

Smooth Manifolds

Fabian Immler and Bohua Zhan

December 14, 2021

Abstract

We formalize the definition and basic properties of smooth manifolds [1] in Isabelle/HOL. Concepts covered include partition of unity, tangent and cotangent spaces, and the fundamental theorem of path integrals. We also examine some concrete manifolds such as spheres and projective spaces. The formalization makes extensive use of the analysis and linear algebra libraries in Isabelle/HOL, in particular its “types-to-sets” mechanism.

Contents

1	Library Additions	3
1.1	Parametricity rules for topology	3
1.2	Miscellaneous	5
1.3	Closed support	5
1.4	Homeomorphism	5
1.5	Generalizations	6
1.6	Equal topologies	6
1.7	Finer topologies	6
1.8	Support	7
1.9	Final topology (Bourbaki, General Topology I, 4.)	7
1.10	Quotient topology	8
1.11	Closure	9
1.12	Compactness	9
1.13	Locally finite	9
1.14	Refinement of cover	10
1.15	Functions as vector space	11
1.16	Additional lemmas	11
1.17	Continuity	11
1.18	(<i>has-derivative</i>)	12
1.19	Differentiable	12
1.20	Frechet derivative	14
1.21	Linear algebra	17
1.22	Extensional function space	18

2	Smooth Functions between Normed Vector Spaces	20
2.1	From/To <i>Multivariate-Taylor.thy</i>	20
2.2	Higher-order differentiable	20
2.3	Higher directional derivatives	25
2.4	Smoothness	29
2.5	Diffeomorphism	33
3	Bump Functions	33
3.1	Construction	34
3.2	Cutoff function	36
3.3	Bump function	36
4	Charts	37
4.1	Definition	37
4.2	Properties	38
4.3	Restriction	40
4.4	Composition	40
5	Topological Manifolds	41
5.1	Defintition	41
5.2	Existence of locally finite cover	41
6	Differentiable/Smooth Manifolds	42
6.1	Smooth compatibility	42
6.2	C^k -Manifold	43
6.2.1	Atlas	43
6.2.2	Submanifold	45
6.3	Differentiable maps	46
6.4	Differentiable functions	49
6.5	Diffeomorphism	51
7	Partitions Of Unity	53
7.1	Regular cover	53
7.2	Partition of unity by smooth functions	54
8	Tangent Space	55
8.1	Real vector (sub)spaces	56
8.2	Derivations	57
8.3	Tangent space	59
8.4	Push-forward on the tangent space	61
8.5	Smooth inclusion map	63
8.6	Tangent space of submanifold	64
8.7	Directional derivatives	64
8.8	Dimension	66

9	Cotangent Space	67
9.1	Dual of a vector space	67
9.2	Dimension of dual space	68
9.3	Dual map	70
9.4	Definition of cotangent space	71
9.5	Pullback of cotangent space	72
9.6	Cotangent field of a function	72
9.7	Tangent field of a path	72
9.8	Integral along a path	72
10	Product Manifold	73
11	Sphere	74
12	Projective Space	76
12.1	Subtype of nonzero elements	76
12.2	Quotient	78
12.3	Proof of Hausdorff property	79
12.4	Charts	80
12.4.1	Chart for last coordinate	80
12.4.2	Charts for first $DIM('a)$ coordinates	81
12.4.3	Atlas	83

1 Library Additions

theory *Analysis-More*

imports *HOL-Analysis.Equivalence-Lebesgue-Henstock-Integration*

HOL-Library.Function-Algebras

HOL-Types-To-Sets.Linear-Algebra-On

begin

lemma *openin-open-Int'[intro]:*

open S ==> openin (top-of-set U) (S ∩ U)

<proof>

1.1 Parametricity rules for topology

TODO: also check with theory *Transfer-Euclidean-Space-Vector* in AFP/ODE...

context includes *lifting-syntax* **begin**

lemma *Sigma-transfer[transfer-rule]:*

(rel-set A ===> (A ===> rel-set B) ===> rel-set (rel-prod A B)) Sigma

Sigma

<proof>

lemma *filterlim-transfer*[*transfer-rule*]:
 (($A \text{====>} B$) $\text{====>} \text{rel-filter } B \text{====>} \text{rel-filter } A \text{====>} (=)$) *filterlim filterlim*
if [*transfer-rule*]: *bi-unique B*
 ⟨*proof*⟩

lemma *nhds-transfer*[*transfer-rule*]:
 ($A \text{====>} \text{rel-filter } A$) *nhds nhds*
if [*transfer-rule*]: *bi-unique A bi-total A (rel-set A $\text{====>} (=)$) open open*
 ⟨*proof*⟩

lemma *at-within-transfer*[*transfer-rule*]:
 ($A \text{====>} \text{rel-set } A \text{====>} \text{rel-filter } A$) *at-within at-within*
if [*transfer-rule*]: *bi-unique A bi-total A (rel-set A $\text{====>} (=)$) open open*
 ⟨*proof*⟩

lemma *continuous-on-transfer*[*transfer-rule*]:
 ($\text{rel-set } A \text{====>} (A \text{====>} B) \text{====>} (=)$) *continuous-on continuous-on*
if [*transfer-rule*]: *bi-unique A bi-total A (rel-set A $\text{====>} (=)$) open open*
bi-unique B bi-total B (rel-set B $\text{====>} (=)$) open open
 ⟨*proof*⟩

lemma *continuous-on-transfer-right-total*[*transfer-rule*]:
 ($\text{rel-set } A \text{====>} (A \text{====>} B) \text{====>} (=)$) ($\lambda X::'a::t2\text{-space set. continuous-on}$
 ($X \cap \text{Collect } AP$)) ($\lambda Y::'b::t2\text{-space set. continuous-on } Y$)
if *DomainA: Domainp A = AP*
and [*folded DomainA, transfer-rule*]: *bi-unique A right-total A (rel-set A $\text{====>} (=)$)*
 (*openin (top-of-set (Collect AP))) open*
bi-unique B bi-total B (rel-set B $\text{====>} (=)$) open open
 ⟨*proof*⟩

lemma *continuous-on-transfer-right-total2*[*transfer-rule*]:
 ($\text{rel-set } A \text{====>} (A \text{====>} B) \text{====>} (=)$) ($\lambda X::'a::t2\text{-space set. continuous-on}$
 X) ($\lambda Y::'b::t2\text{-space set. continuous-on } Y$)
if *DomainB: Domainp B = BP*
and [*folded DomainB, transfer-rule*]: *bi-unique A bi-total A (rel-set A $\text{====>} (=)$)*
 (*open open*
*bi-unique B right-total B (rel-set B $\text{====>} (=)$) ((*openin (top-of-set (Collect*
BP)))) open
 ⟨*proof*⟩*

lemma *generate-topology-transfer*[*transfer-rule*]:
includes *lifting-syntax*
assumes [*transfer-rule*]: *right-total A bi-unique A*
shows ($\text{rel-set } (\text{rel-set } A) \text{====>} \text{rel-set } A \text{====>} (=)$) (*generate-topology o*
 (*insert (Collect (Domainp A))*)) *generate-topology*
 ⟨*proof*⟩

end

1.2 Miscellaneous

lemmas [simp del] = mem-ball

lemma in-closureI[*intro, simp*]: $x \in X \implies x \in \text{closure } X$
<proof>

lemmas open-continuous-vimage = continuous-on-open-vimage[*THEN iffD1, rule-format*]

lemma open-continuous-vimage': open $s \implies \text{continuous-on } s \ f \implies \text{open } B \implies$
open $(s \cap f^{-1} B)$
<proof>

lemma support-on-mono: support-on carrier $f \subseteq \text{support-on carrier } g$
if $\bigwedge x. x \in \text{carrier} \implies f x \neq 0 \implies g x \neq 0$
<proof>

lemma image-prod: $(\lambda(x, y). (f x, g y))^{-1} (A \times B) = f^{-1} A \times g^{-1} B$ <proof>

1.3 Closed support

definition csupport-on $X \ S = \text{closure } (\text{support-on } X \ S)$

lemma closed-csupport-on[*intro, simp*]: closed (csupport-on carrier φ)
<proof>

lemma not-in-csupportD: $x \notin \text{csupport-on carrier } \varphi \implies x \in \text{carrier} \implies \varphi x = 0$
<proof>

lemma csupport-on-mono: csupport-on carrier $f \subseteq \text{csupport-on carrier } g$
if $\bigwedge x. x \in \text{carrier} \implies f x \neq 0 \implies g x \neq 0$
<proof>

1.4 Homeomorphism

lemma homeomorphism-empty[*simp*]:
homeomorphism $\{\}$ $t \ f \ f' \longleftrightarrow t = \{\}$
homeomorphism $s \ \{\}$ $f \ f' \longleftrightarrow s = \{\}$
<proof>

lemma homeomorphism-add:
homeomorphism UNIV UNIV $(\lambda x. x + c) (\lambda x. x - c)$
for $c::\text{real-normed-vector}$
<proof>

lemma in-range-scaleR-iff: $x \in \text{range } ((*_R) c) \longleftrightarrow c = 0 \longrightarrow x = 0$
for $x::\text{real-vector}$
<proof>

lemma homeomorphism-scaleR:

homeomorphism UNIV UNIV ($\lambda x. c *_{\mathbb{R}} x ::: \text{real-normed-vector}$) ($\lambda x. x /_{\mathbb{R}} c$)
if $c \neq 0$
 <proof>

lemma *homeomorphism-prod*:
homeomorphism ($a \times b$) ($c \times d$) ($\lambda(x, y). (f x, g y)$) ($\lambda(x, y). (f' x, g' y)$)
if *homeomorphism* $a c f f'$
 homeomorphism $b d g g'$
 <proof>

1.5 Generalizations

lemma *openin-subtopology-eq-generate-topology*:
openin (*top-of-set* S) $x = \text{generate-topology}$ (*insert* S (($\lambda B. B \cap S$) ' BB)) x
if *open-gen*: *open* = *generate-topology* BB **and** *subset*: $x \subseteq S$
 <proof>

1.6 Equal topologies

lemma *topology-eq-iff*: $t = s \iff (\text{topspace } t = \text{topspace } s \wedge$
 ($\forall x \subseteq \text{topspace } t. \text{openin } t x = \text{openin } s x$)
 <proof>

1.7 Finer topologies

definition *finer-than* (**infix** (*finer'-than*) 50)
where $T1 \text{ finer-than } T2 \iff \text{continuous-map } T1 T2$ ($\lambda x. x$)

lemma *finer-than-iff-nhds*:
 $T1 \text{ finer-than } T2 \iff (\forall X. \text{openin } T2 X \longrightarrow \text{openin } T1 (X \cap \text{topspace } T1)) \wedge$
 ($\text{topspace } T1 \subseteq \text{topspace } T2$)
 <proof>

lemma *continuous-on-finer-topo*:
continuous-map $s t f$
if *continuous-map* $s' t f s$ *finer-than* s'
 <proof>

lemma *continuous-on-finer-topo2*:
continuous-map $s t f$
if *continuous-map* $s t' f t'$ *finer-than* t
 <proof>

lemma *antisym-finer-than*: $S = T$ **if** $S \text{ finer-than } T$ $T \text{ finer-than } S$
 <proof>

lemma *subtopology-finer-than[simp]*: *top-of-set* X *finer-than* *euclidean*
 <proof>

1.8 Support

lemma *support-on-nonneg-sum:*

support-on X $(\lambda x. \sum_{i \in S}. f\ i\ x) = (\bigcup_{i \in S}. \text{support-on } X\ (f\ i))$
if *finite S* $\wedge x\ i. x \in X \implies i \in S \implies f\ i\ x \geq 0$
for $f :: \Rightarrow \Rightarrow :: \text{ordered-comm-monoid-add}$
 $\langle \text{proof} \rangle$

lemma *support-on-nonneg-sum-subset:*

support-on X $(\lambda x. \sum_{i \in S}. f\ i\ x) \subseteq (\bigcup_{i \in S}. \text{support-on } X\ (f\ i))$
for $f :: \Rightarrow \Rightarrow :: \text{ordered-comm-monoid-add}$
 $\langle \text{proof} \rangle$

lemma *support-on-nonneg-sum-subset':*

support-on X $(\lambda x. \sum_{i \in S} x. f\ i\ x) \subseteq (\bigcup_{x \in X}. (\bigcup_{i \in S} x. \text{support-on } X\ (f\ i)))$
for $f :: \Rightarrow \Rightarrow :: \text{ordered-comm-monoid-add}$
 $\langle \text{proof} \rangle$

1.9 Final topology (Bourbaki, General Topology I, 4.)

definition *final-topology X Y f =*

topology $(\lambda U. U \subseteq X \wedge$
 $(\forall i. \text{openin } (Y\ i)\ (f\ i - ' U \cap \text{topspace } (Y\ i))))$

lemma *openin-final-topology:*

openin $(\text{final-topology } X\ Y\ f) =$
 $(\lambda U. U \subseteq X \wedge (\forall i. \text{openin } (Y\ i)\ (f\ i - ' U \cap \text{topspace } (Y\ i))))$
 $\langle \text{proof} \rangle$

lemma *topspace-final-topology:*

topspace $(\text{final-topology } X\ Y\ f) = X$
if $\bigwedge i. f\ i \in \text{topspace } (Y\ i) \rightarrow X$
 $\langle \text{proof} \rangle$

lemma *continuous-on-final-topologyI2:*

continuous-map $(Y\ i)\ (\text{final-topology } X\ Y\ f)\ (f\ i)$
if $\bigwedge i. f\ i \in \text{topspace } (Y\ i) \rightarrow X$
 $\langle \text{proof} \rangle$

lemma *continuous-on-final-topologyI1:*

continuous-map $(\text{final-topology } X\ Y\ f)\ Z\ g$
if hyp: $\bigwedge i. \text{continuous-map } (Y\ i)\ Z\ (g \circ f\ i)$
and that: $\bigwedge i. f\ i \in \text{topspace } (Y\ i) \rightarrow X\ g \in X \rightarrow \text{topspace } Z$
 $\langle \text{proof} \rangle$

lemma *continuous-on-final-topology-iff:*

continuous-map $(\text{final-topology } X\ Y\ f)\ Z\ g \iff (\forall i. \text{continuous-map } (Y\ i)\ Z\ (g \circ f\ i))$
if $\bigwedge i. f\ i \in \text{topspace } (Y\ i) \rightarrow X\ g \in X \rightarrow \text{topspace } Z$

<proof>

1.10 Quotient topology

definition *map-topology* :: ('a \Rightarrow 'b) \Rightarrow 'a topology \Rightarrow 'b topology **where**
map-topology p X = *final-topology* (p ' *topspace* X) (λ -. X) (λ (::unit). p)

lemma *openin-map-topology*:

openin (*map-topology* p X) = (λ U. U \subseteq p ' *topspace* X \wedge *openin* X (p -' U \cap *topspace* X))

<proof>

lemma *topspace-map-topology[simp]*: *topspace* (*map-topology* f T) = f ' *topspace* T

<proof>

lemma *continuous-on-map-topology*:

continuous-map T (*map-topology* f T) f

<proof>

lemma *continuous-map-composeD*:

continuous-map T X (g \circ f) \Longrightarrow g \in f ' *topspace* T \rightarrow *topspace* X

<proof>

lemma *continuous-on-map-topology2*:

continuous-map T X (g \circ f) \longleftrightarrow *continuous-map* (*map-topology* f T) X g

<proof>

lemma *map-sub-finer-than-commute*:

map-topology f (*subtopology* T (f -' X)) *finer-than* *subtopology* (*map-topology* f T) X

<proof>

lemma *sub-map-finer-than-commute*:

subtopology (*map-topology* f T) X *finer-than* *map-topology* f (*subtopology* T (f -' X))

if *openin* T (f -' X)— this is more or less the condition from <https://math.stackexchange.com/questions/705840/quotient-topology-vs-subspace-topology>

<proof>

lemma *subtopology-map-topology*:

subtopology (*map-topology* f T) X = *map-topology* f (*subtopology* T (f -' X))

if *openin* T (f -' X)

<proof>

lemma *quotient-map-map-topology*:

quotient-map X (*map-topology* f X) f

<proof>

lemma *topological-space-quotient*: *class.topological-space* (*openin* (*map-topology* f

euclidean))
if *surj f*
 ⟨*proof*⟩

lemma *t2-space-quotient*: *class.t2-space (open::'b set ⇒ bool)*
if *open-def*: *open = (openin (map-topology (p::'a::t2-space⇒'b::topological-space)*
euclidean))

surj p and open-p: $\bigwedge X. \text{open } X \implies \text{open } (p \text{ ' } X)$ **and** *closed* $\{(x, y). p \ x = p \ y\}$ **(is closed ?R)**
 ⟨*proof*⟩

lemma *second-countable-topology-quotient*: *class.second-countable-topology (open::'b*
set ⇒ bool)
if *open-def*: *open = (openin (map-topology (p::'a::second-countable-topology⇒'b::topological-space)*
euclidean))

surj p and open-p: $\bigwedge X. \text{open } X \implies \text{open } (p \text{ ' } X)$
 ⟨*proof*⟩

1.11 Closure

lemma *closure-Union*: *closure* $(\bigcup X) = (\bigcup x \in X. \text{closure } x)$ **if** *finite X*
 ⟨*proof*⟩

1.12 Compactness

lemma *compact-if-closed-subset-of-compact*:
compact S if closed S compact T S ⊆ T
 ⟨*proof*⟩

1.13 Locally finite

definition *locally-finite-on X I U* $\longleftrightarrow (\forall p \in X. \exists N. p \in N \wedge \text{open } N \wedge \text{finite } \{i \in I. U \ i \cap N \neq \{\}\})$

lemmas *locally-finite-onI = locally-finite-on-def* [THEN *iffD2*, *rule-format*]

lemma *locally-finite-onE*:
assumes *locally-finite-on X I U*
assumes *p ∈ X*
obtains *N where p ∈ N open N finite {i ∈ I. U i ∩ N ≠ {}}*
 ⟨*proof*⟩

lemma *locally-finite-onD*:
assumes *locally-finite-on X I U*
assumes *p ∈ X*
shows *finite {i ∈ I. p ∈ U i}*
 ⟨*proof*⟩

lemma *locally-finite-on-open-coverI*: *locally-finite-on X I U*
if *fin*: $\bigwedge j. j \in I \implies \text{finite } \{i \in I. U \ i \cap U \ j \neq \{\}\}$

and *open-cover*: $X \subseteq (\bigcup_{i \in I} U_i) \wedge i. i \in I \implies \text{open } (U_i)$
 ⟨proof⟩

lemma *locally-finite-compactD*:

finite $\{i \in I. U_i \cap V \neq \{\}\}$

if *lf*: *locally-finite-on* $X I U$

and *compact*: *compact* V

and *subset*: $V \subseteq X$

⟨proof⟩

lemma *closure-Int-open-eq-empty*: $\text{open } S \implies (\text{closure } T \cap S) = \{\} \iff T \cap S = \{\}$

⟨proof⟩

lemma *locally-finite-on-subset*:

assumes *locally-finite-on* $X J U$

assumes $\bigwedge i. i \in I \implies V_i \subseteq U_i \subseteq J$

shows *locally-finite-on* $X I V$

⟨proof⟩

lemma *locally-finite-on-closure*:

locally-finite-on $X I (\lambda x. \text{closure } (U x))$

if *locally-finite-on* $X I U$

⟨proof⟩

lemma *locally-finite-on-closedin-Union-closure*:

closedin (*top-of-set* X) $(\bigcup_{i \in I} \text{closure } (U_i))$

if *locally-finite-on* $X I U \wedge i. i \in I \implies \text{closure } (U_i) \subseteq X$

⟨proof⟩

lemma *closure-subtopology-minimal*:

$S \subseteq T \implies \text{closedin } (\text{top-of-set } X) T \implies \text{closure } S \cap X \subseteq T$

⟨proof⟩

lemma *locally-finite-on-closure-Union*:

$(\bigcup_{i \in I} \text{closure } (U_i)) = \text{closure } (\bigcup_{i \in I} (U_i)) \cap X$

if *locally-finite-on* $X I U \wedge i. i \in I \implies \text{closure } (U_i) \subseteq X$

⟨proof⟩

1.14 Refinement of cover

definition *refines* :: 'a set set \Rightarrow 'a set set \Rightarrow bool (**infix** *refines* 50)

where $A \text{ refines } B \iff (\forall s \in A. (\exists t. t \in B \wedge s \subseteq t))$

lemma *refines-subset*: $x \text{ refines } y \text{ if } z \text{ refines } y \wedge x \subseteq z$

⟨proof⟩

1.15 Functions as vector space

instantiation *fun* :: (*type*, *scaleR*) *scaleR* **begin**

definition *scaleR-fun* :: *real* \Rightarrow (*'a* \Rightarrow *'b*) \Rightarrow *'a* \Rightarrow *'b* **where**
scaleR-fun *r f* = ($\lambda x. r *_{\mathbb{R}} f x$)

lemma *scaleR-fun-beta[simp]*: ($r *_{\mathbb{R}} f$) *x* = $r *_{\mathbb{R}} f x$
<proof>

instance *<proof>*

end

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

1.16 Additional lemmas

lemmas [*simp del*] = *vimage-Un vimage-Int*

lemma *finite-Collect-imageI*: *finite* {*U* \in *f* ' *X*. *P U*} **if** *finite* {*x* \in *X*. *P (f x)*}
<proof>

lemma *plus-compose*: (*x* + *y*) \circ *f* = (*x* \circ *f*) + (*y* \circ *f*)
<proof>

lemma *mult-compose*: (*x* * *y*) \circ *f* = (*x* \circ *f*) * (*y* \circ *f*)
<proof>

lemma *scaleR-compose*: ($c *_{\mathbb{R}} x$) \circ *f* = $c *_{\mathbb{R}}$ (*x* \circ *f*)
<proof>

lemma *image-scaleR-ball*:
fixes *a* :: *'a*::*real-normed-vector*
shows $c \neq 0 \implies (*_{\mathbb{R}}) c$ ' *ball a r* = *ball (c *_ℝ a) (abs c *_ℝ r)*
<proof>

1.17 Continuity

lemma *continuous-within-topologicalE*:
assumes *continuous (at x within s) f*
open B f x \in *B*
obtains *A* **where** *open A x* \in *A* \wedge *y*. *y* \in *s* \implies *y* \in *A* \implies *f y* \in *B*
<proof>

lemma *continuous-within-topologicalE'*:
assumes *continuous (at x) f*
open B f x \in *B*
obtains *A* **where** *open A x* \in *A* f ' *A* \subseteq *B*

<proof>

lemma *continuous-on-inverse: continuous-on S f \implies 0 \notin f ' S \implies continuous-on S (λx . inverse (f x))*
for *f :: \Rightarrow :: real-normed-div-algebra*
<proof>

1.18 (*has-derivative*)

lemma *has-derivative-plus-fun[derivative-intros]:*
(x + y has-derivative x' + y') (at a within A)
if *[derivative-intros]:*
(x has-derivative x') (at a within A)
(y has-derivative y') (at a within A)
<proof>

lemma *has-derivative-scaleR-fun[derivative-intros]:*
*(x *_R y has-derivative x *_R y') (at a within A)*
if *[derivative-intros]:*
(y has-derivative y') (at a within A)
<proof>

lemma *has-derivative-times-fun[derivative-intros]:*
*(x * y has-derivative (λh . x a * y' h + x' h * y a)) (at a within A)*
if *[derivative-intros]:*
(x has-derivative x') (at a within A)
(y has-derivative y') (at a within A)
for *x y :: \Rightarrow 'a :: real-normed-algebra*
<proof>

lemma *real-sqrt-has-derivative-generic:*
x \neq 0 \implies (sqrt has-derivative (((if x > 0 then 1 else -1) * inverse (sqrt x) / 2)) (at x within S)*
<proof>

lemma *sqrt-has-derivative:*
*((λx . sqrt (f x)) has-derivative (λxa . (if 0 < f x then 1 else - 1) / (2 * sqrt (f x)) * f' xa)) (at x within S)*
if *(f has-derivative f') (at x within S) f x \neq 0*
<proof>

lemmas *has-derivative-norm-compose[derivative-intros] = has-derivative-compose[OF - has-derivative-norm]*

1.19 Differentiable

lemmas *differentiable-on-empty[simp]*

lemma *differentiable-transform-eventually: f differentiable (at x within X)*
if *g differentiable (at x within X)*

$f x = g x$
 $\forall_F x \text{ in } (\text{at } x \text{ within } X). f x = g x$
 <proof>

lemma *differentiable-within-eqI*: f differentiable at x within X
 if g differentiable at x within $X \wedge x. x \in X \implies f x = g x$
 $x \in X$ open X
 <proof>

lemma *differentiable-eqI*: f differentiable at x
 if g differentiable at $x \wedge x. x \in X \implies f x = g x$ $x \in X$ open X
 <proof>

lemma *differentiable-on-eqI*:
 f differentiable-on S
 if g differentiable-on $S \wedge x. x \in S \implies f x = g x$ open S
 <proof>

lemma *differentiable-on-comp*: $(f \circ g)$ differentiable-on S
 if g differentiable-on S f differentiable-on $(g \text{ ' } S)$
 <proof>

lemma *differentiable-on-comp2*: $(f \circ g)$ differentiable-on S
 if f differentiable-on T g differentiable-on S $g \text{ ' } S \subseteq T$
 <proof>

lemmas *differentiable-on-compose2* = *differentiable-on-comp2*[unfolded o-def]

lemma *differentiable-on-openD*: f differentiable at x
 if f differentiable-on X open X $x \in X$
 <proof>

lemma *differentiable-on-add-fun*[intro, simp]:
 x differentiable-on $UNIV \implies y$ differentiable-on $UNIV \implies x + y$ differentiable-on
 $UNIV$
 <proof>

lemma *differentiable-on-mult-fun*[intro, simp]:
 x differentiable-on $UNIV \implies y$ differentiable-on $UNIV \implies x * y$ differentiable-on
 $UNIV$
 for $x y :: \Rightarrow 'a :: \text{real-normed-algebra}$
 <proof>

lemma *differentiable-on-scaleR-fun*[intro, simp]:
 y differentiable-on $UNIV \implies x *_{\mathbb{R}} y$ differentiable-on $UNIV$
 <proof>

lemma *sqrt-differentiable*:
 $(\lambda x. \text{sqrt } (f x))$ differentiable (at x within S)

if f differentiable (at x within S) $f' x \neq 0$
(proof)

lemma *sqrt-differentiable-on*: $(\lambda x. \text{sqrt } (f x))$ differentiable-on S
if f differentiable-on S $0 \notin f' S$
(proof)

lemma *differentiable-on-inverse*: f differentiable-on $S \implies 0 \notin f' S \implies (\lambda x. \text{inverse } (f x))$ differentiable-on S
for $f :: \Rightarrow :: \text{real-normed-field}$
(proof)

lemma *differentiable-on-openI*:
 f differentiable-on S
if open $S \wedge x. x \in S \implies \exists f'. (f \text{ has-derivative } f') (at x)$
(proof)

lemmas *differentiable-norm-compose-at* = *differentiable-compose*[OF *differentiable-norm-at*]

lemma *differentiable-on-Pair*:
 f differentiable-on $S \implies g$ differentiable-on $S \implies (\lambda x. (f x, g x))$ differentiable-on S
(proof)

lemma *differentiable-at-fst*:
 $(\lambda x. \text{fst } (f x))$ differentiable at x within X **if** f differentiable at x within X
(proof)

lemma *differentiable-at-snd*:
 $(\lambda x. \text{snd } (f x))$ differentiable at x within X **if** f differentiable at x within X
(proof)

lemmas *frechet-derivative-worksI* = *frechet-derivative-works*[THEN *iffD1*]

lemma *sin-differentiable-at*: $(\lambda x. \text{sin } (f x :: \text{real}))$ differentiable at x within X
if f differentiable at x within X
(proof)

lemma *cos-differentiable-at*: $(\lambda x. \text{cos } (f x :: \text{real}))$ differentiable at x within X
if f differentiable at x within X
(proof)

1.20 Frechet derivative

lemmas *frechet-derivative-transform-within-open-ext* =
fun-cong[OF *frechet-derivative-transform-within-open*]

lemmas *frechet-derivative-at'* = *frechet-derivative-at*[*symmetric*]

lemma *frechet-derivative-plus-fun*:

x differentiable at $a \implies y$ differentiable at $a \implies$
 $\text{frechet-derivative } (x + y) \text{ (at } a) =$
 $\text{frechet-derivative } x \text{ (at } a) + \text{frechet-derivative } y \text{ (at } a)$
(proof)

lemmas *frechet-derivative-plus* = *frechet-derivative-plus-fun*[unfolded plus-fun-def]

lemma *frechet-derivative-zero-fun*: $\text{frechet-derivative } 0 \text{ (at } a) = 0$

(proof)

lemma *frechet-derivative-sin*:

$\text{frechet-derivative } (\lambda x. \sin (f x)) \text{ (at } x) = (\lambda xa. \text{frechet-derivative } f \text{ (at } x) xa * \cos (f x))$
if f differentiable (at x)
for $f :: \Rightarrow \text{real}$
(proof)

lemma *frechet-derivative-cos*:

$\text{frechet-derivative } (\lambda x. \cos (f x)) \text{ (at } x) = (\lambda xa. \text{frechet-derivative } f \text{ (at } x) xa * - \sin (f x))$
if f differentiable (at x)
for $f :: \Rightarrow \text{real}$
(proof)

lemma *differentiable-sum-fun*:

$(\bigwedge i. i \in I \implies (f i \text{ differentiable at } a)) \implies \text{sum } f I \text{ differentiable at } a$
(proof)

lemma *frechet-derivative-sum-fun*:

$(\bigwedge i. i \in I \implies (f i \text{ differentiable at } a)) \implies$
 $\text{frechet-derivative } (\sum i \in I. f i) \text{ (at } a) = (\sum i \in I. \text{frechet-derivative } (f i) \text{ (at } a))$
(proof)

lemma *sum-fun-def*: $(\sum i \in I. f i) = (\lambda x. \sum i \in I. f i x)$

(proof)

lemmas *frechet-derivative-sum* = *frechet-derivative-sum-fun*[unfolded sum-fun-def]

lemma *frechet-derivative-times-fun*:

f differentiable at $a \implies g$ differentiable at $a \implies$
 $\text{frechet-derivative } (f * g) \text{ (at } a) =$
 $(\lambda x. f a * \text{frechet-derivative } g \text{ (at } a) x + \text{frechet-derivative } f \text{ (at } a) x * g a)$
for $f g :: \Rightarrow 'a :: \text{real-normed-algebra}$
(proof)

lemmas *frechet-derivative-times* = *frechet-derivative-times-fun*[unfolded times-fun-def]

lemma *frechet-derivative-scaleR-fun:*

y differentiable at *a* \implies
frechet-derivative (*x* *_R *y*) (at *a*) =
x *_R *frechet-derivative* *y* (at *a*)
(proof)

lemmas *frechet-derivative-scaleR* = *frechet-derivative-scaleR-fun*[unfolded *scaleR-fun-def*]

lemma *frechet-derivative-compose:*

frechet-derivative (*f* o *g*) (at *x*) = *frechet-derivative* (*f*) (at (*g* *x*)) o *frechet-derivative*
g (at *x*)
if *g* differentiable at *x* *f* differentiable at (*g* *x*)
(proof)

lemma *frechet-derivative-compose-eucl:*

frechet-derivative (*f* o *g*) (at *x*) =
($\lambda v. \sum i \in \text{Basis}. ((\text{frechet-derivative } g \text{ (at } x) \ v) \cdot i) *_{\mathbb{R}} \text{frechet-derivative } f \text{ (at } (g \ x)) \ i)$)
(**is** ?*l* = ?*r*)
if *g* differentiable at *x* *f* differentiable at (*g* *x*)
(proof)

lemma *frechet-derivative-works-on-open:*

f differentiable-on *X* \implies open *X* \implies *x* \in *X* \implies
(*f* has-derivative *frechet-derivative* *f* (at *x*)) (at *x*)
and *frechet-derivative-works-on:*
f differentiable-on *X* \implies *x* \in *X* \implies
(*f* has-derivative *frechet-derivative* *f* (at *x* within *X*)) (at *x* within *X*)
(proof)

lemma *frechet-derivative-inverse:* *frechet-derivative* ($\lambda x. \text{inverse } (f \ x)$) (at *x*) =

($\lambda h. -1 / (f \ x)^2 * \text{frechet-derivative } f \text{ (at } x) \ h$)
if *f* differentiable at *x* *f* *x* \neq 0 **for** *f*:: \rightarrow real-normed-field
(proof)

lemma *frechet-derivative-sqrt:* *frechet-derivative* ($\lambda x. \text{sqrt } (f \ x)$) (at *x*) =

($\lambda v. (\text{if } f \ x > 0 \text{ then } 1 \text{ else } -1) / (2 * \text{sqrt } (f \ x)) * \text{frechet-derivative } f \text{ (at } x) \ v$)
if *f* differentiable at *x* *f* *x* \neq 0
(proof)

lemma *frechet-derivative-norm:* *frechet-derivative* ($\lambda x. \text{norm } (f \ x)$) (at *x*) =

($\lambda v. \text{frechet-derivative } f \text{ (at } x) \ v \cdot \text{sgn } (f \ x)$)
if *f* differentiable at *x* *f* *x* \neq 0
for *f*:: \rightarrow real-inner
(proof)

lemma (**in** bounded-linear) *frechet-derivative:*

frechet-derivative *f* (at *x*) = *f*

<proof>

bundle *no-matrix-mult* **begin**
no-notation *matrix-matrix-mult* (**infixl** ** 70)
end

lemma (**in** *bounded-bilinear*) *frechet-derivative*:

includes *no-matrix-mult*

shows

x differentiable at $a \implies y$ differentiable at $a \implies$

$\text{frechet-derivative } (\lambda a. x a ** y a) (at a) =$

$(\lambda h. x a ** \text{frechet-derivative } y (at a) h + \text{frechet-derivative } x (at a) h ** y$

$a)$

<proof>

lemma *frechet-derivative-divide*: $\text{frechet-derivative } (\lambda x. f x / g x) (at x) =$

$(\lambda h. \text{frechet-derivative } f (at x) h / (g x) - \text{frechet-derivative } g (at x) h * f x /$
 $(g x)^2)$

if f differentiable at x g differentiable at x $g x \neq 0$ **for** $f :: \Rightarrow :: \text{real-normed-field}$

<proof>

lemma *frechet-derivative-pair*:

$\text{frechet-derivative } (\lambda x. (f x, g x)) (at x) = (\lambda v. (\text{frechet-derivative } f (at x) v,$
 $\text{frechet-derivative } g (at x) v))$

if f differentiable at x g differentiable at x

<proof>

lemma *frechet-derivative-fst*:

$\text{frechet-derivative } (\lambda x. \text{fst } (f x)) (at x) = (\lambda xa. \text{fst } (\text{frechet-derivative } f (at x) xa))$

if f differentiable at x

for $f :: \Rightarrow (-:: \text{real-normed-vector} \times -:: \text{real-normed-vector})$

<proof>

lemma *frechet-derivative-snd*:

$\text{frechet-derivative } (\lambda x. \text{snd } (f x)) (at x) = (\lambda xa. \text{snd } (\text{frechet-derivative } f (at x)$
 $xa))$

if f differentiable at x

for $f :: \Rightarrow (-:: \text{real-normed-vector} \times -:: \text{real-normed-vector})$

<proof>

lemma *frechet-derivative-eq-vector-derivative-1*:

assumes f differentiable at t

shows $\text{frechet-derivative } f (at t) 1 = \text{vector-derivative } f (at t)$

<proof>

1.21 Linear algebra

lemma (**in** *vector-space*) *dim-pos-finite-dimensional-vector-spaceE*:

assumes $\text{dim } (UNIV :: 'b \text{ set}) > 0$

obtains *basis* **where** *finite-dimensional-vector-space scale basis*
 ⟨*proof*⟩

context *vector-space-on* **begin**

context includes *lifting-syntax* **assumes** $\exists (Rep::'s \Rightarrow 'b) (Abs::'b \Rightarrow 's)$. *type-definition*
Rep Abs S **begin**

interpretation *local-typedef-vector-space-on S scale TYPE('s)* ⟨*proof*⟩

lemmas-with [*var-simplified explicit-ab-group-add,*
unoverload-type 'd,
OF type.ab-group-add-axioms type-vector-space-on-with,
folded dim-S-def,
untransferred,
var-simplified implicit-ab-group-add]:
lt-dim-pos-finite-dimensional-vector-spaceE = vector-space.dim-pos-finite-dimensional-vector-spaceE

end

lemmas-with [*cancel-type-definition,*
OF S-ne,
folded subset-iff',
simplified pred-fun-def, folded finite-dimensional-vector-space-on-with,
simplified— too much?]:
dim-pos-finite-dimensional-vector-spaceE = lt-dim-pos-finite-dimensional-vector-spaceE

end

1.22 Extensional function space

f is zero outside A . We use such functions to canonically represent functions whose domain is A

definition *extensional0* :: *'a set* \Rightarrow (*'a* \Rightarrow *'b::zero*) \Rightarrow *bool*
where *extensional0 A f* = ($\forall x. x \notin A \longrightarrow f x = 0$)

lemma *extensional0-0*[*intro, simp*]: *extensional0 X 0*
 ⟨*proof*⟩

lemma *extensional0-UNIV*[*intro, simp*]: *extensional0 UNIV f*
 ⟨*proof*⟩

lemma *ext-extensional0*:
f = g **if** *extensional0 S f extensional0 S g* $\wedge x. x \in S \Longrightarrow f x = g x$
 ⟨*proof*⟩

lemma *extensional0-add*[*intro, simp*]:
extensional0 S f \Longrightarrow *extensional0 S g* \Longrightarrow *extensional0 S (f + g::=>'a::comm-monoid-add)*
 ⟨*proof*⟩

lemma *extensional0-mult*[*intro, simp*]:
 $extensional0\ S\ x \implies extensional0\ S\ y \implies extensional0\ S\ (x * y)$
for $x\ y :: \Rightarrow 'a :: mult-zero$
<proof>

lemma *extensional0-scaleR*[*intro, simp*]: $extensional0\ S\ f \implies extensional0\ S\ (c *_{\mathbb{R}} f :: \Rightarrow 'a :: real-vector)$
<proof>

lemma *extensional0-outside*: $x \notin S \implies extensional0\ S\ f \implies f\ x = 0$
<proof>

lemma *subspace-extensional0*: $subspace\ (Collect\ (extensional0\ X))$
<proof>

Send the function f to its canonical representative as a function with domain A

definition *restrict0* :: $'a\ set \Rightarrow ('a \Rightarrow 'b :: zero) \Rightarrow 'a \Rightarrow 'b$
where $restrict0\ A\ f\ x = (if\ x \in A\ then\ f\ x\ else\ 0)$

lemma *restrict0-UNIV*[*simp*]: $restrict0\ UNIV = (\lambda x. x)$
<proof>

lemma *extensional0-restrict0*[*intro, simp*]: $extensional0\ A\ (restrict0\ A\ f)$
<proof>

lemma *restrict0-times*: $restrict0\ A\ (x * y) = restrict0\ A\ x * restrict0\ A\ y$
for $x :: 'a \Rightarrow 'b :: mult-zero$
<proof>

lemma *restrict0-apply-in*[*simp*]: $x \in A \implies restrict0\ A\ f\ x = f\ x$
<proof>

lemma *restrict0-apply-out*[*simp*]: $x \notin A \implies restrict0\ A\ f\ x = 0$
<proof>

lemma *restrict0-scaleR*: $restrict0\ A\ (c *_{\mathbb{R}} f :: \Rightarrow 'a :: real-vector) = c *_{\mathbb{R}} restrict0\ A\ f$
<proof>

lemma *restrict0-add*: $restrict0\ A\ (f + g :: \Rightarrow 'a :: real-vector) = restrict0\ A\ f + restrict0\ A\ g$
<proof>

lemma *restrict0-restrict0*: $restrict0\ X\ (restrict0\ Y\ f) = restrict0\ (X \cap Y)\ f$
<proof>

end

2 Smooth Functions between Normed Vector Spaces

```
theory Smooth
  imports
    Analysis-More
begin
```

2.1 From/To Multivariate-Taylor.thy

```
lemma multivariate-Taylor-integral:
  fixes f::'a::real-normed-vector  $\Rightarrow$  'b::banach
  and Df::'a  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  'b
  assumes n > 0
  assumes Df-Nil:  $\bigwedge a x. Df a 0 H = f a$ 
  assumes Df-Cons:  $\bigwedge a i d. a \in \text{closed-segment } X (X + H) \implies i < n \implies$ 
     $((\lambda a. Df a i H) \text{ has-derivative } (Df a (\text{Suc } i)))$  (at a within G)
  assumes cs: closed-segment X (X + H)  $\subseteq$  G
  defines i  $\equiv$   $\lambda x.$ 
     $((1 - x) ^ (n - 1) / \text{fact } (n - 1)) *_{\mathbb{R}} Df (X + x *_{\mathbb{R}} H) n H$ 
  shows multivariate-Taylor-has-integral:
     $(i \text{ has-integral } f (X + H) - (\sum i < n. (1 / \text{fact } i) *_{\mathbb{R}} Df X i H)) \{0..1\}$ 
  and multivariate-Taylor:
     $f (X + H) = (\sum i < n. (1 / \text{fact } i) *_{\mathbb{R}} Df X i H) + \text{integral } \{0..1\} i$ 
  and multivariate-Taylor-integrable:
    i integrable-on {0..1}
<proof>
```

2.2 Higher-order differentiable

```
fun higher-differentiable-on ::
  'a::real-normed-vector set  $\Rightarrow$  ('a  $\Rightarrow$  'b::real-normed-vector)  $\Rightarrow$  nat  $\Rightarrow$  bool where
  higher-differentiable-on S f 0  $\longleftrightarrow$  continuous-on S f
| higher-differentiable-on S f (Suc n)  $\longleftrightarrow$ 
   $(\forall x \in S. f \text{ differentiable } (\text{at } x)) \wedge$ 
   $(\forall v. \text{higher-differentiable-on } S (\lambda x. \text{frechet-derivative } f (\text{at } x) v) n)$ 
```

```
lemma ball-differentiable-atD:  $\forall x \in S. f \text{ differentiable at } x \implies f \text{ differentiable-on } S$ 
<proof>
```

```
lemma higher-differentiable-on-imp-continuous-on:
  continuous-on S f if higher-differentiable-on S f n
<proof>
```

```
lemma higher-differentiable-on-imp-differentiable-on:
  f differentiable-on S if higher-differentiable-on S f k k > 0
<proof>
```

```
lemma higher-differentiable-on-congI:
  assumes open S higher-differentiable-on S g n
  and  $\bigwedge x. x \in S \implies f x = g x$ 
```

shows *higher-differentiable-on S f n*
<proof>

lemma *higher-differentiable-on-cong*:
assumes *open S S = T*
and $\bigwedge x. x \in T \implies f x = g x$
shows *higher-differentiable-on S f n \longleftrightarrow higher-differentiable-on T g n*
<proof>

lemma *higher-differentiable-on-SucD*:
higher-differentiable-on S f n **if** *higher-differentiable-on S f (Suc n)*
<proof>

lemma *higher-differentiable-on-addD*:
higher-differentiable-on S f n **if** *higher-differentiable-on S f (n + m)*
<proof>

lemma *higher-differentiable-on-le*:
higher-differentiable-on S f n **if** *higher-differentiable-on S f m n \leq m*
<proof>

lemma *higher-differentiable-on-open-subsetsI*:
higher-differentiable-on S f n
if $\bigwedge x. x \in S \implies \exists T. x \in T \wedge \text{open } T \wedge \text{higher-differentiable-on } T f n$
<proof>

lemma *higher-differentiable-on-const*: *higher-differentiable-on S ($\lambda x. c$) n*
<proof>

lemma *higher-differentiable-on-id*: *higher-differentiable-on S ($\lambda x. x$) n*
<proof>

lemma *higher-differentiable-on-add*:
higher-differentiable-on S ($\lambda x. f x + g x$) n
if *higher-differentiable-on S f n*
higher-differentiable-on S g n
open S
<proof>

lemma (**in** *bounded-bilinear*) *differentiable*:
($\lambda x. \text{prod } (f x) (g x)$) differentiable at x within S
if *f differentiable at x within S*
g differentiable at x within S
<proof>

context begin
private lemmas *d = bounded-bilinear.differentiable*

lemmas *differentiable-inner* = *bounded-bilinear-inner*[*THEN d*]
and *differentiable-scaleR* = *bounded-bilinear-scaleR*[*THEN d*]
and *differentiable-mult* = *bounded-bilinear-mult*[*THEN d*]
end

lemma (**in** *bounded-bilinear*) *differentiable-on*:
 $(\lambda x. \text{prod } (f x) (g x)) \text{ differentiable-on } S$
if *f differentiable-on S g differentiable-on S*
 $\langle \text{proof} \rangle$

context begin

private lemmas *do* = *bounded-bilinear.differentiable-on*
lemmas *differentiable-on-inner* = *bounded-bilinear-inner*[*THEN do*]
and *differentiable-on-scaleR* = *bounded-bilinear-scaleR*[*THEN do*]
and *differentiable-on-mult* = *bounded-bilinear-mult*[*THEN do*]
end

lemma (**in** *bounded-bilinear*) *higher-differentiable-on*:
 $\text{higher-differentiable-on } S (\lambda x. \text{prod } (f x) (g x)) n$
if
 $\text{higher-differentiable-on } S f n$
 $\text{higher-differentiable-on } S g n$
 $\text{open } S$
 $\langle \text{proof} \rangle$

context begin

private lemmas *hdo* = *bounded-bilinear.higher-differentiable-on*
lemmas *higher-differentiable-on-inner* = *bounded-bilinear-inner*[*THEN hdo*]
and *higher-differentiable-on-scaleR* = *bounded-bilinear-scaleR*[*THEN hdo*]
and *higher-differentiable-on-mult* = *bounded-bilinear-mult*[*THEN hdo*]
end

lemma *higher-differentiable-on-sum*:
 $\text{higher-differentiable-on } S (\lambda x. \sum_{i \in F}. f i x) n$
if $\bigwedge i. i \in F \implies \text{finite } F \implies \text{higher-differentiable-on } S (f i) n \text{ open } S$
 $\langle \text{proof} \rangle$

lemma *higher-differentiable-on-subset*:
 $\text{higher-differentiable-on } S f n$
if $\text{higher-differentiable-on } T f n \text{ } S \subseteq T$
 $\langle \text{proof} \rangle$

lemma *higher-differentiable-on-compose*:
 $\text{higher-differentiable-on } S (f \circ g) n$
if $\text{higher-differentiable-on } T f n \text{ higher-differentiable-on } S g n \text{ } g ' S \subseteq T \text{ open } S$
 $\text{open } T$
for $g::\text{--}\rightarrow::\text{euclidean-space}$ — TODO: can we get around this restriction?
 $\langle \text{proof} \rangle$

lemma *higher-differentiable-on-uminus:*
higher-differentiable-on S $(\lambda x. - f x)$ n
if *higher-differentiable-on* S f n *open* S
 \langle *proof* \rangle

lemma *higher-differentiable-on-minus:*
higher-differentiable-on S $(\lambda x. f x - g x)$ n
if *higher-differentiable-on* S f n
higher-differentiable-on S g n
open S
 \langle *proof* \rangle

lemma *higher-differentiable-on-inverse:*
higher-differentiable-on S $(\lambda x. \text{inverse } (f x))$ n
if *higher-differentiable-on* S f n $0 \notin f' S$ *open* S
for $f :: \Rightarrow :: \text{real-normed-field}$
 \langle *proof* \rangle

lemma *higher-differentiable-on-divide:*
higher-differentiable-on S $(\lambda x. f x / g x)$ n
if
higher-differentiable-on S f n
higher-differentiable-on S g n
 $\bigwedge x. x \in S \implies g x \neq 0$
open S
for $f :: \Rightarrow :: \text{real-normed-field}$
 \langle *proof* \rangle

lemma *differentiable-on-open-Union:*
f differentiable-on $\bigcup S$
if $\bigwedge s. s \in S \implies f \text{ differentiable-on } s$
 $\bigwedge s. s \in S \implies \text{open } s$
 \langle *proof* \rangle

lemma *higher-differentiable-on-open-Union: higher-differentiable-on* $(\bigcup S)$ f n
if $\bigwedge s. s \in S \implies \text{higher-differentiable-on } s$ f n
 $\bigwedge s. s \in S \implies \text{open } s$
 \langle *proof* \rangle

lemma *differentiable-on-open-Un:*
f differentiable-on $S \cup T$
if *f differentiable-on* S
f differentiable-on T
open S *open* T
 \langle *proof* \rangle

lemma *higher-differentiable-on-open-Un: higher-differentiable-on* $(S \cup T)$ f n
if *higher-differentiable-on* S f n

higher-differentiable-on T f n
open S *open* T
 ⟨proof⟩

lemma *higher-differentiable-on-sqrt*: *higher-differentiable-on* S $(\lambda x. \text{sqrt } (f x))$ n
if *higher-differentiable-on* S f n $0 \notin f' S$ *open* S
 ⟨proof⟩

lemma *higher-differentiable-on-frechet-derivativeI*:
higher-differentiable-on X $(\lambda x. \text{frechet-derivative } f \text{ (at } x) h) i$
if *higher-differentiable-on* X f $(\text{Suc } i)$ *open* X $x \in X$
 ⟨proof⟩

lemma *higher-differentiable-on-norm*:
higher-differentiable-on S $(\lambda x. \text{norm } (f x))$ n
if *higher-differentiable-on* S f n $0 \notin f' S$ *open* S
for $f :: \Rightarrow :: \text{real-inner}$
 ⟨proof⟩

declare *higher-differentiable-on.simps* [*simp del*]

lemma *higher-differentiable-on-Pair*:
higher-differentiable-on S f $k \implies \text{higher-differentiable-on } S$ g $k \implies$
higher-differentiable-on S $(\lambda x. (f x, g x))$ k
if *open* S
 ⟨proof⟩

lemma *higher-differentiable-on-compose'*:
higher-differentiable-on S $(\lambda x. f (g x))$ n
if *higher-differentiable-on* T f n *higher-differentiable-on* S g n $g' S \subseteq T$ *open* S
open T
for $g :: \Rightarrow :: \text{euclidean-space}$
 ⟨proof⟩

lemma *higher-differentiable-on-fst*:
higher-differentiable-on $(S \times T)$ fst k
 ⟨proof⟩

lemma *higher-differentiable-on-snd*:
higher-differentiable-on $(S \times T)$ snd k
 ⟨proof⟩

lemma *higher-differentiable-on-fst-comp*:
higher-differentiable-on S $(\lambda x. \text{fst } (f x))$ k
if *higher-differentiable-on* S f k *open* S
 ⟨proof⟩

lemma *higher-differentiable-on-snd-comp*:

higher-differentiable-on S ($\lambda x. \text{snd } (f x)$) *k*
if *higher-differentiable-on S f k open S*
 ⟨*proof*⟩

lemma *higher-differentiable-on-Pair'*:
higher-differentiable-on S f k \implies *higher-differentiable-on T g k* \implies
higher-differentiable-on (S × T) ($\lambda x. (f (\text{fst } x), g (\text{snd } x))$) k
if *S: open S and T: open T*
for *f:::euclidean-space \implies - and g:::euclidean-space \implies -*
 ⟨*proof*⟩

lemma *higher-differentiable-on-sin: higher-differentiable-on S* ($\lambda x. \text{sin } (f x::\text{real})$)
n
and *higher-differentiable-on-cos: higher-differentiable-on S* ($\lambda x. \text{cos } (f x::\text{real})$) *n*
if *f: higher-differentiable-on S f n and S: open S*
 ⟨*proof*⟩

2.3 Higher directional derivatives

primrec *nth-derivative* :: *nat* \implies (*'a::real-normed-vector* \implies *'b::real-normed-vector*)
 \implies *'a* \implies *'a* \implies *'b* **where**
nth-derivative 0 f x h = f x
 | *nth-derivative (Suc i) f x h = nth-derivative i ($\lambda x. \text{frechet-derivative } f \text{ (at } x) h$)*
x h

lemma *frechet-derivative-nth-derivative-commute*:
frechet-derivative ($\lambda x. \text{nth-derivative } i f x h$) (at x) h =
nth-derivative i ($\lambda x. \text{frechet-derivative } f \text{ (at } x) h$) x h
 ⟨*proof*⟩

lemma *nth-derivative-funpow*:
nth-derivative i f x h = (($\lambda f x. \text{frechet-derivative } f \text{ (at } x) h$) $\hat{\sim}$ i) f x
 ⟨*proof*⟩

lemma *nth-derivative-exists*:
 $\exists f'. ((\lambda x. \text{nth-derivative } i f x h) \text{ has-derivative } f') \text{ (at } x) \wedge$
 $f' h = \text{nth-derivative } (Suc\ i) f x h$
if *higher-differentiable-on X f (Suc i) open X x \in X*
 ⟨*proof*⟩

lemma *higher-derivatives-exists*:
assumes *higher-differentiable-on X f n open X*
obtains *Df* **where**
 $\bigwedge a h. Df\ a\ 0\ h = f\ a$
 $\bigwedge a h\ i. i < n \implies a \in X \implies ((\lambda a. Df\ a\ i\ H) \text{ has-derivative } Df\ a\ (Suc\ i)) \text{ (at } a)$
 $\bigwedge a\ i. i \leq n \implies a \in X \implies Df\ a\ i\ H = \text{nth-derivative } i f a H$
 ⟨*proof*⟩

lemma *nth-derivative-differentiable:*

assumes *higher-differentiable-on S f (Suc n) x ∈ S*
shows $(\lambda x. \text{nth-derivative } n \text{ f } x \text{ v})$ *differentiable at x*
<proof>

lemma *higher-differentiable-on-imp-continuous-nth-derivative:*

assumes *higher-differentiable-on S f n*
shows *continuous-on S* $(\lambda x. \text{nth-derivative } n \text{ f } x \text{ v})$
<proof>

lemma *frechet-derivative-at-real-eq-scaleR:*

*frechet-derivative f (at x) v = v *_R frechet-derivative f (at x) 1*
if *f differentiable (at x) NO-MATCH 1 v*
<proof>

lemma *higher-differentiable-on-real-Suc:*

higher-differentiable-on S f (Suc n) ↔
 $(\forall x \in S. f \text{ differentiable (at } x)) \wedge$
 $(\text{higher-differentiable-on } S (\lambda x. \text{frechet-derivative } f \text{ (at } x) 1) \text{ n})$
if *open S*
for *S::real set*
<proof>

lemma *higher-differentiable-on-real-SucI:*

fixes *S::real set*
assumes
 $\lambda x. x \in S \implies (\lambda x. \text{nth-derivative } n \text{ f } x \text{ 1})$ *differentiable at x*
continuous-on S $(\lambda x. \text{nth-derivative (Suc } n) \text{ f } x \text{ 1})$
higher-differentiable-on S f n
and *o: open S*
shows *higher-differentiable-on S f (Suc n)*
<proof>

lemma *higher-differentiable-on-real-Suc':*

open S ⟹ higher-differentiable-on S f (Suc n) ↔
 $(\forall v. \text{continuous-on } S (\lambda x. \text{nth-derivative (Suc } n) \text{ f } x \text{ 1})) \wedge$
 $(\forall x \in S. \forall v. (\lambda x. \text{nth-derivative } n \text{ f } x \text{ 1}) \text{ differentiable (at } x)) \wedge \text{higher-differentiable-on } S \text{ f } n$
for *S::real set*
<proof>

lemma *closed-segment-subsetD:*

$0 \leq x \implies x \leq 1 \implies (X + x *_{\mathbb{R}} H) \in S$
if *closed-segment X (X + H) ⊆ S*
<proof>

lemma *higher-differentiable-Taylor:*

fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{banach}$
and $H::'a$
and $Df::'a \Rightarrow \text{nat} \Rightarrow 'a \Rightarrow 'a \Rightarrow 'b$
assumes $n > 0$
assumes hd : higher-differentiable-on S f n open S
assumes cs : closed-segment X $(X + H) \subseteq S$
defines $i \equiv \lambda x. ((1 - x) \wedge^{(n - 1)} / \text{fact } (n - 1)) *_R \text{nth-derivative } n f (X + x *_R H) H$
shows $(i \text{ has-integral } f (X + H) - (\sum i < n. (1 / \text{fact } i) *_R \text{nth-derivative } i f X H)) \{0..1\}$ **(is ?th1)**
and $f (X + H) = (\sum i < n. (1 / \text{fact } i) *_R \text{nth-derivative } i f X H) + \text{integral } \{0..1\} i$ **(is ?th2)**
and i integrable-on $\{0..1\}$ **(is ?th3)**
 $\langle \text{proof} \rangle$

lemma *frechet-derivative-componentwise*:
 $\text{frechet-derivative } f \text{ (at } a) v = (\sum i \in \text{Basis}. (v \cdot i) * (\text{frechet-derivative } f \text{ (at } a) i))$
if f differentiable at a
for $f::'a::\text{euclidean-space} \Rightarrow \text{real}$
 $\langle \text{proof} \rangle$

lemma *second-derivative-componentwise*:
 $\text{nth-derivative } 2 f a v = (\sum i \in \text{Basis}. (\sum j \in \text{Basis}. \text{frechet-derivative } (\lambda a. \text{frechet-derivative } f \text{ (at } a) j) \text{ (at } a) i * (v \cdot j)) * (v \cdot i))$
if higher-differentiable-on S f 2 **and** S : open S $a \in S$
for $f::'a::\text{euclidean-space} \Rightarrow \text{real}$
 $\langle \text{proof} \rangle$

lemma *higher-differentiable-Taylor1*:
fixes $f::'a::\text{real-normed-vector} \Rightarrow 'b::\text{banach}$
assumes hd : higher-differentiable-on S f 2 open S
assumes cs : closed-segment X $(X + H) \subseteq S$
defines $i \equiv \lambda x. ((1 - x) *_R \text{nth-derivative } 2 f (X + x *_R H) H$
shows $(i \text{ has-integral } f (X + H) - (f X + \text{nth-derivative } 1 f X H)) \{0..1\}$
and $f (X + H) = f X + \text{nth-derivative } 1 f X H + \text{integral } \{0..1\} i$
and i integrable-on $\{0..1\}$
 $\langle \text{proof} \rangle$

lemma *differentiable-on-open-blinfunE*:
assumes f differentiable-on S open S
obtains f' **where** $\bigwedge x. x \in S \implies (f \text{ has-derivative } \text{blinfun-apply } (f' x)) \text{ (at } x)$
 $\langle \text{proof} \rangle$

lemma *continuous-on-blinfunI1*:
 $\text{continuous-on } X f$
if $\bigwedge i. i \in \text{Basis} \implies \text{continuous-on } X (\lambda x. \text{blinfun-apply } (f x) i)$
 $\langle \text{proof} \rangle$

lemma *c1-euclidean-blinfunE*:

fixes $f::'a::\text{euclidean-space}\Rightarrow'b::\text{real-normed-vector}$

assumes $\bigwedge x. x \in S \implies (f \text{ has-derivative } f' x) \text{ (at } x \text{ within } S)$

assumes $\bigwedge i. i \in \text{Basis} \implies \text{continuous-on } S (\lambda x. f' x i)$

obtains bf' **where**

$\bigwedge x. x \in S \implies (f \text{ has-derivative } \text{blinfun-apply } (bf' x)) \text{ (at } x \text{ within } S)$
 $\text{continuous-on } S \text{ } bf'$

$\bigwedge x. x \in S \implies \text{blinfun-apply } (bf' x) = f' x$

<proof>

lemma *continuous-Sigma*:

assumes *defined*: $y \in \text{Pi } T \ X$

assumes *f-cont*: $\text{continuous-on } (\text{Sigma } T \ X) (\lambda(t, x). f t x)$

assumes *y-cont*: $\text{continuous-on } T \ y$

shows $\text{continuous-on } T (\lambda x. f x (y x))$

<proof>

lemma *continuous-on-Times-swap*:

$\text{continuous-on } (X \times Y) (\lambda(x, y). f x y)$

if $\text{continuous-on } (Y \times X) (\lambda(y, x). f x y)$

<proof>

lemma *leibniz-rule'*:

$\bigwedge x. x \in S \implies$

$((\lambda x. \text{integral } (cbox \ a \ b) (f x)) \text{ has-derivative } (\lambda v. \text{integral } (cbox \ a \ b) (\lambda t. f x x t v)))$

$\text{(at } x \text{ within } S)$

$(\lambda x. \text{integral } (cbox \ a \ b) (f x)) \text{ differentiable-on } S$

if *convex* S

and *c1*: $\bigwedge t x. t \in cbox \ a \ b \implies x \in S \implies ((\lambda x. f x t) \text{ has-derivative } f x x t) \text{ (at } x \text{ within } S)$

$\bigwedge i. i \in \text{Basis} \implies \text{continuous-on } (S \times cbox \ a \ b) (\lambda(x, t). f x x t i)$

and *i*: $\bigwedge x. x \in S \implies f x \text{ integrable-on } cbox \ a \ b$

for $S::'a::\text{euclidean-space set}$

and $f::'a \Rightarrow 'b::\text{euclidean-space} \Rightarrow 'c::\text{euclidean-space}$

<proof>

lemmas *leibniz-rule'-interval* = *leibniz-rule'*[**where** $'b=-::\text{ordered-euclidean-space}$,
unfolded cbox-interval]

lemma *leibniz-rule'-higher*:

$\text{higher-differentiable-on } S (\lambda x. \text{integral } (cbox \ a \ b) (f x)) \ k$

if *convex* S *open* S

and *c1*: $\text{higher-differentiable-on } (S \times cbox \ a \ b) (\lambda(x, t). f x x t) \ k$

— this condition is actually too strong: it would suffice if higher partial derivatives (w.r.t. x) are continuous w.r.t. t . but it makes the statement short and no need to introduce new constants

for $S::'a::\text{euclidean-space set}$

and $f::'a \Rightarrow 'b::\text{euclidean-space} \Rightarrow 'c::\text{euclidean-space}$
 $\langle \text{proof} \rangle$

lemmas *leibniz-rule'-higher-interval* = *leibniz-rule'-higher*[**where** $'b=-::\text{ordered-euclidean-space}$,
unfolded cbox-interval]

2.4 Smoothness

definition *k-smooth-on* :: $\text{enat} \Rightarrow 'a::\text{real-normed-vector set} \Rightarrow ('a \Rightarrow 'b::\text{real-normed-vector})$
 $\Rightarrow \text{bool}$

($--\text{smooth}'\text{-on [1000]$) **where**
smooth-on-def: $k\text{-smooth-on } S f = (\forall n \leq k. \text{higher-differentiable-on } S f n)$

abbreviation *smooth-on* $S f \equiv \infty\text{-smooth-on } S f$

lemma *derivative-is-smooth'*:

assumes $(k+1)\text{-smooth-on } S f$
shows $k\text{-smooth-on } S (\lambda x. \text{frechet-derivative } f \text{ (at } x) v)$
 $\langle \text{proof} \rangle$

lemma *derivative-is-smooth*: $\text{smooth-on } S f \Longrightarrow \text{smooth-on } S (\lambda x. \text{frechet-derivative } f \text{ (at } x) v)$

$\langle \text{proof} \rangle$

lemma *smooth-on-imp-continuous-on*: $\text{continuous-on } S f$ **if** $k\text{-smooth-on } S f$

$\langle \text{proof} \rangle$

lemma *smooth-on-imp-differentiable-on[simp]*: f *differentiable-on* S **if** $k\text{-smooth-on } S f$ $k \neq 0$

$\langle \text{proof} \rangle$

lemma *smooth-on-cong*:

assumes $k\text{-smooth-on } S g$ *open* S
and $\bigwedge x. x \in S \Longrightarrow f x = g x$
shows $k\text{-smooth-on } S f$
 $\langle \text{proof} \rangle$

lemma *smooth-on-open-Un*:

$k\text{-smooth-on } S f \Longrightarrow k\text{-smooth-on } T f \Longrightarrow \text{open } S \Longrightarrow \text{open } T \Longrightarrow k\text{-smooth-on } (S \cup T) f$

$\langle \text{proof} \rangle$

lemma *smooth-on-open-subsetsI*:

$k\text{-smooth-on } S f$
if $\bigwedge x. x \in S \Longrightarrow \exists T. x \in T \wedge \text{open } T \wedge k\text{-smooth-on } T f$
 $\langle \text{proof} \rangle$

lemma *smooth-on-const[intro]*: $k\text{-smooth-on } S (\lambda x. c)$

$\langle \text{proof} \rangle$

lemma *smooth-on-id*[intro]: $k\text{-smooth-on } S (\lambda x. x)$
 ⟨proof⟩

lemma *smooth-on-add-fun*: $k\text{-smooth-on } S f \implies k\text{-smooth-on } S g \implies \text{open } S$
 $\implies k\text{-smooth-on } S (f + g)$
 ⟨proof⟩

lemmas *smooth-on-add* = *smooth-on-add-fun*[unfolded plus-fun-def]

lemma *smooth-on-sum*:
 $n\text{-smooth-on } S (\lambda x. \sum_{i \in F}. f i x)$
if $\bigwedge i. i \in F \implies \text{finite } F \implies n\text{-smooth-on } S (f i)$ *open S*
 ⟨proof⟩

lemma (in *bounded-bilinear*) *smooth-on*:
includes *no-matrix-mult*
assumes $k\text{-smooth-on } S f$ $k\text{-smooth-on } S g$ *open S*
shows $k\text{-smooth-on } S (\lambda x. (f x) ** (g x))$
 ⟨proof⟩

lemma *smooth-on-compose2*:
fixes $g :: \Rightarrow :: \text{euclidean-space}$
assumes $k\text{-smooth-on } T f$ $k\text{-smooth-on } S g$ *open U* *open T* $g ' U \subseteq T$ $U \subseteq S$
shows $k\text{-smooth-on } U (f o g)$
 ⟨proof⟩

lemma *smooth-on-compose*:
fixes $g :: \Rightarrow :: \text{euclidean-space}$
assumes $k\text{-smooth-on } T f$ $k\text{-smooth-on } S g$ *open S* *open T* $g ' S \subseteq T$
shows $k\text{-smooth-on } S (f o g)$
 ⟨proof⟩

lemma *smooth-on-subset*:
 $k\text{-smooth-on } S f$
if $k\text{-smooth-on } T f$ $S \subseteq T$
 ⟨proof⟩

context begin

private lemmas $s = \text{bounded-bilinear.smooth-on}$
lemmas *smooth-on-inner* = *bounded-bilinear-inner*[THEN s]
and *smooth-on-scaleR* = *bounded-bilinear-scaleR*[THEN s]
and *smooth-on-mult* = *bounded-bilinear-mult*[THEN s]
end

lemma *smooth-on-divide*: $k\text{-smooth-on } S f \implies k\text{-smooth-on } S g \implies \text{open } S \implies (\bigwedge x.$
 $x \in S \implies g x \neq 0) \implies$
 $k\text{-smooth-on } S (\lambda x. f x / g x)$
for $f :: \Rightarrow :: \text{real-normed-field}$

<proof>

lemma *smooth-on-scaleR-fun*: k -smooth-on S $g \implies$ open $S \implies k$ -smooth-on S
($c *_{\mathbb{R}} g$)
<proof>

lemma *smooth-on-uminus-fun*: k -smooth-on S $g \implies$ open $S \implies k$ -smooth-on S
($- g$)
<proof>

lemmas *smooth-on-uminus* = *smooth-on-uminus-fun*[*unfolded fun-Compl-def*]

lemma *smooth-on-minus-fun*: k -smooth-on S $f \implies k$ -smooth-on S $g \implies$ open S
 $\implies k$ -smooth-on S ($f - g$)
<proof>

lemmas *smooth-on-minus* = *smooth-on-minus-fun*[*unfolded fun-diff-def*]

lemma *smooth-on-times-fun*: k -smooth-on S $f \implies k$ -smooth-on S $g \implies$ open S
 $\implies k$ -smooth-on S ($f * g$)
for $f g :: \Rightarrow :: \text{real-normed-algebra}$
<proof>

lemma *smooth-on-le*:
 l -smooth-on S f
if k -smooth-on S f $l \leq k$
<proof>

lemma *smooth-on-inverse*: k -smooth-on S ($\lambda x. \text{inverse } (f x)$)
if k -smooth-on S f $0 \notin f' S$ open S
for $f :: \Rightarrow :: \text{real-normed-field}$
<proof>

lemma *smooth-on-norm*: k -smooth-on S ($\lambda x. \text{norm } (f x)$)
if k -smooth-on S f $0 \notin f' S$ open S
for $f :: \Rightarrow :: \text{real-inner}$
<proof>

lemma *smooth-on-sqrt*: k -smooth-on S ($\lambda x. \text{sqrt } (f x)$)
if k -smooth-on S f $0 \notin f' S$ open S
<proof>

lemma *smooth-on-frechet-derivative*:
 ∞ -smooth-on UNIV ($\lambda x. \text{frechet-derivative } f \text{ (at } x) v$)
if ∞ -smooth-on UNIV f
— TODO: generalize
<proof>

lemmas *smooth-on-frechet-derivivative-comp* = *smooth-on-compose2*[*OF smooth-on-frechet-derivative*,

unfolded o-def]

lemma *smooth-onD: higher-differentiable-on S f n* **if** *m-smooth-on S f* **enat** $n \leq m$
<proof>

lemma (**in** *bounded-linear*) *higher-differentiable-on: higher-differentiable-on S f n*
<proof>

lemma (**in** *bounded-linear*) *smooth-on: k-smooth-on S f*
<proof>

lemma *smooth-on-snd:*
k-smooth-on S ($\lambda x. \text{snd } (f x)$)
if *k-smooth-on S f* **open** *S*
<proof>

lemma *smooth-on-fst:*
k-smooth-on S ($\lambda x. \text{fst } (f x)$)
if *k-smooth-on S f* **open** *S*
<proof>

lemma *smooth-on-sin: n-smooth-on S* ($\lambda x. \text{sin } (f x::\text{real})$) **if** *n-smooth-on S f* **open** *S*
<proof>

lemma *smooth-on-cos: n-smooth-on S* ($\lambda x. \text{cos } (f x::\text{real})$) **if** *n-smooth-on S f* **open** *S*
<proof>

lemma *smooth-on-Taylor2E:*
fixes $f::'a::\text{euclidean-space} \Rightarrow \text{real}$
assumes *hd: ∞ -smooth-on UNIV f*
obtains *g* **where** $\bigwedge Y. f Y = f X + \text{frechet-derivative } f \text{ (at } X) (Y - X) + (\sum i \in \text{Basis}. (\sum j \in \text{Basis}. ((Y - X) \cdot j) * ((Y - X) \cdot i) * g i j Y))$
 $\bigwedge i j. i \in \text{Basis} \implies j \in \text{Basis} \implies \infty\text{-smooth-on UNIV } (g i j)$
— **TODO:** generalize
<proof>

lemma *smooth-on-Pair:*
k-smooth-on S ($\lambda x. (f x, g x)$)
if *open S* *k-smooth-on S f* *k-smooth-on S g*
<proof>

lemma *smooth-on-Pair':*
k-smooth-on (S \times T) ($\lambda x. (f (\text{fst } x), g (\text{snd } x))$)

if $\text{open } S \text{ open } T \text{ } k\text{-smooth-on } S \text{ } f \text{ } k\text{-smooth-on } T \text{ } g$
for $f::\text{-}::\text{euclidean-space}\Rightarrow\text{-}$ **and** $g::\text{-}::\text{euclidean-space}\Rightarrow\text{-}$
 <proof>

2.5 Diffeomorphism

definition $\text{diffeomorphism } k \text{ } S \text{ } T \text{ } p \text{ } p' \longleftrightarrow$
 $k\text{-smooth-on } S \text{ } p \wedge k\text{-smooth-on } T \text{ } p' \wedge \text{homeomorphism } S \text{ } T \text{ } p \text{ } p'$

lemma $\text{diffeomorphism-imp-homeomorphism}$:

assumes $\text{diffeomorphism } k \text{ } s \text{ } t \text{ } p \text{ } p'$
shows $\text{homeomorphism } s \text{ } t \text{ } p \text{ } p'$
 <proof>

lemma diffeomorphismD :

assumes $\text{diffeomorphism } k \text{ } S \text{ } T \text{ } f \text{ } g$
shows $\text{diffeomorphism-smoothD}$: $k\text{-smooth-on } S \text{ } f \text{ } k\text{-smooth-on } T \text{ } g$
and $\text{diffeomorphism-inverseD}$: $\bigwedge x. x \in S \Longrightarrow g (f x) = x \wedge y. y \in T \Longrightarrow f (g y) = y$
and $\text{diffeomorphism-image-eq}$: $(f \text{ ` } S = T) (g \text{ ` } T = S)$
 <proof>

lemma $\text{diffeomorphism-compose}$:

$\text{diffeomorphism } n \text{ } S \text{ } T \text{ } f \text{ } g \Longrightarrow \text{diffeomorphism } n \text{ } T \text{ } U \text{ } h \text{ } k \Longrightarrow \text{open } S \Longrightarrow \text{open } T$
 $\Longrightarrow \text{open } U \Longrightarrow$
 $\text{diffeomorphism } n \text{ } S \text{ } U (h \circ f) (g \circ k)$
for $f::\text{-}\Rightarrow\text{-}::\text{euclidean-space}$
 <proof>

lemma $\text{diffeomorphism-add}$: $\text{diffeomorphism } k \text{ } UNIV \text{ } UNIV (\lambda x. x + c) (\lambda x. x - c)$
 <proof>

lemma $\text{diffeomorphism-scaleR}$: $\text{diffeomorphism } k \text{ } UNIV \text{ } UNIV (\lambda x. c *_{\mathbb{R}} x) (\lambda x. x /_{\mathbb{R}} c)$
if $c \neq 0$
 <proof>

end

3 Bump Functions

theory Bump-Function

imports Smooth

$\text{HOL-Analysis.Weierstrass-Theorems}$

begin

3.1 Construction

context begin

qualified definition $f :: \text{real} \Rightarrow \text{real}$ **where**
 $f\ t = (\text{if } t > 0 \text{ then } \text{exp}(-\text{inverse } t) \text{ else } 0)$

lemma $f\text{-nonpos}[simp]: x \leq 0 \implies f\ x = 0$
 $\langle \text{proof} \rangle$

lemma $\text{exp-inv-limit-0-right}$:
 $((\lambda(t::\text{real}). \text{exp}(-\text{inverse } t)) \longrightarrow 0) (\text{at-right } 0)$
 $\langle \text{proof} \rangle$

lemma $\forall_F\ t$ in $\text{at-right } 0$. $((\lambda x. \text{inverse } (x \wedge \text{Suc } k)) \text{ has-real-derivative}$
 $- (\text{inverse } (t \wedge \text{Suc } k) * ((1 + \text{real } k) * t \wedge k) * \text{inverse } (t \wedge \text{Suc } k))) (\text{at } t)$
 $\langle \text{proof} \rangle$

lemma $\text{exp-inv-limit-0-right-gen}'$:
 $((\lambda(t::\text{real}). \text{inverse } (t \wedge k) / \text{exp}(\text{inverse } t)) \longrightarrow 0) (\text{at-right } 0)$
 $\langle \text{proof} \rangle$

lemma $\text{exp-inv-limit-0-right-gen}$:
 $((\lambda(t::\text{real}). \text{exp}(-\text{inverse } t) / t \wedge k) \longrightarrow 0) (\text{at-right } 0)$
 $\langle \text{proof} \rangle$

lemma $f\text{-limit-0-right}: (f \longrightarrow 0) (\text{at-right } 0)$
 $\langle \text{proof} \rangle$

lemma $f\text{-limit-0}: (f \longrightarrow 0) (\text{at } 0)$
 $\langle \text{proof} \rangle$

lemma $f\text{-tendsto}: (f \longrightarrow f\ x) (\text{at } x)$
 $\langle \text{proof} \rangle$

lemma $f\text{-continuous}: \text{continuous-on } S\ f$
 $\langle \text{proof} \rangle$

lemma $\text{continuous-on-real-polynomial-function}$:
 $\text{continuous-on } S\ p$ **if** $\text{real-polynomial-function } p$
 $\langle \text{proof} \rangle$

lemma $f\text{-nth-derivative-is-poly}$:
 $\text{higher-differentiable-on } \{0<..\} f\ k \wedge$
 $(\exists p. \text{real-polynomial-function } p \wedge (\forall t>0. \text{nth-derivative } k\ f\ t\ 1 = p\ t / (t \wedge (2$
 $* k)) * \text{exp}(-\text{inverse } t)))$
 $\langle \text{proof} \rangle$

lemma $f\text{-has-derivative-at-neg}$:
 $x < 0 \implies (f \text{ has-derivative } (\lambda x. 0)) (\text{at } x)$

<proof>

lemma *f-differentiable-at-neg:*

$x < 0 \implies f$ differentiable at x

<proof>

lemma *frechet-derivative-f-at-neg:*

$x \in \{..<0\} \implies \text{frechet-derivative } f \text{ (at } x) = (\lambda x. 0)$

<proof>

lemma *f-nth-derivative-lt-0:*

higher-differentiable-on $\{..<0\}$ f $k \wedge (\forall t < 0. \text{nth-derivative } k \text{ } f \text{ } t \text{ } 1 = 0)$

<proof>

lemma *netlimit-at-left:* netlimit (at-left x) = x for $x::\text{real}$

<proof>

lemma *netlimit-at-right:* netlimit (at-right x) = x for $x::\text{real}$

<proof>

lemma *has-derivative-split-at:*

(g has-derivative g') (at x)

if

(g has-derivative g') (at-left x)

(g has-derivative g') (at-right x)

for $x::\text{real}$

<proof>

lemma *has-derivative-at-left-at-right':*

(g has-derivative g') (at x)

if

(g has-derivative g') (at x within $\{..x\}$)

(g has-derivative g') (at x within $\{x..\}$)

for $x::\text{real}$

<proof>

lemma *real-polynomial-function-tendsto:*

($p \longrightarrow p$ x) (at x within X) **if** real-polynomial-function p

<proof>

lemma *f-nth-derivative-cases:*

higher-differentiable-on UNIV f $k \wedge$

$(\forall t \leq 0. \text{nth-derivative } k \text{ } f \text{ } t \text{ } 1 = 0) \wedge$

$(\exists p. \text{real-polynomial-function } p \wedge$

$(\forall t > 0. \text{nth-derivative } k \text{ } f \text{ } t \text{ } 1 = p \text{ } t / (t \wedge (2 * k)) * \text{exp}(-\text{inverse } t))$)

<proof>

lemma *f-smooth-on:* k -smooth-on S f

and *f-higher-differentiable-on*: *higher-differentiable-on* S f n
<proof>

lemma *f-compose-smooth-on*: *k-smooth-on* S $(\lambda x. f (g x))$
if *k-smooth-on* S g *open* S
<proof>

lemma *f-nonneg*: $f x \geq 0$
<proof>

lemma *f-pos-iff*: $f x > 0 \longleftrightarrow x > 0$
<proof>

lemma *f-eq-zero-iff*: $f x = 0 \longleftrightarrow x \leq 0$
<proof>

3.2 Cutoff function

definition $h t = f (2 - t) / (f (2 - t) + f (t - 1))$

lemma *denominator-pos*: $f (2 - t) + f (t - 1) > 0$
<proof>

lemma *denominator-nonzero*: $f (2 - t) + f (t - 1) = 0 \longleftrightarrow \text{False}$
<proof>

lemma *h-range*: $0 \leq h t \leq 1$
<proof>

lemma *h-pos*: $t < 2 \implies 0 < h t$
and *h-less-one*: $1 < t \implies h t < 1$
<proof>

lemma *h-eq-0*: $h t = 0$ **if** $t \geq 2$
<proof>

lemma *h-eq-1*: $h t = 1$ **if** $t \leq 1$
<proof>

lemma *h-compose-smooth-on*: *k-smooth-on* S $(\lambda x. h (g x))$
if *k-smooth-on* S g *open* S
<proof>

3.3 Bump function

definition $H :: \text{real} \text{inner} \Rightarrow \text{real}$ **where** $H x = h (\text{norm } x)$

lemma *H-range*: $0 \leq H x \leq 1$
<proof>

lemma *H-eq-one*: $H x = 1$ **if** $x \in \text{cball } 0 \ 1$
 ⟨proof⟩

lemma *H-pos*: $H x > 0$ **if** $x \in \text{ball } 0 \ 2$
 ⟨proof⟩

lemma *H-eq-zero*: $H x = 0$ **if** $x \notin \text{ball } 0 \ 2$
 ⟨proof⟩

lemma *H-neq-zeroD*: $H x \neq 0 \implies x \in \text{ball } 0 \ 2$
 ⟨proof⟩

lemma *H-smooth-on*: $k\text{-smooth-on UNIV } H$
 ⟨proof⟩

lemma *H-compose-smooth-on*: $k\text{-smooth-on } S \ (\lambda x. H (g x))$ **if** $k\text{-smooth-on } S \ g$
open S
for $g :: - \Rightarrow \text{::euclidean-space}$
 ⟨proof⟩

end

end

4 Charts

theory *Chart*
imports *Analysis-More*
begin

4.1 Definition

A chart on M is a homeomorphism from an open subset of M to an open subset of some Euclidean space E . Here d and d' are open subsets of M and E , respectively, $f: d \rightarrow d'$ is the mapping, and $f': d' \rightarrow d$ is the inverse mapping.

typedef (**overloaded**) $('a::\text{topological-space}, 'e::\text{euclidean-space}) \text{ chart} =$
 $\{(d::'a \text{ set}, d'::'e \text{ set}, f, f') .$
 $\text{open } d \wedge \text{open } d' \wedge \text{homeomorphism } d \ d' \ f \ f'\}$
 ⟨proof⟩

setup-lifting *type-definition-chart*

lift-definition *apply-chart*:: $('a::\text{topological-space}, 'e::\text{euclidean-space}) \text{ chart} \Rightarrow 'a$
 $\Rightarrow 'e$
is $\lambda(d, d', f, f'). f$ ⟨proof⟩

declare $[[\text{coercion } \text{apply-chart}]]$

lift-definition *inv-chart*::('a::topological-space, 'e::euclidean-space) *chart* \Rightarrow 'e \Rightarrow 'a
is $\lambda(d, d', f, f'). f'$ *<proof>*

lift-definition *domain*::('a::topological-space, 'e::euclidean-space) *chart* \Rightarrow 'a *set*
is $\lambda(d, d', f, f'). d$ *<proof>*

lift-definition *codomain*::('a::topological-space, 'e::euclidean-space) *chart* \Rightarrow 'e *set*
is $\lambda(d, d', f, f'). d'$ *<proof>*

4.2 Properties

lemma *open-domain*[*intro, simp*]: *open* (*domain c*)
and *open-codomain*[*intro, simp*]: *open* (*codomain c*)
and *chart-homeomorphism*: *homeomorphism* (*domain c*) (*codomain c*) *c* (*inv-chart c*)
<proof>

lemma *at-within-domain*: *at x within domain c* = *at x* **if** $x \in \text{domain } c$
<proof>

lemma *at-within-codomain*: *at x within codomain c* = *at x* **if** $x \in \text{codomain } c$
<proof>

lemma
chart-in-codomain[*intro, simp*]: $x \in \text{domain } c \implies c \ x \in \text{codomain } c$
and *inv-chart-inverse*[*simp*]: $x \in \text{domain } c \implies \text{inv-chart } c \ (c \ x) = x$
and *inv-chart-in-domain*[*intro, simp*]: $y \in \text{codomain } c \implies \text{inv-chart } c \ y \in \text{domain } c$
and *chart-inverse-inv-chart*[*simp*]: $y \in \text{codomain } c \implies c \ (\text{inv-chart } c \ y) = y$
and *image-domain-eq*: $c \ ' (\text{domain } c) = \text{codomain } c$
and *inv-image-codomain-eq*[*simp*]: $\text{inv-chart } c \ ' (\text{codomain } c) = \text{domain } c$
and *continuous-on-domain*: *continuous-on* (*domain c*) *c*
and *continuous-on-codomain*: *continuous-on* (*codomain c*) (*inv-chart c*)
<proof>

lemma *chart-eqI*: $c = d$
if *domain c* = *domain d*
codomain c = *codomain d*
 $\bigwedge x. c \ x = d \ x$
 $\bigwedge x. \text{inv-chart } c \ x = \text{inv-chart } d \ x$
<proof>

lemmas *continuous-on-chart*[*continuous-intros*] =
continuous-on-compose2[*OF continuous-on-domain*]
continuous-on-compose2[*OF continuous-on-codomain*]

lemma *continuous-apply-chart*: *continuous* (*at x within X*) *c* **if** $x \in \text{domain } c$

<proof>

lemma *continuous-inv-chart*: *continuous (at x within X) (inv-chart c) if x ∈ codomain c*

<proof>

lemmas *apply-chart-tendsto*[*tendsto-intros*] = *isCont-tendsto-compose*[*OF continuous-apply-chart, rotated*]

lemmas *inv-chart-tendsto*[*tendsto-intros*] = *isCont-tendsto-compose*[*OF continuous-inv-chart, rotated*]

lemma *continuous-within-compose2'*:

continuous (at (f x) within t) g ⇒ f ' s ⊆ t ⇒

continuous (at x within s) f ⇒

continuous (at x within s) (λx. g (f x))

<proof>

lemmas *continuous-chart*[*continuous-intros*] =

continuous-within-compose2'[*OF continuous-apply-chart*]

continuous-within-compose2'[*OF continuous-inv-chart*]

lemma *continuous-on-chart-inv*:

assumes *continuous-on s (apply-chart c o f)*

f ' s ⊆ domain c

shows *continuous-on s f*

<proof>

lemma *continuous-on-chart-inv'*:

assumes *continuous-on (apply-chart c ' s) (f o inv-chart c)*

s ⊆ domain c

shows *continuous-on s f*

<proof>

lemma *inj-on-apply-chart*: *inj-on (apply-chart f) (domain f)*

<proof>

lemma *apply-chart-Int*: *f ' (X ∩ Y) = f ' X ∩ f ' Y if X ⊆ domain f Y ⊆ domain f*

<proof>

lemma *chart-image-eq-vimage*: *c ' X = inv-chart c -' X ∩ codomain c*

if *X ⊆ domain c*

<proof>

lemma *open-chart-image*[*simp, intro*]: *open (c ' X)*

if *open X X ⊆ domain c*

<proof>

lemma *open-inv-chart-image*[*simp, intro*]: *open (inv-chart c ' X)*

if $\text{open } X \ X \subseteq \text{codomain } c$
 ⟨proof⟩

lemma *homeomorphism-UNIV-imp-open-map*:
 $\text{homeomorphism UNIV UNIV } p \ p' \implies \text{open } f' \implies \text{open } (p \text{ ' } f')$
 ⟨proof⟩

4.3 Restriction

lemma *homeomorphism-restrict*:
 $\text{homeomorphism } (a \cap s) (b \cap f' - ' s) \ f \ f' \ \text{if } \text{homeomorphism } a \ b \ f \ f'$
 ⟨proof⟩

lift-definition *restrict-chart::'a set \Rightarrow ('a::t2-space, 'e::euclidean-space) chart \Rightarrow ('a, 'e) chart*
is $\lambda S. \lambda(d, d', f, f'). \text{ if } \text{open } S \text{ then } (d \cap S, d' \cap f' - ' S, f, f') \text{ else } (\{\}, \{\}, f, f')$
 ⟨proof⟩

lemma *restrict-chart-restrict-chart*:
 $\text{restrict-chart } X \ (\text{restrict-chart } Y \ c) = \text{restrict-chart } (X \cap Y) \ c$
if $\text{open } X \ \text{open } Y$
 ⟨proof⟩

lemma *domain-restrict-chart[simp]*: $\text{open } S \implies \text{domain } (\text{restrict-chart } S \ c) = \text{domain } c \cap S$
and *domain-restrict-chart-if*: $\text{domain } (\text{restrict-chart } S \ c) = (\text{if } \text{open } S \text{ then } \text{domain } c \cap S \text{ else } \{\})$
and *codomain-restrict-chart[simp]*: $\text{open } S \implies \text{codomain } (\text{restrict-chart } S \ c) = \text{codomain } c \cap \text{inv-chart } c - ' S$
and *codomain-restrict-chart-if*: $\text{codomain } (\text{restrict-chart } S \ c) = (\text{if } \text{open } S \text{ then } \text{codomain } c \cap \text{inv-chart } c - ' S \text{ else } \{\})$
and *apply-chart-restrict-chart[simp]*: $\text{apply-chart } (\text{restrict-chart } S \ c) = \text{apply-chart } c$
and *inv-chart-restrict-chart[simp]*: $\text{inv-chart } (\text{restrict-chart } S \ c) = \text{inv-chart } c$
 ⟨proof⟩

4.4 Composition

lift-definition *compose-chart::('e \Rightarrow 'e) \Rightarrow ('e \Rightarrow 'e) \Rightarrow ('a::topological-space, 'e::euclidean-space) chart \Rightarrow ('a, 'e) chart*
is $\lambda p \ p'. \lambda(d, d', f, f'). \text{ if } \text{homeomorphism UNIV UNIV } p \ p' \text{ then } (d, p \text{ ' } d', p \ o \ f, f' \ o \ p') \text{ else } (\{\}, \{\}, f, f')$
 ⟨proof⟩

lemma *compose-chart-apply-chart[simp]*: $\text{apply-chart } (\text{compose-chart } p \ p' \ c) = p \ o \ \text{apply-chart } c$
and *compose-chart-inv-chart[simp]*: $\text{inv-chart } (\text{compose-chart } p \ p' \ c) = \text{inv-chart } c \ o \ p'$


```

and domain-compose-chart[simp]: domain (compose-chart p p' c) = domain c
and codomain-compose-chart[simp]: codomain (compose-chart p p' c) = p '
codomain c
if homeomorphism UNIV UNIV p p'
⟨proof⟩

end

```

5 Topological Manifolds

```

theory Topological-Manifold
imports Chart
begin

```

Definition of topological manifolds. Existence of locally finite cover.

5.1 Defintition

We define topological manifolds as a second-countable Hausdorff space, where every point in the carrier set has a neighborhood that is homeomorphic to an open subset of the Euclidean space. Here topological manifolds are specified by a set of charts, and the carrier set is simply defined to be the union of the domain of the charts.

```

locale manifold =
fixes charts::('a::{second-countable-topology, t2-space}, 'e::euclidean-space) chart
set
begin

```

```

definition carrier = (⋃ (domain ' charts))

```

```

lemma open-carrier[intro, simp]: open carrier
⟨proof⟩

```

```

lemma carrierE:
assumes x ∈ carrier
obtains c where c ∈ charts x ∈ domain c
⟨proof⟩

```

```

lemma domain-subset-carrier[simp]: domain c ⊆ carrier if c ∈ charts
⟨proof⟩

```

```

lemma in-domain-in-carrier[intro, simp]: c ∈ charts ⇒ x ∈ domain c ⇒ x ∈
carrier
⟨proof⟩

```

5.2 Existence of locally finite cover

Every point has a precompact neighborhood.

lemma *precompact-neighborhoodE*:
assumes $x \in \text{carrier}$
obtains C **where** $x \in C$ *open* C *compact* (*closure* C) $\text{closure } C \subseteq \text{carrier}$
 $\langle \text{proof} \rangle$

There exists a covering of the carrier by precompact sets.

lemma *precompact-open-coverE*:
obtains $U::\text{nat} \Rightarrow 'a$ *set*
where $(\bigcup i. U\ i) = \text{carrier} \wedge i. \text{open } (U\ i) \wedge i. \text{compact } (\text{closure } (U\ i))$
 $\wedge i. \text{closure } (U\ i) \subseteq \text{carrier}$
 $\langle \text{proof} \rangle$

There exists a locally finite covering of the carrier by precompact sets.

lemma *precompact-locally-finite-open-coverE*:
obtains $W::\text{nat} \Rightarrow 'a$ *set*
where $\text{carrier} = (\bigcup i. W\ i) \wedge i. \text{open } (W\ i) \wedge i. \text{compact } (\text{closure } (W\ i))$
 $\wedge i. \text{closure } (W\ i) \subseteq \text{carrier}$
locally-finite-on carrier UNIV W
 $\langle \text{proof} \rangle$

end

end

6 Differentiable/Smooth Manifolds

theory *Differentiable-Manifold*

imports

Smooth

Topological-Manifold

begin

6.1 Smooth compatibility

definition *smooth-compat::enat* $\Rightarrow ('a::\text{topological-space}, 'e::\text{euclidean-space}) \text{chart} \Rightarrow ('a,$
 $'e) \text{chart} \Rightarrow \text{bool}$

$(\text{--smooth}'\text{-compat } [1000])$

where

smooth-compat $k\ c1\ c2 \longleftrightarrow$

$(k\text{-smooth-on } (c1\ ' (\text{domain } c1 \cap \text{domain } c2)) (c2 \circ \text{inv-chart } c1) \wedge$

$k\text{-smooth-on } (c2\ ' (\text{domain } c1 \cap \text{domain } c2)) (c1 \circ \text{inv-chart } c2))$

lemma *smooth-compat-D1*:

$k\text{-smooth-on } (c1\ ' (\text{domain } c1 \cap \text{domain } c2)) (c2 \circ \text{inv-chart } c1)$

if $k\text{-smooth-compat } c1\ c2$

$\langle \text{proof} \rangle$

lemma *smooth-compat-D2*:

k-smooth-on (*c2* ' (*domain c1* \cap *domain c2*)) (*c1* \circ *inv-chart c2*)
if *k-smooth-compat c1 c2*
 <proof>

lemma *smooth-compat-refl*: *k-smooth-compat x x*
 <proof>

lemma *smooth-compat-commute*: *k-smooth-compat x y* \longleftrightarrow *k-smooth-compat y x*
 <proof>

lemma *smooth-compat-restrict-chartI*:
k-smooth-compat (restrict-chart S c) c'
if *k-smooth-compat c c'*
 <proof>

lemma *smooth-compat-restrict-chartI2*:
k-smooth-compat c' (restrict-chart S c)
if *k-smooth-compat c' c*
 <proof>

lemma *smooth-compat-restrict-chartD*:
domain c1 \subseteq *U* \implies *open U* \implies *k-smooth-compat c1 (restrict-chart U c2)* \implies
k-smooth-compat c1 c2
 <proof>

lemma *smooth-compat-restrict-chartD2*:
domain c1 \subseteq *U* \implies *open U* \implies *k-smooth-compat (restrict-chart U c2) c1* \implies
k-smooth-compat c2 c1
 <proof>

lemma *smooth-compat-le*:
l-smooth-compat c1 c2 **if** *k-smooth-compat c1 c2* $l \leq k$
 <proof>

6.2 C^k -Manifold

locale *c-manifold* = *manifold* +
fixes *k::enat*
assumes *pairwise-compat*: *c1* \in *charts* \implies *c2* \in *charts* \implies *k-smooth-compat c1 c2*
begin

6.2.1 Atlas

definition *atlas* :: ('a, 'b) *chart set* **where**
atlas = {*c*. *domain c* \subseteq *carrier* \wedge ($\forall c' \in$ *charts*. *k-smooth-compat c c'*)}

lemma *charts-subset-atlas*: *charts* \subseteq *atlas*
 <proof>

lemma *in-charts-in-atlas*[intro]: $x \in \text{charts} \implies x \in \text{atlas}$
 ⟨proof⟩

lemma *maximal-atlas*:

$c \in \text{atlas}$
if $\bigwedge c'. c' \in \text{atlas} \implies k\text{-smooth-compat } c \ c'$
 $\text{domain } c \subseteq \text{carrier}$
 ⟨proof⟩

lemma *chart-compose-lemma*:

fixes $c1 \ c2$
defines [*simp*]: $U \equiv \text{domain } c1$
defines [*simp*]: $V \equiv \text{domain } c2$
assumes *subsets*: $U \cap V \subseteq \text{carrier}$
assumes $\bigwedge c. c \in \text{charts} \implies k\text{-smooth-compat } c1 \ c$
 $\bigwedge c. c \in \text{charts} \implies k\text{-smooth-compat } c2 \ c$
shows $k\text{-smooth-on } (c1 \ ' (U \cap V)) \ (c2 \circ \text{inv-chart } c1)$
 ⟨proof⟩

lemma *smooth-compat-trans*: $k\text{-smooth-compat } c1 \ c2$

if $\bigwedge c. c \in \text{charts} \implies k\text{-smooth-compat } c1 \ c$
 $\bigwedge c. c \in \text{charts} \implies k\text{-smooth-compat } c2 \ c$
 $\text{domain } c1 \cap \text{domain } c2 \subseteq \text{carrier}$
 ⟨proof⟩

lemma *maximal-atlas'*:

$c \in \text{atlas}$
if $\bigwedge c'. c' \in \text{charts} \implies k\text{-smooth-compat } c \ c'$
 $\text{domain } c \subseteq \text{carrier}$
 ⟨proof⟩

lemma *atlas-is-atlas*: $k\text{-smooth-compat } a1 \ a2$

if $a1 \in \text{atlas} \ a2 \in \text{atlas}$
 ⟨proof⟩

lemma *domain-atlas-subset-carrier*: $c \in \text{atlas} \implies \text{domain } c \subseteq \text{carrier}$

and *in-carrier-atlasI*[intro, simp]: $c \in \text{atlas} \implies x \in \text{domain } c \implies x \in \text{carrier}$
 ⟨proof⟩

lemma *atlasE*:

assumes $x \in \text{carrier}$
obtains c **where** $c \in \text{atlas} \ x \in \text{domain } c$
 ⟨proof⟩

lemma *restrict-chart-in-atlas*: $\text{restrict-chart } S \ c \in \text{atlas}$ **if** $c \in \text{atlas}$

⟨proof⟩

lemma *atlas-restrictE*:

assumes $x \in \text{carrier } x \in X \text{ open } X$
obtains c **where** $c \in \text{atlas } x \in \text{domain } c \text{ domain } c \subseteq X$
 $\langle \text{proof} \rangle$

lemma *open-ball-chartE*:

assumes $x \in U \text{ open } U \subseteq \text{carrier}$
obtains $c \ r$ **where**
 $c \in \text{atlas}$
 $x \in \text{domain } c \text{ domain } c \subseteq U \text{ codomain } c = \text{ball } (c \ x) \ r \ r > 0$
 $\langle \text{proof} \rangle$

lemma *smooth-compat-compose-chart*:

fixes c'
assumes $k\text{-smooth-compat } c \ c'$
assumes $\text{diffeo: diffeomorphism } k \ \text{UNIV } \text{UNIV } p \ p'$
shows $k\text{-smooth-compat } (\text{compose-chart } p \ p' \ c) \ c'$
 $\langle \text{proof} \rangle$

lemma *compose-chart-in-atlas*:

assumes $c \in \text{atlas}$
assumes $\text{diffeo: diffeomorphism } k \ \text{UNIV } \text{UNIV } p \ p'$
shows $\text{compose-chart } p \ p' \ c \in \text{atlas}$
 $\langle \text{proof} \rangle$

lemma *open-centered-ball-chartE*:

assumes $x \in U \text{ open } U \subseteq \text{carrier } e > 0$
obtains c **where**
 $c \in \text{atlas } x \in \text{domain } c \ c \ x = x0 \text{ domain } c \subseteq U \text{ codomain } c = \text{ball } x0 \ e$
 $\langle \text{proof} \rangle$

end

6.2.2 Submanifold

definition (in *manifold*) $\text{charts-submanifold } S = (\text{restrict-chart } S \ \text{'charts})$

locale $c\text{-manifold}' = c\text{-manifold}$

locale $\text{submanifold} = c\text{-manifold}' \ \text{charts } k$ — breaks infinite loop for sublocale sub

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

and $k +$

fixes $S::'a \ \text{set}$

assumes $\text{open-submanifold: open } S$

begin

lemma $\text{charts-submanifold: } c\text{-manifold } (\text{charts-submanifold } S) \ k$

$\langle \text{proof} \rangle$

sublocale *sub*: *c-manifold* (*charts-submanifold* *S*) *k*
⟨*proof*⟩

lemma *carrier-submanifold[simp]*: *sub.carrier* = *S* ∩ *carrier*
⟨*proof*⟩

lemma *restrict-chart-carrier[simp]*:
restrict-chart carrier x = x
if *x* ∈ *charts*
⟨*proof*⟩

lemma *charts-submanifold-carrier[simp]*: *charts-submanifold carrier* = *charts*
⟨*proof*⟩

lemma *charts-submanifold-Int-carrier*:
charts-submanifold (S ∩ *carrier)* = *charts-submanifold S*
⟨*proof*⟩

lemma *submanifold-atlasE*:
assumes *c* ∈ *sub.atlas*
shows *c* ∈ *atlas*
⟨*proof*⟩

lemma *submanifold-atlasI*:
restrict-chart S c ∈ *sub.atlas*
if *c* ∈ *atlas*
⟨*proof*⟩

end

lemma (**in** *c-manifold*) *restrict-chart-carrier[simp]*:
restrict-chart carrier x = x
if *x* ∈ *charts*
⟨*proof*⟩

lemma (**in** *c-manifold*) *charts-submanifold-carrier[simp]*: *charts-submanifold carrier* = *charts*
⟨*proof*⟩

6.3 Differentiable maps

locale *c-manifolds* =
src: *c-manifold charts1 k* +
dest: *c-manifold charts2 k* **for** *k charts1 charts2*

locale *diff* = *c-manifolds k charts1 charts2*
for *k*

```

and charts1 :: ('a::{t2-space,second-countable-topology}, 'e::euclidean-space)
chart set
and charts2 :: ('b::{t2-space,second-countable-topology}, 'f::euclidean-space)
chart set
+
fixes f :: ('a  $\Rightarrow$  'b)
assumes exists-smooth-on:  $x \in \text{src.carrier} \implies$ 
 $\exists c1 \in \text{src.atlas}. \exists c2 \in \text{dest.atlas}.$ 
 $x \in \text{domain } c1 \wedge$ 
 $f \text{ ' domain } c1 \subseteq \text{domain } c2 \wedge$ 
 $k\text{-smooth-on } (\text{codomain } c1) (c2 \circ f \circ \text{inv-chart } c1)$ 
begin

```

```

lemma defined:  $f \text{ ' src.carrier} \subseteq \text{dest.carrier}$ 
<proof>

```

end

context *c-manifolds* **begin**

```

lemma diff-iff:  $\text{diff } k \text{ charts1 charts2 } f \iff$ 
 $(\forall x \in \text{src.carrier}. \exists c1 \in \text{src.atlas}. \exists c2 \in \text{dest.atlas}.$ 
 $x \in \text{domain } c1 \wedge$ 
 $f \text{ ' domain } c1 \subseteq \text{domain } c2 \wedge$ 
 $k\text{-smooth-on } (\text{codomain } c1) (\text{apply-chart } c2 \circ f \circ \text{inv-chart } c1))$ 
(is ?l  $\iff (\forall x \in -. ?r x)$ )
<proof>

```

end

context *diff* **begin**

```

lemma diffE:
assumes  $x \in \text{src.carrier}$ 
obtains  $c1::('a, 'e) \text{ chart}$ 
and  $c2::('b, 'f) \text{ chart}$ 
where
 $c1 \in \text{src.atlas } c2 \in \text{dest.atlas } x \in \text{domain } c1 \text{ ' domain } c1 \subseteq \text{domain } c2$ 
 $k\text{-smooth-on } (\text{codomain } c1) (\text{apply-chart } c2 \circ f \circ \text{inv-chart } c1)$ 
<proof>

```

```

lemma continuous-at:  $\text{continuous } (\text{at } x \text{ within } T) f \text{ if } x \in \text{src.carrier}$ 
<proof>

```

```

lemma continuous-on:  $\text{continuous-on } \text{src.carrier } f$ 
<proof>

```

```

lemmas continuous-on-intro[continuous-intros] = continuous-on-compose2[OF con-
tinuous-on -]

```

lemmas *continuous-within*[*continuous-intros*] = *continuous-within-compose3*[*OF continuous-at*]

lemmas *tendsto*[*tendsto-intros*] = *isCont-tendsto-compose*[*OF continuous-at*]

lemma *diff-chartsD*:

assumes $d1 \in \text{src.atlas}$ $d2 \in \text{dest.atlas}$

shows $k\text{-smooth-on}$ ($\text{codomain } d1 \cap \text{inv-chart } d1 - ' (\text{src.carrier} \cap f - ' \text{domain } d2)$)

($\text{apply-chart } d2 \circ f \circ \text{inv-chart } d1$)

$\langle \text{proof} \rangle$

lemma *diff-between-chartsE*:

assumes $d1 \in \text{src.atlas}$ $d2 \in \text{dest.atlas}$

assumes $y \in \text{domain } d1$ $y \in \text{src.carrier}$ $f y \in \text{domain } d2$

obtains X **where**

$k\text{-smooth-on } X$ ($\text{apply-chart } d2 \circ f \circ \text{inv-chart } d1$)

$d1 y \in X$

$\text{open } X$

$X = \text{codomain } d1 \cap \text{inv-chart } d1 - ' (\text{src.carrier} \cap f - ' \text{domain } d2)$

$\langle \text{proof} \rangle$

end

lemma *diff-compose*:

$\text{diff } k M1 M3 (g \circ f)$

if $\text{diff } k M1 M2 f \text{ diff } k M2 M3 g$

$\langle \text{proof} \rangle$

context *diff* **begin**

lemma *diff-submanifold*: $\text{diff } k (\text{src.charts-submanifold } S) \text{ charts2 } f$

if $\text{open } S$

$\langle \text{proof} \rangle$

lemma *diff-submanifold2*: $\text{diff } k \text{ charts1 } (\text{dest.charts-submanifold } S) f$

if $\text{open } S f - ' \text{src.carrier} \subseteq S$

$\langle \text{proof} \rangle$

end

context *c-manifolds* **begin**

lemma *diff-localI*: $\text{diff } k \text{ charts1 } \text{ charts2 } f$

if $\bigwedge x. x \in \text{src.carrier} \implies \text{diff } k (\text{src.charts-submanifold } (U x)) \text{ charts2 } f$

$\bigwedge x. x \in \text{src.carrier} \implies \text{open } (U x)$

$\bigwedge x. x \in \text{src.carrier} \implies x \in (U x)$

$\langle \text{proof} \rangle$


```

lemma diff-open-coverI: diff k charts1 charts2 f
  if diff:  $\bigwedge u. u \in U \implies \text{diff } k \text{ (src.charts-submanifold } u) \text{ charts2 } f$ 
  and op:  $\bigwedge u. u \in U \implies \text{open } u$ 
  and cover:  $\text{src.carrier} \subseteq \bigcup U$ 
  <proof>

lemma diff-open-Un: diff k charts1 charts2 f
  if diff k (src.charts-submanifold U) charts2 f
  diff k (src.charts-submanifold V) charts2 f
  and open U open V src.carrier  $\subseteq U \cup V$ 
  <proof>

end

context c-manifold begin

sublocale self: c-manifolds k charts charts
  <proof>

lemma diff-id: diff k charts charts ( $\lambda x. x$ )
  <proof>

lemma c-manifold-order-le: c-manifold charts l if  $l \leq k$ 
  <proof>

lemma in-atlas-order-le:  $c \in \text{c-manifold.atlas charts } l$  if  $l \leq k$   $c \in \text{atlas}$ 
  <proof>

end

context c-manifolds begin

lemma c-manifolds-order-le: c-manifolds l charts1 charts2 if  $l \leq k$ 
  <proof>

end

context diff begin

lemma diff-order-le: diff l charts1 charts2 f if  $l \leq k$ 
  <proof>

end

```

6.4 Differentiable functions

lift-definition *chart-eucl*::(*'a*::*euclidean-space*, *'a*) *chart* is
 (*UNIV*, *UNIV*, $\lambda x. x$, $\lambda x. x$)

<proof>

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

lemma *chart-eucl-simps*[*simp*]:
 domain chart-eucl = *UNIV*
 codomain chart-eucl = *UNIV*
 apply-chart chart-eucl = ($\lambda x. x$)
 inv-chart chart-eucl = ($\lambda x. x$)
<proof>

locale *diff-fun* = *diff* *k charts charts-eucl f*
 for *k charts* **and** *f::'a::{t2-space,second-countable-topology}* \Rightarrow *'b::euclidean-space*

lemma *diff-fun-compose*:
 diff-fun k M1 (g o f)
 if *diff k M1 M2 f diff-fun k M2 g*
<proof>

lemma *c1-manifold-atlas-eucl*: *c-manifold charts-eucl k*
<proof>

interpretation *manifold-eucl*: *c-manifold charts-eucl k*
<proof>

lemma *chart-eucl-in-atlas*[*intro,simp*]: *chart-eucl* \in *manifold-eucl.atlas k*
<proof>

lemma *apply-chart-smooth-on*:
 k-smooth-on (domain c) c **if** *c* \in *manifold-eucl.atlas k*
<proof>

lemma *inv-chart-smooth-on*: *k-smooth-on (codomain c) (inv-chart c)* **if** *c* \in *manifold-eucl.atlas k*
<proof>

lemma *smooth-on-chart-inv*:
 fixes *c::('a::euclidean-space, 'a) chart*
 assumes *k-smooth-on X (apply-chart c o f)*
 assumes *continuous-on X f*
 assumes *c* \in *manifold-eucl.atlas k open X f ' X* \subseteq *domain c*
 shows *k-smooth-on X f*
<proof>

lemma *smooth-on-chart-inv2*:
 fixes *c::('a::euclidean-space, 'a) chart*
 assumes *k-smooth-on (c ' X) (f o inv-chart c)*
 assumes *c* \in *manifold-eucl.atlas k open X X* \subseteq *domain c*
 shows *k-smooth-on X f*

<proof>

context *diff-fun* **begin**

lemma *diff-fun-order-le*: *diff-fun* *l* *charts* *f* **if** $l \leq k$
<proof>

end

6.5 Diffeomorphism

locale *diffeomorphism* = *diff* *k* *charts1* *charts2* *f* + *inv*: *diff* *k* *charts2* *charts1* *f'*
for *k* *charts1* *charts2* *f* *f'* +
assumes *f-inv[simp]*: $\bigwedge x. x \in \text{src.carrier} \implies f' (f x) = x$
and *f'-inv[simp]*: $\bigwedge y. y \in \text{dest.carrier} \implies f (f' y) = y$

context *c-manifold* **begin**

sublocale *manifold-eucl*: *c-manifolds* *k* *charts* {*chart-eucl*}
rewrites *diff* *k* *charts* {*chart-eucl*} = *diff-fun* *k* *charts*
<proof>

lemma *diff-funI*:

diff-fun *k* *charts* *f*
if ($\bigwedge x. x \in \text{carrier} \implies \exists c1 \in \text{atlas}. x \in \text{domain } c1 \wedge (k\text{-smooth-on } (\text{codomain } c1) (f \circ \text{inv-chart } c1))$)
<proof>

end

lemma (**in** *diff*) *diff-cong*: *diff* *k* *charts1* *charts2* *g* **if** $\bigwedge x. x \in \text{src.carrier} \implies f x = g x$
<proof>

context *diff-fun* **begin**

lemma *diff-fun-cong*: *diff-fun* *k* *charts* *g* **if** $\bigwedge x. x \in \text{src.carrier} \implies f x = g x$
<proof>

lemma *diff-funD*:

$\exists c1 \in \text{src.atlas}. x \in \text{domain } c1 \wedge (k\text{-smooth-on } (\text{codomain } c1) (f \circ \text{inv-chart } c1))$
if $x: x \in \text{src.carrier}$
<proof>

lemma *diff-funE*:

assumes $x \in \text{src.carrier}$
obtains *c1* **where**

$c1 \in \text{src.atlas } x \in \text{domain } c1 \text{ } k\text{-smooth-on } (\text{codomain } c1) (f \circ \text{inv-chart } c1)$
 ⟨proof⟩

lemma *diff-fun-between-chartsD*:

assumes $c \in \text{src.atlas } x \in \text{domain } c$

shows $k\text{-smooth-on } (\text{codomain } c) (f \circ \text{inv-chart } c)$

⟨proof⟩

lemma *diff-fun-submanifold*: $\text{diff-fun } k \text{ } (\text{src.charts-submanifold } S) f$

if [simp]: $\text{open } S$

⟨proof⟩

end

context *c-manifold* **begin**

lemma *diff-fun-zero*: $\text{diff-fun } k \text{ charts } 0$

⟨proof⟩

lemma *diff-fun-const*: $\text{diff-fun } k \text{ charts } (\lambda x. c)$

⟨proof⟩

lemma *diff-fun-add*: $\text{diff-fun } k \text{ charts } (a + b)$ **if** $\text{diff-fun } k \text{ charts } a \text{ } \text{diff-fun } k \text{ charts } b$

⟨proof⟩

lemma *diff-fun-sum*: $\text{diff-fun } k \text{ charts } (\lambda x. \sum_{i \in S} f i x)$ **if** $\bigwedge i. i \in S \implies \text{diff-fun } k \text{ charts } (f i)$

⟨proof⟩

lemma *diff-fun-scaleR*: $\text{diff-fun } k \text{ charts } (\lambda x. a x *_R b x)$

if $\text{diff-fun } k \text{ charts } a \text{ } \text{diff-fun } k \text{ charts } b$

⟨proof⟩

lemma *diff-fun-scaleR-left*: $\text{diff-fun } k \text{ charts } (c *_R b)$

if $\text{diff-fun } k \text{ charts } b$

⟨proof⟩

lemma *diff-fun-times*: $\text{diff-fun } k \text{ charts } (a * b)$ **if** $\text{diff-fun } k \text{ charts } a \text{ } \text{diff-fun } k \text{ charts } b$

for $a \text{ } b :: \Rightarrow \text{::real-normed-algebra}$

⟨proof⟩

lemma *diff-fun-divide*: $\text{diff-fun } k \text{ charts } (\lambda x. a x / b x)$

if $\text{diff-fun } k \text{ charts } a \text{ } \text{diff-fun } k \text{ charts } b$

and $\text{nz}: \bigwedge x. x \in \text{carrier} \implies b x \neq 0$

for $a \text{ } b :: \Rightarrow \text{::real-normed-field}$

⟨proof⟩

lemma *subspace-Collect-diff-fun:*
subspace (Collect (diff-fun k charts))
 ⟨*proof*⟩

end

lemma *manifold-eucl-carrier[simp]: manifold-eucl.carrier = UNIV*
 ⟨*proof*⟩

lemma *diff-fun-charts-euclD: k-smooth-on UNIV g if diff-fun k charts-eucl g*
 ⟨*proof*⟩

lemma *diff-fun-charts-euclI: diff-fun k charts-eucl g if k-smooth-on UNIV g*
 ⟨*proof*⟩

end

7 Partitions Of Unity

theory *Partition-Of-Unity*
imports *Bump-Function Differentiable-Manifold*
begin

7.1 Regular cover

context *c-manifold* **begin**

A cover is regular if, in addition to being countable and locally finite, the codomain of every chart is the open ball of radius 3, such that the inverse image of open balls of radius 1 also cover the manifold.

definition *regular-cover I (ψ::'a ⇒ ('a, 'b) chart) ←→*
countable I ∧
carrier = (⋃ i ∈ I. domain (ψ i)) ∧
locally-finite-on carrier I (domain o ψ) ∧
(∀ i ∈ I. codomain (ψ i) = ball 0 3) ∧
carrier = (⋃ i ∈ I. inv-chart (ψ i) ' ball 0 1)

Every covering has a refinement that is a regular cover.

lemma *reguler-refinementE:*
fixes *ℳ::'a ⇒ 'a set*
assumes *cover: carrier ⊆ (⋃ i ∈ I. ℳ i) and open-cover: ∧ i. i ∈ I ⇒ open (ℳ i)*
obtains *N::nat set and ψ::nat ⇒ ('a, 'b) chart*
where *∧ i. i ∈ N ⇒ ψ i ∈ atlas (domain o ψ) ' N refines ℳ ' I regular-cover*
N ψ
 ⟨*proof*⟩

lemma *diff-apply-chart:*

diff k (charts-submanifold (domain ψ)) charts-eucl ψ if $\psi \in \text{atlas}$
 ⟨proof⟩

lemma *diff-inv-chart:*

diff k (manifold-eucl.charts-submanifold (codomain c)) charts (inv-chart c) if $c \in \text{atlas}$
 ⟨proof⟩

lemma *chart-inj-on [simp]:*

fixes $c :: ('a, 'b)$ chart
assumes $x \in \text{domain } c \ y \in \text{domain } c$
shows $c \ x = c \ y \longleftrightarrow x = y$
 ⟨proof⟩

7.2 Partition of unity by smooth functions

Given any open cover X indexed by a set A , there exists a family of smooth functions φ indexed by A , such that $0 \leq \varphi \leq 1$, the (closed) support of each $\varphi \ i$ is contained in $X \ i$, the supports are locally finite, and the sum of $\varphi \ i$ is the constant function 1.

theorem *partitions-of-unityE:*

fixes $A :: 'i$ set and $X :: 'i \Rightarrow 'a$ set
assumes $\text{carrier} \subseteq (\bigcup i \in A. X \ i)$
assumes $\bigwedge i. i \in A \Longrightarrow \text{open } (X \ i)$
obtains $\varphi :: 'i \Rightarrow 'a \Rightarrow \text{real}$
where $\bigwedge i. i \in A \Longrightarrow \text{diff-fun } k \text{ charts } (\varphi \ i)$
and $\bigwedge i \ x. i \in A \Longrightarrow x \in \text{carrier} \Longrightarrow 0 \leq \varphi \ i \ x$
and $\bigwedge i \ x. i \in A \Longrightarrow x \in \text{carrier} \Longrightarrow \varphi \ i \ x \leq 1$
and $\bigwedge x. x \in \text{carrier} \Longrightarrow (\sum i \in \{i \in A. \varphi \ i \ x \neq 0\}. \varphi \ i \ x) = 1$
and $\bigwedge i. i \in A \Longrightarrow \text{csupport-on carrier } (\varphi \ i) \cap \text{carrier} \subseteq X \ i$
and *locally-finite-on carrier* $A \ (\lambda i. \text{csupport-on carrier } (\varphi \ i))$
 ⟨proof⟩

Given $A \subseteq U \subseteq \text{carrier}$, where A is closed and U is open, there exists a differentiable function ψ such that $0 \leq \psi \leq 1$, $\psi = 1$ on A , and the support of ψ is contained in U .

lemma *smooth-bump-functionE:*

assumes *closedin (top-of-set carrier)* A
and $A \subseteq U \ U \subseteq \text{carrier}$ open U
obtains $\psi :: 'a \Rightarrow \text{real}$ **where**
diff-fun k charts ψ
 $\bigwedge x. x \in \text{carrier} \Longrightarrow 0 \leq \psi \ x$
 $\bigwedge x. x \in \text{carrier} \Longrightarrow \psi \ x \leq 1$
 $\bigwedge x. x \in A \Longrightarrow \psi \ x = 1$
csupport-on carrier $\psi \cap \text{carrier} \subseteq U$
 ⟨proof⟩

definition *diff-fun-on* $A \ f \longleftrightarrow$

($\exists W. A \subseteq W \wedge W \subseteq \text{carrier} \wedge \text{open } W \wedge$
 $(\exists f'. \text{diff-fun } k (\text{charts-submanifold } W) f' \wedge (\forall x \in A. f x = f' x))$)

lemma *diff-fun-onE*:

assumes *diff-fun-on* $A f$

obtains $W f'$ **where**

$A \subseteq W \ W \subseteq \text{carrier} \ \text{open } W \ \text{diff-fun } k (\text{charts-submanifold } W) f'$

$\bigwedge x. x \in A \implies f x = f' x$

<proof>

lemma *diff-fun-onI*:

assumes $A \subseteq W \ W \subseteq \text{carrier} \ \text{open } W \ \text{diff-fun } k (\text{charts-submanifold } W) f'$

$\bigwedge x. x \in A \implies f x = f' x$

shows *diff-fun-on* $A f$

<proof>

Extension lemma:

Given $A \subseteq U \subseteq \text{carrier}$, where A is closed and U is open, and a differentiable function f on A , there exists a differentiable function f' agreeing with f on A , and where the support of f' is contained in U .

lemma *extension-lemmaE*:

fixes $f::'a \Rightarrow 'e::\text{euclidean-space}$

assumes *closedin* (*top-of-set* *carrier*) A

assumes *diff-fun-on* $A f \ A \subseteq U \ U \subseteq \text{carrier} \ \text{open } U$

obtains f' **where**

diff-fun $k \ \text{charts} \ f'$

$\bigwedge x. x \in A \implies f' x = f x$

csupport-on carrier $f' \cap \text{carrier} \subseteq U$

<proof>

end

end

8 Tangent Space

theory *Tangent-Space*

imports *Partition-Of-Unity*

begin

lemma *linear-imp-linear-on*: *linear-on* $A \ B \ \text{scaleR} \ \text{scaleR} \ f$ **if** *linear* f

subspace $A \ \text{subspace } B$

<proof>

lemma (**in** *vector-space-pair-on*)

linear-sum':

$\forall x. x \in S1 \implies f x \in S2 \implies$

$\forall x. x \in S \implies g x \in S1 \implies$

$linear-on\ S1\ S2\ scale1\ scale2\ f \implies$
 $f\ (sum\ g\ S) = (\sum_{a \in S}. f\ (g\ a))$
 <proof>

8.1 Real vector (sub)spaces

locale *real-vector-space-on* = **fixes** S **assumes** *subspace*: *subspace* S
begin

sublocale *vector-space-on* S *scaleR*
rewrites *span-eq-real*: *local.span* = *real-vector.span*
and *dependent-eq-real*: *local.dependent* = *real-vector.dependent*
and *subspace-eq-real*: *local.subspace* = *real-vector.subspace*
 <proof>

lemma *dim-eq*: *local.dim* X = *real-vector.dim* X **if** $X \subseteq S$
 <proof>

end

locale *real-vector-space-pair-on* = *vs1*: *real-vector-space-on* S + *vs2*: *real-vector-space-on*
 T **for** $S\ T$
begin

sublocale *vector-space-pair-on* $S\ T$ *scaleR* *scaleR*
rewrites *span-eq-real1*: *module-on.span* *scaleR* = *vs1.span*
and *dependent-eq-real1*: *module-on.dependent* *scaleR* = *vs1.dependent*
and *subspace-eq-real1*: *module-on.subspace* *scaleR* = *vs1.subspace*
and *span-eq-real2*: *module-on.span* *scaleR* = *vs2.span*
and *dependent-eq-real2*: *module-on.dependent* *scaleR* = *vs2.dependent*
and *subspace-eq-real2*: *module-on.subspace* *scaleR* = *vs2.subspace*
 <proof>

end

locale *finite-dimensional-real-vector-space-on* = *real-vector-space-on* S **for** S +
fixes *basis* :: 'a *set*
assumes *finite-dimensional-basis*: *finite* *basis* \neg *dependent* *basis* *span* *basis* = S
basis $\subseteq S$
begin

sublocale *finite-dimensional-vector-space-on* S *scaleR* *basis*
rewrites *span-eq-real*: *local.span* = *real-vector.span*
and *dependent-eq-real*: *local.dependent* = *real-vector.dependent*
and *subspace-eq-real*: *local.subspace* = *real-vector.subspace*
 <proof>

end


```

locale finite-dimensional-real-vector-space-pair-1-on =
  vs1: finite-dimensional-real-vector-space-on S1 basis +
  vs2: real-vector-space-on S2
  for S1 S2 basis
begin

sublocale finite-dimensional-vector-space-pair-1-on S1 S2 scaleR scaleR basis
  rewrites span-eq-real1: module-on.span scaleR = vs1.span
    and dependent-eq-real1: module-on.dependent scaleR = vs1.dependent
    and subspace-eq-real1: module-on.subspace scaleR = vs1.subspace
    and span-eq-real2: module-on.span scaleR = vs2.span
    and dependent-eq-real2: module-on.dependent scaleR = vs2.dependent
    and subspace-eq-real2: module-on.subspace scaleR = vs2.subspace
  <proof>

end

locale finite-dimensional-real-vector-space-pair-on =
  vs1: finite-dimensional-real-vector-space-on S1 Basis1 +
  vs2: finite-dimensional-real-vector-space-on S2 Basis2
  for S1 S2 Basis1 Basis2
begin

sublocale finite-dimensional-real-vector-space-pair-1-on S1 S2 Basis1
  <proof>

sublocale finite-dimensional-vector-space-pair-on S1 S2 scaleR scaleR Basis1 Basis2
  rewrites module-on.span scaleR = vs1.span
    and module-on.dependent scaleR = vs1.dependent
    and module-on.subspace scaleR = vs1.subspace
    and module-on.span scaleR = vs2.span
    and module-on.dependent scaleR = vs2.dependent
    and module-on.subspace scaleR = vs2.subspace
  <proof>

end

```

8.2 Derivations

context *c-manifold* **begin**

Set of C^k differentiable functions on carrier, where the smooth structure is given by charts. We assume f is zero outside carrier

definition *diff-fun-space* :: ($'a \Rightarrow \text{real}$) set **where**
diff-fun-space = { f . *diff-fun k charts f* \wedge *extensional0 carrier f*}

lemma *diff-fun-spaceD*: *diff-fun k charts f* **if** $f \in$ *diff-fun-space*
<proof>

lemma *diff-fun-space-order-le*: $\text{diff-fun-space} \subseteq \text{c-manifold.diff-fun-space charts } l$
if $l \leq k$
 ⟨proof⟩

lemma *diff-fun-space-extensionalD*:
 $g \in \text{diff-fun-space} \implies \text{extensional0 carrier } g$
 ⟨proof⟩

lemma *diff-fun-space-eq*: $\text{diff-fun-space} = \{f. \text{diff-fun } k \text{ charts } f\} \cap \{f. \text{extensional0 carrier } f\}$
 ⟨proof⟩

lemma *subspace-diff-fun-space[intro, simp]*:
 $\text{subspace diff-fun-space}$
 ⟨proof⟩

lemma *diff-fun-space-times*: $f * g \in \text{diff-fun-space}$
if $f \in \text{diff-fun-space } g \in \text{diff-fun-space}$
 ⟨proof⟩

lemma *diff-fun-space-add*: $f + g \in \text{diff-fun-space}$
if $f \in \text{diff-fun-space } g \in \text{diff-fun-space}$
 ⟨proof⟩

Set of differentiable functions is a vector space

sublocale *diff-fun-space*: $\text{vector-space-pair-on diff-fun-space UNIV::real set scaleR scaleR}$
 ⟨proof⟩

Linear functional from differentiable functions to real numbers

abbreviation *linear-diff-fun* $\equiv \text{linear-on diff-fun-space (UNIV::real set) scaleR scaleR}$

Definition of a derivation.

A linear functional X is a derivation if it additionally satisfies the property $X (f * g) = f p * X g + g p * X f$. This is suppose to represent the product rule.

definition *is-derivation* :: $(('a \Rightarrow \text{real}) \Rightarrow \text{real}) \Rightarrow 'a \Rightarrow \text{bool}$ **where**
 $\text{is-derivation } X p \longleftrightarrow (\text{linear-diff-fun } X \wedge$
 $(\forall f g. f \in \text{diff-fun-space} \longrightarrow g \in \text{diff-fun-space} \longrightarrow X (f * g) = f p * X g + g p * X f))$

lemma *is-derivationI*:
 $\text{is-derivation } X p$
if $\text{linear-diff-fun } X$
 $\bigwedge f g. f \in \text{diff-fun-space} \implies g \in \text{diff-fun-space} \implies X (f * g) = f p * X g + g p * X f$

<proof>

lemma *is-derivationD*:

assumes *is-derivation* X p

shows *is-derivation-linear-on*: *linear-diff-fun* X

and *is-derivation-derivation*: $\bigwedge f g. f \in \text{diff-fun-space} \implies g \in \text{diff-fun-space}$
 $\implies X (f * g) = f p * X g + g p * X f$

<proof>

Differentiable functions on the Euclidean space

lemma *manifold-eucl-diff-fun-space-iff[simp]*:

$g \in \text{manifold-eucl.diff-fun-space } k \iff k\text{-smooth-on } UNIV g$

<proof>

8.3 Tangent space

Definition of the tangent space.

The tangent space at a point p is defined to be the set of derivations. Note we need to restrict the domain of the functional to differentiable functions.

definition *tangent-space* :: ' $a \Rightarrow ((a \Rightarrow \text{real}) \Rightarrow \text{real})$ set **where**

tangent-space $p = \{X. \text{is-derivation } X p \wedge \text{extensional0 diff-fun-space } X\}$

lemma *tangent-space-eq*: $\text{tangent-space } p = \{X. \text{is-derivation } X p\} \cap \{X. \text{extensional0 diff-fun-space } X\}$

<proof>

lemma *mem-tangent-space*: $X \in \text{tangent-space } p \iff \text{is-derivation } X p \wedge \text{extensional0 diff-fun-space } X$

<proof>

lemma *tangent-spaceI*:

$X \in \text{tangent-space } p$

if

extensional0 diff-fun-space X

linear-diff-fun X

$\bigwedge f g. f \in \text{diff-fun-space} \implies g \in \text{diff-fun-space} \implies X (f * g) = f p * X g + g p * X f$

<proof>

lemma *tangent-spaceD*:

assumes $X \in \text{tangent-space } p$

shows *tangent-space-linear-on*: *linear-diff-fun* X

and *tangent-space-restrict*: *extensional0 diff-fun-space* X

and *tangent-space-derivation*: $\bigwedge f g. f \in \text{diff-fun-space} \implies g \in \text{diff-fun-space}$
 $\implies X (f * g) = f p * X g + g p * X f$

<proof>

lemma *is-derivation-0*: *is-derivation* 0 p

<proof>

lemma *is-derivation-add*: *is-derivation* $(x + y)$ p
if x : *is-derivation* x p **and** y : *is-derivation* y p
<proof>

lemma *is-derivation-scaleR*: *is-derivation* $(c *_{\mathbb{R}} x)$ p
if x : *is-derivation* x p
<proof>

lemma *subspace-is-derivation*: *subspace* $\{X. \text{is-derivation } X \text{ } p\}$
<proof>

lemma *subspace-tangent-space*: *subspace* $(\text{tangent-space } p)$
<proof>

sublocale *tangent-space*: *real-vector-space-on* *tangent-space* p
<proof>

lemma *tangent-space-dim-eq*: *tangent-space.dim* p $X = \text{dim } X$
if $X \subseteq \text{tangent-space } p$
<proof>

properties of derivations

lemma *restrict0-in-fun-space*: *restrict0* *carrier* $f \in \text{diff-fun-space}$
if *diff-fun* k *charts* f
<proof>

lemma *restrict0-const-diff-fun-space*: *restrict0* *carrier* $(\lambda x. c) \in \text{diff-fun-space}$
<proof>

lemma *derivation-one-eq-zero*: $X (\text{restrict0 } \text{carrier } (\lambda x. 1)) = 0$ (**is** X ? $f1 = -$)
if $X \in \text{tangent-space } p$ $p \in \text{carrier}$
<proof>

lemma *derivation-const-eq-zero*: $X (\text{restrict0 } \text{carrier } (\lambda x. c)) = 0$
if $X \in \text{tangent-space } p$ $p \in \text{carrier}$
<proof>

lemma *derivation-times-eq-zeroI*: $X (f * g) = 0$ **if** $X: X \in \text{tangent-space } p$
and d : $f \in \text{diff-fun-space}$ $g \in \text{diff-fun-space}$
and z : $f \text{ } p = 0$ $g \text{ } p = 0$
<proof>

lemma *derivation-zero-localI*: $X f = 0$
if *open* W $p \in W$ $W \subseteq \text{carrier}$
 $X \in \text{tangent-space } p$
 $f \in \text{diff-fun-space}$
 $\bigwedge x. x \in W \implies f x = 0$

<proof>

lemma *derivation-eq-localI*: $X f = X g$
if *open* U $p \in U$ $U \subseteq \text{carrier}$
 $X \in \text{tangent-space } p$
 $f \in \text{diff-fun-space}$
 $g \in \text{diff-fun-space}$
 $\bigwedge x. x \in U \implies f x = g x$
<proof>

end

8.4 Push-forward on the tangent space

context *diff* **begin**

Push-forward on tangent spaces.

Given an element of the tangent space at *src*, considered as a functional X , the push-forward of X is a functional at *dest*, mapping g to $X (g \circ f)$.

definition *push-forward* :: $((\text{'a} \Rightarrow \text{real}) \Rightarrow \text{real}) \Rightarrow (\text{'b} \Rightarrow \text{real}) \Rightarrow \text{real}$ **where**
 $\text{push-forward } X = \text{restrict0 } \text{dest.diff-fun-space } (\lambda g. X (\text{restrict0 } \text{src.carrier } (g \circ f)))$

lemma *extensional-push-forward*: $\text{extensional0 } \text{dest.diff-fun-space } (\text{push-forward } X)$
<proof>

lemma *linear-push-forward*: $\text{linear } \text{push-forward}$
<proof>

Properties of push-forwards

lemma *restrict-compose-in-diff-fun-space*:
 $x \in \text{dest.diff-fun-space} \implies \text{restrict0 } \text{src.carrier } (x \circ f) \in \text{src.diff-fun-space}$
<proof>

Push-forward of a linear functional is a linear

lemma *linear-on-diff-fun-push-forward*:
 $\text{dest.linear-diff-fun } (\text{push-forward } X)$
if $\text{src.linear-diff-fun } X$
<proof>

Push-forward preserves the product rule

lemma *push-forward-is-derivation*:
 $\text{push-forward } X (x * y) = x (f p) * \text{push-forward } X y + y (f p) * \text{push-forward } X x$
(is ?l = ?r)
if $\text{deriv}: \bigwedge x y. x \in \text{src.diff-fun-space} \implies y \in \text{src.diff-fun-space} \implies X (x * y) = x p * X y + y p * X x$
and $\text{dx}: x \in \text{dest.diff-fun-space}$

and $dy: y \in \text{dest.diff-fun-space}$
and $p: p \in \text{src.carrier}$
 ⟨proof⟩

Combining, we show that the push-forward of a derivation is a derivation

lemma *push-forward-in-tangent-space*:
 $\text{push-forward } \langle \text{src.tangent-space } p \rangle \subseteq \text{dest.tangent-space } (f p)$
if $p \in \text{src.carrier}$
 ⟨proof⟩

end

Functoriality of push-forward: identity

context *c-manifold* **begin**

lemma *push-forward-id*:
 $\text{diff.push-forward } k \text{ charts } \text{charts } f X = X$
if $\bigwedge x. x \in \text{carrier} \implies f x = x$
 $X \in \text{tangent-space } p \ p \in \text{carrier}$
 ⟨proof⟩

end

Functoriality of push-forward: composition

lemma *push-forward-compose*:
 $\text{diff.push-forward } k \ M2 \ M3 \ g \ (\text{diff.push-forward } k \ M1 \ M2 \ f \ X) = \text{diff.push-forward}$
 $k \ M1 \ M3 \ (g \ o \ f) \ X$
if $X \in \text{c-manifold.tangent-space } M1 \ k \ p \ p \in \text{manifold.carrier } M1$
and $df: \text{diff } k \ M1 \ M2 \ f$ **and** $dg: \text{diff } k \ M2 \ M3 \ g$
 ⟨proof⟩

context *diffeomorphism* **begin**

If f is a diffeomorphism, then the push-forward f^* is a bijection

lemma *inv-push-forward-inverse*: $\text{push-forward } (\text{inv.push-forward } X) = X$
if $X \in \text{dest.tangent-space } p \ p \in \text{dest.carrier}$
 ⟨proof⟩

lemma *push-forward-inverse*: $\text{inv.push-forward } (\text{push-forward } X) = X$
if $X \in \text{src.tangent-space } p \ p \in \text{src.carrier}$
 ⟨proof⟩

lemma *bij-betw-push-forward*:
 $\text{bij-betw push-forward } (\text{src.tangent-space } p) \ (\text{dest.tangent-space } (f p))$
if $p: p \in \text{src.carrier}$
 ⟨proof⟩

lemma *dim-tangent-space-src-dest-eq*: $\text{dim } (\text{src.tangent-space } p) = \text{dim } (\text{dest.tangent-space } (f p))$

if $p: p \in \text{src.carrier}$ **and** $\text{dim}(\text{dest.tangent-space } (f p)) > 0$
 ⟨proof⟩

lemma *dim-tangent-space-src-dest-eq2*: $\text{dim}(\text{src.tangent-space } p) = \text{dim}(\text{dest.tangent-space } (f p))$

if $p: p \in \text{src.carrier}$ **and** $\text{dim}(\text{src.tangent-space } p) > 0$
 ⟨proof⟩

end

8.5 Smooth inclusion map

context *submanifold* **begin**

lemma *diff-inclusion*: $\text{diff } k(\text{charts-submanifold } S) \text{ charts } (\lambda x. x)$
 ⟨proof⟩

sublocale *inclusion*: $\text{diff } k \text{ charts-submanifold } S \text{ charts } \lambda x. x$
 ⟨proof⟩

lemma *linear-on-push-forward-inclusion*:

linear-on (*sub.tangent-space* p) (*tangent-space* p) *scaleR scaleR inclusion.push-forward*
 ⟨proof⟩

Extension lemma: given a differentiable function on S , and a closed set $B \subseteq S$, there exists a function f' agreeing with f on B , such that the support of f' is contained in S .

lemma *extension-lemma-submanifoldE*:

fixes $f::'a \Rightarrow 'e::\text{euclidean-space}$

assumes $f: \text{diff-fun } k(\text{charts-submanifold } S) f$

and $B: \text{closed } B \ B \subseteq \text{sub.carrier}$

obtains f' **where**

$\text{diff-fun } k \text{ charts } f'$

$(\bigwedge x. x \in B \implies f' x = f x)$

csupport-on carrier $f' \cap \text{carrier} \subseteq \text{sub.carrier}$

⟨proof⟩

lemma *inj-on-push-forward-inclusion*: *inj-on inclusion.push-forward* (*sub.tangent-space* p)

if $p: p \in \text{sub.carrier}$

⟨proof⟩

lemma *surj-on-push-forward-inclusion*:

inclusion.push-forward '*sub.tangent-space* $p \supseteq \text{tangent-space } p$

if $p: p \in \text{sub.carrier}$

⟨proof⟩

end

8.6 Tangent space of submanifold

lemma *span-idem*: $\text{span } X = X$ if subspace X
 ⟨proof⟩

context *submanifold begin*

lemma *dim-tangent-space*: $\text{dim} (\text{tangent-space } p) = \text{dim} (\text{sub.tangent-space } p)$
 if $p \in \text{sub.carrier}$ $\text{dim} (\text{sub.tangent-space } p) > 0$
 ⟨proof⟩

lemma *dim-tangent-space2*: $\text{dim} (\text{tangent-space } p) = \text{dim} (\text{sub.tangent-space } p)$
 if $p \in \text{sub.carrier}$ $\text{dim} (\text{tangent-space } p) > 0$
 ⟨proof⟩

end

8.7 Directional derivatives

When the manifold is the Euclidean space, The Frechet derivative at a in the direction of v is an element of the tangent space at a .

definition *directional-derivative::enat* $\Rightarrow 'a \Rightarrow 'a::\text{euclidean-space} \Rightarrow$
 ($'a \Rightarrow \text{real}$) $\Rightarrow \text{real}$ **where**
 $\text{directional-derivative } k \ a \ v = \text{restrict0} (\text{manifold-eucl.diff-fun-space } k) (\lambda f. \text{frechet-derivative } f \ (at \ a) \ v)$

lemma *extensional0-directional-derivative*:
 $\text{extensional0} (\text{manifold-eucl.diff-fun-space } k) (\text{directional-derivative } k \ a \ v)$
 ⟨proof⟩

lemma *extensional0-directional-derivative-le*:
 $\text{extensional0} (\text{manifold-eucl.diff-fun-space } k) (\text{directional-derivative } k' \ a \ v)$
 if $k \leq k'$
 ⟨proof⟩

lemma *directional-derivative-add[simp]*: $\text{directional-derivative } k \ a \ (x + y) = \text{directional-derivative } k \ a \ x + \text{directional-derivative } k \ a \ y$
and *directional-derivative-scaleR[simp]*: $\text{directional-derivative } k \ a \ (c *_{\mathbb{R}} x) = c *_{\mathbb{R}} \text{directional-derivative } k \ a \ x$
 if $k \neq 0$
 ⟨proof⟩

lemma *linear-directional-derivative*: $k \neq 0 \implies \text{linear} (\text{directional-derivative } k \ a)$
 ⟨proof⟩

lemma *frechet-derivative-inner[simp]*:
 $\text{frechet-derivative} (\lambda x. x \cdot j) \ (at \ a) = (\lambda x. x \cdot j)$
 ⟨proof⟩

lemma *smooth-on-inner-const*[simp]: k -smooth-on UNIV $(\lambda x. x \cdot j)$
 ⟨proof⟩

lemma *directional-derivative-inner*[simp]: $\text{directional-derivative } k \ a \ x \ (\lambda x. x \cdot j)$
 $= x \cdot j$
 ⟨proof⟩

lemma *sum-apply*: $\text{sum } f \ X \ i = \text{sum } (\lambda x. f \ x \ i) \ X$
 ⟨proof⟩

lemma *inj-on-directional-derivative*: $\text{inj-on } (\text{directional-derivative } k \ a) \ S$ **if** $k \neq 0$
 ⟨proof⟩

lemma *directional-derivative-eq-frechet-derivative*:
 $\text{directional-derivative } k \ a \ v \ f = \text{frechet-derivative } f \ (\text{at } a) \ v$
if k -smooth-on UNIV f
 ⟨proof⟩

lemma *directional-derivative-linear-on-diff-fun-space*:
 $k \neq 0 \implies \text{manifold-eucl.linear-diff-fun } k \ (\text{directional-derivative } k \ a \ x)$
 ⟨proof⟩

lemma *directional-derivative-is-derivation*:
 $\text{directional-derivative } k \ a \ x \ (f * g) = f \ a * \text{directional-derivative } k \ a \ x \ g + g \ a * \text{directional-derivative } k \ a \ x \ f$
if $f \in \text{manifold-eucl.diff-fun-space } k \ g \in \text{manifold-eucl.diff-fun-space } k \ k \neq 0$
 ⟨proof⟩

lemma *directional-derivative-in-tangent-space*[intro, simp]:
 $k \neq 0 \implies \text{directional-derivative } k \ a \ x \in \text{manifold-eucl.tangent-space } k \ a$ **for** x
 ⟨proof⟩

context *c-manifold* **begin**

lemma *is-derivation-order-le*:
 $\text{is-derivation } X \ p$
if $l \leq k$ *c-manifold.is-derivation charts* $l \ X \ p$
 ⟨proof⟩

end

lemma *smooth-on-imp-differentiable-on*: f *differentiable-on* S
if k -smooth-on $S \ f \ k > 0$
 ⟨proof⟩

Key result: for the Euclidean space, the Frechet derivatives are the only elements of the tangent space.

This result only holds for smooth manifolds, not for C^k differentiable man-

ifolds. Smoothness is used at a key point in the proof.

lemma *surj-directional-derivative:*

range (directional-derivative k a) = manifold-eucl.tangent-space k a

if $k = \infty$

<proof>

lemma *span-directional-derivative:*

span (directional-derivative ∞ a 'Basis) = manifold-eucl.tangent-space ∞ a

<proof>

lemma *directional-derivative-in-span:*

directional-derivative ∞ a x \in span (directional-derivative ∞ a 'Basis)

<proof>

lemma *linear-on-directional-derivative:*

*k \neq 0 \implies linear-on UNIV (manifold-eucl.tangent-space k a) (*_R) (*_R) (directional-derivative k a)*

<proof>

The directional derivatives at Basis forms a basis of the tangent space at a

interpretation *manifold-eucl: finite-dimensional-real-vector-space-on*

manifold-eucl.tangent-space ∞ a directional-derivative ∞ a 'Basis

<proof>

lemma *independent-directional-derivative:*

k \neq 0 \implies independent (directional-derivative k a 'Basis)

<proof>

8.8 Dimension

For the Euclidean space, the dimension of the tangent space equals the dimension of the original space.

lemma *dim-eucl-tangent-space:*

dim (manifold-eucl.tangent-space ∞ a) = DIM('a) for a::'a::euclidean-space

<proof>

context *c-manifold begin*

For a general manifold, the dimension of the tangent space at point p equals the dimension of the manifold.

lemma *dim-tangent-space: dim (tangent-space p) = DIM('b) if p: p \in carrier and*

smooth: k = ∞

<proof>

end

end

9 Cotangent Space

```
theory Cotangent-Space
  imports Tangent-Space
begin
```

9.1 Dual of a vector space

abbreviation $\text{linear-fun-on } S \equiv \text{linear-on } S \text{ (UNIV::real set) scaleR scaleR}$

definition $\text{dual-space} :: 'a::\text{real-vector set} \Rightarrow ('a \Rightarrow \text{real}) \text{ set}$ **where**
 $\text{dual-space } S = \{E. \text{linear-fun-on } S E \wedge \text{extensional0 } S E\}$

lemma dual-space-eq :

$\text{dual-space } S = \{E. \text{linear-fun-on } S E\} \cap \{E. \text{extensional0 } S E\}$
(proof)

lemma mem-dual-space :

$E \in \text{dual-space } S \iff \text{linear-fun-on } S E \wedge \text{extensional0 } S E$
(proof)

lemma dual-spaceI :

$E \in \text{dual-space } S$
if $\text{extensional0 } S E$ **linear-fun-on } S E
(proof)**

lemma dual-spaceD :

assumes $E \in \text{dual-space } S$
shows $\text{dual-space-linear-on}: \text{linear-fun-on } S E$
and $\text{dual-space-restrict[simp]}: \text{extensional0 } S E$
(proof)

lemma $\text{linear-fun-on-zero}$:

$\text{linear-fun-on } S 0$
if $\text{subspace } S$
(proof)

lemma $\text{linear-fun-on } S x \implies a \in S \implies b \in S \implies x (a + b) = x a + x b$

(proof)

lemma linear-fun-on-add :

$\text{linear-fun-on } S (x + y)$
if $x: \text{linear-fun-on } S x$ **and** $y: \text{linear-fun-on } S y$ **and** $S: \text{subspace } S$
(proof)

lemma $\text{linear-fun-on-scaleR}$:

$\text{linear-fun-on } S (c *_R x)$
if $x: \text{linear-fun-on } S x$ **and** $S: \text{subspace } S$
(proof)

lemma *subspace-linear-fun-on:*
subspace {*E*. *linear-fun-on* *S* *E*}
if *subspace* *S*
 ⟨*proof*⟩

lemma *subspace-dual-space:*
subspace (*dual-space* *S*)
if *subspace* *S*
 ⟨*proof*⟩

9.2 Dimension of dual space

Mapping from S to the dual of S

context *fixes* B S **assumes** B : *independent* B *span* $B = S$
begin

definition *inner-Basis* a $b = (\sum_{i \in B}. \text{representation } B \ a \ i * \text{representation } B \ b \ i)$
 — TODO: move to library

definition *std-dual* :: ' a ': *real-vector* \Rightarrow (' $a \Rightarrow$ *real*) **where**
std-dual $a = \text{restrict0 } S \ (\text{restrict0 } S \ (\lambda b. \text{inner-Basis } a \ b))$

lemma *inner-Basis-add:*
 $b1 \in S \implies b2 \in S \implies \text{inner-Basis } (b1 + b2) \ v = \text{inner-Basis } b1 \ v + \text{inner-Basis } b2 \ v$
 ⟨*proof*⟩

lemma *inner-Basis-add2:*
 $b1 \in S \implies b2 \in S \implies \text{inner-Basis } v \ (b1 + b2) = \text{inner-Basis } v \ b1 + \text{inner-Basis } v \ b2$
 ⟨*proof*⟩

lemma *inner-Basis-scale:*
 $b1 \in S \implies \text{inner-Basis } (c *_R \ b1) \ v = c * \text{inner-Basis } b1 \ v$
 ⟨*proof*⟩

lemma *inner-Basis-scale2:*
 $b1 \in S \implies \text{inner-Basis } v \ (c *_R \ b1) = c * \text{inner-Basis } v \ b1$
 ⟨*proof*⟩

lemma *inner-Basis-minus:*
 $b1 \in S \implies b2 \in S \implies \text{inner-Basis } (b1 - b2) \ v = \text{inner-Basis } b1 \ v - \text{inner-Basis } b2 \ v$
and *inner-Basis-minus2:*
 $b1 \in S \implies b2 \in S \implies \text{inner-Basis } v \ (b1 - b2) = \text{inner-Basis } v \ b1 - \text{inner-Basis } v \ b2$
 ⟨*proof*⟩

lemma *sum-zero-representation*:

$v = 0$

if $\bigwedge b. b \in B \implies \text{representation } B \ v \ b = 0$ **and** $v: v \in S$
<proof>

lemma *inner-Basis-0[simp]*: *inner-Basis* $0 \ a = 0$ *inner-Basis* $a \ 0 = 0$

<proof>

lemma *inner-Basis-eq-zeroI*: $a = 0$ **if** *inner-Basis* $a \ a = 0$

and *finite* $B \ a \in S$

<proof>

lemma *inner-Basis-zero*: *inner-Basis* $a \ a = 0 \iff a = 0$

if *finite* $B \ a \in S$

<proof>

lemma *subspace-S*: *subspace* S

<proof>

interpretation S : *real-vector-space-on* S

<proof>

interpretation *dual*: *real-vector-space-on* *dual-space* S

<proof>

lemma *std-dual-linear*:

linear-on S (*dual-space* S) *scaleR* *scaleR* *std-dual*

<proof>

lemma *image-std-dual*:

std-dual $' S \subseteq$ *dual-space* S

if *subspace* S

<proof>

lemma *inj-std-dual*:

inj-on *std-dual* S

if *subspace* S *finite* B

<proof>

lemma *inner-Basis-sum*:

$(\bigwedge i. i \in I \implies x \ i \in S) \implies \text{inner-Basis } (\sum i \in I. x \ i) \ v = (\sum i \in I. \text{inner-Basis } (x \ i) \ v)$

<proof>

lemma *inner-Basis-sum2*:

$(\bigwedge i. i \in I \implies x \ i \in S) \implies \text{inner-Basis } v \ (\sum i \in I. x \ i) = (\sum i \in I. \text{inner-Basis } v \ (x \ i))$

<proof>

lemma *B-sub-S*: $B \subseteq S$
<proof>

lemma *inner-Basis-eq-representation*:
inner-Basis i $x = \text{representation } B$ x i
if $i \in B$ *finite* B
<proof>

lemma *surj-std-dual*:
std-dual $'S \supseteq \text{dual-space } S$ **if** *subspace* S *finite* B
<proof>

lemma *std-dual-bij-betw*:
bij-betw (*std-dual*) S (*dual-space* S)
if *finite* B
<proof>

lemma *std-dual-eq-dual-space*: *finite* $B \implies \text{std-dual } 'S = \text{dual-space } S$
<proof>

lemma *dim-dual-space*:
assumes *finite* B
shows $\text{dim } (\text{dual-space } S) = \text{dim } S$
<proof>

end

9.3 Dual map

context *real-vector-space-pair-on* **begin**

definition *dual-map* $:: ('a \Rightarrow 'b) \Rightarrow ('b \Rightarrow \text{real}) \Rightarrow ('a \Rightarrow \text{real})$ **where**
dual-map f $y = \text{restrict0 } S$ $(\lambda x. y (f x))$

lemma *subspace-dual-S*: *subspace* (*dual-space* S)
<proof>

lemma *subspace-dual-T*: *subspace* (*dual-space* T)
<proof>

lemma *dual-map-linear*:
linear-on (*dual-space* T) (*dual-space* S) *scaleR* *scaleR* (*dual-map* f)
<proof>

lemma *image-dual-map*:
dual-map f $'(\text{dual-space } T) \subseteq \text{dual-space } S$
if f : *linear-on* S T *scaleR* *scaleR* f **and**
defined: f $'S \subseteq T$
<proof>

end

Functoriality of dual map: identity

context *real-vector-space-on* **begin**

lemma *dual-map-id*:

real-vector-space-pair-on.dual-map $S f y = y$

if $f: \bigwedge x. x \in S \implies f x = x$ **and** $y: y \in \text{dual-space } S$
(*proof*)

end

abbreviation *dual-map* \equiv *real-vector-space-pair-on.dual-map*

lemmas *dual-map-def* = *real-vector-space-pair-on.dual-map-def*

Functoriality of dual map: composition

lemma *dual-map-compose*:

dual-map $S f$ (*dual-map* $T g$ x) = *dual-map* S ($g \circ f$) x

if $x \in \text{dual-space } U$ **and** *linear-on* $T U$ *scaleR* *scaleR* g

and $f: \text{linear-on } S T$ *scaleR* *scaleR* f

and *defined*: $f ' S \subseteq T$

and ST : *real-vector-space-pair-on* $S T$

and TU : *real-vector-space-pair-on* $T U$

(*proof*)

9.4 Definition of cotangent space

context *c-manifold* **begin**

definition *cotangent-space* :: $'a \Rightarrow ((('a \Rightarrow \text{real}) \Rightarrow \text{real}) \Rightarrow \text{real})$ **set where**
cotangent-space $p = \text{dual-space } (\text{tangent-space } p)$

lemma *subspace-cotangent-space*:

subspace (*cotangent-space* p)

(*proof*)

sublocale *cotangent-space*: *real-vector-space-on* *cotangent-space* p

(*proof*)

lemma *cotangent-space-dim-eq*: *cotangent-space.dim* $p X = \text{dim } X$

if $X \subseteq \text{cotangent-space } p$

(*proof*)

lemma *dim-cotangent-space*:

dim (*cotangent-space* p) = $\text{DIM}('b)$ **if** $p \in \text{carrier}$ **and** $k = \infty$

(*proof*)

end

9.5 Pullback of cotangent space

context *diff* begin

definition *pull-back* :: 'a \Rightarrow (((('b \Rightarrow real) \Rightarrow real) \Rightarrow real) \Rightarrow (('a \Rightarrow real) \Rightarrow real) \Rightarrow real **where**

pull-back p = dual-map (src.tangent-space p) push-forward

lemma

linear-pullback: linear-on (dest.cotangent-space (f p)) (src.cotangent-space p) scaleR scaleR (pull-back p) **and**

image-pullback: pull-back p ' (dest.cotangent-space (f p)) \subseteq src.cotangent-space p

if p \in src.carrier

<proof>

end

9.6 Cotangent field of a function

context *c-manifold* begin

Given a function f, the cotangent vector of f at a point p is defined as follows: given a tangent vector X at p, considered as a functional, evaluate X on f.

definition *cotangent-field* :: ('a \Rightarrow real) \Rightarrow 'a \Rightarrow (((('a \Rightarrow real) \Rightarrow real) \Rightarrow real) \Rightarrow real) **where**

cotangent-field f p = restrict0 (tangent-space p) ($\lambda X. X f$)

lemma *cotangent-field-is-cotangent*:

cotangent-field f p \in cotangent-space p

<proof>

9.7 Tangent field of a path

Given a path c, the tangent vector of c at real number x (or at point c(x)) is defined as follows: given a function f, take the derivative of the real-valued function f \circ c.

definition *tangent-field* :: (real \Rightarrow 'a) \Rightarrow real \Rightarrow (('a \Rightarrow real) \Rightarrow real) **where**

tangent-field c x = restrict0 diff-fun-space ($\lambda f. \text{frechet-derivative } (f \circ c) \text{ (at } x) 1$)

lemma *tangent-field-is-tangent*:

tangent-field c x \in tangent-space (c x)

if c-smooth: diff k charts-eucl charts c **and** smooth: k > 0

<proof>

9.8 Integral along a path

lemma *fundamental-theorem-of-path-integral*:

$((\lambda x. (\text{cotangent-field } f (c \ x)) (\text{tangent-field } c \ x)) \text{ has-integral } f (c \ b) - f (c \ a))$
 $\{a..b\}$
if $ab: a \leq b$ **and** $f: f \in \text{diff-fun-space}$ **and** $c: \text{diff } k \text{ charts-eucl charts } c$ **and** $k: k \neq 0$
 $\langle \text{proof} \rangle$

end

end

10 Product Manifold

theory *Product-Manifold*
imports *Differentiable-Manifold*
begin

locale *c-manifold-prod* =
 $m1: c\text{-manifold charts1 } k$ +
 $m2: c\text{-manifold charts2 } k$ **for** k *charts1 charts2*

begin

lift-definition *prod-chart* :: $('a, 'b) \text{ chart} \Rightarrow ('c, 'd) \text{ chart} \Rightarrow ('a \times 'c, 'b \times 'd) \text{ chart}$
is $\lambda(d::'a \text{ set}, d'::'b \text{ set}, f::'a \Rightarrow 'b, f'::'b \Rightarrow 'a).$
 $\lambda(e::'c \text{ set}, e'::'d \text{ set}, g::'c \Rightarrow 'd, g'::'d \Rightarrow 'c).$
 $(d \times e, d' \times e', \lambda(x,y). (f \ x, g \ y), \lambda(x,y). (f' \ x, g' \ y))$
 $\langle \text{proof} \rangle$

lemma *domain-prod-chart[simp]*: $\text{domain } (\text{prod-chart } c1 \ c2) = \text{domain } c1 \times \text{domain } c2$
and *codomain-prod-chart[simp]*: $\text{codomain } (\text{prod-chart } c1 \ c2) = \text{codomain } c1 \times \text{codomain } c2$
and *apply-prod-chart[simp]*: $\text{apply-chart } (\text{prod-chart } c1 \ c2) = (\lambda(x,y). (c1 \ x, c2 \ y))$
and *inv-chart-prod-chart[simp]*: $\text{inv-chart } (\text{prod-chart } c1 \ c2) = (\lambda(x,y). (\text{inv-chart } c1 \ x, \text{inv-chart } c2 \ y))$
 $\langle \text{proof} \rangle$

lemma *prod-chart-compat*:
 $k\text{-smooth-compat } (\text{prod-chart } c1 \ c2) (\text{prod-chart } d1 \ d2)$
if *compat1*: $k\text{-smooth-compat } c1 \ d1$ **and** *compat2*: $k\text{-smooth-compat } c2 \ d2$
 $\langle \text{proof} \rangle$

definition *prod-charts* :: $('a \times 'c, 'b \times 'd) \text{ chart set}$ **where**
 $\text{prod-charts} = \{\text{prod-chart } c1 \ c2 \mid c1 \ c2. c1 \in \text{charts1} \wedge c2 \in \text{charts2}\}$

lemma *c-manifold-atlas-product*: $c\text{-manifold prod-charts } k$
 $\langle \text{proof} \rangle$

end

end

11 Sphere

theory *Sphere*

imports *Differentiable-Manifold*

begin

typedef (overloaded) ('a::real-normed-vector) *sphere* =
 {a::'a×real. norm a = 1}
 ⟨proof⟩

setup-lifting *type-definition-sphere*

lift-definition *top-sphere* :: ('a::real-normed-vector) *sphere* **is** (0, 1) ⟨proof⟩

lift-definition *st-proj1* :: ('a::real-normed-vector) *sphere* ⇒ 'a **is**
 λ(x,z). x /_R (1 - z) ⟨proof⟩

lift-definition *st-proj1-inv* :: ('a::real-normed-vector) ⇒ 'a *sphere* **is**
 λx. ((2 / ((norm x) ^ 2 + 1)) *_R x, ((norm x) ^ 2 - 1) / ((norm x) ^ 2 + 1))
 ⟨proof⟩

lift-definition *bot-sphere* :: ('a::real-normed-vector) *sphere* **is** (0, -1) ⟨proof⟩

lift-definition *st-proj2* :: ('a::real-normed-vector) *sphere* ⇒ 'a **is**
 λ(x,z). x /_R (1 + z) ⟨proof⟩

lift-definition *st-proj2-inv* :: ('a::real-normed-vector) ⇒ 'a *sphere* **is**
 λx. ((2 / ((norm x) ^ 2 + 1)) *_R x, (1 - (norm x) ^ 2) / ((norm x) ^ 2 + 1))
 ⟨proof⟩

instantiation *sphere* :: (real-normed-vector) *topological-space*

begin

lift-definition *open-sphere* :: 'a *sphere* *set* ⇒ bool **is**
 openin (*subtopology* (*euclidean*::('a×real) *topology*) {a. norm a = 1}) ⟨proof⟩

instance

⟨proof⟩

end

instance *sphere* :: (real-normed-vector) *t2-space*

<proof>

instance *sphere* :: (*euclidean-space*) *second-countable-topology*
<proof>

lemma *transfer-continuous-on1* [*transfer-rule*]:

includes *lifting-syntax*

shows (*rel-set* (=) \implies ((=) \implies *pcr-sphere* (=)) \implies (=)) ($\lambda X :: 'a :: t2\text{-space}$
set. continuous-on X) *continuous-on*

<proof>

lemma *transfer-continuous-on2* [*transfer-rule*]:

includes *lifting-syntax*

shows (*rel-set* (*pcr-sphere* (=)) \implies (*pcr-sphere* (=) \implies (=)) \implies (=))
($\lambda X. \text{continuous-on } (X \cap \{x. \text{norm } x = 1\})$) ($\lambda X. \text{continuous-on } X$)

<proof>

lemma *st-proj1-inv-continuous*:

continuous-on UNIV st-proj1-inv

<proof>

lemma *st-proj1-continuous*:

continuous-on (UNIV - {top-sphere}) st-proj1

<proof>

lemma *st-proj1-inv*: *st-proj1-inv* (*st-proj1* *x*) = *x*

if *x* \neq *top-sphere*

<proof>

lemma *st-proj1-inv-inv*: *st-proj1* (*st-proj1-inv* *x*) = *x*

<proof>

lemma *st-proj1-inv-ne-top*: *st-proj1-inv* *xa* \neq *top-sphere*

<proof>

lemma *homeomorphism-st-proj1*: *homeomorphism* (*UNIV* - {*top-sphere*}) *UNIV*

st-proj1 st-proj1-inv

<proof>

lemma *st-proj2-inv-continuous*:

continuous-on UNIV st-proj2-inv

<proof>

lemma *st-proj2-continuous*:

continuous-on (UNIV - {bot-sphere}) st-proj2

<proof>

lemma *st-proj2-inv*: *st-proj2-inv* (*st-proj2* *x*) = *x*

if *x* \neq *bot-sphere*

<proof>

lemma *st-proj2-inv-inv*: *st-proj2 (st-proj2-inv x) = x*
<proof>

lemma *st-proj2-inv-ne-top*: *st-proj2-inv xa ≠ bot-sphere*
<proof>

lemma *homeomorphism-st-proj2*: *homeomorphism (UNIV - {bot-sphere}) UNIV*
st-proj2 st-proj2-inv
<proof>

lift-definition *st-proj1-chart* :: (*'a sphere, 'a::euclidean-space*) *chart*
is (*UNIV - {top-sphere::'a sphere}, UNIV::'a set, st-proj1, st-proj1-inv*)
<proof>

lift-definition *st-proj2-chart* :: (*'a sphere, 'a::euclidean-space*) *chart*
is (*UNIV - {bot-sphere::'a sphere}, UNIV::'a set, st-proj2, st-proj2-inv*)
<proof>

lemma *st-projs-compat*:
includes *lifting-syntax*
shows ∞ -*smooth-compat st-proj1-chart st-proj2-chart*
<proof>

definition *charts-sphere* :: (*'a::euclidean-space sphere, 'a*) *chart set where*
charts-sphere ≡ {st-proj1-chart, st-proj2-chart}

lemma *c-manifold-atlas-sphere*: *c-manifold charts-sphere* ∞
<proof>

end

12 Projective Space

theory *Projective-Space*
imports *Differentiable-Manifold HOL-Library.Quotient-Set*
begin

Some of the main things to note here: double transfer (\rightarrow nonzero \rightarrow quotient)

12.1 Subtype of nonzero elements

lemma *open-ne-zero*: *open {x::'a::t1-space. x ≠ c}*
<proof>

typedef (**overloaded**) *'a::euclidean-space nonzero* = *UNIV - {0::'a::euclidean-space}*
<proof>

```

setup-lifting type-definition-nonzero

instantiation nonzero :: (euclidean-space) topological-space
begin

lift-definition open-nonzero::'a nonzero set  $\Rightarrow$  bool is open::'a set  $\Rightarrow$  bool  $\langle$ proof $\rangle$ 

instance
   $\langle$ proof $\rangle$ 

end

lemma open-nonzero-openin-transfer:
  (rel-set (pcr-nonzero A)  $===>$  (=)) (openin (top-of-set (Collect (Domainp (pcr-nonzero
  A)))) open
  if is-equality A
   $\langle$ proof $\rangle$ 

instantiation nonzero :: (euclidean-space) scaleR
begin
lift-definition scaleR-nonzero::real  $\Rightarrow$  'a nonzero  $\Rightarrow$  'a nonzero is  $\lambda$  c x. if c = 0
  then One else c *R x
   $\langle$ proof $\rangle$ 
instance  $\langle$ proof $\rangle$ 

end

instantiation nonzero :: (euclidean-space) plus
begin
lift-definition plus-nonzero::'a nonzero  $\Rightarrow$  'a nonzero  $\Rightarrow$  'a nonzero is  $\lambda$  x y. if x
  + y = 0 then One else x + y
   $\langle$ proof $\rangle$ 
instance  $\langle$ proof $\rangle$ 
end

instantiation nonzero :: (euclidean-space) minus
begin
lift-definition minus-nonzero::'a nonzero  $\Rightarrow$  'a nonzero  $\Rightarrow$  'a nonzero is  $\lambda$  x y. if
  x = y then One else x - y
   $\langle$ proof $\rangle$ 
instance  $\langle$ proof $\rangle$ 
end

instantiation nonzero :: (euclidean-space) dist
begin
lift-definition dist-nonzero::'a nonzero  $\Rightarrow$  'a nonzero  $\Rightarrow$  real is dist  $\langle$ proof $\rangle$ 
instance  $\langle$ proof $\rangle$ 
end

```

```

instantiation nonzero :: (euclidean-space) norm
begin
lift-definition norm-nonzero::'a nonzero  $\Rightarrow$  real is norm <proof>
instance <proof>
end

```

```

instance nonzero :: (euclidean-space) t2-space
  <proof>

```

```

lemma scaleR-one-nonzero[simp]: 1 *R x = (x::- nonzero)
  <proof>

```

```

lemma scaleR-scaleR-nonzero[simp]: b  $\neq$  0  $\implies$  scaleR a (scaleR b x) = scaleR (a
* b) (x::- nonzero)
  <proof>

```

```

instance nonzero :: (euclidean-space) second-countable-topology
  <proof>

```

12.2 Quotient

```

inductive proj-rel :: 'a::euclidean-space nonzero  $\Rightarrow$  'a nonzero  $\Rightarrow$  bool for x where
  c  $\neq$  0  $\implies$  proj-rel x (c *R x)

```

```

lemma proj-rel-parametric: (pcr-nonzero A  $\implies$  pcr-nonzero A  $\implies$  (=))
proj-rel proj-rel
if [transfer-rule]: ((=)  $\implies$  pcr-nonzero A  $\implies$  pcr-nonzero A) (*R) (*R)
  bi-unique A
  <proof>

```

```

quotient-type (overloaded) 'a proj-space = ('a::euclidean-space  $\times$  real) nonzero
/ proj-rel
morphisms rep-proj Proj
parametric proj-rel-parametric
  <proof>

```

```

lemma surj-Proj: surj Proj
  <proof>

```

```

definition proj-topology :: 'a::euclidean-space proj-space topology where
  proj-topology = map-topology Proj euclidean

```

```

instantiation proj-space :: (euclidean-space) topological-space
begin

```

```

definition open-proj-space :: 'a proj-space set  $\Rightarrow$  bool where
  open-proj-space = openin (map-topology Proj euclidean)

```

lemma *topspace-map-Proj*: $\text{topspace } (\text{map-topology } \text{Proj } \text{euclidean}) = \text{UNIV}$
 ⟨proof⟩

instance
 ⟨proof⟩

end

lemma *open-vimage-ProjI*: $\text{open } T \implies \text{open } (\text{Proj } - ' T)$
 ⟨proof⟩

lemma *open-vimage-ProjD*: $\text{open } (\text{Proj } - ' T) \implies \text{open } T$
 ⟨proof⟩

lemma *open-vimage-Proj-iff[simp]*: $\text{open } (\text{Proj } - ' T) = \text{open } T$
 ⟨proof⟩

lemma *euclidean-proj-space-def*: $\text{euclidean} = \text{map-topology } \text{Proj } \text{euclidean}$
 ⟨proof⟩

lemma *continuous-on-proj-spaceI*: $\text{continuous-on } (S) f \text{ if } \text{continuous-on } (\text{Proj } - ' S) (f \circ \text{Proj}) \text{ open } (S)$
for $f :: \text{proj-space} \Rightarrow -$
 ⟨proof⟩

lemma *saturate-eq*: $\text{Proj } - ' \text{Proj } ' X = (\bigcup c \in \text{UNIV} - \{0\}. (*_R) c ' X)$
 ⟨proof⟩

lemma *open-scaling-nonzero*: $c \neq 0 \implies \text{open } s \implies \text{open } ((*_R) c ' s :: 'a :: \text{euclidean-space nonzero set})$
 ⟨proof⟩

12.3 Proof of Hausdorff property

lemma *Proj-open-map*: $\text{open } (\text{Proj } ' X) \text{ if } \text{open } X$
 ⟨proof⟩

lemma *proj-rel-transfer[transfer-rule]*:
 ($\text{pcr-nonzero } A \implies \text{pcr-nonzero } A \implies (=)$) $(\lambda x a. \exists c. a = \text{sr } c x \wedge c \neq 0) \text{ proj-rel}$
if $[\text{transfer-rule}]$: ($(=) \implies \text{pcr-nonzero } A \implies \text{pcr-nonzero } A$) $\text{sr } (*_R)$
 $\text{bi-unique } A$
 ⟨proof⟩

lemma *bool-aux*: $a \wedge (a \longrightarrow b) \longleftrightarrow a \wedge b$ ⟨proof⟩

lemma *closed-proj-rel*: $\text{closed } \{(x :: 'a :: \text{euclidean-space nonzero}, y :: 'a \text{ nonzero}). \text{proj-rel } x y\}$
 ⟨proof⟩

lemma *closed-Proj-rel*: $\text{closed } \{(x, y). \text{Proj } x = \text{Proj } y\}$

<proof>

instance *proj-space* :: (*euclidean-space*) *t2-space*
<proof>

instance *proj-space* :: (*euclidean-space*) *second-countable-topology*
<proof>

12.4 Charts

12.4.1 Chart for last coordinate

lift-definition *chart-last-nonzero* :: (*'a::euclidean-space* × *real*) *nonzero* ⇒ *'a is*
 $\lambda(x,c). x /_R c$ *<proof>*

lemma *chart-last-nonzero-scaleR[simp]*: $c \neq 0 \implies \text{chart-last-nonzero } (c *_R n) =$
chart-last-nonzero n
<proof>

lift-definition *chart-last* :: *'a::euclidean-space* *proj-space* ⇒ *'a is* *chart-last-nonzero*
<proof>

lift-definition *chart-last-inv-nonzero* :: *'a* ⇒ (*'a::euclidean-space* × *real*) *nonzero is*
 $\lambda x. (x, 1)$
<proof>

lift-definition *chart-last-inv* :: *'a* ⇒ *'a::euclidean-space* *proj-space is* *chart-last-inv-nonzero*
<proof>

lift-definition *chart-last-domain-nonzeroP* :: (*'a::euclidean-space* × *real*) *nonzero*
⇒ *bool is*
 $\lambda x. \text{snd } x \neq 0$ *<proof>*

lift-definition *chart-last-domainP* :: *'a::euclidean-space* *proj-space* ⇒ *bool is* *chart-last-domain-nonzeroP*
<proof>

lemma *open-chart-last-domain*: *open (Collect chart-last-domainP)*
<proof>

lemma *Proj-vimage-chart-last-domainP*: *Proj -' Collect chart-last-domainP =*
Collect (chart-last-domain-nonzeroP)
<proof>

lemma *chart-last-continuous*:

notes [*transfer-rule*] = *open-nonzero-openin-transfer*

shows *continuous-on (Collect chart-last-domainP) chart-last*
<proof>

lemma *chart-last-inv-continuous*:

notes [*transfer-rule*] = *open-nonzero-openin-transfer*

shows *continuous-on UNIV chart-last-inv*
<proof>

lemma *proj-rel-iff*: $\text{proj-rel } a \ b \longleftrightarrow (\exists c \neq 0. \ b = c *_{\mathbb{R}} a)$
<proof>

lemma *chart-last-inverse*: $\text{chart-last-inv } (\text{chart-last } x) = x$ **if** *chart-last-domainP*
x
<proof>

lemma *chart-last-inv-inverse*: $\text{chart-last } (\text{chart-last-inv } x) = x$
<proof>

lemma *chart-last-domainP-chart-last-inv*: $\text{chart-last-domainP } (\text{chart-last-inv } x)$
<proof>

lemma *homeomorphism-chart-last*:
homeomorphism (Collect chart-last-domainP) UNIV chart-last chart-last-inv
<proof>

lift-definition *last-chart*::('a::euclidean-space proj-space, 'a) **chart is**
(Collect chart-last-domainP, UNIV, chart-last, chart-last-inv)
<proof>

12.4.2 Charts for first $\text{DIM}('a)$ coordinates

lift-definition *chart-basis-nonzero* :: 'a \Rightarrow ('a::euclidean-space \times real) nonzero \Rightarrow 'a
is
 $\lambda b. \lambda(x, c). (x + (c - x \cdot b) *_{\mathbb{R}} b) /_{\mathbb{R}} (x \cdot b)$ *<proof>*

lift-definition *chart-basis* :: 'a \Rightarrow 'a::euclidean-space proj-space \Rightarrow 'a **is**
chart-basis-nonzero
<proof>

lift-definition *chart-basis-domain-nonzeroP* :: 'a \Rightarrow ('a::euclidean-space \times real) nonzero
 \Rightarrow bool **is**
 $\lambda b (x, -). (x \cdot b) \neq 0$ *<proof>*

lift-definition *chart-basis-domainP* :: 'a \Rightarrow 'a::euclidean-space proj-space \Rightarrow bool
is *chart-basis-domain-nonzeroP*
<proof>

lemma *Proj-vimage-chart-basis-domainP*:
Proj - 'Collect (chart-basis-domainP b) = Collect (chart-basis-domain-nonzeroP
b)
<proof>

lemma *open-chart-basis-domain*: *open (Collect (chart-basis-domainP b))*

<proof>

lemma *chart-basis-continuous:*

notes [*transfer-rule*] = *open-nonzero-openin-transfer*

shows *continuous-on* (*Collect* (*chart-basis-domainP* b)) (*chart-basis* b)

<proof>

context

fixes *b::'a::euclidean-space*

assumes *b: b ∈ Basis*

begin

lemma *b-neq0: b ≠ 0 <proof>*

lift-definition *chart-basis-inv-nonzero :: 'a ⇒ ('a::euclidean-space × real) nonzero*

is

$\lambda x. (x + (1 - x \cdot b) *_{\mathbb{R}} b, x \cdot b)$

<proof>

lift-definition *chart-basis-inv :: 'a ⇒ 'a::euclidean-space proj-space is*

chart-basis-inv-nonzero <proof>

lemma *chart-basis-inv-continuous:*

notes [*transfer-rule*] = *open-nonzero-openin-transfer*

shows *continuous-on UNIV chart-basis-inv*

<proof>

lemma *chart-basis-inv-inverse: chart-basis b (chart-basis-inv x) = x*

<proof>

lemma *chart-basis-inverse: chart-basis-inv (chart-basis b x) = x if chart-basis-domainP*

b x

<proof>

lemma *chart-basis-domainP-chart-basis-inv: chart-basis-domainP b (chart-basis-inv*

x)

<proof>

lemma *homeomorphism-chart-basis:*

homeomorphism (Collect (chart-basis-domainP b)) UNIV (chart-basis b) chart-basis-inv

<proof>

lift-definition *basis-chart::('a proj-space, 'a) chart*

is (*Collect* (*chart-basis-domainP* b), *UNIV*, *chart-basis* b, *chart-basis-inv*)

<proof>

end

12.4.3 Atlas

definition *charts-proj-space* = *insert last-chart (basis-chart ‘Basis)*

lemma *chart-last-basis-defined*:

chart-last-domainP xa \implies chart-basis-domainP b xa \implies chart-last xa \cdot b \neq 0
<proof>

lemma *chart-basis-last-defined*:

*b \in Basis \implies chart-last-domainP xa \implies chart-basis-domainP b xa \implies chart-basis
b xa \cdot b \neq 0*
<proof>

lemma *compat-last-chart*: ∞ -smooth-*compat last-chart (basis-chart b)*

if [*transfer-rule*]: *b \in Basis*
<proof>

lemma *smooth-on-basis-comp-inv*: *smooth-on {x. (x + (1 - x \cdot a) \cdot b) \neq 0}*
(chart-basis b \circ chart-basis-inv a)

if [*transfer-rule*]: *a \in Basis b \in Basis*
<proof>

lemma *chart-basis-basis-defined*:

*a \neq b \implies chart-basis-domainP a xa \implies chart-basis-domainP b xa \implies chart-basis
a xa \cdot b \neq 0*
if *a \in Basis b \in Basis*
<proof>

lemma *compat-basis-chart*: ∞ -smooth-*compat (basis-chart a) (basis-chart b)*

if [*transfer-rule*]: *a \in Basis b \in Basis*
<proof>

lemma *c-manifold-proj-space*: *c-manifold charts-proj-space ∞*

<proof>

end

References

- [1] J. M. Lee. *Introduction to Smooth Manifolds*. Springer-Verlag, New York, 2012.