Smooth Manifolds

Fabian Immler and Bohua Zhan

March 19, 2025

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

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

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

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

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

lemma *continuous-on-transfer-right-total*[*transfer-rule*]:
(*rel-set A* ===> (*A* ===> *B*) ===> (=)) (λX::′*a*::*t2-space set*. *continuous-on* (*X* ∩ *Collect AP*)) (λY::′*b*::*t2-space set*. *continuous-on Y*)
if *DomainA*: *Domainp A* = *AP*
and [*folded DomainA*, *transfer-rule*]: *bi-unique A right-total A* (*rel-set A* ===> (=)) (*openin* (*top-of-set* (*Collect AP*))) *open*
bi-unique B bi-total B (*rel-set B* ===> (=)) *open open*
⟨*proof*⟩

lemma *continuous-on-transfer-right-total2*[*transfer-rule*]:
(*rel-set A* ===> (*A* ===> *B*) ===> (=)) (λX::′*a*::*t2-space set*. *continuous-on X*) (λY::′*b*::*t2-space set*. *continuous-on Y*)
if *DomainB*: *Domainp B* = *BP*
and [*folded DomainB*, *transfer-rule*]: *bi-unique A bi-total A* (*rel-set A* ===> (=)) *open open*
bi-unique B right-total B (*rel-set B* ===> (=)) ((*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 (*rel-set* (*rel-set A*) ===> *rel-set A* ===> (=)) (*generate-topology o* (*insert* (*Collect* (*Domainp A*)))) *generate-topology*
⟨*proof*⟩

end

1.2 Miscellaneous

lemmas [*simp del*] = *mem-ball*

lemma *in-closureI*[*intro*, *simp*]: $x \in X \Longrightarrow x \in$ *closure* X
⟨*proof*⟩

lemmas *open-continuous-vimage* = *continuous-on-open-vimage*[*THEN iffD1*, *rule-format*]
lemma *open-continuous-vimage*′: *open* $s \Longrightarrow$ *continuous-on* $s\ f \Longrightarrow$ *open* $B \Longrightarrow$ *open* $(s \cap f -` B)$
⟨*proof*⟩

lemma *support-on-mono*: *support-on carrier* $f \subseteq$ *support-on carrier* g
if $\bigwedge x.\ x \in$ *carrier* $\Longrightarrow f\ x \neq 0 \Longrightarrow g\ x \neq 0$
⟨*proof*⟩

lemma *image-prod*: $(\lambda(x, y).\ (f\ x, g\ y))$ ` $(A \times B) = f$ ` $A \times g$ ` B ⟨*proof*⟩

1.3 Closed support

definition *csupport-on* $X\ S$ = *closure* (*support-on* $X\ S$)

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

lemma *not-in-csupportD*: $x \notin$ *csupport-on carrier* $\varphi \Longrightarrow x \in$ *carrier* $\Longrightarrow \varphi\ x = 0$
⟨*proof*⟩

lemma *csupport-on-mono*: *csupport-on carrier* $f \subseteq$ *csupport-on carrier* g
if $\bigwedge x.\ x \in$ *carrier* $\Longrightarrow f\ x \neq 0 \Longrightarrow g\ x \neq 0$
⟨*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*::-::*real-normed-vector*
⟨*proof*⟩

lemma *in-range-scaleR-iff*: $x \in$ *range* $((*_R)\ c) \longleftrightarrow c = 0 \longrightarrow x = 0$
for *x*::-::*real-vector*
⟨*proof*⟩

lemma *homeomorphism-scaleR*:

homeomorphism UNIV UNIV ($\lambda x.\ c *_R x$::-::*real-normed-vector*) ($\lambda x.\ x /_R c$)
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 = *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*: $t = s \longleftrightarrow$ (*topspace* t = *topspace* s $\wedge$
($\forall x \subseteq$*topspace* t. *openin* $t\ x$ = *openin* $s\ x$))
⟨*proof*⟩

1.7 Finer topologies

definition *finer-than* (**infix** ‹(*finer′-than*)› *50*)
where *T1 finer-than T2* $\longleftrightarrow$ *continuous-map T1 T2* ($\lambda x.\ x$)

lemma *finer-than-iff-nhds*:
T1 finer-than T2 $\longleftrightarrow$ ($\forall X$. *openin T2 X* $\longrightarrow$ *openin T1* ($X \cap$ *topspace T1*)) $\wedge$
(*topspace T1* $\subseteq$ *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 finer-than T T 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*:
  $\textit{support-on } X\ (\lambda x.\ \sum i{\in}S.\ f\ i\ x) = (\bigcup i{\in}S.\ \textit{support-on } X\ (f\ i))$
  if $\textit{finite } S\ \bigwedge x\ i\ .\ x \in X \Longrightarrow i \in S \Longrightarrow f\ i\ x \geq 0$
  for f::-⇒-⇒-::*ordered-comm-monoid-add*
  ⟨*proof*⟩

lemma *support-on-nonneg-sum-subset*:
  $\textit{support-on } X\ (\lambda x.\ \sum i{\in}S.\ f\ i\ x) \subseteq (\bigcup i{\in}S.\ \textit{support-on } X\ (f\ i))$
  for f::-⇒-⇒-::*ordered-comm-monoid-add*
⟨*proof*⟩

lemma *support-on-nonneg-sum-subset′*:
  $\textit{support-on } X\ (\lambda x.\ \sum i{\in}S\ x.\ f\ i\ x) \subseteq (\bigcup x{\in}X.\ (\bigcup i{\in}S\ x.\ \textit{support-on } X\ (f\ i)))$
  for f::-⇒-⇒-::*ordered-comm-monoid-add*
  ⟨*proof*⟩

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

definition $\textit{final-topology } X\ Y\ f =$
  $\textit{topology } (\lambda U.\ U \subseteq X \wedge$
    $(\forall i.\ \textit{openin } (Y\ i)\ (f\ i\ -`\ U \cap \textit{topspace } (Y\ i))))$

lemma *openin-final-topology*:
  $\textit{openin } (\textit{final-topology } X\ Y\ f) =$
    $(\lambda U.\ U \subseteq X \wedge (\forall i.\ \textit{openin } (Y\ i)\ (f\ i\ -`\ U \cap \textit{topspace } (Y\ i))))$
  ⟨*proof*⟩

lemma *topspace-final-topology*:
  $\textit{topspace } (\textit{final-topology } X\ Y\ f) = X$
  if $\bigwedge i.\ f\ i \in \textit{topspace } (Y\ i) \rightarrow X$
⟨*proof*⟩

lemma *continuous-on-final-topologyI2*:
  $\textit{continuous-map } (Y\ i)\ (\textit{final-topology } X\ Y\ f)\ (f\ i)$
  if $\bigwedge i.\ f\ i \in \textit{topspace } (Y\ i) \rightarrow X$
  ⟨*proof*⟩

lemma *continuous-on-final-topologyI1*:
  $\textit{continuous-map } (\textit{final-topology } X\ Y\ f)\ Z\ g$
  if *hyp*: $\bigwedge i.\ \textit{continuous-map } (Y\ i)\ Z\ (g\ o\ f\ i)$
    and *that*: $\bigwedge i.\ f\ i \in \textit{topspace } (Y\ i) \rightarrow X\ g \in X \rightarrow \textit{topspace } Z$
  ⟨*proof*⟩

lemma *continuous-on-final-topology-iff*:
  $\textit{continuous-map } (\textit{final-topology } X\ Y\ f)\ Z\ g \longleftrightarrow (\forall i.\ \textit{continuous-map } (Y\ i)\ Z\ (g\ o\ f\ i))$
  if $\bigwedge i.\ f\ i \in \textit{topspace } (Y\ i) \rightarrow X\ g \in X \rightarrow \textit{topspace } Z$

⟨*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) = (λU. $U \subseteq p$ ‘ *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 $\to$ *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*: $\bigwedge X.$ *open X* ⟹ *open* (*p* ‘ *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*: $\bigwedge X.$ *open X* ⟹ *open* (*p* ‘ *X*)
⟨*proof*⟩

1.11 Closure

lemma *closure-Union*: *closure* ($\bigcup$ *X*) = ($\bigcup$ *x*∈*X*. *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* ⟷ (∀ *p*∈*X*. ∃ *N*. *p*∈*N* ∧ *open N* ∧ *finite* {*i*∈*I*. *U i* ∩ *N* ≠ {}})

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* ∈ *I* ⟹ *finite* {*i*∈*I*. *U i* ∩ *U j* ≠ {}}

and *open-cover*: $X \subseteq (\bigcup i{\in}I.\ U\ i)$ $\bigwedge i.\ i \in I \Longrightarrow open\ (U\ i)$
⟨*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*: *open* $S \Longrightarrow (closure\ T \cap S) = \{\} \longleftrightarrow T \cap S = \{\}$
⟨*proof*⟩

lemma *locally-finite-on-subset*:
assumes *locally-finite-on* $X\ J\ U$
assumes $\bigwedge i.\ i \in I \Longrightarrow V\ i \subseteq U\ i\ I \subseteq J$
shows *locally-finite-on* $X\ I\ V$
⟨*proof*⟩

lemma *locally-finite-on-closure*:
locally-finite-on $X\ I\ (\lambda x.\ closure\ (U\ x))$
if *locally-finite-on* $X\ I\ U$
⟨*proof*⟩

lemma *locally-finite-on-closedin-Union-closure*:
closedin (*top-of-set* X) $(\bigcup i{\in}I.\ closure\ (U\ i))$
if *locally-finite-on* $X\ I\ U$ $\bigwedge i.\ i \in I \Longrightarrow closure\ (U\ i) \subseteq X$
⟨*proof*⟩

lemma *closure-subtopology-minimal*:
$S \subseteq T \Longrightarrow closedin\ (top\text{-}of\text{-}set\ X)\ T \Longrightarrow closure\ S \cap X \subseteq T$
⟨*proof*⟩

lemma *locally-finite-on-closure-Union*:
$(\bigcup i{\in}I.\ closure\ (U\ i)) = closure\ (\bigcup i{\in}I.\ (U\ i)) \cap X$
if *locally-finite-on* $X\ I\ U$ $\bigwedge i.\ i \in I \Longrightarrow closure\ (U\ i) \subseteq X$
⟨*proof*⟩

1.14 Refinement of cover

definition *refines* :: $'a\ set\ set \Rightarrow\ 'a\ set\ set \Rightarrow bool$ (**infix** ‹*refines*› *50*)
where A *refines* $B \longleftrightarrow (\forall s{\in}A.\ (\exists t.\ t \in B \wedge s \subseteq t))$

lemma *refines-subset*: x *refines* y **if** z *refines* y $x \subseteq z$
⟨*proof*⟩

1.15 Functions as vector space

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

definition *scaleR-fun* :: *real* ⇒ ($'a$ ⇒ $'b$) ⇒ $'a$ ⇒ $'b$ **where**
scaleR-fun r f = ($\lambda x.\ r *_R f\ x$)

lemma *scaleR-fun-beta*[*simp*]: $(r *_R f)\ x = r *_R f\ x$
⟨*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*: $(c *_R x) \circ f = c *_R (x \circ f)$
⟨*proof*⟩

lemma *image-scaleR-ball*:
fixes *a* :: $'a$::*real-normed-vector*
shows $c \neq 0 \Longrightarrow (*_R)\ c\ `\ ball\ a\ r = ball\ (c *_R a)\ (abs\ c *_R r)$
⟨*proof*⟩

1.17 Continuity

lemma *continuous-within-topologicalE*:
assumes *continuous* (*at x within s*) *f*
open B f x ∈ *B*
obtains *A* **where** *open A x* ∈ *A* $\bigwedge y.\ y \in s \Longrightarrow y \in A \Longrightarrow f\ y \in B$
⟨*proof*⟩

lemma *continuous-within-topologicalE'*:
assumes *continuous* (*at x*) *f*
open B f x ∈ *B*
obtains *A* **where** *open A x* ∈ *A f* ‘ *A* ⊆ *B*

⟨*proof*⟩

lemma *continuous-on-inverse*: *continuous-on* $S\,f \Longrightarrow 0 \notin f\ `\ S \Longrightarrow$ *continuous-on* $S\ (\lambda x.\ inverse\ (f\,x))$
for f::-⇒-::*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* $(\lambda h.\ x\ a * y'\ h + x'\ h * y\ a))$ (*at a within A*)
if [*derivative-intros*]:
$(x$ *has-derivative* $x')$ (*at a within A*)
$(y$ *has-derivative* $y')$ (*at a within A*)
for $x\ y$::-⇒′a::*real-normed-algebra*
⟨*proof*⟩

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

lemma *sqrt-has-derivative*:
$((\lambda x.\ sqrt\ (f\,x))$ *has-derivative* $(\lambda xa.$ (*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*)

$f\ x = g\ x$
$\forall_F\ x\ in\ (at\ x\ within\ X).\ f\ x = g\ x$
⟨*proof*⟩

lemma *differentiable-within-eqI*: *f differentiable at x within X*
if *g differentiable at x within X* $\bigwedge x.\ x \in X \Longrightarrow f\ x = g\ x$
$x \in X$ *open X*
⟨*proof*⟩

lemma *differentiable-eqI*: *f differentiable at x*
if *g differentiable at x* $\bigwedge x.\ x \in X \Longrightarrow f\ x = g\ x\ x \in X$ *open X*
⟨*proof*⟩

lemma *differentiable-on-eqI*:
f differentiable-on S
if *g differentiable-on S* $\bigwedge x.\ x \in S \Longrightarrow f\ x = g\ x$ *open S*
⟨*proof*⟩

lemma *differentiable-on-comp*: (*f o g*) *differentiable-on S*
if *g differentiable-on S f differentiable-on* (*g ‘ S*)
⟨*proof*⟩

lemma *differentiable-on-comp2*: (*f o g*) *differentiable-on S*
if *f differentiable-on T g differentiable-on S g ‘ 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 $\Longrightarrow$ *y differentiable-on UNIV* $\Longrightarrow$ *x* + *y differentiable-on UNIV*
⟨*proof*⟩

lemma *differentiable-on-mult-fun*[*intro*, *simp*]:
x differentiable-on UNIV $\Longrightarrow$ *y differentiable-on UNIV* $\Longrightarrow$ *x* $*$ *y differentiable-on UNIV*
for *x y*::-⇒'*a*::*real-normed-algebra*
⟨*proof*⟩

lemma *differentiable-on-scaleR-fun*[*intro*, *simp*]:
y differentiable-on UNIV $\Longrightarrow$ $x *_R y$ *differentiable-on UNIV*
⟨*proof*⟩

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

if f *differentiable* (*at* x *within* S) $f\ x \neq 0$
⟨*proof*⟩

lemma *sqrt-differentiable-on*: ($\lambda x.$ *sqrt* ($f\ x$)) *differentiable-on* S
if f *differentiable-on* S $0 \notin f$ ‘ S
⟨*proof*⟩

lemma *differentiable-on-inverse*: f *differentiable-on* $S \Longrightarrow 0 \notin f$ ‘ $S \Longrightarrow$ ($\lambda x.$ *inverse* ($f\ x$)) *differentiable-on* S
for f::-⇒-::*real-normed-field*
⟨*proof*⟩

lemma *differentiable-on-openI*:
f *differentiable-on* S
if *open* S $\bigwedge x.\ x \in S \Longrightarrow \exists f'.$ (f *has-derivative* f') (*at* x)
⟨*proof*⟩

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

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

lemma *differentiable-at-fst*:
($\lambda x.$ *fst* ($f\ x$)) *differentiable at* x *within* X **if** f *differentiable at* x *within* X
⟨*proof*⟩

lemma *differentiable-at-snd*:
($\lambda x.$ *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*: ($\lambda x.$ *sin* ($f\ x$::*real*)) *differentiable at* x *within* X
if f *differentiable at* x *within* X
⟨*proof*⟩

lemma *cos-differentiable-at*: ($\lambda x.$ *cos* ($f\ x$::*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 $\Longrightarrow$ *y differentiable at a* $\Longrightarrow$
  frechet-derivative $(x + y)$ *(at a)* =
    frechet-derivative x (at a) + *frechet-derivative y (at a)*
  ⟨*proof*⟩

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

lemma *frechet-derivative-zero-fun*: *frechet-derivative 0 (at a)* = *0*
  ⟨*proof*⟩

lemma *frechet-derivative-sin*:
  frechet-derivative (λx. *sin* (*f x*)) (*at x*) = (λxa. *frechet-derivative f* (*at x*) *xa* ∗ *cos* (*f x*))
  if *f differentiable* (*at x*)
  for *f*::-⇒*real*
  ⟨*proof*⟩

lemma *frechet-derivative-cos*:
  frechet-derivative (λx. *cos* (*f x*)) (*at x*) = (λxa. *frechet-derivative f* (*at x*) *xa* ∗ − *sin* (*f x*))
  if *f differentiable* (*at x*)
  for *f*::-⇒*real*
  ⟨*proof*⟩

lemma *differentiable-sum-fun*:
  ($\bigwedge i$. $i \in I \Longrightarrow$ (*f i differentiable at a*)) $\Longrightarrow$ *sum f I differentiable at a*
  ⟨*proof*⟩

lemma *frechet-derivative-sum-fun*:
  ($\bigwedge i$. $i \in I \Longrightarrow$ (*f i differentiable at a*)) $\Longrightarrow$
  frechet-derivative ($\sum i{\in}I$. *f i*) (*at a*) = ($\sum i{\in}I$. *frechet-derivative* (*f i*) (*at a*))
  ⟨*proof*⟩

lemma *sum-fun-def*: ($\sum i{\in}I$. *f i*) = (λx. $\sum i{\in}I$. *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 $\Longrightarrow$ *g differentiable at a* $\Longrightarrow$
  frechet-derivative (*f* ∗ *g*) (*at a*) =
  (λx. *f a* ∗ *frechet-derivative g* (*at a*) *x* + *frechet-derivative f* (*at a*) *x* ∗ *g a*)
  for *f g*::-⇒'*a*::*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 $\Longrightarrow$
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) =
$(\lambda v. \sum i{\in}Basis.$ *((frechet-derivative g (at x) v)* $\cdot$ *i)* $*_R$ *frechet-derivative f (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 $\Longrightarrow$ *open X* $\Longrightarrow$ $x \in X \Longrightarrow$
(f has-derivative frechet-derivative f (at x)) (at x)
and *frechet-derivative-works-on*:
f differentiable-on X $\Longrightarrow$ $x \in X \Longrightarrow$
(f has-derivative frechet-derivative f (at x within X)) (at x within X)
⟨*proof*⟩

lemma *frechet-derivative-inverse*: *frechet-derivative* (λx. *inverse (f x)) (at x)* =
$(\lambda h. - 1 / (f\,x)^2 *$ *frechet-derivative f (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* (λx. *sqrt (f x)) (at x)* =
(λv. (*if f x* > *0 then 1 else* $-$*1*) / (*2* $*$ *sqrt (f x)*) $*$ *frechet-derivative f (at x) v*)
if *f differentiable at x f x* $\neq$ *0*
⟨*proof*⟩

lemma *frechet-derivative-norm*: *frechet-derivative* (λx. *norm (f x)) (at x)* =
(λv. *frechet-derivative f (at x) v* $\cdot$ *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 *matrix-mult*
begin
notation *matrix-matrix-mult* (**infixl** ‹∗∗› *70*)
end

lemma (**in** *bounded-bilinear*) *frechet-derivative*:
includes *no matrix-mult*
shows
x differentiable at a $\Longrightarrow$ *y differentiable at a* $\Longrightarrow$
frechet-derivative (λa. *x a* ∗∗ *y a*) (*at a*) =
(λh. *x a* ∗∗ *frechet-derivative y* (*at a*) *h* + *frechet-derivative x* (*at a*) *h* ∗∗ *y a*)
⟨*proof*⟩

lemma *frechet-derivative-divide*: *frechet-derivative* (λx. *f x* / *g x*) (*at x*) =
(λh. *frechet-derivative f* (*at x*) *h* / (*g x*) −*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*::-⇒-::*real-normed-field*
⟨*proof*⟩

lemma *frechet-derivative-pair*:
frechet-derivative (λx. (*f x*, *g x*)) (*at x*) = (λv. (*frechet-derivative f* (*at x*) *v*, *frechet-derivative g* (*at x*) *v*))
if *f differentiable* (*at x*) *g differentiable* (*at x*)
⟨*proof*⟩

lemma *frechet-derivative-fst*:
frechet-derivative (λx. *fst* (*f x*)) (*at x*) = (λxa. *fst* (*frechet-derivative f* (*at x*) *xa*))
if (*f differentiable at x*)
for *f*::-⇒(-::*real-normed-vector* × -::*real-normed-vector*)
⟨*proof*⟩

lemma *frechet-derivative-snd*:
frechet-derivative (λx. *snd* (*f x*)) (*at x*) = (λxa. *snd* (*frechet-derivative f* (*at x*) *xa*))
if (*f differentiable at x*)
for *f*::-⇒(-::*real-normed-vector* × -::*real-normed-vector*)
⟨*proof*⟩

lemma *frechet-derivative-eq-vector-derivative-1*:
assumes *f differentiable at t*
shows *frechet-derivative f* (*at t*) *1* = *vector-derivative f* (*at t*)
⟨*proof*⟩

1.21 Linear algebra

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

assumes *dim* $(UNIV::'b\ set) > 0$
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* $S\ g$ $\bigwedge x.\ x \in S \Longrightarrow f\ x = g\ x$
⟨*proof*⟩

lemma *extensional0-add*[*intro*, *simp*]:
extensional0 $S\ f \Longrightarrow$ *extensional0* $S\ g \Longrightarrow$ *extensional0* $S\ (f + g::\_\Rightarrow 'a::comm\text{-}monoid\text{-}add)$

⟨*proof*⟩

lemma *extensinoal0-mult*[*intro*, *simp*]:
extensional0 S x $\Longrightarrow$ *extensional0 S y* $\Longrightarrow$ *extensional0 S* $(x * y)$
for *x y*::-⇒′*a*::*mult-zero*
⟨*proof*⟩

lemma *extensional0-scaleR*[*intro*, *simp*]: *extensional0 S f* $\Longrightarrow$ *extensional0 S* (c $*_R$ *f*::-⇒′*a*::*real-vector*)
⟨*proof*⟩

lemma *extensional0-outside*: $x \notin S \Longrightarrow$ *extensional0 S f* $\Longrightarrow f\ x = 0$
⟨*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* ⇒ (′*a* ⇒ ′*b*::*zero*) ⇒ ′*a* ⇒ ′*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*⇒′*b*::*mult-zero*
⟨*proof*⟩

lemma *restrict0-apply-in*[*simp*]: $x \in A \Longrightarrow$ *restrict0 A f x* = *f x*
⟨*proof*⟩

lemma *restrict0-apply-out*[*simp*]: $x \notin A \Longrightarrow$ *restrict0 A f x* = *0*
⟨*proof*⟩

lemma *restrict0-scaleR*: *restrict0 A* (c $*_R$ *f*::-⇒′*a*::*real-vector*) = c $*_R$ *restrict0 A f*
⟨*proof*⟩

lemma *restrict0-add*: *restrict0 A* (*f* + *g*::-⇒′*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 $f::'a::real\text{-}normed\text{-}vector \Rightarrow 'b::banach$
    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*: $\bigwedge a\ i\ d.\ a \in closed\text{-}segment\ X\ (X + H) \Longrightarrow i < n \Longrightarrow$
    $((\lambda a.\ Df\ a\ i\ H)\ has\text{-}derivative\ (Df\ a\ (Suc\ i)))\ (at\ a\ within\ G)$
  assumes *cs*: $closed\text{-}segment\ X\ (X + H) \subseteq G$
  defines $i \equiv \lambda x.$
    $((1 - x)\ \hat{}\ (n - 1)\ /\ fact\ (n - 1)) *_R Df\ (X + x *_R H)\ n\ H$
  shows *multivariate-Taylor-has-integral*:
    $(i\ has\text{-}integral\ f\ (X + H) - (\sum i{<}n.\ (1\ /\ fact\ i) *_R Df\ X\ i\ H))\ \{0..1\}$
  and *multivariate-Taylor*:
    $f\ (X + H) = (\sum i{<}n.\ (1\ /\ fact\ i) *_R Df\ X\ i\ H) + integral\ \{0..1\}\ i$
  and *multivariate-Taylor-integrable*:
    $i\ integrable\text{-}on\ \{0..1\}$
$\langle proof \rangle$

2.2 Higher-order differentiable

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

lemma *ball-differentiable-atD*: $\forall x{\in}S.\ f\ differentiable\ at\ x \Longrightarrow f\ differentiable\text{-}on\ S$
  $\langle proof \rangle$

lemma *higher-differentiable-on-imp-continuous-on*:
  $continuous\text{-}on\ S\ f$ **if** $higher\text{-}differentiable\text{-}on\ S\ f\ n$
  $\langle proof \rangle$

lemma *higher-differentiable-on-imp-differentiable-on*:
  $f\ differentiable\text{-}on\ S$ **if** $higher\text{-}differentiable\text{-}on\ S\ f\ k\ k > 0$
  $\langle proof \rangle$

lemma *higher-differentiable-on-congI*:
  assumes *open S higher-differentiable-on S g n*
    and $\bigwedge x.\ x \in S \Longrightarrow f\ x = g\ x$
  shows *higher-differentiable-on S f n*
  ⟨*proof*⟩

lemma *higher-differentiable-on-cong*:
  assumes *open S* $S = T$
    and $\bigwedge x.\ x \in T \Longrightarrow f\ x = g\ x$
  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 $\bigwedge x.\ x \in S \Longrightarrow \exists\, T.\ x \in T \wedge$ *open T* $\wedge$ *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*:
 $(\lambda x.\ prod\ (f\ x)\ (g\ x))$ *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*:
  $(\lambda x.\ prod\ (f\ x)\ (g\ x))$ *differentiable-on S*
  if *f differentiable-on S g differentiable-on S*
  ⟨*proof*⟩

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*:
  higher-differentiable-on S $(\lambda x.\ prod\ (f\ x)\ (g\ x))$ *n*
  if
    higher-differentiable-on S f n
    higher-differentiable-on S g n
    open S
  ⟨*proof*⟩

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*:
  higher-differentiable-on S $(\lambda x.\ \sum i{\in}F.\ f\ i\ x)$ *n*
  if $\bigwedge i.\ i \in F \Longrightarrow$ *finite F* $\Longrightarrow$ *higher-differentiable-on S* $(f\ i)$ *n open S*
  ⟨*proof*⟩

lemma *higher-differentiable-on-subset*:
  higher-differentiable-on S f n
  if *higher-differentiable-on T f n* $S \subseteq T$
  ⟨*proof*⟩

lemma *higher-differentiable-on-compose*:
  higher-differentiable-on S $(f\ o\ g)$ *n*
  if *higher-differentiable-on T f n higher-differentiable-on S g n* g ‘ $S \subseteq T$ *open S*
open T

for g::-⇒-::*euclidean-space*— TODO: can we get around this restriction?
⟨*proof*⟩

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

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
⟨*proof*⟩

lemma *higher-differentiable-on-inverse*:
higher-differentiable-on S ($\lambda x.$ *inverse* ($f\ x$)) *n*
if *higher-differentiable-on S f n* $0 \notin f$ ‘ S *open S*
for f::-⇒-::*real-normed-field*
⟨*proof*⟩

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
$\bigwedge x.\ x \in S \Longrightarrow g\ x \neq 0$
open S
for f::-⇒-::*real-normed-field*
⟨*proof*⟩

lemma *differentiable-on-open-Union*:
f differentiable-on $\bigcup S$
if $\bigwedge s.\ s \in S \Longrightarrow$ *f differentiable-on s*
$\bigwedge s.\ s \in S \Longrightarrow$ *open s*
⟨*proof*⟩

lemma *higher-differentiable-on-open-Union*: *higher-differentiable-on* ($\bigcup S$) *f n*
if $\bigwedge s.\ s \in S \Longrightarrow$ *higher-differentiable-on s f n*
$\bigwedge s.\ s \in S \Longrightarrow$ *open s*
⟨*proof*⟩

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

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

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

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

lemma *higher-differentiable-on-Pair*:
higher-differentiable-on S f k $\Longrightarrow$ *higher-differentiable-on S g k* $\Longrightarrow$
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*::-⇒-::*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 $(\lambda x.$ *fst* $(f\ x))$ *k*
if *higher-differentiable-on S f k open S*

⟨*proof*⟩

lemma *higher-differentiable-on-snd-comp*:
higher-differentiable-on S ($\lambda x.$ *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 $\Longrightarrow$ *higher-differentiable-on T g k* $\Longrightarrow$
higher-differentiable-on ($S \times T$) ($\lambda x.$ (*f* (*fst x*), *g* (*snd x*))) *k*
if *S*: *open S* **and** *T*: *open T*
for *f*::-::*euclidean-space*⇒- **and** *g*::-::*euclidean-space*⇒-
⟨*proof*⟩

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

2.3 Higher directional derivatives

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

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

lemma *nth-derivative-funpow*:
nth-derivative i f x h = (($\lambda f\ x.$ *frechet-derivative f* (*at x*) *h*) ^^ *i*) *f x*
⟨*proof*⟩

lemma *nth-derivative-exists*:
$\exists f'.$ (($\lambda x.$ *nth-derivative i f x h*) *has-derivative* f') (*at x*) $\wedge$
f' *h* = *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*
$\bigwedge a\ h\ i.$ $i < n \Longrightarrow a \in X \Longrightarrow$ (($\lambda a.$ *Df a i H*) *has-derivative Df a* (*Suc i*)) (*at*

a)
  $\bigwedge a\ i.\ i \le n \Longrightarrow a \in X \Longrightarrow Df\ a\ i\ H = nth\text{-}derivative\ i\ f\ a\ H$
⟨*proof*⟩

lemma *nth-derivative-differentiable*:
  assumes *higher-differentiable-on S f* (*Suc n*) $x \in S$
  shows (λx. *nth-derivative n f x v*) *differentiable at x*
  ⟨*proof*⟩

lemma *higher-differentiable-on-imp-continuous-nth-derivative*:
  assumes *higher-differentiable-on S f n*
  shows *continuous-on S* (λx. *nth-derivative n f x 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*) $\longleftrightarrow$
    ($\forall x \in S$. *f differentiable* (*at x*)) $\wedge$
     (*higher-differentiable-on S* (λx. *frechet-derivative f* (*at x*) *1*) *n*)
  if *open S*
  for *S*::*real set*
  ⟨*proof*⟩

lemma *higher-differentiable-on-real-SucI*:
  fixes *S*::*real set*
  assumes
    $\bigwedge x.\ x \in S \Longrightarrow$ (λx. *nth-derivative n f x 1*) *differentiable at x*
    continuous-on S (λx. *nth-derivative* (*Suc n*) *f x 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 $\Longrightarrow$ *higher-differentiable-on S f* (*Suc n*) $\longleftrightarrow$
    ($\forall v$. *continuous-on S* (λx. *nth-derivative* (*Suc n*) *f x 1*)) $\wedge$
   ($\forall x \in S$. $\forall v$. (λx. *nth-derivative n f x 1*) *differentiable* (*at x*)) $\wedge$ *higher-differentiable-on S f n*
  for *S*::*real set*
  ⟨*proof*⟩

lemma *closed-segment-subsetD*:
  $0 \le x \Longrightarrow x \le 1 \Longrightarrow (X + x *_R H) \in S$
  if *closed-segment X* (*X* + *H*) $\subseteq S$
  ⟨*proof*⟩

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

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

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

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

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

lemma *continuous-on-blinfunI1*:

continuous-on X f
if $\bigwedge i.\ i \in Basis \Longrightarrow$ *continuous-on X* ($\lambda x.$ *blinfun-apply* (*f x*) *i*)
⟨*proof*⟩

lemma *c1-euclidean-blinfunE*:
fixes *f*::′*a*::*euclidean-space*⇒′*b*::*real-normed-vector*
assumes $\bigwedge x.\ x \in S \Longrightarrow$ (*f has-derivative f′ x*) (*at x within S*)
assumes $\bigwedge i.\ i \in Basis \Longrightarrow$ *continuous-on S* ($\lambda x.$ *f′ x i*)
obtains *bf′* **where**
$\bigwedge x.\ x \in S \Longrightarrow$ (*f has-derivative blinfun-apply* (*bf′ x*)) (*at x within S*)
continuous-on S bf′
$\bigwedge x.\ x \in S \Longrightarrow$ *blinfun-apply* (*bf′ x*) = *f′ x*
⟨*proof*⟩

lemma *continuous-Sigma*:
assumes *defined*: $y \in Pi\ T\ X$
assumes *f-cont*: *continuous-on* (*Sigma T X*) ($\lambda(t, x).$ *f t x*)
assumes *y-cont*: *continuous-on T y*
shows *continuous-on T* ($\lambda x.$ *f x* (*y x*))
⟨*proof*⟩

lemma *continuous-on-Times-swap*:
continuous-on ($X \times Y$) ($\lambda(x, y).$ *f x y*)
if *continuous-on* ($Y \times X$) ($\lambda(y, x).$ *f x y*)
⟨*proof*⟩

lemma *leibniz-rule′*:
$\bigwedge x.\ x \in S \Longrightarrow$
((λ*x. integral* (*cbox a b*) (*f x*)) *has-derivative* (λ*v. integral* (*cbox a b*) (λ*t. fx x t v*)))
(*at x within S*)
(λ*x. integral* (*cbox a b*) (*f x*)) *differentiable-on S*
if *convex S*
and *c1*: $\bigwedge t\ x.\ t \in cbox\ a\ b \Longrightarrow x \in S \Longrightarrow$ ((λ*x. f x t*) *has-derivative fx x t*) (*at x within S*)
$\bigwedge i.\ i \in Basis \Longrightarrow$ *continuous-on* ($S \times$ *cbox a b*) ($\lambda(x, t).$ *fx x t i*)
and *i*: $\bigwedge x.\ x \in S \Longrightarrow$ *f x integrable-on cbox a b*
for *S*::′*a*::*euclidean-space set*
and *f*::′*a* ⇒ ′*b*::*euclidean-space* ⇒ ′*c*::*euclidean-space*
⟨*proof*⟩

lemmas *leibniz-rule′-interval* = *leibniz-rule′*[**where** ′*b*=-::*ordered-euclidean-space*,
unfolded cbox-interval]

lemma *leibniz-rule′-higher*:
higher-differentiable-on S (λ*x. integral* (*cbox a b*) (*f x*)) *k*
if *convex S open S*
and *c1*: *higher-differentiable-on* (*S*×*cbox a b*) ($\lambda(x, t).$ *f 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::$'a$::*euclidean-space set*
    and f::$'a \Rightarrow$ $'b$::*euclidean-space* $\Rightarrow$ $'c$::*euclidean-space*
  ⟨*proof*⟩

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

2.4 Smoothness

definition *k-smooth-on* :: *enat* ⇒$'a$::*real-normed-vector set* ⇒ ($'a$ ⇒ $'b$::*real-normed-vector*) ⇒ *bool*
  (‹-−*smooth′-on*› [*1000*]) **where**
 smooth-on-def: *k*−*smooth-on S f* = (∀ *n*≤*k*. *higher-differentiable-on S f n*)

abbreviation *smooth-on S f* ≡ ∞−*smooth-on S f*

lemma *derivative-is-smooth′*:
  assumes (*k*+*1*)−*smooth-on S f*
  shows *k*−*smooth-on S* (λ*x*. *frechet-derivative f* (*at x*) *v*)
  ⟨*proof*⟩

lemma *derivative-is-smooth*: *smooth-on S f* ⟹ *smooth-on S* (λ*x*. *frechet-derivative f* (*at x*) *v*)
  ⟨*proof*⟩

lemma *smooth-on-imp-continuous-on*: *continuous-on S f* **if** *k*−*smooth-on S f*
  ⟨*proof*⟩

lemma *smooth-on-imp-differentiable-on*[*simp*]: *f differentiable-on S* **if** *k*−*smooth-on S f k* ≠ *0*
  ⟨*proof*⟩

lemma *smooth-on-cong*:
  assumes *k*−*smooth-on S g open S*
    and ⋀*x*. *x* ∈ *S* ⟹ *f x* = *g x*
  shows *k*−*smooth-on S f*
  ⟨*proof*⟩

lemma *smooth-on-open-Un*:
  k−*smooth-on S f* ⟹ *k*−*smooth-on T f* ⟹ *open S* ⟹ *open T* ⟹ *k*−*smooth-on* (*S* ∪ *T*) *f*
  ⟨*proof*⟩

lemma *smooth-on-open-subsetsI*:
  k−*smooth-on S f*
  if ⋀*x*. *x* ∈ *S* ⟹ ∃ *T*. *x* ∈ *T* ∧ *open T* ∧ *k*−*smooth-on T f*
  ⟨*proof*⟩

lemma *smooth-on-const*[*intro*]: k−*smooth-on* S ($\lambda x.\ c$)
⟨*proof*⟩

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

lemma *smooth-on-add-fun*: k−*smooth-on* S f $\Longrightarrow$ k−*smooth-on* S g $\Longrightarrow$ *open* S $\Longrightarrow$ k−*smooth-on* S (f + g)
⟨*proof*⟩

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

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

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

lemma *smooth-on-compose2*:
fixes g::-⇒-::*euclidean-space*
assumes k−*smooth-on* T f k−*smooth-on* S g *open* U *open* T g ‘ $U \subseteq T$ $U \subseteq S$
shows k−*smooth-on* U (f *o* g)
⟨*proof*⟩

lemma *smooth-on-compose*:
fixes g::-⇒-::*euclidean-space*
assumes k−*smooth-on* T f k−*smooth-on* S g *open* S *open* T g ‘ $S \subseteq T$
shows k−*smooth-on* S (f *o* g)
⟨*proof*⟩

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

context begin
private lemmas s = *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−*smooth-on* S f $\Longrightarrow$ k−*smooth-on* S g $\Longrightarrow$ *open* S $\Longrightarrow$($\bigwedge x.$

$x \in S \Longrightarrow g\ x \neq 0) \Longrightarrow$
  k−*smooth-on* S ($\lambda x.\ f\ x\ /\ g\ x$)
  for f::-⇒-::*real-normed-field*
  ⟨*proof*⟩

lemma *smooth-on-scaleR-fun*: k−*smooth-on* S $g \Longrightarrow$ *open* $S \Longrightarrow k$−*smooth-on* S $(c *_R g)$
  ⟨*proof*⟩

lemma *smooth-on-uminus-fun*: k−*smooth-on* S $g \Longrightarrow$ *open* $S \Longrightarrow k$−*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 $f \Longrightarrow k$−*smooth-on* S $g \Longrightarrow$ *open* $S \Longrightarrow k$−*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 $f \Longrightarrow k$−*smooth-on* S $g \Longrightarrow$ *open* $S \Longrightarrow k$−*smooth-on* S $(f * g)$
  for f g::- ⇒-::*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 ($\lambda x.$ *inverse* $(f\ x)$)
  if k−*smooth-on* S f $0 \notin f$ ‘ S *open* S
  for f::- ⇒-::*real-normed-field*
  ⟨*proof*⟩

lemma *smooth-on-norm*: k−*smooth-on* S ($\lambda x.$ *norm* $(f\ x)$)
  if k−*smooth-on* S f $0 \notin f$ ‘ S *open* S
  for f::- ⇒-::*real-inner*
  ⟨*proof*⟩

lemma *smooth-on-sqrt*: k−*smooth-on* S ($\lambda x.$ *sqrt* $(f\ x)$)
  if k−*smooth-on* S f $0 \notin f$ ‘ S *open* S
  ⟨*proof*⟩

lemma *smooth-on-frechet-derivative*:
  ∞−*smooth-on UNIV* ($\lambda x.$ *frechet-derivative* f (*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* ≤ *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* (λ*x*. *snd* (*f x*))
if *k*−*smooth-on S f open S*
⟨*proof*⟩

lemma *smooth-on-fst*:
k−*smooth-on S* (λ*x*. *fst* (*f x*))
if *k*−*smooth-on S f open S*
⟨*proof*⟩

lemma *smooth-on-sin*: *n*−*smooth-on S* (λ*x*. *sin* (*f x*::*real*)) **if** *n*−*smooth-on S f open S*
⟨*proof*⟩

lemma *smooth-on-cos*: *n*−*smooth-on S* (λ*x*. *cos* (*f x*::*real*)) **if** *n*−*smooth-on S f open S*
⟨*proof*⟩

lemma *smooth-on-Taylor2E*:
fixes *f*::′*a*::*euclidean-space* ⇒ *real*
assumes *hd*: ∞−*smooth-on UNIV f*
obtains *g* **where** ⋀*Y*.
f Y = *f X* + *frechet-derivative f* (*at X*) (*Y* − *X*) + (∑*i*∈*Basis*. (∑*j*∈*Basis*. ((*Y* − *X*) · *j*) ∗ ((*Y* − *X*) · *i*) ∗ *g i j Y*))
⋀*i j*. *i* ∈ *Basis* ⟹ *j* ∈ *Basis* ⟹ ∞−*smooth-on UNIV* (*g i j*)
— TODO: generalize
⟨*proof*⟩

lemma *smooth-on-Pair*:
k−*smooth-on S* (λ*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 $(S \times T)$ $(\lambda x.\ (f\ (fst\ x),\ g\ (snd\ x)))$
if *open S open T k−smooth-on S f k−smooth-on T g*
for *f*::-::*euclidean-space*⇒- **and** *g*::-::*euclidean-space*⇒-
⟨*proof*⟩

2.5 Diffeomorphism

definition *diffeomorphism k S T p p′* ⟷
k−smooth-on S p ∧ *k−smooth-on T p′* ∧ *homeomorphism S T p p′*

lemma *diffeomorphism-imp-homeomorphism*:
assumes *diffeomorphism k s t p p′*
shows *homeomorphism s t p p′*
⟨*proof*⟩

lemma *diffeomorphismD*:
assumes *diffeomorphism k S T f g*
shows *diffeomorphism-smoothD*: *k−smooth-on S f k−smooth-on T g*
and *diffeomorphism-inverseD*: $\bigwedge x.\ x \in S \Longrightarrow g\ (f\ x) = x\ \bigwedge y.\ y \in T \Longrightarrow f\ (g\ y) = y$
and *diffeomorphism-image-eq*: $(f\ \text{‘}\ S = T)\ (g\ \text{‘}\ T = S)$
⟨*proof*⟩

lemma *diffeomorphism-compose*:
diffeomorphism n S T f g ⟹ *diffeomorphism n T U h k* ⟹ *open S* ⟹ *open T* ⟹ *open U* ⟹
diffeomorphism n S U $(h \circ f)\ (g \circ k)$
for *f*::-⇒-::*euclidean-space*
⟨*proof*⟩

lemma *diffeomorphism-add*: *diffeomorphism k UNIV UNIV* $(\lambda x.\ x + c)\ (\lambda x.\ x - c)$
⟨*proof*⟩

lemma *diffeomorphism-scaleR*: *diffeomorphism k UNIV UNIV* $(\lambda x.\ c *_R x)\ (\lambda x.\ x /_R c)$
if $c \neq 0$
⟨*proof*⟩

end

3 Bump Functions

theory *Bump-Function*
imports *Smooth*
HOL−Analysis.Weierstrass-Theorems
begin

3.1 Construction

context begin

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

lemma *f-nonpos*[*simp*]: $x \leq 0 \Longrightarrow f\ x = 0$
⟨*proof*⟩

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

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

lemma *exp-inv-limit-0-right-gen′*:
$((\lambda(t::real).\ inverse\ (t\ \hat{}\ k)\ /\ exp(inverse\ t)) \longrightarrow 0)\ (at\text{-}right\ 0)$
⟨*proof*⟩

lemma *exp-inv-limit-0-right-gen*:
$((\lambda(t::real).\ exp(-inverse\ t)\ /\ t\ \hat{}\ k) \longrightarrow 0)\ (at\text{-}right\ 0)$
⟨*proof*⟩

lemma *f-limit-0-right*: $(f \longrightarrow 0)\ (at\text{-}right\ 0)$
⟨*proof*⟩

lemma *f-limit-0*: $(f \longrightarrow 0)\ (at\ 0)$
⟨*proof*⟩

lemma *f-tendsto*: $(f \longrightarrow f\ x)\ (at\ x)$
⟨*proof*⟩

lemma *f-continuous*: *continuous-on S f*
⟨*proof*⟩

lemma *continuous-on-real-polynomial-function*:
continuous-on S p **if** *real-polynomial-function p*
⟨*proof*⟩

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

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

$\langle proof \rangle$

lemma *f-differentiable-at-neg*:
$x < 0 \Longrightarrow f$ *differentiable at* x
$\langle proof \rangle$

lemma *frechet-derivative-f-at-neg*:
$x \in \{..<0\} \Longrightarrow$ *frechet-derivative* f (*at* x) $= (\lambda x.\ 0)$
$\langle proof \rangle$

lemma *f-nth-derivative-lt-0*:
higher-differentiable-on $\{..<0\}$ f k $\wedge$ ($\forall t<0.$ *nth-derivative* k f t $1 = 0$)
$\langle proof \rangle$

lemma *netlimit-at-left*: *netlimit* (*at-left* x) $= x$ **for** x::*real*
$\langle proof \rangle$

lemma *netlimit-at-right*: *netlimit* (*at-right* x) $= x$ **for** x::*real*
$\langle proof \rangle$

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::*real*
$\langle proof \rangle$

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::*real*
$\langle proof \rangle$

lemma *real-polynomial-function-tendsto*:
($p \longrightarrow p\ x$) (*at* x *within* X) **if** *real-polynomial-function* p
$\langle proof \rangle$

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

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*: $f\ (2 - t) + f\ (t - 1) = 0 \longleftrightarrow$ *False*
⟨*proof*⟩

lemma *h-range*: $0 \leq h\ t\ h\ t \leq 1$
⟨*proof*⟩

lemma *h-pos*: $t < 2 \Longrightarrow 0 < h\ t$
and *h-less-one*: $1 < t \Longrightarrow h\ t < 1$
⟨*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*::-::*real-inner* ⇒ *real* **where** $H\ x = h$ (*norm* x)

lemma *H-range*: $0 \leq H\ x\ H\ x \leq 1$
⟨*proof*⟩

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

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

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

lemma *H-neq-zeroD*: $H\ x \neq 0 \Longrightarrow x \in ball\ 0\ 2$
⟨*proof*⟩

lemma *H-smooth-on*: *k−smooth-on UNIV H*
⟨*proof*⟩

lemma *H-compose-smooth-on*: *k−smooth-on S* (*λx. H* (*g x*)) **if** *k−smooth-on S g* *open S*
for *g* :: - ⇒ -::*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: $d \to d'$ is the mapping, and f': $d' \to d$ is the inverse mapping.

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

setup-lifting *type-definition-chart*

lift-definition *apply-chart*::(*'a*::*topological-space*, *'e*::*euclidean-space*) *chart* ⇒ *'a* ⇒ *'e*
is λ(*d*, *d'*, *f*, *f'*). *f* ⟨*proof*⟩

declare [[*coercion 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** $x \in$ *domain c*
⟨*proof*⟩

lemma *at-within-codomain*: *at x within codomain c* = *at x* **if** $x \in$ *codomain c*
⟨*proof*⟩

lemma
chart-in-codomain[*intro*, *simp*]: $x \in$ *domain c* $\Longrightarrow$ *c x* $\in$ *codomain c*
and *inv-chart-inverse*[*simp*]: $x \in$ *domain c* $\Longrightarrow$ *inv-chart c* (*c x*) = *x*
and *inv-chart-in-domain*[*intro*, *simp*]:$y \in$ *codomain c* $\Longrightarrow$ *inv-chart c y* $\in$ *domain c*
and *chart-inverse-inv-chart*[*simp*]: $y \in$ *codomain c* $\Longrightarrow$ *c* (*inv-chart c y*) = *y*
and *image-domain-eq*: *c* ‘ (*domain c*) = *codomain c*
and *inv-image-codomain-eq*[*simp*]: *inv-chart c* ‘ (*codomain c*) = *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*
$\bigwedge x.$ *inv-chart c x* = *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** $x \in$ *domain c*

⟨*proof*⟩

lemma *continuous-inv-chart*: *continuous* (*at* *x* *within* *X*) (*inv-chart* *c*) **if** $x \in$ *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* $\Longrightarrow$ *f* ‘ *s* $\subseteq$ *t* $\Longrightarrow$
continuous (*at x within s*) *f* $\Longrightarrow$
continuous (*at x within s*) ($\lambda x.$ *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* $\subseteq$ *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 $\subseteq$ *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* ‘ ($X \cap Y$) = *f* ‘ $X \cap$ *f* ‘ *Y* **if** $X \subseteq$ *domain f* $Y \subseteq$ *domain f*
⟨*proof*⟩

lemma *chart-image-eq-vimage*: *c* ‘ *X* = *inv-chart c* −‘ $X \cap$ *codomain c*
if $X \subseteq$ *domain c*
⟨*proof*⟩

lemma *open-chart-image*[*simp*, *intro*]: *open* (*c* ‘ *X*)
if *open X* $X \subseteq$ *domain c*
⟨*proof*⟩

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

if *open X X* $\subseteq$ *codomain c*
⟨*proof*⟩

lemma *homeomorphism-UNIV-imp-open-map*:
homeomorphism UNIV UNIV p p' $\Longrightarrow$ *open f'* $\Longrightarrow$ *open* (*p* ‘ *f'*)
⟨*proof*⟩

4.3 Restriction

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

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

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

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

4.4 Composition

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

lemma *compose-chart-apply-chart*[*simp*]: *apply-chart* (*compose-chart p p' c*) = *p o apply-chart c*
and *compose-chart-inv-chart*[*simp*]: *inv-chart* (*compose-chart p p' c*) = *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* = ($\bigcup$(*domain* ‘ *charts*))

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

lemma *carrierE*:
assumes $x \in$ *carrier*
obtains *c* **where** $c \in$ *charts* $x \in$ *domain c*
⟨*proof*⟩

lemma *domain-subset-carrier*[*simp*]: *domain c* $\subseteq$ *carrier* **if** $c \in$ *charts*
⟨*proof*⟩

lemma *in-domain-in-carrier*[*intro*, *simp*]: $c \in$ *charts* $\Longrightarrow$ $x \in$ *domain c* $\Longrightarrow$ $x \in$ *carrier*
⟨*proof*⟩

5.2 Existence of locally finite cover

Every point has a precompact neighborhood.

lemma *precompact-neighborhoodE*:
  assumes $x \in$ *carrier*
  obtains C **where** $x \in C$ *open* C *compact* (*closure* C) *closure* $C \subseteq$ *carrier*
⟨*proof*⟩

There exists a covering of the carrier by precompact sets.

lemma *precompact-open-coverE*:
  obtains U::*nat*⇒′*a set*
  where $(\bigcup i.\ U\ i)$ = *carrier* $\bigwedge i.$ *open* $(U\ i)$ $\bigwedge i.$ *compact* (*closure* $(U\ i)$)
    $\bigwedge i.$ *closure* $(U\ i) \subseteq$ *carrier*
⟨*proof*⟩

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

lemma *precompact-locally-finite-open-coverE*:
  obtains W::*nat*⇒′*a set*
  where *carrier* = $(\bigcup i.\ W\ i)$ $\bigwedge i.$ *open* $(W\ i)$ $\bigwedge i.$ *compact* (*closure* $(W\ i)$)
    $\bigwedge i.$ *closure* $(W\ i) \subseteq$ *carrier*
    locally-finite-on carrier UNIV W
⟨*proof*⟩

end

end

6 Differentiable/Smooth Manifolds

theory *Differentiable-Manifold*
  imports
    Smooth
    Topological-Manifold
begin

6.1 Smooth compatibility

definition *smooth-compat*::*enat* ⇒ (′*a*::*topological-space*, ′*e*::*euclidean-space*)*chart*⇒(′*a*, ′*e*)*chart*⇒*bool*
  (‹-−*smooth*′-*compat*› [*1000*])
  where
  smooth-compat k c1 c2 ⟷
    (*k*−*smooth-on* (*c1* ‘ (*domain c1* ∩ *domain c2*)) (*c2* ∘ *inv-chart c1*) ∧
     k−*smooth-on* (*c2* ‘ (*domain c1* ∩ *domain c2*)) (*c1* ∘ *inv-chart c2*))

lemma *smooth-compat-D1*:
  k−*smooth-on* (*c1* ‘ (*domain c1* ∩ *domain c2*)) (*c2* ∘ *inv-chart c1*)
  if *k*−*smooth-compat c1 c2*
⟨*proof*⟩

lemma *smooth-compat-D2*:

k−smooth-on (*c2* ‘ (*domain c1* ∩ *domain c2*)) (*c1* ∘ *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* ⟷ *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 ⊆ *U* ⟹ *open U* ⟹ *k−smooth-compat c1* (*restrict-chart U c2*) ⟹ *k−smooth-compat c1 c2*
⟨*proof*⟩

lemma *smooth-compat-restrict-chartD2*:
domain c1 ⊆ *U* ⟹ *open U* ⟹ *k−smooth-compat* (*restrict-chart U c2*) *c1* ⟹ *k−smooth-compat c2 c1*
⟨*proof*⟩

lemma *smooth-compat-le*:
l−smooth-compat c1 c2 **if** *k−smooth-compat c1 c2 l* ≤ *k*
⟨*proof*⟩

6.2 *C*^*k*-Manifold

locale *c-manifold* = *manifold* +
fixes *k*::*enat*
assumes *pairwise-compat*: *c1* ∈ *charts* ⟹ *c2* ∈ *charts* ⟹ *k−smooth-compat c1 c2*
begin

6.2.1 Atlas

definition *atlas* :: (′*a*, ′*b*) *chart set* **where**
atlas = {*c*. *domain c* ⊆ *carrier* ∧ (∀ *c′* ∈ *charts*. *k−smooth-compat c c′*)}

lemma *charts-subset-atlas*: *charts* ⊆ *atlas*
⟨*proof*⟩

lemma *in-charts-in-atlas*[*intro*]: $x \in charts \Longrightarrow x \in atlas$
⟨*proof*⟩

lemma *maximal-atlas*:
$c \in atlas$
if $\bigwedge c'.\ c' \in atlas \Longrightarrow k{-}smooth\text{-}compat\ c\ c'$
domain c $\subseteq$ *carrier*
⟨*proof*⟩

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

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

lemma *maximal-atlas'*:
$c \in atlas$
if $\bigwedge c'.\ c' \in charts \Longrightarrow k{-}smooth\text{-}compat\ c\ c'$
domain c $\subseteq$ *carrier*
⟨*proof*⟩

lemma *atlas-is-atlas*: $k{-}smooth\text{-}compat\ a1\ a2$
if $a1 \in atlas\ a2 \in atlas$
⟨*proof*⟩

lemma *domain-atlas-subset-carrier*: $c \in atlas \Longrightarrow domain\ c \subseteq carrier$
and *in-carrier-atlasI*[*intro*, *simp*]: $c \in atlas \Longrightarrow x \in domain\ c \Longrightarrow x \in carrier$
⟨*proof*⟩

lemma *atlasE*:
assumes $x \in carrier$
obtains *c* **where** $c \in atlas\ x \in domain\ c$
⟨*proof*⟩

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

lemma *atlas-restrictE*:

assumes $x \in$ *carrier* $x \in X$ *open* X
obtains c **where** $c \in$ *atlas* $x \in$ *domain* c *domain* $c \subseteq X$
⟨*proof*⟩

lemma *open-ball-chartE*:
assumes $x \in U$ *open* U $U \subseteq$ *carrier*
obtains c r **where**
$c \in$ *atlas*
$x \in$ *domain* c *domain* $c \subseteq U$ *codomain* $c =$ *ball* $(c\ x)$ r $r > 0$
⟨*proof*⟩

lemma *smooth-compat-compose-chart*:
fixes c'
assumes k–*smooth-compat* c c'
assumes *diffeo*: *diffeomorphism* k *UNIV UNIV* p p'
shows k–*smooth-compat* (*compose-chart* p p' c) c'
⟨*proof*⟩

lemma *compose-chart-in-atlas*:
assumes $c \in$ *atlas*
assumes *diffeo*: *diffeomorphism* k *UNIV UNIV* p p'
shows *compose-chart* p p' $c \in$ *atlas*
⟨*proof*⟩

lemma *open-centered-ball-chartE*:
assumes $x \in U$ *open* U $U \subseteq$ *carrier* $e > 0$
obtains c **where**
$c \in$ *atlas* $x \in$ *domain* c $c\ x = x0$ *domain* $c \subseteq U$ *codomain* $c =$ *ball* $x0$ e
⟨*proof*⟩

end

6.2.2 Submanifold

definition (**in** *manifold*) *charts-submanifold* S = (*restrict-chart* S ‘ *charts*)

locale *c-manifold'* = *c-manifold*

locale *submanifold* = *c-manifold'* *charts* k — breaks infinite loop for sublocale sub
for *charts*::('*a*::{*t2-space*,*second-countable-topology*}, '*b*::*euclidean-space*) *chart set*
and k +
fixes S::'*a set*
assumes *open-submanifold*: *open* S
begin

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

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* ⇒ *'b*)
assumes *exists-smooth-on*: *x* ∈ *src.carrier* ⟹
∃ *c1*∈*src.atlas*. ∃ *c2*∈*dest.atlas*.
x ∈ *domain c1* ∧
f ‘ *domain c1* ⊆ *domain c2* ∧
k−*smooth-on* (*codomain c1*) (*c2* ∘ *f* ∘ *inv-chart c1*)
begin

lemma *defined*: *f* ‘ *src.carrier* ⊆ *dest.carrier*
⟨*proof*⟩

end

context *c-manifolds* **begin**

lemma *diff-iff*: *diff k charts1 charts2 f* ⟷
(∀ *x*∈*src.carrier*. ∃ *c1*∈*src.atlas*. ∃ *c2*∈*dest.atlas*.
x ∈ *domain c1* ∧
f ‘ *domain c1* ⊆ *domain c2* ∧
k−*smooth-on* (*codomain c1*) (*apply-chart c2* ∘ *f* ∘ *inv-chart c1*))
(**is** *?l* ⟷ (∀ *x*∈-. *?r x*))
⟨*proof*⟩

end

context *diff* **begin**

lemma *diffE*:
assumes *x* ∈ *src.carrier*
obtains *c1*::(*'a*, *'e*) *chart*
and *c2*::(*'b*, *'f*) *chart*
where
c1 ∈ *src.atlas c2* ∈ *dest.atlas x* ∈ *domain c1 f* ‘ *domain c1* ⊆ *domain c2*
k−*smooth-on* (*codomain c1*) (*apply-chart c2* ∘ *f* ∘ *inv-chart c1*)
⟨*proof*⟩

lemma *continuous-at*: *continuous* (*at x within T*) *f* **if** *x* ∈ *src.carrier*
⟨*proof*⟩

lemma *continuous-on*: *continuous-on src.carrier f*
⟨*proof*⟩

lemmas *continuous-on-intro*[*continuous-intros*] = *continuous-on-compose2*[*OF continuous-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* ∈ *src.atlas* *d2* ∈ *dest.atlas*
  shows *k*−*smooth-on* (*codomain d1* ∩ *inv-chart d1* −‘ (*src.carrier* ∩ *f* −‘ *domain d2*))
      (*apply-chart d2* ∘ *f* ∘ *inv-chart d1*)
⟨*proof*⟩

lemma *diff-between-chartsE*:
  assumes *d1* ∈ *src.atlas* *d2* ∈ *dest.atlas*
  assumes *y* ∈ *domain d1* *y* ∈ *src.carrier* *f y* ∈ *domain d2*
  obtains *X* **where**
    k−*smooth-on X* (*apply-chart d2* ∘ *f* ∘ *inv-chart d1*)
    d1 y ∈ *X*
    open X
    X = *codomain d1* ∩ *inv-chart d1* −‘ (*src.carrier* ∩ *f* −‘ *domain d2*)
⟨*proof*⟩

end

lemma *diff-compose*:
  diff k M1 M3 (*g* ∘ *f*)
  if *diff k M1 M2 f* *diff k M2 M3 g*
⟨*proof*⟩

context *diff* **begin**

lemma *diff-submanifold*: *diff k* (*src.charts-submanifold S*) *charts2 f*
  if *open S*
⟨*proof*⟩

lemma *diff-submanifold2*: *diff k charts1* (*dest.charts-submanifold S*) *f*
  if *open S* *f* ‘ *src.carrier* ⊆ *S*
⟨*proof*⟩

end

context *c-manifolds* **begin**

lemma *diff-localI*: *diff k charts1 charts2 f*
  if ⋀*x*. *x* ∈ *src.carrier* ⟹ *diff k* (*src.charts-submanifold* (*U x*)) *charts2 f*
    ⋀*x*. *x* ∈ *src.carrier* ⟹ *open* (*U x*)
    ⋀*x*. *x* ∈ *src.carrier* ⟹ *x* ∈ (*U x*)
⟨*proof*⟩

lemma *diff-open-coverI*: *diff k charts1 charts2 f*
if *diff*: $\bigwedge u.\ u \in U \Longrightarrow$ *diff k* (*src.charts-submanifold u*) *charts2 f*
and *op*: $\bigwedge u.\ u \in U \Longrightarrow$ *open u*
and *cover*: *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*: $c \in$ *c-manifold.atlas charts l* **if** $l \leq k$ $c \in$ *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*} ⇒ ′*b*::*euclidean-space*

lemma *diff-fun-compose*:
diff-fun k M1 ($g \circ f$)
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* ∈ *manifold-eucl.atlas k*
⟨*proof*⟩

lemma *apply-chart-smooth-on*:
k−*smooth-on* (*domain c*) *c* **if** *c* ∈ *manifold-eucl.atlas k*
⟨*proof*⟩

lemma *inv-chart-smooth-on*: *k*−*smooth-on* (*codomain c*) (*inv-chart c*) **if** *c* ∈ *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* ∘ *f*)
assumes *continuous-on X f*
assumes *c* ∈ *manifold-eucl.atlas k open X f* ‘ *X* ⊆ *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* ∈ *manifold-eucl.atlas k open X X* ⊆ *domain c*
shows *k*−*smooth-on X f*

⟨*proof*⟩

context *diff-fun* **begin**

lemma *diff-fun-order-le*: *diff-fun l charts f* **if** $l \le k$
⟨*proof*⟩

end

6.5 Diffeormorphism

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

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 $(\bigwedge x.\ x{\in}carrier \Longrightarrow \exists\, c1{\in}atlas.\ x \in domain\ c1 \wedge (k{-}smooth\text{-}on\ (codomain\ c1)\ (f \circ inv\text{-}chart\ c1)))$
⟨*proof*⟩

end

lemma (**in** *diff*) *diff-cong*: *diff k charts1 charts2 g* **if** $\bigwedge x.\ x \in src.carrier \Longrightarrow f\ x = g\ x$
⟨*proof*⟩

context *diff-fun* **begin**

lemma *diff-fun-cong*: *diff-fun k charts g* **if** $\bigwedge x.\ x \in src.carrier \Longrightarrow f\ x = g\ x$
⟨*proof*⟩

lemma *diff-funD*:
$\exists\, c1{\in}src.atlas.\ x \in domain\ c1 \wedge (k{-}smooth\text{-}on\ (codomain\ c1)\ (f \circ inv\text{-}chart\ c1))$
if *x*: $x \in src.carrier$
⟨*proof*⟩

lemma *diff-funE*:
assumes $x \in src.carrier$
obtains *c1* **where**

c1∈*src.atlas x* ∈ *domain c1 k*−*smooth-on* (*codomain c1*) (*f* ∘ *inv-chart c1*)
⟨*proof*⟩

lemma *diff-fun-between-chartsD*:
assumes *c* ∈ *src.atlas x* ∈ *domain c*
shows *k*−*smooth-on* (*codomain c*) (*f* ∘ *inv-chart c*)
⟨*proof*⟩

lemma *diff-fun-submanifold*: *diff-fun k* (*src.charts-submanifold S*) *f*
if [*simp*]: *open S*
⟨*proof*⟩

end

context *c-manifold* **begin**

lemma *diff-fun-zero*: *diff-fun k charts 0*
⟨*proof*⟩

lemma *diff-fun-const*: *diff-fun k charts* ($\lambda x.$ *c*)
⟨*proof*⟩

lemma *diff-fun-add*: *diff-fun k charts* (*a* + *b*) **if** *diff-fun k charts a diff-fun k charts b*
⟨*proof*⟩

lemma *diff-fun-sum*: *diff-fun k charts* ($\lambda x.$ $\sum$ *i*∈*S*. *f i x*) **if** $\bigwedge i.$ *i* ∈ *S* ⟹ *diff-fun k charts* (*f i*)
⟨*proof*⟩

lemma *diff-fun-scaleR*: *diff-fun k charts* ($\lambda x.$ *a x* $*_R$ *b x*)
if *diff-fun k charts a diff-fun k charts b*
⟨*proof*⟩

lemma *diff-fun-scaleR-left*: *diff-fun k charts* (*c* $*_R$ *b*)
if *diff-fun k charts b*
⟨*proof*⟩

lemma *diff-fun-times*: *diff-fun k charts* (*a* ∗ *b*) **if** *diff-fun k charts a diff-fun k charts b*
for *a b*::- ⇒ -::*real-normed-algebra*
⟨*proof*⟩

lemma *diff-fun-divide*: *diff-fun k charts* ($\lambda x.$ *a x* / *b x*)
if *diff-fun k charts a diff-fun k charts b*
and *nz*: $\bigwedge x.$ *x* ∈ *carrier* ⟹ *b x* ≠ *0*
for *a b*::- ⇒ -::*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::$'i \Rightarrow ('a, 'b)$ *chart*) $\longleftrightarrow$
countable I $\wedge$
carrier = ($\bigcup i \in I$. *domain* (ψ *i*)) $\wedge$
locally-finite-on carrier I (*domain o* ψ) $\wedge$
($\forall i \in I$. *codomain* (ψ *i*) = *ball 0 3*) $\wedge$
carrier = ($\bigcup i \in I$. *inv-chart* (ψ *i*) ‘ *ball 0 1*)

Every covering has a refinement that is a regular cover.

lemma *reguler-refinementE*:
fixes $\mathcal{X}$::$'i \Rightarrow 'a$ *set*
assumes *cover*: *carrier* $\subseteq$ ($\bigcup i \in I$. $\mathcal{X}$ *i*) **and** *open-cover*: $\bigwedge i$. $i \in I \Longrightarrow$ *open* ($\mathcal{X}$ *i*)
obtains *N*::*nat set* **and** ψ::*nat* $\Rightarrow$ $('a, 'b)$ *chart*
where $\bigwedge i$. $i \in N \Longrightarrow \psi$ *i* $\in$ *atlas* (*domain o* ψ) ‘ *N refines* $\mathcal{X}$ ‘ *I regular-cover N* ψ
⟨*proof*⟩

lemma *diff-apply-chart*:

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

lemma *diff-inv-chart*:
diff k (*manifold-eucl.charts-submanifold* (*codomain c*)) *charts* (*inv-chart c*) **if** *c* $\in$ *atlas*
⟨*proof*⟩

lemma *chart-inj-on* [*simp*]:
fixes c :: (*'a*, *'b*) *chart*
assumes $x \in$ *domain c* $y \in$ *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 *carrier* $\subseteq$ ($\bigcup i \in A.\ X\ i$)
assumes $\bigwedge i.\ i \in A \Longrightarrow$ *open* ($X\ i$)
obtains φ::*'i* $\Rightarrow$ *'a* $\Rightarrow$ *real*
where $\bigwedge i.\ i \in A \Longrightarrow$ *diff-fun k charts* ($\varphi\ i$)
and $\bigwedge i\ x.\ i \in A \Longrightarrow x \in$ *carrier* $\Longrightarrow 0 \leq \varphi\ i\ x$
and $\bigwedge i\ x.\ i \in A \Longrightarrow x \in$ *carrier* $\Longrightarrow \varphi\ i\ x \leq 1$
and $\bigwedge x.\ x \in$ *carrier* $\Longrightarrow (\sum i \in \{i \in A.\ \varphi\ i\ x \neq 0\}.\ \varphi\ i\ x) = 1$
and $\bigwedge i.\ i \in A \Longrightarrow$ *csupport-on carrier* ($\varphi\ i$) $\cap$ *carrier* $\subseteq X\ i$
and *locally-finite-on carrier A* ($\lambda i.$ *csupport-on carrier* ($\varphi\ i$))
⟨*proof*⟩

Given $A \subseteq U \subseteq$ *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 $A \subseteq U\ U \subseteq$ *carrier open* U
obtains ψ::*'a* $\Rightarrow$ *real* **where**
diff-fun k charts ψ
$\bigwedge x.\ x \in$ *carrier* $\Longrightarrow 0 \leq \psi\ x$
$\bigwedge x.\ x \in$ *carrier* $\Longrightarrow \psi\ x \leq 1$
$\bigwedge x.\ x \in A \Longrightarrow \psi\ x = 1$
csupport-on carrier $\psi \cap$ *carrier* $\subseteq U$
⟨*proof*⟩

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

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

lemma *diff-fun-onE*:
assumes *diff-fun-on A f*
obtains $W\ f'$ **where**
$A \subseteq W\ W \subseteq \mathit{carrier}\ \mathit{open}\ W\ \mathit{diff\text{-}fun}\ k\ (\mathit{charts\text{-}submanifold}\ W)\ f'$
$\bigwedge x.\ x \in A \Longrightarrow f\ x = f'\ x$
⟨*proof*⟩

lemma *diff-fun-onI*:
assumes $A \subseteq W\ W \subseteq \mathit{carrier}\ \mathit{open}\ W\ \mathit{diff\text{-}fun}\ k\ (\mathit{charts\text{-}submanifold}\ W)\ f'$
$\bigwedge x.\ x \in A \Longrightarrow f\ x = f'\ x$
shows *diff-fun-on A f*
⟨*proof*⟩

Extension lemma:

Given $A \subseteq U \subseteq \mathit{carrier}$, 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 $f{::}'a \Rightarrow {}'e{::}\mathit{euclidean\text{-}space}$
assumes *closedin* (*top-of-set carrier*) *A*
assumes $\mathit{diff\text{-}fun\text{-}on}\ A\ f\ A \subseteq U\ U \subseteq \mathit{carrier}\ \mathit{open}\ U$
obtains f' **where**
$\mathit{diff\text{-}fun}\ k\ \mathit{charts}\ f'$
$\bigwedge x.\ x \in A \Longrightarrow f'\ x = f\ x$
$\mathit{csupport\text{-}on}\ \mathit{carrier}\ f' \cap \mathit{carrier} \subseteq U$
⟨*proof*⟩

end

end

8 Tangent Space

theory *Tangent-Space*
imports *Partition-Of-Unity*
begin

lemma *linear-imp-linear-on*: *linear-on A B scaleR scaleR f* **if** *linear f*
subspace A subspace B
⟨*proof*⟩

lemma (**in** *vector-space-pair-on*)
linear-sum':
$\forall\, x.\ x \in S1 \longrightarrow f\ x \in S2 \Longrightarrow$
$\forall\, x.\ x \in S \longrightarrow g\ x \in S1 \Longrightarrow$

linear-on S1 S2 scale1 scale2 f $\Longrightarrow$
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* ¬ *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* :: ($'a \Rightarrow$ *real*) *set* **where**
diff-fun-space = {*f*. *diff-fun k charts f* ∧ *extensional0 carrier f*}

lemma *diff-fun-spaceD*: *diff-fun k charts f* **if** *f* ∈ *diff-fun-space*
⟨*proof*⟩

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

lemma *diff-fun-space-extensionalD*:
g $\in$ *diff-fun-space* $\Longrightarrow$ *extensional0 carrier g*
⟨*proof*⟩

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

lemma *subspace-diff-fun-space*[*intro*, *simp*]:
subspace diff-fun-space
⟨*proof*⟩

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

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

Set of differentiable functions is a vector space

sublocale *diff-fun-space*: *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* ≡ *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* ⇒ *real*) ⇒ *real*) ⇒ *'a* ⇒ *bool* **where**
is-derivation X p ⟷ (*linear-diff-fun X* ∧
(∀*f g*. *f* ∈ *diff-fun-space* ⟶ *g* ∈ *diff-fun-space* ⟶ *X* (*f* ∗ *g*) = *f p* ∗ *X g* + *g p* ∗ *X f*))

lemma *is-derivationI*:
is-derivation X p
if *linear-diff-fun X*
⋀*f g*. *f* ∈ *diff-fun-space* ⟹ *g* ∈ *diff-fun-space* ⟹ *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*: $\bigwedge f\ g.\ f \in$ *diff-fun-space* $\Longrightarrow g \in$ *diff-fun-space* $\Longrightarrow X\ (f * g) = f\ p * X\ g + g\ p * X\ f$
  ⟨*proof*⟩

Differentiable functions on the Euclidean space

lemma *manifold-eucl-diff-fun-space-iff* [*simp*]:
  $g \in$ *manifold-eucl.diff-fun-space k* $\longleftrightarrow$ *k−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* :: $'a \Rightarrow (('a \Rightarrow real) \Rightarrow real)$ *set* **where**
  tangent-space p = $\{X.$ *is-derivation X p* $\wedge$ *extensional0 diff-fun-space X*$\}$

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

lemma *mem-tangent-space*: $X \in$ *tangent-space p* $\longleftrightarrow$ *is-derivation X p* $\wedge$ *extensional0 diff-fun-space X*
  ⟨*proof*⟩

lemma *tangent-spaceI*:
  $X \in$ *tangent-space p*
  if
    extensional0 diff-fun-space X
    linear-diff-fun X
    $\bigwedge f\ g.\ f \in$ *diff-fun-space* $\Longrightarrow g \in$ *diff-fun-space* $\Longrightarrow X\ (f * g) = f\ p * X\ g + g\ p * X\ f$
  ⟨*proof*⟩

lemma *tangent-spaceD*:
  assumes $X \in$ *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*: $\bigwedge f\ g.\ f \in$ *diff-fun-space* $\Longrightarrow g \in$ *diff-fun-space* $\Longrightarrow X\ (f * g) = f\ p * X\ g + g\ p * X\ f$
  ⟨*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* $(c *_R x)$ *p*
if *x*: *is-derivation x p*
⟨*proof*⟩

lemma *subspace-is-derivation*: *subspace* {*X*. *is-derivation X p*}
⟨*proof*⟩

lemma *subspace-tangent-space*: *subspace* (*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* = *dim X*
if $X \subseteq$ *tangent-space p*
⟨*proof*⟩

properties of derivations

lemma *restrict0-in-fun-space*: *restrict0 carrier f* $\in$ *diff-fun-space*
if *diff-fun k charts f*
⟨*proof*⟩

lemma *restrict0-const-diff-fun-space*: *restrict0 carrier* ($\lambda x.$ *c*) $\in$ *diff-fun-space*
⟨*proof*⟩

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

lemma *derivation-const-eq-zero*: *X* (*restrict0 carrier* ($\lambda x.$ *c*)) = *0*
if $X \in$ *tangent-space p* $p \in$ *carrier*
⟨*proof*⟩

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

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

⟨*proof*⟩

lemma *derivation-eq-localI*: $X\, f = X\, g$
  if *open* U $p \in U$ $U \subseteq$ *carrier*
    $X \in$ *tangent-space* p
    $f \in$ *diff-fun-space*
    $g \in$ *diff-fun-space*
    $\bigwedge x.\ x \in U \Longrightarrow f\, x = g\, x$
⟨*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* :: $(('a \Rightarrow real) \Rightarrow real) \Rightarrow ('b \Rightarrow real) \Rightarrow real$ **where**
  push-forward X = *restrict0 dest.diff-fun-space* $(\lambda g.\ X$ (*restrict0 src.carrier* $(g \circ f)))$

lemma *extensional-push-forward*: *extensional0 dest.diff-fun-space* (*push-forward* X)
  ⟨*proof*⟩

lemma *linear-push-forward*: *linear push-forward*
  ⟨*proof*⟩

Properties of push-forwards

lemma *restrict-compose-in-diff-fun-space*:
  $x \in$ *dest.diff-fun-space* $\Longrightarrow$ *restrict0 src.carrier* $(x \circ f) \in$ *src.diff-fun-space*
  ⟨*proof*⟩

Push-forward of a linear functional is a linear

lemma *linear-on-diff-fun-push-forward*:
  dest.linear-diff-fun (*push-forward* X)
  if *src.linear-diff-fun* X
⟨*proof*⟩

Push-forward preserves the product rule

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

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

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

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

end

Functoriality of push-forward: identity

context *c-manifold* **begin**

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

end

Functoriality of push-forward: composition

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

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

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

lemma *dim-tangent-space-src-dest-eq*: *dim* (*src.tangent-space p*) = *dim* (*dest.tangent-space* (*f p*))

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

lemma *dim-tangent-space-src-dest-eq2*: *dim* (*src.tangent-space* p) = *dim* (*dest.tangent-space* (f p))
if p: $p \in$ *src.carrier* **and** *dim* (*src.tangent-space* p) > 0
⟨*proof*⟩

end

8.5 Smooth inclusion map

context *submanifold* **begin**

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

sublocale *inclusion*: *diff* k *charts-submanifold* S *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::$'a \Rightarrow 'e$::*euclidean-space*
assumes f: *diff-fun* k (*charts-submanifold* S) f
and B: *closed* B $B \subseteq$ *sub.carrier*
obtains f' **where**
diff-fun k *charts* f'
($\bigwedge x.\ x \in B \Longrightarrow f'\ x = f\ x$)
csupport-on carrier $f' \cap$ *carrier* $\subseteq$ *sub.carrier*
⟨*proof*⟩

lemma *inj-on-push-forward-inclusion*: *inj-on inclusion.push-forward* (*sub.tangent-space* p)
if p: $p \in$ *sub.carrier*
⟨*proof*⟩

lemma *surj-on-push-forward-inclusion*:
inclusion.push-forward ‘ *sub.tangent-space* $p \supseteq$ *tangent-space* p
if p: $p \in$ *sub.carrier*
⟨*proof*⟩

end

8.6 Tangent space of submanifold

lemma *span-idem*: *span X* = *X* **if** *subspace X*
⟨*proof*⟩

context *submanifold* **begin**

lemma *dim-tangent-space*: *dim* (*tangent-space p*) = *dim* (*sub.tangent-space p*)
if *p* ∈ *sub.carrier dim* (*sub.tangent-space p*) > *0*
⟨*proof*⟩

lemma *dim-tangent-space2*: *dim* (*tangent-space p*) = *dim* (*sub.tangent-space p*)
if *p* ∈ *sub.carrier dim* (*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* ⇒ *'a* ⇒ *'a*::*euclidean-space* ⇒
(*'a* ⇒ *real*) ⇒ *real* **where**
directional-derivative k a v = *restrict0* (*manifold-eucl.diff-fun-space k*) (*λf. frechet-derivative f* (*at a*) *v*)

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

lemma *extensional0-directional-derivative-le*:
extensional0 (*manifold-eucl.diff-fun-space k*) (*directional-derivative k' a v*)
if *k* ≤ *k'*
⟨*proof*⟩

lemma *directional-derivative-add*[*simp*]: *directional-derivative k a* (*x* + *y*) = *directional-derivative k a x* + *directional-derivative k a y*
and *directional-derivative-scaleR*[*simp*]: *directional-derivative k a* (*c* $*_R$ *x*) = *c* $*_R$ *directional-derivative k a x*
if *k* ≠ *0*
⟨*proof*⟩

lemma *linear-directional-derivative*: *k* ≠ *0* ⟹ *linear* (*directional-derivative k a*)
⟨*proof*⟩

lemma *frechet-derivative-inner*[*simp*]:
frechet-derivative (*λx. x* · *j*) (*at a*) = (*λx. x* · *j*)
⟨*proof*⟩

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

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

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

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

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

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

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

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

context *c-manifold* **begin**

lemma *is-derivation-order-le*:
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 $C\hat{}k$ differentiable manifolds. 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*:
$k \neq 0 \Longrightarrow$ *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*:
$k \neq 0 \Longrightarrow$ *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*: $k = \infty$
⟨*proof*⟩

end

end

9 Cotangent Space

theory *Cotangent-Space*
  imports *Tangent-Space*
begin

9.1 Dual of a vector space

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

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

lemma *dual-space-eq*:
  $\textit{dual-space}\ S = \{E.\ \textit{linear-fun-on}\ S\ E\} \cap \{E.\ \textit{extensional0}\ S\ E\}$
  ⟨*proof*⟩

lemma *mem-dual-space*:
  $E \in \textit{dual-space}\ S \longleftrightarrow \textit{linear-fun-on}\ S\ E \wedge \textit{extensional0}\ S\ E$
  ⟨*proof*⟩

lemma *dual-spaceI*:
  $E \in \textit{dual-space}\ S$
  if $\textit{extensional0}\ S\ E\ \textit{linear-fun-on}\ S\ E$
  ⟨*proof*⟩

lemma *dual-spaceD*:
  assumes $E \in \textit{dual-space}\ S$
  shows *dual-space-linear-on*: $\textit{linear-fun-on}\ S\ E$
    and *dual-space-restrict*[*simp*]: $\textit{extensional0}\ S\ E$
  ⟨*proof*⟩

lemma *linear-fun-on-zero*:
  $\textit{linear-fun-on}\ S\ 0$
  if $\textit{subspace}\ S$
  ⟨*proof*⟩

lemma $\textit{linear-fun-on}\ S\ x \Longrightarrow a \in S \Longrightarrow b \in S \Longrightarrow x\ (a + b) = x\ a + x\ b$
  ⟨*proof*⟩

lemma *linear-fun-on-add*:
  $\textit{linear-fun-on}\ S\ (x + y)$
  if x: $\textit{linear-fun-on}\ S\ x$ **and** y: $\textit{linear-fun-on}\ S\ y$ **and** S: $\textit{subspace}\ S$
  ⟨*proof*⟩

lemma *linear-fun-on-scaleR*:
  $\textit{linear-fun-on}\ S\ (c *_R x)$
  if x: $\textit{linear-fun-on}\ S\ x$ **and** S: $\textit{subspace}\ S$
  ⟨*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*∈*B*. *representation B a i* ∗ *representation B b i*)
— TODO: move to library

definition *std-dual* :: *'a::real-vector* ⇒ (*'a* ⇒ *real*) **where**
std-dual a = *restrict0 S* (*restrict0 S* (λ*b*. *inner-Basis a b*))

lemma *inner-Basis-add*:
b1 ∈ *S* ⟹ *b2* ∈ *S* ⟹ *inner-Basis* (*b1* + *b2*) *v* = *inner-Basis b1 v* + *inner-Basis b2 v*
⟨*proof*⟩

lemma *inner-Basis-add2*:
b1 ∈ *S* ⟹ *b2* ∈ *S* ⟹ *inner-Basis v* (*b1* + *b2*) = *inner-Basis v b1* + *inner-Basis v b2*
⟨*proof*⟩

lemma *inner-Basis-scale*:
b1 ∈ *S* ⟹ *inner-Basis* ($c *_R b1$) *v* = *c* ∗ *inner-Basis b1 v*
⟨*proof*⟩

lemma *inner-Basis-scale2*:
b1 ∈ *S* ⟹ *inner-Basis v* ($c *_R b1$) = *c* ∗ *inner-Basis v b1*
⟨*proof*⟩

lemma *inner-Basis-minus*:
b1 ∈ *S* ⟹ *b2* ∈ *S* ⟹ *inner-Basis* (*b1* − *b2*) *v* = *inner-Basis b1 v* − *inner-Basis b2 v*
and *inner-Basis-minus2*:
b1 ∈ *S* ⟹ *b2* ∈ *S* ⟹ *inner-Basis v* (*b1* − *b2*) = *inner-Basis v b1* − *inner-Basis v b2*
⟨*proof*⟩

lemma *sum-zero-representation*:
  $v = 0$
  if $\bigwedge b.\ b \in B \Longrightarrow$ *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* $a\ a = 0 \longleftrightarrow a = 0$
  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*:
  $(\bigwedge i.\ i \in I \Longrightarrow x\ i \in S) \Longrightarrow$ *inner-Basis* $(\sum i{\in}I.\ x\ i)\ v = (\sum i{\in}I.$ *inner-Basis* $(x\ i)\ v)$
  ⟨*proof*⟩

lemma *inner-Basis-sum2*:
  $(\bigwedge i.\ i \in I \Longrightarrow x\ i \in S) \Longrightarrow$ *inner-Basis* $v\ (\sum i{\in}I.\ x\ i) = (\sum i{\in}I.$ *inner-Basis* $v\ (x\ i))$
  ⟨*proof*⟩

lemma *B-sub-S*: $B \subseteq S$
⟨*proof*⟩

lemma *inner-Basis-eq-representation*:
inner-Basis $i\ x$ = *representation* $B\ x\ i$
if $i \in B$ *finite* B
⟨*proof*⟩

lemma *surj-std-dual*:
std-dual ‘ $S \supseteq$ *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* $B \Longrightarrow$ *std-dual* ‘ S = *dual-space* S
⟨*proof*⟩

lemma *dim-dual-space*:
assumes *finite* B
shows *dim* (*dual-space* S) = *dim* S
⟨*proof*⟩

end

9.3 Dual map

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

definition *dual-map* :: $('a \Rightarrow 'b) \Rightarrow ('b \Rightarrow real) \Rightarrow ('a \Rightarrow real)$ **where**
dual-map $f\ y$ = *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 ‘ (*dual-space* T) $\subseteq$ *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*:
  $real\text{-}vector\text{-}space\text{-}pair\text{-}on.dual\text{-}map\ S\ f\ y = y$
  if f: $\bigwedge x.\ x \in S \Longrightarrow f\ x = x$ **and** y: $y \in dual\text{-}space\ S$
⟨*proof*⟩

end

abbreviation $dual\text{-}map \equiv real\text{-}vector\text{-}space\text{-}pair\text{-}on.dual\text{-}map$
lemmas $dual\text{-}map\text{-}def = real\text{-}vector\text{-}space\text{-}pair\text{-}on.dual\text{-}map\text{-}def$

Functoriality of dual map: composition

lemma *dual-map-compose*:
  $dual\text{-}map\ S\ f\ (dual\text{-}map\ T\ g\ x) = dual\text{-}map\ S\ (g \circ f)\ x$
  if $x \in dual\text{-}space\ U$ **and** $linear\text{-}on\ T\ U\ scaleR\ scaleR\ g$
  and f: $linear\text{-}on\ S\ T\ scaleR\ scaleR\ f$
  and *defined*: $f\ `\ S \subseteq T$
  and ST: $real\text{-}vector\text{-}space\text{-}pair\text{-}on\ S\ T$
  and TU: $real\text{-}vector\text{-}space\text{-}pair\text{-}on\ T\ U$
⟨*proof*⟩

9.4 Definition of cotangent space

context *c-manifold* **begin**

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

lemma *subspace-cotangent-space*:
  $subspace\ (cotangent\text{-}space\ p)$
  ⟨*proof*⟩

sublocale *cotangent-space*: $real\text{-}vector\text{-}space\text{-}on\ cotangent\text{-}space\ p$
  ⟨*proof*⟩

lemma *cotangent-space-dim-eq*: $cotangent\text{-}space.dim\ p\ X = dim\ X$
  if $X \subseteq cotangent\text{-}space\ p$
⟨*proof*⟩

lemma *dim-cotangent-space*:
  $dim\ (cotangent\text{-}space\ p) = DIM('b)$ **if** $p \in carrier$ **and** $k = \infty$
⟨*proof*⟩

end

9.5 Pullback of cotangent space

context *diff* **begin**

definition *pull-back* :: $'a \Rightarrow (((\prime b \Rightarrow real) \Rightarrow real) \Rightarrow real) \Rightarrow ((\prime a \Rightarrow real) \Rightarrow real)$ $\Rightarrow$ *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*)) ⊆ *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* :: $(\prime a \Rightarrow real) \Rightarrow \prime a \Rightarrow (((\prime a \Rightarrow real) \Rightarrow real) \Rightarrow real)$ **where**
cotangent-field f p = *restrict0* (*tangent-space p*) ($\lambda X.\ X\ f$)

lemma *cotangent-field-is-cotangent*:
cotangent-field f p ∈ *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* :: $(real \Rightarrow \prime a) \Rightarrow real \Rightarrow ((\prime a \Rightarrow real) \Rightarrow real)$ **where**
tangent-field c x = *restrict0 diff-fun-space* ($\lambda f.$ *frechet-derivative* ($f \circ c$) (*at x*) *1*)

lemma *tangent-field-is-tangent*:
tangent-field c x ∈ *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*:

$((\lambda x.\ (\textit{cotangent-field}\ f\ (c\ x))\ (\textit{tangent-field}\ c\ x))\ \textit{has-integral}\ f\ (c\ b) - f\ (c\ a))$ $\{a..b\}$
if *ab*: $a \leq b$ **and** *f*: $f \in$ *diff-fun-space* **and** *c*: *diff k charts-eucl charts c* **and** *k*: $k \neq 0$
⟨*proof*⟩

end

end

10 Product Manifold

theory *Product-Manifold*
imports *Differentiable-Manifold*
begin

locale *c-manifold-prod* =
m1: *c-manifold charts1 k* +
m2: *c-manifold charts2 k* **for** *k charts1 charts2*

begin

lift-definition *prod-chart* :: $('a, 'b)$ *chart* $\Rightarrow$ $('c, 'd)$ *chart* $\Rightarrow$ $('a \times 'c, 'b \times 'd)$ *chart*
is $\lambda(d::'a\ set,\ d'::'b\ set,\ f::'a \Rightarrow 'b,\ f'::'b \Rightarrow 'a).$
$\lambda(e::'c\ set,\ e'::'d\ set,\ g::'c \Rightarrow 'd,\ g'::'d \Rightarrow 'c).$
$(d \times e,\ d' \times e',\ \lambda(x,y).\ (f\ x,\ g\ y),\ \lambda(x,y).\ (f'\ x,\ g'\ y))$
⟨*proof*⟩

lemma *domain-prod-chart*[*simp*]: *domain* (*prod-chart c1 c2*) = *domain c1* × *domain c2*
and *codomain-prod-chart*[*simp*]: *codomain* (*prod-chart c1 c2*) = *codomain c1* × *codomain c2*
and *apply-prod-chart*[*simp*]: *apply-chart* (*prod-chart c1 c2*) = ($\lambda(x,y)$. (*c1 x*, *c2 y*))
and *inv-chart-prod-chart*[*simp*]: *inv-chart* (*prod-chart c1 c2*) = ($\lambda(x,y)$. (*inv-chart c1 x*, *inv-chart c2 y*))
⟨*proof*⟩

lemma *prod-chart-compat*:
k−smooth-compat (*prod-chart c1 c2*) (*prod-chart d1 d2*)
if *compat1*: *k−smooth-compat c1 d1* **and** *compat2*: *k−smooth-compat c2 d2*
⟨*proof*⟩

definition *prod-charts* :: $('a \times 'c, 'b \times 'd)$ *chart set* **where**
prod-charts = {*prod-chart c1 c2* | *c1 c2*. *c1* ∈ *charts1* ∧ *c2* ∈ *charts2*}

lemma *c-manifold-atlas-product*: *c-manifold prod-charts k*
⟨*proof*⟩

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**
$\lambda(x,z).\ x\ /_R\ (1 - z)$ ⟨*proof*⟩

lift-definition *st-proj1-inv* :: (*'a::real-normed-vector*) ⇒ *'a sphere* **is**
$\lambda x.\ ((2\ /\ ((norm\ x)\ \hat{}\ 2 + 1))\ *_R\ x,\ ((norm\ x)\ \hat{}\ 2 - 1)\ /\ ((norm\ x)\ \hat{}\ 2 + 1))$
⟨*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**
$\lambda(x,z).\ x\ /_R\ (1 + z)$ ⟨*proof*⟩

lift-definition *st-proj2-inv* :: (*'a::real-normed-vector*) ⇒ *'a sphere* **is**
$\lambda x.\ ((2\ /\ ((norm\ x)\ \hat{}\ 2 + 1))\ *_R\ x,\ (1 - (norm\ x)\ \hat{}\ 2)\ /\ ((norm\ x)\ \hat{}\ 2 + 1))$
⟨*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* (=) ===> ((=) ===> *pcr-sphere* (=)) ===> (=)) (λX::$'a$::*t2-space set*. *continuous-on* X) *continuous-on*
⟨*proof*⟩

lemma *transfer-continuous-on2*[*transfer-rule*]:
includes *lifting-syntax*
shows (*rel-set* (*pcr-sphere* (=)) ===> (*pcr-sphere* (=) ===> (=)) ===> (=)) (λX. *continuous-on* ($X \cap \{x.$ *norm* $x = 1\}$)) (λX. *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 (-> nonzero -> 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* ⇒ *bool* **is** *open*::*'a set* ⇒ *bool* ⟨*proof*⟩

instance
⟨*proof*⟩

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*
⟨*proof*⟩

instantiation *nonzero* :: (*euclidean-space*) *scaleR*
begin
lift-definition *scaleR-nonzero*::*real* ⇒ *'a nonzero* ⇒ *'a nonzero* **is** $\lambda c\ x$. *if* $c = 0$ *then One else* $c *_R x$
⟨*proof*⟩
instance ⟨*proof*⟩

end

instantiation *nonzero* :: (*euclidean-space*) *plus*
begin
lift-definition *plus-nonzero*::*'a nonzero* ⇒ *'a nonzero* ⇒ *'a nonzero* **is** $\lambda x\ y$. *if* $x + y = 0$ *then One else* $x + y$
⟨*proof*⟩
instance ⟨*proof*⟩
end

instantiation *nonzero* :: (*euclidean-space*) *minus*
begin
lift-definition *minus-nonzero*::*'a nonzero* ⇒ *'a nonzero* ⇒ *'a nonzero* **is** $\lambda x\ y$. *if* $x = y$ *then One else* $x - y$
⟨*proof*⟩
instance ⟨*proof*⟩
end

instantiation *nonzero* :: (*euclidean-space*) *dist*
begin
lift-definition *dist-nonzero*::*'a nonzero* ⇒ *'a nonzero* ⇒ *real* **is** *dist* ⟨*proof*⟩
instance ⟨*proof*⟩
end

instantiation *nonzero* :: (*euclidean-space*) *norm*
begin
lift-definition *norm-nonzero*::*'a nonzero* ⇒ *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*]: $b \neq 0 \Longrightarrow$ *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* ⇒ *'a nonzero* ⇒ *bool* **for** *x* **where**
$c \neq 0 \Longrightarrow$ *proj-rel x* $(c *_R x)$

lemma *proj-rel-parametric*: (*pcr-nonzero A* ===> *pcr-nonzero A* ===> (=)) *proj-rel proj-rel*
if [*transfer-rule*]: ((=) ===> *pcr-nonzero A* ===> *pcr-nonzero A*) $(*_R)$ $(*_R)$
bi-unique A
⟨*proof*⟩

quotient-type (**overloaded**) *'a proj-space* = (*'a*::*euclidean-space* × *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* ⇒ *bool* **where**
open-proj-space = *openin* (*map-topology Proj euclidean*)

lemma *topspace-map-Proj*: *topspace* (*map-topology Proj euclidean*) = *UNIV*
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *open-vimage-ProjI*: *open T* $\Longrightarrow$ *open* (*Proj* −‘ *T*)
⟨*proof*⟩
lemma *open-vimage-ProjD*: *open* (*Proj* −‘ *T*) $\Longrightarrow$ *open T*
⟨*proof*⟩
lemma *open-vimage-Proj-iff* [*simp*]: *open* (*Proj* −‘ *T*) = *open T*
⟨*proof*⟩

lemma *euclidean-proj-space-def*: *euclidean* = *map-topology Proj euclidean*
⟨*proof*⟩

lemma *continuous-on-proj-spaceI*: *continuous-on* (*S*) *f* **if** *continuous-on* (*Proj* −‘ *S*) (*f o Proj*) *open* (*S*)
for *f*::- *proj-space* ⇒ -
⟨*proof*⟩

lemma *saturate-eq*: *Proj* −‘ *Proj* ‘ *X* = ($\bigcup$ *c*∈*UNIV*−{*0*}. ($*_R$) *c* ‘ *X*)
⟨*proof*⟩

lemma *open-scaling-nonzero*: *c* ≠ *0* $\Longrightarrow$ *open s* $\Longrightarrow$ *open* (($*_R$) *c* ‘ *s*::′*a*::*euclidean-space nonzero set*)
⟨*proof*⟩

12.3 Proof of Hausdorff property

lemma *Proj-open-map*: *open* (*Proj* ‘ *X*) **if** *open X*
⟨*proof*⟩

lemma *proj-rel-transfer* [*transfer-rule*]:
(*pcr-nonzero A* ===> *pcr-nonzero A* ===> (=)) (λ*x a*. ∃ *c*. *a* = *sr c x* ∧ *c* ≠ *0*) *proj-rel*
if [*transfer-rule*]: ((=) ===> *pcr-nonzero A* ===> *pcr-nonzero A*) *sr* ($*_R$)
bi-unique A
⟨*proof*⟩

lemma *bool-aux*: *a* ∧ (*a* ⟶ *b*) ⟷ *a* ∧ *b* ⟨*proof*⟩

lemma *closed-proj-rel*: *closed* {(*x*::′*a*::*euclidean-space nonzero*, *y*::′*a nonzero*). *proj-rel x y*}
⟨*proof*⟩

lemma *closed-Proj-rel*: *closed* {(*x*, *y*). *Proj x* = *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*]: $c \neq 0 \Longrightarrow$ *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**
$\lambda x.$ *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*: *proj-rel a b* $\longleftrightarrow$ ($\exists$ *c*$\neq$*0*. *b* = *c* $*_R$ *a*)
⟨*proof*⟩

lemma *chart-last-inverse*: *chart-last-inv* (*chart-last x*) = *x* **if** *chart-last-domainP x*
⟨*proof*⟩

lemma *chart-last-inv-inverse*: *chart-last* (*chart-last-inv x*) = *x*
⟨*proof*⟩

lemma *chart-last-domainP-chart-last-inv*: *chart-last-domainP* (*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 *DIM*(*'a*) coordinates

lift-definition *chart-basis-nonzero* :: *'a* ⇒ (*'a*::*euclidean-space*×*real*)*nonzero* ⇒ *'a* **is**
λb. $\lambda(x,c)$. (x + (c − x · b) $*_R$ b) $/_R$ (x · b) ⟨*proof*⟩

lift-definition *chart-basis* :: *'a* ⇒ *'a*::*euclidean-space proj-space* ⇒ *'a* **is**
chart-basis-nonzero
⟨*proof*⟩

lift-definition *chart-basis-domain-nonzeroP* :: *'a* ⇒ (*'a*::*euclidean-space*×*real*) *nonzero* ⇒ *bool* **is**
λb (x, -). (x · b) ≠ *0* ⟨*proof*⟩

lift-definition *chart-basis-domainP* :: *'a* ⇒ *'a*::*euclidean-space proj-space* ⇒ *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
$\lambda x.\ (x + (1 - x \cdot b) *_R b,\ x \cdot b)$
⟨*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 $\Longrightarrow$ *chart-basis-domainP b xa* $\Longrightarrow$ *chart-last xa* $\cdot$ *b* $\neq$ *0*
⟨*proof*⟩

lemma *chart-basis-last-defined*:
b $\in$ *Basis* $\Longrightarrow$ *chart-last-domainP xa* $\Longrightarrow$ *chart-basis-domainP b xa* $\Longrightarrow$ *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* $\{x.\ (x + (1 - x \cdot a) *_R a) \cdot b \neq 0\}$ (*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* $\Longrightarrow$ *chart-basis-domainP a xa* $\Longrightarrow$ *chart-basis-domainP b xa* $\Longrightarrow$ *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.