

Lie Groups and Algebras

Richard Schmoetten and Jacques D. Fleuriot

February 6, 2026

Abstract

Lie Groups are formalised as locales, building on the theory of Smooth Manifolds [1]. We formalise the diffeomorphism group of a manifold, and the action of a Lie group on a manifold. The general linear group is shown to be a Lie group by proving properties of the determinant, and matrix inverses. We also develop a theory of smooth vector fields on a C^∞ manifold M , defined as smooth maps from the manifold to its tangent bundle TM . We employ a shortcut that avoids difficulties in defining the tangent bundle as a manifold, but which still leads to vector fields with the properties one would expect. Notably, they are derivations $C^\infty(M) \rightarrow C^\infty(M)$. We construct *the* Lie algebra of a Lie group as an algebra of left-invariant smooth vector fields. Our main reference for the mathematics of smooth manifolds is Lee's textbook [2], which also contains material on Lie groups and algebras.

Contents

1	Abstract algebra locales over a field	3
1.1	Bilinearity, Jacobi identity	3
1.2	Unital and associative algebras	5
1.3	Lie algebra (locale)	7
1.4	Division algebras	8
2	Continuity of the determinant (and other maps)	11
3	Component expressions for inverse matrices over fields	12
4	Smoothness of real matrix operations and <i>det</i>	13
4.1	Smoothness of matrix multiplication	13
4.2	Smoothness of \prod and <i>det</i>	14
4.3	Smoothness of matrix inversion	15
5	Smooth vector fields	17
5.1	(Smooth) vector fields on an (entire) manifold.	17
5.1.1	Charts for the tangent bundle	17

5.1.2	Proofs about <i>apply-chart-TM</i> that mimic the properties of <i>('a, 'b) chart</i>	18
5.1.3	Differentiability of vector fields	21
5.2	Smoothness criterion for a vector field in a single chart.	24
5.2.1	Connecting the types <i>'a ⇒ ('a ⇒ real) ⇒ real</i> (used for <i>smooth-vector-field-local</i>) and <i>'a ⇒ 'a × (('a ⇒ real) ⇒ real)</i> (used for <i>λcharts k. c-manifold.section-of-TM-on-charts k (manifold.carrier charts)</i>).	25
5.2.2	Some theorems about smooth vector fields, locally and globally.	26
5.3	Smooth vector fields as maps $C^\infty(M) \rightarrow C^\infty(M)$	32
5.4	Smooth vector fields are derivations	32
5.5	Derivations are smooth vector fields	33
6	The Lie bracket of smooth vector fields	34
6.1	General lemmas	34
6.2	Properties of the Lie bracket on \mathfrak{X}	35
7	Definition of Lie Groups (as Locales)	37
7.1	Topological groups	37
7.2	Lie groups	37
7.3	Some lemmas about Lie groups (and other needed results).	39
8	Morphisms of Lie groups, actions and representations	40
8.1	Morphism of Lie groups.	40
8.2	Action of a Lie group on a manifold.	41
8.3	Action of a Lie Group on itself.	42
8.3.1	The left action.	42
8.3.2	The right action.	44
9	Models/Instances	46
9.1	Euclidean Space	46
9.1.1	Euclidean Spaces are Lie groups under (+).	46
9.2	The real numbers as a Lie group	47
10	The Lie algebra of a Lie Group	47
10.1	(Left-)invariant vector fields	48
11	Matrix Groups	50
11.1	Entry Type	50
11.2	$\text{Mat}(n, F)$	50
11.3	$\text{GL}(n, F)$	51

theory *Algebra-On*

```

imports
  HOL-Types-To-Sets.Linear-Algebra-On
  Jacobson-Basic-Algebra.Ring-Theory
begin

```

1 Abstract algebra locales over a *field*

... with carrier set and some implicit operations (only algebraic multiplication, scaling, and derived constants are not implicit).

For full generality, one could define an algebra as a ring that is also a module (rather than a vector space, i.e. have a (non/commutative) base ring instead of a base field).

1.1 Bilinearity, Jacobi identity

lemma (in *module-hom-on*) *mem-hom*:

```

assumes  $x \in S1$ 
shows  $f x \in S2$ 
<proof>

```

locale *bilinear-on* =

```

  vector-space-pair-on  $V W$  scale $V$  scale $W$  +
  vector-space-on  $X$  scale $X$ 

```

for $V::'b::ab-group-add$ set **and** $W::'c::ab-group-add$ set **and** $X::'d::ab-group-add$ set

```

  and scale $V::'a::field \Rightarrow 'b \Rightarrow 'b$  (infixr  $\langle \bullet_V \rangle$  75)

```

```

  and scale $W::'a \Rightarrow 'c \Rightarrow 'c$  (infixr  $\langle \bullet_W \rangle$  75)

```

```

  and scale $X::'a \Rightarrow 'd \Rightarrow 'd$  (infixr  $\langle \bullet_X \rangle$  75) +

```

```

fixes  $f::'b \Rightarrow 'c \Rightarrow 'd$ 

```

```

assumes linear $L: w \in W \Rightarrow$  linear-on  $V X$  scale $V$  scale $X$  ( $\lambda v. f v w$ )

```

```

  and linear $R: v \in V \Rightarrow$  linear-on  $W X$  scale $W$  scale $X$  ( $\lambda w. f v w$ )

```

begin

lemma *linearL'*: $\llbracket v \in V; w \in W \rrbracket \Rightarrow f (a \bullet_V v) w = a \bullet_X (f v w)$

```

 $\llbracket v \in V; v' \in V; w \in W \rrbracket \Rightarrow f (v + v') w = (f v w) + (f v' w)$ 

```

<proof>

lemma *linearR'*: $\llbracket v \in V; w \in W \rrbracket \Rightarrow f v (a \bullet_W w) = a \bullet_X (f v w)$

```

 $\llbracket v \in V; w \in W; w' \in W \rrbracket \Rightarrow f v (w + w') = (f v w) + (f v w')$ 

```

<proof>

lemma *bilinear-zero* [*simp*]:

```

shows  $w \in W \Rightarrow f 0 w = 0$   $v \in V \Rightarrow f v 0 = 0$ 

```

<proof>

lemma *bilinear-uminus* [*simp*]:

```

assumes  $v: v \in V$  and  $w: w \in W$ 

```

shows $f (-v) w = - (f v w) f v (-w) = - (f v w)$
 ⟨*proof*⟩

end

For bilinear maps, "alternating" means the same as "skew-symmetric", which is the same as "anti-symmetric".

locale *alternating-bilinear-on* = *bilinear-on* S S S *scale* *scale* *scale* **for** S *scale* f
 +
assumes *alternating*: $x \in S \implies f x x = 0$
begin

lemma *antisym*:
assumes $x \in S$ $y \in S$
shows $(f x y) + (f y x) = 0$
 ⟨*proof*⟩

lemma *antisym'*:
assumes $x \in S$ $y \in S$
shows $(f x y) = - (f y x)$
 ⟨*proof*⟩

lemma *antisym-uminus*:
assumes $x \in S$ $y \in S$
shows $f (-x) y = f y x$ $f x (-y) = f y x$
 ⟨*proof*⟩

end

abbreviation (*input*) *jacobi-identity-with*:: $'a \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
where *jacobi-identity-with* *zero-add* *f-add* *f-mult* $x y z \equiv$
 $zero-add = f-add (f-add (f-mult x (f-mult y z)) (f-mult y (f-mult z x))) (f-mult z (f-mult x y))$

abbreviation (*input*) *jacobi-identity*:: $('a::\{monoid-add\}) \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow bool$
where *jacobi-identity* $f-mult x y z \equiv jacobi-identity-with 0 (+) f-mult x y z$

lemma (**in** *module-hom-on*) *mapsto-zero*: $f 0 = 0$
 ⟨*proof*⟩

lemma (**in** *module-hom-on*) *mapsto-uminus*: $a \in S1 \implies f (-a) = - f a$
 ⟨*proof*⟩

lemma (**in** *module-hom-on*) *mapsto-closed*: $a \in S1 \implies f a \in S2$

<proof>

1.2 Unital and associative algebras

```
locale algebra-on = bilinear-on S S S scale scale scale amult
  for S
  and scale :: 'a::field  $\Rightarrow$  'b::ab-group-add  $\Rightarrow$  'b (infixr <*_S> 75)
  and amult (infixr <•> 74) +
  assumes amult-closed [simp]:  $a \in S \Rightarrow b \in S \Rightarrow amult\ a\ b \in S$ 
begin
```

lemma

```
shows distR:  $\llbracket x \in S; y \in S; z \in S \rrbracket \Rightarrow (x+y) \bullet z = x \bullet z + y \bullet z$ 
  and distL:  $\llbracket x \in S; y \in S; z \in S \rrbracket \Rightarrow z \bullet (x+y) = z \bullet x + z \bullet y$ 
  and scalar-compat :  $\llbracket x \in S; y \in S \rrbracket \Rightarrow (a *_S x) \bullet (b *_S y) = (a*b) *_S (x \bullet y)$ 
<proof>
```

lemma scalar-compat' [simp]:

```
shows  $\llbracket x \in S; y \in S \rrbracket \Rightarrow (a *_S x) \bullet y = a *_S (x \bullet y)$ 
  and  $\llbracket x \in S; y \in S \rrbracket \Rightarrow x \bullet (a *_S y) = a *_S (x \bullet y)$ 
<proof>
```

end

Sometimes an associative algebra is defined as a ring that is also a module (over a comm. ring), with the module and scalar multiplication being compatible, and the ring and module addition being the same. That definition implies an associative algebra is also unital, i.e. there is a multiplicative identity; in contrast, our definition doesn't. This is in agreement with how a 'a needs no identity, and an additional type class `typ>'a::ring-1` is provided (instead of the terminology of `rng` vs. `ring`).

```
locale assoc-algebra-on = algebra-on +
  assumes amult-assoc:  $\llbracket x \in S; y \in S; z \in S \rrbracket \Rightarrow (x \bullet y) \bullet z = x \bullet (y \bullet z)$ 
```

```
locale unital-algebra-on = algebra-on +
  fixes a-id
  assumes amult-id [simp]:  $a \text{-id} \in S\ a \in S \Rightarrow a \bullet a \text{-id} = a\ a \in S \Rightarrow a \text{-id} \bullet a = a$ 
begin
```

lemma id-neq-0-iff: $\exists a \in S. \exists b \in S. a \neq b \iff 0 \neq a \text{-id}$
<proof>

lemma id-neq-0-if:

```
shows  $a \in S \Rightarrow b \in S \Rightarrow a \neq b \Rightarrow 0 \neq a \text{-id}$ 
  and  $\text{card } S \geq 2 \Rightarrow 0 \neq a \text{-id}$ 
  and infinite S  $\Rightarrow 0 \neq a \text{-id}$ 
<proof>
```

lemma *id-neq-0-implies-elements* : $\exists a \in S. \exists b \in S. a \neq b$ **if** $0 \neq a\text{-id}$
 ⟨proof⟩

lemma *id-neq-0-implies-card*:
assumes $0 \neq a\text{-id}$
obtains $\text{card } S \geq 2 \mid \text{infinite } S$
 ⟨proof⟩

lemma *id-unique* [simp]:
fixes *other-id*
assumes $\text{other-id} \in S \wedge a. a \in S \implies a \bullet \text{other-id} = a \wedge \text{other-id} \bullet a = a$
shows $\text{other-id} = a\text{-id}$
 ⟨proof⟩

end

locale *assoc-algebra-1-on* = *assoc-algebra-on* + *unital-algebra-on* +
assumes *id-neq-0* [simp]: $a\text{-id} \neq 0$ — this is as in the class *ring-1*, and merely
 assures S has at least two elements
begin

lemma *is-ring-1-axioms*:
shows $\bigwedge a \ b \ c. a \in S \implies b \in S \implies c \in S \implies a \bullet b \bullet c = a \bullet (b \bullet c)$
and $\bigwedge a. a \in S \implies a\text{-id} \bullet a = a$
and $\bigwedge a. a \in S \implies a \bullet a\text{-id} = a$
and $\bigwedge a \ b \ c. a \in S \implies b \in S \implies c \in S \implies (a + b) \bullet c = a \bullet c + b \bullet c$
and $\bigwedge a \ b \ c. a \in S \implies b \in S \implies c \in S \implies a \bullet (b + c) = a \bullet b + a \bullet c$
 ⟨proof⟩

lemma *inverse-unique* [simp]:
assumes $a: a \in S \ a \neq 0$
and $x: x \in S \ a \bullet x = a\text{-id} \wedge x \bullet a = a\text{-id}$
and $y: y \in S \ a \bullet y = a\text{-id} \wedge y \bullet a = a\text{-id}$
shows $x = y$
 ⟨proof⟩

lemma *inverse-unique'*:
assumes $a: a \in S \ a \neq 0$
and *inv-ex*: $\exists x \in S. a \bullet x = a\text{-id} \wedge x \bullet a = a\text{-id}$
shows $\exists ! x \in S. a \bullet x = a\text{-id} \wedge x \bullet a = a\text{-id}$
 ⟨proof⟩

end

lemma *algebra-onI* [intro]:
fixes *scale* :: ' $a::\text{field} \Rightarrow 'b::\text{ab-group-add} \Rightarrow 'b$ (**infixr** $\langle *_S \rangle$ 75)
and *amult* (**infixr** $\langle \bullet \rangle$ 74)
assumes *vector-space-on* S *scale*

and *distR*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies (x+y) \bullet z = x \bullet z + y \bullet z$
and *distL*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies z \bullet (x+y) = z \bullet x + z \bullet y$
and *scalar-compat*: $\bigwedge a x y. \llbracket x \in S; y \in S \rrbracket \implies (a *_S x) \bullet y = a *_S (x \bullet y) \wedge x$
• $(a *_S y) = a *_S (x \bullet y)$
and *closure*: $\bigwedge x y. \llbracket x \in S; y \in S \rrbracket \implies x \bullet y \in S$
shows *algebra-on S scale amult*
<proof>

lemma (in *vector-space-on*) *scalar-compat-iff*:

fixes *scale-notation* (**infixr** $\langle *_S \rangle$ 75)
and *amult* (**infixr** $\langle \bullet \rangle$ 74)
defines *scale-notation* \equiv *scale*
assumes *distR*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies (x+y) \bullet z = x \bullet z + y \bullet z$
and *distL*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies z \bullet (x+y) = z \bullet x + z \bullet y$
shows $(\forall a. \forall x \in S. \forall y \in S. (a *_S x) \bullet y = a *_S (x \bullet y) \wedge x \bullet (a *_S y) = a *_S (x \bullet y)) \longleftrightarrow$
 $(\forall a b. \forall x \in S. \forall y \in S. (a *_S x) \bullet (b *_S y) = (a *_S b) *_S (x \bullet y))$
<proof>

lemma (in *vector-space-on*) *algebra-onI*:

fixes *scale-notation* (**infixr** $\langle *_S \rangle$ 75)
and *amult* (**infixr** $\langle \bullet \rangle$ 74)
defines *scale-notation* \equiv *scale*
assumes *distR*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies (x+y) \bullet z = x \bullet z + y \bullet z$
and *distL*: $\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \implies z \bullet (x+y) = z \bullet x + z \bullet y$
and *scalar-compat*: $\bigwedge a x y. \llbracket x \in S; y \in S \rrbracket \implies (a *_S x) \bullet y = a *_S (x \bullet y) \wedge x$
• $(a *_S y) = a *_S (x \bullet y)$
and *closure*: $\bigwedge x y. \llbracket x \in S; y \in S \rrbracket \implies x \bullet y \in S$
shows *algebra-on S scale amult*
<proof>

1.3 Lie algebra (locale)

List syntax interferes with the standard notation for the Lie bracket, so it can be disabled it here. Instead, we add a delimiter to the notation for Lie brackets, which also helps with unambiguous parsing.

locale *lie-algebra* = *algebra-on g scale lie-bracket + alternating-bilinear-on g scale lie-bracket*

for *g*

and *scale* :: *'a::field* \Rightarrow *'b::ab-group-add* \Rightarrow *'b* (**infixr** $\langle *_S \rangle$ 75)

and *lie-bracket* :: *'b* \Rightarrow *'b* \Rightarrow *'b* ($\langle [-;-] \rangle$ 74) +

assumes *jacobi*: $\llbracket x \in \mathfrak{g}; y \in \mathfrak{g}; z \in \mathfrak{g} \rrbracket \implies 0 = [x; [y; z]] + [y; [z; x]] + [z; [x; y]]$

lemma (in *algebra-on*) *lie-algebraI*:

assumes *alternating*: $\forall x \in S. \text{amult } x x = 0$

and *jacobi*: $\forall x \in S. \forall y \in S. \forall z \in S. \text{jacobi-identity amult } x y z$

shows *lie-algebra S scale amult*

<proof>

lemma (in *vector-space-on*) *lie-algebraI*:

fixes *lie-bracket* :: 'b \Rightarrow 'b \Rightarrow 'b (*<[-;-]>* 74)

and *scale-notation* (**infixr** *<*_S>* 75)

defines *scale-notation* \equiv *scale*

assumes *distributivity*:

$\bigwedge x y z. \llbracket x \in S; y \in S; z \in S \rrbracket \Longrightarrow [(x+y); z] = [x; z] + [y; z] \wedge [z; (x+y)] = [z; x] + [z; y]$

and *scalar-compatibility*:

$\bigwedge a x y. \llbracket x \in S; y \in S \rrbracket \Longrightarrow [(a *_{S} x); y] = a *_{S} ([x; y]) \wedge [x; (a *_{S} y)] = a *_{S} ([x; y])$

and *closure*: $\bigwedge x y. \llbracket x \in S; y \in S \rrbracket \Longrightarrow [x; y] \in S$

and *alternating*: $\forall x \in S. \text{lie-bracket } x x = 0$

and *jacobi*: $\forall x \in S. \forall y \in S. \forall z \in S. \text{jacobi-identity lie-bracket } x y z$

shows *lie-algebra S scale lie-bracket*

<proof>

context *lie-algebra begin*

lemma *jacobi-alt*:

assumes *x*: $x \in \mathfrak{g}$ **and** *y*: $y \in \mathfrak{g}$ **and** *z*: $z \in \mathfrak{g}$

shows $[x; [y; z]] = [[x; y]; z] + [y; [x; z]]$

<proof>

lemma *lie-subalgebra*:

assumes *h*: $\mathfrak{h} \subseteq \mathfrak{g}$ *m1.subspace h* **and** *closed*: $\bigwedge x y. x \in \mathfrak{h} \Longrightarrow y \in \mathfrak{h} \Longrightarrow \text{lie-bracket } x y \in \mathfrak{h}$

shows *lie-algebra h scale lie-bracket*

<proof>

end

1.4 Division algebras

abbreviation (in *algebra-on*) *is-left-divisor* $x a b \equiv x \in S \wedge a = \text{amult } x b$

abbreviation (in *algebra-on*) *is-right-divisor* $x a b \equiv x \in S \wedge a = \text{amult } b x$

locale *div-algebra-on = algebra-on +*

fixes *divL*:: 'a \Rightarrow 'a \Rightarrow 'a

and *divR*:: 'a \Rightarrow 'a \Rightarrow 'a

assumes *divL*: $\llbracket a \in S; b \in S; b \neq 0 \rrbracket \Longrightarrow \text{is-left-divisor } (\text{divL } a b) a b$

$\llbracket a \in S; b \in S; b \neq 0 \rrbracket \Longrightarrow \text{is-left-divisor } y a b \Longrightarrow y = (\text{divL } a b)$

and *divR*: $\llbracket a \in S; b \in S; b \neq 0 \rrbracket \Longrightarrow \text{is-right-divisor } (\text{divR } a b) a b$

$\llbracket a \in S; b \in S; b \neq 0 \rrbracket \Longrightarrow \text{is-right-divisor } y a b \Longrightarrow y = (\text{divR } a b)$

begin

In terms of the vocabulary of division rings, the expression $a = \text{divL } a b$ means that $\text{divL } a b$ is a left divisor of a , and conversely that a is a

right multiple of $divL\ a\ b$.

For $b = 0$, the divisors still exist as members of the correct type (necessarily), but they have no properties. Similarly for correctly-typed input outside the algebra.

lemma [*simp*]:
assumes $a \in S\ b \in S\ b \neq 0$
shows $divL'$: $divL\ a\ b \in S\ (divL\ a\ b) \bullet b = a\ \forall y \in S. a = y \bullet b \longrightarrow y = divL\ a\ b$
and $divR'$: $divR\ a\ b \in S\ b \bullet (divR\ a\ b) = a\ \forall y \in S. a = b \bullet y \longrightarrow y = divR\ a\ b$
<proof>
end

lemma (in *algebra-on*) *div-algebra-onI*:
assumes $\forall a \in S. \forall b \in S. b \neq 0 \longrightarrow (\exists !x \in S. a = b \bullet x) \wedge (\exists !y \in S. a = y \bullet b)$
shows *div-algebra-on* $S\ scale\ amult\ (\lambda a\ b. THE\ y. y \in S \wedge a = y \bullet b)\ (\lambda a\ b. THE\ x. x \in S \wedge a = b \bullet x)$
<proof>

lemma (in *assoc-algebra-1-on*) *div-algebra-onI'*:
fixes $ainv\ adivL\ adivR$
defines $ainv\ a \equiv (THE\ x. x \in S \wedge a-id = x \bullet a \wedge a-id = a \bullet x)$
and $adivL\ b\ a \equiv b \bullet (ainv\ a)$
and $adivR\ b\ a \equiv (ainv\ a) \bullet b$
assumes $\forall a \in S. a \neq 0 \longrightarrow (\exists x \in S. a-id = x \bullet a \wedge a-id = a \bullet x)$
shows *div-algebra-on* $S\ scale\ amult\ adivL\ adivR$
<proof>

lemma (in *assoc-algebra-on*) *div-algebra-on-imp-inverse*:
assumes *div-algebra-on* $S\ scale\ amult\ divL\ divR\ card\ S \geq 2 \vee infinite\ S$
shows $\exists a-id \in S. (\forall a \in S. a \bullet a-id = a \wedge a-id \bullet a = a) \wedge (\forall a \in S. a \neq 0 \longrightarrow divL\ a-id\ a = divR\ a-id\ a)$
<proof>

lemma (in *assoc-algebra-on*) *assoc-div-algebra-on-iff*:
assumes $card\ S \geq 2 \vee infinite\ S$
shows $(\exists\ divL\ divR. div-algebra-on\ S\ scale\ amult\ divL\ divR) \longleftrightarrow (\exists\ id. unital-algebra-on\ S\ scale\ amult\ id \wedge (\forall a \in S. a \neq 0 \longrightarrow (\exists x \in S. a \bullet x = id \wedge x \bullet a = id)))$
<proof>

locale *assoc-div-algebra-on* =
assoc-algebra-1-on $S\ scale\ amult\ a-id\ +$
div-algebra-on $S\ scale\ amult\ \lambda a\ b. amult\ a\ (a-inv\ b)\ \lambda a\ b. amult\ (a-inv\ b)\ a$
for S
and $scale :: 'a::field \Rightarrow 'b::ab-group-add \Rightarrow 'b$ (**infixr** $\langle *_{S} \rangle$ 75)
and $amult :: 'b \Rightarrow 'b \Rightarrow 'b$ (**infixr** $\langle \bullet \rangle$ 74)

```

and a-id :: 'b(<1>)
and a-inv :: 'b=>'b
begin

```

The definition *assoc-div-algebra-on* is justified by $2 \leq \text{card } S \vee \text{infinite } S \implies (\exists \text{divL divR. div-algebra-on } S (*_S) (\bullet) \text{divL divR}) = (\exists \text{id. unital-algebra-on } S (*_S) (\bullet) \text{id} \wedge (\forall a \in S. a \neq 0 \longrightarrow (\exists x \in S. a \bullet x = \text{id} \wedge x \bullet a = \text{id})))$ above: If we have an associative algebra already, the only way it can be a division algebra is to be unital as well. Since now left and right divisors can be defined through multiplicative inverses, we take only the inverse as a locale parameter, and construct the divisors. The only case we miss here (due to the requirement $\mathbf{1} \neq 0$) is the trivial algebra, which contains only the zero element (which acts as identity as well). This is for compatibility with the standard Isabelle/HOL type classes, which are subclasses of *zero-neq-one*.

```

abbreviation (input) divL :: 'b=>'b=>'b
where divL a b  $\equiv$  amult a (a-inv b)

```

```

abbreviation (input) divR :: 'b=>'b=>'b
where divR a b  $\equiv$  amult (a-inv b) a

```

```

lemma div-self-eq-id:
assumes a ∈ S a ≠ 0
shows divL a a = a-id
and divR a a = a-id
<proof>

```

```

end

```

```

locale finite-dimensional-assoc-div-algebra-on =
  assoc-div-algebra-on S scale amult a-id a-inv +
  finite-dimensional-vector-space-on S scale basis
for S :: '<'b::ab-group-add set>
and scale :: '<'a::field => 'b => 'b> (infixr '<*_S>' 75)
and amult :: '<'b=>'b=>'b> (infixr '<•>' 74)
and a-id :: '<'b> (<1>)
and a-inv :: '<'b=>'b>
and basis :: '<'b set>

```

```

lemma (in assoc-div-algebra-on) finite-dimensional-assoc-div-algebra-onI [intro]:
fixes basis :: 'b set
assumes finite-Basis: finite basis
and independent-Basis:  $\neg$  m1.dependent basis
and span-Basis: m1.span basis = S
and basis-subset: basis  $\subseteq$  S
shows finite-dimensional-assoc-div-algebra-on S scale amult a-id a-inv basis

```

<proof>

end

theory *Linear-Algebra-More*

imports

HOL-Analysis.Analysis

Smooth-Manifolds.Smooth

Transfer-Cayley-Hamilton

begin

2 Continuity of the determinant (and other maps)

lemma *continuous-on-proj: continuous-on s fst continuous-on s snd*

<proof>

lemma *continuous-on-plus:*

fixes *s::('a × 'a::topological-monoid-add) set*

shows *continuous-on s (λ(x,y). x+y)*

<proof>

lemma *continuous-on-times:*

fixes *s::('a × 'a::real-normed-algebra) set*

shows *continuous-on s (λ(x,y). x*y)*

<proof>

lemma *continuous-on-times':*

fixes *s::('a × 'a::topological-monoid-mult) set*

shows *continuous-on s (λ(x,y). x*y)*

<proof>

Only functions between *real-normed-vector* spaces can be *bounded-linear...*

lemma *continuous-on-nth-of-vec:*

fixes *s::('a::real-normed-field,'n::finite)vec set*

shows *continuous-on s (λx. x \$ n)*

<proof>

lemma *bounded-linear-mat-ijth[intro]: bounded-linear (λx. x \$ i \$ j)*

<proof>

lemma *continuous-on-ijth-of-mat:*

fixes *s::('a::real-normed-field,'n::finite)square-matrix set*

shows *continuous-on s (λx. x \$ i \$ j)*

<proof>

lemma *continuous-on-det:*

fixes *s::('a::real-normed-field,'n::finite)square-matrix set*

shows *continuous-on s det*

<proof>

lemma *invertible-inv-ex:*

fixes $a::'a::\text{semiring-1}^{\wedge}n^{\wedge}n$

assumes *invertible a*

shows $(\text{matrix-inv } a)**a = \text{mat } 1 \ a**(\text{matrix-inv } a) = \text{mat } 1$

<proof>

A similar result to the below already exists for fields, see e.g. *invertible-left-inverse*. This is more general, as it applies to any semiring (with 1).

lemma *invertible-matrix-inv:*

fixes $a::'a::\text{semiring-1}^{\wedge}n^{\wedge}n$

assumes *invertible a*

shows *invertible (matrix-inv a)*

<proof>

3 Component expressions for inverse matrices over fields

lemma *inv-adj-det-field-component:*

fixes $i\ j::'n::\text{finite}$ **and** $A\ A'::'a::\text{field}^{\wedge}n^{\wedge}n$

defines $\text{inv}A: A' \equiv \text{map-matrix } (\lambda x. x / (\det A)) (\text{adjugate } A)$

assumes *invertible A*

shows $(A**A')\$i\$j = (\text{if } i=j \text{ then } 1 \text{ else } 0)$

<proof>

lemma *inverse-adjugate-det-2:*

fixes $A::'a::\text{field}^{\wedge}n^{\wedge}n$

assumes *invertible A*

shows $\text{matrix-inv } A = \text{map-matrix } (\lambda x. x / (\det A)) (\text{adjugate } A)$

(is $\text{matrix-inv } A = ?A')$

<proof>

lemma *inverse-adjugate-det:*

fixes $A::'a::\text{field}^{\wedge}n^{\wedge}n$

assumes *invertible A*

shows $\text{matrix-inv } A = (1 / (\det A)) *_s (\text{adjugate } A)$

<proof>

lemma *transpose-component:* $(\text{transpose } A)\$i\$j = A\$j\i

<proof>

lemma *matrix-inverse-component:*

fixes $A::'a::\text{field}^{\wedge}n^{\wedge}n$ **and** $i\ j::'n::\text{finite}$

assumes *invertible A*

shows $(\text{matrix-inv } A)\$i\$j = \det (\chi\ k\ l. \text{if } k = j \wedge l = i \text{ then } 1 \text{ else if } k = j \vee l = i \text{ then } 0 \text{ else } A\ \$\ k\ \$\ l) / (\det A)$

<proof>

lemma *matrix-adjugate-component:*

fixes $A::'a::field^{n \times n}$ **and** $i\ j::'n::finite$

assumes *invertible A*

shows $(adjugate\ A)\$i\$j = det\ (\chi\ k\ l.\ if\ k = j \wedge l = i\ then\ 1\ else\ if\ k = j \vee l = i\ then\ 0\ else\ A\ \$\ k\ \$\ l)$

<proof>

4 Smoothness of real matrix operations and *det*

4.1 Smoothness of matrix multiplication

lemma *smooth-on-ijth-of-mat:*

fixes $s::('a::real-normed-field, 'n::finite)\ square-matrix\ set$

shows *smooth-on s* $(\lambda x.\ x\ \$\ i\ \$\ j)$

<proof>

Notice the following result holds only for matrices over the real numbers. (Try removing the type annotations: Isabelle automatically casts to the indicated type anyway.) This is because only real inner product spaces are defined: thus whatever "base field" a matrix is defined over, is implicitly assumed to also be a real inner product space (as is possible, for example, for \mathbb{C} with the normal inner product of \mathbb{R}^2), and the inner product is built on top of the existing one to return a *real* result.

lemma *matrix-matrix-mul-component-real:*

fixes $A::real^{k \times n}$

and $B::real^{m \times k}$

shows $A**B = (\chi\ i\ j.\ inner\ (row\ i\ A)\ (column\ j\ B))$

and $A**B = (\chi\ i\ j.\ inner\ (A\$i)\ (transpose\ B\$j))$

<proof>

lemma *matrix-inner-sum:*

shows $x \cdot y = (\sum_{i \in UNIV} \sum_{j \in UNIV} (x\$i\$j) \cdot (y\$i\$j))$

and $x \cdot y = (\sum_{(i,j) \in UNIV} (x\$i\$j) \cdot (y\$i\$j))$

<proof>

lemma *matrix-norm-sum-sqrs:*

shows $norm\ x = sqrt(\sum_{i \in UNIV} \sum_{j \in UNIV} (norm\ (x\$i\$j))^2)$

and $norm\ x = sqrt(\sum_{(i,j) \in UNIV} (norm\ (x\$i\$j))^2)$

<proof>

lemma *norm-transpose:*

shows $norm\ x = norm\ (transpose\ x)$

<proof>

lemma *matrix-norm-inner*:

fixes $x :: \text{real}^n \text{ } ^m$

shows $\text{norm } x = \text{sqrt}(\sum_{(i,j) \in \text{UNIV}} (x \$ i \$ j) \cdot (x \$ i \$ j))$

<proof>

lemma *matrix-norm-row*:

shows $\text{norm } x = \text{sqrt}(\sum_{i \in \text{UNIV}} (\text{norm } (\text{row } i \ x))^2)$

<proof>

lemma *matrix-norm-column*:

shows $\text{norm } x = \text{sqrt}(\sum_{j \in \text{UNIV}} (\text{norm } (\text{column } j \ x))^2)$

<proof>

lemma *mat-mul-indexed*: $(A ** B) \$ i \$ j = (\sum_{k \in \text{UNIV}} A \$ i \$ k * B \$ k \$ j)$

<proof>

lemma *norm-matrix-mult-ineq*:

fixes $A :: \text{real}^l \text{ } ^n$

and $B :: \text{real}^m \text{ } ^l$

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

<proof>

lemma *bounded-bilinear-matrix-mult*: *bounded-bilinear* $((**)$

$:: \text{real}^l \text{ } ^m \Rightarrow \text{real}^n \text{ } ^l \Rightarrow \text{real}^n \text{ } ^m$)

<proof>

lemma *smooth-on-matrix-mult*:

fixes $f :: 'a :: \text{real-normed-vector} \Rightarrow (\text{real}^n \text{ } ^m)$

assumes $k\text{-smooth-on } S \ f \ k\text{-smooth-on } S \ g \ \text{open } S$

shows $k\text{-smooth-on } S \ (\lambda x. f \ x ** g \ x)$

<proof>

4.2 Smoothness of \prod and *det*

lemma *higher-differentiable-on-prod*:

fixes $f :: - \Rightarrow - \Rightarrow 'c :: \{\text{real-normed-algebra}, \text{comm-monoid-mult}\}$

assumes $\bigwedge i. i \in F \Rightarrow \text{finite } F \Rightarrow \text{higher-differentiable-on } S \ (f \ i) \ n \ \text{open } S$

shows $\text{higher-differentiable-on } S \ (\lambda x. \prod_{i \in F} f \ i \ x) \ n$

<proof>

lemma *smooth-on-prod*:

fixes $f :: - \Rightarrow - \Rightarrow 'c :: \{\text{real-normed-algebra}, \text{comm-monoid-mult}\}$

assumes $(\bigwedge i. i \in F \implies \text{finite } F \implies k\text{-smooth-on } S (f i))$ *open S*
shows $k\text{-smooth-on } S (\lambda x. \prod i \in F. f i x)$
 $\langle \text{proof} \rangle$

lemma *smooth-on-det*:
fixes $s::('a::\text{real-normed-field}, 'n::\text{finite})\text{square-matrix set}$
assumes *open s*
shows $k\text{-smooth-on } s \text{ det}$
 $\langle \text{proof} \rangle$

4.3 Smoothness of matrix inversion

lemma *invertible-mat-1*: *invertible (mat 1)*
 $\langle \text{proof} \rangle$

lemma *continuous-on-vec*:
assumes $\bigwedge i. \text{continuous-on } S (\lambda x. f x \$ i)$
shows *continuous-on S f*
 $\langle \text{proof} \rangle$

lemma *frechet-derivative-eucl*:
fixes $f::'a::\text{euclidean-space} \Rightarrow 'b::\text{real-normed-vector}$
assumes *f differentiable at x*
shows $\text{frechet-derivative } f \text{ (at } x) =$
 $(\lambda v. \sum i \in \text{Basis}. (v \cdot i) *_{\mathbb{R}} \text{frechet-derivative } f \text{ (at } x) i)$
 $\langle \text{proof} \rangle$

TODO! This should maybe be changed in *Finite-Cartesian-Product.norm-le-l1-cart*.
That result only works for real^n , this one should work for all $'a::\text{real-normed-vector}^n$.

lemma *norm-le-l1-cart'*: $\text{norm } x \leq \text{sum}(\lambda i. \text{norm } (x \$ i))$ *UNIV*
 $\langle \text{proof} \rangle$

lemma *bounded-linear-vec-nth-fun*:
fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^m$
assumes $\bigwedge i. \text{bounded-linear } (\lambda x. (f x) \$ i)$
shows *bounded-linear f*
 $\langle \text{proof} \rangle$

lemma *has-derivative-vec-lambda* [*derivative-intros*]:
fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^m$
assumes $\bigwedge i. ((\lambda x. (f x) \$ i) \text{ has-derivative } (\lambda x. (f' x) \$ i)) \text{ (at } x \text{ within } s)$
shows $(f \text{ has-derivative } f') \text{ (at } x \text{ within } s)$
 $\langle \text{proof} \rangle$

lemma *has-derivative-vec-lambda-2*:
fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^m$
assumes $\bigwedge i. ((\lambda x. (f x) \$ i) \text{ has-derivative } (f' i)) \text{ (at } x \text{ within } s)$
shows $(f \text{ has-derivative } (\lambda x. \chi i. f' i x)) \text{ (at } x \text{ within } s)$
 $\langle \text{proof} \rangle$

lemma *differentiable-componentwise*:

fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^{\wedge m}$
assumes $\bigwedge i. (\lambda x. f\ x\ \$\ i)$ *differentiable (at x within s)*
shows f *differentiable (at x within s)*
 $\langle\text{proof}\rangle$

lemma *frechet-derivative-vec*:

fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^{\wedge m}$
assumes $\bigwedge i. (\lambda x. f\ x\ \$\ i)$ *differentiable (at x)*
shows $\text{frechet-derivative } f\ (\text{at } x) = (\lambda v. \chi\ i. (\text{frechet-derivative } (\lambda x. f\ x\ \$\ i)\ (\text{at } x)\ v))$
 $\langle\text{proof}\rangle$

lemma *higher-differentiable-on-vec*:

fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^{\wedge m}$
assumes $\bigwedge i. \text{higher-differentiable-on } S\ (\lambda x. (f\ x)\ \$\ i)\ n$
and *open S*
shows *higher-differentiable-on S f n*
 $\langle\text{proof}\rangle$

lemma *smooth-on-vec*:

fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{real-normed-vector}^{\wedge m}$
assumes $\bigwedge i. k\text{-smooth-on } S\ (\lambda x. (f\ x)\ \$\ i)$ *open S*
shows *k-smooth-on S f*
 $\langle\text{proof}\rangle$

lemma *smooth-on-mat*:

fixes $f::('a::\text{real-normed-vector}) \Rightarrow ('b::\text{real-normed-vector}^{\wedge k\ \wedge l})$
assumes $\bigwedge i\ j. k\text{-smooth-on } S\ (\lambda x. (f\ x)\ \$\ i\ \$\ j)$ *open S*
shows *k-smooth-on S f*
 $\langle\text{proof}\rangle$

This type constraint is annoying. The *euclidean-space* is inherited from *higher-differentiable-on-compose*, where it is marked as: ‘TODO: can we get around this restriction’. Notice this type constraint is exactly *real-normed-eucl* as defined in *Classical-Groups*.

lemma *smooth-on-matrix-inv-component*:

fixes $S::('a::\{\text{euclidean-space, real-normed-field}\}^{\wedge n\ \wedge n})$ *set*
assumes $\forall A \in S. \text{invertible } A$ *open S*
shows *k-smooth-on S* $(\lambda A. (\text{matrix-inv } A)\ \$\ i\ \$\ j)$
 $\langle\text{proof}\rangle$

lemma *fin-sum-over-delta*:

fixes $f::'n::\text{finite} \Rightarrow 'a::\text{semiring-1}$
shows $(\sum (i::'n::\text{finite}) \in \text{UNIV}. ((\text{if } i=j \text{ then } 1 \text{ else } 0) * f\ i)) = f\ j$
 $\langle\text{proof}\rangle$

lemma *matrix-is-linear-map*:

fixes $A :: ('a :: \{real\text{-algebra-1}, comm\text{-semiring-1}\})^{\wedge m} \wedge^{\wedge n}$ — again, real-based entries only...

shows $linear ((*v) A) \wedge matrix ((*v) A) = A$
<proof>

lemma *smooth-on-matrix-inv*:

assumes $\forall A. A \in S \longrightarrow invertible A \text{ open } S$

shows $k\text{-smooth-on } S \text{ (matrix-inv :: 'a :: \{euclidean-space, real-normed-field\})}^{\wedge n} \wedge^{\wedge n} \Rightarrow 'a \wedge^{\wedge n} \wedge^{\wedge n}$
<proof>

end

5 Smooth vector fields

theory *Smooth-Vector-Fields*

imports

More-Manifolds

begin

Type synonyms for use later: these already follow our later split between defining “charts” for the tangent bundle as a product, and talking about vector fields as maps $p \mapsto v \in T_p M$ as well as sections of the tangent bundle $M \rightarrow TM$.

type-synonym $'a \text{ tangent-bundle} = 'a \times (('a \Rightarrow real) \Rightarrow real)$

type-synonym $'a \text{ vector-field} = 'a \Rightarrow (('a \Rightarrow real) \Rightarrow real)$

5.1 (Smooth) vector fields on an (entire) manifold.

Since we only get an isomorphism between tangent vectors and directional derivatives in the smooth case of $k = \infty$, we create a locale for infinitely smooth manifolds.

locale *smooth-manifold* = *c-manifold charts* ∞ **for** *charts*

context *c-manifold* **begin**

5.1.1 Charts for the tangent bundle

definition *in-TM* :: $'a \Rightarrow (('a \Rightarrow real) \Rightarrow real) \Rightarrow bool$

where $in\text{-TM } p \ v \equiv p \in carrier \wedge v \in tangent\text{-space } p$

abbreviation $TM \equiv \{(p, v). in\text{-TM } p \ v\}$

lemma *in-TM-E* [*elim*]:

assumes $in-TM\ p\ v$
shows $v \in tangent-space\ p\ p \in carrier$
 $\langle proof \rangle$

lemma $TM-PairE\ [elim]$:
assumes $(p,v) \in TM$
shows $v \in tangent-space\ p\ p \in carrier$
 $\langle proof \rangle$

lemma $TM-E\ [elim]$:
assumes $x \in TM$
shows $snd\ x \in tangent-space\ (fst\ x)\ fst\ x \in carrier$
 $\langle proof \rangle$

We can construct a chart for $tangent-space\ p$ given a chart around p . Notice the appearance of $charts$ in the definition, which specifies that we're charting the set $tangent-space\ p$, not $c-manifold.tangent-space\ (charts-submanifold\ c) \times p$.

definition $apply-chart-TM :: ('a,'b)chart \Rightarrow 'a\ tangent-bundle \Rightarrow 'b \times 'b$
where $apply-chart-TM\ c \equiv \lambda(p,v). (c\ p\ ,\ c-manifold-point.tangent-chart-fun\ charts \times c\ p\ v)$

definition $inv-chart-TM :: ('a,'b)chart \Rightarrow ('b \times 'b) \Rightarrow 'a \times (('a \Rightarrow real) \Rightarrow real)$
where $inv-chart-TM\ c \equiv \lambda((p::'b),(v::'b)). (inv-chart\ c\ p\ ,\ c-manifold-point.coordinate-vector\ charts \times c\ (inv-chart\ c\ p)\ v)$

definition $domain-TM :: ('a,'b)\ chart \Rightarrow ('a \times (('a \Rightarrow real) \Rightarrow real))\ set$
where $domain-TM\ c \equiv \{(p, v). p \in domain\ c \wedge v \in tangent-space\ p\}$

definition $codomain-TM :: ('a,'b)\ chart \Rightarrow ('b \times 'b)\ set$
where $codomain-TM\ c \equiv \{(p, v). p \in codomain\ c\}$

definition $restrict-chart-TM\ S\ c \equiv apply-chart-TM\ (restrict-chart\ S\ c)$

definition $restrict-domain-TM\ S\ c \equiv domain-TM\ (restrict-chart\ S\ c)$

definition $restrict-codomain-TM\ S\ c \equiv codomain-TM\ (restrict-chart\ S\ c)$

definition $restrict-inv-chart-TM\ S\ c \equiv inv-chart-TM\ (restrict-chart\ S\ c)$

5.1.2 Proofs about $apply-chart-TM$ that mimic the properties of $('a, 'b)\ chart$.

lemma $domain-TM$:
assumes $c \in atlas$
shows $domain-TM\ c \subseteq TM$
 $\langle proof \rangle$

lemma $codomain-TM-alt$: $codomain-TM\ c = codomain\ c \times (UNIV :: 'b\ set)$
 $\langle proof \rangle$

lemma $open-codomain-TM$:

assumes $c \in \text{atlas}$
shows $\text{open} (\text{codomain-TM } c)$
 $\langle \text{proof} \rangle$

end

context *smooth-manifold* **begin**

lemma *apply-chart-TM-inverse* [*simp*]:

assumes $c: c \in \text{atlas}$
shows $\bigwedge p v. (p,v) \in \text{domain-TM } c \implies \text{inv-chart-TM } c (\text{apply-chart-TM } c (p,v))$
 $= (p,v)$
and $\bigwedge x u. (x,u) \in \text{codomain-TM } c \implies \text{apply-chart-TM } c (\text{inv-chart-TM } c$
 $(x,u)) = (x,u)$
 $\langle \text{proof} \rangle$

lemma *image-domain-TM-eq*:

assumes $c \in \text{atlas}$
shows $\text{apply-chart-TM } c \text{ ' domain-TM } c = \text{codomain-TM } c$
 $\langle \text{proof} \rangle$

lemma *inv-image-codomain-TM-eq*:

assumes $c \in \text{atlas}$
shows $\text{inv-chart-TM } c \text{ ' codomain-TM } c = \text{domain-TM } c$
 $\langle \text{proof} \rangle$

lemma (**in** *c-manifold*) *restrict-domain-TM-intersection*:

shows $\text{restrict-domain-TM} (\text{domain } c1 \cap \text{domain } c2) c1 = \text{domain-TM } c1 \cap$
 $\text{domain-TM } c2$
 $\langle \text{proof} \rangle$

lemma (**in** *c-manifold*) *restrict-domain-TM-intersection'*:

shows $\text{restrict-domain-TM} (\text{domain } c1 \cap \text{domain } c2) c2 = \text{domain-TM } c1 \cap$
 $\text{domain-TM } c2$
 $\langle \text{proof} \rangle$

lemma (**in** *c-manifold*) *restrict-domain-TM*:

assumes $\text{open } S \subseteq \text{domain } c$
shows $\text{restrict-domain-TM } S c = \{(p, v). p \in S \wedge v \in \text{tangent-space } p\}$
 $\langle \text{proof} \rangle$

lemma *image-restrict-domain-TM-eq*:

assumes $c \in \text{atlas}$
shows $\text{restrict-chart-TM } S c \text{ ' restrict-domain-TM } S c = \text{restrict-codomain-TM}$

$S \ c$
 $\langle \text{proof} \rangle$

lemma *inv-image-restrict-codomain-TM-eq*:
assumes $c \in \text{atlas}$
shows $\text{restrict-inv-chart-TM } S \ c \ ' \ \text{restrict-codomain-TM } S \ c = \text{restrict-domain-TM } S \ c$
 $\langle \text{proof} \rangle$

lemma *codomain-restrict-chart-TM[simp]*:
assumes $c \in \text{atlas}$ *open* S
shows $\text{restrict-codomain-TM } S \ c = \text{codomain-TM } c \cap \text{inv-chart-TM } c \ - \ ' \ \{(p, v). p \in S \wedge v \in \text{tangent-space } p\}$
 $\langle \text{proof} \rangle$

lemma (*in c-manifold*) *image-subset-TM-eq [simp]*:
assumes $S \subseteq \text{domain-TM } c$
shows $\text{apply-chart-TM } c \ ' \ S \subseteq \text{codomain-TM } c$
 $\langle \text{proof} \rangle$

lemma (*in c-manifold*) *image-subset-restrict-TM-eq [simp]*:
assumes $T \subseteq \text{restrict-domain-TM } S \ c$
shows $\text{restrict-chart-TM } S \ c \ ' \ T \subseteq \text{restrict-codomain-TM } S \ c$
 $\langle \text{proof} \rangle$

lemma *restrict-chart-domain-Int*:
assumes $c1 \in \text{atlas}$
shows $\text{apply-chart-TM } c1 \ ' \ (\text{domain-TM } c1 \cap \text{domain-TM } c2) = \text{restrict-chart-TM } (\text{domain } c1 \cap \text{domain } c2) \ c1 \ ' \ (\text{restrict-domain-TM } (\text{domain } c1 \cap \text{domain } c2) \ c1)$
(**is** $\langle ?\text{TM-dom-Int} = ?\text{restr-TM-dom} \rangle$)
 $\langle \text{proof} \rangle$

lemma *open-intersection-TM*:
assumes $c1 \in \text{atlas}$
shows *open* $(\text{apply-chart-TM } c1 \ ' \ (\text{domain-TM } c1 \cap \text{domain-TM } c2))$
 $\langle \text{proof} \rangle$

lemma *apply-restrict-chart-TM*:
assumes $c: c \in \text{atlas}$ **and** $S: \text{open } S \ S \subseteq \text{domain } c \ x \in \text{restrict-domain-TM } S \ c$
shows $\text{apply-chart-TM } c \ x = \text{restrict-chart-TM } S \ c \ x$
 $\langle \text{proof} \rangle$

lemma *inverse-restrict-chart-TM*:

assumes $c: c \in \text{atlas}$ **and** $S: \text{open } S \ S \subseteq \text{domain } c \ x \in \text{restrict-codomain-TM } S$
 c
shows $\text{inv-chart-TM } c \ x = \text{restrict-inv-chart-TM } S \ c \ x$
 $\langle \text{proof} \rangle$

lemma (**in** $c\text{-manifold-point}$) $d\kappa\text{-inv-directional-derivative-eq}$:
assumes $k = \infty$
shows $d\kappa^{-1} (\text{directional-derivative } k \ (\psi \ p) \ x) = \text{restrict0} (\text{diffeo-}\psi.\text{dest.diff-fun-space})$
 $(\lambda f. \text{frechet-derivative } f \ (\text{at } (\psi \ p)) \ x)$
 $\langle \text{proof} \rangle$

lemma $\text{smooth-on-compat-charts-TM}$:
assumes $c1 \in \text{atlas}$ $c2 \in \text{atlas}$
shows $\text{smooth-on} (c1 \text{ ' } (\text{domain } c1 \cap \text{domain } c2) \times \text{UNIV})$
 $(\lambda x. \text{frechet-derivative} ((\lambda y. (\text{restrict-chart} (\text{domain } c1 \cap \text{domain } c2) \ c2) \ y \cdot$
 $i) \circ \text{inv-chart} (\text{restrict-chart} (\text{domain } c1 \cap \text{domain } c2) \ c1)) (\text{at } (\text{fst } x)) (\text{snd } x))$
 $(\text{is } \langle \text{smooth-on } ?D \ (\lambda x. \text{frechet-derivative} ((\lambda y. ?r2 \ y \cdot i) \circ ?r1i) (\text{at } (\text{fst } x))$
 $(\text{snd } x)) \rangle)$
 $\langle \text{proof} \rangle$

lemma atlas-TM :
assumes $c1 \in \text{atlas}$ $c2 \in \text{atlas}$
shows $\text{smooth-on} ((\text{apply-chart-TM } c1) \text{ ' } (\text{domain-TM } c1 \cap \text{domain-TM } c2))$
 $((\text{apply-chart-TM } c2) \circ (\text{inv-chart-TM } c1))$
 $(\text{is } \langle \text{smooth-on } (?c1 \text{ ' } (?dom1 \cap ?dom2)) ((?c2) \circ (?i1)) \rangle)$
 $\langle \text{proof} \rangle$

lemma $\text{atlas-TM}'$:
assumes $c1 \in \text{atlas}$ $c2 \in \text{atlas}$
shows $\text{smooth-on} ((\text{apply-chart-TM } c2) \text{ ' } (\text{domain-TM } c1 \cap \text{domain-TM } c2))$
 $((\text{apply-chart-TM } c1) \circ (\text{inv-chart-TM } c2))$
 $\langle \text{proof} \rangle$

end

5.1.3 Differentiability of vector fields

context $c\text{-manifold}$ **begin**

abbreviation $k\text{-diff-from-M-to-TM-at-in} :: \text{enat} \Rightarrow 'a \Rightarrow ('a, 'b) \text{chart} \Rightarrow ('a \Rightarrow 'a$
 $\text{tangent-bundle}) \Rightarrow \text{bool}$

where $k\text{-diff-from-M-to-TM-at-in } k' \ x \ c \ X \equiv x \in \text{domain } c \wedge X \text{ ' } \text{domain } c \subseteq$
 $\text{domain-TM } c \wedge k'\text{-smooth-on} (\text{codomain } c) (\text{apply-chart-TM } c \circ X \circ \text{inv-chart } c)$

— Compare this definition to $\text{diff-axioms } ?k \ ?charts1.0 \ ?charts2.0 \ ?f \equiv \forall x. x \in$
 $\text{manifold.carrier } ?charts1.0 \longrightarrow (\exists c1 \in c\text{-manifold.atlas } ?charts1.0 \ ?k. \exists c2 \in c\text{-manifold.atlas}$

?charts2.0 ?k. $x \in \text{domain } c1 \wedge ?f \text{ ' domain } c1 \subseteq \text{domain } c2 \wedge ?k\text{-smooth-on}$
(codomain c1) (apply-chart c2 \circ ?f \circ inv-chart c1)). It's the same, except the
charts for TM aren't of type ('a, 'b) chart.

definition *k-diff-from-M-to-TM* ($\langle \text{--diff'-from'-M'-to'-TM} \rangle$ [1000])

where *diff-from-M-to-TM-def: $k'\text{-diff-from-M-to-TM } X \equiv \forall x. x \in \text{carrier} \longrightarrow$*
($\exists c \in \text{atlas. } k\text{-diff-from-M-to-TM-at-in } k' x c X$)

abbreviation *continuous-from-M-to-TM $\equiv 0\text{-diff-from-M-to-TM}$*

abbreviation (in *smooth-manifold*) *smooth-from-M-to-TM $\equiv k\text{-diff-from-M-to-TM}$*

∞

lemma *diff-from-M-to-TM-E:*

assumes *$k'\text{-diff-from-M-to-TM } X x \in \text{carrier}$*

obtains *c where $c \in \text{atlas } x \in \text{domain } c X \text{ ' domain } c \subseteq \text{domain-TM } c k'\text{-smooth-on}$*
(codomain c) (apply-chart-TM $c \circ X \circ \text{inv-chart } c$)

\langle proof \rangle

lemma *continuous-from-M-to-TM-D:*

assumes *continuous-from-M-to-TM $X x \in \text{carrier}$*

obtains *c where $c \in \text{atlas } x \in \text{domain } c X \text{ ' domain } c \subseteq \text{domain-TM } c \text{ continu-}$*
ous-on (codomain c) (apply-chart-TM $c \circ X \circ \text{inv-chart } c$)

\langle proof \rangle

definition *section-of-TM-def: section-of-TM-on $S X \equiv \forall p \in S. (X p) \in \text{TM} \wedge \text{fst}$*
($X p$) = p

abbreviation *section-of-TM $\equiv \text{section-of-TM-on carrier}$*

lemma *section-of-TM-subset:*

assumes *section-of-TM-on $S X T \subseteq S$*

shows *section-of-TM-on $T X$*

\langle proof \rangle

lemma *section-domain-TM:*

assumes *section-of-TM-on (domain c) X*

shows *$X \text{ ' domain } c \subseteq \text{domain-TM } c$*

\langle proof \rangle

lemma *section-domain-TM':*

assumes *section-of-TM $X c \in \text{atlas}$*

shows *$X \text{ ' domain } c \subseteq \text{domain-TM } c$*

\langle proof \rangle

lemma *section-vimage-domain-TM:*

assumes *section-of-TM $X c \in \text{atlas}$*

shows *$\text{carrier} \cap X \text{ - ' domain-TM } c = \text{domain } c$*

\langle proof \rangle

end

context *smooth-manifold begin*

Show that a smooth/differentiable vector field is smooth in any chart. This would be $\llbracket \text{diff } ?k \text{ } ?charts1.0 \text{ } ?charts2.0 \text{ } ?f; ?d1.0 \in c\text{-manifold.atlas } ?charts1.0 \text{ } ?k; ?d2.0 \in c\text{-manifold.atlas } ?charts2.0 \text{ } ?k \rrbracket \implies ?k\text{-smooth-on } (codomain \text{ } ?d1.0 \cap inv\text{-chart } ?d1.0 - ' (manifold.carrier \text{ } ?charts1.0 \cap ?f - ' domain \text{ } ?d2.0)) (apply\text{-chart } ?d2.0 \circ ?f \circ inv\text{-chart } ?d1.0)$ if we could write TM as a c -manifold; it relies on the compatibility of charts for TM given in $\llbracket smooth\text{-manifold } ?charts; ?c1.0 \in c\text{-manifold.atlas } ?charts \infty; ?c2.0 \in c\text{-manifold.atlas } ?charts \infty \rrbracket \implies smooth\text{-on } (c\text{-manifold.apply-chart-TM } ?charts \text{ } ?c1.0 - ' (c\text{-manifold.domain-TM } ?charts \infty \text{ } ?c1.0 \cap c\text{-manifold.domain-TM } ?charts \infty \text{ } ?c2.0)) (c\text{-manifold.apply-chart-TM } ?charts \text{ } ?c2.0 \circ c\text{-manifold.inv-chart-TM } ?charts \text{ } ?c1.0)$.

lemma *diff-from-M-to-TM-chartsD*:

assumes $X: k\text{-diff-from-M-to-TM } k' \text{ } X \text{ section-of-TM } X$ **and** $c: c \in atlas$
shows $k'\text{-smooth-on } (codomain \text{ } c) (apply\text{-chart-TM } c \circ X \circ inv\text{-chart } c)$
<proof>

definition *smooth-section-of-TM* $X \equiv section\text{-of-TM } X \wedge smooth\text{-from-M-to-TM } X$

abbreviation *set-of-smooth-sections-of-TM* (\mathfrak{X})

where $set\text{-of-smooth-sections-of-TM} \equiv \{X. smooth\text{-section-of-TM } X\}$

lemma *in \mathfrak{X} -E*:

assumes $X \in \mathfrak{X} \text{ } p \in carrier$
shows $(\exists c \in atlas. p \in domain \text{ } c \wedge X - ' domain \text{ } c \subseteq domain\text{-TM } c \wedge smooth\text{-on } (codomain \text{ } c) (apply\text{-chart-TM } c \circ X \circ inv\text{-chart } c))$
and $snd \text{ } (X \text{ } p) \in tangent\text{-space } p$
and $fst \text{ } (X \text{ } p) = p$
<proof>

lemma *in \mathfrak{X} -chartsD*:

assumes $X \in \mathfrak{X} \text{ } c \in atlas$
shows $smooth\text{-on } (codomain \text{ } c) (apply\text{-chart-TM } c \circ X \circ inv\text{-chart } c)$
<proof>

end

A vector field is smooth if it is smooth as a map $M \rightarrow TM$. As a shortcut, we define a smooth vector field as one that is smooth in the chart - this avoids problems with defining a $('a \times (('a \Rightarrow real) \Rightarrow real), 'b)$ chart. We also introduce a duality of predicates with strongly related meaning: this allows us to consider vector fields as either maps $'a \Rightarrow ('a \Rightarrow real) \Rightarrow real$, i.e. mapping a point to a vector; or maps $'a \Rightarrow 'a \times (('a \Rightarrow real) \Rightarrow real)$,

i.e. sections of TM properly speaking.

context *c-manifold* **begin**

definition *rough-vector-field* :: 'a vector-field \Rightarrow bool

where *rough-vector-field* $X \equiv$ extensional0 carrier $X \wedge (\forall p \in \text{carrier}. X p \in \text{tangent-space } p)$

lemma *rough-vector-fieldE* [elim]:

assumes *rough-vector-field* X

shows $\bigwedge p. X p \in \text{tangent-space } p$ extensional0 carrier X

<proof>

lemma *rough-vector-field-subset*:

assumes *rough-vector-field* $X T \subseteq \text{carrier}$

shows *rough-vector-field* (*restrict0* $T X$)

<proof>

end

abbreviation (*input*) *vec-field-apply-fun* :: 'a vector-field \Rightarrow ('a \Rightarrow real) \Rightarrow ('a \Rightarrow real)
(**infix** '<'> 100)

where *vec-field-apply-fun* $X f \equiv \lambda p. X p f$

lemma (**in** *c-manifold*) *vec-field-apply-fun-cong*:

assumes X : *rough-vector-field* X **and** U : open $U U \subseteq \text{carrier} \forall x \in U. f x = g x$

and f : $f \in \text{diff-fun-space}$ **and** g : $g \in \text{diff-fun-space}$

shows $\forall p \in U. X p f = X p g$

<proof>

lemma (**in** *c-manifold*) *ext0-vec-field-apply-fun*:

assumes X : *rough-vector-field* X

shows extensional0 *diff-fun-space* (*vec-field-apply-fun* X)

<proof>

5.2 Smoothness criterion for a vector field in a single chart.

A smooth vector field is one that is infinitely differentiable when expanded in the charting Euclidean space using $\llbracket c\text{-manifold-point } ?charts \ ?k \ ?\psi \ ?p; \ ?v \in c\text{-manifold.tangent-space } ?charts \ ?k \ ?p; \ ?k = \infty \rrbracket \Longrightarrow \ ?v = (\sum_{i \in \text{Basis}. c\text{-manifold-point.component-function } ?charts \ ?k \ ?\psi \ ?p} \ ?v \ i *_{\mathbb{R}} c\text{-manifold-point.coordinate-vector } ?charts \ ?k \ ?\psi \ ?p \ i)$. This should be the chart that makes each tangent space into a manifold anyway, but the type constraints are tricky to satisfy.

Since tangent spaces at the same point differ between a manifold and a submanifold, it's important to note that the differentiability condition can be relaxed to only apply to a subset, but the tangent bundle is always the

disjoint union of tangent spaces of the *entire* manifold, which implies the chart function for the tangent space is defined in the entire manifold, not a submanifold.

```

locale smooth-vector-field-local = c-manifold-local charts ∞  $\psi$  for charts  $\psi$  +
  fixes  $X$ 
  assumes vector-field:  $\forall p \in \text{domain } \psi. X p \in \text{tangent-space } p$ 
  and smooth-in-chart: diff-fun ∞ (charts-submanifold (domain  $\psi$ )) ( $\lambda p. (c\text{-manifold-point.tangent-chart-fun } \text{charts} \ \psi \ p) (X \ p)$ )
begin
lemma rough-vector-field: rough-vector-field (restrict0 (domain  $\psi$ )  $X$ )
   $\langle \text{proof} \rangle$ 
end

```

5.2.1 Connecting the types $'a \Rightarrow ('a \Rightarrow \text{real}) \Rightarrow \text{real}$ (used for *smooth-vector-field-local*)
and $'a \Rightarrow 'a \times (('a \Rightarrow \text{real}) \Rightarrow \text{real})$ (used for $\lambda \text{charts } k. c\text{-manifold.section-of-TM-on charts } k$ (*manifold.carrier charts*)).

context *c-manifold* **begin**

```

lemma fst-apply-chart-TM-id [simp]: (fst ∘ (apply-chart-TM  $\psi$  ∘  $X$  ∘ inv-chart  $\psi$ ))  $x = x$ 
if section-of-TM-on (domain  $\psi$ )  $X \ \psi \in \text{atlas } x \in \text{codomain } \psi$  for  $x$ 
   $\langle \text{proof} \rangle$ 

```

The justification for the definition of *smooth-vector-field-local* is the lemma below, connecting it to the smoothness requirement used to define the set of smooth sections \mathfrak{X} .

```

lemma apply-chart-TM-chartX:
  fixes  $X :: ('a \Rightarrow 'a \times (('a \Rightarrow \text{real}) \Rightarrow \text{real}))$  and  $c :: ('a, 'b)$  chart and chart-X
   $:: 'a \Rightarrow 'b$ 
  defines chart-X  $\equiv \lambda p. (c\text{-manifold-point.tangent-chart-fun } \text{charts} \ \infty \ c \ p) (\text{snd } (X \ p))$ 
  assumes  $k: k = \infty$  and  $X: \text{section-of-TM-on } (\text{domain } c) \ X$  and  $c: c \in \text{atlas}$ 
  shows smooth-on (codomain  $c$ ) (apply-chart-TM  $c \ \circ \ X \ \circ \ \text{inv-chart } c$ )  $\longleftrightarrow$  diff-fun
  ∞ (charts-submanifold (domain  $c$ )) chart-X
  (is  $\langle ?\text{smooth-in-chart-TM } c \ X \ \longleftrightarrow \ ?\text{diff-domain } c \ \text{chart-X} \rangle$ )
   $\langle \text{proof} \rangle$ 

```

end

context *smooth-vector-field-local* **begin**

```

definition chart-X  $\equiv \lambda p. (c\text{-manifold-point.tangent-chart-fun } \text{charts} \ \infty \ \psi \ p) (X \ p)$ 

```

```

lemma smooth-in-chart-X [simp]: diff-fun ∞ (charts-submanifold (domain  $\psi$ )) chart-X

```

<proof>

lemma *apply-chart-TM-chart-X*:

*smooth-on (codomain ψ) (apply-chart-TM $\psi \circ (\lambda p. (p, X p)) \circ \text{inv-chart } \psi) \longleftrightarrow$
diff-fun ∞ (charts-submanifold (domain ψ)) chart-X
*<proof>**

end

5.2.2 Some theorems about smooth vector fields, locally and globally.

context *c-manifold-local begin*

It is often convenient to keep a stronger handle on which chart we're (locally) working in. Since the first component of the *apply-chart-TM* is just the identity, we can safely omit it for a lot of our reasoning about smoothness in a chart (see $\llbracket \text{section-of-TM-on (domain ?}\psi) ?X; ?\psi \in \text{atlas}; ?x \in \text{codomain ?}\psi \rrbracket \implies (\text{fst} \circ (\text{apply-chart-TM } ?\psi \circ ?X \circ \text{inv-chart } ?\psi)) ?x = ?x$ and $\llbracket k = \infty; \text{section-of-TM-on (domain ?}c) ?X; ?c \in \text{atlas} \rrbracket \implies \text{smooth-on (codomain ?}c) (\text{apply-chart-TM } ?c \circ ?X \circ \text{inv-chart } ?c) = \text{diff-fun } \infty (\text{charts-submanifold (domain ?}c)) (\lambda p. \text{c-manifold-point.tangent-chart-fun charts } \infty ?c p (\text{snd } (?X p)))$).

definition *vector-field-component* :: ('a \Rightarrow (('a \Rightarrow real) \Rightarrow real)) \Rightarrow 'b \Rightarrow 'a \Rightarrow real

where *vector-field-component* $X i \equiv \lambda p. (\text{c-manifold-point.component-function charts } k \psi p) (X p) i$

definition *coordinate-vector-field* :: 'b \Rightarrow ('a \Rightarrow (('a \Rightarrow real) \Rightarrow real))

where *coordinate-vector-field* $i p \equiv \text{c-manifold-point.coordinate-vector charts } k \psi p i$

Eqn. 8.2, page 175, Lee 2012

lemma *vector-field-local-representation*:

assumes $k: k = \infty$ **and** $X: \text{rough-vector-field } X$ **and** $p: p \in \text{domain } \psi$

shows $X p = (\sum_{i \in \text{Basis}. (\text{vector-field-component } X i p) *_{\mathbb{R}} (\text{coordinate-vector-field } i p))$

<proof>

definition *local-coord-at* :: 'a \Rightarrow 'b \Rightarrow 'a \Rightarrow real

where *local-coord-at* $q i \equiv \text{restrict0 (domain } \psi) (\lambda y::'a. (\psi y - \psi q) \cdot i)$

lemma *local-coord-diff-fun*:

assumes $k: k = \infty$ **and** $q: q \in \text{domain } \psi$

shows *local-coord-at* $q i \in \text{sub-}\psi.\text{sub.diff-fun-space}$

<proof>

lemma *vector-apply-coord-at*:

fixes x_ψ **defines** [*simp*]: $x_\psi \equiv \text{local-coord-at}$

assumes $q: q \in \text{domain } \psi$ **and** $p: p \in \text{domain } \psi$ **and** $X: X \in \text{tangent-space } q$ **and**
 $k: k = \infty$

shows $(d\iota^{-1} q) X (x_\psi p i) = (d\iota^{-1} q) X (x_\psi q i)$

<proof>

end

context *c-manifold* **begin**

abbreviation (*input*) *real-linear-on S1 S2* \equiv *linear-on S1 S2 scaleR scaleR*

— Sometimes we want to apply a vector field meaningfully to a function that is in the *c-manifold.diff-fun-space* of a submanifold (e.g. a single chart). For this to make sense, the function has to be in the correct space, and the submanifold's carrier set has to be open.

definition *vec-field-apply-fun-in-at* :: (*'a vector-field*) \Rightarrow (*'a \Rightarrow real*) \Rightarrow *'a set* \Rightarrow *'a \Rightarrow real*

where *vec-field-apply-fun-in-at* $X f U q = \text{restrict0 } (\text{tangent-space } q)$

(*the-inv-into*

(*c-manifold.tangent-space (charts-submanifold U) k q*)

(*diff.push-forward k (charts-submanifold U) charts ($\lambda x. x$)*))

($X q$) f

abbreviation *vec-field-restr* :: (*'a vector-field*) \Rightarrow *'a set* \Rightarrow (*'a vector-field*)

where *vec-field-restr* $X U q f \equiv \text{restrict0 } U (\text{vec-field-apply-fun-in-at } X f U) q$

notation *vec-field-restr* ($\langle \cdot \rangle$) [*60,60*]

lemma (**in** *smooth-manifold*) *vec-field-restr*: $(X \upharpoonright U) p \in \text{c-manifold.tangent-space } (\text{charts-submanifold } U) \infty p$

if *open U* $U \subseteq \text{carrier rough-vector-field } X$ **for** $U X$

<proof>

lemma *vec-field-apply-fun-alt'*:

assumes *open U* $q \in U$ $f \in \text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) k$
rough-vector-field X

shows *vec-field-apply-fun-in-at* $X f U q = (\text{the-inv-into } (\text{c-manifold.tangent-space } (\text{charts-submanifold } U) k q) (\text{diff.push-forward } k (\text{charts-submanifold } U) \text{charts } (\lambda x. x))) (X q) f$

<proof>

lemma *vec-field-apply-fun-alt*:

assumes *open U* $q \in U$ $f \in \text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) k$
rough-vector-field X

shows *vec-field-restr* $X U q f = (\text{the-inv-into } (\text{c-manifold.tangent-space } (\text{charts-submanifold } U) k q) (\text{diff.push-forward } k (\text{charts-submanifold } U) \text{charts } (\lambda x. x))) (X q) f$

<proof>

lemma (in *submanifold*) *vec-field-apply-fun-sub*:
assumes $q \in \text{carrier } q \in S \ f \in \text{sub.diff-fun-space rough-vector-field } X$
shows *vec-field-apply-fun-in-at* $X \ f \ (S \cap \text{carrier}) \ q = (\text{the-inv-into } (\text{sub.tangent-space } q) \ \text{inclusion.push-forward}) \ (X \ q) \ f$
 ⟨proof⟩

lemma *vec-field-apply-fun-in-open[simp]*: *vec-field-apply-fun-in-at* $X \ f' \ U \ p = X \ p \ f$
if $U: p \in U \ \text{open } U \ U \subseteq \text{carrier}$
and $f: f \in \text{diff-fun-space } f' \in \text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) \ k \ \forall x \in U. f \ x = f' \ x$
and $X: \text{rough-vector-field } X$
 ⟨proof⟩

lemma *open-imp-submanifold*: *submanifold charts* $k \ S$ **if** *open* S
 ⟨proof⟩

lemmas *charts-submanifold* = *submanifold.charts-submanifold*[*OF open-imp-submanifold*]

lemma *charts-submanifold-Int*:
manifold.charts-submanifold (*charts-submanifold* U) $N = \text{charts-submanifold } (N \cap U)$
if *open* N *open* U
 ⟨proof⟩

lemma *vec-field-apply-fun-in-restrict0[simp]*:
vec-field-restr $X \ U \ p \ f = \text{vec-field-restr } X \ N \ p \ (\text{restrict0 } N \ f)$
if $U: \text{open } U \ U \subseteq \text{carrier}$ **and** $N: p \in N \ N \subseteq U \ \text{open } N$
and $f: f \in \text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) \ k$
and $X: \text{rough-vector-field } X$
 ⟨proof⟩

lemma (in *submanifold*) *vec-field-apply-fun-in-open[simp]*:
vec-field-restr $X \ S \ p \ f' = X \ p \ f$
if $S: S \subseteq \text{carrier}$
and $N: \text{open } N \ N \subseteq S \ p \in N$
and $f: f \in \text{diff-fun-space } f' \in \text{sub.diff-fun-space } \forall x \in N. f \ x = f' \ x$
and $X: \text{rough-vector-field } X$
 ⟨proof⟩

lemma (in *smooth-manifold*) *vec-field-apply-fun-in-restrict0'*:
restrict0 $U \ (X \ f) = X \ \upharpoonright U \ (\text{restrict0 } U \ f)$
if $U: \text{open } U \ U \subseteq \text{carrier}$ **and** $f: f \in \text{diff-fun-space}$ **and** $X: \text{rough-vector-field } X$

for $U X f$
 ⟨proof⟩

lemma (in *submanifold*) *vec-field-apply-fun-in-open* [simp]:
vec-field-restr $X S p f' = X p f$
if $S: p \in S \ S \subseteq \text{carrier}$
and $f: f \in \text{diff-fun-space} \ f' \in \text{sub.diff-fun-space} \ \forall x \in S. f x = f' x$
and $X: \text{rough-vector-field } X$
 ⟨proof⟩

lemma (in *c-manifold*) *vec-field-apply-fun-in-chart* [simp]:
vec-field-apply-fun-in-at $X f (\text{domain } c) p = X p f$
if $p: p \in \text{domain } c$ **and** $c: c \in \text{atlas}$
and $f: f \in \text{diff-fun-space} \ f \in \text{c-manifold.diff-fun-space} (\text{charts-submanifold} (\text{domain } c)) k$
and $X: \text{rough-vector-field } X$
 ⟨proof⟩

end

context *c-manifold-local* **begin**

lemma *vec-field-apply-fun-eq-component*:
fixes x_ψ **defines** [simp]: $x_\psi \equiv \text{local-coord-at}$
assumes $q: q \in \text{domain } \psi$ **and** $p: p \in \text{domain } \psi$ **and** $X: \text{rough-vector-field } X$ **and**
 $k: k = \infty$
shows *vec-field-apply-fun-in-at* $X (x_\psi q i) (\text{domain } \psi) q = \text{vector-field-component} X i q$
 ⟨proof⟩

Prop 8.1, page 175, Lee 2012. The main difference is that our vector field X here is only a map $M \rightarrow \text{snd } TM$, not a section $M \rightarrow TM$ properly speaking. See also $\llbracket k = \infty; \text{section-of-TM-on } (\text{domain } ?c) ?X; ?c \in \text{atlas} \rrbracket \implies \text{smooth-on } (\text{codomain } ?c) (\text{apply-chart-TM } ?c \circ ?X \circ \text{inv-chart } ?c) = \text{diff-fun } \infty (\text{charts-submanifold } (\text{domain } ?c)) (\lambda p. \text{c-manifold-point.tangent-chart-fun charts } \infty ?c p (\text{snd } (?X p)))$.

lemma *vector-field-smooth-local-iff*:
assumes $k: k = \infty$ **and** $X: \forall p \in \text{domain } \psi. X p \in \text{tangent-space } p$
shows *smooth-vector-field-local* $\text{charts } \psi X \longleftrightarrow (\forall i \in \text{Basis. diff-fun-on } (\text{domain } \psi) (\text{vector-field-component } X i))$
 (is ⟨?smooth-vf $X \longleftrightarrow (\forall i \in \text{Basis. ?diff-component } X i) \rangle$)
 ⟨proof⟩

end

lemma *smooth-vector-field-imp-local'*:
fixes $X \psi X_\psi$ **defines** $X_\psi \equiv \text{restrict0 } (\text{domain } \psi) X$
assumes *smooth-vector-field* $X \psi \in \text{atlas}$
shows *smooth-vector-field-local charts* ψX_ψ
 $\langle \text{proof} \rangle$

lemma *smooth-vector-field-if-local*:
assumes $\forall p \in \text{carrier}. \exists c \in \text{atlas}. p \in \text{domain } c \wedge \text{smooth-vector-field-local charts } c X \text{ extensional0 carrier } X$
shows *smooth-vector-field* X
 $\langle \text{proof} \rangle$

lemma *smooth-vector-field-iff-local*:
assumes *extensional0 carrier* X
shows $(\forall c \in \text{atlas}. \text{smooth-vector-field-local charts } c X) \longleftrightarrow \text{smooth-vector-field } X$
 $\langle \text{proof} \rangle$

lemma (**in** *smooth-manifold*) *smooth-vector-field-local*:
assumes $c \in \text{atlas} \forall p \in \text{domain } c. X p \in \text{tangent-space } p$
shows *smooth-vector-field-local charts* $c X \longleftrightarrow$
 $\text{smooth-on } (\text{codomain } c) (\text{apply-chart-TM } c \circ (\lambda p. (p, X p)) \circ \text{inv-chart } c)$

$\langle \text{proof} \rangle$

lemma (**in** *c-manifold*) *diff-fun-deriv-chart'*:
fixes $i::'b$
assumes $c:c \in \text{atlas}$ **and** $f:\text{diff-fun-on } (\text{domain } c) f$ **and** $k:k > 0$
shows *diff-fun* $(k-1) (\text{charts-submanifold } (\text{domain } c)) (\lambda x. \text{frechet-derivative } (f \circ \text{inv-chart } c) (\text{at } (c x)) i)$
 $\langle \text{proof} \rangle$

lemma *diff-fun-deriv-chart*:
fixes $i::'b$
assumes $c:c \in \text{atlas}$ **and** $f:\text{diff-fun-on } (\text{domain } c) f$
shows *diff-fun* $\infty (\text{charts-submanifold } (\text{domain } c)) (\lambda x. \text{frechet-derivative } (f \circ \text{inv-chart } c) (\text{at } (c x)) i)$
 $\langle \text{proof} \rangle$

lemma (**in** *c-manifolds*) *diff-localI2*: *diff k charts1 charts2 f*

if $\forall x \in \text{src.carrier}. (\exists U. \text{diff } k (\text{src.charts-submanifold } U) \text{ charts2 } f \wedge \text{open } U \wedge x \in U)$
 ⟨proof⟩

5.3 Smooth vector fields as maps $C^\infty(M) \rightarrow C^\infty(M)$.

Proposition 8.14 in Lee 2012.

lemma *vector-field-smooth-iff*:

assumes X : *rough-vector-field* X

shows *smooth-vector-field* $X \longleftrightarrow (\forall f \in \text{diff-fun-space}. (X \text{ " } f) \in \text{diff-fun-space})$

(**is** ⟨?LHS \longleftrightarrow ?RHS1⟩)

and *smooth-vector-field* $X \longleftrightarrow (\forall U f. \text{open } U \wedge U \subseteq \text{carrier} \wedge f \in (\text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) \infty) \longrightarrow$

$\text{diff-fun } \infty (\text{charts-submanifold } U)$

$(\text{vec-field-apply-fun-in-at } X f U))$

(**is** ⟨?LHS \longleftrightarrow ?RHS2⟩)

⟨proof⟩

lemma *vector-field-smooth-iff'*:

fixes $C\text{-inf}$

defines $\bigwedge U. C\text{-inf } U \equiv \text{c-manifold.diff-fun-space } (\text{charts-submanifold } U) \infty$

assumes X : *rough-vector-field* X

shows *smooth-vector-field* $X \longleftrightarrow (\forall f \in \text{diff-fun-space}. (X \text{ " } f) \in \text{diff-fun-space})$

and *smooth-vector-field* $X \longleftrightarrow (\forall U f. \text{open } U \wedge U \subseteq \text{carrier} \wedge f \in C\text{-inf } U \longrightarrow$

$\text{diff-fun-on } U (X \upharpoonright U \text{ " } f))$

⟨proof⟩

lemma *smooth-vf-diff-fun-space*:

assumes X : *smooth-vector-field* X

and f : $f \in \text{diff-fun-space}$

shows $X \text{ " } f \in \text{diff-fun-space}$

⟨proof⟩

end

5.4 Smooth vector fields are derivations

context $c\text{-manifold}$ **begin**

— Generalising *is-derivation* (which might have been called *is-derivation-at*) over the carrier set. Relative to that definition, we also add a condition on the codomain.

definition *is-derivation-on* :: $((a \Rightarrow \text{real}) \Rightarrow (a \Rightarrow \text{real})) \Rightarrow \text{bool}$ **where**

is-derivation-on $D \equiv \text{real-linear-on diff-fun-space diff-fun-space } D \wedge$

$(\forall f \in \text{diff-fun-space}. \forall g \in \text{diff-fun-space}. D (f * g) = f * (D g) +$

$g * (D f)) \wedge$

$D \text{ ' } \text{diff-fun-space} \subseteq \text{diff-fun-space}$

lemma *vec-field-linear-on*:
assumes X : *rough-vector-field* X
and b : $b1 \in \text{diff-fun-space}$ $b2 \in \text{diff-fun-space}$
shows $X \ " (b1+b2) = (X \ " b1 + X \ " b2)$ $X \ " (r *_{\mathbb{R}} b1) = (r *_{\mathbb{R}} (X \ " b1))$
 $\langle \text{proof} \rangle$

lemma *linear-on-vec-field*:
assumes *rough-vector-field* X
shows *real-linear-on* *diff-fun-space* *diff-fun-space* $((\ ") X)$
 $\langle \text{proof} \rangle$

lemma *product-rule- vf* :
assumes X : *rough-vector-field* X
and $f \in \text{diff-fun-space}$ $g \in \text{diff-fun-space}$
shows $X \ " (f * g) = f * (X \ " g) + g * (X \ " f)$
 $\langle \text{proof} \rangle$

end

context *smooth-manifold* **begin**

lemma *vector-field-is-derivation*:
assumes X : *smooth-vector-field* X
shows *is-derivation-on* $(\lambda f. X \ " f)$
 $\langle \text{proof} \rangle$

5.5 Derivations are smooth vector fields

lemma *extensional-derivation-is-smooth-vector-field*:
fixes $D :: ('a \Rightarrow \text{real}) \Rightarrow ('a \Rightarrow \text{real})$ **and** $X :: 'a \Rightarrow ('a \Rightarrow \text{real}) \Rightarrow \text{real}$
defines $[\text{simp}]$: $X \equiv \lambda p. \lambda f. D \ f \ p$
assumes *der-D*: *is-derivation-on* D
and *ext-X*: *extensional0* *carrier* X
and *ext-D*: *extensional0* *diff-fun-space* D
shows *smooth-vector-field* X
 $\langle \text{proof} \rangle$

lemma *extensional-derivation-is-smooth-vector-field'*:
fixes $D :: ('a \Rightarrow \text{real}) \Rightarrow ('a \Rightarrow \text{real})$
assumes *der-D*: *is-derivation-on* D
and *ext-X*: *extensional0* *carrier* $(\lambda p \ f. D \ f \ p)$
and *ext-D*: *extensional0* *diff-fun-space* D
obtains X **where** *smooth-vector-field* X **and** $\forall f \in \text{diff-fun-space}. D \ f = X \ " f$
 $\langle \text{proof} \rangle$

theorem *smooth-vector-field-iff-derivation*:

fixes *extensional-derivation* **defines** $\wedge D. \text{extensional-derivation } D \equiv$
*is-derivation-on } D \wedge \text{extensional0 carrier } (\lambda p f. D f p) \wedge \text{extensional0 diff-fun-space } D
shows *smooth-vector-field } X \implies extensional-derivation } ($\lambda f. X " f$)*
and *extensional-derivation } D \implies smooth-vector-field } ($\lambda p f. D f p$)*
 <proof>
end
end*

6 The Lie bracket of smooth vector fields

theory *Manifold-Lie-Bracket*

imports

Smooth-Vector-Fields

Algebra-On

begin

definition *lie-bracket-of-smooth-vector-fields* :: 'a vector-field \Rightarrow 'a vector-field \Rightarrow 'a vector-field

where *lie-bracket-of-smooth-vector-fields } X Y \equiv $\lambda p::'a. \lambda f::'a \Rightarrow \text{real. } X p (Y " f) - Y p (X " f)$*

notation *lie-bracket-of-smooth-vector-fields* (<[-;-]> [65,65])

lemma *lie-bracket-def*: $[X; Y] p f = X p (Y " f) - Y p (X " f)$

<proof>

context *c-manifold* **begin**

6.1 General lemmas

lemma *is-derivation-uminus*: *is-derivation } (-x) p* **if** *x: is-derivation } x p*

<proof>

lemma *is-derivation-minus*: *is-derivation } (x - y) p*

if *x: is-derivation } x p* **and** *y: is-derivation } y p*

<proof>

lemma *diff-fun-space-minus*: *f - g \in diff-fun-space*

if *f \in diff-fun-space* *g \in diff-fun-space*

<proof>

lemma *rough-vector-field-add*:

assumes *rough-vector-field } X* *rough-vector-field } Y*

shows *rough-vector-field } (X + Y)*

<proof>

abbreviation (*input*) $\text{scaleR-vf} \equiv \text{scaleR} :: \text{real} \Rightarrow 'a \text{ vector-field} \Rightarrow 'a \text{ vector-field}$

lemma scaleR-vf : $\text{scaleR-vf} = (\lambda r X p f. r * X p f)$ $\langle \text{proof} \rangle$

lemma $\text{rough-vector-field-scaleR}$:

assumes $\text{rough-vector-field } X$

shows $\text{rough-vector-field} (\text{scaleR-vf } a X)$

$\langle \text{proof} \rangle$

6.2 Properties of the Lie bracket on \mathfrak{X}

lemma $\text{lie-bracket-antisym}$: $[X; Y] = -[Y; X]$

$\langle \text{proof} \rangle$

lemma ext0-lie-bracket :

shows $\text{extensional0 carrier } X \Longrightarrow \text{extensional0 carrier } Y \Longrightarrow \text{extensional0 carrier } [X; Y]$

and $\text{rough-vector-field } X \Longrightarrow \text{rough-vector-field } Y \Longrightarrow \text{extensional0 diff-fun-space} (\text{vec-field-apply-fun } [X; Y])$

$\langle \text{proof} \rangle$

end

context smooth-manifold **begin**

A nice computational proof that I try to keep close-ish to Lee's original pen-and-paper [?, p. 186].

lemma $\text{product-rule-lie-bracket}$:

assumes X : $\text{smooth-vector-field } X$

and Y : $\text{smooth-vector-field } Y$

and diff-funs : $f \in \text{diff-fun-space } g \in \text{diff-fun-space}$

shows $[X; Y] \text{ '' } (f * g) = f * [X; Y] \text{ '' } g + g * [X; Y] \text{ '' } f$

$\langle \text{proof} \rangle$

lemma $\text{lie-bracket-is-derivation-on}$:

assumes X : $\text{smooth-vector-field } X$

and Y : $\text{smooth-vector-field } Y$

shows $\text{is-derivation-on} (\lambda f. [X; Y] \text{ '' } f)$

$\langle \text{proof} \rangle$

This is Lee's [?, Lemma 8.25].

lemma $\text{lie-bracket-closed}$:

assumes X : $\text{smooth-vector-field } X$

and Y : $\text{smooth-vector-field } Y$

shows $\text{smooth-vector-field } [X; Y]$

$\langle \text{proof} \rangle$

lemma

assumes X : *smooth-vector-field* X

and Y : *smooth-vector-field* Y

and Z : *smooth-vector-field* Z

shows *lie-bracket-add-left*: $[X+Y;Z] = [X;Z] + [Y;Z]$

and *lie-bracket-add-right*: $[X;Y+Z] = ([X;Y] + [X;Z])$

<proof>

lemma

assumes X : *smooth-vector-field* X

and Y : *smooth-vector-field* Y

shows *lie-bracket-scale-left*: $[scaleR-vf\ a\ X; Y] = scaleR-vf\ a\ [X; Y]$

and *lie-bracket-scale-right*: $[X; scaleR-vf\ a\ Y] = scaleR-vf\ a\ [X; Y]$

<proof>

lemmas *lie-bracket-bilinear-simps* $[simp] = lie-bracket-scale-left$

lie-bracket-scale-right

lie-bracket-add-left

lie-bracket-add-right

lemma (*in module-hom-on*) *diff*:

$b1 \in S1 \implies b2 \in S1 \implies f\ (b1 - b2) = f\ b1 - f\ b2$

<proof>

lemma *lie-bracket-jacobi*: $[X; [Y;Z]] + [Y;[Z;X]] + [Z;[X;Y]] = 0$

if X : *smooth-vector-field* X

and Y : *smooth-vector-field* Y

and Z : *smooth-vector-field* Z

<proof>

definition $SVF \equiv \{X. \text{smooth-vector-field } X\}$

lemma *lie-algebra-of-smooth-vector-fields*: *lie-algebra* SVF *scaleR-vf* *lie-bracket-of-smooth-vector-fields*

<proof>

end

end

theory *Lie-Group*

```

imports
  HOL-Analysis.Analysis
  HOL-Eisbach.Eisbach
  More-Manifolds
begin

```

7 Definition of Lie Groups (as Locales)

Some abbreviations for easier reading first. A binary operation is colloquially said continuous/smooth/differentiable on a manifold M if it is so on the product manifold M^2 . We fix the types of the binary operations in two of the definitions below, as the target space is made explicit only in the third (the one using $\text{diff } \infty$).

```

abbreviation (input) continuous-on-product-manifold charts (binop::'a⇒'a⇒'a::{second-countable-topology,t2}
≡
  continuous-on (c-manifold-prod.carrier charts charts) (λ(a,b). binop a b)
abbreviation (input) smooth-on-product-manifold charts (binop::'a⇒'a⇒'a::{second-countable-topology,real-n}
≡
  smooth-on (c-manifold-prod.carrier charts charts) (λ(a,b). binop a b)
abbreviation (input) diff-on-product-manifold charts binop ≡
  diff ∞ (c-manifold-prod.prod-charts charts charts) (λ(a,b). binop a b)

```

7.1 Topological groups

A group with a topology, such that the group operations are continuous.

```

locale topological-group =
  manifold charts + group-on-with carrier tms tms-one dvsn invs
  for charts::('a::{t2-space,second-countable-topology}, 'e::euclidean-space) chart set
  and tms tms-one dvsn invs +
  assumes cts-mult: continuous-on-product-manifold charts tms
  and cts-inv: continuous-on carrier invs

```

7.2 Lie groups

A Lie group is a group on a set, but instead of a carrier set, we specify a set of charts, which imply the carrier set as a (smooth) manifold M . Internally, we consider the product manifold, to define smoothness of multiplication $M \times M \rightarrow M$. It may be overkill to keep inverse and division separate, considering *group-on-with* includes an axiom to relate the two, but this is how it's done in other Isabelle theories, so I'll keep it. It gives some extra flexibility, and an intro lemma using the more traditional group parameters (an operation, and an identity) and axioms is already provided in $\llbracket \forall a \in ?G. \forall b \in ?G. ?mult\ a\ b \in ?G; \forall a \in ?G. \forall b \in ?G. \forall c \in ?G. ?mult\ (?mult\ a\ b)\ c = ?mult\ a\ (?mult\ b\ c); ?e \in ?G \wedge (\forall a \in ?G. ?mult\ ?e\ a = a \wedge ?mult\ a\ ?e = a); \forall x \in ?G. \exists y. y \in ?G \wedge ?mult\ x\ y = ?e \wedge ?mult\ y\ x = ?e \rrbracket \implies \text{group-on-with } ?G\ ?mult\ ?e\ (\lambda x\ z. ?mult\ x\ (THE\ y. y \in ?G \wedge ?mult\ z\ y =$

$?e \wedge ?mult\ y\ z = ?e$) ($\lambda x. THE\ y. y \in ?G \wedge ?mult\ x\ y = ?e \wedge ?mult\ y\ x = ?e$).

locale *lie-group* =

c-manifold charts ∞ + *group-on-with carrier tms tms-one dvsn invs*
for *charts::('a::{t2-space,second-countable-topology}, 'e::euclidean-space) chart set*
and *tms tms-one dvsn invs* +
assumes *smooth-mult: diff-on-product-manifold charts tms*
and *smooth-inv: diff ∞ charts charts invs*

We can make a shortened locale for Lie groups where the inversion and division are implied. This does *not* say anything about the implementation of inversion or division outside the carrier set. See also *grp-on*.

locale *lie-grp* =

c-manifold charts ∞ + *grp-on carrier tms one*
for *charts::('a::{t2-space,second-countable-topology}, 'e::euclidean-space) chart set*
and *tms one* +
— multiplication and inversion are smooth
assumes *smooth-mult: diff-on-product-manifold charts tms*
and *smooth-inv: diff ∞ charts charts invs*

begin

lemma *is-lie-group: lie-group charts tms one mns invs*

<proof>

sublocale *lie-group charts tms one mns invs*

<proof>

end

lemma *lie-group-imp-lie-grp:*

assumes *lie-group charts pls one any-mns any-invs*

shows *lie-grp charts pls one*

<proof>

We give a few intro rules for the *lie-group* predicate, as well as an Eisbach method for further breaking down the proof of smoothness of the multiplication and inversion maps. This should lead to fairly organised proofs that some structure is a *lie-group*. In general, I would prefer *group-manifold-imp-lie-group2* to *group-manifold-imp-lie-group*.

lemma *group-manifold-imp-lie-group [intro]:*

assumes *is-manifold: c-manifold c ∞*

and *is-group: group-on-with (\bigcup (domain ' c)) tms tms-1 dvsn invs*

and *smooth-mult: diff ∞ (c-manifold-prod.prod-charts c c) c ($\lambda(a,b). tms\ a\ b$)*

and *smooth-inv: diff ∞ c c invs*

shows *lie-group c tms tms-1 dvsn invs*

<proof>

lemma *group-manifold-imp-lie-group2 [intro]:*

```

assumes is-manifold: c-manifold c  $\infty$ 
and is-group: group-on-with ( $\bigcup$  (domain ‘ c)) tms tms-1 dvsn invs
and smooth-mult: diff-axioms  $\infty$  (c-manifold-prod.prod-charts c c) c ( $\lambda(a,b)$ .
tms a b)
and smooth-inv: diff-axioms  $\infty$  c c invs
shows lie-group c tms tms-1 dvsn invs
⟨proof⟩

```

```

lemma lie-grpI [intro]:
fixes tms tms-1 c
defines invs  $\equiv$  grp-on.invs ( $\bigcup$  (domain ‘ c)) tms tms-1
assumes is-manifold: c-manifold c  $\infty$ 
and is-group: grp-on ( $\bigcup$  (domain ‘ c)) tms tms-1
and smooth-mult: diff-axioms  $\infty$  (c-manifold-prod.prod-charts c c) c ( $\lambda(a,b)$ .
tms a b)
and smooth-inv: diff-axioms  $\infty$  c c invs
shows lie-grp c tms tms-1
⟨proof⟩

```

A small method to unfold the axioms of differentiability of group operations. Allows for succinct goals to be stated while quickly unfolding to a useful level of technicality.

```

method unfold-diff-axioms = (
  unfold diff-axioms-def,
  rule allI,
  rule impI,
  (rule becI)+,
  (rule conjI),
  rule-tac[2] conjI
)

```

7.3 Some lemmas about Lie groups (and other needed results).

```

context lie-group begin

```

```

lemma obtain-chart-cover:
assumes  $S \subseteq$  carrier
obtains C where  $\forall c \in C. c \in$  atlas  $\forall s \in S. \exists c \in C. s \in$  domain c
⟨proof⟩

```

```

lemma open-covered-by-charts:
assumes  $S \subseteq$  carrier open S
obtains C where  $\forall c \in C. c \in$  atlas  $S = \bigcup \{$  domain c  $\mid c. c \in C\}$ 
⟨proof⟩

```

```

lemma lie-prod: c-manifold-prod  $\infty$  charts charts
⟨proof⟩

```

interpretation *lie-prod: c-manifold-prod* ∞ *charts charts*
 ⟨*proof*⟩

lemma *continuous-on-tms:*
assumes $x \in \text{carrier}$
shows *continuous-on carrier* $(\lambda y. \text{tms } x \ y)$
and *continuous-on carrier* $(\lambda y. \text{tms } y \ x)$
 ⟨*proof*⟩

lemma *diff-tms:*
assumes $x \in \text{carrier}$
shows *diff* ∞ *charts charts* $(\lambda y. \text{tms } x \ y)$
and *diff* ∞ *charts charts* $(\lambda y. \text{tms } y \ x)$
 ⟨*proof*⟩

lemma *diff-tms-invs:*
assumes $x \in \text{carrier}$
shows *diff* ∞ *charts charts* $(\lambda y. \text{tms } (\text{invs } x) \ y)$
and *diff* ∞ *charts charts* $(\lambda y. \text{tms } y \ (\text{invs } x))$
 ⟨*proof*⟩

lemma *diff-tms-invs':*
assumes $x \in \text{carrier}$
shows *diff* ∞ *charts charts* $(\lambda y. \text{tms } x \ (\text{invs } y))$
and *diff* ∞ *charts charts* $(\lambda y. \text{tms } (\text{invs } y) \ x)$
 ⟨*proof*⟩

end

8 Morphisms of Lie groups, actions and representations

8.1 Morphism of Lie groups.

locale *lie-group-pair* =
L1: lie-group $c1 \ t1 \ i1 \ d1 \ m1$ +
L2: lie-group $c2 \ t2 \ i2 \ d2 \ m2$
for $c1 :: ('a::\{\text{second-countable-topology}, t2\text{-space}\}, 'b::\text{euclidean-space})$ *chart set*
and $c2 :: ('c::\{\text{second-countable-topology}, t2\text{-space}\}, 'd::\text{euclidean-space})$ *chart set*
and $t1 \ t2$ **and** $i1 \ i2$ **and** $d1 \ d2$ **and** $m1 \ m2$

locale *lie-group-morphism-with* =
lie-group-pair $c1 \ c2 \ t1 \ t2 \ i1 \ i2 \ d1 \ d2 \ m1 \ m2$ +
diff ∞ $c1 \ c2 \ f$ +
group-hom-betw $L1.\text{carrier} \ L2.\text{carrier} \ t1 \ t2 \ i1 \ i2 \ d1 \ d2 \ m1 \ m2 \ f$
for $c1 :: ('a::\{\text{second-countable-topology}, t2\text{-space}\}, 'b::\text{euclidean-space})$ *chart set*
and $c2 :: ('c::\{\text{second-countable-topology}, t2\text{-space}\}, 'd::\text{euclidean-space})$ *chart set*

set

and $t1\ t2$ and $i1\ i2$ and $d1\ d2$ and $m1\ m2$ and f

lemma (in *lie-group-pair*) *lie-group-morphismI*:

assumes $diff \infty c1\ c2\ f$

and *group-hom*: $\forall x \in L1.carrier. \forall y \in L1.carrier. f\ (t1\ x\ y) = t2\ (f\ x)\ (f\ y)$

and *closure*: $\forall x \in L1.carrier. f\ x \in L2.carrier$

shows *lie-group-morphism-with* $c1\ c2\ t1\ t2\ i1\ i2\ d1\ d2\ m1\ m2\ f$

<proof>

lemma (in *lie-group*) *lie-group-morphismI*:

assumes *lie-group* $c2\ t2\ i2\ d2\ m2$

and $diff \infty charts\ c2\ f$

and *group-hom*: $\forall x \in carrier. \forall y \in carrier. f\ (tms\ x\ y) = t2\ (f\ x)\ (f\ y)$

and *closure*: $\forall x \in carrier. f\ x \in (manifold.carrier\ c2)$

shows *lie-group-morphism-with charts* $c2\ tms\ t2\ tms-one\ i2\ d2\ m2\ f$

<proof>

locale *lie-group-isomorphism* =

lie-group-pair $c1\ c2\ t1\ t2\ i1\ i2\ d1\ d2\ m1\ m2$ +

diffeomorphism $\infty c1\ c2\ f\ f'$ +

group-hom-betw $L1.carrier\ L2.carrier\ t1\ t2\ i1\ i2\ d1\ d2\ m1\ m2\ f$

for $c1 :: ('a::\{second-countable-topology,t2-space\}, 'b::euclidean-space)\ chart\ set$

and $c2 :: ('c::\{second-countable-topology,t2-space\}, 'd::euclidean-space)\ chart$

set

and $t1\ t2$ and $i1\ i2$ and $d1\ d2$ and $m1\ m2$ and $f\ f'$

8.2 Action of a Lie group on a manifold.

abbreviation (*input*) *diff-action-map g-charts m-charts action* \equiv

$diff \infty (c-manifold-prod.prod-charts\ g-charts\ m-charts)\ m-charts\ action$

A Lie group action is a homomorphism from the Lie group to the automorphism group of a space, here a manifold, which is differentiable (smooth). I take here the more explicit definition given in Kirillov's lecture notes (2008; page 12), and derive the more abstract version later (after showing *c-manifold.Diff* is not just a group, but a Lie group).

Take care: there are now two manifolds, of which the Lie group is the primary one as far as namespace is concerned. Everything pertaining to the manifold acted upon is accessed with qualified syntax. This disappears for Lie groups acting on themselves.

locale *lie-group-action* =

lie-group charts tms tms-one d2sn invs + $M: c-manifold\ m-charts\ k$

for $charts::('a::\{t2-space,second-countable-topology\}, 'e::euclidean-space)\ chart\ set$

and $tms\ tms-one\ d2sn\ invs$

and $m-charts::('b::\{t2-space,second-countable-topology\}, 'f::euclidean-space)\ chart$

set and k +

fixes *action* ($\langle \rho \rangle$)

assumes *act-diff*: $g \in \text{carrier} \implies (\varrho g) \in M.\text{Diff}$
and *act-one*: $\varrho \text{ tms-one} = M.\text{Diff-id}$
and *act-hom*: $f \in G \implies g \in G \implies \varrho (\text{tms } f g) = M.\text{Diff-comp } (\varrho f) (\varrho g)$
and *act-diff-prod*: *diff-action-map charts m-charts* $(\lambda(g,m). \text{the } ((\varrho g) m))$

After proving Diff is a group, some of these axioms can be replaced.

locale *lie-group-action'* =
lie-group charts tms tms-one dvn invs +
M: c-manifold m-charts k +
A: group-hom-betw carrier M.Diff tms M.Diff-comp tms-one M.Diff-id dvn M.Diff-comp-inv
invs M.Diff-inv ϱ
for *charts*::('a::{t2-space,second-countable-topology}, 'e::euclidean-space) *chart set*
and *tms tms-one dvn invs*
and *m-charts*::('b::{t2-space,second-countable-topology}, 'f::euclidean-space) *chart set* **and** *k*
and $\varrho :: 'a \Rightarrow ('b \multimap 'b) +$
assumes *diff-action-map: diff-action-map charts m-charts* $(\lambda(g,m). \text{the } ((\varrho g) m))$

8.3 Action of a Lie Group on itself.

context *lie-group begin*

abbreviation (*input*) *left-self-action* :: $'a \Rightarrow 'a \Rightarrow 'a$ ($\langle \mathcal{L} \rightarrow [91] \rangle$)
where *left-self-action* $g g' \equiv \text{tms } g g'$

abbreviation *left-action* :: $'a \Rightarrow ('a \multimap 'a)$
where *left-action* $g \equiv (\lambda x. \text{if } x \in \text{carrier} \text{ then } \text{Some } (\text{left-self-action } g x) \text{ else } \text{None})$

abbreviation (*input*) *right-self-action* :: $'a \Rightarrow 'a \Rightarrow 'a$ ($\langle \mathcal{R} \rightarrow [91] \rangle$)
where *right-self-action* $g g' \equiv \text{tms } g' (invs g)$

abbreviation *right-action* :: $'a \Rightarrow ('a \multimap 'a)$
where *right-action* $g \equiv (\lambda x. \text{if } x \in \text{carrier} \text{ then } \text{Some } (\text{right-self-action } g x) \text{ else } \text{None})$

abbreviation (*input*) *adjoint-self-action* :: $'a \Rightarrow 'a \Rightarrow 'a$
where *adjoint-self-action* $g g' \equiv \text{tms } g (\text{tms } g' (invs g))$

8.3.1 The left action.

lemma *L-action-in*: $(\text{left-self-action } g g') \in \text{carrier}$ **if** $g \in \text{carrier } g' \in \text{carrier}$
<proof>

lemma *the-left-action*: $\text{left-self-action } x y = \text{the } (\text{left-action } x y)$ **if** $y \in \text{carrier}$
<proof>

lemma *L-action-invs*: $(\text{left-self-action } (invs x) \circ \text{left-self-action } x) y = y$
 $(\text{left-self-action } x \circ \text{left-self-action } (invs x)) y = y$
if $x \in \text{carrier } y \in \text{carrier}$

<proof>

lemma *L-homeomorphism: homeomorphism carrier carrier* $(\mathcal{L} \ x) (\mathcal{L} \ (invs \ x))$ **if**
 $x \in carrier$
<proof>

lemma *L-homeomorphism': homeomorphism carrier carrier* $(\mathcal{L} \ (invs \ x)) (\mathcal{L} \ x)$
if $x \in carrier$
<proof>

lemma *L-homeomorphism-chart: homeomorphism (domain c)* $(\mathcal{L} \ x \ ' \ domain \ c) (\mathcal{L} \ x)$
 $(\mathcal{L} \ (invs \ x))$
if $x \in carrier \ c \in atlas$
<proof>

lemma *L-homeomorphism-chart': homeomorphism* $(\mathcal{L} \ x \ ' \ domain \ c) (domain \ c)$
 $(\mathcal{L} \ (invs \ x)) (\mathcal{L} \ x)$
if $x \in carrier \ c \in atlas$
<proof>

lemma *L-open-map:*
assumes $x \in carrier \ open \ S \ S \subseteq carrier$
shows $open \ (\mathcal{L} \ x \ ' \ S)$
<proof>

lift-definition *L-chart :: 'a \Rightarrow ('a,'e) chart \Rightarrow ('a,'e) chart*
is $\lambda x. \lambda(d,d',f,f'). \text{ if } x \in carrier \wedge d \subseteq carrier \text{ then } (\mathcal{L} \ x \ ' \ d, d', f \circ \mathcal{L} \ (invs \ x), \mathcal{L} \ x \circ f') \text{ else } (\{\}, \{\}, f, f')$
<proof>

lemma *L-chart-apply-chart[simp]: apply-chart (L-chart x c) = apply-chart c \circ \mathcal{L}*
 $(invs \ x)$
and *L-chart-inv-chart[simp]: inv-chart (L-chart x c) = $\mathcal{L} \ x \circ inv-chart \ c$*
and *domain-L-chart[simp]: domain (L-chart x c) = $\mathcal{L} \ x \ ' \ domain \ c$*
and *codomain-L-chart[simp]: codomain (L-chart x c) = codomain c*
if $x \in carrier \ c \in atlas$
<proof>

lemma *L-chart-apply-chart'[simp]: apply-chart (L-chart x c) = apply-chart c \circ \mathcal{L}*
 $(invs \ x)$
and *L-chart-inv-chart'[simp]: inv-chart (L-chart x c) = $\mathcal{L} \ x \circ inv-chart \ c$*
and *domain-L-chart'[simp]: domain (L-chart x c) = $\mathcal{L} \ x \ ' \ domain \ c$*
and *codomain-L-chart'[simp]: codomain (L-chart x c) = codomain c*
if $x \in carrier \ domain \ c \subseteq carrier$
<proof>

lemma *smooth-compat-L-chart:*
assumes $x \in carrier \ c \in atlas \ c' \in atlas$
shows $\infty\text{-smooth-compat} \ (L\text{-chart} \ x \ c) \ c'$

<proof>

lemma *L-chart-compat:*

assumes $x \in \text{carrier } c \in \text{atlas}$

shows $\infty\text{-smooth-compat } c \text{ (L-chart } x \text{ c)}$

<proof>

lemma *L-chart-in-atlas: L-chart $x \text{ c} \in \text{atlas}$ if $x \in \text{carrier } c \in \text{atlas}$*

<proof>

lemma *left-action-automorphic: c-automorphism $\infty \text{ charts } (\mathcal{L} \ x) \ (\mathcal{L} \ (\text{invs } x))$*

if $x \in \text{carrier}$

<proof>

lemma *left-action-in-Diff: left-action $x \in \text{Diff}$ if $x \in \text{carrier}$*

<proof>

lemma *diff-the-L: diff $\infty \text{ (c-manifold-prod.prod-charts charts charts) charts } (\lambda(g, m). \text{ the (left-action } g \ m))$*

(is diff $\infty \text{ ?prod-charts charts ?L}$)

<proof>

lemma *left-action: lie-group-action' charts tms tms-one dvsn invs charts $\infty \text{ left-action}$*

<proof>

sublocale *left-action: lie-group-action' charts tms tms-one dvsn invs charts $\infty \text{ left-action}$*

<proof>

8.3.2 The right action.

lemma *R-action-in: (right-self-action $g \ g') \in \text{carrier}$ if $g \in \text{carrier } g' \in \text{carrier}$*

<proof>

lemma *the-right-action: right-self-action $x \ y = \text{the (right-action } x \ y)$ if $y \in \text{carrier}$*

<proof>

lemma *R-action-invs: (right-self-action (invs x) \circ right-self-action x) $y = y$*

(right-self-action $x \circ$ right-self-action (invs x)) $y = y$

if $x \in \text{carrier } y \in \text{carrier}$

<proof>

lemma *R-homeomorphism: homeomorphism carrier carrier $(\mathcal{R} \ x) \ (\mathcal{R} \ (\text{invs } x))$*

if $x \in \text{carrier}$

<proof>

lemma *R-homeomorphism': homeomorphism carrier carrier $(\mathcal{R} \ (\text{invs } x)) \ (\mathcal{R} \ x)$*

if $x \in \text{carrier}$

<proof>

lemma *R-homeomorphism-chart*: *homeomorphism* (domain c) ($\mathcal{R} x \text{ ' domain c}$)
($\mathcal{R} x$) ($\mathcal{R} (\text{invs } x)$)
if $x \in \text{carrier } c \in \text{atlas}$
 $\langle \text{proof} \rangle$

lemma *R-homeomorphism-chart'*: *homeomorphism* ($\mathcal{R} x \text{ ' domain c}$) (domain c)
($\mathcal{R} (\text{invs } x)$) ($\mathcal{R} x$)
if $x \in \text{carrier } c \in \text{atlas}$
 $\langle \text{proof} \rangle$

lemma *R-open-map*:
assumes $x \in \text{carrier open } S \ S \subseteq \text{carrier}$
shows *open* ($\mathcal{R} x \text{ ' } S$)
 $\langle \text{proof} \rangle$

lift-definition *R-chart* :: $'a \Rightarrow ('a, 'e) \text{ chart} \Rightarrow ('a, 'e) \text{ chart}$
is $\lambda x. \lambda (d, d', f, f'). \text{ if } x \in \text{carrier} \wedge d \subseteq \text{carrier} \text{ then } (\mathcal{R} x \text{ ' } d, d', f \circ \mathcal{R} (\text{invs } x), \mathcal{R} x \circ f') \text{ else } (\{\}, \{\}, f, f')$
 $\langle \text{proof} \rangle$

lemma *R-chart-apply-chart[simp]*: *apply-chart* (*R-chart* x c) = *apply-chart* c $\circ \mathcal{R}$
(*invs* x)
and *R-chart-inv-chart[simp]*: *inv-chart* (*R-chart* x c) = $\mathcal{R} x \circ \text{inv-chart } c$
and *domain-R-chart[simp]*: *domain* (*R-chart* x c) = $\mathcal{R} x \text{ ' domain } c$
and *codomain-R-chart[simp]*: *codomain* (*R-chart* x c) = *codomain* c
if $x \in \text{carrier } c \in \text{atlas}$
 $\langle \text{proof} \rangle$

lemma *R-chart-apply-chart'[simp]*: *apply-chart* (*R-chart* x c) = *apply-chart* c $\circ \mathcal{R}$
(*invs* x)
and *R-chart-inv-chart'[simp]*: *inv-chart* (*R-chart* x c) = $\mathcal{R} x \circ \text{inv-chart } c$
and *domain-R-chart'[simp]*: *domain* (*R-chart* x c) = $\mathcal{R} x \text{ ' domain } c$
and *codomain-R-chart'[simp]*: *codomain* (*R-chart* x c) = *codomain* c
if $x \in \text{carrier domain } c \subseteq \text{carrier}$
 $\langle \text{proof} \rangle$

lemma *smooth-compat-R-chart*:
assumes $x \in \text{carrier } c \in \text{atlas } c' \in \text{atlas}$
shows $\infty\text{-smooth-compat}$ (*R-chart* x c) c'
 $\langle \text{proof} \rangle$

lemma *R-chart-compat*:
assumes $x \in \text{carrier } c \in \text{atlas}$
shows $\infty\text{-smooth-compat } c$ (*R-chart* x c)
 $\langle \text{proof} \rangle$

lemma *R-chart-in-atlas*: *R-chart* x c $\in \text{atlas}$ **if** $x \in \text{carrier } c \in \text{atlas}$
 $\langle \text{proof} \rangle$

lemma *right-action-automorphic*: *c-automorphism* ∞ *charts* (\mathcal{R} x) (\mathcal{R} (*invs* x))
if $x \in \text{carrier}$
 $\langle \text{proof} \rangle$

lemma *right-action-in-Diff*: *right-action* $x \in \text{Diff}$ **if** $x \in \text{carrier}$
 $\langle \text{proof} \rangle$

end

9 Models/Instances

9.1 Euclidean Space

Euclidean spaces are dealt with at the start of the section “Differentiable Functions” in *Smooth-Manifolds.Differentiable-Manifold*. Therefore, this section is really just a “trivial” exercise to get used to things.

9.1.1 Euclidean Spaces are Lie groups under (+).

locale *euclidean-lie-group-add*
begin

abbreviation C
where $C \equiv \text{manifold-eucl.carrier}$

abbreviation $C\text{-prod}$
where $C\text{-prod} \equiv \text{manifold.carrier prod-charts-eucl}$

lemma *eucl-is-group*: *group-on-with* C (+) 0 (−) *uminus*
 $\langle \text{proof} \rangle$

lemma *prod-domain-codomain*: *domain* *prod-chart-eucl* = $C \times C$ *C* \times *C* = $C\text{-prod}$
codomain *prod-chart-eucl* = $C \times C$
 $\langle \text{proof} \rangle$

lemma *smooth-on-add-const*: *smooth-on* C ($\lambda a. a+b$)
 $\langle \text{proof} \rangle$

lemma *smooth-binop-diff*:
fixes $\text{tms}::'a \Rightarrow 'a \Rightarrow 'a::\text{euclidean-space}$
assumes *smooth-on* $C\text{-prod}$ ($\lambda(a,b). \text{tms } a \ b$)
shows *diff* ∞ *prod-charts-eucl* *charts-eucl* ($\lambda(x, y). \text{tms } x \ y$)
 $\langle \text{proof} \rangle$

lemma *smooth-unop-diff*:
fixes $\text{invs}::'a \Rightarrow 'a::\text{euclidean-space}$
assumes *smooth-on* C *invs*

shows $\text{diff} \infty \text{charts-eucl charts-eucl invs}$
<proof>

lemma *eucl-smooth-group-imp-lie-group*:
assumes *is-group: group-on-with C tms tms-1 dvsn invs*
and *smooth-mult: smooth-on C-prod ($\lambda(a,b). tms a b$)*
and *smooth-inv: smooth-on C invs*
shows *lie-group charts-eucl tms tms-1 dvsn invs*
<proof>

Any Euclidean space is a Lie group under addition.

theorem *lie-group-eucl: lie-group charts-eucl (+) 0 (-) uminus*
<proof>

interpretation *lie-group-eucl: lie-group charts-eucl (+) 0 (-) uminus*
<proof>

end

9.2 The real numbers as a Lie group

lift-definition *chart-real::(real, real) chart is*
(UNIV, UNIV, $\lambda x. x, \lambda x. x$)
<proof>

abbreviation *charts-real \equiv {chart-real}*

lemma *chart-real-is-eucl: charts-eucl = charts-real chart-eucl = chart-real*
<proof>

theorem *lie-group-real: lie-group charts-real (+) 0 (-) uminus*
<proof>

end

10 The Lie algebra of a Lie Group

theory *Lie-Algebra*
imports
Lie-Group
Manifold-Lie-Bracket
Smooth-Manifolds.Cotangent-Space
begin

sublocale *lie-group \subseteq smooth-manifold <proof>*

locale *lie-algebra-morphism =*

```

src: lie-algebra S1 scale1 bracket1 +
dest: lie-algebra S2 scale2 bracket2 +
linear-on S1 S2 scale1 scale2 f
for S1 S2
  and scale1::'a::field  $\Rightarrow$  'b  $\Rightarrow$  'b::ab-group-add and scale2::'a::field  $\Rightarrow$  'c  $\Rightarrow$ 
'c::ab-group-add
  and bracket1 and bracket2
  and f +
  assumes bracket-hom:  $\bigwedge X Y. X \in S1 \implies Y \in S1 \implies f (bracket1 X Y) =$ 
bracket2 (f X) (f Y)

```

Multiple isomorphic Lie algebras can be referred to as “the” Lie algebra \mathfrak{g} of a given Lie group G . One Lie algebra is already guaranteed to exist for any Lie group by virtue of *smooth-manifold ?charts \implies lie-algebra (smooth-manifold.SVF ?charts) (*_R) lie-bracket-of-smooth-vector-fields*. We give an isomorphism between the subalgebra of *left-invariant* (smooth) vector fields and the tangent space at identity, and take the latter to be “the” Lie algebra \mathfrak{g} .

context *lie-group begin*

Some notation, for simplicity: the Lie group (or here, its carrier) is G , and the tangent space at the identity (the Lie algebra) is \mathfrak{g} .

notation *carrier* ($\langle G \rangle$)

definition *tangent-space-at-identity* ($\langle \mathfrak{g} \rangle$)

where *tangent-space-at-identity* = *tangent-space tms-one*

10.1 (Left-)invariant vector fields

A vector field X is invariant under some k -smooth map F if the vector assigned to a point $F(p)$ by X is the same as the vector assigned by (the push-forward under) F to the vector $X(p)$. Essentially, F and X “commute”.

definition (in *c-manifold*) *vector-field-invariant-under* :: 'a vector-field \Rightarrow ('a \Rightarrow 'a) \Rightarrow bool

(**infix** *invariant'-under* 80)

where *X invariant-under F* $\equiv \forall p \in \text{carrier}. \forall f \in \text{diff-fun-space}.$

$X (F p) f = (\text{diff.push-forward } k \text{ charts } \text{charts } F) (X p) f$

— TODO this could be in an instance of *diff* going from a manifold to itself, rather than *diffeomorphism*, i.e. an endomorphism rather than an automorphism.

definition (in *c-automorphism*) *invariant* :: 'a vector-field \Rightarrow bool

where *invariant X* $\equiv \forall p \in \text{carrier}. \forall g \in \text{src.diff-fun-space}. X (f p) g = \text{push-forward } (X p) g$

lemma (in *c-automorphism*) *invariant-simp*: *src.vector-field-invariant-under X f* = *invariant X*

<proof>

lemma (in *c-manifold*) *vector-field-invariant-underD*: $X (F p) f = X p (restrict0 carrier (f \circ F))$

if X *invariant-under F diff k charts charts F p ∈ carrier f ∈ diff-fun-space*
 ⟨proof⟩

lemma (in *c-manifold*) *vector-field-invariant-underI*: X *invariant-under F*

if $\text{diff } k \text{ charts charts } F \wedge p f. p \in \text{carrier} \implies f \in \text{diff-fun-space} \implies X (F p) f = X p (restrict0 carrier (f \circ F))$
 ⟨proof⟩

notation *vector-field-invariant-under* (**infix** ⟨*invariant'-under*⟩ 80)

abbreviation *L-invariant X* $\equiv \forall p \in \text{carrier}. X$ *invariant-under* ($\mathcal{L} p$)

lemma *L-invariantD [dest]*: $X (tms p q) f = X q (restrict0 G (f \circ (\mathcal{L} p)))$

if L -*invariant X p ∈ G q ∈ G f ∈ diff-fun-space*
 ⟨proof⟩

lemma *L-invariantI [intro]*: L -*invariant X*

if $\wedge p q f. p \in \text{carrier} \implies q \in \text{carrier} \implies f \in \text{diff-fun-space} \implies X (tms p q) f = X q (restrict0 carrier (f \circ (\mathcal{L} p)))$
 ⟨proof⟩

lemma *lie-bracket-left-invariant*:

assumes L -*invariant X smooth-vector-field X*

and L -*invariant Y smooth-vector-field Y*

shows L -*invariant [X; Y] smooth-vector-field [X; Y]*

⟨proof⟩

In fact, left-invariant smooth vector fields form a Lie subalgebra.

lemma *subspace-of-left-invariant-svf*:

fixes $\mathfrak{X}_{\mathcal{L}}$ **defines** $\mathfrak{X}_{\mathcal{L}} \equiv \{X \in SVF. L\text{-invariant } X\}$

shows *subspace* $\mathfrak{X}_{\mathcal{L}}$

⟨proof⟩

lemma *lie-algebra-of-left-invariant-svf*:

fixes $\mathfrak{X}_{\mathcal{L}}$ **defines** $\mathfrak{X}_{\mathcal{L}} \equiv \{X. \text{smooth-vector-field } X \wedge L\text{-invariant } X\}$

shows *lie-algebra* $\mathfrak{X}_{\mathcal{L}} (*_R) (\lambda X Y. [X; Y])$

⟨proof⟩

end

end

theory *Classical-Groups*

imports

Lie-Group

Linear-Algebra-More

begin

11 Matrix Groups

11.1 Entry Type

What would be a good type for the entries of our matrices? Ideally, I would be able to talk about matrices over reals \mathbb{R} , the complex numbers \mathbb{C} , and the quaternionic skew-field \mathbb{H} . This is hard: only algebras and inner product spaces over \mathbb{R} are well-supported in Isabelle's Main.

For now, for simplicity, I will work with real matrices only. Alternatively, one could try to characterise the type class containing \mathbb{R} , \mathbb{C} , and \mathbb{H} only. Below is a first attempt to maintain at least some generality. I give some trivial type instantiations, as a basic check.

However, locales are the way to go, in my opinion.

```
class real-normed-eucl = real-normed-field + euclidean-space
```

```
instance real-normed-eucl ⊆ euclidean-space ⟨proof⟩
```

```
instance real-normed-eucl ⊆ real-normed-field ⟨proof⟩
```

```
instance real-normed-eucl ⊆ topological-space ⟨proof⟩
```

```
instance real-normed-eucl ⊆ comm-ring ⟨proof⟩
```

```
instance real-normed-eucl ⊆ comm-ring-1 ⟨proof⟩
```

```
instance real-normed-eucl ⊆ real-algebra-1 ⟨proof⟩
```

```
instance vec :: (real-normed-eucl, finite) topological-space ⟨proof⟩
```

```
instance vec :: (real-normed-eucl, finite) euclidean-space ⟨proof⟩
```

```
instance real :: real-normed-eucl ⟨proof⟩
```

```
instance complex :: real-normed-eucl ⟨proof⟩
```

11.2 Mat(n, F)

The set of all $'n$ -vectors over a *topological-space* is a *topological-space*: this is proved in *Finite-Cartesian-Product*. Similar for vectors over a *euclidean-space*. Therefore, a vector of vectors over a topological space (i.e. a matrix) is also a topological space. We can thus define the identity as a chart; this is not superbly useful, but serves as a template for charts for the multiplicative matrix groups later on.

```
lift-definition chart-mat::('a::real-normed-eucl,'n::finite)square-matrix, ('a,'n)square-matrix) chart
  is (UNIV, UNIV, λm. m, λm. m)
  ⟨proof⟩
```

11.3 $GL(n, F)$

We define polymorphic abbreviations for the carrier set of the general linear group as a matrix group over a commutative ring. This group can be considered as the automorphism group on arbitrary modules of non-commutative rings too, but one loses the isomorphism with matrices, and I'm mostly interested in much more specific general linear groups anyway (namely, over real and complex numbers). Using commutative rings (with 1) also means that determinants play nicely.

abbreviation $in\text{-}GL::('a::comm\text{-}ring\text{-}1, 'n::finite)square\text{-}matrix \Rightarrow bool$
where $in\text{-}GL \equiv invertible$
abbreviation GL **where** $GL \equiv Collect\ in\text{-}GL$

As an example for making the polymorphic GL concrete, we specify the general linear group in four real/complex dimensions.

abbreviation $GL_{R4}::(real,4)square\text{-}matrix\ set$ **where** $GL_{R4} \equiv GL$
abbreviation $GL_{C4}::(complex,4)square\text{-}matrix\ set$ **where** $GL_{C4} \equiv GL$

PROBLEM: the inner product on the LHS is real, not complex, which is why the commented line (involving complex multiplication) cannot work (it only passes type checking because $complex\text{-}of\text{-}real$ is a coercion).

lemma

assumes $x \in GL_{C4}$

shows $((row\ i\ x \cdot row\ i\ x)::real) = (\sum_{j \in UNIV}. (row\ i\ x)\$j \cdot (row\ i\ x)\$j)$
 $\langle proof \rangle$

We now define the chart that makes $GL(n, F)$ a Lie group. Since a chart is a homeomorphism, we first need to show that GL is an open set. Notice this GL is already restricted to have much more powerful entries, since we require topology (continuity) now.

lemma $GL\text{-}preimage\text{-}det: det - ' (UNIV - \{0::'a::real\text{-}normed\text{-}eucl\}) = GL$
 $\langle proof \rangle$

lemma $open\text{-}GL: open (GL::('a::real\text{-}normed\text{-}eucl, 'n::finite)square\text{-}matrix\ set)$
 $\langle proof \rangle$

lift-definition $chart\text{-}GL::(('a::real\text{-}normed\text{-}eucl, 'n::finite)square\text{-}matrix, ('a, 'n)square\text{-}matrix)chart$
is $(GL, GL, \lambda m. m, \lambda m. m)$
 $\langle proof \rangle$

lift-definition $real\text{-}chart\text{-}GL::((real, 'n::finite)square\text{-}matrix, (real, 'n)square\text{-}matrix)chart$
is $(GL, GL, \lambda m. m, \lambda m. m)$
 $\langle proof \rangle$

lemma $transfer\text{-}GL [simp]:$

shows $domain\ chart\text{-}GL = GL$

and $codomain\ chart\text{-}GL = GL$

and $apply\text{-}chart\ chart\text{-}GL = (\lambda x. x)$

and *inv-chart chart-GL* = ($\lambda x. x$)
(*proof*)

abbreviation *charts-GL* **where** *charts-GL* \equiv {*chart-GL*}

abbreviation *real-charts-GL* **where** *real-charts-GL* \equiv {*real-chart-GL*}

interpretation *manifold-GL*: *c-manifold charts-GL k*
(*proof*)

abbreviation *prod-chart-GL* :: ((*'a*::*real-normed-eucl*, *'b*::*finite*)*square-matrix* \times (*'a*, *'b*)*square-matrix*, (*'a*, *'b*)*square-matrix* \times (*'a*, *'b*)*square-matrix*) *chart*
where *prod-chart-GL* \equiv *c-manifold-prod.prod-chart chart-GL chart-GL*

abbreviation *prod-charts-GL* :: ((*'a*::*real-normed-eucl*, *'b*::*finite*)*square-matrix* \times (*'a*, *'b*)*square-matrix*, (*'a*, *'b*)*square-matrix* \times (*'a*, *'b*)*square-matrix*) *chart set*
where *prod-charts-GL* \equiv *c-manifold-prod.prod-charts charts-GL charts-GL*

interpretation *prod-manifold-GL*: *c-manifold-prod k*
charts-GL::((*'a*::*real-normed-eucl*, *'n*::*finite*)*square-matrix*, (*'a*, *'n*)*square-matrix*) *chart set*
charts-GL::((*'a*::*real-normed-eucl*, *'n*::*finite*)*square-matrix*, (*'a*, *'n*)*square-matrix*) *chart set*
(*proof*)

abbreviation *prod-GL-carrier* \equiv *manifold.carrier prod-manifold-GL.prod-charts*

abbreviation *prod-GL-atlas* \equiv *c-manifold.atlas prod-manifold-GL.prod-charts* ∞

lemma *transfer-prod-GL* [*simp*]:
shows *domain prod-chart-GL* = *GL* \times *GL*
and *codomain prod-chart-GL* = *GL* \times *GL*
and *apply-chart prod-chart-GL* = ($\lambda x. x$)
and *inv-chart prod-chart-GL* = ($\lambda x. x$)
(*proof*)

lemma *manifold-GL-carrier* [*simp*]: *manifold-GL.carrier* = *GL*
(*proof*)

lemma *prod-manifold-GL-carrier* [*simp*]: *prod-GL-carrier* = *GL* \times *GL*
(*proof*)

The following lemma basically just does unfolding and type checking. Possibly useful once general results for *charts-GL* need to be specified down to *real-charts-GL*.

lemma *real-GL-is-a-GL*:
shows *real-chart-GL* = *chart-GL*
and *real-charts-GL* = *charts-GL*
and *manifold.carrier* (*c-manifold-prod.prod-charts real-charts-GL real-charts-GL*) = *prod-GL-carrier*

<proof>

lemma *mult-closed-on-GL:*

fixes *f-mult* :: ('a,'b)square-matrix × ('a,'b)square-matrix

⇒ ('a::comm-ring-1, 'b::finite) square-matrix

defines *f-mult*: *f-mult* ≡ (λ(x, y). x ** y)

shows *f-mult* ' (GL × GL) ⊆ GL

<proof>

lemma *GL-group-mult-right-div:*

shows *group-on-with* (domain chart-GL) (**) (mat 1) (λm₁ m₂. m₁ ** matrix-inv m₂) matrix-inv

<proof>

lemma *smooth-on-proj: smooth-on prod-GL-carrier fst smooth-on prod-GL-carrier snd*

<proof>

lemma *mult-smooth-on-real-GL:*

fixes *f-mult* :: (real,'n)square-matrix × (real,'n)square-matrix ⇒ (real,'n::finite)square-matrix

defines *f-mult*: *f-mult* ≡ (λ(x, y). x ** y)

shows *smooth-on* (GL × GL) *f-mult*

<proof>

lemma *mult-smooth-on-GL-expanded:*

assumes *x* ∈ prod-GL-carrier

shows *x* ∈ domain prod-chart-GL

and (λ(x, y). x ** y) ' domain prod-chart-GL ⊆ domain chart-GL

and *smooth-on* (codomain prod-chart-GL) (apply-chart chart-GL ∘ (λ(x, y). x ** y) ∘ inv-chart prod-chart-GL)

<proof>

lemma *mult-smooth-on-real-GL-expanded:*

fixes *f-mult* :: (real,'n)square-matrix × (real,'n)square-matrix ⇒ (real,'n::finite)square-matrix

and *x* :: (real,'n)square-matrix × (real,'n)square-matrix

defines *f-mult*: *f-mult* ≡ (λ(x, y). x ** y)

assumes *x* ∈ prod-GL-carrier

shows *x* ∈ domain prod-chart-GL

and *f-mult* ' domain prod-chart-GL ⊆ domain chart-GL

and *smooth-on* (codomain prod-chart-GL) (apply-chart chart-GL ∘ *f-mult* ∘ inv-chart prod-chart-GL)

<proof>

theorem *real-GL-Lie-group: lie-group real-charts-GL (**) (mat 1) ($\lambda m_1 m_2. m_1$
** (matrix-inv m_2)) matrix-inv*
<proof>

corollary *real-GL-Lie-grp: lie-grp real-charts-GL (**) (mat 1)*
<proof>

end

References

- [1] F. Immler and B. Zhan. Smooth manifolds. *Archive of Formal Proofs*, October 2018. https://isa-afp.org/entries/Smooth_Manifolds.html, Formal proof development.
- [2] J. M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, New York, NY, 2012.