----------|---|-----------|
| 5 | Square Matrices | 16 |
| 5.1 | Matrix arithmetic | 17 |
| 5.2 | Smatrix multiplication | 18 |
| 5.3 | Numeral syntax for matrices | 19 |
| 5.4 | Matrices are a real vector space | 19 |
| 5.5 | Applying a matrix to a vector | 20 |
| 5.6 | Matrix norm | 21 |
| 5.7 | Vector outer product | 23 |
| 6 | Gradient Derivatives | 24 |
| 6.1 | Gradient Derivative | 24 |
| 7 | Pairs as Vector Spaces | 26 |
| 7.1 | Vector arithmetic | 26 |
| 7.2 | Product is a normed vector space | 29 |
| 7.3 | Product is an inner product space | 29 |
| 8 | Cross Products | 29 |
| 9 | Quaternions | 30 |
| 9.1 | Definition | 31 |
| 9.2 | Addition and Subtraction | 31 |
| 9.3 | Multiplication | 31 |
| 9.4 | Vector space | 32 |
| 9.5 | Norm and inner product | 32 |
| 9.6 | Conjugate | 33 |
| 9.7 | Inverse | 33 |

1 Constructing Bounded Linear Operators

theory *BoundedLinear*

imports *Lim*

begin

lemma *bounded-linear-zero*: *bounded-linear* ($\lambda x. 0$)

<proof>

lemma *bounded-linear-ident*: *bounded-linear* ($\lambda x. x$)

<proof>

lemma *bounded-linear-uminus*: *bounded-linear* *uminus*

<proof>

lemma *bounded-linear-compose*:

includes *bounded-linear* *f*

includes *bounded-linear* *g*

shows *bounded-linear* ($\lambda x. f (g x)$)

<proof>

lemma *bounded-linear-o*:

$\llbracket \textit{bounded-linear } f; \textit{ bounded-linear } g \rrbracket \implies \textit{bounded-linear } (f \circ g)$

<proof>

lemma *bounded-linear-add*:

includes *bounded-linear* *f*

includes *bounded-linear* *g*

shows *bounded-linear* ($\lambda x. f x + g x$)

<proof>

lemma *bounded-linear-minus*:

bounded-linear *f* $\implies \textit{bounded-linear } (\lambda x. - f x)$

<proof>

lemma *bounded-linear-diff*:

$\llbracket \textit{bounded-linear } f; \textit{ bounded-linear } g \rrbracket \implies \textit{bounded-linear } (\lambda x. f x - g x)$

<proof>

lemma *bounded-linear-setsum*:

$\forall i \in A. \textit{bounded-linear } (f i) \implies \textit{bounded-linear } (\lambda x. \sum_{i \in A} f i x)$

<proof>

lemmas *bounded-linear-scaleR* =

scaleR.bounded-linear-right [*THEN* *bounded-linear-compose*]

lemmas *bounded-linear-mult-const* =

mult.bounded-linear-left [*THEN* *bounded-linear-compose*]

```

lemmas bounded-linear-const-mult =
  mult.bounded-linear-right [THEN bounded-linear-compose]

```

```

end

```

2 Frechet Derivative

```

theory FrechetDeriv
imports BoundedLinear
begin

```

```

definition

```

```

  fderiv ::
    ['a::real-normed-vector ⇒ 'b::real-normed-vector, 'a, 'a ⇒ 'b] ⇒ bool
    — Frechet derivative: D is derivative of function f at x
      ((FDERIV (-)/ (-)/ :> (-)) [1000, 1000, 60] 60) where
    FDERIV f x :> D = (bounded-linear D ∧
      (λh. norm (f (x + h) - f x - D h) / norm h) -- 0 --> 0)

```

```

lemma FDERIV-I:

```

```

  [[bounded-linear D; (λh. norm (f (x + h) - f x - D h) / norm h) -- 0 -->
  0]]
  ⇒ FDERIV f x :> D
  <proof>

```

```

lemma FDERIV-D:

```

```

  FDERIV f x :> D ⇒ (λh. norm (f (x + h) - f x - D h) / norm h) -- 0
  --> 0
  <proof>

```

```

lemma FDERIV-bounded-linear: FDERIV f x :> D ⇒ bounded-linear D

```

```

  <proof>

```

```

lemma FDERIV-const: FDERIV (λx. k) x :> (λh. 0)

```

```

  <proof>

```

```

lemma FDERIV-ident: FDERIV (λx. x) x :> (λh. h)

```

```

  <proof>

```

2.1 Addition

```

lemma norm-ratio-ineq:

```

```

  fixes x y :: 'a::real-normed-vector
  fixes h :: 'b::real-normed-vector
  shows norm (x + y) / norm h ≤ norm x / norm h + norm y / norm h
  <proof>

```

```

lemma FDERIV-add:

```

assumes $f: FDERIV\ f\ x\ :>\ F$
assumes $g: FDERIV\ g\ x\ :>\ G$
shows $FDERIV\ (\lambda x. f\ x + g\ x)\ x\ :>\ (\lambda h. F\ h + G\ h)$
 <proof>

2.2 Subtraction

lemma *FDERIV-minus*:
 $FDERIV\ f\ x\ :>\ F \implies FDERIV\ (\lambda x. -\ f\ x)\ x\ :>\ (\lambda h. -\ F\ h)$
 <proof>

lemma *FDERIV-diff*:
 $\llbracket FDERIV\ f\ x\ :>\ F; FDERIV\ g\ x\ :>\ G \rrbracket$
 $\implies FDERIV\ (\lambda x. f\ x - g\ x)\ x\ :>\ (\lambda h. F\ h - G\ h)$
 <proof>

2.3 Continuity

lemma *FDERIV-isCont*:
assumes $f: FDERIV\ f\ x\ :>\ F$
shows $isCont\ f\ x$
 <proof>

2.4 Composition

lemma *real-divide-cancel-lemma*:
fixes $a\ b\ c :: 'a::\{field, division-by-zero\}$
shows $(b = 0 \implies a = 0) \implies (a / b) * (b / c) = a / c$
 <proof>

lemma *FDERIV-compose*:
fixes $f :: 'a::real-normed-vector \Rightarrow 'b::real-normed-vector$
fixes $g :: 'b::real-normed-vector \Rightarrow 'c::real-normed-vector$
assumes $f: FDERIV\ f\ x\ :>\ F$
assumes $g: FDERIV\ g\ (f\ x)\ :>\ G$
shows $FDERIV\ (\lambda x. g\ (f\ x))\ x\ :>\ (\lambda h. G\ (F\ h))$
 <proof>

2.5 Product Rule

lemma (in *bounded-bilinear*) *FDERIV-lemma*:
 $a' ** b' - a ** b - (a ** B + A ** b)$
 $= a ** (b' - b - B) + (a' - a - A) ** b' + A ** (b' - b)$
 <proof>

lemma (in *bounded-bilinear*) *FDERIV*:
fixes $x :: 'd::real-normed-vector$
assumes $f: FDERIV\ f\ x\ :>\ F$
assumes $g: FDERIV\ g\ x\ :>\ G$
shows $FDERIV\ (\lambda x. f\ x ** g\ x)\ x\ :>\ (\lambda h. f\ x ** G\ h + F\ h ** g\ x)$

<proof>

lemmas *FDERIV-mult* = *mult.FDERIV*

lemmas *FDERIV-scaleR* = *scaleR.FDERIV*

2.6 Powers

lemma *FDERIV-power-Suc*:

fixes $x :: 'a::\{\text{real-normed-algebra}, \text{recpower}, \text{comm-ring-1}\}$

shows *FDERIV* $(\lambda x. x \wedge \text{Suc } n) x :=> (\lambda h. (1 + \text{of-nat } n) * x \wedge n * h)$

<proof>

lemma *FDERIV-power*:

fixes $x :: 'a::\{\text{real-normed-algebra}, \text{recpower}, \text{comm-ring-1}\}$

shows *FDERIV* $(\lambda x. x \wedge n) x :=> (\lambda h. \text{of-nat } n * x \wedge (n - 1) * h)$

<proof>

2.7 Inverse

lemma *FDERIV-inverse*:

fixes $x :: 'a::\text{real-normed-div-algebra}$

assumes $x: x \neq 0$

shows *FDERIV inverse* $x :=> (\lambda h. - (inverse\ x * h * inverse\ x))$

(**is** *FDERIV ?inv - :=> -*)

<proof>

2.8 Alternate definition

lemma *field-fderiv-def*:

fixes $x :: 'a::\text{real-normed-field}$ **shows**

FDERIV f $x :=> (\lambda h. h * D) = (\lambda h. (f (x + h) - f x) / h) \text{ --- } 0 \text{ ---} > D$

<proof>

end

3 Inner Product Spaces

theory *InnerProduct*

imports *Complex FrechetDeriv*

begin

3.1 Real inner product spaces

class *inner* = *type* +

fixes $inner :: 'a \Rightarrow 'a \Rightarrow \text{real}$

class *real-inner* = *real-vector* + *inner* + *sgn-div-norm* +

assumes *inner-commute*: $\text{inner } x \ y = \text{inner } y \ x$
and *inner-left-distrib*: $\text{inner } (x + y) \ z = \text{inner } x \ z + \text{inner } y \ z$
and *inner-scaleR-left*: $\text{inner } (\text{scaleR } r \ x) \ y = r * (\text{inner } x \ y)$
and *inner-ge-zero*: $0 \leq \text{inner } x \ x$
and *inner-eq-zero*: $(\text{inner } x \ x = 0) = (x = 0)$
and *norm-eq-sqrt-inner*: $\text{norm } x = \text{sqrt } (\text{inner } x \ x)$

lemma *inner-right-distrib*:
fixes $x \ y \ z :: 'a::\text{real-inner}$
shows $\text{inner } x \ (y + z) = \text{inner } x \ y + \text{inner } x \ z$
<proof>

lemma *inner-scaleR-right*:
fixes $x \ y \ z :: 'a::\text{real-inner}$
shows $\text{inner } x \ (\text{scaleR } r \ y) = r * (\text{inner } x \ y)$
<proof>

interpretation *inner-left*: *additive* $[(\lambda x::'a::\text{real-inner}. \text{inner } x \ y)]$
<proof>

interpretation *inner-right*: *additive* $[(\lambda y::'a::\text{real-inner}. \text{inner } x \ y)]$
<proof>

lemmas *inner-zero-left* [simp] = *inner-left.zero*
lemmas *inner-zero-right* [simp] = *inner-right.zero*
lemmas *inner-left-diff-distrib* = *inner-left.diff*
lemmas *inner-right-diff-distrib* = *inner-right.diff*

lemmas *inner-distrib* = *inner-left-distrib inner-right-distrib*
lemmas *inner-diff* = *inner-left-diff-distrib inner-right-diff-distrib*
lemmas *inner-scaleR* = *inner-scaleR-left inner-scaleR-right*

lemma *zero-less-inner-iff*: $(0 < \text{inner } x \ x) = (x \neq (0::'a::\text{real-inner}))$
<proof>

lemma *Cauchy-Schwartz-ineq*:
fixes $x \ y :: 'a::\text{real-inner}$
shows $(\text{inner } x \ y)^2 \leq \text{inner } x \ x * \text{inner } y \ y$
<proof>

lemma *norm-squared-eq-inner*: $(\text{norm } x)^2 = \text{inner } x \ (x::'a::\text{real-inner})$
<proof>

lemma *Cauchy-Schwartz-ineq2*:
fixes $x \ y :: 'a::\text{real-inner}$
shows $|\text{inner } x \ y| \leq \text{norm } x * \text{norm } y$
<proof>

instance *real-inner* \subseteq *real-normed-vector*

<proof>

interpretation *inner*:

bounded-bilinear [*inner*::'*a*::*real-inner* \Rightarrow '*a* \Rightarrow *real*]

<proof>

interpretation *inner-left*:

bounded-linear [$\lambda x::'a::real-inner. inner\ x\ y$]

<proof>

interpretation *inner-right*:

bounded-linear [$\lambda y::'a::real-inner. inner\ x\ y$]

<proof>

3.2 Instances

instantiation *real* :: *real-inner*

begin

definition

inner-real-def [*simp*]: *inner* = *op* *

instance

<proof>

end

lemma *add-nonneg-eq-0-iff*:

fixes *x y* :: '*a*::*pordered-ab-group-add*

assumes *x*: $0 \leq x$ **and** *y*: $0 \leq y$

shows $x + y = 0 \iff x = 0 \wedge y = 0$

<proof>

instantiation *complex* :: *real-inner*

begin

definition

inner-complex-def: $inner\ x\ y = Re\ x * Re\ y + Im\ x * Im\ y$

instance

<proof>

end

end

4 Finite-Dimensional Vectors

```
theory VectorType
imports InnerProduct
begin
```

4.1 Type definition

```
typedef (open) ('a,'n) vec (infixl ^ 80) = UNIV :: ('n  $\Rightarrow$  'a) set <proof>
```

```
declare Abs-vec-inverse [simplified, iff]
```

```
lemma expand-vec-eq:
```

```
   $x = y \iff (\forall i. \text{Rep-vec } x \ i = \text{Rep-vec } y \ i)$ 
  <proof>
```

```
lemma vec-ext:
```

```
   $(\bigwedge i. \text{Rep-vec } x \ i = \text{Rep-vec } y \ i) \implies x = y$ 
  <proof>
```

4.2 Vector arithmetic

```
instantiation vec :: (zero, type) zero
begin
```

```
definition
```

```
  zero-vec-def:  $0 = \text{Abs-vec } (\lambda i. 0)$ 
```

```
instance <proof>
end
```

```
instantiation vec :: (plus, type) plus
begin
```

```
definition
```

```
  plus-vec-def:  $x + y = \text{Abs-vec } (\lambda i. \text{Rep-vec } x \ i + \text{Rep-vec } y \ i)$ 
```

```
instance <proof>
end
```

```
instantiation vec :: (minus, type) minus
begin
```

```
definition
```

```
  minus-vec-def:  $x - y = \text{Abs-vec } (\lambda i. \text{Rep-vec } x \ i - \text{Rep-vec } y \ i)$ 
```

```
instance <proof>
end
```

```
instantiation vec :: (uminus, type) uminus
```

begin

definition

uminus-vec-def: $- x = \text{Abs-vec } (\lambda i. - \text{Rep-vec } x \ i)$

instance $\langle \text{proof} \rangle$

end

lemma *Rep-vec-zero* [*simp*]: $\text{Rep-vec } 0 \ i = 0$

$\langle \text{proof} \rangle$

lemma *Rep-vec-add* [*simp*]:

$\text{Rep-vec } (x + y) \ i = \text{Rep-vec } x \ i + \text{Rep-vec } y \ i$

$\langle \text{proof} \rangle$

lemma *Rep-vec-diff* [*simp*]:

$\text{Rep-vec } (x - y) \ i = \text{Rep-vec } x \ i - \text{Rep-vec } y \ i$

$\langle \text{proof} \rangle$

lemma *Rep-vec-uminus* [*simp*]:

$\text{Rep-vec } (- x) \ i = - \text{Rep-vec } x \ i$

$\langle \text{proof} \rangle$

Addition is associative

instance *vec* :: (*semigroup-add*, *type*) *semigroup-add*

$\langle \text{proof} \rangle$

Addition is commutative

instance *vec* :: (*ab-semigroup-add*, *type*) *ab-semigroup-add*

$\langle \text{proof} \rangle$

Zero is additive identity

instance *vec* :: (*comm-monoid-add*, *type*) *comm-monoid-add*

$\langle \text{proof} \rangle$

Addition has cancellation property

instance *vec* :: (*cancel-semigroup-add*, *type*) *cancel-semigroup-add*

$\langle \text{proof} \rangle$

instance *vec* :: (*cancel-ab-semigroup-add*, *type*) *cancel-ab-semigroup-add*

$\langle \text{proof} \rangle$

Negation is additive inverse

instance *vec* :: (*ab-group-add*, *type*) *ab-group-add*

$\langle \text{proof} \rangle$

4.3 Vector is a real vector space

instantiation *vec* :: (*scaleR*, *type*) *scaleR*

begin

definition

scaleR-vec-def: $\text{scaleR } r \ x = \text{Abs-vec } (\lambda i. \text{scaleR } r \ (\text{Rep-vec } x \ i))$

instance $\langle \text{proof} \rangle$

end

lemma *Rep-vec-scaleR* [*simp*]:

$\text{Rep-vec } (\text{scaleR } r \ x) \ i = \text{scaleR } r \ (\text{Rep-vec } x \ i)$
 $\langle \text{proof} \rangle$

instance *vec* :: $(\text{real-vector}, \text{type}) \ \text{real-vector}$

$\langle \text{proof} \rangle$

4.4 Square root of sum of squares

definition

$\text{set-l2 } f \ A = \text{sqrt } (\sum_{i \in A}. (f \ i)^2)$

lemma *set-l2-infinite* [*simp*]: $\neg \text{finite } A \implies \text{set-l2 } f \ A = 0$

$\langle \text{proof} \rangle$

lemma *set-l2-empty* [*simp*]: $\text{set-l2 } f \ \{\} = 0$

$\langle \text{proof} \rangle$

lemma *set-l2-insert* [*simp*]:

$\llbracket \text{finite } F; a \notin F \rrbracket \implies$
 $\text{set-l2 } f \ (\text{insert } a \ F) = \text{sqrt } ((f \ a)^2 + (\text{set-l2 } f \ F)^2)$

$\langle \text{proof} \rangle$

lemma *set-l2-nonneg* [*simp*]: $0 \leq \text{set-l2 } f \ A$

$\langle \text{proof} \rangle$

lemma *set-l2-0'*: $\forall a \in A. f \ a = 0 \implies \text{set-l2 } f \ A = 0$

$\langle \text{proof} \rangle$

lemma *set-l2-mono*:

assumes $\bigwedge i. i \in K \implies f \ i \leq g \ i$

assumes $\bigwedge i. i \in K \implies 0 \leq f \ i$

shows $\text{set-l2 } f \ K \leq \text{set-l2 } g \ K$

$\langle \text{proof} \rangle$

lemma *set-l2-right-distrib*:

$0 \leq r \implies r * \text{set-l2 } f \ A = \text{set-l2 } (\lambda x. r * f \ x) \ A$

$\langle \text{proof} \rangle$

lemma *set-l2-left-distrib*:

$0 \leq r \implies \text{set-l2 } f \ A * r = \text{set-l2 } (\lambda x. f \ x * r) \ A$

<proof>

lemma *setsum-nonneg-eq-0-iff*:

fixes $f :: 'a \Rightarrow 'b :: \text{pordered-ab-group-add}$

shows $\llbracket \text{finite } A; \forall x \in A. 0 \leq f x \rrbracket \implies \text{setsum } f A = 0 \longleftrightarrow (\forall x \in A. f x = 0)$

<proof>

lemma *set-l2-eq-0-iff*: $\text{finite } A \implies \text{set-l2 } f A = 0 \longleftrightarrow (\forall x \in A. f x = 0)$

<proof>

lemma *set-l2-triangle-ineq*:

shows $\text{set-l2 } (\lambda i. f i + g i) A \leq \text{set-l2 } f A + \text{set-l2 } g A$

<proof>

lemma *sqrt-sum-squares-le-sum*:

$\llbracket 0 \leq x; 0 \leq y \rrbracket \implies \text{sqrt } (x^2 + y^2) \leq x + y$

<proof>

lemma *set-l2-le-setsum* [rule-format]:

$(\forall i \in A. 0 \leq f i) \longrightarrow \text{set-l2 } f A \leq \text{setsum } f A$

<proof>

lemma *sqrt-sum-squares-le-sum-abs*: $\text{sqrt } (x^2 + y^2) \leq |x| + |y|$

<proof>

lemma *set-l2-le-setsum-abs*: $\text{set-l2 } f A \leq (\sum i \in A. |f i|)$

<proof>

lemma *set-l2-mult-ineq-lemma*:

fixes $a b c d :: \text{real}$

shows $2 * (a * c) * (b * d) \leq a^2 * d^2 + b^2 * c^2$

<proof>

lemma *set-l2-mult-ineq*: $(\sum i \in A. |f i| * |g i|) \leq \text{set-l2 } f A * \text{set-l2 } g A$

<proof>

lemma *member-le-set-l2*: $\llbracket \text{finite } A; i \in A \rrbracket \implies f i \leq \text{set-l2 } f A$

<proof>

4.5 Vectors are a real normed vector space

instantiation $vec :: (\text{norm}, \text{finite}) \text{ norm}$

begin

definition

norm-vec-def: $\text{norm } x = \text{set-l2 } (\lambda i. \text{norm } (\text{Rep-vec } x i)) \text{ UNIV}$

instance *<proof>*

end

lemma *vec-norm-ge-zero*:

fixes $x :: 'a::\text{real-normed-vector} \wedge 'n::\text{finite}$

shows $0 \leq \text{norm } x$

$\langle \text{proof} \rangle$

lemma *vec-norm-eq-zero*:

fixes $x :: 'a::\text{real-normed-vector} \wedge 'n::\text{finite}$

shows $(\text{norm } x = 0) = (x = 0)$

$\langle \text{proof} \rangle$

lemma *vec-norm-triangle-ineq*:

fixes $x \ y :: 'a::\text{real-normed-vector} \wedge 'n::\text{finite}$

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

$\langle \text{proof} \rangle$

lemma *vec-norm-scaleR*:

fixes $x :: 'a::\text{real-normed-vector} \wedge 'n::\text{finite}$

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

$\langle \text{proof} \rangle$

instantiation *vec* :: $(\text{real-normed-vector}, \text{finite}) \text{ real-normed-vector}$

begin

definition

sgn-vec-def: $\text{sgn } (x::'a \wedge 'b) = \text{scaleR } (\text{inverse } (\text{norm } x)) \ x$

instance

$\langle \text{proof} \rangle$

end

lemma *norm-Rep-vec-le*: $\text{norm } (\text{Rep-vec } x \ i) \leq \text{norm } x$

$\langle \text{proof} \rangle$

interpretation *Rep-vec*: *bounded-linear* $[\lambda x. \text{Rep-vec } x \ i]$

$\langle \text{proof} \rangle$

4.6 Vectors are an inner product space

instantiation *vec* :: $(\text{inner}, \text{finite}) \text{ inner}$

begin

definition

inner-vec-def: $\text{inner } x \ y = (\sum i \in \text{UNIV}. \text{inner } (\text{Rep-vec } x \ i) (\text{Rep-vec } y \ i))$

instance $\langle \text{proof} \rangle$

end

lemma *vec-inner-commute*:
fixes $x\ y :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $\text{inner } x\ y = \text{inner } y\ x$
 $\langle \text{proof} \rangle$

lemma *vec-inner-left-distrib*:
fixes $x\ y\ z :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $\text{inner } (x + y)\ z = \text{inner } x\ z + \text{inner } y\ z$
 $\langle \text{proof} \rangle$

lemma *vec-inner-scaleR-left*:
fixes $x\ y :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $\text{inner } (\text{scaleR } r\ x)\ y = r * \text{inner } x\ y$
 $\langle \text{proof} \rangle$

lemma *vec-inner-ge-zero*:
fixes $x :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $0 \leq \text{inner } x\ x$
 $\langle \text{proof} \rangle$

lemma *vec-inner-eq-zero*:
fixes $x :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $(\text{inner } x\ x = 0) = (x = 0)$
 $\langle \text{proof} \rangle$

lemma *vec-norm-eq-sqrt-inner*:
fixes $x :: 'a::\text{real-inner} \wedge 'n::\text{finite}$
shows $\text{norm } x = \text{sqrt } (\text{inner } x\ x)$
 $\langle \text{proof} \rangle$

instance $\text{vec} :: (\text{real-inner}, \text{finite}) \text{ real-inner}$
 $\langle \text{proof} \rangle$

4.7 Vector is a functor

definition
 $\text{vec-map} :: ('a \Rightarrow 'b) \Rightarrow 'a \wedge 'n::\text{finite} \Rightarrow 'b \wedge 'n$ **where**
 $\text{vec-map } f\ x = \text{Abs-vec } (\lambda i. f (\text{Rep-vec } x\ i))$

lemma *Rep-vec-vec-map [simp]*: $\text{Rep-vec } (\text{vec-map } f\ x)\ i = f (\text{Rep-vec } x\ i)$
 $\langle \text{proof} \rangle$

lemma *vec-map-id*: $\text{vec-map } \text{id} = \text{id}$
 $\langle \text{proof} \rangle$

lemma *vec-map-o*: $\text{vec-map } (f \circ g) = \text{vec-map } f \circ \text{vec-map } g$
 $\langle \text{proof} \rangle$

lemma *vec-map-ident [simp]*: $\text{vec-map } (\lambda a. a) = (\lambda x. x)$

<proof>

lemma *vec-map-map*: $vec\text{-map } f (vec\text{-map } g x) = vec\text{-map } (\lambda a. f (g a)) x$
<proof>

4.8 Vector dot product

definition

$dot :: 'a ^ 'n \Rightarrow 'a :: finite \Rightarrow 'a :: semiring-0$ **where**
 $dot x y = (\sum_{i \in UNIV. Rep\text{-vec } x i * Rep\text{-vec } y i})$

lemma *dot-right-distrib*: $dot x (y + z) = dot x y + dot x z$
<proof>

lemma *dot-left-distrib*: $dot (x + y) z = dot x z + dot y z$
<proof>

lemma *dot-right-scaleR*:

fixes $x y :: 'a :: real\text{-algebra} ^ 'n :: finite$
shows $dot x (scaleR r y) = scaleR r (dot x y)$
<proof>

lemma *dot-left-scaleR*:

fixes $x y :: 'a :: real\text{-algebra} ^ 'n :: finite$
shows $dot (scaleR r x) y = scaleR r (dot x y)$
<proof>

lemma *norm-dot-ineq*:

fixes $x y :: 'a :: real\text{-normed-algebra} ^ 'n :: finite$
shows $norm (dot x y) \leq norm x * norm y$
<proof>

interpretation *dot*:

bounded-bilinear [$\lambda x y. dot x y :: 'a :: real\text{-normed-algebra}$]
<proof>

lemma *dot-right-zero*: $dot x 0 = 0$
<proof>

lemma *dot-left-zero*: $dot 0 y = 0$
<proof>

4.9 Single-component vectors

definition

$vec1 :: 'n :: finite \Rightarrow 'a :: zero \Rightarrow 'a ^ 'n$ **where**
 $vec1 j x = Abs\text{-vec } (\lambda i. \text{if } i = j \text{ then } x \text{ else } 0)$

lemma *Rep-vec-vec1 [simp]*:

$Rep\text{-vec } (vec1 j x) i = (\text{if } i = j \text{ then } x \text{ else } 0)$

<proof>

lemma *vec1-zero* [*simp*]: $vec1\ j\ 0 = 0$

<proof>

lemma *vec1-add*:

fixes $x\ y :: 'a::comm-monoid-add$

shows $vec1\ j\ (x + y) = vec1\ j\ x + vec1\ j\ y$

<proof>

lemma *vec1-diff*:

fixes $x\ y :: 'a::ab-group-add$

shows $vec1\ j\ (x - y) = vec1\ j\ x - vec1\ j\ y$

<proof>

lemma *vec1-scaleR*:

fixes $x :: 'a::real-vector$

shows $vec1\ j\ (scaleR\ r\ x) = scaleR\ r\ (vec1\ j\ x)$

<proof>

interpretation *vec1*: *additive* [$\lambda x::'a::ab-group-add. vec1\ j\ x$]

<proof>

lemma *norm-vec1* [*simp*]:

fixes $a :: 'a::real-normed-vector$

shows $norm\ (vec1\ j\ a) = norm\ a$

<proof>

interpretation *vec1*:

bounded-linear [$\lambda x::'a::real-normed-vector. vec1\ j\ x$]

<proof>

lemma *Rep-vec-setsum*: $Rep-vec\ (setsum\ f\ A)\ i = (\sum\ x \in A. Rep-vec\ (f\ x)\ i)$

<proof>

lemma *setsum-vec1*: $(\sum\ i \in UNIV. vec1\ i\ (f\ i)) = Abs-vec\ f$

<proof>

end

5 Square Matrices

theory *SquareMatrix*

imports *VectorType*

begin

typedef (**open**) $('a, 'n)$ *smatrix* = *UNIV* :: $('n \Rightarrow 'n \Rightarrow 'a)$ *set* *<proof>*

declare *Abs-matrix-inverse* [*simplified, iff*]

lemma *expand-matrix-eq*:

$A = B \iff (\forall i j. \text{Rep-matrix } A \ i \ j = \text{Rep-matrix } B \ i \ j)$
<proof>

lemma *matrix-ext*:

$(\bigwedge i j. \text{Rep-matrix } A \ i \ j = \text{Rep-matrix } B \ i \ j) \implies A = B$
<proof>

5.1 Matrix arithmetic

instantiation *matrix* :: (*zero, type*) *zero*
begin

definition

zero-matrix-def: $0 = \text{Abs-matrix } (\lambda i j. 0)$

instance *<proof>*
end

instantiation *matrix* :: (*plus, type*) *plus*
begin

definition

plus-matrix-def:
 $A + B = \text{Abs-matrix } (\lambda i j. \text{Rep-matrix } A \ i \ j + \text{Rep-matrix } B \ i \ j)$

instance *<proof>*
end

instantiation *matrix* :: (*minus, type*) *minus*
begin

definition

minus-matrix-def:
 $A - B = \text{Abs-matrix } (\lambda i j. \text{Rep-matrix } A \ i \ j - \text{Rep-matrix } B \ i \ j)$

instance *<proof>*
end

instantiation *matrix* :: (*uminus, type*) *uminus*
begin

definition

uminus-matrix-def:
 $- A = \text{Abs-matrix } (\lambda i j. - \text{Rep-matrix } A \ i \ j)$

instance *<proof>*

end

lemma *Rep-matrix-zero* [*simp*]: $Rep\text{-matrix } 0 \ i \ j = 0$
<proof>

lemma *Rep-matrix-add* [*simp*]:
 $Rep\text{-matrix } (A + B) \ i \ j = Rep\text{-matrix } A \ i \ j + Rep\text{-matrix } B \ i \ j$
<proof>

lemma *Rep-matrix-diff* [*simp*]:
 $Rep\text{-matrix } (A - B) \ i \ j = Rep\text{-matrix } A \ i \ j - Rep\text{-matrix } B \ i \ j$
<proof>

lemma *Rep-matrix-uminus* [*simp*]:
 $Rep\text{-matrix } (- A) \ i \ j = - Rep\text{-matrix } A \ i \ j$
<proof>

instance *smatrix* :: (*semigroup-add, type*) *semigroup-add*
<proof>

instance *smatrix* :: (*ab-semigroup-add, type*) *ab-semigroup-add*
<proof>

instance *smatrix* :: (*comm-monoid-add, type*) *comm-monoid-add*
<proof>

instance *smatrix* :: (*cancel-semigroup-add, type*) *cancel-semigroup-add*
<proof>

instance *smatrix* :: (*cancel-ab-semigroup-add, type*) *cancel-ab-semigroup-add*
<proof>

instance *smatrix* :: (*ab-group-add, type*) *ab-group-add*
<proof>

5.2 Smatrix multiplication

instantiation *smatrix* :: (*semiring-0, finite*) *semiring-0*
begin

definition

times-smatrix-def:

$A * B = Abs\text{-smatrix}$

$(\lambda i \ j. \sum_{k \in UNIV} Rep\text{-matrix } A \ i \ k * Rep\text{-matrix } B \ k \ j)$

lemma *Rep-matrix-mult* [*simp*]:
 $Rep\text{-matrix } (A * B) \ i \ j =$
 $(\sum_{k \in UNIV} Rep\text{-matrix } A \ i \ k * Rep\text{-matrix } B \ k \ j)$
<proof>

```

instance ⟨proof⟩

end

instance smatrix :: (semiring-0-cancel, finite) semiring-0-cancel ⟨proof⟩

instance smatrix :: (ring, finite) ring ⟨proof⟩

instantiation smatrix :: (semiring-1, finite) semiring-1
begin

definition
  one-smatrix-def:
    1 = Abs-smatrix (λi j. if i = j then 1 else 0)

lemma Rep-smatrix-one [simp]:
  Rep-smatrix 1 i j = (if i = j then 1 else 0)
  ⟨proof⟩

instance ⟨proof⟩

end

instance smatrix :: (semiring-1-cancel, finite) semiring-1-cancel ⟨proof⟩

instance smatrix :: (ring-1, finite) ring-1 ⟨proof⟩

```

5.3 Numeral syntax for matrices

```

instantiation smatrix :: (ring-1, finite) number
begin

definition
  number-of-smatrix-def: (number-of w::(-,-)smatrix) = of-int w

instance ⟨proof⟩
end

```

Unfortunately class *number-ring* requires commutativity of multiplication.

5.4 Matrices are a real vector space

```

instantiation smatrix :: (scaleR, finite) scaleR
begin

definition
  scaleR-smatrix-def:
    scaleR r A = Abs-smatrix (λi j. scaleR r (Rep-smatrix A i j))

```

instance $\langle proof \rangle$
end

lemma *Rep-matrix-scaleR* [simp]:
 $Rep\text{-matrix} (scaleR\ r\ A)\ i\ j = scaleR\ r\ (Rep\text{-matrix}\ A\ i\ j)$
 $\langle proof \rangle$

instance *smatrix* :: (real-vector, finite) real-vector
 $\langle proof \rangle$

instance *smatrix* :: (real-algebra, finite) real-algebra
 $\langle proof \rangle$

instance *smatrix* :: (real-algebra-1, finite) real-algebra-1 $\langle proof \rangle$

5.5 Applying a matrix to a vector

definition

$mmult :: ('a::semiring-0, 'n::finite)\ smatrix \Rightarrow 'a ^ 'n \Rightarrow 'a ^ 'n$

where

$mmult\ A\ x = Abs\text{-vec} (\lambda i. \sum_{j \in UNIV}. Rep\text{-matrix}\ A\ i\ j * Rep\text{-vec}\ x\ j)$

lemma *Rep-vec-mmult* [simp]:

$Rep\text{-vec} (mmult\ A\ x)\ i = (\sum_{j \in UNIV}. Rep\text{-matrix}\ A\ i\ j * Rep\text{-vec}\ x\ j)$
 $\langle proof \rangle$

lemma *mmult-mmult*: $mmult\ A\ (mmult\ B\ x) = mmult\ (A * B)\ x$
 $\langle proof \rangle$

lemma *mmult-right-distrib*: $mmult\ A\ (x + y) = mmult\ A\ x + mmult\ A\ y$
 $\langle proof \rangle$

lemma *mmult-left-distrib*: $mmult\ (A + B)\ x = mmult\ A\ x + mmult\ B\ x$
 $\langle proof \rangle$

interpretation *mmult-right*:

$additive\ [\lambda x. mmult\ A\ x :: 'a::ring ^ 'n::finite]$
 $\langle proof \rangle$

interpretation *mmult-left*:

$additive\ [\lambda A. mmult\ A\ x :: 'a::ring ^ 'n::finite]$
 $\langle proof \rangle$

declare *mmult-right.zero* [simp]

declare *mmult-left.zero* [simp]

lemma *mmult-vec1*: $mmult\ A\ (vec1\ j\ a) = Abs\text{-vec} (\lambda i. Rep\text{-matrix}\ A\ i\ j * a)$
 $\langle proof \rangle$

Class *semiring-1* is required, since class *semiring-0* would allow multiplication in the ring to always return zero.

lemma *smatrix-mmult-ext*:
fixes $A B :: ('a::\text{semiring-1}, 'n::\text{finite}) \text{smatrix}$
assumes $\text{mmult-eq}: \bigwedge x. \text{mmult } A \ x = \text{mmult } B \ x$
shows $A = B$
 $\langle \text{proof} \rangle$

lemma *expand-smatrix-eq-mmult*:
fixes $A B :: ('a::\text{semiring-1}, 'n::\text{finite}) \text{smatrix}$
shows $A = B \longleftrightarrow (\forall x. \text{mmult } A \ x = \text{mmult } B \ x)$
 $\langle \text{proof} \rangle$

lemma *mmult-right-scaleR*:
fixes $A :: ('a::\text{real-algebra}, 'n::\text{finite}) \text{smatrix}$
shows $\text{mmult } A \ (\text{scaleR } r \ x) = \text{scaleR } r \ (\text{mmult } A \ x)$
 $\langle \text{proof} \rangle$

lemma *mmult-left-scaleR*:
fixes $A :: ('a::\text{real-algebra}, 'n::\text{finite}) \text{smatrix}$
shows $\text{mmult } (\text{scaleR } r \ A) \ x = \text{scaleR } r \ (\text{mmult } A \ x)$
 $\langle \text{proof} \rangle$

lemma *mmult-1-left [simp]*:
fixes $x :: 'a::\text{semiring-1} \wedge 'n::\text{finite}$
shows $\text{mmult } 1 \ x = x$
 $\langle \text{proof} \rangle$

lemma *bounded-mmult*:
fixes $A :: ('a::\text{real-normed-algebra}, 'n::\text{finite}) \text{smatrix}$
shows $\exists K. \forall x. \text{norm } (\text{mmult } A \ x) \leq \text{norm } x * K$
 $\langle \text{proof} \rangle$

lemma *bounded-linear-mmult-right*:
fixes $A :: ('a::\text{real-normed-algebra}, 'n::\text{finite}) \text{smatrix}$
shows *bounded-linear* $(\text{mmult } A)$
 $\langle \text{proof} \rangle$

interpretation *mmult-right*:
bounded-linear $[\text{mmult } (A::('a::\text{real-normed-algebra}, 'n::\text{finite}) \text{smatrix})]$
 $\langle \text{proof} \rangle$

5.6 Matrix norm

instantiation *smatrix* :: $(\text{real-normed-algebra}, \text{finite}) \text{norm}$
begin

definition
norm-smatrix-def:

$norm\ A = (THE\ r.\ isLub\ UNIV\ \{norm\ (mmult\ A\ x)\ |\ x.\ norm\ x\ \leq\ 1\}\ r)$

instance $\langle proof \rangle$
end

lemma *reals-complete1*:
 assumes *notempty-S*: $\exists x.\ x \in S$
 and *exists-Ub*: $\exists y.\ isUb\ (UNIV::real\ set)\ S\ y$
 shows $\exists! t.\ isLub\ (UNIV::real\ set)\ S\ t$
 $\langle proof \rangle$

lemma *isLub-smatrix-norm*:
 $isLub\ UNIV\ \{norm\ (mmult\ A\ x)\ |\ x.\ norm\ x\ \leq\ 1\}\ (norm\ A)$
 $\langle proof \rangle$

lemma *smatrix-norm-geI*:
 fixes $A :: ('a::real-normed-algebra,\ 'n::finite)\ smatrix$
 shows $\exists x.\ y = norm\ (mmult\ A\ x) \wedge norm\ x \leq 1 \implies y \leq norm\ A$
 $\langle proof \rangle$

lemma *smatrix-norm-leI*:
 fixes $A :: ('a::real-normed-algebra,\ 'n::finite)\ smatrix$
 assumes $\bigwedge x.\ norm\ x \leq 1 \implies norm\ (mmult\ A\ x) \leq y$
 shows $norm\ A \leq y$
 $\langle proof \rangle$

lemma *norm-mmult-ineq*: $norm\ (mmult\ A\ x) \leq norm\ A * norm\ x$
 $\langle proof \rangle$

lemma *smatrix-norm-ge-zero*:
 fixes $A :: ('a::real-normed-algebra,\ 'n::finite)\ smatrix$
 shows $0 \leq norm\ A$
 $\langle proof \rangle$

lemma *smatrix-norm-eq-zero*:
 fixes $A :: ('a::real-normed-algebra-1,\ 'n::finite)\ smatrix$
 shows $norm\ A = 0 \iff A = 0$
 $\langle proof \rangle$

lemma *smatrix-norm-triangle-ineq*:
 fixes $A\ B :: ('a::real-normed-algebra,\ 'n::finite)\ smatrix$
 shows $norm\ (A + B) \leq norm\ A + norm\ B$
 $\langle proof \rangle$

lemma *smatrix-norm-scaleR*:
 fixes $A :: ('a::real-normed-algebra,\ 'n::finite)\ smatrix$
 shows $norm\ (scaleR\ r\ A) = |r| * norm\ A$
 $\langle proof \rangle$

lemma *smatrix-norm-mult-ineq*:

fixes $A B :: ('a::\text{real-normed-algebra}, 'n::\text{finite}) \text{ smatrix}$

shows $\text{norm } (A * B) \leq \text{norm } A * \text{norm } B$

<proof>

lemma *smatrix-norm-1*:

$\text{norm } (1::('a::\text{real-normed-algebra-1}, 'n::\text{finite}) \text{ smatrix}) = 1$

<proof>

instantiation *smatrix* :: $(\text{real-normed-algebra-1}, \text{finite}) \text{ real-normed-algebra-1}$

begin

definition

sgn-smatrix-def: $\text{sgn } (x::(-, -) \text{ smatrix}) = \text{scaleR } (\text{inverse } (\text{norm } x)) x$

instance

<proof>

end

5.7 Vector outer product

definition

outer :: $'a \wedge 'n \Rightarrow 'a \wedge 'n \Rightarrow ('a::\text{semiring}, 'n::\text{finite}) \text{ smatrix}$ **where**

$\text{outer } x y = \text{Abs-smatrix } (\lambda i j. \text{Rep-vec } x i * \text{Rep-vec } y j)$

lemma *Rep-smatrix-outer* [*simp*]:

$\text{Rep-smatrix } (\text{outer } x y) i j = \text{Rep-vec } x i * \text{Rep-vec } y j$

<proof>

lemma *outer-right-distrib*: $\text{outer } x (y + z) = \text{outer } x y + \text{outer } x z$

<proof>

lemma *outer-left-distrib*: $\text{outer } (x + y) z = \text{outer } x z + \text{outer } y z$

<proof>

interpretation *outer-right*:

additive [$\lambda y. \text{outer } x y :: ('a::\text{ring}, 'n::\text{finite}) \text{ smatrix}$]

<proof>

interpretation *outer-left*:

additive [$\lambda x. \text{outer } x y :: ('a::\text{ring}, 'n::\text{finite}) \text{ smatrix}$]

<proof>

lemma *outer-right-scaleR*:

fixes $x y :: 'a::\text{real-algebra} \wedge 'n::\text{finite}$

shows $\text{outer } x (\text{scaleR } r y) = \text{scaleR } r (\text{outer } x y)$

<proof>

```

lemma outer-left-scaleR:
  fixes  $x\ y :: 'a::\text{real-algebra} \wedge 'n::\text{finite}$ 
  shows  $\text{outer} (\text{scaleR } r\ x)\ y = \text{scaleR } r (\text{outer } x\ y)$ 
  <proof>

lemma mmult-outer:  $\text{mmult} (\text{outer } x\ y)\ z = \text{vec-map} (\lambda a. a * \text{dot } y\ z)\ x$ 
  <proof>

lemma norm-mmult-outer-ineq:
  fixes  $x\ y\ z :: 'a::\text{real-normed-algebra} \wedge 'n::\text{finite}$ 
  shows  $\text{norm} (\text{mmult} (\text{outer } x\ y)\ z) \leq \text{norm } x * \text{norm } y * \text{norm } z$ 
  <proof>

lemma norm-outer-ineq:
  fixes  $x\ y\ z :: 'a::\text{real-normed-algebra} \wedge 'n::\text{finite}$ 
  shows  $\text{norm} (\text{outer } x\ y) \leq \text{norm } x * \text{norm } y$ 
  <proof>

interpretation outer: bounded-bilinear [outer]
  <proof>

end

```

6 Gradient Derivatives

```

theory GradientDeriv
imports InnerProduct FrechetDeriv
begin

```

6.1 Gradient Derivative

definition

```

gderiv ::
  [ $'a::\text{real-inner} \Rightarrow \text{real}, 'a, 'a] \Rightarrow \text{bool}$ 
  ((GDERIV (-)/ (-)/ :> (-)) [1000, 1000, 60] 60) where
  GDERIV  $f\ x :> D = \text{FDERIV } f\ x :> (\lambda h. \text{inner } h\ D)$ 

```

```

lemma deriv-fderiv:  $\text{DERIV } f\ x :> D = \text{FDERIV } f\ x :> (\lambda h. h * D)$ 
  <proof>

```

```

lemma gderiv-deriv [simp]:  $\text{GDERIV } f\ x :> D = \text{DERIV } f\ x :> D$ 
  <proof>

```

lemma *GDERIV-DERIV-compose*:

```

  [[ $\text{GDERIV } f\ x :> D; \text{DERIV } g\ (f\ x) :> E$ ]
   $\implies \text{GDERIV } (\lambda x. g\ (f\ x))\ x :> \text{scaleR } E\ D$ 
  <proof>

```

lemma *FDERIV-subst*: $\llbracket FDERIV\ f\ x\ :>\ D;\ D = E \rrbracket \implies FDERIV\ f\ x\ :>\ E$
 ⟨proof⟩

lemma *GDERIV-subst*: $\llbracket GDERIV\ f\ x\ :>\ D;\ D = E \rrbracket \implies GDERIV\ f\ x\ :>\ E$
 ⟨proof⟩

lemma *GDERIV-const*: $GDERIV\ (\lambda x. k)\ x\ :>\ 0$
 ⟨proof⟩

lemma *GDERIV-add*:
 $\llbracket GDERIV\ f\ x\ :>\ D;\ GDERIV\ g\ x\ :>\ E \rrbracket$
 $\implies GDERIV\ (\lambda x. f\ x + g\ x)\ x\ :>\ D + E$
 ⟨proof⟩

lemma *GDERIV-minus*:
 $GDERIV\ f\ x\ :>\ D \implies GDERIV\ (\lambda x. -\ f\ x)\ x\ :>\ -\ D$
 ⟨proof⟩

lemma *GDERIV-diff*:
 $\llbracket GDERIV\ f\ x\ :>\ D;\ GDERIV\ g\ x\ :>\ E \rrbracket$
 $\implies GDERIV\ (\lambda x. f\ x - g\ x)\ x\ :>\ D - E$
 ⟨proof⟩

lemma *GDERIV-scaleR*:
 $\llbracket DERIV\ f\ x\ :>\ D;\ GDERIV\ g\ x\ :>\ E \rrbracket$
 $\implies GDERIV\ (\lambda x. scaleR\ (f\ x)\ (g\ x))\ x$
 $\quad :>\ (scaleR\ (f\ x)\ E + scaleR\ D\ (g\ x))$
 ⟨proof⟩

lemma *GDERIV-mult*:
 $\llbracket GDERIV\ f\ x\ :>\ D;\ GDERIV\ g\ x\ :>\ E \rrbracket$
 $\implies GDERIV\ (\lambda x. f\ x * g\ x)\ x\ :>\ scaleR\ (f\ x)\ E + scaleR\ (g\ x)\ D$
 ⟨proof⟩

lemma *GDERIV-inverse*:
 $\llbracket GDERIV\ f\ x\ :>\ D;\ f\ x \neq 0 \rrbracket$
 $\implies GDERIV\ (\lambda x. inverse\ (f\ x))\ x\ :>\ -\ (inverse\ (f\ x))^2 *_{\mathbb{R}}\ D$
 ⟨proof⟩

lemma *GDERIV-norm*: $x \neq 0 \implies GDERIV\ (\lambda x. norm\ x)\ x\ :>\ sgn\ x$
 ⟨proof⟩

lemmas *FDERIV-norm = GDERIV-norm* [unfolded gderiv-def]

lemma *isCont-sgn*: $x \neq 0 \implies isCont\ (\lambda x. sgn\ x)\ x$
 ⟨proof⟩

end

7 Pairs as Vector Spaces

```
theory PairVector
imports GradientDeriv
begin
```

7.1 Vector arithmetic

```
instantiation * :: (zero, zero) zero
begin
```

definition

zero-prod-def: $0 = (0, 0)$

```
instance <proof>
end
```

```
instantiation * :: (plus, plus) plus
begin
```

definition

plus-prod-def: $A + B = (fst A + fst B, snd A + snd B)$

```
instance <proof>
end
```

```
instantiation * :: (minus, minus) minus
begin
```

definition

minus-prod-def: $A - B = (fst A - fst B, snd A - snd B)$

```
instance <proof>
end
```

```
instantiation * :: (uminus, uminus) uminus
begin
```

definition

uminus-prod-def: $- A = (- fst A, - snd A)$

```
instance <proof>
end
```

```
instantiation * :: (scaleR, scaleR) scaleR
begin
```

definition

scaleR-prod-def: $scaleR r A = (scaleR r (fst A), scaleR r (snd A))$

instance $\langle proof \rangle$
end

instantiation * :: (norm, norm) norm
begin

definition
norm-prod-def: $norm\ A = \text{sqrt}\ ((norm\ (fst\ A))^2 + (norm\ (snd\ A))^2)$

instance $\langle proof \rangle$
end

instantiation * :: (inner, inner) inner
begin

definition
inner-prod-def: $inner\ A\ B = inner\ (fst\ A)\ (fst\ B) + inner\ (snd\ A)\ (snd\ B)$

instance $\langle proof \rangle$
end

instantiation * :: (sgn-div-norm, sgn-div-norm) sgn-div-norm
begin

definition
sgn-prod-def: $sgn\ (x::'a \times 'b) = \text{scaleR}\ (\text{inverse}\ (norm\ x))\ x$

instance
 $\langle proof \rangle$

end

lemma *fst-zero* [*simp*]: $fst\ 0 = 0$
 $\langle proof \rangle$

lemma *snd-zero* [*simp*]: $snd\ 0 = 0$
 $\langle proof \rangle$

lemma *fst-add* [*simp*]: $fst\ (A + B) = fst\ A + fst\ B$
 $\langle proof \rangle$

lemma *snd-add* [*simp*]: $snd\ (A + B) = snd\ A + snd\ B$
 $\langle proof \rangle$

lemma *add-Pair* [*simp*]: $(a, b) + (c, d) = (a + c, b + d)$
 $\langle proof \rangle$

lemma *fst-diff* [*simp*]: $fst\ (A - B) = fst\ A - fst\ B$
 $\langle proof \rangle$

lemma *snd-diff* [*simp*]: $snd (A - B) = snd A - snd B$
<proof>

lemma *diff-Pair* [*simp*]: $(a, b) - (c, d) = (a - c, b - d)$
<proof>

lemma *fst-uminus* [*simp*]: $fst (- A) = - fst A$
<proof>

lemma *snd-uminus* [*simp*]: $snd (- A) = - snd A$
<proof>

lemma *uminus-Pair* [*simp*]: $-(a, b) = (- a, - b)$
<proof>

lemma *fst-scaleR* [*simp*]: $fst (scaleR r A) = scaleR r (fst A)$
<proof>

lemma *snd-scaleR* [*simp*]: $snd (scaleR r A) = scaleR r (snd A)$
<proof>

lemma *scaleR-Pair* [*simp*]: $scaleR r (a, b) = (scaleR r a, scaleR r b)$
<proof>

lemma *norm-Pair* [*simp*]: $norm (a, b) = sqrt ((norm a)^2 + (norm b)^2)$
<proof>

lemma *inner-Pair* [*simp*]: $inner (a, b) (c, d) = inner a c + inner b d$
<proof>

lemmas *expand-prod-eq = Pair-fst-snd-eq*

instance * :: (*semigroup-add*, *semigroup-add*) *semigroup-add*
<proof>

instance * :: (*ab-semigroup-add*, *ab-semigroup-add*) *ab-semigroup-add*
<proof>

instance * :: (*comm-monoid-add*, *comm-monoid-add*) *comm-monoid-add*
<proof>

instance * :: (*cancel-semigroup-add*, *cancel-semigroup-add*) *cancel-semigroup-add*
<proof>

instance * :: (*cancel-ab-semigroup-add*, *cancel-ab-semigroup-add*) *cancel-ab-semigroup-add*
<proof>

instance * :: (*ab-group-add*, *ab-group-add*) *ab-group-add*

<proof>

instance * :: (*real-vector*, *real-vector*) *real-vector*
<proof>

7.2 Product is a normed vector space

instance * :: (*real-normed-vector*, *real-normed-vector*) *real-normed-vector*
<proof>

interpretation *fst*: *bounded-linear* [*fst*]
<proof>

interpretation *snd*: *bounded-linear* [*snd*]
<proof>

lemma *LIMSEQ-Pair*:

$\llbracket X \text{ ----> } a; Y \text{ ----> } b \rrbracket \implies (\lambda n. (X\ n, Y\ n)) \text{ ----> } (a, b)$
<proof>

lemma *Cauchy-Pair*:

$\llbracket \text{Cauchy } X; \text{Cauchy } Y \rrbracket \implies \text{Cauchy } (\lambda n. (X\ n, Y\ n))$
<proof>

instance * :: (*banach*, *banach*) *banach*
<proof>

7.3 Product is an inner product space

instance * :: (*real-inner*, *real-inner*) *real-inner*
<proof>

end

8 Cross Products

theory *CrossProduct*

imports *PairVector*

begin

types *'a triple* = *'a* * *'a* * *'a*

types *R3* = *real* * *real* * *real*

lemma *norm* ((*2,3,6*)::*R3*) = 7
<proof>

lemma *inner* ((*1,2,3*)::*R3*) (*4,5,6*) = ?*x*

<proof>

definition

cross :: *real triple* \Rightarrow *real triple* \Rightarrow *real triple* **where**

cross = $(\lambda(a, b, c). \lambda(x, y, z).$

$(b*z - c*y, c*x - a*z, a*y - b*x))$

lemma *cross-Pair*:

cross (a, b, c) (x, y, z) = $(b*z - c*y, c*x - a*z, a*y - b*x)$

<proof>

lemma *left-cross-distrib*: *cross* (A + B) C = *cross* A C + *cross* B C

<proof>

lemma *right-cross-distrib*: *cross* A (B + C) = *cross* A B + *cross* A C

<proof>

lemma *cross-scaleR-left*:

cross (scaleR r A) B = scaleR r (i*cross* A B)

<proof>

lemma *cross-scaleR-right*:

cross A (scaleR r B) = scaleR r (i*cross* A B)

<proof>

lemma *Lagrange-identity*:

$(\text{norm } A)^2 * (\text{norm } B)^2 = (\text{norm } (\text{inner } A B))^2 + (\text{norm } (\text{cross } A B))^2$

<proof>

lemma *norm-cross-ineq*: $\text{norm } (\text{cross } A B) \leq \text{norm } A * \text{norm } B$

<proof>

interpretation *bounded-bilinear-cross*: *bounded-bilinear* [*cross*]

<proof>

lemma *cross-anticommutate*: *cross* A B = - *cross* B A

<proof>

end

9 Quaternions

theory *Quaternion*

imports *InnerProduct*

begin

9.1 Definition

datatype *quaternion* = *Quaternion complex complex*

9.2 Addition and Subtraction

instantiation *quaternion* :: *ab-group-add*
begin

definition

zero-quaternion-def: $0 = \text{Quaternion } 0 \ 0$

fun *plus-quaternion* **where**

quaternion-plus:

$\text{Quaternion } a \ b + \text{Quaternion } c \ d = \text{Quaternion } (a + c) \ (b + d)$

fun *minus-quaternion* **where**

quaternion-minus:

$\text{Quaternion } a \ b - \text{Quaternion } c \ d = \text{Quaternion } (a - c) \ (b - d)$

fun *uminus-quaternion* **where**

quaternion-uminus:

$- \text{Quaternion } a \ b = \text{Quaternion } (- a) \ (- b)$

instance *<proof>*

end

lemma *quaternion-eq-zero* [*simp*]:

$\text{Quaternion } a \ b = 0 \iff a = 0 \wedge b = 0$

<proof>

9.3 Multiplication

lemma *complex-cnj-scaleR*: $\text{cnj } (\text{scaleR } r \ x) = \text{scaleR } r \ (\text{cnj } x)$

<proof>

lemmas *complex-cnj-simps* =

complex-cnj-add

complex-cnj-diff

complex-cnj-minus

complex-cnj-mult

complex-cnj-divide

complex-cnj-inverse

complex-cnj-scaleR

lemmas *ring-distrib* =

left-distrib

right-distrib

left-diff-distrib

right-diff-distrib

instantiation *quaternion* :: *ring-1*
begin

fun *times-quaternion* **where**

quaternion-times:

$Quaternion\ a\ b * Quaternion\ c\ d =$

$Quaternion\ (a * c - d * cnj\ b)\ (cnj\ a * d + c * b)$

definition

one-quaternion-def: $1 = Quaternion\ 1\ 0$

instance $\langle proof \rangle$

end

9.4 Vector space

lemmas *scaleR-distrib* =

scaleR-left-distrib

scaleR-right-distrib

scaleR-left-diff-distrib

scaleR-right-diff-distrib

instantiation *quaternion* :: *real-algebra-1*
begin

fun *scaleR-quaternion* **where**

quaternion-scaleR:

$scaleR\ r\ (Quaternion\ a\ b) = Quaternion\ (scaleR\ r\ a)\ (scaleR\ r\ b)$

instance $\langle proof \rangle$

end

lemma *quaternion-of-real-eq*:

$of-real\ r = Quaternion\ (of-real\ r)\ 0$

$\langle proof \rangle$

9.5 Norm and inner product

instantiation *quaternion* :: *real-inner*
begin

fun *inner-quaternion* **where**

quaternion-inner:

$inner\ (Quaternion\ a\ b)\ (Quaternion\ c\ d) = inner\ a\ c + inner\ b\ d$

fun *norm-quaternion* **where**

quaternion-norm:
 $norm (Quaternion\ a\ b) = sqrt\ ((norm\ a)^2 + (norm\ b)^2)$

definition

sgn-quaternion-def:
 $sgn\ (x::quaternion) = scaleR\ (inverse\ (norm\ x))\ x$

instance $\langle proof \rangle$

end

9.6 Conjugate

fun

$qcnj :: quaternion \Rightarrow quaternion$

where

quaternion-cnj:
 $qcnj\ (Quaternion\ a\ b) = Quaternion\ (cnj\ a)\ (-\ b)$

lemma *complex-cnj-mult:* $cnj\ x * x = of-real\ ((norm\ x)^2)$
 $\langle proof \rangle$

lemma *complex-mult-cnj:* $x * cnj\ x = of-real\ ((norm\ x)^2)$
 $\langle proof \rangle$

lemma *quaternion-cnj-mult:* $qcnj\ x * x = of-real\ ((norm\ x)^2)$
 $\langle proof \rangle$

lemma *quaternion-mult-cnj:* $x * qcnj\ x = of-real\ ((norm\ x)^2)$
 $\langle proof \rangle$

9.7 Inverse

instantiation $quaternion :: division-ring$
begin

definition

inverse-quaternion-def:
 $inverse\ x = scaleR\ (inverse\ ((norm\ x)^2))\ (qcnj\ x)$

instance $\langle proof \rangle$

end

instance $quaternion :: real-normed-div-algebra$
 $\langle proof \rangle$

end