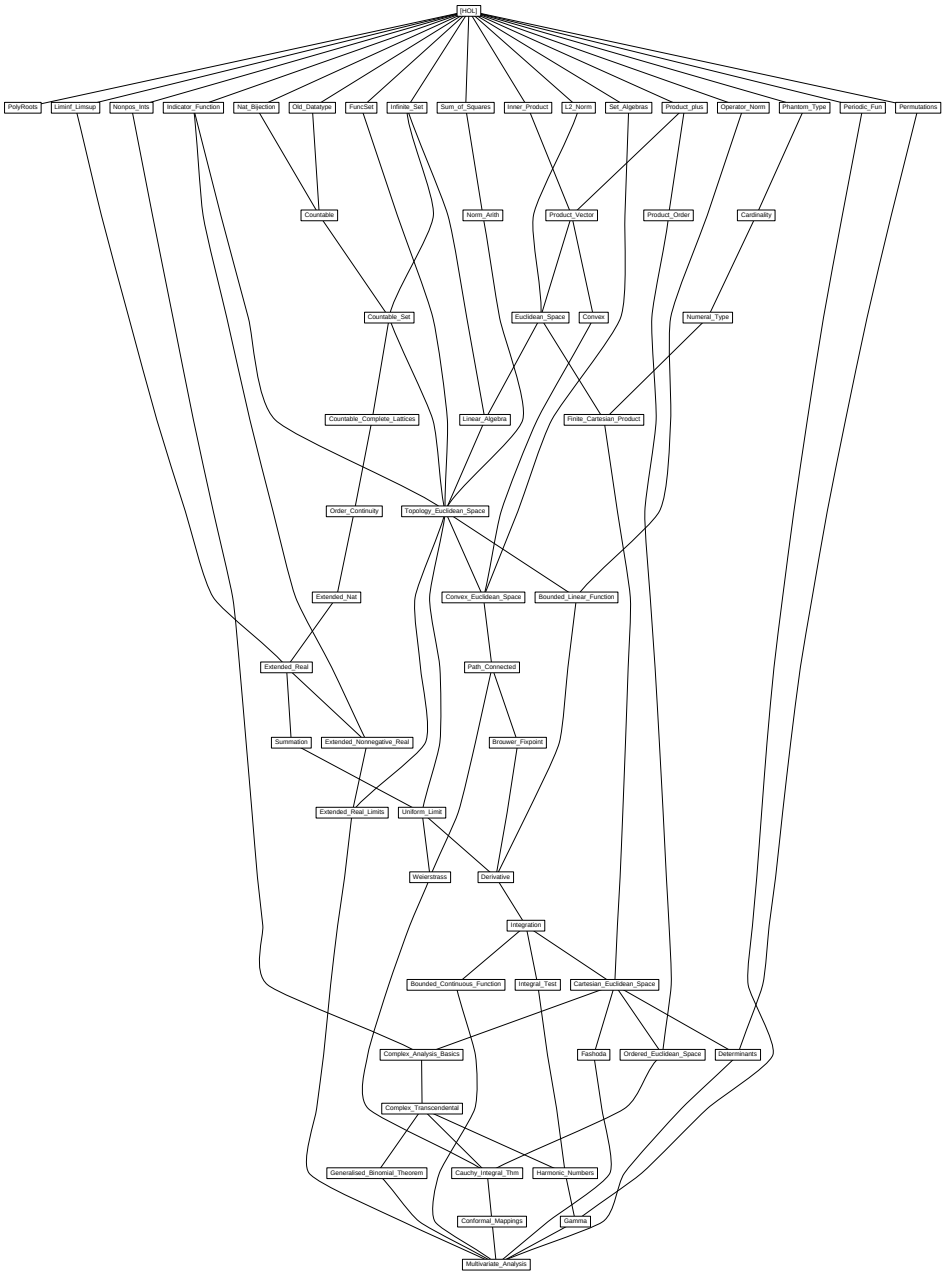
Multivariate Analysis

April 17, 2016

Contents


HOL1
PolyRoots
Limit_Limsup
Nonpos_Ints
Indicator_Function
Nat_Bijection
Old_Datatype
FuncSet
Infinite_Set
Sum_of_Squares
Inner_Product
L2_Norm
Set_Algebras
Product_plus
Operator_Norm
Phantom_Type
Periodic_Fun
Permutations
Countable
Norm_Arith
Product_Vector
Product_Order
Cardinality
Countable_Set
Euclidean_Space
Convex
Numeral_Type
Countable_Complete_Lattices
Linear_Algebra
Finite_Cartesian_Product
Order_Continuity
Topology_Euclidean_Space
Extended_Nat
Convex_Euclidean_Space
Bounded_Linear_Function
Extended_Real
Path_Connected
Summation
Extended_Nonnegative_Real
Brouwer_Fixpoint
Extended_Real_Limits
Uniform_Limit
Weierstrass
Derivative
Integration
Bounded_Continuous_Function
Integral_Test
Cartesian_Euclidean_Space
Complex_Analysis_Basics
Fashoda
Ordered_Euclidean_Space
Determinants
Complex_Transcendental
Generalised_Binomial_Theorem
Cauchy_Integral_Thm
Harmonic_Numbers
Conformal_Mappings
Gamma
Multivariate_Analysis

1 Indicator Function

theory *Indicator-Function*
imports *Complex-Main*
begin

definition *indicator S x* = (*if x* ∈ *S then 1 else 0*)

lemma *indicator-simps*[*simp*]:
x ∈ *S* ⟹ *indicator S x* = *1*
x ∉ *S* ⟹ *indicator S x* = *0*
⟨*proof*⟩

lemma *indicator-pos-le*[*intro*, *simp*]: (*0*::*'a*::*linordered-semidom*) ≤ *indicator S x*
and *indicator-le-1*[*intro*, *simp*]: *indicator S x* ≤ (*1*::*'a*::*linordered-semidom*)
⟨*proof*⟩

lemma *indicator-abs-le-1*: |*indicator S x*| ≤ (*1*::*'a*::*linordered-idom*)
⟨*proof*⟩

lemma *indicator-eq-0-iff*: *indicator A x* = (*0*::-::*zero-neq-one*) ⟷ *x* ∉ *A*
⟨*proof*⟩

lemma *indicator-eq-1-iff*: *indicator A x* = (*1*::-::*zero-neq-one*) ⟷ *x* ∈ *A*
⟨*proof*⟩

lemma *split-indicator*: *P* (*indicator S x*) ⟷ ((*x* ∈ *S* ⟶ *P 1*) ∧ (*x* ∉ *S* ⟶ *P 0*))
⟨*proof*⟩

lemma *split-indicator-asm*: *P* (*indicator S x*) ⟷ (¬ (*x* ∈ *S* ∧ ¬ *P 1* ∨ *x* ∉ *S* ∧ ¬ *P 0*))
⟨*proof*⟩

lemma *indicator-inter-arith*: *indicator* (*A* ∩ *B*) *x* = *indicator A x* ∗ (*indicator B x*::*'a*::*semiring-1*)
⟨*proof*⟩

lemma *indicator-union-arith*: *indicator* (*A* ∪ *B*) *x* = *indicator A x* + *indicator B x* − *indicator A x* ∗ (*indicator B x*::*'a*::*ring-1*)
⟨*proof*⟩

lemma *indicator-inter-min*: *indicator* (*A* ∩ *B*) *x* = *min* (*indicator A x*) (*indicator B x*::*'a*::*linordered-semidom*)
and *indicator-union-max*: *indicator* (*A* ∪ *B*) *x* = *max* (*indicator A x*) (*indicator B x*::*'a*::*linordered-semidom*)
⟨*proof*⟩

lemma *indicator-disj-union*: *A* ∩ *B* = {} ⟹ *indicator* (*A* ∪ *B*) *x* = (*indicator*

$A\ x + indicator\ B\ x::'a::linordered\text{-}semidom)$
⟨*proof*⟩

lemma *indicator-compl*: $indicator\ (-\ A)\ x = 1 - (indicator\ A\ x::'a::ring\text{-}1)$
and *indicator-diff*: $indicator\ (A - B)\ x = indicator\ A\ x * (1 - indicator\ B\ x::'a::ring\text{-}1)$
⟨*proof*⟩

lemma *indicator-times*: $indicator\ (A \times B)\ x = indicator\ A\ (fst\ x) * (indicator\ B\ (snd\ x)::'a::semiring\text{-}1)$
⟨*proof*⟩

lemma *indicator-sum*: $indicator\ (A <+> B)\ x = (case\ x\ of\ Inl\ x \Rightarrow indicator\ A\ x \mid Inr\ x \Rightarrow indicator\ B\ x)$
⟨*proof*⟩

lemma *indicator-image*: $inj\ f \Longrightarrow indicator\ (f\ `\ X)\ (f\ x) = (indicator\ X\ x::-::zero\text{-}neq\text{-}one)$
⟨*proof*⟩

lemma *indicator-vimage*: $indicator\ (f\ -`\ A)\ x = indicator\ A\ (f\ x)$
⟨*proof*⟩

lemma
fixes $f :: 'a \Rightarrow 'b::semiring\text{-}1$ **assumes** *finite A*
shows *setsum-mult-indicator*[*simp*]: $(\sum x \in A.\ f\ x * indicator\ B\ x) = (\sum x \in A \cap B.\ f\ x)$
and *setsum-indicator-mult*[*simp*]: $(\sum x \in A.\ indicator\ B\ x * f\ x) = (\sum x \in A \cap B.\ f\ x)$
⟨*proof*⟩

lemma *setsum-indicator-eq-card*:
assumes *finite A*
shows $(\sum x \in A.\ indicator\ B\ x) = card\ (A\ Int\ B)$
⟨*proof*⟩

lemma *setsum-indicator-scaleR*[*simp*]:
finite A $\Longrightarrow$
$(\sum x \in A.\ indicator\ (B\ x)\ (g\ x) *_R f\ x) = (\sum x \in \{x \in A.\ g\ x \in B\ x\}.\ f\ x::'a::real\text{-}vector)$
⟨*proof*⟩

lemma *LIMSEQ-indicator-incseq*:
assumes *incseq A*
shows $(\lambda i.\ indicator\ (A\ i)\ x :: 'a :: \{topological\text{-}space, one, zero\}) \longrightarrow indicator\ (\bigcup i.\ A\ i)\ x$
⟨*proof*⟩

lemma *LIMSEQ-indicator-UN*:
$(\lambda k.\ indicator\ (\bigcup i<k.\ A\ i)\ x :: 'a :: \{topological\text{-}space, one, zero\}) \longrightarrow$

indicator $(\bigcup i.\ A\ i)\ x$
⟨*proof*⟩

lemma *LIMSEQ-indicator-decseq*:
assumes *decseq A*
shows $(\lambda i.\ indicator\ (A\ i)\ x :: 'a :: \{topological\text{-}space, one, zero\}) \longrightarrow indicator\ (\bigcap i.\ A\ i)\ x$
⟨*proof*⟩

lemma *LIMSEQ-indicator-INT*:
$(\lambda k.\ indicator\ (\bigcap i{<}k.\ A\ i)\ x :: 'a :: \{topological\text{-}space, one, zero\}) \longrightarrow indicator\ (\bigcap i.\ A\ i)\ x$
⟨*proof*⟩

lemma *indicator-add*:
$A \cap B = \{\} \Longrightarrow (indicator\ A\ x::\text{-}::monoid\text{-}add) + indicator\ B\ x = indicator\ (A \cup B)\ x$
⟨*proof*⟩

lemma *of-real-indicator*: $of\text{-}real\ (indicator\ A\ x) = indicator\ A\ x$
⟨*proof*⟩

lemma *real-of-nat-indicator*: $real\ (indicator\ A\ x :: nat) = indicator\ A\ x$
⟨*proof*⟩

lemma *abs-indicator*: $|indicator\ A\ x :: 'a::linordered\text{-}idom| = indicator\ A\ x$
⟨*proof*⟩

lemma *mult-indicator-subset*:
$A \subseteq B \Longrightarrow indicator\ A\ x * indicator\ B\ x = (indicator\ A\ x :: 'a::\{comm\text{-}semiring\text{-}1\})$
⟨*proof*⟩

lemma *indicator-sums*:
assumes $\bigwedge i\ j.\ i \neq j \Longrightarrow A\ i \cap A\ j = \{\}$
shows $(\lambda i.\ indicator\ (A\ i)\ x::real)\ sums\ indicator\ (\bigcup i.\ A\ i)\ x$
⟨*proof*⟩

end

2 Old Datatype package: constructing datatypes from Cartesian Products and Disjoint Sums

theory *Old-Datatype*
imports *../Main*
keywords *old-datatype* :: *thy-decl*
begin

⟨*ML*⟩

2.1 The datatype universe

definition *Node* = {*p*. *EX f x k*. *p* = (*f* :: *nat* => *'b* + *nat*, *x* ::*'a* + *nat*) & *f k* = *Inr 0*}

typedef (*'a*, *'b*) *node* = *Node* :: ((*nat* => *'b* + *nat*) * (*'a* + *nat*)) *set*
morphisms *Rep-Node Abs-Node*
⟨*proof*⟩

Datatypes will be represented by sets of type *node*

type-synonym *'a item* = (*'a*, *unit*) *node set*
type-synonym (*'a*, *'b*) *dtree* = (*'a*, *'b*) *node set*

definition *Push* :: [(*'b* + *nat*), *nat* => (*'b* + *nat*)] => (*nat* => (*'b* + *nat*))

where *Push* == (%*b h*. *case-nat b h*)

definition *Push-Node* :: [(*'b* + *nat*), (*'a*, *'b*) *node*] => (*'a*, *'b*) *node*
where *Push-Node* == (%*n x*. *Abs-Node* (*apfst* (*Push n*) (*Rep-Node x*)))

definition *Atom* :: (*'a* + *nat*) => (*'a*, *'b*) *dtree*
where *Atom* == (%*x*. {*Abs-Node*((%*k*. *Inr 0*, *x*))})
definition *Scons* :: [(*'a*, *'b*) *dtree*, (*'a*, *'b*) *dtree*] => (*'a*, *'b*) *dtree*
where *Scons M N* == (*Push-Node* (*Inr 1*) ‘ *M*) *Un* (*Push-Node* (*Inr* (*Suc 1*)) ‘ *N*)

definition *Leaf* :: *'a* => (*'a*, *'b*) *dtree*
where *Leaf* == *Atom o Inl*
definition *Numb* :: *nat* => (*'a*, *'b*) *dtree*
where *Numb* == *Atom o Inr*

definition *In0* :: (*'a*, *'b*) *dtree* => (*'a*, *'b*) *dtree*
where *In0*(*M*) == *Scons* (*Numb 0*) *M*
definition *In1* :: (*'a*, *'b*) *dtree* => (*'a*, *'b*) *dtree*
where *In1*(*M*) == *Scons* (*Numb 1*) *M*

definition *Lim* :: (*'b* => (*'a*, *'b*) *dtree*) => (*'a*, *'b*) *dtree*
where *Lim f* == ⋃{*z*. *? x*. *z* = *Push-Node* (*Inl x*) ‘ (*f x*)}

definition *ndepth* :: (*'a*, *'b*) *node* => *nat*
where *ndepth*(*n*) == (%(*f*,*x*). *LEAST k*. *f k* = *Inr 0*) (*Rep-Node n*)
definition *ntrunc* :: [*nat*, (*'a*, *'b*) *dtree*] => (*'a*, *'b*) *dtree*

where *ntrunc k N == {n. n:N & ndepth(n)<k}*

definition *uprod* :: *[('a, 'b) dtree set, ('a, 'b) dtree set]=> ('a, 'b) dtree set*
where *uprod A B == UN x:A. UN y:B. { Scons x y }*
definition *usum* :: *[('a, 'b) dtree set, ('a, 'b) dtree set]=> ('a, 'b) dtree set*
where *usum A B == In0`A Un In1`B*

definition *Split* :: *[[('a, 'b) dtree, ('a, 'b) dtree]=>'c, ('a, 'b) dtree] => 'c*
where *Split c M == THE u. EX x y. M = Scons x y & u = c x y*

definition *Case* :: *[[('a, 'b) dtree]=>'c, [('a, 'b) dtree]=>'c, ('a, 'b) dtree] => 'c*
where *Case c d M == THE u. (EX x . M = In0(x) & u = c(x)) | (EX y . M = In1(y) & u = d(y))*

definition *dprod* :: *[(('a, 'b) dtree * ('a, 'b) dtree)set, (('a, 'b) dtree * ('a, 'b) dtree)set]*
*=> (('a, 'b) dtree * ('a, 'b) dtree)set*
where *dprod r s == UN (x,x'):r. UN (y,y'):s. {(Scons x y, Scons x' y')}*

definition *dsum* :: *[(('a, 'b) dtree * ('a, 'b) dtree)set, (('a, 'b) dtree * ('a, 'b) dtree)set]*
*=> (('a, 'b) dtree * ('a, 'b) dtree)set*
where *dsum r s == (UN (x,x'):r. {(In0(x),In0(x'))}) Un (UN (y,y'):s. {(In1(y),In1(y'))})*

lemma *apfst-convE*:
[| q = apfst f p; !!x y. [| p = (x,y); q = (f(x),y) |] ==> R |] ==> R
⟨*proof*⟩

lemma *Push-inject1*: *Push i f = Push j g ==> i=j*
⟨*proof*⟩

lemma *Push-inject2*: *Push i f = Push j g ==> f=g*
⟨*proof*⟩

lemma *Push-inject*:
[| Push i f =Push j g; [| i=j; f=g |] ==> P |] ==> P
⟨*proof*⟩

lemma *Push-neq-K0*: *Push (Inr (Suc k)) f = (%z. Inr 0) ==> P*
⟨*proof*⟩

lemmas *Abs-Node-inj* = *Abs-Node-inject* [*THEN* [*2*] *rev-iffD1*]

lemma *Node-K0-I*: (%*k*. *Inr 0*, *a*) : *Node*
⟨*proof*⟩

lemma *Node-Push-I*: *p*: *Node* ==> *apfst* (*Push i*) *p* : *Node*
⟨*proof*⟩

2.2 Freeness: Distinctness of Constructors

lemma *Scons-not-Atom* [*iff*]: *Scons M N* ≠ *Atom*(*a*)
⟨*proof*⟩

lemmas *Atom-not-Scons* [*iff*] = *Scons-not-Atom* [*THEN not-sym*]

lemma *inj-Atom*: *inj*(*Atom*)
⟨*proof*⟩
lemmas *Atom-inject* = *inj-Atom* [*THEN injD*]

lemma *Atom-Atom-eq* [*iff*]: (*Atom*(*a*)=*Atom*(*b*)) = (*a*=*b*)
⟨*proof*⟩

lemma *inj-Leaf*: *inj*(*Leaf*)
⟨*proof*⟩

lemmas *Leaf-inject* [*dest*!] = *inj-Leaf* [*THEN injD*]

lemma *inj-Numb*: *inj*(*Numb*)
⟨*proof*⟩

lemmas *Numb-inject* [*dest*!] = *inj-Numb* [*THEN injD*]

lemma *Push-Node-inject*:
[| *Push-Node i m* =*Push-Node j n*; [| *i*=*j*; *m*=*n* |] ==> *P*
|] ==> *P*
⟨*proof*⟩

lemma *Scons-inject-lemma1*: *Scons M N <= Scons M' N' ==> M<=M'*
⟨*proof*⟩

lemma *Scons-inject-lemma2*: *Scons M N <= Scons M' N' ==> N<=N'*
⟨*proof*⟩

lemma *Scons-inject1*: *Scons M N = Scons M' N' ==> M=M'*
⟨*proof*⟩

lemma *Scons-inject2*: *Scons M N = Scons M' N' ==> N=N'*
⟨*proof*⟩

lemma *Scons-inject*:
[| *Scons M N = Scons M' N'*; [| *M=M'*; *N=N'* |] ==> *P* |] ==> *P*
⟨*proof*⟩

lemma *Scons-Scons-eq* [*iff*]: (*Scons M N = Scons M' N'*) = (*M=M'* & *N=N'*)
⟨*proof*⟩

lemma *Scons-not-Leaf* [*iff*]: *Scons M N* ≠ *Leaf*(*a*)
⟨*proof*⟩

lemmas *Leaf-not-Scons* [*iff*] = *Scons-not-Leaf* [*THEN not-sym*]

lemma *Scons-not-Numb* [*iff*]: *Scons M N* ≠ *Numb*(*k*)
⟨*proof*⟩

lemmas *Numb-not-Scons* [*iff*] = *Scons-not-Numb* [*THEN not-sym*]

lemma *Leaf-not-Numb* [*iff*]: *Leaf*(*a*) ≠ *Numb*(*k*)
⟨*proof*⟩

lemmas *Numb-not-Leaf* [*iff*] = *Leaf-not-Numb* [*THEN not-sym*]

lemma *ndepth-K0*: *ndepth* (*Abs-Node*(%*k*. *Inr* 0, *x*)) = 0
⟨*proof*⟩

lemma *ndepth-Push-Node-aux*:
case-nat (*Inr* (*Suc i*)) *f k* = *Inr* 0 −−> *Suc*(*LEAST x*. *f x* = *Inr* 0) <= *k*
⟨*proof*⟩

lemma *ndepth-Push-Node*:
ndepth (*Push-Node* (*Inr* (*Suc i*)) *n*) = *Suc*(*ndepth*(*n*))
⟨*proof*⟩

lemma *ntrunc-0* [*simp*]: *ntrunc* 0 *M* = {}
⟨*proof*⟩

lemma *ntrunc-Atom* [*simp*]: *ntrunc* (*Suc k*) (*Atom a*) = *Atom*(*a*)
⟨*proof*⟩

lemma *ntrunc-Leaf* [*simp*]: *ntrunc* (*Suc k*) (*Leaf a*) = *Leaf*(*a*)
⟨*proof*⟩

lemma *ntrunc-Numb* [*simp*]: *ntrunc* (*Suc k*) (*Numb i*) = *Numb*(*i*)
⟨*proof*⟩

lemma *ntrunc-Scons* [*simp*]:
ntrunc (*Suc k*) (*Scons M N*) = *Scons* (*ntrunc k M*) (*ntrunc k N*)
⟨*proof*⟩

lemma *ntrunc-one-In0* [*simp*]: *ntrunc* (*Suc 0*) (*In0 M*) = {}
⟨*proof*⟩

lemma *ntrunc-In0* [*simp*]: *ntrunc* (*Suc*(*Suc k*)) (*In0 M*) = *In0* (*ntrunc* (*Suc k*) *M*)
⟨*proof*⟩

lemma *ntrunc-one-In1* [*simp*]: *ntrunc* (*Suc 0*) (*In1 M*) = {}
⟨*proof*⟩

lemma *ntrunc-In1* [*simp*]: *ntrunc* (*Suc*(*Suc k*)) (*In1 M*) = *In1* (*ntrunc* (*Suc k*) *M*)
⟨*proof*⟩

2.3 Set Constructions

lemma *uprodI* [*intro*!]: [| M:A; N:B |] ==> *Scons* M N : *uprod* A B
⟨*proof*⟩

lemma *uprodE* [*elim*!]:
[| c : *uprod* A B;
!!x y. [| x:A; y:B; c = *Scons* x y |] ==> P
|] ==> P
⟨*proof*⟩

lemma *uprodE2*: [| *Scons* M N : *uprod* A B; [| M:A; N:B |] ==> P |] ==> P
⟨*proof*⟩

lemma *usum-In0I* [*intro*]: M:A ==> *In0*(M) : *usum* A B
⟨*proof*⟩

lemma *usum-In1I* [*intro*]: N:B ==> *In1*(N) : *usum* A B
⟨*proof*⟩

lemma *usumE* [*elim*!]:
[| u : *usum* A B;
!!x. [| x:A; u=*In0*(x) |] ==> P;
!!y. [| y:B; u=*In1*(y) |] ==> P
|] ==> P
⟨*proof*⟩

lemma *In0-not-In1* [*iff*]: *In0*(M) ≠ *In1*(N)
⟨*proof*⟩

lemmas *In1-not-In0* [*iff*] = *In0-not-In1* [*THEN not-sym*]

lemma *In0-inject*: *In0*(M) = *In0*(N) ==> M=N
⟨*proof*⟩

lemma *In1-inject*: *In1*(M) = *In1*(N) ==> M=N
⟨*proof*⟩

lemma *In0-eq* [*iff*]: (*In0* M = *In0* N) = (M=N)
⟨*proof*⟩

lemma *In1-eq* [*iff*]: $(In1\ M = In1\ N) = (M{=}N)$
⟨*proof*⟩

lemma *inj-In0*: *inj In0*
⟨*proof*⟩

lemma *inj-In1*: *inj In1*
⟨*proof*⟩

lemma *Lim-inject*: $Lim\ f = Lim\ g \Longrightarrow f = g$
⟨*proof*⟩

lemma *ntrunc-subsetI*: $ntrunc\ k\ M <= M$
⟨*proof*⟩

lemma *ntrunc-subsetD*: $(!!k.\ ntrunc\ k\ M <= N) \Longrightarrow M{<=}N$
⟨*proof*⟩

lemma *ntrunc-equality*: $(!!k.\ ntrunc\ k\ M = ntrunc\ k\ N) \Longrightarrow M{=}N$
⟨*proof*⟩

lemma *ntrunc-o-equality*:
$[|\ !!k.\ (ntrunc(k)\ o\ h1) = (ntrunc(k)\ o\ h2)\ |] \Longrightarrow h1{=}h2$
⟨*proof*⟩

lemma *uprod-mono*: $[|\ A{<=}A';\ \ B{<=}B'\ |] \Longrightarrow uprod\ A\ B <= uprod\ A'\ B'$
⟨*proof*⟩

lemma *usum-mono*: $[|\ A{<=}A';\ \ B{<=}B'\ |] \Longrightarrow usum\ A\ B <= usum\ A'\ B'$
⟨*proof*⟩

lemma *Scons-mono*: $[|\ M{<=}M';\ \ N{<=}N'\ |] \Longrightarrow Scons\ M\ N <= Scons\ M'\ N'$
⟨*proof*⟩

lemma *In0-mono*: $M{<=}N \Longrightarrow In0(M) <= In0(N)$
⟨*proof*⟩

lemma *In1-mono*: $M{<=}N \Longrightarrow In1(M) <= In1(N)$
⟨*proof*⟩

lemma *Split* [*simp*]: $Split\ c\ (Scons\ M\ N) = c\ M\ N$
⟨*proof*⟩

lemma *Case-In0* [*simp*]: $Case\ c\ d\ (In0\ M) = c(M)$
⟨*proof*⟩

lemma *Case-In1* [*simp*]: $Case\ c\ d\ (In1\ N) = d(N)$
⟨*proof*⟩

lemma *ntrunc-UN1*: $ntrunc\ k\ (UN\ x.\ f(x)) = (UN\ x.\ ntrunc\ k\ (f\ x))$
⟨*proof*⟩

lemma *Scons-UN1-x*: $Scons\ (UN\ x.\ f\ x)\ M = (UN\ x.\ Scons\ (f\ x)\ M)$
⟨*proof*⟩

lemma *Scons-UN1-y*: $Scons\ M\ (UN\ x.\ f\ x) = (UN\ x.\ Scons\ M\ (f\ x))$
⟨*proof*⟩

lemma *In0-UN1*: $In0(UN\ x.\ f(x)) = (UN\ x.\ In0(f(x)))$
⟨*proof*⟩

lemma *In1-UN1*: $In1(UN\ x.\ f(x)) = (UN\ x.\ In1(f(x)))$
⟨*proof*⟩

lemma *dprodI* [*intro*!]:
$[|\ (M,M'):r;\ \ (N,N'):s\ |] ==> (Scons\ M\ N,\ Scons\ M'\ N') : dprod\ r\ s$
⟨*proof*⟩

lemma *dprodE* [*elim*!]:
$[|\ c : dprod\ r\ s;$
$\quad !!x\ y\ x'\ y'.\ [|\ (x,x') : r;\ \ (y,y') : s;$
$\qquad\qquad c = (Scons\ x\ y,\ Scons\ x'\ y')\ |] ==> P$
$\quad |] ==> P$
⟨*proof*⟩

lemma *dsum-In0I* [*intro*]: (*M*,*M'*):*r* ==> (*In0*(*M*), *In0*(*M'*)) : *dsum r s*
⟨*proof*⟩

lemma *dsum-In1I* [*intro*]: (*N*,*N'*):*s* ==> (*In1*(*N*), *In1*(*N'*)) : *dsum r s*
⟨*proof*⟩

lemma *dsumE* [*elim*!]:
[| *w* : *dsum r s*;
!!*x x'*. [| (*x*,*x'*) : *r*; *w* = (*In0*(*x*), *In0*(*x'*)) |] ==> *P*;
!!*y y'*. [| (*y*,*y'*) : *s*; *w* = (*In1*(*y*), *In1*(*y'*)) |] ==> *P*
|] ==> *P*
⟨*proof*⟩

lemma *dprod-mono*: [| *r*<=*r'*; *s*<=*s'* |] ==> *dprod r s* <= *dprod r' s'*
⟨*proof*⟩

lemma *dsum-mono*: [| *r*<=*r'*; *s*<=*s'* |] ==> *dsum r s* <= *dsum r' s'*
⟨*proof*⟩

lemma *dprod-Sigma*: (*dprod* (*A* × *B*) (*C* × *D*)) <= (*uprod A C*) × (*uprod B D*)
⟨*proof*⟩

lemmas *dprod-subset-Sigma* = *subset-trans* [*OF dprod-mono dprod-Sigma*]

lemma *dprod-subset-Sigma2*:
(*dprod* (*Sigma A B*) (*Sigma C D*)) <= *Sigma* (*uprod A C*) (*Split* (%*x y*. *uprod* (*B x*) (*D y*)))
⟨*proof*⟩

lemma *dsum-Sigma*: (*dsum* (*A* × *B*) (*C* × *D*)) <= (*usum A C*) × (*usum B D*)
⟨*proof*⟩

lemmas *dsum-subset-Sigma* = *subset-trans* [*OF dsum-mono dsum-Sigma*]

lemma *Domain-dprod* [*simp*]: *Domain* (*dprod r s*) = *uprod* (*Domain r*) (*Domain s*)
⟨*proof*⟩

lemma *Domain-dsum* [*simp*]: *Domain* (*dsum r s*) = *usum* (*Domain r*) (*Domain s*)
⟨*proof*⟩

hides popular names

hide-type (**open**) *node item*
hide-const (**open**) *Push Node Atom Leaf Numb Lim Split Case*

⟨*ML*⟩

end

3 Bijections between natural numbers and other types

theory *Nat-Bijection*
imports *Main*
begin

3.1 Type *nat* × *nat*

Triangle numbers: 0, 1, 3, 6, 10, 15, ...

definition *triangle* :: *nat* ⇒ *nat*
where *triangle n* = (*n* ∗ *Suc n*) *div 2*

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

lemma *triangle-Suc* [*simp*]: *triangle* (*Suc n*) = *triangle n* + *Suc n*
⟨*proof*⟩

definition *prod-encode* :: *nat* × *nat* ⇒ *nat*
where *prod-encode* = (λ(*m*, *n*). *triangle* (*m* + *n*) + *m*)

In this auxiliary function, *triangle k* + *m* is an invariant.

fun *prod-decode-aux* :: *nat* ⇒ *nat* ⇒ *nat* × *nat*
where
prod-decode-aux k m =
(*if m* ≤ *k then* (*m*, *k* − *m*) *else prod-decode-aux* (*Suc k*) (*m* − *Suc k*))

declare *prod-decode-aux.simps* [*simp del*]

definition *prod-decode* :: *nat* ⇒ *nat* × *nat*
where *prod-decode* = *prod-decode-aux 0*

lemma *prod-encode-prod-decode-aux*:
prod-encode (*prod-decode-aux k m*) = *triangle k* + *m*

⟨*proof*⟩

lemma *prod-decode-inverse* [*simp*]: *prod-encode* (*prod-decode n*) = *n*
⟨*proof*⟩

lemma *prod-decode-triangle-add*: *prod-decode* (*triangle k* + *m*) = *prod-decode-aux k m*
⟨*proof*⟩

lemma *prod-encode-inverse* [*simp*]: *prod-decode* (*prod-encode x*) = *x*
⟨*proof*⟩

lemma *inj-prod-encode*: *inj-on prod-encode A*
⟨*proof*⟩

lemma *inj-prod-decode*: *inj-on prod-decode A*
⟨*proof*⟩

lemma *surj-prod-encode*: *surj prod-encode*
⟨*proof*⟩

lemma *surj-prod-decode*: *surj prod-decode*
⟨*proof*⟩

lemma *bij-prod-encode*: *bij prod-encode*
⟨*proof*⟩

lemma *bij-prod-decode*: *bij prod-decode*
⟨*proof*⟩

lemma *prod-encode-eq*: *prod-encode x* = *prod-encode y* ⟷ *x* = *y*
⟨*proof*⟩

lemma *prod-decode-eq*: *prod-decode x* = *prod-decode y* ⟷ *x* = *y*
⟨*proof*⟩

Ordering properties

lemma *le-prod-encode-1*: *a* ≤ *prod-encode* (*a*, *b*)
⟨*proof*⟩

lemma *le-prod-encode-2*: *b* ≤ *prod-encode* (*a*, *b*)
⟨*proof*⟩

3.2 Type *nat* + *nat*

definition *sum-encode* :: *nat* + *nat* ⇒ *nat*
where
sum-encode x = (*case x of Inl a* ⇒ *2* ∗ *a* | *Inr b* ⇒ *Suc* (*2* ∗ *b*))

definition *sum-decode* :: *nat* ⇒ *nat* + *nat*
where
sum-decode n = (*if even n then Inl* (*n div 2*) *else Inr* (*n div 2*))

lemma *sum-encode-inverse* [*simp*]: *sum-decode* (*sum-encode x*) = *x*
⟨*proof*⟩

lemma *sum-decode-inverse* [*simp*]: *sum-encode* (*sum-decode n*) = *n*
⟨*proof*⟩

lemma *inj-sum-encode*: *inj-on sum-encode A*
⟨*proof*⟩

lemma *inj-sum-decode*: *inj-on sum-decode A*
⟨*proof*⟩

lemma *surj-sum-encode*: *surj sum-encode*
⟨*proof*⟩

lemma *surj-sum-decode*: *surj sum-decode*
⟨*proof*⟩

lemma *bij-sum-encode*: *bij sum-encode*
⟨*proof*⟩

lemma *bij-sum-decode*: *bij sum-decode*
⟨*proof*⟩

lemma *sum-encode-eq*: *sum-encode x* = *sum-encode y* ⟷ *x* = *y*
⟨*proof*⟩

lemma *sum-decode-eq*: *sum-decode x* = *sum-decode y* ⟷ *x* = *y*
⟨*proof*⟩

3.3 Type *int*

definition *int-encode* :: *int* ⇒ *nat*
where
int-encode i = *sum-encode* (*if* 0 ≤ *i then Inl* (*nat i*) *else Inr* (*nat* (− *i* − 1)))

definition *int-decode* :: *nat* ⇒ *int*
where
int-decode n = (*case sum-decode n of Inl a* ⇒ *int a* | *Inr b* ⇒ − *int b* − 1)

lemma *int-encode-inverse* [*simp*]: *int-decode* (*int-encode x*) = *x*
⟨*proof*⟩

lemma *int-decode-inverse* [*simp*]: *int-encode* (*int-decode n*) = *n*
⟨*proof*⟩

lemma *inj-int-encode*: *inj-on int-encode A*
⟨*proof*⟩

lemma *inj-int-decode*: *inj-on int-decode A*
⟨*proof*⟩

lemma *surj-int-encode*: *surj int-encode*
⟨*proof*⟩

lemma *surj-int-decode*: *surj int-decode*
⟨*proof*⟩

lemma *bij-int-encode*: *bij int-encode*
⟨*proof*⟩

lemma *bij-int-decode*: *bij int-decode*
⟨*proof*⟩

lemma *int-encode-eq*: *int-encode x = int-encode y* ⟷ *x = y*
⟨*proof*⟩

lemma *int-decode-eq*: *int-decode x = int-decode y* ⟷ *x = y*
⟨*proof*⟩

3.4 Type *nat list*

fun *list-encode* :: *nat list* ⇒ *nat*
where
list-encode [] = *0*
| *list-encode* (*x* # *xs*) = *Suc* (*prod-encode* (*x*, *list-encode xs*))

function *list-decode* :: *nat* ⇒ *nat list*
where
list-decode 0 = []
| *list-decode* (*Suc n*) = (*case prod-decode n of* (*x*, *y*) ⇒ *x* # *list-decode y*)
⟨*proof*⟩

termination *list-decode*
⟨*proof*⟩

lemma *list-encode-inverse* [*simp*]: *list-decode* (*list-encode x*) = *x*
⟨*proof*⟩

lemma *list-decode-inverse* [*simp*]: *list-encode* (*list-decode n*) = *n*
⟨*proof*⟩

lemma *inj-list-encode*: *inj-on list-encode A*
⟨*proof*⟩

lemma *inj-list-decode*: *inj-on list-decode A*
⟨*proof*⟩

lemma *surj-list-encode*: *surj list-encode*
⟨*proof*⟩

lemma *surj-list-decode*: *surj list-decode*
⟨*proof*⟩

lemma *bij-list-encode*: *bij list-encode*
⟨*proof*⟩

lemma *bij-list-decode*: *bij list-decode*
⟨*proof*⟩

lemma *list-encode-eq*: *list-encode x = list-encode y ⟷ x = y*
⟨*proof*⟩

lemma *list-decode-eq*: *list-decode x = list-decode y ⟷ x = y*
⟨*proof*⟩

3.5 Finite sets of naturals

3.5.1 Preliminaries

lemma *finite-vimage-Suc-iff*: *finite (Suc −‘ F) ⟷ finite F*
⟨*proof*⟩

lemma *vimage-Suc-insert-0*: *Suc −‘ insert 0 A = Suc −‘ A*
⟨*proof*⟩

lemma *vimage-Suc-insert-Suc*:
Suc −‘ insert (Suc n) A = insert n (Suc −‘ A)
⟨*proof*⟩

lemma *div2-even-ext-nat*:
fixes *x y* :: *nat*
assumes *x div 2 = y div 2*
and *even x ⟷ even y*
shows *x = y*
⟨*proof*⟩

3.5.2 From sets to naturals

definition *set-encode* :: *nat set ⇒ nat*
where *set-encode = setsum (op ^ 2)*

lemma *set-encode-empty* [*simp*]: *set-encode {} = 0*
⟨*proof*⟩

lemma *set-encode-inf*: $\sim$ *finite A* $\Longrightarrow$ *set-encode A = 0*
⟨*proof*⟩

lemma *set-encode-insert* [*simp*]:
⟦*finite A*; *n* $\notin$ *A*⟧ $\Longrightarrow$ *set-encode* (*insert n A*) = *2^n* + *set-encode A*
⟨*proof*⟩

lemma *even-set-encode-iff*: *finite A* $\Longrightarrow$ *even* (*set-encode A*) $\longleftrightarrow$ *0* $\notin$ *A*
⟨*proof*⟩

lemma *set-encode-vimage-Suc*: *set-encode* (*Suc* −‘ *A*) = *set-encode A div 2*
⟨*proof*⟩

lemmas *set-encode-div-2* = *set-encode-vimage-Suc* [*symmetric*]

3.5.3 From naturals to sets

definition *set-decode* :: *nat* ⇒ *nat set*
where *set-decode x* = {*n*. *odd* (*x div 2* ^ *n*)}

lemma *set-decode-0* [*simp*]: *0* $\in$ *set-decode x* $\longleftrightarrow$ *odd x*
⟨*proof*⟩

lemma *set-decode-Suc* [*simp*]:
Suc n $\in$ *set-decode x* $\longleftrightarrow$ *n* $\in$ *set-decode* (*x div 2*)
⟨*proof*⟩

lemma *set-decode-zero* [*simp*]: *set-decode 0* = {}
⟨*proof*⟩

lemma *set-decode-div-2*: *set-decode* (*x div 2*) = *Suc* −‘ *set-decode x*
⟨*proof*⟩

lemma *set-decode-plus-power-2*:
n $\notin$ *set-decode z* $\Longrightarrow$ *set-decode* (*2* ^ *n* + *z*) = *insert n* (*set-decode z*)
⟨*proof*⟩

lemma *finite-set-decode* [*simp*]: *finite* (*set-decode n*)
⟨*proof*⟩

3.5.4 Proof of isomorphism

lemma *set-decode-inverse* [*simp*]: *set-encode* (*set-decode n*) = *n*
⟨*proof*⟩

lemma *set-encode-inverse* [*simp*]: *finite A* $\Longrightarrow$ *set-decode* (*set-encode A*) = *A*
⟨*proof*⟩

lemma *inj-on-set-encode*: *inj-on set-encode* (*Collect finite*)

⟨*proof*⟩

lemma *set-encode-eq*:
⟦*finite A*; *finite B*⟧ ⟹ *set-encode A* = *set-encode B* ⟷ *A* = *B*
⟨*proof*⟩

lemma *subset-decode-imp-le*:
assumes *set-decode m* ⊆ *set-decode n*
shows *m* ≤ *n*
⟨*proof*⟩

end

4 Encoding (almost) everything into natural numbers

theory *Countable*
imports *Old-Datatype* ~~/*src*/*HOL*/*Rat Nat-Bijection*
begin

4.1 The class of countable types

class *countable* =
assumes *ex-inj*: ∃ *to-nat* :: 'a ⇒ *nat*. *inj to-nat*

lemma *countable-classI*:
fixes *f* :: 'a ⇒ *nat*
assumes ⋀*x y*. *f x* = *f y* ⟹ *x* = *y*
shows *OFCLASS*('a, *countable-class*)
⟨*proof*⟩

4.2 Conversion functions

definition *to-nat* :: 'a::*countable* ⇒ *nat* **where**
to-nat = (*SOME f*. *inj f*)

definition *from-nat* :: *nat* ⇒ 'a::*countable* **where**
from-nat = *inv* (*to-nat* :: 'a ⇒ *nat*)

lemma *inj-to-nat* [*simp*]: *inj to-nat*
⟨*proof*⟩

lemma *inj-on-to-nat*[*simp*, *intro*]: *inj-on to-nat S*
⟨*proof*⟩

lemma *surj-from-nat* [*simp*]: *surj from-nat*
⟨*proof*⟩

lemma *to-nat-split* [*simp*]: *to-nat x* = *to-nat y* ⟷ *x* = *y*

⟨*proof*⟩

lemma *from-nat-to-nat* [*simp*]:
from-nat (*to-nat x*) = *x*
⟨*proof*⟩

4.3 Finite types are countable

subclass (**in** *finite*) *countable*
⟨*proof*⟩

4.4 Automatically proving countability of old-style datatypes

context
begin

qualified inductive *finite-item* :: *'a Old-Datatype.item* ⇒ *bool* **where**
undefined: *finite-item undefined*
| *In0*: *finite-item x* ⟹ *finite-item* (*Old-Datatype.In0 x*)
| *In1*: *finite-item x* ⟹ *finite-item* (*Old-Datatype.In1 x*)
| *Leaf*: *finite-item* (*Old-Datatype.Leaf a*)
| *Scons*: ⟦*finite-item x*; *finite-item y*⟧ ⟹ *finite-item* (*Old-Datatype.Scons x y*)

qualified function *nth-item* :: *nat* ⇒ (*'a*::*countable*) *Old-Datatype.item*
where
nth-item 0 = *undefined*
| *nth-item* (*Suc n*) =
(*case sum-decode n of*
Inl i ⇒
(*case sum-decode i of*
Inl j ⇒ *Old-Datatype.In0* (*nth-item j*)
| *Inr j* ⇒ *Old-Datatype.In1* (*nth-item j*))
| *Inr i* ⇒
(*case sum-decode i of*
Inl j ⇒ *Old-Datatype.Leaf* (*from-nat j*)
| *Inr j* ⇒
(*case prod-decode j of*
(*a*, *b*) ⇒ *Old-Datatype.Scons* (*nth-item a*) (*nth-item b*))))
⟨*proof*⟩

lemma *le-sum-encode-Inl*: *x* ≤ *y* ⟹ *x* ≤ *sum-encode* (*Inl y*)
⟨*proof*⟩

lemma *le-sum-encode-Inr*: *x* ≤ *y* ⟹ *x* ≤ *sum-encode* (*Inr y*)
⟨*proof*⟩ **termination**
⟨*proof*⟩

lemma *nth-item-covers*: *finite-item x* ⟹ ∃ *n*. *nth-item n* = *x*
⟨*proof*⟩

theorem *countable-datatype*:
fixes *Rep* :: $'b \Rightarrow ('a::countable)$ *Old-Datatype.item*
fixes *Abs* :: $('a::countable)$ *Old-Datatype.item* $\Rightarrow 'b$
fixes *rep-set* :: $('a::countable)$ *Old-Datatype.item* $\Rightarrow$ *bool*
assumes *type*: *type-definition Rep Abs* (*Collect rep-set*)
assumes *finite-item*: $\bigwedge x.$ *rep-set* $x \Longrightarrow$ *finite-item* x
shows *OFCLASS*($'b$, *countable-class*)
⟨*proof*⟩

⟨*ML*⟩

end

4.5 Automatically proving countability of datatypes

⟨*ML*⟩

4.6 More Countable types

Naturals

instance *nat* :: *countable*
⟨*proof*⟩

Pairs

instance *prod* :: (*countable*, *countable*) *countable*
⟨*proof*⟩

Sums

instance *sum* :: (*countable*, *countable*) *countable*
⟨*proof*⟩

Integers

instance *int* :: *countable*
⟨*proof*⟩

Options

instance *option* :: (*countable*) *countable*
⟨*proof*⟩

Lists

instance *list* :: (*countable*) *countable*
⟨*proof*⟩

String literals

instance *String.literal* :: *countable*
⟨*proof*⟩

Functions

instance *fun* :: (*finite*, *countable*) *countable*
⟨*proof*⟩

Typereps

instance *typerep* :: *countable*
⟨*proof*⟩

4.7 The rationals are countably infinite

definition *nat-to-rat-surj* :: *nat* ⇒ *rat* **where**
nat-to-rat-surj n = (*let* (*a*, *b*) = *prod-decode n in Fract* (*int-decode a*) (*int-decode b*))

lemma *surj-nat-to-rat-surj*: *surj nat-to-rat-surj*
⟨*proof*⟩

lemma *Rats-eq-range-nat-to-rat-surj*: ℚ = *range nat-to-rat-surj*
⟨*proof*⟩

context *field-char-0*
begin

lemma *Rats-eq-range-of-rat-o-nat-to-rat-surj*:
ℚ = *range* (*of-rat* ∘ *nat-to-rat-surj*)
⟨*proof*⟩

lemma *surj-of-rat-nat-to-rat-surj*:
r ∈ ℚ ⟹ ∃ *n*. *r* = *of-rat* (*nat-to-rat-surj n*)
⟨*proof*⟩

end

instance *rat* :: *countable*
⟨*proof*⟩

end

5 Infinite Sets and Related Concepts

theory *Infinite-Set*
imports *Main*
begin

The set of natural numbers is infinite.

lemma *infinite-nat-iff-unbounded-le*: *infinite* (*S*::*nat set*) ⟷ (∀ *m*. ∃ *n*≥*m*. *n* ∈ *S*)
⟨*proof*⟩

lemma *infinite-nat-iff-unbounded*: *infinite* (*S*::*nat set*) ⟷ (∀ *m*. ∃ *n*>*m*. *n* ∈ *S*)
⟨*proof*⟩

lemma *finite-nat-iff-bounded*: *finite* (*S*::*nat set*) ⟷ (∃ *k*. *S* ⊆ {..<*k*})
⟨*proof*⟩

lemma *finite-nat-iff-bounded-le*: *finite* (*S*::*nat set*) ⟷ (∃ *k*. *S* ⊆ {.. *k*})
⟨*proof*⟩

lemma *finite-nat-bounded*: *finite* (*S*::*nat set*) ⟹ ∃ *k*. *S* ⊆ {..<*k*}
⟨*proof*⟩

For a set of natural numbers to be infinite, it is enough to know that for any number larger than some k, there is some larger number that is an element of the set.

lemma *unbounded-k-infinite*: ∀ *m*>*k*. ∃ *n*>*m*. *n* ∈ *S* ⟹ *infinite* (*S*::*nat set*)
⟨*proof*⟩

lemma *nat-not-finite*: *finite* (*UNIV*::*nat set*) ⟹ *R*
⟨*proof*⟩

lemma *range-inj-infinite*:
inj (*f*::*nat* ⇒ *'a*) ⟹ *infinite* (*range f*)
⟨*proof*⟩

The set of integers is also infinite.

lemma *infinite-int-iff-infinite-nat-abs*: *infinite* (*S*::*int set*) ⟷ *infinite* ((*nat* ∘ *abs*) ` *S*)
⟨*proof*⟩

proposition *infinite-int-iff-unbounded-le*: *infinite* (*S*::*int set*) ⟷ (∀ *m*. ∃ *n*. |*n*| ≥ *m* ∧ *n* ∈ *S*)
⟨*proof*⟩

proposition *infinite-int-iff-unbounded*: *infinite* (*S*::*int set*) ⟷ (∀ *m*. ∃ *n*. |*n*| > *m* ∧ *n* ∈ *S*)
⟨*proof*⟩

proposition *finite-int-iff-bounded*: *finite* (*S*::*int set*) ⟷ (∃ *k*. *abs* ` *S* ⊆ {..<*k*})
⟨*proof*⟩

proposition *finite-int-iff-bounded-le*: *finite* (*S*::*int set*) ⟷ (∃ *k*. *abs* ` *S* ⊆ {.. *k*})
⟨*proof*⟩

5.1 Infinitely Many and Almost All

We often need to reason about the existence of infinitely many (resp., all but finitely many) objects satisfying some predicate, so we introduce corresponding binders and their proof rules.

lemma *not-INFM* [*simp*]: $\neg\ (INFM\ x.\ P\ x) \longleftrightarrow (MOST\ x.\ \neg\ P\ x)$ ⟨*proof*⟩
lemma *not-MOST* [*simp*]: $\neg\ (MOST\ x.\ P\ x) \longleftrightarrow (INFM\ x.\ \neg\ P\ x)$ ⟨*proof*⟩

lemma *INFM-const* [*simp*]: $(INFM\ x{::}'a.\ P) \longleftrightarrow P \wedge infinite\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *MOST-const* [*simp*]: $(MOST\ x{::}'a.\ P) \longleftrightarrow P \vee finite\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *INFM-imp-distrib*: $(INFM\ x.\ P\ x \longrightarrow Q\ x) \longleftrightarrow ((MOST\ x.\ P\ x) \longrightarrow (INFM\ x.\ Q\ x))$
⟨*proof*⟩

lemma *MOST-imp-iff*: $MOST\ x.\ P\ x \Longrightarrow (MOST\ x.\ P\ x \longrightarrow Q\ x) \longleftrightarrow (MOST\ x.\ Q\ x)$
⟨*proof*⟩

lemma *INFM-conjI*: $INFM\ x.\ P\ x \Longrightarrow MOST\ x.\ Q\ x \Longrightarrow INFM\ x.\ P\ x \wedge Q\ x$
⟨*proof*⟩

Properties of quantifiers with injective functions.

lemma *INFM-inj*: $INFM\ x.\ P\ (f\ x) \Longrightarrow inj\ f \Longrightarrow INFM\ x.\ P\ x$
⟨*proof*⟩

lemma *MOST-inj*: $MOST\ x.\ P\ x \Longrightarrow inj\ f \Longrightarrow MOST\ x.\ P\ (f\ x)$
⟨*proof*⟩

Properties of quantifiers with singletons.

lemma *not-INFM-eq* [*simp*]:
$\neg\ (INFM\ x.\ x = a)$
$\neg\ (INFM\ x.\ a = x)$
⟨*proof*⟩

lemma *MOST-neq* [*simp*]:
$MOST\ x.\ x \neq a$
$MOST\ x.\ a \neq x$
⟨*proof*⟩

lemma *INFM-neq* [*simp*]:
$(INFM\ x{::}'a.\ x \neq a) \longleftrightarrow infinite\ (UNIV{::}'a\ set)$
$(INFM\ x{::}'a.\ a \neq x) \longleftrightarrow infinite\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *MOST-eq* [*simp*]:
$(MOST\ x{::}'a.\ x = a) \longleftrightarrow finite\ (UNIV{::}'a\ set)$
$(MOST\ x{::}'a.\ a = x) \longleftrightarrow finite\ (UNIV{::}'a\ set)$
⟨*proof*⟩

lemma *MOST-eq-imp*:

MOST x. x = a ⟶ P x
MOST x. a = x ⟶ P x
⟨*proof*⟩

Properties of quantifiers over the naturals.

lemma *MOST-nat*: $(\forall_\infty n.\ P\ (n::nat)) \longleftrightarrow (\exists m.\ \forall n{>}m.\ P\ n)$
⟨*proof*⟩

lemma *MOST-nat-le*: $(\forall_\infty n.\ P\ (n::nat)) \longleftrightarrow (\exists m.\ \forall n{\geq}m.\ P\ n)$
⟨*proof*⟩

lemma *INFM-nat*: $(\exists_\infty n.\ P\ (n::nat)) \longleftrightarrow (\forall m.\ \exists n{>}m.\ P\ n)$
⟨*proof*⟩

lemma *INFM-nat-le*: $(\exists_\infty n.\ P\ (n::nat)) \longleftrightarrow (\forall m.\ \exists n{\geq}m.\ P\ n)$
⟨*proof*⟩

lemma *MOST-INFM*: *infinite* $(UNIV::'a\ set) \Longrightarrow MOST\ x::'a.\ P\ x \Longrightarrow INFM\ x::'a.\ P\ x$
⟨*proof*⟩

lemma *MOST-Suc-iff*: $(MOST\ n.\ P\ (Suc\ n)) \longleftrightarrow (MOST\ n.\ P\ n)$
⟨*proof*⟩

lemma
shows *MOST-SucI*: $MOST\ n.\ P\ n \Longrightarrow MOST\ n.\ P\ (Suc\ n)$
and *MOST-SucD*: $MOST\ n.\ P\ (Suc\ n) \Longrightarrow MOST\ n.\ P\ n$
⟨*proof*⟩

lemma *MOST-ge-nat*: $MOST\ n::nat.\ m \leq n$
⟨*proof*⟩

lemma *Inf-many-def*: *Inf-many P* ⟷ *infinite* $\{x.\ P\ x\}$ ⟨*proof*⟩
lemma *Alm-all-def*: *Alm-all P* ⟷ $\neg\ (INFM\ x.\ \neg\ P\ x)$ ⟨*proof*⟩
lemma *INFM-iff-infinite*: $(INFM\ x.\ P\ x) \longleftrightarrow$ *infinite* $\{x.\ P\ x\}$ ⟨*proof*⟩
lemma *MOST-iff-cofinite*: $(MOST\ x.\ P\ x) \longleftrightarrow$ *finite* $\{x.\ \neg\ P\ x\}$ ⟨*proof*⟩
lemma *INFM-EX*: $(\exists_\infty x.\ P\ x) \Longrightarrow (\exists x.\ P\ x)$ ⟨*proof*⟩
lemma *ALL-MOST*: $\forall x.\ P\ x \Longrightarrow \forall_\infty x.\ P\ x$ ⟨*proof*⟩
lemma *INFM-mono*: $\exists_\infty x.\ P\ x \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow Q\ x) \Longrightarrow \exists_\infty x.\ Q\ x$ ⟨*proof*⟩
lemma *MOST-mono*: $\forall_\infty x.\ P\ x \Longrightarrow (\bigwedge x.\ P\ x \Longrightarrow Q\ x) \Longrightarrow \forall_\infty x.\ Q\ x$ ⟨*proof*⟩
lemma *INFM-disj-distrib*: $(\exists_\infty x.\ P\ x \vee Q\ x) \longleftrightarrow (\exists_\infty x.\ P\ x) \vee (\exists_\infty x.\ Q\ x)$ ⟨*proof*⟩
lemma *MOST-rev-mp*: $\forall_\infty x.\ P\ x \Longrightarrow \forall_\infty x.\ P\ x \longrightarrow Q\ x \Longrightarrow \forall_\infty x.\ Q\ x$ ⟨*proof*⟩
lemma *MOST-conj-distrib*: $(\forall_\infty x.\ P\ x \wedge Q\ x) \longleftrightarrow (\forall_\infty x.\ P\ x) \wedge (\forall_\infty x.\ Q\ x)$ ⟨*proof*⟩
lemma *MOST-conjI*: $MOST\ x.\ P\ x \Longrightarrow MOST\ x.\ Q\ x \Longrightarrow MOST\ x.\ P\ x \wedge Q\ x$ ⟨*proof*⟩
lemma *INFM-finite-Bex-distrib*: *finite A* $\Longrightarrow (INFM\ y.\ \exists x{\in}A.\ P\ x\ y) \longleftrightarrow (\exists x{\in}A.$

INFM y. P x y) ⟨*proof*⟩
lemma *MOST-finite-Ball-distrib*: *finite A* ⟹ (*MOST y.* ∀ *x*∈*A. P x y*) ⟷ (∀ *x*∈*A. MOST y. P x y*) ⟨*proof*⟩
lemma *INFM-E*: *INFM x. P x* ⟹ (⋀*x. P x* ⟹ *thesis*) ⟹ *thesis* ⟨*proof*⟩
lemma *MOST-I*: (⋀*x. P x*) ⟹ *MOST x. P x* ⟨*proof*⟩
lemmas *MOST-iff-finiteNeg* = *MOST-iff-cofinite*

5.2 Enumeration of an Infinite Set

The set's element type must be wellordered (e.g. the natural numbers).

Could be generalized to *enumerate′ S n* = (*SOME t. t* ∈ *s* ∧ *finite* {*s* ∈ *S. s* < *t*} ∧ *card* {*s* ∈ *S. s* < *t*} = *n*).

primrec (**in** *wellorder*) *enumerate* :: *'a set* ⇒ *nat* ⇒ *'a*
where
enumerate-0: *enumerate S 0* = (*LEAST n. n* ∈ *S*)
| *enumerate-Suc*: *enumerate S* (*Suc n*) = *enumerate* (*S* − {*LEAST n. n* ∈ *S*}) *n*

lemma *enumerate-Suc′*: *enumerate S* (*Suc n*) = *enumerate* (*S* − {*enumerate S 0*}) *n*
⟨*proof*⟩

lemma *enumerate-in-set*: *infinite S* ⟹ *enumerate S n* ∈ *S*
⟨*proof*⟩

declare *enumerate-0* [*simp del*] *enumerate-Suc* [*simp del*]

lemma *enumerate-step*: *infinite S* ⟹ *enumerate S n* < *enumerate S* (*Suc n*)
⟨*proof*⟩

lemma *enumerate-mono*: *m* < *n* ⟹ *infinite S* ⟹ *enumerate S m* < *enumerate S n*
⟨*proof*⟩

lemma *le-enumerate*:
assumes *S*: *infinite S*
shows *n* ≤ *enumerate S n*
⟨*proof*⟩

lemma *enumerate-Suc″*:
fixes *S* :: *'a*::*wellorder set*
assumes *infinite S*
shows *enumerate S* (*Suc n*) = (*LEAST s. s* ∈ *S* ∧ *enumerate S n* < *s*)
⟨*proof*⟩

lemma *enumerate-Ex*:
assumes *S*: *infinite* (*S*::*nat set*)
shows *s* ∈ *S* ⟹ ∃ *n. enumerate S n* = *s*

⟨*proof*⟩

lemma *bij-enumerate*:
fixes *S* :: *nat set*
assumes *S*: *infinite S*
shows *bij-betw* (*enumerate S*) *UNIV S*
⟨*proof*⟩

end

6 Countable sets

theory *Countable-Set*
imports *Countable Infinite-Set*
begin

6.1 Predicate for countable sets

definition *countable* :: *'a set* ⇒ *bool* **where**
countable S ⟷ (∃*f*::*'a* ⇒ *nat*. *inj-on f S*)

lemma *countableE*:
assumes *S*: *countable S* **obtains** *f* :: *'a* ⇒ *nat* **where** *inj-on f S*
⟨*proof*⟩

lemma *countableI*: *inj-on* (*f*::*'a* ⇒ *nat*) *S* ⟹ *countable S*
⟨*proof*⟩

lemma *countableI'*: *inj-on* (*f*::*'a* ⇒ *'b*::*countable*) *S* ⟹ *countable S*
⟨*proof*⟩

lemma *countableE-bij*:
assumes *S*: *countable S* **obtains** *f* :: *nat* ⇒ *'a* **and** *C* :: *nat set* **where** *bij-betw f C S*
⟨*proof*⟩

lemma *countableI-bij*: *bij-betw f* (*C*::*nat set*) *S* ⟹ *countable S*
⟨*proof*⟩

lemma *countable-finite*: *finite S* ⟹ *countable S*
⟨*proof*⟩

lemma *countableI-bij1*: *bij-betw f A B* ⟹ *countable A* ⟹ *countable B*
⟨*proof*⟩

lemma *countableI-bij2*: *bij-betw f B A* ⟹ *countable A* ⟹ *countable B*
⟨*proof*⟩

lemma *countable-iff-bij*[*simp*]: *bij-betw f A B* ⟹ *countable A* ⟷ *countable B*

⟨*proof*⟩

lemma *countable-subset*: $A \subseteq B \Longrightarrow$ *countable* $B \Longrightarrow$ *countable* A
⟨*proof*⟩

lemma *countableI-type*[*intro*, *simp*]: *countable* (A:: $'a$ :: *countable set*)
⟨*proof*⟩

6.2 Enumerate a countable set

lemma *countableE-infinite*:
assumes *countable S infinite S*
obtains e :: $'a \Rightarrow$ *nat* **where** *bij-betw e S UNIV*
⟨*proof*⟩

lemma *countable-enum-cases*:
assumes *countable S*
obtains (*finite*) f :: $'a \Rightarrow$ *nat* **where** *finite S bij-betw f S* {..<*card S*}
| (*infinite*) f :: $'a \Rightarrow$ *nat* **where** *infinite S bij-betw f S UNIV*
⟨*proof*⟩

definition *to-nat-on* :: $'a$ *set* $\Rightarrow$ $'a \Rightarrow$ *nat* **where**
to-nat-on S = (*SOME f*. *if finite S then bij-betw f S* {..< *card S*} *else bij-betw f S UNIV*)

definition *from-nat-into* :: $'a$ *set* $\Rightarrow$ *nat* $\Rightarrow$ $'a$ **where**
from-nat-into S n = (*if n* ∈ *to-nat-on S* ' *S then inv-into S* (*to-nat-on S*) *n else SOME s*. *s*∈*S*)

lemma *to-nat-on-finite*: *finite S* $\Longrightarrow$ *bij-betw* (*to-nat-on S*) *S* {..< *card S*}
⟨*proof*⟩

lemma *to-nat-on-infinite*: *countable S* $\Longrightarrow$ *infinite S* $\Longrightarrow$ *bij-betw* (*to-nat-on S*) *S UNIV*
⟨*proof*⟩

lemma *bij-betw-from-nat-into-finite*: *finite S* $\Longrightarrow$ *bij-betw* (*from-nat-into S*) {..< *card S*} *S*
⟨*proof*⟩

lemma *bij-betw-from-nat-into*: *countable S* $\Longrightarrow$ *infinite S* $\Longrightarrow$ *bij-betw* (*from-nat-into S*) *UNIV S*
⟨*proof*⟩

lemma *inj-on-to-nat-on*[*intro*]: *countable A* $\Longrightarrow$ *inj-on* (*to-nat-on A*) *A*
⟨*proof*⟩

lemma *to-nat-on-inj*[*simp*]:
countable A $\Longrightarrow$ $a \in A \Longrightarrow b \in A \Longrightarrow$ *to-nat-on A a* = *to-nat-on A b* $\longleftrightarrow$ $a =$

b
⟨*proof*⟩

lemma *from-nat-into-to-nat-on*[*simp*]: *countable A* ⟹ *a* ∈ *A* ⟹ *from-nat-into A* (*to-nat-on A a*) = *a*
⟨*proof*⟩

lemma *subset-range-from-nat-into*: *countable A* ⟹ *A* ⊆ *range* (*from-nat-into A*)
⟨*proof*⟩

lemma *from-nat-into*: *A* ≠ {} ⟹ *from-nat-into A n* ∈ *A*
⟨*proof*⟩

lemma *range-from-nat-into-subset*: *A* ≠ {} ⟹ *range* (*from-nat-into A*) ⊆ *A*
⟨*proof*⟩

lemma *range-from-nat-into*[*simp*]: *A* ≠ {} ⟹ *countable A* ⟹ *range* (*from-nat-into A*) = *A*
⟨*proof*⟩

lemma *image-to-nat-on*: *countable A* ⟹ *infinite A* ⟹ *to-nat-on A* ' *A* = *UNIV*
⟨*proof*⟩

lemma *to-nat-on-surj*: *countable A* ⟹ *infinite A* ⟹ ∃ *a*∈*A*. *to-nat-on A a* = *n*
⟨*proof*⟩

lemma *to-nat-on-from-nat-into*[*simp*]: *n* ∈ *to-nat-on A* ' *A* ⟹ *to-nat-on A* (*from-nat-into A n*) = *n*
⟨*proof*⟩

lemma *to-nat-on-from-nat-into-infinite*[*simp*]:
countable A ⟹ *infinite A* ⟹ *to-nat-on A* (*from-nat-into A n*) = *n*
⟨*proof*⟩

lemma *from-nat-into-inj*:
countable A ⟹ *m* ∈ *to-nat-on A* ' *A* ⟹ *n* ∈ *to-nat-on A* ' *A* ⟹
from-nat-into A m = *from-nat-into A n* ⟷ *m* = *n*
⟨*proof*⟩

lemma *from-nat-into-inj-infinite*[*simp*]:
countable A ⟹ *infinite A* ⟹ *from-nat-into A m* = *from-nat-into A n* ⟷ *m* = *n*
⟨*proof*⟩

lemma *eq-from-nat-into-iff*:
countable A ⟹ *x* ∈ *A* ⟹ *i* ∈ *to-nat-on A* ' *A* ⟹ *x* = *from-nat-into A i* ⟷ *i* = *to-nat-on A x*
⟨*proof*⟩

lemma *from-nat-into-surj*: *countable A* $\Longrightarrow$ *a* $\in$ *A* $\Longrightarrow$ $\exists$ *n. from-nat-into A n = a*
⟨*proof*⟩

lemma *from-nat-into-inject*[*simp*]:
A $\neq$ {} $\Longrightarrow$ *countable A* $\Longrightarrow$ *B* $\neq$ {} $\Longrightarrow$ *countable B* $\Longrightarrow$ *from-nat-into A = from-nat-into B* $\longleftrightarrow$ *A = B*
⟨*proof*⟩

lemma *inj-on-from-nat-into*: *inj-on from-nat-into* ({*A. A* $\neq$ {} $\wedge$ *countable A*})
⟨*proof*⟩

6.3 Closure properties of countability

lemma *countable-SIGMA*[*intro*, *simp*]:
countable I $\Longrightarrow$ ($\bigwedge$*i. i* $\in$ *I* $\Longrightarrow$ *countable* (*A i*)) $\Longrightarrow$ *countable* (*SIGMA i* : *I. A i*)
⟨*proof*⟩

lemma *countable-image*[*intro*, *simp*]:
assumes *countable A*
shows *countable* (*f*‘*A*)
⟨*proof*⟩

lemma *countable-image-inj-on*: *countable* (*f* ‘ *A*) $\Longrightarrow$ *inj-on f A* $\Longrightarrow$ *countable A*
⟨*proof*⟩

lemma *countable-UN*[*intro*, *simp*]:
fixes *I* :: *'i set* **and** *A* :: *'i* => *'a set*
assumes *I*: *countable I*
assumes *A*: $\bigwedge$*i. i* $\in$ *I* $\Longrightarrow$ *countable* (*A i*)
shows *countable* ($\bigcup$ *i*$\in$*I. A i*)
⟨*proof*⟩

lemma *countable-Un*[*intro*]: *countable A* $\Longrightarrow$ *countable B* $\Longrightarrow$ *countable* (*A* $\cup$ *B*)
⟨*proof*⟩

lemma *countable-Un-iff*[*simp*]: *countable* (*A* $\cup$ *B*) $\longleftrightarrow$ *countable A* $\wedge$ *countable B*
⟨*proof*⟩

lemma *countable-Plus*[*intro*, *simp*]:
countable A $\Longrightarrow$ *countable B* $\Longrightarrow$ *countable* (*A* <+> *B*)
⟨*proof*⟩

lemma *countable-empty*[*intro*, *simp*]: *countable* {}
⟨*proof*⟩

lemma *countable-insert*[*intro*, *simp*]: *countable A* $\Longrightarrow$ *countable* (*insert a A*)

⟨*proof*⟩

lemma *countable-Int1*[*intro*, *simp*]: *countable A* ⟹ *countable* (*A* ∩ *B*)
⟨*proof*⟩

lemma *countable-Int2*[*intro*, *simp*]: *countable B* ⟹ *countable* (*A* ∩ *B*)
⟨*proof*⟩

lemma *countable-INT*[*intro*, *simp*]: *i* ∈ *I* ⟹ *countable* (*A i*) ⟹ *countable* (⋂*i*∈*I*. *A i*)
⟨*proof*⟩

lemma *countable-Diff*[*intro*, *simp*]: *countable A* ⟹ *countable* (*A* − *B*)
⟨*proof*⟩

lemma *countable-insert-eq* [*simp*]: *countable* (*insert x A*) = *countable A*
⟨*proof*⟩

lemma *countable-vimage*: *B* ⊆ *range f* ⟹ *countable* (*f* −‘ *B*) ⟹ *countable B*
⟨*proof*⟩

lemma *surj-countable-vimage*: *surj f* ⟹ *countable* (*f* −‘ *B*) ⟹ *countable B*
⟨*proof*⟩

lemma *countable-Collect*[*simp*]: *countable A* ⟹ *countable* {*a* ∈ *A*. *φ a*}
⟨*proof*⟩

lemma *countable-Image*:
assumes ⋀*y*. *y* ∈ *Y* ⟹ *countable* (*X* ‘‘ {*y*})
assumes *countable Y*
shows *countable* (*X* ‘‘ *Y*)
⟨*proof*⟩

lemma *countable-relpow*:
fixes *X* :: *'a rel*
assumes *Image-X*: ⋀*Y*. *countable Y* ⟹ *countable* (*X* ‘‘ *Y*)
assumes *Y*: *countable Y*
shows *countable* ((*X* ^^ *i*) ‘‘ *Y*)
⟨*proof*⟩

lemma *countable-funpow*:
fixes *f* :: *'a set* ⇒ *'a set*
assumes ⋀*A*. *countable A* ⟹ *countable* (*f A*)
and *countable A*
shows *countable* ((*f* ^^ *n*) *A*)
⟨*proof*⟩

lemma *countable-rtrancl*:
(⋀*Y*. *countable Y* ⟹ *countable* (*X* ‘‘ *Y*)) ⟹ *countable Y* ⟹ *countable* (*X*^*

‘‘ *Y*)
⟨*proof*⟩

lemma *countable-lists*[*intro*, *simp*]:
assumes *A*: *countable A* **shows** *countable* (*lists A*)
⟨*proof*⟩

lemma *Collect-finite-eq-lists*: *Collect finite* = *set* ‘ *lists UNIV*
⟨*proof*⟩

lemma *countable-Collect-finite*: *countable* (*Collect* (*finite*::′*a*::*countable set*⇒*bool*))
⟨*proof*⟩

lemma *countable-rat*: *countable* ℚ
⟨*proof*⟩

lemma *Collect-finite-subset-eq-lists*: {*A*. *finite A* ∧ *A* ⊆ *T*} = *set* ‘ *lists T*
⟨*proof*⟩

lemma *countable-Collect-finite-subset*:
countable T ⟹ *countable* {*A*. *finite A* ∧ *A* ⊆ *T*}
⟨*proof*⟩

lemma *countable-set-option* [*simp*]: *countable* (*set-option x*)
⟨*proof*⟩

6.4 Misc lemmas

lemma *infinite-countable-subset′*:
assumes *X*: *infinite X* **shows** ∃ *C*⊆*X*. *countable C* ∧ *infinite C*
⟨*proof*⟩

lemma *countable-all*:
assumes *S*: *countable S*
shows (∀ *s*∈*S*. *P s*) ⟷ (∀ *n*::*nat*. *from-nat-into S n* ∈ *S* ⟶ *P* (*from-nat-into S n*))
⟨*proof*⟩

lemma *finite-sequence-to-countable-set*:
assumes *countable X* **obtains** *F* **where** ⋀*i*. *F i* ⊆ *X* ⋀*i*. *F i* ⊆ *F* (*Suc i*) ⋀*i*. *finite* (*F i*) (⋃ *i*. *F i*) = *X*
⟨*proof*⟩

lemma *transfer-countable*[*transfer-rule*]:
bi-unique R ⟹ *rel-fun* (*rel-set R*) *op* = *countable countable*
⟨*proof*⟩

6.5 Uncountable

abbreviation *uncountable* **where**

uncountable A ≡ ¬ *countable A*

lemma *uncountable-def*: *uncountable A* ⟷ *A* ≠ {} ∧ ¬ (∃*f*::(*nat* ⇒ 'a). *range f* = *A*)
⟨*proof*⟩

lemma *uncountable-bij-betw*: *bij-betw f A B* ⟹ *uncountable B* ⟹ *uncountable A*
⟨*proof*⟩

lemma *uncountable-infinite*: *uncountable A* ⟹ *infinite A*
⟨*proof*⟩

lemma *uncountable-minus-countable*:
uncountable A ⟹ *countable B* ⟹ *uncountable* (*A* − *B*)
⟨*proof*⟩

lemma *countable-Diff-eq* [*simp*]: *countable* (*A* − {*x*}) = *countable A*
⟨*proof*⟩

end

7 Pi and Function Sets

theory *FuncSet*
imports *Hilbert-Choice Main*
begin

definition *Pi* :: 'a *set* ⇒ ('a ⇒ 'b *set*) ⇒ ('a ⇒ 'b) *set*
where *Pi A B* = {*f*. ∀ *x*. *x* ∈ *A* ⟶ *f x* ∈ *B x*}

definition *extensional* :: 'a *set* ⇒ ('a ⇒ 'b) *set*
where *extensional A* = {*f*. ∀ *x*. *x* ∉ *A* ⟶ *f x* = *undefined*}

definition *restrict* :: ('a ⇒ 'b) ⇒ 'a *set* ⇒ 'a ⇒ 'b
where *restrict f A* = (λ*x*. *if x* ∈ *A then f x else undefined*)

abbreviation *funcset* :: 'a *set* ⇒ 'b *set* ⇒ ('a ⇒ 'b) *set* (**infixr** → *60*)
where *A* → *B* ≡ *Pi A* (λ-. *B*)

syntax (*ASCII*)
-Pi :: *pttrn* ⇒ 'a *set* ⇒ 'b *set* ⇒ ('a ⇒ 'b) *set* ((*3PI* -:-./ -) *10*)
-lam :: *pttrn* ⇒ 'a *set* ⇒ 'a ⇒ 'b ⇒ ('a ⇒ 'b) ((*3%*-:-./ -) [*0*,*0*,*3*] *3*)
syntax
-Pi :: *pttrn* ⇒ 'a *set* ⇒ 'b *set* ⇒ ('a ⇒ 'b) *set* ((*3*Π -∈-./ -) *10*)
-lam :: *pttrn* ⇒ 'a *set* ⇒ ('a ⇒ 'b) ⇒ ('a ⇒ 'b) ((*3*λ-∈-./ -) [*0*,*0*,*3*] *3*)
translations
Π *x*∈*A*. *B* ⇌ *CONST Pi A* (λ*x*. *B*)
λ*x*∈*A*. *f* ⇌ *CONST restrict* (λ*x*. *f*) *A*

definition *compose* :: $'a\ set \Rightarrow ('b \Rightarrow 'c) \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'c)$
where $compose\ A\ g\ f = (\lambda x{\in}A.\ g\ (f\ x))$

7.1 Basic Properties of *Pi*

lemma *Pi-I*[*intro*!]: $(\bigwedge x.\ x \in A \Longrightarrow f\ x \in B\ x) \Longrightarrow f \in Pi\ A\ B$
⟨*proof*⟩

lemma *Pi-I'*[*simp*]: $(\bigwedge x.\ x \in A \longrightarrow f\ x \in B\ x) \Longrightarrow f \in Pi\ A\ B$
⟨*proof*⟩

lemma *funcsetI*: $(\bigwedge x.\ x \in A \Longrightarrow f\ x \in B) \Longrightarrow f \in A \to B$
⟨*proof*⟩

lemma *Pi-mem*: $f \in Pi\ A\ B \Longrightarrow x \in A \Longrightarrow f\ x \in B\ x$
⟨*proof*⟩

lemma *Pi-iff*: $f \in Pi\ I\ X \longleftrightarrow (\forall\, i{\in}I.\ f\ i \in X\ i)$
⟨*proof*⟩

lemma *PiE* [*elim*]: $f \in Pi\ A\ B \Longrightarrow (f\ x \in B\ x \Longrightarrow Q) \Longrightarrow (x \notin A \Longrightarrow Q) \Longrightarrow Q$
⟨*proof*⟩

lemma *Pi-cong*: $(\bigwedge w.\ w \in A \Longrightarrow f\ w = g\ w) \Longrightarrow f \in Pi\ A\ B \longleftrightarrow g \in Pi\ A\ B$
⟨*proof*⟩

lemma *funcset-id* [*simp*]: $(\lambda x.\ x) \in A \to A$
⟨*proof*⟩

lemma *funcset-mem*: $f \in A \to B \Longrightarrow x \in A \Longrightarrow f\ x \in B$
⟨*proof*⟩

lemma *funcset-image*: $f \in A \to B \Longrightarrow f\ `\ A \subseteq B$
⟨*proof*⟩

lemma *image-subset-iff-funcset*: $F\ `\ A \subseteq B \longleftrightarrow F \in A \to B$
⟨*proof*⟩

lemma *Pi-eq-empty*[*simp*]: $(\Pi\ x \in A.\ B\ x) = \{\} \longleftrightarrow (\exists\, x{\in}A.\ B\ x = \{\})$
⟨*proof*⟩

lemma *Pi-empty* [*simp*]: $Pi\ \{\}\ B = UNIV$
⟨*proof*⟩

lemma *Pi-Int*: $Pi\ I\ E \cap Pi\ I\ F = (\Pi\ i{\in}I.\ E\ i \cap F\ i)$
⟨*proof*⟩

lemma *Pi-UN*:

fixes $A :: nat \Rightarrow 'i \Rightarrow 'a\ set$
assumes *finite I*
and *mono*: $\bigwedge i\ n\ m.\ i \in I \Longrightarrow n \leq m \Longrightarrow A\ n\ i \subseteq A\ m\ i$
shows $(\bigcup n.\ Pi\ I\ (A\ n)) = (\Pi\ i{\in}I.\ \bigcup n.\ A\ n\ i)$
⟨*proof*⟩

lemma *Pi-UNIV* [*simp*]: $A \to UNIV = UNIV$
⟨*proof*⟩

Covariance of Pi-sets in their second argument

lemma *Pi-mono*: $(\bigwedge x.\ x \in A \Longrightarrow B\ x \subseteq C\ x) \Longrightarrow Pi\ A\ B \subseteq Pi\ A\ C$
⟨*proof*⟩

Contravariance of Pi-sets in their first argument

lemma *Pi-anti-mono*: $A' \subseteq A \Longrightarrow Pi\ A\ B \subseteq Pi\ A'\ B$
⟨*proof*⟩

lemma *prod-final*:
assumes *1*: $fst \circ f \in Pi\ A\ B$
and *2*: $snd \circ f \in Pi\ A\ C$
shows $f \in (\Pi\ z \in A.\ B\ z \times C\ z)$
⟨*proof*⟩

lemma *Pi-split-domain*[*simp*]: $x \in Pi\ (I \cup J)\ X \longleftrightarrow x \in Pi\ I\ X \wedge x \in Pi\ J\ X$
⟨*proof*⟩

lemma *Pi-split-insert-domain*[*simp*]: $x \in Pi\ (insert\ i\ I)\ X \longleftrightarrow x \in Pi\ I\ X \wedge x\ i \in X\ i$
⟨*proof*⟩

lemma *Pi-cancel-fupd-range*[*simp*]: $i \notin I \Longrightarrow x \in Pi\ I\ (B(i := b)) \longleftrightarrow x \in Pi\ I\ B$
⟨*proof*⟩

lemma *Pi-cancel-fupd*[*simp*]: $i \notin I \Longrightarrow x(i := a) \in Pi\ I\ B \longleftrightarrow x \in Pi\ I\ B$
⟨*proof*⟩

lemma *Pi-fupd-iff*: $i \in I \Longrightarrow f \in Pi\ I\ (B(i := A)) \longleftrightarrow f \in Pi\ (I - \{i\})\ B \wedge f\ i \in A$
⟨*proof*⟩

7.2 Composition With a Restricted Domain: *compose*

lemma *funcset-compose*: $f \in A \to B \Longrightarrow g \in B \to C \Longrightarrow compose\ A\ g\ f \in A \to C$
⟨*proof*⟩

lemma *compose-assoc*:
assumes $f \in A \to B$

and $g \in B \to C$
and $h \in C \to D$
shows *compose A h (compose A g f) = compose A (compose B h g) f*
⟨*proof*⟩

lemma *compose-eq*: $x \in A \Longrightarrow$ *compose A g f x = g (f x)*
⟨*proof*⟩

lemma *surj-compose*: f ‘ $A = B \Longrightarrow g$ ‘ $B = C \Longrightarrow$ *compose A g f* ‘ $A = C$
⟨*proof*⟩

7.3 Bounded Abstraction: *restrict*

lemma *restrict-cong*: $I = J \Longrightarrow (\bigwedge i.\ i \in J$ =*simp*=> $f\ i = g\ i) \Longrightarrow$ *restrict f I = restrict g J*
⟨*proof*⟩

lemma *restrict-in-funcset*: $(\bigwedge x.\ x \in A \Longrightarrow f\ x \in B) \Longrightarrow (\lambda x{\in}A.\ f\ x) \in A \to B$
⟨*proof*⟩

lemma *restrictI*[*intro!*]: $(\bigwedge x.\ x \in A \Longrightarrow f\ x \in B\ x) \Longrightarrow (\lambda x{\in}A.\ f\ x) \in Pi\ A\ B$
⟨*proof*⟩

lemma *restrict-apply*[*simp*]: $(\lambda y{\in}A.\ f\ y)\ x =$ (*if* $x \in A$ *then f x else undefined*)
⟨*proof*⟩

lemma *restrict-apply'*: $x \in A \Longrightarrow (\lambda y{\in}A.\ f\ y)\ x = f\ x$
⟨*proof*⟩

lemma *restrict-ext*: $(\bigwedge x.\ x \in A \Longrightarrow f\ x = g\ x) \Longrightarrow (\lambda x{\in}A.\ f\ x) = (\lambda x{\in}A.\ g\ x)$
⟨*proof*⟩

lemma *restrict-UNIV*: *restrict f UNIV = f*
⟨*proof*⟩

lemma *inj-on-restrict-eq* [*simp*]: *inj-on (restrict f A) A = inj-on f A*
⟨*proof*⟩

lemma *Id-compose*: $f \in A \to B \Longrightarrow f \in$ *extensional A* $\Longrightarrow$ *compose A* $(\lambda y{\in}B.\ y)$ *f = f*
⟨*proof*⟩

lemma *compose-Id*: $g \in A \to B \Longrightarrow g \in$ *extensional A* $\Longrightarrow$ *compose A g* $(\lambda x{\in}A.\ x) = g$
⟨*proof*⟩

lemma *image-restrict-eq* [*simp*]: (*restrict f A*) ‘ $A =$ *f* ‘ *A*
⟨*proof*⟩

lemma *restrict-restrict*[*simp*]: *restrict* (*restrict f A*) *B* = *restrict f* (*A* ∩ *B*)
⟨*proof*⟩

lemma *restrict-fupd*[*simp*]: *i* ∉ *I* ⟹ *restrict* (*f* (*i* := *x*)) *I* = *restrict f I*
⟨*proof*⟩

lemma *restrict-upd*[*simp*]: *i* ∉ *I* ⟹ (*restrict f I*)(*i* := *y*) = *restrict* (*f*(*i* := *y*)) (*insert i I*)
⟨*proof*⟩

lemma *restrict-Pi-cancel*: *restrict x I* ∈ *Pi I A* ⟷ *x* ∈ *Pi I A*
⟨*proof*⟩

7.4 Bijections Between Sets

The definition of *bij-betw* is in *Fun.thy*, but most of the theorems belong here, or need at least *Hilbert-Choice*.

lemma *bij-betwI*:
assumes *f* ∈ *A* → *B*
and *g* ∈ *B* → *A*
and *g-f*: ⋀*x*. *x*∈*A* ⟹ *g* (*f x*) = *x*
and *f-g*: ⋀*y*. *y*∈*B* ⟹ *f* (*g y*) = *y*
shows *bij-betw f A B*
⟨*proof*⟩

lemma *bij-betw-imp-funcset*: *bij-betw f A B* ⟹ *f* ∈ *A* → *B*
⟨*proof*⟩

lemma *inj-on-compose*: *bij-betw f A B* ⟹ *inj-on g B* ⟹ *inj-on* (*compose A g f*) *A*
⟨*proof*⟩

lemma *bij-betw-compose*: *bij-betw f A B* ⟹ *bij-betw g B C* ⟹ *bij-betw* (*compose A g f*) *A C*
⟨*proof*⟩

lemma *bij-betw-restrict-eq* [*simp*]: *bij-betw* (*restrict f A*) *A B* = *bij-betw f A B*
⟨*proof*⟩

7.5 Extensionality

lemma *extensional-empty*[*simp*]: *extensional* {} = {λ*x*. *undefined*}
⟨*proof*⟩

lemma *extensional-arb*: *f* ∈ *extensional A* ⟹ *x* ∉ *A* ⟹ *f x* = *undefined*
⟨*proof*⟩

lemma *restrict-extensional* [*simp*]: *restrict f A* ∈ *extensional A*
⟨*proof*⟩

lemma *compose-extensional* [*simp*]: *compose A f g* ∈ *extensional A*
⟨*proof*⟩

lemma *extensionalityI*:
assumes *f* ∈ *extensional A*
and *g* ∈ *extensional A*
and ⋀*x*. *x* ∈ *A* ⟹ *f x* = *g x*
shows *f* = *g*
⟨*proof*⟩

lemma *extensional-restrict*: *f* ∈ *extensional A* ⟹ *restrict f A* = *f*
⟨*proof*⟩

lemma *extensional-subset*: *f* ∈ *extensional A* ⟹ *A* ⊆ *B* ⟹ *f* ∈ *extensional B*
⟨*proof*⟩

lemma *inv-into-funcset*: *f* ‘ *A* = *B* ⟹ (λ*x*∈*B*. *inv-into A f x*) ∈ *B* → *A*
⟨*proof*⟩

lemma *compose-inv-into-id*: *bij-betw f A B* ⟹ *compose A* (λ*y*∈*B*. *inv-into A f y*) *f* = (λ*x*∈*A*. *x*)
⟨*proof*⟩

lemma *compose-id-inv-into*: *f* ‘ *A* = *B* ⟹ *compose B f* (λ*y*∈*B*. *inv-into A f y*) = (λ*x*∈*B*. *x*)
⟨*proof*⟩

lemma *extensional-insert*[*intro*, *simp*]:
assumes *a* ∈ *extensional* (*insert i I*)
shows *a*(*i* := *b*) ∈ *extensional* (*insert i I*)
⟨*proof*⟩

lemma *extensional-Int*[*simp*]: *extensional I* ∩ *extensional I′* = *extensional* (*I* ∩ *I′*)
⟨*proof*⟩

lemma *extensional-UNIV*[*simp*]: *extensional UNIV* = *UNIV*
⟨*proof*⟩

lemma *restrict-extensional-sub*[*intro*]: *A* ⊆ *B* ⟹ *restrict f A* ∈ *extensional B*
⟨*proof*⟩

lemma *extensional-insert-undefined*[*intro*, *simp*]:
a ∈ *extensional* (*insert i I*) ⟹ *a*(*i* := *undefined*) ∈ *extensional I*
⟨*proof*⟩

lemma *extensional-insert-cancel*[*intro*, *simp*]:
a ∈ *extensional I* ⟹ *a* ∈ *extensional* (*insert i I*)

⟨*proof*⟩

7.6 Cardinality

lemma *card-inj*: $f \in A \to B \Longrightarrow$ *inj-on* $f\ A \Longrightarrow$ *finite* $B \Longrightarrow$ *card* $A \leq$ *card* B
⟨*proof*⟩

lemma *card-bij*:
assumes $f \in A \to B$ *inj-on* $f\ A$
and $g \in B \to A$ *inj-on* $g\ B$
and *finite* A *finite* B
shows *card* $A =$ *card* B
⟨*proof*⟩

7.7 Extensional Function Spaces

definition $PiE :: 'a\ set \Rightarrow ('a \Rightarrow 'b\ set) \Rightarrow ('a \Rightarrow 'b)\ set$
where $PiE\ S\ T = Pi\ S\ T \cap extensional\ S$

abbreviation $Pi_E\ A\ B \equiv PiE\ A\ B$

syntax (*ASCII*)
$-PiE :: pttrn \Rightarrow 'a\ set \Rightarrow 'b\ set \Rightarrow ('a \Rightarrow 'b)\ set$ (*3PIE -:-./ -) 10*)
syntax
$-PiE :: pttrn \Rightarrow 'a\ set \Rightarrow 'b\ set \Rightarrow ('a \Rightarrow 'b)\ set$ ($3\Pi_E$ -∈-./ -) *10*)
translations
$\Pi_E\ x{\in}A.\ B \rightleftharpoons CONST\ Pi_E\ A\ (\lambda x.\ B)$

abbreviation *extensional-funcset* $:: 'a\ set \Rightarrow 'b\ set \Rightarrow ('a \Rightarrow 'b)\ set$ (**infixr** $\to_E$ *60*)
where $A \to_E B \equiv (\Pi_E\ i{\in}A.\ B)$

lemma *extensional-funcset-def*: *extensional-funcset* $S\ T = (S \to T) \cap$ *extensional* S
⟨*proof*⟩

lemma *PiE-empty-domain*[*simp*]: $PiE\ \{\}\ T = \{\lambda x.\ undefined\}$
⟨*proof*⟩

lemma *PiE-UNIV-domain*: $PiE\ UNIV\ T = Pi\ UNIV\ T$
⟨*proof*⟩

lemma *PiE-empty-range*[*simp*]: $i \in I \Longrightarrow F\ i = \{\} \Longrightarrow (\Pi_E\ i{\in}I.\ F\ i) = \{\}$
⟨*proof*⟩

lemma *PiE-eq-empty-iff*: $Pi_E\ I\ F = \{\} \longleftrightarrow (\exists\, i{\in}I.\ F\ i = \{\})$
⟨*proof*⟩

lemma *PiE-arb*: $f \in PiE\ S\ T \Longrightarrow x \notin S \Longrightarrow f\ x = undefined$
⟨*proof*⟩

lemma *PiE-mem*: $f \in PiE\ S\ T \Longrightarrow x \in S \Longrightarrow f\ x \in T\ x$
⟨*proof*⟩

lemma *PiE-fun-upd*: $y \in T\ x \Longrightarrow f \in PiE\ S\ T \Longrightarrow f(x := y) \in PiE\ (insert\ x\ S)\ T$
⟨*proof*⟩

lemma *fun-upd-in-PiE*: $x \notin S \Longrightarrow f \in PiE\ (insert\ x\ S)\ T \Longrightarrow f(x := undefined) \in PiE\ S\ T$
⟨*proof*⟩

lemma *PiE-insert-eq*: $PiE\ (insert\ x\ S)\ T = (\lambda(y,\ g).\ g(x := y))\ `\ (T\ x \times PiE\ S\ T)$
⟨*proof*⟩

lemma *PiE-Int*: $Pi_E\ I\ A \cap Pi_E\ I\ B = Pi_E\ I\ (\lambda x.\ A\ x \cap B\ x)$
⟨*proof*⟩

lemma *PiE-cong*: $(\bigwedge i.\ i{\in}I \Longrightarrow A\ i = B\ i) \Longrightarrow Pi_E\ I\ A = Pi_E\ I\ B$
⟨*proof*⟩

lemma *PiE-E* [*elim*]:
assumes $f \in PiE\ A\ B$
obtains $x \in A$ **and** $f\ x \in B\ x$
| $x \notin A$ **and** $f\ x = undefined$
⟨*proof*⟩

lemma *PiE-I*[*intro!*]:
$(\bigwedge x.\ x \in A \Longrightarrow f\ x \in B\ x) \Longrightarrow (\bigwedge x.\ x \notin A \Longrightarrow f\ x = undefined) \Longrightarrow f \in PiE\ A\ B$
⟨*proof*⟩

lemma *PiE-mono*: $(\bigwedge x.\ x \in A \Longrightarrow B\ x \subseteq C\ x) \Longrightarrow PiE\ A\ B \subseteq PiE\ A\ C$
⟨*proof*⟩

lemma *PiE-iff*: $f \in PiE\ I\ X \longleftrightarrow (\forall\, i{\in}I.\ f\ i \in X\ i) \wedge f \in extensional\ I$
⟨*proof*⟩

lemma *PiE-restrict*[*simp*]: $f \in PiE\ A\ B \Longrightarrow restrict\ f\ A = f$
⟨*proof*⟩

lemma *restrict-PiE*[*simp*]: $restrict\ f\ I \in PiE\ I\ S \longleftrightarrow f \in Pi\ I\ S$
⟨*proof*⟩

lemma *PiE-eq-subset*:
assumes *ne*: $\bigwedge i.\ i \in I \Longrightarrow F\ i \neq \{\}\ \bigwedge i.\ i \in I \Longrightarrow F'\ i \neq \{\}$
and *eq*: $Pi_E\ I\ F = Pi_E\ I\ F'$
and $i \in I$

shows $F\ i \subseteq F'\ i$
⟨*proof*⟩

lemma *PiE-eq-iff-not-empty*:
assumes *ne*: $\bigwedge i.\ i \in I \Longrightarrow F\ i \neq \{\}$ $\bigwedge i.\ i \in I \Longrightarrow F'\ i \neq \{\}$
shows $Pi_E\ I\ F = Pi_E\ I\ F' \longleftrightarrow (\forall\, i{\in}I.\ F\ i = F'\ i)$
⟨*proof*⟩

lemma *PiE-eq-iff*:
$Pi_E\ I\ F = Pi_E\ I\ F' \longleftrightarrow (\forall\, i{\in}I.\ F\ i = F'\ i) \vee ((\exists\, i{\in}I.\ F\ i = \{\}) \wedge (\exists\, i{\in}I.\ F'\ i = \{\}))$
⟨*proof*⟩

lemma *extensional-funcset-fun-upd-restricts-rangeI*:
$\forall\, y \in S.\ f\ x \neq f\ y \Longrightarrow f \in (insert\ x\ S) \to_E T \Longrightarrow f(x := undefined) \in S \to_E (T - \{f\ x\})$
⟨*proof*⟩

lemma *extensional-funcset-fun-upd-extends-rangeI*:
assumes $a \in T$ $f \in S \to_E (T - \{a\})$
shows $f(x := a) \in insert\ x\ S \to_E T$
⟨*proof*⟩

7.7.1 Injective Extensional Function Spaces

lemma *extensional-funcset-fun-upd-inj-onI*:
assumes $f \in S \to_E (T - \{a\})$
and *inj-on f S*
shows *inj-on* $(f(x := a))\ S$
⟨*proof*⟩

lemma *extensional-funcset-extend-domain-inj-on-eq*:
assumes $x \notin S$
shows $\{f.\ f \in (insert\ x\ S) \to_E T \wedge inj\text{-}on\ f\ (insert\ x\ S)\} =$
$(\lambda(y,\ g).\ g(x{:=}y))$ ‘ $\{(y,\ g).\ y \in T \wedge g \in S \to_E (T - \{y\}) \wedge inj\text{-}on\ g\ S\}$
⟨*proof*⟩

lemma *extensional-funcset-extend-domain-inj-onI*:
assumes $x \notin S$
shows *inj-on* $(\lambda(y,\ g).\ g(x := y))$ $\{(y,\ g).\ y \in T \wedge g \in S \to_E (T - \{y\}) \wedge inj\text{-}on\ g\ S\}$
⟨*proof*⟩

7.7.2 Cardinality

lemma *finite-PiE*: *finite* $S \Longrightarrow (\bigwedge i.\ i \in S \Longrightarrow finite\ (T\ i)) \Longrightarrow finite\ (\Pi_E\ i \in S.\ T\ i)$
⟨*proof*⟩

lemma *inj-combinator*: $x \notin S \Longrightarrow inj\text{-}on\ (\lambda(y,\ g).\ g(x := y))\ (T\ x \times Pi_E\ S\ T)$

⟨*proof*⟩

lemma *card-PiE*: *finite* $S \Longrightarrow$ *card* $(\Pi_E\ i \in S.\ T\ i) = (\prod\ i \in S.\ card\ (T\ i))$
⟨*proof*⟩

end

8 Square root of sum of squares

theory *L2-Norm*
imports *NthRoot*
begin

definition
$setL2\ f\ A = sqrt\ (\sum i \in A.\ (f\ i)^2)$

lemma *setL2-cong*:
$[\![A = B;\ \bigwedge x.\ x \in B \Longrightarrow f\ x = g\ x]\!] \Longrightarrow setL2\ f\ A = setL2\ g\ B$
⟨*proof*⟩

lemma *strong-setL2-cong*:
$[\![A = B;\ \bigwedge x.\ x \in B =simp=> f\ x = g\ x]\!] \Longrightarrow setL2\ f\ A = setL2\ g\ B$
⟨*proof*⟩

lemma *setL2-infinite* [*simp*]: $\neg$ *finite* $A \Longrightarrow setL2\ f\ A = 0$
⟨*proof*⟩

lemma *setL2-empty* [*simp*]: $setL2\ f\ \{\} = 0$
⟨*proof*⟩

lemma *setL2-insert* [*simp*]:
[*finite* F; $a \notin F$] $\Longrightarrow$
$setL2\ f\ (insert\ a\ F) = sqrt\ ((f\ a)^2 + (setL2\ f\ F)^2)$
⟨*proof*⟩

lemma *setL2-nonneg* [*simp*]: $0 \leq setL2\ f\ A$
⟨*proof*⟩

lemma *setL2-0'*: $\forall a \in A.\ f\ a = 0 \Longrightarrow setL2\ f\ A = 0$
⟨*proof*⟩

lemma *setL2-constant*: $setL2\ (\lambda x.\ y)\ A = sqrt\ (of\text{-}nat\ (card\ A)) * |y|$
⟨*proof*⟩

lemma *setL2-mono*:
assumes $\bigwedge i.\ i \in K \Longrightarrow f\ i \leq g\ i$
assumes $\bigwedge i.\ i \in K \Longrightarrow 0 \leq f\ i$
shows $setL2\ f\ K \leq setL2\ g\ K$
⟨*proof*⟩

lemma *setL2-strict-mono*:
assumes *finite K* **and** $K \neq \{\}$
assumes $\bigwedge i.\ i \in K \Longrightarrow f\ i < g\ i$
assumes $\bigwedge i.\ i \in K \Longrightarrow 0 \leq f\ i$
shows *setL2 f K* $<$ *setL2 g K*
⟨*proof*⟩

lemma *setL2-right-distrib*:
$0 \leq r \Longrightarrow r * setL2\ f\ A = setL2\ (\lambda x.\ r * f\ x)\ A$
⟨*proof*⟩

lemma *setL2-left-distrib*:
$0 \leq r \Longrightarrow setL2\ f\ A * r = setL2\ (\lambda x.\ f\ x * r)\ A$
⟨*proof*⟩

lemma *setL2-eq-0-iff*: *finite A* $\Longrightarrow$ *setL2 f A* $= 0 \longleftrightarrow (\forall x{\in}A.\ f\ x = 0)$
⟨*proof*⟩

lemma *setL2-triangle-ineq*:
shows *setL2* $(\lambda i.\ f\ i + g\ i)\ A \leq setL2\ f\ A + setL2\ g\ A$
⟨*proof*⟩

lemma *sqrt-sum-squares-le-sum*:
$[\![0 \leq x;\ 0 \leq y]\!] \Longrightarrow sqrt\ (x^2 + y^2) \leq x + y$
⟨*proof*⟩

lemma *setL2-le-setsum* [*rule-format*]:
$(\forall i{\in}A.\ 0 \leq f\ i) \longrightarrow setL2\ f\ A \leq setsum\ f\ A$
⟨*proof*⟩

lemma *sqrt-sum-squares-le-sum-abs*: $sqrt\ (x^2 + y^2) \leq |x| + |y|$
⟨*proof*⟩

lemma *setL2-le-setsum-abs*: *setL2 f A* $\leq (\sum i{\in}A.\ |f\ i|)$
⟨*proof*⟩

lemma *setL2-mult-ineq-lemma*:
fixes *a b c d* :: *real*
shows $2 * (a * c) * (b * d) \leq a^2 * d^2 + b^2 * c^2$
⟨*proof*⟩

lemma *setL2-mult-ineq*: $(\sum i{\in}A.\ |f\ i| * |g\ i|) \leq setL2\ f\ A * setL2\ g\ A$
⟨*proof*⟩

lemma *member-le-setL2*: $[\![$*finite A*; $i \in A]\!] \Longrightarrow f\ i \leq setL2\ f\ A$
⟨*proof*⟩

end

9 Inner Product Spaces and the Gradient Derivative

theory *Inner-Product*
imports ~~/*src*/*HOL*/*Complex-Main*
begin

9.1 Real inner product spaces

Temporarily relax type constraints for *open*, *uniformity*, *dist*, and *norm*.

⟨*ML*⟩

class *real-inner* = *real-vector* + *sgn-div-norm* + *dist-norm* + *uniformity-dist* + *open-uniformity* +
  fixes *inner* :: *'a* ⇒ *'a* ⇒ *real*
  assumes *inner-commute*: *inner x y* = *inner y x*
  and *inner-add-left*: *inner* (*x* + *y*) *z* = *inner x z* + *inner y z*
  and *inner-scaleR-left* [*simp*]: *inner* (*scaleR r x*) *y* = *r* ∗ (*inner x y*)
  and *inner-ge-zero* [*simp*]: *0* ≤ *inner x x*
  and *inner-eq-zero-iff* [*simp*]: *inner x x* = *0* ⟷ *x* = *0*
  and *norm-eq-sqrt-inner*: *norm x* = *sqrt* (*inner x x*)
begin

lemma *inner-zero-left* [*simp*]: *inner 0 x* = *0*
  ⟨*proof*⟩

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

lemma *inner-diff-left*: *inner* (*x* − *y*) *z* = *inner x z* − *inner y z*
  ⟨*proof*⟩

lemma *inner-setsum-left*: *inner* (∑ *x*∈*A*. *f x*) *y* = (∑ *x*∈*A*. *inner* (*f x*) *y*)
  ⟨*proof*⟩

Transfer distributivity rules to right argument.

lemma *inner-add-right*: *inner x* (*y* + *z*) = *inner x y* + *inner x z*
  ⟨*proof*⟩

lemma *inner-scaleR-right* [*simp*]: *inner x* (*scaleR r y*) = *r* ∗ (*inner x y*)
  ⟨*proof*⟩

lemma *inner-zero-right* [*simp*]: *inner x 0* = *0*
  ⟨*proof*⟩

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

lemma *inner-diff-right*: *inner x (y − z) = inner x y − inner x z*
⟨*proof*⟩

lemma *inner-setsum-right*: *inner x (∑ y∈A. f y) = (∑ y∈A. inner x (f y))*
⟨*proof*⟩

lemmas *inner-add* [*algebra-simps*] = *inner-add-left inner-add-right*
lemmas *inner-diff* [*algebra-simps*] = *inner-diff-left inner-diff-right*
lemmas *inner-scaleR* = *inner-scaleR-left inner-scaleR-right*

Legacy theorem names

lemmas *inner-left-distrib* = *inner-add-left*
lemmas *inner-right-distrib* = *inner-add-right*
lemmas *inner-distrib* = *inner-left-distrib inner-right-distrib*

lemma *inner-gt-zero-iff* [*simp*]: *0 < inner x x ⟷ x ≠ 0*
⟨*proof*⟩

lemma *power2-norm-eq-inner*: *(norm x)^2 = inner x x*
⟨*proof*⟩

Identities involving real multiplication and division.

lemma *inner-mult-left*: *inner (of-real m * a) b = m * (inner a b)*
⟨*proof*⟩

lemma *inner-mult-right*: *inner a (of-real m * b) = m * (inner a b)*
⟨*proof*⟩

lemma *inner-mult-left'*: *inner (a * of-real m) b = m * (inner a b)*
⟨*proof*⟩

lemma *inner-mult-right'*: *inner a (b * of-real m) = (inner a b) * m*
⟨*proof*⟩

lemma *Cauchy-Schwarz-ineq*:
*(inner x y)^2 ≤ inner x x * inner y y*
⟨*proof*⟩

lemma *Cauchy-Schwarz-ineq2*:
*|inner x y| ≤ norm x * norm y*
⟨*proof*⟩

lemma *norm-cauchy-schwarz*: *inner x y ≤ norm x * norm y*
⟨*proof*⟩

subclass *real-normed-vector*
⟨*proof*⟩

end

lemma *inner-divide-left*:
fixes $a :: 'a :: \{real\text{-}inner, real\text{-}div\text{-}algebra\}$
shows $inner\ (a\ /\ of\text{-}real\ m)\ b = (inner\ a\ b)\ /\ m$
⟨*proof*⟩

lemma *inner-divide-right*:
fixes $a :: 'a :: \{real\text{-}inner, real\text{-}div\text{-}algebra\}$
shows $inner\ a\ (b\ /\ of\text{-}real\ m) = (inner\ a\ b)\ /\ m$
⟨*proof*⟩

Re-enable constraints for *open*, *uniformity*, *dist*, and *norm*.

⟨*ML*⟩

lemma *bounded-bilinear-inner*:
bounded-bilinear $(inner :: 'a :: real\text{-}inner \Rightarrow 'a \Rightarrow real)$
⟨*proof*⟩

lemmas *tendsto-inner* [*tendsto-intros*] =
bounded-bilinear.tendsto [*OF bounded-bilinear-inner*]

lemmas *isCont-inner* [*simp*] =
bounded-bilinear.isCont [*OF bounded-bilinear-inner*]

lemmas *has-derivative-inner* [*derivative-intros*] =
bounded-bilinear.FDERIV [*OF bounded-bilinear-inner*]

lemmas *bounded-linear-inner-left* =
bounded-bilinear.bounded-linear-left [*OF bounded-bilinear-inner*]

lemmas *bounded-linear-inner-right* =
bounded-bilinear.bounded-linear-right [*OF bounded-bilinear-inner*]

lemmas *bounded-linear-inner-left-comp* = *bounded-linear-inner-left*[*THEN bounded-linear-compose*]

lemmas *bounded-linear-inner-right-comp* = *bounded-linear-inner-right*[*THEN bounded-linear-compose*]

lemmas *has-derivative-inner-right* [*derivative-intros*] =
bounded-linear.has-derivative [*OF bounded-linear-inner-right*]

lemmas *has-derivative-inner-left* [*derivative-intros*] =
bounded-linear.has-derivative [*OF bounded-linear-inner-left*]

lemma *differentiable-inner* [*simp*]:
f differentiable (*at x within s*) $\Longrightarrow$ *g differentiable at x within s* $\Longrightarrow$ ($\lambda x.$ *inner* (*f x*) (*g x*)) *differentiable at x within s*
⟨*proof*⟩

9.2 Class instances

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

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

instance
⟨*proof*⟩

end

instantiation *complex* :: *real-inner*
begin

definition *inner-complex-def*:
inner x y = *Re x* ∗ *Re y* + *Im x* ∗ *Im y*

instance
⟨*proof*⟩

end

lemma *complex-inner-1* [*simp*]: *inner 1 x* = *Re x*
⟨*proof*⟩

lemma *complex-inner-1-right* [*simp*]: *inner x 1* = *Re x*
⟨*proof*⟩

lemma *complex-inner-ii-left* [*simp*]: *inner ii x* = *Im x*
⟨*proof*⟩

lemma *complex-inner-ii-right* [*simp*]: *inner x ii* = *Im x*
⟨*proof*⟩

9.3 Gradient derivative

definition
gderiv ::
['*a*::*real-inner* ⇒ *real*, '*a*, '*a*] ⇒ *bool*
((*GDERIV* (-)/ (-)/ :> (-)) [*1000*, *1000*, *60*] *60*)
where
GDERIV f x :> *D* ⟷ *FDERIV f x* :> (λ*h*. *inner h D*)

lemma *gderiv-deriv* [*simp*]: *GDERIV f x* :> *D* ⟷ *DERIV f x* :> *D*
⟨*proof*⟩

lemma *GDERIV-DERIV-compose*:
⟦*GDERIV f x* :> *df*; *DERIV g* (*f x*) :> *dg*⟧
⟹ *GDERIV* (λ*x*. *g* (*f x*)) *x* :> *scaleR dg df*

⟨*proof*⟩

lemma *has-derivative-subst*: ⟦*FDERIV f x :> df*; *df = d*⟧ ⟹ *FDERIV f x :> d*
⟨*proof*⟩

lemma *GDERIV-subst*: ⟦*GDERIV f x :> df*; *df = d*⟧ ⟹ *GDERIV f x :> d*
⟨*proof*⟩

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

lemma *GDERIV-add*:
⟦*GDERIV f x :> df*; *GDERIV g x :> dg*⟧
⟹ *GDERIV* (λ*x*. *f x + g x*) *x :> df + dg*
⟨*proof*⟩

lemma *GDERIV-minus*:
GDERIV f x :> df ⟹ *GDERIV* (λ*x*. *− f x*) *x :> − df*
⟨*proof*⟩

lemma *GDERIV-diff*:
⟦*GDERIV f x :> df*; *GDERIV g x :> dg*⟧
⟹ *GDERIV* (λ*x*. *f x − g x*) *x :> df − dg*
⟨*proof*⟩

lemma *GDERIV-scaleR*:
⟦*DERIV f x :> df*; *GDERIV g x :> dg*⟧
⟹ *GDERIV* (λ*x*. *scaleR* (*f x*) (*g x*)) *x*
:> (*scaleR* (*f x*) *dg* + *scaleR df* (*g x*))
⟨*proof*⟩

lemma *GDERIV-mult*:
⟦*GDERIV f x :> df*; *GDERIV g x :> dg*⟧
⟹ *GDERIV* (λ*x*. *f x ∗ g x*) *x :> scaleR* (*f x*) *dg* + *scaleR* (*g x*) *df*
⟨*proof*⟩

lemma *GDERIV-inverse*:
⟦*GDERIV f x :> df*; *f x ≠ 0*⟧
⟹ *GDERIV* (λ*x*. *inverse* (*f x*)) *x :> −* (*inverse* (*f x*))^2 *∗$_R$ df*
⟨*proof*⟩

lemma *GDERIV-norm*:
assumes *x ≠ 0* **shows** *GDERIV* (λ*x*. *norm x*) *x :> sgn x*
⟨*proof*⟩

lemmas *has-derivative-norm = GDERIV-norm* [*unfolded gderiv-def*]

end

10 Additive group operations on product types

theory *Product-plus*
imports *Main*
begin

10.1 Operations

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

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

instance ⟨*proof*⟩
end

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

definition *plus-prod-def*:
$x + y = (fst\ x + fst\ y, snd\ x + snd\ y)$

instance ⟨*proof*⟩
end

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

definition *minus-prod-def*:
$x - y = (fst\ x - fst\ y, snd\ x - snd\ y)$

instance ⟨*proof*⟩
end

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

definition *uminus-prod-def*:
$- x = (- fst\ x, - snd\ x)$

instance ⟨*proof*⟩
end

lemma *fst-zero* [*simp*]: $fst\ 0 = 0$
⟨*proof*⟩

lemma *snd-zero* [*simp*]: $snd\ 0 = 0$
⟨*proof*⟩

lemma *fst-add* [*simp*]: $fst\ (x + y) = fst\ x + fst\ y$

⟨*proof*⟩

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

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

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

lemma *fst-uminus* [*simp*]: *fst* (− *x*) = − *fst x*
⟨*proof*⟩

lemma *snd-uminus* [*simp*]: *snd* (− *x*) = − *snd x*
⟨*proof*⟩

lemma *add-Pair* [*simp*]: (*a*, *b*) + (*c*, *d*) = (*a* + *c*, *b* + *d*)
⟨*proof*⟩

lemma *diff-Pair* [*simp*]: (*a*, *b*) − (*c*, *d*) = (*a* − *c*, *b* − *d*)
⟨*proof*⟩

lemma *uminus-Pair* [*simp*, *code*]: − (*a*, *b*) = (− *a*, − *b*)
⟨*proof*⟩

10.2 Class instances

instance *prod* :: (*semigroup-add*, *semigroup-add*) *semigroup-add*
⟨*proof*⟩

instance *prod* :: (*ab-semigroup-add*, *ab-semigroup-add*) *ab-semigroup-add*
⟨*proof*⟩

instance *prod* :: (*monoid-add*, *monoid-add*) *monoid-add*
⟨*proof*⟩

instance *prod* :: (*comm-monoid-add*, *comm-monoid-add*) *comm-monoid-add*
⟨*proof*⟩

instance *prod* :: (*cancel-semigroup-add*, *cancel-semigroup-add*) *cancel-semigroup-add*
⟨*proof*⟩

instance *prod* :: (*cancel-ab-semigroup-add*, *cancel-ab-semigroup-add*) *cancel-ab-semigroup-add*
⟨*proof*⟩

instance *prod* :: (*cancel-comm-monoid-add*, *cancel-comm-monoid-add*) *cancel-comm-monoid-add*
⟨*proof*⟩

instance *prod* :: (*group-add*, *group-add*) *group-add*
⟨*proof*⟩

instance *prod* :: (*ab-group-add*, *ab-group-add*) *ab-group-add*
⟨*proof*⟩

lemma *fst-setsum*: *fst* $(\sum x{\in}A.\ f\ x) = (\sum x{\in}A.\ fst\ (f\ x))$
⟨*proof*⟩

lemma *snd-setsum*: *snd* $(\sum x{\in}A.\ f\ x) = (\sum x{\in}A.\ snd\ (f\ x))$
⟨*proof*⟩

lemma *setsum-prod*: $(\sum x{\in}A.\ (f\ x,\ g\ x)) = (\sum x{\in}A.\ f\ x, \sum x{\in}A.\ g\ x)$
⟨*proof*⟩

end

11 Cartesian Products as Vector Spaces

theory *Product-Vector*
imports *Inner-Product Product-plus*
begin

11.1 Product is a real vector space

instantiation *prod* :: (*real-vector*, *real-vector*) *real-vector*
begin

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

lemma *fst-scaleR* [*simp*]: *fst* (*scaleR r A*) = *scaleR r* (*fst A*)
⟨*proof*⟩

lemma *snd-scaleR* [*simp*]: *snd* (*scaleR r A*) = *scaleR r* (*snd A*)
⟨*proof*⟩

lemma *scaleR-Pair* [*simp*]: *scaleR r* (*a*, *b*) = (*scaleR r a*, *scaleR r b*)
⟨*proof*⟩

instance
⟨*proof*⟩

end

11.2 Product is a metric space

instantiation *prod* :: (*metric-space*, *metric-space*) *dist*
begin

definition *dist-prod-def*[*code del*]:
dist x y = sqrt ((dist (fst x) (fst y))² + (dist (snd x) (snd y))²)

instance ⟨*proof*⟩
end

instantiation *prod* :: (*metric-space*, *metric-space*) *uniformity-dist*
begin

definition [*code del*]:
(*uniformity* :: (('*a* × '*b*) × ('*a* × '*b*)) *filter*) =
(*INF e*:{*0* <..}. *principal* {(*x*, *y*). *dist x y* < *e*})

instance
⟨*proof*⟩
end

declare *uniformity-Abort*[**where** '*a*='*a* :: *metric-space* × '*b* :: *metric-space*, *code*]

instantiation *prod* :: (*metric-space*, *metric-space*) *metric-space*
begin

lemma *dist-Pair-Pair*: *dist* (*a*, *b*) (*c*, *d*) = *sqrt* ((*dist a c*)² + (*dist b d*)²)
⟨*proof*⟩

lemma *dist-fst-le*: *dist* (*fst x*) (*fst y*) ≤ *dist x y*
⟨*proof*⟩

lemma *dist-snd-le*: *dist* (*snd x*) (*snd y*) ≤ *dist x y*
⟨*proof*⟩

instance
⟨*proof*⟩

end

declare [[*code abort*: *dist*::('*a*::*metric-space*∗'*b*::*metric-space*)⇒('*a*∗'*b*) ⇒ *real*]]

lemma *Cauchy-fst*: *Cauchy X* ⟹ *Cauchy* (λ*n*. *fst* (*X n*))
⟨*proof*⟩

lemma *Cauchy-snd*: *Cauchy X* ⟹ *Cauchy* (λ*n*. *snd* (*X n*))
⟨*proof*⟩

lemma *Cauchy-Pair*:
assumes *Cauchy X* **and** *Cauchy Y*
shows *Cauchy* (λ*n*. (*X n*, *Y n*))
⟨*proof*⟩

11.3 Product is a complete metric space

instance *prod* :: (*complete-space*, *complete-space*) *complete-space*
⟨*proof*⟩

11.4 Product is a normed vector space

instantiation *prod* :: (*real-normed-vector*, *real-normed-vector*) *real-normed-vector*
begin

definition *norm-prod-def*[*code del*]:
norm x = *sqrt* ((*norm* (*fst x*))^2 + (*norm* (*snd x*))^2)

definition *sgn-prod-def*:
sgn (*x*::'*a* × '*b*) = *scaleR* (*inverse* (*norm x*)) *x*

lemma *norm-Pair*: *norm* (*a*, *b*) = *sqrt* ((*norm a*)^2 + (*norm b*)^2)
⟨*proof*⟩

instance
⟨*proof*⟩

end

declare [[*code abort*: *norm*::('*a*::*real-normed-vector* * '*b*::*real-normed-vector*) ⇒ *real*]]

instance *prod* :: (*banach*, *banach*) *banach* ⟨*proof*⟩

11.4.1 Pair operations are linear

lemma *bounded-linear-fst*: *bounded-linear fst*
⟨*proof*⟩

lemma *bounded-linear-snd*: *bounded-linear snd*
⟨*proof*⟩

lemmas *bounded-linear-fst-comp* = *bounded-linear-fst*[*THEN bounded-linear-compose*]

lemmas *bounded-linear-snd-comp* = *bounded-linear-snd*[*THEN bounded-linear-compose*]

lemma *bounded-linear-Pair*:
assumes *f*: *bounded-linear f*
assumes *g*: *bounded-linear g*
shows *bounded-linear* (λ*x*. (*f x*, *g x*))
⟨*proof*⟩

11.4.2 Frechet derivatives involving pairs

lemma *has-derivative-Pair* [*derivative-intros*]:

assumes *f*: (*f has-derivative f'*) (*at x within s*) **and** *g*: (*g has-derivative g'*) (*at x within s*)
shows ((λ*x*. (*f x*, *g x*)) *has-derivative* (λ*h*. (*f' h*, *g' h*))) (*at x within s*)
⟨*proof*⟩

lemmas *has-derivative-fst* [*derivative-intros*] = *bounded-linear.has-derivative* [*OF bounded-linear-fst*]
lemmas *has-derivative-snd* [*derivative-intros*] = *bounded-linear.has-derivative* [*OF bounded-linear-snd*]

lemma *has-derivative-split* [*derivative-intros*]:
(((λ*p*. *f* (*fst p*) (*snd p*)) *has-derivative f'*) *F* ⟹ ((λ(*a*, *b*). *f a b*) *has-derivative f'*) *F*
⟨*proof*⟩

11.5 Product is an inner product space

instantiation *prod* :: (*real-inner*, *real-inner*) *real-inner*
begin

definition *inner-prod-def*:
inner x y = *inner* (*fst x*) (*fst y*) + *inner* (*snd x*) (*snd y*)

lemma *inner-Pair* [*simp*]: *inner* (*a*, *b*) (*c*, *d*) = *inner a c* + *inner b d*
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *inner-Pair-0*: *inner x* (*0*, *b*) = *inner* (*snd x*) *b inner x* (*a*, *0*) = *inner* (*fst x*) *a*
⟨*proof*⟩

lemma
fixes *x* :: *'a*::*real-normed-vector*
shows *norm-Pair1* [*simp*]: *norm* (*0*,*x*) = *norm x*
and *norm-Pair2* [*simp*]: *norm* (*x*,*0*) = *norm x*
⟨*proof*⟩

lemma *norm-commute*: *norm* (*x*,*y*) = *norm* (*y*,*x*)
⟨*proof*⟩

lemma *norm-fst-le*: *norm x* ≤ *norm* (*x*,*y*)
⟨*proof*⟩

lemma *norm-snd-le*: *norm y* ≤ *norm* (*x*,*y*)
⟨*proof*⟩

end

12 Finite-Dimensional Inner Product Spaces

theory *Euclidean-Space*
imports
L2-Norm
~~/src/HOL/Library/Inner-Product
~~/src/HOL/Library/Product-Vector
begin

12.1 Type class of Euclidean spaces

class *euclidean-space* = *real-inner* +
fixes *Basis* :: *'a set*
assumes *nonempty-Basis* [*simp*]: *Basis* ≠ {}
assumes *finite-Basis* [*simp*]: *finite Basis*
assumes *inner-Basis*:
⟦*u* ∈ *Basis*; *v* ∈ *Basis*⟧ ⟹ *inner u v* = (*if u* = *v then 1 else 0*)
assumes *euclidean-all-zero-iff*:
(∀ *u*∈*Basis*. *inner x u* = *0*) ⟷ (*x* = *0*)

abbreviation *dimension* :: (*'a*::*euclidean-space*) *itself* ⇒ *nat* **where**
dimension TYPE(*'a*) ≡ *card* (*Basis* :: *'a set*)

syntax *-type-dimension* :: *type* => *nat* ((*1DIM*/(*1'*(*-'*))))

translations *DIM*('*t*) == *CONST dimension* (*TYPE*('*t*))

lemma (**in** *euclidean-space*) *norm-Basis*[*simp*]: *u* ∈ *Basis* ⟹ *norm u* = *1*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *inner-same-Basis*[*simp*]: *u* ∈ *Basis* ⟹ *inner u u* = *1*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *inner-not-same-Basis*: *u* ∈ *Basis* ⟹ *v* ∈ *Basis* ⟹ *u* ≠ *v* ⟹ *inner u v* = *0*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *sgn-Basis*: *u* ∈ *Basis* ⟹ *sgn u* = *u*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *Basis-zero* [*simp*]: *0* ∉ *Basis*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *nonzero-Basis*: *u* ∈ *Basis* ⟹ *u* ≠ *0*
⟨*proof*⟩

lemma (**in** *euclidean-space*) *SOME-Basis*: $(SOME\ i.\ i \in Basis) \in Basis$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *inner-setsum-left-Basis*[*simp*]:
$b \in Basis \Longrightarrow inner\ (\sum i{\in}Basis.\ f\ i *_R i)\ b = f\ b$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-eqI*:
assumes *b*: $\bigwedge b.\ b \in Basis \Longrightarrow inner\ x\ b = inner\ y\ b$ **shows** $x = y$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-eq-iff*:
$x = y \longleftrightarrow (\forall b{\in}Basis.\ inner\ x\ b = inner\ y\ b)$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-representation-setsum*:
$(\sum i{\in}Basis.\ f\ i *_R i) = b \longleftrightarrow (\forall i{\in}Basis.\ f\ i = inner\ b\ i)$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-representation-setsum'*:
$b = (\sum i{\in}Basis.\ f\ i *_R i) \longleftrightarrow (\forall i{\in}Basis.\ f\ i = inner\ b\ i)$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-representation*: $(\sum b{\in}Basis.\ inner\ x\ b *_R b) = x$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *choice-Basis-iff*:
fixes $P :: 'a \Rightarrow real \Rightarrow bool$
shows $(\forall i{\in}Basis.\ \exists x.\ P\ i\ x) \longleftrightarrow (\exists x.\ \forall i{\in}Basis.\ P\ i\ (inner\ x\ i))$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-representation-setsum-fun*:
$(\lambda x.\ \sum b{\in}Basis.\ inner\ (f\ x)\ b *_R b) = f$
⟨*proof*⟩

lemma *euclidean-isCont*:
assumes $\bigwedge b.\ b \in Basis \Longrightarrow isCont\ (\lambda x.\ (inner\ (f\ x)\ b) *_R b)\ x$
shows *isCont f x*
⟨*proof*⟩

lemma *DIM-positive*: $0 < DIM('a{::}euclidean\text{-}space)$
⟨*proof*⟩

12.2 Subclass relationships

instance *euclidean-space* ⊆ *perfect-space*
⟨*proof*⟩

12.3 Class instances

12.3.1 Type *real*

instantiation *real* :: *euclidean-space*
begin

definition
[*simp*]: *Basis* = {*1*::*real*}

instance
⟨*proof*⟩

end

lemma *DIM-real*[*simp*]: *DIM*(*real*) = *1*
⟨*proof*⟩

12.3.2 Type *complex*

instantiation *complex* :: *euclidean-space*
begin

definition *Basis-complex-def*:
Basis = {*1*, *ii*}

instance
⟨*proof*⟩

end

lemma *DIM-complex*[*simp*]: *DIM*(*complex*) = *2*
⟨*proof*⟩

12.3.3 Type *'a* × *'b*

instantiation *prod* :: (*euclidean-space*, *euclidean-space*) *euclidean-space*
begin

definition
Basis = (λ*u*. (*u*, *0*)) ' *Basis* ∪ (λ*v*. (*0*, *v*)) ' *Basis*

lemma *setsum-Basis-prod-eq*:
fixes *f*::('*a*∗'*b*)⇒('*a*∗'*b*)
shows *setsum f Basis* = *setsum* (λ*i*. *f* (*i*, *0*)) *Basis* + *setsum* (λ*i*. *f* (*0*, *i*)) *Basis*
⟨*proof*⟩

instance ⟨*proof*⟩

lemma *DIM-prod*[*simp*]: $DIM('a \times 'b) = DIM('a) + DIM('b)$
⟨*proof*⟩

end

end

13 Elementary linear algebra on Euclidean spaces

theory *Linear-Algebra*
imports
Euclidean-Space
~~/*src*/*HOL*/*Library*/*Infinite-Set*
begin

lemma *cond-application-beta*: (*if b then f else g*) *x* = (*if b then f x else g x*)
⟨*proof*⟩

notation *inner* (**infix** · *70*)

lemma *square-bound-lemma*:
fixes *x* :: *real*
shows *x* < (*1* + *x*) ∗ (*1* + *x*)
⟨*proof*⟩

lemma *square-continuous*:
fixes *e* :: *real*
shows *e* > *0* ⟹ ∃ *d*. *0* < *d* ∧ (∀ *y*. |*y* − *x*| < *d* ⟶ |*y* ∗ *y* − *x* ∗ *x*| < *e*)
⟨*proof*⟩

Hence derive more interesting properties of the norm.

lemma *norm-eq-0-dot*: *norm x* = *0* ⟷ *x* · *x* = (*0*::*real*)
⟨*proof*⟩

lemma *norm-triangle-sub*:
fixes *x y* :: *'a*::*real-normed-vector*
shows *norm x* ≤ *norm y* + *norm* (*x* − *y*)
⟨*proof*⟩

lemma *norm-le*: *norm x* ≤ *norm y* ⟷ *x* · *x* ≤ *y* · *y*
⟨*proof*⟩

lemma *norm-lt*: *norm x* < *norm y* ⟷ *x* · *x* < *y* · *y*
⟨*proof*⟩

lemma *norm-eq*: *norm x* = *norm y* ⟷ *x* · *x* = *y* · *y*
⟨*proof*⟩

lemma *norm-eq-1*: *norm x* = *1* ⟷ *x* · *x* = *1*

⟨*proof*⟩

Squaring equations and inequalities involving norms.

lemma *dot-square-norm*: $x \cdot x = (norm\ x)^2$
⟨*proof*⟩

lemma *norm-eq-square*: $norm\ x = a \longleftrightarrow 0 \leq a \wedge x \cdot x = a^2$
⟨*proof*⟩

lemma *norm-le-square*: $norm\ x \leq a \longleftrightarrow 0 \leq a \wedge x \cdot x \leq a^2$
⟨*proof*⟩

lemma *norm-ge-square*: $norm\ x \geq a \longleftrightarrow a \leq 0 \vee x \cdot x \geq a^2$
⟨*proof*⟩

lemma *norm-lt-square*: $norm\ x < a \longleftrightarrow 0 < a \wedge x \cdot x < a^2$
⟨*proof*⟩

lemma *norm-gt-square*: $norm\ x > a \longleftrightarrow a < 0 \vee x \cdot x > a^2$
⟨*proof*⟩

Dot product in terms of the norm rather than conversely.

lemmas *inner-simps* = *inner-add-left inner-add-right inner-diff-right inner-diff-left inner-scaleR-left inner-scaleR-right*

lemma *dot-norm*: $x \cdot y = ((norm\ (x + y))^2 - (norm\ x)^2 - (norm\ y)^2)\ /\ 2$
⟨*proof*⟩

lemma *dot-norm-neg*: $x \cdot y = (((norm\ x)^2 + (norm\ y)^2) - (norm\ (x - y))^2)\ /\ 2$
⟨*proof*⟩

Equality of vectors in terms of $op \cdot$ products.

lemma *vector-eq*: $x = y \longleftrightarrow x \cdot x = x \cdot y \wedge y \cdot y = x \cdot x$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *norm-triangle-half-r*:
$norm\ (y - x1) < e\ /\ 2 \Longrightarrow norm\ (y - x2) < e\ /\ 2 \Longrightarrow norm\ (x1 - x2) < e$
⟨*proof*⟩

lemma *norm-triangle-half-l*:
assumes $norm\ (x - y) < e\ /\ 2$
and $norm\ (x' - y) < e\ /\ 2$
shows $norm\ (x - x') < e$
⟨*proof*⟩

lemma *norm-triangle-le*: $norm\ x + norm\ y \leq e \Longrightarrow norm\ (x + y) \leq e$
⟨*proof*⟩

lemma *norm-triangle-lt*: *norm x + norm y < e ⟹ norm (x + y) < e*
⟨*proof*⟩

lemma *setsum-clauses*:
shows *setsum f {} = 0*
and *finite S ⟹ setsum f (insert x S) = (if x ∈ S then setsum f S else f x + setsum f S)*
⟨*proof*⟩

lemma *setsum-norm-le*:
fixes *f* :: *'a ⇒ 'b*::*real-normed-vector*
assumes *fg*: *∀ x ∈ S. norm (f x) ≤ g x*
shows *norm (setsum f S) ≤ setsum g S*
⟨*proof*⟩

lemma *setsum-norm-bound*:
fixes *f* :: *'a ⇒ 'b*::*real-normed-vector*
assumes *K*: *∀ x ∈ S. norm (f x) ≤ K*
shows *norm (setsum f S) ≤ of-nat (card S) * K*
⟨*proof*⟩

lemma *setsum-group*:
assumes *fS*: *finite S* **and** *fT*: *finite T* **and** *fST*: *f ' S ⊆ T*
shows *setsum (λy. setsum g {x. x ∈ S ∧ f x = y}) T = setsum g S*
⟨*proof*⟩

lemma *vector-eq-ldot*: *(∀ x. x · y = x · z) ⟷ y = z*
⟨*proof*⟩

lemma *vector-eq-rdot*: *(∀ z. x · z = y · z) ⟷ x = y*
⟨*proof*⟩

13.1 Orthogonality.

context *real-inner*
begin

definition *orthogonal x y ⟷ x · y = 0*

lemma *orthogonal-clauses*:
orthogonal a 0
*orthogonal a x ⟹ orthogonal a (c *_R_ x)*
orthogonal a x ⟹ orthogonal a (− x)
orthogonal a x ⟹ orthogonal a y ⟹ orthogonal a (x + y)
orthogonal a x ⟹ orthogonal a y ⟹ orthogonal a (x − y)
orthogonal 0 a
*orthogonal x a ⟹ orthogonal (c *_R_ x) a*
orthogonal x a ⟹ orthogonal (− x) a
orthogonal x a ⟹ orthogonal y a ⟹ orthogonal (x + y) a

orthogonal x a $\Longrightarrow$ *orthogonal y a* $\Longrightarrow$ *orthogonal* $(x - y)$ *a*
⟨*proof*⟩

end

lemma *orthogonal-commute*: *orthogonal x y* $\longleftrightarrow$ *orthogonal y x*
⟨*proof*⟩

13.2 Linear functions.

lemma *linear-iff*:
linear f $\longleftrightarrow$ $(\forall\, x\ y.\ f\ (x + y) = f\ x + f\ y) \wedge (\forall\, c\ x.\ f\ (c *_R x) = c *_R f\ x)$
(**is** *linear f* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *linear-compose-cmul*: *linear f* $\Longrightarrow$ *linear* $(\lambda x.\ c *_R f\ x)$
⟨*proof*⟩

lemma *linear-compose-neg*: *linear f* $\Longrightarrow$ *linear* $(\lambda x.\ -\ f\ x)$
⟨*proof*⟩

lemma *linear-compose-add*: *linear f* $\Longrightarrow$ *linear g* $\Longrightarrow$ *linear* $(\lambda x.\ f\ x + g\ x)$
⟨*proof*⟩

lemma *linear-compose-sub*: *linear f* $\Longrightarrow$ *linear g* $\Longrightarrow$ *linear* $(\lambda x.\ f\ x - g\ x)$
⟨*proof*⟩

lemma *linear-compose*: *linear f* $\Longrightarrow$ *linear g* $\Longrightarrow$ *linear* $(g \circ f)$
⟨*proof*⟩

lemma *linear-id*: *linear id*
⟨*proof*⟩

lemma *linear-zero*: *linear* $(\lambda x.\ 0)$
⟨*proof*⟩

lemma *linear-compose-setsum*:
assumes *lS*: $\forall\, a \in S.$ *linear* $(f\ a)$
shows *linear* $(\lambda x.$ *setsum* $(\lambda a.\ f\ a\ x)\ S)$
⟨*proof*⟩

lemma *linear-0*: *linear f* $\Longrightarrow$ $f\ 0 = 0$
⟨*proof*⟩

lemma *linear-cmul*: *linear f* $\Longrightarrow$ $f\ (c *_R x) = c *_R f\ x$
⟨*proof*⟩

lemma *linear-neg*: *linear f* $\Longrightarrow$ $f\ (-\ x) = -\ f\ x$
⟨*proof*⟩

lemma *linear-add*: *linear f* $\Longrightarrow$ *f* (*x* + *y*) = *f x* + *f y*
⟨*proof*⟩

lemma *linear-sub*: *linear f* $\Longrightarrow$ *f* (*x* − *y*) = *f x* − *f y*
⟨*proof*⟩

lemma *linear-setsum*:
assumes *f*: *linear f*
shows *f* (*setsum g S*) = *setsum* (*f* ∘ *g*) *S*
⟨*proof*⟩

lemma *linear-setsum-mul*:
assumes *lin*: *linear f*
shows *f* (*setsum* (λ*i*. *c i* $*_R$ *v i*) *S*) = *setsum* (λ*i*. *c i* $*_R$ *f* (*v i*)) *S*
⟨*proof*⟩

lemma *linear-injective-0*:
assumes *lin*: *linear f*
shows *inj f* ⟷ (∀ *x*. *f x* = *0* ⟶ *x* = *0*)
⟨*proof*⟩

lemma *linear-scaleR* [*simp*]: *linear* (λ*x*. *scaleR c x*)
⟨*proof*⟩

lemma *linear-scaleR-left* [*simp*]: *linear* (λ*r*. *scaleR r x*)
⟨*proof*⟩

lemma *injective-scaleR*: *c* ≠ *0* $\Longrightarrow$ *inj* (λ*x*::'*a*::*real-vector*. *scaleR c x*)
⟨*proof*⟩

lemma *linear-add-cmul*:
assumes *linear f*
shows *f* (*a* $*_R$ *x* + *b* $*_R$ *y*) = *a* $*_R$ *f x* + *b* $*_R$ *f y*
⟨*proof*⟩

lemma *linear-componentwise*:
fixes *f*:: '*a*::*euclidean-space* ⇒ '*b*::*real-inner*
assumes *lf*: *linear f*
shows (*f x*) · *j* = (∑ *i*∈*Basis*. (*x*·*i*) * (*f i*·*j*)) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

13.3 Bilinear functions.

definition *bilinear f* ⟷ (∀ *x*. *linear* (λ*y*. *f x y*)) ∧ (∀ *y*. *linear* (λ*x*. *f x y*))

lemma *bilinear-ladd*: *bilinear h* $\Longrightarrow$ *h* (*x* + *y*) *z* = *h x z* + *h y z*
⟨*proof*⟩

lemma *bilinear-radd*: *bilinear h* $\Longrightarrow$ *h x* (*y* + *z*) = *h x y* + *h x z*
⟨*proof*⟩

lemma *bilinear-lmul*: *bilinear h* $\Longrightarrow$ *h* (*c* $*_R$ *x*) *y* = *c* $*_R$ *h x y*
⟨*proof*⟩

lemma *bilinear-rmul*: *bilinear h* $\Longrightarrow$ *h x* (*c* $*_R$ *y*) = *c* $*_R$ *h x y*
⟨*proof*⟩

lemma *bilinear-lneg*: *bilinear h* $\Longrightarrow$ *h* (− *x*) *y* = − *h x y*
⟨*proof*⟩

lemma *bilinear-rneg*: *bilinear h* $\Longrightarrow$ *h x* (− *y*) = − *h x y*
⟨*proof*⟩

lemma (**in** *ab-group-add*) *eq-add-iff*: *x* = *x* + *y* $\longleftrightarrow$ *y* = *0*
⟨*proof*⟩

lemma *bilinear-lzero*:
assumes *bilinear h*
shows *h 0 x* = *0*
⟨*proof*⟩

lemma *bilinear-rzero*:
assumes *bilinear h*
shows *h x 0* = *0*
⟨*proof*⟩

lemma *bilinear-lsub*: *bilinear h* $\Longrightarrow$ *h* (*x* − *y*) *z* = *h x z* − *h y z*
⟨*proof*⟩

lemma *bilinear-rsub*: *bilinear h* $\Longrightarrow$ *h z* (*x* − *y*) = *h z x* − *h z y*
⟨*proof*⟩

lemma *bilinear-setsum*:
assumes *bh*: *bilinear h*
and *fS*: *finite S*
and *fT*: *finite T*
shows *h* (*setsum f S*) (*setsum g T*) = *setsum* (λ(*i*,*j*). *h* (*f i*) (*g j*)) (*S* × *T*)
⟨*proof*⟩

13.4 Adjoints.

definition *adjoint f* = (*SOME f′*. ∀ *x y*. *f x* · *y* = *x* · *f′ y*)

lemma *adjoint-unique*:
assumes ∀ *x y*. *inner* (*f x*) *y* = *inner x* (*g y*)
shows *adjoint f* = *g*
⟨*proof*⟩

TODO: The following lemmas about adjoints should hold for any Hilbert space (i.e. complete inner product space). (see http://en.wikipedia.org/wiki/Hermitian_adjoint)

lemma *adjoint-works*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *lf*: *linear f*
shows $x \cdot adjoint\ f\ y = f\ x \cdot y$
⟨*proof*⟩

lemma *adjoint-clauses*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *lf*: *linear f*
shows $x \cdot adjoint\ f\ y = f\ x \cdot y$
and $adjoint\ f\ y \cdot x = y \cdot f\ x$
⟨*proof*⟩

lemma *adjoint-linear*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *lf*: *linear f*
shows *linear* (*adjoint f*)
⟨*proof*⟩

lemma *adjoint-adjoint*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *lf*: *linear f*
shows *adjoint* (*adjoint f*) = *f*
⟨*proof*⟩

13.5 Interlude: Some properties of real sets

lemma *seq-mono-lemma*:
assumes $\forall (n::nat) \geq m.\ (d\ n :: real) < e\ n$
and $\forall n \geq m.\ e\ n \leq e\ m$
shows $\forall n \geq m.\ d\ n < e\ m$
⟨*proof*⟩

lemma *infinite-enumerate*:
assumes *fS*: *infinite S*
shows $\exists r.\ subseq\ r \wedge (\forall n.\ r\ n \in S)$
⟨*proof*⟩

lemma *approachable-lt-le*: $(\exists (d::real) > 0.\ \forall x.\ f\ x < d \longrightarrow P\ x) \longleftrightarrow (\exists d{>}0.\ \forall x.\ f\ x \leq d \longrightarrow P\ x)$
⟨*proof*⟩

lemma *approachable-lt-le2*: — like the above, but pushes aside an extra formula
$(\exists (d::real) > 0.\ \forall x.\ Q\ x \longrightarrow f\ x < d \longrightarrow P\ x) \longleftrightarrow (\exists d{>}0.\ \forall x.\ f\ x \leq d \longrightarrow Q\ x \longrightarrow P\ x)$
⟨*proof*⟩

lemma *triangle-lemma*:
fixes $x\ y\ z :: real$
assumes x: $0 \leq x$
and y: $0 \leq y$
and z: $0 \leq z$
and xy: $x^2 \leq y^2 + z^2$
shows $x \leq y + z$
⟨*proof*⟩

13.6 A generic notion of "hull" (convex, affine, conic hull and closure).

definition *hull* :: $('a\ set \Rightarrow bool) \Rightarrow 'a\ set \Rightarrow 'a\ set$ (**infixl** *hull* 75)
where $S\ hull\ s = \bigcap \{t.\ S\ t \wedge s \subseteq t\}$

lemma *hull-same*: $S\ s \Longrightarrow S\ hull\ s = s$
⟨*proof*⟩

lemma *hull-in*: $(\bigwedge T.\ Ball\ T\ S \Longrightarrow S\ (\bigcap T)) \Longrightarrow S\ (S\ hull\ s)$
⟨*proof*⟩

lemma *hull-eq*: $(\bigwedge T.\ Ball\ T\ S \Longrightarrow S\ (\bigcap T)) \Longrightarrow (S\ hull\ s) = s \longleftrightarrow S\ s$
⟨*proof*⟩

lemma *hull-hull*: $S\ hull\ (S\ hull\ s) = S\ hull\ s$
⟨*proof*⟩

lemma *hull-subset*[*intro*]: $s \subseteq (S\ hull\ s)$
⟨*proof*⟩

lemma *hull-mono*: $s \subseteq t \Longrightarrow (S\ hull\ s) \subseteq (S\ hull\ t)$
⟨*proof*⟩

lemma *hull-antimono*: $\forall x.\ S\ x \longrightarrow T\ x \Longrightarrow (T\ hull\ s) \subseteq (S\ hull\ s)$
⟨*proof*⟩

lemma *hull-minimal*: $s \subseteq t \Longrightarrow S\ t \Longrightarrow (S\ hull\ s) \subseteq t$
⟨*proof*⟩

lemma *subset-hull*: $S\ t \Longrightarrow S\ hull\ s \subseteq t \longleftrightarrow s \subseteq t$
⟨*proof*⟩

lemma *hull-UNIV*: $S\ hull\ UNIV = UNIV$
⟨*proof*⟩

lemma *hull-unique*: $s \subseteq t \Longrightarrow S\ t \Longrightarrow (\bigwedge t'.\ s \subseteq t' \Longrightarrow S\ t' \Longrightarrow t \subseteq t') \Longrightarrow (S\ hull\ s = t)$
⟨*proof*⟩

lemma *hull-induct*: $(\bigwedge x.\ x \in S \Longrightarrow P\ x) \Longrightarrow Q\ \{x.\ P\ x\} \Longrightarrow \forall x \in Q\ hull\ S.\ P\ x$
⟨*proof*⟩

lemma *hull-inc*: $x \in S \Longrightarrow x \in P\ hull\ S$
⟨*proof*⟩

lemma *hull-union-subset*: $(S\ hull\ s) \cup (S\ hull\ t) \subseteq (S\ hull\ (s \cup t))$
⟨*proof*⟩

lemma *hull-union*:
assumes T: $\bigwedge T.\ Ball\ T\ S \Longrightarrow S\ (\bigcap T)$
shows $S\ hull\ (s \cup t) = S\ hull\ (S\ hull\ s \cup S\ hull\ t)$
⟨*proof*⟩

lemma *hull-redundant-eq*: $a \in (S\ hull\ s) \longleftrightarrow S\ hull\ (insert\ a\ s) = S\ hull\ s$
⟨*proof*⟩

lemma *hull-redundant*: $a \in (S\ hull\ s) \Longrightarrow S\ hull\ (insert\ a\ s) = S\ hull\ s$
⟨*proof*⟩

13.7 Archimedean properties and useful consequences

Bernoulli's inequality

proposition *Bernoulli-inequality*:
fixes x :: *real*
assumes $-1 \leq x$
shows $1 + n * x \leq (1 + x)\ \hat{}\ n$
⟨*proof*⟩

corollary *Bernoulli-inequality-even*:
fixes x :: *real*
assumes *even n*
shows $1 + n * x \leq (1 + x)\ \hat{}\ n$
⟨*proof*⟩

corollary *real-arch-pow*:
fixes x :: *real*
assumes x: $1 < x$
shows $\exists n.\ y < x \hat{} n$
⟨*proof*⟩

corollary *real-arch-pow-inv*:
fixes $x\ y$ :: *real*
assumes y: $y > 0$
and *x1*: $x < 1$
shows $\exists n.\ x \hat{} n < y$
⟨*proof*⟩

lemma *forall-pos-mono*:
$(\bigwedge d\ e{::}real.\ d < e \Longrightarrow P\ d \Longrightarrow P\ e) \Longrightarrow$
$(\bigwedge n{::}nat.\ n \neq 0 \Longrightarrow P\ (inverse\ (real\ n))) \Longrightarrow (\bigwedge e.\ 0 < e \Longrightarrow P\ e)$
⟨*proof*⟩

lemma *forall-pos-mono-1*:
$(\bigwedge d\ e{::}real.\ d < e \Longrightarrow P\ d \Longrightarrow P\ e) \Longrightarrow$
$(\bigwedge n.\ P\ (inverse\ (real\ (Suc\ n)))) \Longrightarrow 0 < e \Longrightarrow P\ e$
⟨*proof*⟩

13.8 A bit of linear algebra.

definition (**in** *real-vector*) *subspace* :: *'a set* ⇒ *bool*
where *subspace S* ⟷ $0 \in S \wedge (\forall x \in S.\ \forall y \in S.\ x + y \in S) \wedge (\forall c.\ \forall x \in S.\ c *_R x \in S)$

definition (**in** *real-vector*) *span S* = (*subspace hull S*)
definition (**in** *real-vector*) *dependent S* ⟷ $(\exists a \in S.\ a \in span\ (S - \{a\}))$
abbreviation (**in** *real-vector*) *independent s* ≡ ¬ *dependent s*

Closure properties of subspaces.

lemma *subspace-UNIV*[*simp*]: *subspace UNIV*
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-0*: *subspace S* ⟹ $0 \in S$
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-add*: *subspace S* ⟹ $x \in S \Longrightarrow y \in S \Longrightarrow x + y \in S$
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-mul*: *subspace S* ⟹ $x \in S \Longrightarrow c *_R x \in S$
⟨*proof*⟩

lemma *subspace-neg*: *subspace S* ⟹ $x \in S \Longrightarrow - x \in S$
⟨*proof*⟩

lemma *subspace-sub*: *subspace S* ⟹ $x \in S \Longrightarrow y \in S \Longrightarrow x - y \in S$
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-setsum*:
assumes *sA*: *subspace A*
and *f*: $\forall x{\in}B.\ f\ x \in A$
shows *setsum f B* ∈ *A*
⟨*proof*⟩

lemma *subspace-linear-image*:
assumes *lf*: *linear f*
and *sS*: *subspace S*

shows *subspace* $(f\ `\ S)$
⟨*proof*⟩

lemma *subspace-linear-vimage*: *linear* $f \Longrightarrow$ *subspace* $S \Longrightarrow$ *subspace* $(f\ -`\ S)$
⟨*proof*⟩

lemma *subspace-linear-preimage*: *linear* $f \Longrightarrow$ *subspace* $S \Longrightarrow$ *subspace* $\{x.\ f\ x \in S\}$
⟨*proof*⟩

lemma *subspace-trivial*: *subspace* $\{0\}$
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-inter*: *subspace* $A \Longrightarrow$ *subspace* $B \Longrightarrow$ *subspace* $(A \cap B)$
⟨*proof*⟩

lemma *subspace-Times*: *subspace* $A \Longrightarrow$ *subspace* $B \Longrightarrow$ *subspace* $(A \times B)$
⟨*proof*⟩

Properties of span.

lemma (**in** *real-vector*) *span-mono*: $A \subseteq B \Longrightarrow$ *span* $A \subseteq$ *span* B
⟨*proof*⟩

lemma (**in** *real-vector*) *subspace-span*: *subspace* (*span* S)
⟨*proof*⟩

lemma (**in** *real-vector*) *span-clauses*:
$a \in S \Longrightarrow a \in$ *span* S
$0 \in$ *span* S
$x \in$ *span* $S \Longrightarrow y \in$ *span* $S \Longrightarrow x + y \in$ *span* S
$x \in$ *span* $S \Longrightarrow c *_R x \in$ *span* S
⟨*proof*⟩

lemma *span-unique*:
$S \subseteq T \Longrightarrow$ *subspace* $T \Longrightarrow (\bigwedge T'.\ S \subseteq T' \Longrightarrow$ *subspace* $T' \Longrightarrow T \subseteq T') \Longrightarrow$ *span* $S = T$
⟨*proof*⟩

lemma *span-minimal*: $S \subseteq T \Longrightarrow$ *subspace* $T \Longrightarrow$ *span* $S \subseteq T$
⟨*proof*⟩

lemma (**in** *real-vector*) *span-induct*:
assumes x: $x \in$ *span* S
and P: *subspace* P
and SP: $\bigwedge x.\ x \in S \Longrightarrow x \in P$
shows $x \in P$
⟨*proof*⟩

lemma *span-empty*[*simp*]: *span* {} = {*0*}
⟨*proof*⟩

lemma (**in** *real-vector*) *independent-empty* [*iff*]: *independent* {}
⟨*proof*⟩

lemma *dependent-single*[*simp*]: *dependent* {*x*} ⟷ *x* = *0*
⟨*proof*⟩

lemma (**in** *real-vector*) *independent-mono*: *independent A* ⟹ *B* ⊆ *A* ⟹ *independent B*
⟨*proof*⟩

lemma (**in** *real-vector*) *span-subspace*: *A* ⊆ *B* ⟹ *B* ≤ *span A* ⟹ *subspace B* ⟹ *span A* = *B*
⟨*proof*⟩

lemma (**in** *real-vector*) *span-induct'*:
assumes *SP*: ∀ *x* ∈ *S*. *P x*
and *P*: *subspace* {*x*. *P x*}
shows ∀ *x* ∈ *span S*. *P x*
⟨*proof*⟩

inductive-set (**in** *real-vector*) *span-induct-alt-help* **for** *S* :: *'a set*
where
span-induct-alt-help-0: *0* ∈ *span-induct-alt-help S*
| *span-induct-alt-help-S*:
x ∈ *S* ⟹ *z* ∈ *span-induct-alt-help S* ⟹
(*c* *$_R$* *x* + *z*) ∈ *span-induct-alt-help S*

lemma *span-induct-alt'*:
assumes *h0*: *h 0*
and *hS*: ⋀*c x y*. *x* ∈ *S* ⟹ *h y* ⟹ *h* (*c* *$_R$* *x* + *y*)
shows ∀ *x* ∈ *span S*. *h x*
⟨*proof*⟩

lemma *span-induct-alt*:
assumes *h0*: *h 0*
and *hS*: ⋀*c x y*. *x* ∈ *S* ⟹ *h y* ⟹ *h* (*c* *$_R$* *x* + *y*)
and *x*: *x* ∈ *span S*
shows *h x*
⟨*proof*⟩

Individual closure properties.

lemma *span-span*: *span* (*span A*) = *span A*
⟨*proof*⟩

lemma (**in** *real-vector*) *span-superset*: *x* ∈ *S* ⟹ *x* ∈ *span S*
⟨*proof*⟩

lemma (**in** *real-vector*) *span-0*: $0 \in span\ S$
⟨*proof*⟩

lemma *span-inc*: $S \subseteq span\ S$
⟨*proof*⟩

lemma (**in** *real-vector*) *dependent-0*:
assumes $0 \in A$
shows *dependent A*
⟨*proof*⟩

lemma (**in** *real-vector*) *span-add*: $x \in span\ S \Longrightarrow y \in span\ S \Longrightarrow x + y \in span\ S$
⟨*proof*⟩

lemma (**in** *real-vector*) *span-mul*: $x \in span\ S \Longrightarrow c *_R x \in span\ S$
⟨*proof*⟩

lemma *span-neg*: $x \in span\ S \Longrightarrow - x \in span\ S$
⟨*proof*⟩

lemma *span-sub*: $x \in span\ S \Longrightarrow y \in span\ S \Longrightarrow x - y \in span\ S$
⟨*proof*⟩

lemma (**in** *real-vector*) *span-setsum*: $\forall x{\in}A.\ f\ x \in span\ S \Longrightarrow setsum\ f\ A \in span\ S$
⟨*proof*⟩

lemma *span-add-eq*: $x \in span\ S \Longrightarrow x + y \in span\ S \longleftrightarrow y \in span\ S$
⟨*proof*⟩

Mapping under linear image.

lemma *span-linear-image*:
assumes *lf*: *linear f*
shows $span\ (f\ `\ S) = f\ `\ span\ S$
⟨*proof*⟩

lemma *span-union*: $span\ (A \cup B) = (\lambda(a,\ b).\ a + b)\ `\ (span\ A \times span\ B)$
⟨*proof*⟩

The key breakdown property.

lemma *span-singleton*: $span\ \{x\} = range\ (\lambda k.\ k *_R x)$
⟨*proof*⟩

lemma *span-insert*: $span\ (insert\ a\ S) = \{x.\ \exists k.\ (x - k *_R a) \in span\ S\}$
⟨*proof*⟩

lemma *span-breakdown*:

assumes *bS*: $b \in S$
and *aS*: $a \in span\ S$
shows $\exists k.\ a - k *_R b \in span\ (S - \{b\})$
⟨*proof*⟩

lemma *span-breakdown-eq*: $x \in span\ (insert\ a\ S) \longleftrightarrow (\exists k.\ x - k *_R a \in span\ S)$
⟨*proof*⟩

Hence some "reversal" results.

lemma *in-span-insert*:
assumes *a*: $a \in span\ (insert\ b\ S)$
and *na*: $a \notin span\ S$
shows $b \in span\ (insert\ a\ S)$
⟨*proof*⟩

lemma *in-span-delete*:
assumes *a*: $a \in span\ S$
and *na*: $a \notin span\ (S - \{b\})$
shows $b \in span\ (insert\ a\ (S - \{b\}))$
⟨*proof*⟩

Transitivity property.

lemma *span-redundant*: $x \in span\ S \Longrightarrow span\ (insert\ x\ S) = span\ S$
⟨*proof*⟩

lemma *span-trans*:
assumes *x*: $x \in span\ S$
and *y*: $y \in span\ (insert\ x\ S)$
shows $y \in span\ S$
⟨*proof*⟩

lemma *span-insert-0*[*simp*]: $span\ (insert\ 0\ S) = span\ S$
⟨*proof*⟩

An explicit expansion is sometimes needed.

lemma *span-explicit*:
$span\ P = \{y.\ \exists S\ u.\ finite\ S \wedge S \subseteq P \wedge setsum\ (\lambda v.\ u\ v *_R v)\ S = y\}$
(**is** - = *?E* **is** - = $\{y.\ ?h\ y\}$ **is** - = $\{y.\ \exists S\ u.\ ?Q\ S\ u\ y\}$)
⟨*proof*⟩

lemma *dependent-explicit*:
$dependent\ P \longleftrightarrow (\exists S\ u.\ finite\ S \wedge S \subseteq P \wedge (\exists v{\in}S.\ u\ v \neq 0 \wedge setsum\ (\lambda v.\ u\ v *_R v)\ S = 0))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *span-finite*:
assumes *fS*: *finite S*

shows *span S* = {*y*. ∃ *u*. *setsum* (λ*v*. *u v* ∗$_R$ *v*) *S* = *y*}
(**is** - = *?rhs*)
⟨*proof*⟩

This is useful for building a basis step-by-step.

lemma *independent-insert*:
independent (*insert a S*) ⟷
(*if a* ∈ *S then independent S else independent S* ∧ *a* ∉ *span S*)
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

The degenerate case of the Exchange Lemma.

lemma *spanning-subset-independent*:
assumes *BA*: *B* ⊆ *A*
and *iA*: *independent A*
and *AsB*: *A* ⊆ *span B*
shows *A* = *B*
⟨*proof*⟩

The general case of the Exchange Lemma, the key to what follows.

lemma *exchange-lemma*:
assumes *f*:*finite t*
and *i*: *independent s*
and *sp*: *s* ⊆ *span t*
shows ∃ *t′*. *card t′* = *card t* ∧ *finite t′* ∧ *s* ⊆ *t′* ∧ *t′* ⊆ *s* ∪ *t* ∧ *s* ⊆ *span t′*
⟨*proof*⟩

This implies corresponding size bounds.

lemma *independent-span-bound*:
assumes *f*: *finite t*
and *i*: *independent s*
and *sp*: *s* ⊆ *span t*
shows *finite s* ∧ *card s* ≤ *card t*
⟨*proof*⟩

lemma *finite-Atleast-Atmost-nat*[*simp*]: *finite* {*f x* |*x*. *x*∈ (*UNIV*::*′a*::*finite set*)}
⟨*proof*⟩

13.9 Euclidean Spaces as Typeclass

lemma *independent-Basis*: *independent Basis*
⟨*proof*⟩

lemma *span-Basis* [*simp*]: *span Basis* = *UNIV*
⟨*proof*⟩

lemma *in-span-Basis*: *x* ∈ *span Basis*
⟨*proof*⟩

lemma *Basis-le-norm*: $b \in Basis \Longrightarrow |x \cdot b| \leq norm\ x$
⟨*proof*⟩

lemma *norm-bound-Basis-le*: $b \in Basis \Longrightarrow norm\ x \leq e \Longrightarrow |x \cdot b| \leq e$
⟨*proof*⟩

lemma *norm-bound-Basis-lt*: $b \in Basis \Longrightarrow norm\ x < e \Longrightarrow |x \cdot b| < e$
⟨*proof*⟩

lemma *norm-le-l1*: $norm\ x \leq (\sum b{\in}Basis.\ |x \cdot b|)$
⟨*proof*⟩

lemma *setsum-norm-allsubsets-bound*:
fixes $f :: 'a \Rightarrow 'n{::}euclidean\text{-}space$
assumes *fP*: *finite P*
and *fPs*: $\bigwedge Q.\ Q \subseteq P \Longrightarrow norm\ (setsum\ f\ Q) \leq e$
shows $(\sum x{\in}P.\ norm\ (f\ x)) \leq 2 * real\ DIM('n) * e$
⟨*proof*⟩

13.10 Linearity and Bilinearity continued

lemma *linear-bounded*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes *lf*: *linear f*
shows $\exists B.\ \forall x.\ norm\ (f\ x) \leq B * norm\ x$
⟨*proof*⟩

lemma *linear-conv-bounded-linear*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
shows *linear f* $\longleftrightarrow$ *bounded-linear f*
⟨*proof*⟩

lemmas *linear-linear* = *linear-conv-bounded-linear*[*symmetric*]

lemma *linear-bounded-pos*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes *lf*: *linear f*
shows $\exists B > 0.\ \forall x.\ norm\ (f\ x) \leq B * norm\ x$
⟨*proof*⟩

lemma *bounded-linearI'*:
fixes $f ::'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes $\bigwedge x\ y.\ f\ (x + y) = f\ x + f\ y$
and $\bigwedge c\ x.\ f\ (c *_R x) = c *_R f\ x$
shows *bounded-linear f*
⟨*proof*⟩

lemma *bilinear-bounded*:
fixes $h :: 'm{::}euclidean\text{-}space \Rightarrow 'n{::}euclidean\text{-}space \Rightarrow 'k{::}real\text{-}normed\text{-}vector$

assumes *bh*: *bilinear h*
shows $\exists B.\ \forall x\ y.\ norm\ (h\ x\ y) \leq B * norm\ x * norm\ y$
⟨*proof*⟩

lemma *bilinear-conv-bounded-bilinear*:
fixes *h* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space* ⇒ *'c*::*real-normed-vector*
shows *bilinear h* ⟷ *bounded-bilinear h*
⟨*proof*⟩

lemma *bilinear-bounded-pos*:
fixes *h* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space* ⇒ *'c*::*real-normed-vector*
assumes *bh*: *bilinear h*
shows $\exists B > 0.\ \forall x\ y.\ norm\ (h\ x\ y) \leq B * norm\ x * norm\ y$
⟨*proof*⟩

13.11 We continue.

lemma *independent-bound*:
fixes *S* :: *'a*::*euclidean-space set*
shows *independent S* ⟹ *finite S* ∧ *card S* ≤ *DIM*(*'a*)
⟨*proof*⟩

corollary
fixes *S* :: *'a*::*euclidean-space set*
assumes *independent S*
shows *independent-imp-finite*: *finite S* **and** *independent-card-le*:*card S* ≤ *DIM*(*'a*)
⟨*proof*⟩

lemma *dependent-biggerset*:
fixes *S* :: *'a*::*euclidean-space set*
shows (*finite S* ⟹ *card S* > *DIM*(*'a*)) ⟹ *dependent S*
⟨*proof*⟩

Hence we can create a maximal independent subset.

lemma *maximal-independent-subset-extend*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *sv*: *S* ⊆ *V*
and *iS*: *independent S*
shows $\exists B.\ S \subseteq B \wedge B \subseteq V \wedge independent\ B \wedge V \subseteq span\ B$
⟨*proof*⟩

lemma *maximal-independent-subset*:
∃(*B*:: (*'a*::*euclidean-space*) *set*). *B*⊆ *V* ∧ *independent B* ∧ *V* ⊆ *span B*
⟨*proof*⟩

Notion of dimension.

definition *dim V* = (*SOME n*. ∃ *B*. *B* ⊆ *V* ∧ *independent B* ∧ *V* ⊆ *span B* ∧ *card B* = *n*)

lemma *basis-exists*:
$\exists B.\ (B :: ('a{::}euclidean\text{-}space)\ set) \subseteq V \wedge independent\ B \wedge V \subseteq span\ B \wedge (card\ B = dim\ V)$
⟨*proof*⟩

corollary *dim-le-card*:
fixes $s :: 'a{::}euclidean\text{-}space\ set$
shows $finite\ s \Longrightarrow dim\ s \leq card\ s$
⟨*proof*⟩

Consequences of independence or spanning for cardinality.

lemma *independent-card-le-dim*:
fixes $B :: 'a{::}euclidean\text{-}space\ set$
assumes $B \subseteq V$
and *independent B*
shows $card\ B \leq dim\ V$
⟨*proof*⟩

lemma *span-card-ge-dim*:
fixes $B :: 'a{::}euclidean\text{-}space\ set$
shows $B \subseteq V \Longrightarrow V \subseteq span\ B \Longrightarrow finite\ B \Longrightarrow dim\ V \leq card\ B$
⟨*proof*⟩

lemma *basis-card-eq-dim*:
fixes $V :: 'a{::}euclidean\text{-}space\ set$
shows $B \subseteq V \Longrightarrow V \subseteq span\ B \Longrightarrow independent\ B \Longrightarrow finite\ B \wedge card\ B = dim\ V$
⟨*proof*⟩

lemma *dim-unique*:
fixes $B :: 'a{::}euclidean\text{-}space\ set$
shows $B \subseteq V \Longrightarrow V \subseteq span\ B \Longrightarrow independent\ B \Longrightarrow card\ B = n \Longrightarrow dim\ V = n$
⟨*proof*⟩

More lemmas about dimension.

lemma *dim-UNIV*: $dim\ (UNIV :: 'a{::}euclidean\text{-}space\ set) = DIM('a)$
⟨*proof*⟩

lemma *dim-subset*:
fixes $S :: 'a{::}euclidean\text{-}space\ set$
shows $S \subseteq T \Longrightarrow dim\ S \leq dim\ T$
⟨*proof*⟩

lemma *dim-subset-UNIV*:
fixes $S :: 'a{::}euclidean\text{-}space\ set$
shows $dim\ S \leq DIM('a)$
⟨*proof*⟩

Converses to those.

lemma *card-ge-dim-independent*:
fixes $B :: 'a::euclidean\text{-}space\ set$
assumes BV: $B \subseteq V$
and iB: *independent* B
and dVB: *dim* $V \leq$ *card* B
shows $V \subseteq$ *span* B
⟨*proof*⟩

lemma *card-le-dim-spanning*:
assumes BV: $(B:: ('a::euclidean\text{-}space)\ set) \subseteq V$
and VB: $V \subseteq$ *span* B
and fB: *finite* B
and dVB: *dim* $V \geq$ *card* B
shows *independent* B
⟨*proof*⟩

lemma *card-eq-dim*:
fixes $B :: 'a::euclidean\text{-}space\ set$
shows $B \subseteq V \Longrightarrow$ *card* $B =$ *dim* $V \Longrightarrow$ *finite* $B \Longrightarrow$ *independent* $B \longleftrightarrow V \subseteq$ *span* B
⟨*proof*⟩

More general size bound lemmas.

lemma *independent-bound-general*:
fixes $S :: 'a::euclidean\text{-}space\ set$
shows *independent* $S \Longrightarrow$ *finite* $S \wedge$ *card* $S \leq$ *dim* S
⟨*proof*⟩

lemma *dependent-biggerset-general*:
fixes $S :: 'a::euclidean\text{-}space\ set$
shows (*finite* $S \Longrightarrow$ *card* $S >$ *dim* S) $\Longrightarrow$ *dependent* S
⟨*proof*⟩

lemma *dim-span* [*simp*]:
fixes $S :: 'a::euclidean\text{-}space\ set$
shows *dim* (*span* S) = *dim* S
⟨*proof*⟩

lemma *subset-le-dim*:
fixes $S :: 'a::euclidean\text{-}space\ set$
shows $S \subseteq$ *span* $T \Longrightarrow$ *dim* $S \leq$ *dim* T
⟨*proof*⟩

lemma *span-eq-dim*:
fixes $S :: 'a::euclidean\text{-}space\ set$
shows *span* $S =$ *span* $T \Longrightarrow$ *dim* $S =$ *dim* T
⟨*proof*⟩

lemma *spans-image*:

assumes *lf*: *linear f*
and *VB*: $V \subseteq span\ B$
shows $f\ `\ V \subseteq span\ (f\ `\ B)$
⟨*proof*⟩

lemma *dim-image-le*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes *lf*: *linear f*
shows $dim\ (f\ `\ S) \leq dim\ (S)$
⟨*proof*⟩

Relation between bases and injectivity/surjectivity of map.

lemma *spanning-surjective-image*:
assumes *us*: $UNIV \subseteq span\ S$
and *lf*: *linear f*
and *sf*: *surj f*
shows $UNIV \subseteq span\ (f\ `\ S)$
⟨*proof*⟩

lemma *independent-injective-image*:
assumes *iS*: *independent S*
and *lf*: *linear f*
and *fi*: *inj f*
shows *independent* $(f\ `\ S)$
⟨*proof*⟩

Picking an orthogonal replacement for a spanning set.

lemma *vector-sub-project-orthogonal*:
fixes $b\ x :: 'a::euclidean\text{-}space$
shows $b \cdot (x - ((b \cdot x)\ /\ (b \cdot b)) *_R b) = 0$
⟨*proof*⟩

lemma *pairwise-orthogonal-insert*:
assumes *pairwise orthogonal S*
and $\bigwedge y.\ y \in S \Longrightarrow orthogonal\ x\ y$
shows *pairwise orthogonal* $(insert\ x\ S)$
⟨*proof*⟩

lemma *basis-orthogonal*:
fixes $B :: 'a::real\text{-}inner\ set$
assumes *fB*: *finite B*
shows $\exists C.\ finite\ C \wedge card\ C \leq card\ B \wedge span\ C = span\ B \wedge pairwise\ orthogonal\ C$
(**is** $\exists C.\ ?P\ B\ C$)
⟨*proof*⟩

lemma *orthogonal-basis-exists*:
fixes $V :: ('a::euclidean\text{-}space)\ set$
shows $\exists B.\ independent\ B \wedge B \subseteq span\ V \wedge V \subseteq span\ B \wedge (card\ B = dim\ V)$

$\wedge$ *pairwise orthogonal B*
⟨*proof*⟩

lemma *span-eq*: *span S* = *span T* $\longleftrightarrow$ *S* $\subseteq$ *span T* $\wedge$ *T* $\subseteq$ *span S*
⟨*proof*⟩

Low-dimensional subset is in a hyperplane (weak orthogonal complement).

lemma *span-not-univ-orthogonal*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *sU*: *span S* $\neq$ *UNIV*
shows $\exists$ *a*::*'a*. *a* $\neq$ *0* $\wedge$ ($\forall$ *x* $\in$ *span S*. *a* $\cdot$ *x* = *0*)
⟨*proof*⟩

lemma *span-not-univ-subset-hyperplane*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *SU*: *span S* $\neq$ *UNIV*
shows $\exists$ *a*. *a* $\neq$*0* $\wedge$ *span S* $\subseteq$ {*x*. *a* $\cdot$ *x* = *0*}
⟨*proof*⟩

lemma *lowdim-subset-hyperplane*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *d*: *dim S* < *DIM*(*'a*)
shows $\exists$ *a*::*'a*. *a* $\neq$ *0* $\wedge$ *span S* $\subseteq$ {*x*. *a* $\cdot$ *x* = *0*}
⟨*proof*⟩

We can extend a linear basis-basis injection to the whole set.

lemma *linear-indep-image-lemma*:
assumes *lf*: *linear f*
and *fB*: *finite B*
and *ifB*: *independent* (*f* ' *B*)
and *fi*: *inj-on f B*
and *xsB*: *x* $\in$ *span B*
and *fx*: *f x* = *0*
shows *x* = *0*
⟨*proof*⟩

We can extend a linear mapping from basis.

lemma *linear-independent-extend-lemma*:
fixes *f* :: *'a*::*real-vector* $\Rightarrow$ *'b*::*real-vector*
assumes *fi*: *finite B*
and *ib*: *independent B*
shows $\exists$ *g*.
($\forall$ *x*$\in$ *span B*. $\forall$ *y*$\in$ *span B*. *g* (*x* + *y*) = *g x* + *g y*) $\wedge$
($\forall$ *x*$\in$ *span B*. $\forall$ *c*. *g* (*c*$*_R$ *x*) = *c* $*_R$ *g x*) $\wedge$
($\forall$ *x*$\in$ *B*. *g x* = *f x*)
⟨*proof*⟩

lemma *linear-independent-extend*:
fixes *B* :: *'a*::*euclidean-space set*

assumes *iB*: *independent B*
shows $\exists\, g.$ *linear* $g \wedge (\forall\, x{\in}B.\ g\ x = f\ x)$
⟨*proof*⟩

Can construct an isomorphism between spaces of same dimension.

lemma *subspace-isomorphism*:
fixes S :: *'a*::*euclidean-space set*
and T :: *'b*::*euclidean-space set*
assumes *s*: *subspace S*
and *t*: *subspace T*
and *d*: *dim S* = *dim T*
shows $\exists f.$ *linear f* $\wedge$ *f* ‘ *S* = *T* $\wedge$ *inj-on f S*
⟨*proof*⟩

Linear functions are equal on a subspace if they are on a spanning set.

lemma *subspace-kernel*:
assumes *lf*: *linear f*
shows *subspace* $\{x.\ f\ x = 0\}$
⟨*proof*⟩

lemma *linear-eq-0-span*:
assumes *lf*: *linear f* **and** *f0*: $\forall\, x{\in}B.\ f\ x = 0$
shows $\forall\, x \in$ *span B*. *f x* = *0*
⟨*proof*⟩

lemma *linear-eq-0*:
assumes *lf*: *linear f*
and *SB*: $S \subseteq$ *span B*
and *f0*: $\forall\, x{\in}B.\ f\ x = 0$
shows $\forall\, x \in S.\ f\ x = 0$
⟨*proof*⟩

lemma *linear-eq*:
assumes *lf*: *linear f*
and *lg*: *linear g*
and *S*: $S \subseteq$ *span B*
and *fg*: $\forall\ x{\in}\ B.\ f\ x = g\ x$
shows $\forall\, x{\in}\ S.\ f\ x = g\ x$
⟨*proof*⟩

lemma *linear-eq-stdbasis*:
fixes *f* :: *'a*::*euclidean-space* ⇒ -
assumes *lf*: *linear f*
and *lg*: *linear g*
and *fg*: $\forall\, b{\in}$*Basis*. *f b* = *g b*
shows *f* = *g*
⟨*proof*⟩

Similar results for bilinear functions.

lemma *bilinear-eq*:
assumes *bf*: *bilinear f*
and *bg*: *bilinear g*
and *SB*: $S \subseteq span\ B$
and *TC*: $T \subseteq span\ C$
and *fg*: $\forall x{\in}\ B.\ \forall y{\in}\ C.\ f\ x\ y = g\ x\ y$
shows $\forall x{\in}S.\ \forall y{\in}T.\ f\ x\ y = g\ x\ y$
⟨*proof*⟩

lemma *bilinear-eq-stdbasis*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space* ⇒ -
assumes *bf*: *bilinear f*
and *bg*: *bilinear g*
and *fg*: $\forall i{\in}Basis.\ \forall j{\in}Basis.\ f\ i\ j = g\ i\ j$
shows $f = g$
⟨*proof*⟩

Detailed theorems about left and right invertibility in general case.

lemma *linear-injective-left-inverse*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *lf*: *linear f*
and *fi*: *inj f*
shows $\exists g.\ linear\ g \wedge g \circ f = id$
⟨*proof*⟩

lemma *linear-surjective-right-inverse*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *lf*: *linear f*
and *sf*: *surj f*
shows $\exists g.\ linear\ g \wedge f \circ g = id$
⟨*proof*⟩

An injective map $'a \Rightarrow 'b$ is also surjective.

lemma *linear-injective-imp-surjective*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *fi*: *inj f*
shows *surj f*
⟨*proof*⟩

And vice versa.

lemma *surjective-iff-injective-gen*:
assumes *fS*: *finite S*
and *fT*: *finite T*
and *c*: *card S* = *card T*
and *ST*: *f* ' *S* ⊆ *T*
shows $(\forall y \in T.\ \exists x \in S.\ f\ x = y) \longleftrightarrow inj\text{-}on\ f\ S$
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *linear-surjective-imp-injective*:
fixes f :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *sf*: *surj f*
shows *inj f*
⟨*proof*⟩

Hence either is enough for isomorphism.

lemma *left-right-inverse-eq*:
assumes *fg*: $f \circ g = id$
and *gh*: $g \circ h = id$
shows $f = h$
⟨*proof*⟩

lemma *isomorphism-expand*:
$f \circ g = id \wedge g \circ f = id \longleftrightarrow (\forall x.\ f\ (g\ x) = x) \wedge (\forall x.\ g\ (f\ x) = x)$
⟨*proof*⟩

lemma *linear-injective-isomorphism*:
fixes f :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *fi*: *inj f*
shows $\exists f'.\ linear\ f' \wedge (\forall x.\ f'\ (f\ x) = x) \wedge (\forall x.\ f\ (f'\ x) = x)$
⟨*proof*⟩

lemma *linear-surjective-isomorphism*:
fixes f :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *sf*: *surj f*
shows $\exists f'.\ linear\ f' \wedge (\forall x.\ f'\ (f\ x) = x) \wedge (\forall x.\ f\ (f'\ x) = x)$
⟨*proof*⟩

Left and right inverses are the same for $'a \Rightarrow 'a$.

lemma *linear-inverse-left*:
fixes f :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *lf'*: *linear f'*
shows $f \circ f' = id \longleftrightarrow f' \circ f = id$
⟨*proof*⟩

Moreover, a one-sided inverse is automatically linear.

lemma *left-inverse-linear*:
fixes f :: *'a*::*euclidean-space* ⇒ *'a*::*euclidean-space*
assumes *lf*: *linear f*
and *gf*: $g \circ f = id$
shows *linear g*
⟨*proof*⟩

lemma *inj-linear-imp-inv-linear*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'a::euclidean\text{-}space$
assumes *linear f inj f* **shows** *linear* (*inv f*)
⟨*proof*⟩

13.12 Infinity norm

definition *infnorm* $(x::'a::euclidean\text{-}space) = Sup\ \{|x \cdot b|\ |b.\ b \in Basis\}$

lemma *infnorm-set-image*:
fixes $x :: 'a::euclidean\text{-}space$
shows $\{|x \cdot i|\ |i.\ i \in Basis\} = (\lambda i.\ |x \cdot i|)\ `\ Basis$
⟨*proof*⟩

lemma *infnorm-Max*:
fixes $x :: 'a::euclidean\text{-}space$
shows $infnorm\ x = Max\ ((\lambda i.\ |x \cdot i|)\ `\ Basis)$
⟨*proof*⟩

lemma *infnorm-set-lemma*:
fixes $x :: 'a::euclidean\text{-}space$
shows *finite* $\{|x \cdot i|\ |i.\ i \in Basis\}$
and $\{|x \cdot i|\ |i.\ i \in Basis\} \neq \{\}$
⟨*proof*⟩

lemma *infnorm-pos-le*:
fixes $x :: 'a::euclidean\text{-}space$
shows $0 \leq infnorm\ x$
⟨*proof*⟩

lemma *infnorm-triangle*:
fixes $x :: 'a::euclidean\text{-}space$
shows $infnorm\ (x + y) \leq infnorm\ x + infnorm\ y$
⟨*proof*⟩

lemma *infnorm-eq-0*:
fixes $x :: 'a::euclidean\text{-}space$
shows $infnorm\ x = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *infnorm-0*: $infnorm\ 0 = 0$
⟨*proof*⟩

lemma *infnorm-neg*: $infnorm\ (-\ x) = infnorm\ x$
⟨*proof*⟩

lemma *infnorm-sub*: $infnorm\ (x - y) = infnorm\ (y - x)$
⟨*proof*⟩

lemma *real-abs-sub-infnorm*: $|infnorm\ x - infnorm\ y| \leq infnorm\ (x - y)$
⟨*proof*⟩

lemma *real-abs-infnorm*: $|infnorm\ x| = infnorm\ x$
⟨*proof*⟩

lemma *Basis-le-infnorm*:
fixes $x :: 'a$::*euclidean-space*
shows $b \in Basis \Longrightarrow |x \cdot b| \leq infnorm\ x$
⟨*proof*⟩

lemma *infnorm-mul*: $infnorm\ (a *_R x) = |a| * infnorm\ x$
⟨*proof*⟩

lemma *infnorm-mul-lemma*: $infnorm\ (a *_R x) \leq |a| * infnorm\ x$
⟨*proof*⟩

lemma *infnorm-pos-lt*: $infnorm\ x > 0 \longleftrightarrow x \neq 0$
⟨*proof*⟩

Prove that it differs only up to a bound from Euclidean norm.

lemma *infnorm-le-norm*: $infnorm\ x \leq norm\ x$
⟨*proof*⟩

lemma (**in** *euclidean-space*) *euclidean-inner*: $inner\ x\ y = (\sum b{\in}Basis.\ (x \cdot b) * (y \cdot b))$
⟨*proof*⟩

lemma *norm-le-infnorm*:
fixes $x :: 'a$::*euclidean-space*
shows $norm\ x \leq sqrt\ DIM('a) * infnorm\ x$
⟨*proof*⟩

lemma *tendsto-infnorm* [*tendsto-intros*]:
assumes $(f \longrightarrow a)\ F$
shows $((\lambda x.\ infnorm\ (f\ x)) \longrightarrow infnorm\ a)\ F$
⟨*proof*⟩

Equality in Cauchy-Schwarz and triangle inequalities.

lemma *norm-cauchy-schwarz-eq*: $x \cdot y = norm\ x * norm\ y \longleftrightarrow norm\ x *_R y = norm\ y *_R x$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *norm-cauchy-schwarz-abs-eq*:
$|x \cdot y| = norm\ x * norm\ y \longleftrightarrow$
$norm\ x *_R y = norm\ y *_R x \vee norm\ x *_R y = -\ norm\ y *_R x$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *norm-triangle-eq*:
fixes $x\ y :: 'a$::*real-inner*
shows $norm\ (x + y) = norm\ x + norm\ y \longleftrightarrow norm\ x *_R y = norm\ y *_R x$
⟨*proof*⟩

13.13 Collinearity

definition *collinear* :: $'a$::*real-vector set* ⇒ *bool*
where $collinear\ S \longleftrightarrow (\exists u.\ \forall x \in S.\ \forall\ y \in S.\ \exists c.\ x - y = c *_R u)$

lemma *collinear-empty* [*iff*]: *collinear* {}
⟨*proof*⟩

lemma *collinear-sing* [*iff*]: *collinear* $\{x\}$
⟨*proof*⟩

lemma *collinear-2* [*iff*]: *collinear* $\{x, y\}$
⟨*proof*⟩

lemma *collinear-lemma*: $collinear\ \{0, x, y\} \longleftrightarrow x = 0 \vee y = 0 \vee (\exists c.\ y = c *_R x)$
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *norm-cauchy-schwarz-equal*: $|x \cdot y| = norm\ x * norm\ y \longleftrightarrow collinear\ \{0, x, y\}$
⟨*proof*⟩

end

14 A decision procedure for universal multivariate real arithmetic with addition, multiplication and ordering using semidefinite programming

theory *Sum-of-Squares*
imports *Complex-Main*
begin

⟨*ML*⟩

end

15 General linear decision procedure for normed spaces

theory *Norm-Arith*
imports ~~/*src/HOL/Library/Sum-of-Squares*
begin

lemma *norm-cmul-rule-thm*:
fixes $x :: 'a$::*real-normed-vector*
shows $b \geq norm\ x \Longrightarrow |c| * b \geq norm\ (scaleR\ c\ x)$
⟨*proof*⟩

lemma *norm-add-rule-thm*:
fixes *x1 x2* $:: 'a$::*real-normed-vector*
shows $norm\ x1 \leq b1 \Longrightarrow norm\ x2 \leq b2 \Longrightarrow norm\ (x1 + x2) \leq b1 + b2$
⟨*proof*⟩

lemma *ge-iff-diff-ge-0*:
fixes $a :: 'a$::*linordered-ring*
shows $a \geq b \equiv a - b \geq 0$
⟨*proof*⟩

lemma *pth-1*:
fixes $x :: 'a$::*real-normed-vector*
shows $x \equiv scaleR\ 1\ x$ ⟨*proof*⟩

lemma *pth-2*:
fixes $x :: 'a$::*real-normed-vector*
shows $x - y \equiv x + -y$
⟨*proof*⟩

lemma *pth-3*:
fixes $x :: 'a$::*real-normed-vector*
shows $- x \equiv scaleR\ (-1)\ x$
⟨*proof*⟩

lemma *pth-4*:
fixes $x :: 'a$::*real-normed-vector*
shows $scaleR\ 0\ x \equiv 0$
and $scaleR\ c\ 0 = (0::'a)$
⟨*proof*⟩

lemma *pth-5*:
fixes $x :: 'a$::*real-normed-vector*
shows $scaleR\ c\ (scaleR\ d\ x) \equiv scaleR\ (c * d)\ x$
⟨*proof*⟩

lemma *pth-6*:

fixes $x :: 'a$::*real-normed-vector*
shows *scaleR c* $(x + y) \equiv$ *scaleR c x* + *scaleR c y*
⟨*proof*⟩

lemma *pth-7*:
fixes $x :: 'a$::*real-normed-vector*
shows $0 + x \equiv x$
and $x + 0 \equiv x$
⟨*proof*⟩

lemma *pth-8*:
fixes $x :: 'a$::*real-normed-vector*
shows *scaleR c x* + *scaleR d x* $\equiv$ *scaleR* $(c + d)$ *x*
⟨*proof*⟩

lemma *pth-9*:
fixes $x :: 'a$::*real-normed-vector*
shows (*scaleR c x* + *z*) + *scaleR d x* $\equiv$ *scaleR* $(c + d)$ *x* + *z*
and *scaleR c x* + (*scaleR d x* + *z*) $\equiv$ *scaleR* $(c + d)$ *x* + *z*
and (*scaleR c x* + *w*) + (*scaleR d x* + *z*) $\equiv$ *scaleR* $(c + d)$ *x* + $(w + z)$
⟨*proof*⟩

lemma *pth-a*:
fixes $x :: 'a$::*real-normed-vector*
shows *scaleR 0 x* + *y* $\equiv$ *y*
⟨*proof*⟩

lemma *pth-b*:
fixes $x :: 'a$::*real-normed-vector*
shows *scaleR c x* + *scaleR d y* $\equiv$ *scaleR c x* + *scaleR d y*
and (*scaleR c x* + *z*) + *scaleR d y* $\equiv$ *scaleR c x* + (*z* + *scaleR d y*)
and *scaleR c x* + (*scaleR d y* + *z*) $\equiv$ *scaleR c x* + (*scaleR d y* + *z*)
and (*scaleR c x* + *w*) + (*scaleR d y* + *z*) $\equiv$ *scaleR c x* + (*w* + (*scaleR d y* + *z*))
⟨*proof*⟩

lemma *pth-c*:
fixes $x :: 'a$::*real-normed-vector*
shows *scaleR c x* + *scaleR d y* $\equiv$ *scaleR d y* + *scaleR c x*
and (*scaleR c x* + *z*) + *scaleR d y* $\equiv$ *scaleR d y* + (*scaleR c x* + *z*)
and *scaleR c x* + (*scaleR d y* + *z*) $\equiv$ *scaleR d y* + (*scaleR c x* + *z*)
and (*scaleR c x* + *w*) + (*scaleR d y* + *z*) $\equiv$ *scaleR d y* + ((*scaleR c x* + *w*) + *z*)
⟨*proof*⟩

lemma *pth-d*:
fixes $x :: 'a$::*real-normed-vector*
shows $x + 0 \equiv x$
⟨*proof*⟩

lemma *norm-imp-pos-and-ge*:
fixes x :: *'a*::*real-normed-vector*
shows $norm\ x \equiv n \Longrightarrow norm\ x \geq 0 \wedge n \geq norm\ x$
⟨*proof*⟩

lemma *real-eq-0-iff-le-ge-0*:
fixes x :: *real*
shows $x = 0 \equiv x \geq 0 \wedge -x \geq 0$
⟨*proof*⟩

lemma *norm-pths*:
fixes x :: *'a*::*real-normed-vector*
shows $x = y \longleftrightarrow norm\ (x - y) \leq 0$
and $x \neq y \longleftrightarrow \neg\ (norm\ (x - y) \leq 0)$
⟨*proof*⟩

lemmas *arithmetic-simps* =
arith-simps
add-numeral-special
add-neg-numeral-special
mult-1-left
mult-1-right

⟨*ML*⟩

Hence more metric properties.

lemma *dist-triangle-add*:
fixes $x\ y\ x'\ y'$ :: *'a*::*real-normed-vector*
shows $dist\ (x + y)\ (x' + y') \leq dist\ x\ x' + dist\ y\ y'$
⟨*proof*⟩

lemma *dist-triangle-add-half*:
fixes $x\ x'\ y\ y'$ :: *'a*::*real-normed-vector*
shows $dist\ x\ x' < e\ /\ 2 \Longrightarrow dist\ y\ y' < e\ /\ 2 \Longrightarrow dist(x + y)\ (x' + y') < e$
⟨*proof*⟩

end

16 Elementary topology in Euclidean space.

theory *Topology-Euclidean-Space*
imports
~~/*src*/*HOL*/*Library*/*Indicator-Function*
~~/*src*/*HOL*/*Library*/*Countable-Set*
~~/*src*/*HOL*/*Library*/*FuncSet*
Linear-Algebra
Norm-Arith
begin

lemma *image-affinity-interval*:
fixes $c :: 'a{::}ordered\text{-}real\text{-}vector$
shows $((\lambda x.\ m *_R x + c)\ `\ \{a..b\}) = (if\ \{a..b\}=\{\}\ then\ \{\}$
$\quad else\ if\ 0 <= m\ then\ \{m *_R a + c\ ..\ m *_R b + c\}$
$\quad else\ \{m *_R b + c\ ..\ m *_R a + c\})$
⟨*proof*⟩

lemma *countable-PiE*:
$finite\ I \Longrightarrow (\bigwedge i.\ i \in I \Longrightarrow countable\ (F\ i)) \Longrightarrow countable\ (PiE\ I\ F)$
⟨*proof*⟩

lemma *continuous-on-cases*:
$closed\ s \Longrightarrow closed\ t \Longrightarrow continuous\text{-}on\ s\ f \Longrightarrow continuous\text{-}on\ t\ g \Longrightarrow$
$\forall x.\ (x{\in}s \wedge \neg\ P\ x) \vee (x \in t \wedge P\ x) \longrightarrow f\ x = g\ x \Longrightarrow$
$continuous\text{-}on\ (s \cup t)\ (\lambda x.\ if\ P\ x\ then\ f\ x\ else\ g\ x)$
⟨*proof*⟩

16.1 Topological Basis

context *topological-space*
begin

definition *topological-basis* $B \longleftrightarrow$
$(\forall b{\in}B.\ open\ b) \wedge (\forall x.\ open\ x \longrightarrow (\exists B'.\ B' \subseteq B \wedge \bigcup B' = x))$

lemma *topological-basis*:
$topological\text{-}basis\ B \longleftrightarrow (\forall x.\ open\ x \longleftrightarrow (\exists B'.\ B' \subseteq B \wedge \bigcup B' = x))$
⟨*proof*⟩

lemma *topological-basis-iff*:
assumes $\bigwedge B'.\ B' \in B \Longrightarrow open\ B'$
shows $topological\text{-}basis\ B \longleftrightarrow (\forall O'.\ open\ O' \longrightarrow (\forall x{\in}O'.\ \exists B'{\in}B.\ x \in B' \wedge B' \subseteq O'))$
(**is** - $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *topological-basisI*:
assumes $\bigwedge B'.\ B' \in B \Longrightarrow open\ B'$
and $\bigwedge O'\ x.\ open\ O' \Longrightarrow x \in O' \Longrightarrow \exists B'{\in}B.\ x \in B' \wedge B' \subseteq O'$
shows *topological-basis B*
⟨*proof*⟩

lemma *topological-basisE*:
fixes O'
assumes *topological-basis B*
and *open* O'
and $x \in O'$
obtains B' **where** $B' \in B\ x \in B'\ B' \subseteq O'$

⟨*proof*⟩

lemma *topological-basis-open*:
assumes *topological-basis B*
and $X \in B$
shows *open X*
⟨*proof*⟩

lemma *topological-basis-imp-subbasis*:
assumes *B*: *topological-basis B*
shows *open* = *generate-topology B*
⟨*proof*⟩

lemma *basis-dense*:
fixes *B* :: *'a set set*
and *f* :: *'a set* ⇒ *'a*
assumes *topological-basis B*
and *choosefrom-basis*: $\bigwedge B'.\ B' \neq \{\} \Longrightarrow f\ B' \in B'$
shows $\forall X.$ *open* $X \longrightarrow X \neq \{\} \longrightarrow (\exists B' \in B.\ f\ B' \in X)$
⟨*proof*⟩

end

lemma *topological-basis-prod*:
assumes *A*: *topological-basis A*
and *B*: *topological-basis B*
shows *topological-basis* ((λ(*a*, *b*). *a* × *b*) ' (*A* × *B*))
⟨*proof*⟩

16.2 Countable Basis

locale *countable-basis* =
fixes *B* :: *'a*::*topological-space set set*
assumes *is-basis*: *topological-basis B*
and *countable-basis*: *countable B*
begin

lemma *open-countable-basis-ex*:
assumes *open X*
shows $\exists B' \subseteq B.\ X = \bigcup B'$
⟨*proof*⟩

lemma *open-countable-basisE*:
assumes *open X*
obtains *B'* **where** $B' \subseteq B\ X = \bigcup B'$
⟨*proof*⟩

lemma *countable-dense-exists*:
$\exists D$::*'a set*. *countable D* ∧ $(\forall X.$ *open* $X \longrightarrow X \neq \{\} \longrightarrow (\exists d \in D.\ d \in X))$

⟨*proof*⟩

lemma *countable-dense-setE*:
obtains $D :: 'a\ set$
where *countable* D $\bigwedge X.$ *open* $X \Longrightarrow X \neq \{\} \Longrightarrow \exists\, d \in D.\ d \in X$
⟨*proof*⟩

end

lemma (**in** *first-countable-topology*) *first-countable-basisE*:
obtains A **where** *countable* A $\bigwedge a.\ a \in A \Longrightarrow x \in a$ $\bigwedge a.\ a \in A \Longrightarrow$ *open* a
$\bigwedge S.$ *open* $S \Longrightarrow x \in S \Longrightarrow (\exists\, a{\in}A.\ a \subseteq S)$
⟨*proof*⟩

lemma (**in** *first-countable-topology*) *first-countable-basis-Int-stableE*:
obtains A **where** *countable* A $\bigwedge a.\ a \in A \Longrightarrow x \in a$ $\bigwedge a.\ a \in A \Longrightarrow$ *open* a
$\bigwedge S.$ *open* $S \Longrightarrow x \in S \Longrightarrow (\exists\, a{\in}A.\ a \subseteq S)$
$\bigwedge a\ b.\ a \in A \Longrightarrow b \in A \Longrightarrow a \cap b \in A$
⟨*proof*⟩

lemma (**in** *topological-space*) *first-countableI*:
assumes *countable* A
and *1*: $\bigwedge a.\ a \in A \Longrightarrow x \in a$ $\bigwedge a.\ a \in A \Longrightarrow$ *open* a
and *2*: $\bigwedge S.$ *open* $S \Longrightarrow x \in S \Longrightarrow \exists\, a{\in}A.\ a \subseteq S$
shows $\exists\, A{::}nat \Rightarrow 'a\ set.\ (\forall\, i.\ x \in A\ i \wedge open\ (A\ i)) \wedge (\forall\, S.\ open\ S \wedge x \in S$
$\longrightarrow (\exists\, i.\ A\ i \subseteq S))$
⟨*proof*⟩

instance *prod* :: (*first-countable-topology*, *first-countable-topology*) *first-countable-topology*
⟨*proof*⟩

class *second-countable-topology* = *topological-space* +
assumes *ex-countable-subbasis*:
$\exists\, B{::}'a{::}$*topological-space set set*. *countable* $B \wedge$ *open* = *generate-topology* B
begin

lemma *ex-countable-basis*: $\exists\, B{::}'a\ set\ set.$ *countable* $B \wedge$ *topological-basis* B
⟨*proof*⟩

end

sublocale *second-countable-topology* <
countable-basis SOME B. *countable* $B \wedge$ *topological-basis* B
⟨*proof*⟩

instance *prod* :: (*second-countable-topology*, *second-countable-topology*) *second-countable-topology*
⟨*proof*⟩

instance *second-countable-topology* ⊆ *first-countable-topology*

⟨*proof*⟩

16.3 Polish spaces

Textbooks define Polish spaces as completely metrizable. We assume the topology to be complete for a given metric.

class *polish-space* = *complete-space* + *second-countable-topology*

16.4 General notion of a topology as a value

definition *istopology* $L \longleftrightarrow$
$L\ \{\} \wedge (\forall S\ T.\ L\ S \longrightarrow L\ T \longrightarrow L\ (S \cap T)) \wedge (\forall K.\ Ball\ K\ L \longrightarrow L\ (\bigcup K))$

typedef $'a$ *topology* = $\{L::('a\ set) \Rightarrow bool.\ istopology\ L\}$
morphisms *openin topology*
⟨*proof*⟩

lemma *istopology-openin*[*intro*]: *istopology*(*openin U*)
⟨*proof*⟩

lemma *topology-inverse'*: *istopology U* $\Longrightarrow$ *openin* (*topology U*) = *U*
⟨*proof*⟩

lemma *topology-inverse-iff*: *istopology U* $\longleftrightarrow$ *openin* (*topology U*) = *U*
⟨*proof*⟩

lemma *topology-eq*: *T1* = *T2* $\longleftrightarrow$ $(\forall S.$ *openin T1 S* $\longleftrightarrow$ *openin T2 S*)
⟨*proof*⟩

Infer the "universe" from union of all sets in the topology.

definition *topspace* $T = \bigcup \{S.\ openin\ T\ S\}$

16.4.1 Main properties of open sets

lemma *openin-clauses*:
fixes *U* :: $'a$ *topology*
shows
openin U {}
$\bigwedge S\ T.$ *openin U S* $\Longrightarrow$ *openin U T* $\Longrightarrow$ *openin U* (*S*∩*T*)
$\bigwedge K.\ (\forall S \in K.\ openin\ U\ S) \Longrightarrow openin\ U\ (\bigcup K)$
⟨*proof*⟩

lemma *openin-subset*[*intro*]: *openin U S* $\Longrightarrow$ *S* $\subseteq$ *topspace U*
⟨*proof*⟩

lemma *openin-empty*[*simp*]: *openin U* {}
⟨*proof*⟩

lemma *openin-Int*[*intro*]: *openin U S* $\Longrightarrow$ *openin U T* $\Longrightarrow$ *openin U* (*S* ∩ *T*)

⟨*proof*⟩

lemma *openin-Union*[*intro*]: $(\bigwedge S.\ S \in K \Longrightarrow \mathit{openin}\ U\ S) \Longrightarrow \mathit{openin}\ U\ (\bigcup K)$
⟨*proof*⟩

lemma *openin-Un*[*intro*]: $\mathit{openin}\ U\ S \Longrightarrow \mathit{openin}\ U\ T \Longrightarrow \mathit{openin}\ U\ (S \cup T)$
⟨*proof*⟩

lemma *openin-topspace*[*intro*, *simp*]: $\mathit{openin}\ U\ (\mathit{topspace}\ U)$
⟨*proof*⟩

lemma *openin-subopen*: $\mathit{openin}\ U\ S \longleftrightarrow (\forall x \in S.\ \exists T.\ \mathit{openin}\ U\ T \wedge x \in T \wedge T \subseteq S)$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

16.4.2 Closed sets

definition $\mathit{closedin}\ U\ S \longleftrightarrow S \subseteq \mathit{topspace}\ U \wedge \mathit{openin}\ U\ (\mathit{topspace}\ U - S)$

lemma *closedin-subset*: $\mathit{closedin}\ U\ S \Longrightarrow S \subseteq \mathit{topspace}\ U$
⟨*proof*⟩

lemma *closedin-empty*[*simp*]: $\mathit{closedin}\ U\ \{\}$
⟨*proof*⟩

lemma *closedin-topspace*[*intro*, *simp*]: $\mathit{closedin}\ U\ (\mathit{topspace}\ U)$
⟨*proof*⟩

lemma *closedin-Un*[*intro*]: $\mathit{closedin}\ U\ S \Longrightarrow \mathit{closedin}\ U\ T \Longrightarrow \mathit{closedin}\ U\ (S \cup T)$
⟨*proof*⟩

lemma *Diff-Inter*[*intro*]: $A - \bigcap S = \bigcup \{A - s | s.\ s \in S\}$
⟨*proof*⟩

lemma *closedin-Inter*[*intro*]:
assumes *Ke*: $K \neq \{\}$
and *Kc*: $\bigwedge S.\ S \in K \Longrightarrow \mathit{closedin}\ U\ S$
shows $\mathit{closedin}\ U\ (\bigcap K)$
⟨*proof*⟩

lemma *closedin-INT*[*intro*]:
assumes $A \neq \{\}$ $\bigwedge x.\ x \in A \Longrightarrow \mathit{closedin}\ U\ (B\ x)$
shows $\mathit{closedin}\ U\ (\bigcap x \in A.\ B\ x)$
⟨*proof*⟩

lemma *closedin-Int*[*intro*]: $\mathit{closedin}\ U\ S \Longrightarrow \mathit{closedin}\ U\ T \Longrightarrow \mathit{closedin}\ U\ (S \cap T)$

⟨*proof*⟩

lemma *openin-closedin-eq*: *openin U S* ⟷ *S* ⊆ *topspace U* ∧ *closedin U* (*topspace U* − *S*)
⟨*proof*⟩

lemma *openin-closedin*: *S* ⊆ *topspace U* ⟹ (*openin U S* ⟷ *closedin U* (*topspace U* − *S*))
⟨*proof*⟩

lemma *openin-diff*[*intro*]:
assumes *oS*: *openin U S*
and *cT*: *closedin U T*
shows *openin U* (*S* − *T*)
⟨*proof*⟩

lemma *closedin-diff*[*intro*]:
assumes *oS*: *closedin U S*
and *cT*: *openin U T*
shows *closedin U* (*S* − *T*)
⟨*proof*⟩

16.4.3 Subspace topology

definition *subtopology U V* = *topology* (λ*T*. ∃ *S*. *T* = *S* ∩ *V* ∧ *openin U S*)

lemma *istopology-subtopology*: *istopology* (λ*T*. ∃ *S*. *T* = *S* ∩ *V* ∧ *openin U S*)
(**is** *istopology ?L*)
⟨*proof*⟩

lemma *openin-subtopology*: *openin* (*subtopology U V*) *S* ⟷ (∃ *T*. *openin U T* ∧ *S* = *T* ∩ *V*)
⟨*proof*⟩

lemma *topspace-subtopology*: *topspace* (*subtopology U V*) = *topspace U* ∩ *V*
⟨*proof*⟩

lemma *closedin-subtopology*: *closedin* (*subtopology U V*) *S* ⟷ (∃ *T*. *closedin U T* ∧ *S* = *T* ∩ *V*)
⟨*proof*⟩

lemma *openin-subtopology-refl*: *openin* (*subtopology U V*) *V* ⟷ *V* ⊆ *topspace U*
⟨*proof*⟩

lemma *subtopology-superset*:
assumes *UV*: *topspace U* ⊆ *V*
shows *subtopology U V* = *U*
⟨*proof*⟩

lemma *subtopology-topspace*[*simp*]: *subtopology U* (*topspace U*) = *U*
⟨*proof*⟩

lemma *subtopology-UNIV*[*simp*]: *subtopology U UNIV* = *U*
⟨*proof*⟩

lemma *openin-subtopology-empty*:
openin (*subtopology U* {}) *s* ⟷ *s* = {}
⟨*proof*⟩

lemma *closedin-subtopology-empty*:
closedin (*subtopology U* {}) *s* ⟷ *s* = {}
⟨*proof*⟩

lemma *closedin-subtopology-refl*:
closedin (*subtopology U u*) *u* ⟷ *u* ⊆ *topspace U*
⟨*proof*⟩

lemma *openin-imp-subset*:
openin (*subtopology U s*) *t* ⟹ *t* ⊆ *s*
⟨*proof*⟩

lemma *closedin-imp-subset*:
closedin (*subtopology U s*) *t* ⟹ *t* ⊆ *s*
⟨*proof*⟩

lemma *openin-subtopology-Un*:
openin (*subtopology U t*) *s* ∧ *openin* (*subtopology U u*) *s*
⟹ *openin* (*subtopology U* (*t* ∪ *u*)) *s*
⟨*proof*⟩

16.4.4 The standard Euclidean topology

definition *euclidean* :: *'a*::*topological-space topology*
where *euclidean* = *topology open*

lemma *open-openin*: *open S* ⟷ *openin euclidean S*
⟨*proof*⟩

lemma *topspace-euclidean*: *topspace euclidean* = *UNIV*
⟨*proof*⟩

lemma *topspace-euclidean-subtopology*[*simp*]: *topspace* (*subtopology euclidean S*)
= *S*
⟨*proof*⟩

lemma *closed-closedin*: *closed S* ⟷ *closedin euclidean S*
⟨*proof*⟩

lemma *open-subopen*: *open S* $\longleftrightarrow$ ($\forall$ *x*$\in$*S*. $\exists$ *T*. *open T* $\wedge$ *x* $\in$ *T* $\wedge$ *T* $\subseteq$ *S*)
⟨*proof*⟩

lemma *openin-subtopology-self* [*simp*]: *openin* (*subtopology euclidean S*) *S*
⟨*proof*⟩

Basic "localization" results are handy for connectedness.

lemma *openin-open*: *openin* (*subtopology euclidean U*) *S* $\longleftrightarrow$ ($\exists$ *T*. *open T* $\wedge$ (*S* = *U* $\cap$ *T*))
⟨*proof*⟩

lemma *openin-open-Int*[*intro*]: *open S* $\Longrightarrow$ *openin* (*subtopology euclidean U*) (*U* $\cap$ *S*)
⟨*proof*⟩

lemma *open-openin-trans*[*trans*]:
open S $\Longrightarrow$ *open T* $\Longrightarrow$ *T* $\subseteq$ *S* $\Longrightarrow$ *openin* (*subtopology euclidean S*) *T*
⟨*proof*⟩

lemma *open-subset*: *S* $\subseteq$ *T* $\Longrightarrow$ *open S* $\Longrightarrow$ *openin* (*subtopology euclidean T*) *S*
⟨*proof*⟩

lemma *closedin-closed*: *closedin* (*subtopology euclidean U*) *S* $\longleftrightarrow$ ($\exists$ *T*. *closed T* $\wedge$ *S* = *U* $\cap$ *T*)
⟨*proof*⟩

lemma *closedin-closed-Int*: *closed S* $\Longrightarrow$ *closedin* (*subtopology euclidean U*) (*U* $\cap$ *S*)
⟨*proof*⟩

lemma *closed-closedin-trans*:
closed S $\Longrightarrow$ *closed T* $\Longrightarrow$ *T* $\subseteq$ *S* $\Longrightarrow$ *closedin* (*subtopology euclidean S*) *T*
⟨*proof*⟩

lemma *closed-subset*: *S* $\subseteq$ *T* $\Longrightarrow$ *closed S* $\Longrightarrow$ *closedin* (*subtopology euclidean T*) *S*
⟨*proof*⟩

lemma *openin-euclidean-subtopology-iff*:
fixes *S U* :: *'a*::*metric-space set*
shows *openin* (*subtopology euclidean U*) *S* $\longleftrightarrow$
S $\subseteq$ *U* $\wedge$ ($\forall$ *x*$\in$*S*. $\exists$ *e*>*0*. $\forall$ *x'*$\in$*U*. *dist x' x* < *e* $\longrightarrow$ *x'*$\in$ *S*)
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *connected-openin*:
connected s $\longleftrightarrow$
$\sim$($\exists$ *e1 e2*. *openin* (*subtopology euclidean s*) *e1* $\wedge$

openin (*subtopology euclidean s*) *e2* ∧
s ⊆ *e1* ∪ *e2* ∧ *e1* ∩ *e2* = {} ∧ *e1* ≠ {} ∧ *e2* ≠ {})
⟨*proof*⟩

lemma *connected-openin-eq*:
connected s ⟷
~(∃ *e1 e2*. *openin* (*subtopology euclidean s*) *e1* ∧
openin (*subtopology euclidean s*) *e2* ∧
e1 ∪ *e2* = *s* ∧ *e1* ∩ *e2* = {} ∧
e1 ≠ {} ∧ *e2* ≠ {})
⟨*proof*⟩

lemma *connected-closedin*:
connected s ⟷
~(∃ *e1 e2*.
closedin (*subtopology euclidean s*) *e1* ∧
closedin (*subtopology euclidean s*) *e2* ∧
s ⊆ *e1* ∪ *e2* ∧ *e1* ∩ *e2* = {} ∧
e1 ≠ {} ∧ *e2* ≠ {})
⟨*proof*⟩

lemma *connected-closedin-eq*:
connected s ⟷
~(∃ *e1 e2*.
closedin (*subtopology euclidean s*) *e1* ∧
closedin (*subtopology euclidean s*) *e2* ∧
e1 ∪ *e2* = *s* ∧ *e1* ∩ *e2* = {} ∧
e1 ≠ {} ∧ *e2* ≠ {})
⟨*proof*⟩

These "transitivity" results are handy too

lemma *openin-trans*[*trans*]:
openin (*subtopology euclidean T*) *S* ⟹ *openin* (*subtopology euclidean U*) *T* ⟹
openin (*subtopology euclidean U*) *S*
⟨*proof*⟩

lemma *openin-open-trans*: *openin* (*subtopology euclidean T*) *S* ⟹ *open T* ⟹
open S
⟨*proof*⟩

lemma *closedin-trans*[*trans*]:
closedin (*subtopology euclidean T*) *S* ⟹ *closedin* (*subtopology euclidean U*) *T*
⟹
closedin (*subtopology euclidean U*) *S*
⟨*proof*⟩

lemma *closedin-closed-trans*: *closedin* (*subtopology euclidean T*) *S* ⟹ *closed T*
⟹ *closed S*
⟨*proof*⟩

lemma *openin-subtopology-Int-subset*:
⟦*openin* (*subtopology euclidean u*) (*u* ∩ *S*); *v* ⊆ *u*⟧ ⟹ *openin* (*subtopology euclidean v*) (*v* ∩ *S*)
⟨*proof*⟩

lemma *openin-open-eq*: *open s* ⟹ (*openin* (*subtopology euclidean s*) *t* ⟷ *open t* ∧ *t* ⊆ *s*)
⟨*proof*⟩

16.5 Open and closed balls

definition *ball* :: 'a::*metric-space* ⇒ *real* ⇒ 'a *set*
where *ball x e* = {*y*. *dist x y* < *e*}

definition *cball* :: 'a::*metric-space* ⇒ *real* ⇒ 'a *set*
where *cball x e* = {*y*. *dist x y* ≤ *e*}

definition *sphere* :: 'a::*metric-space* ⇒ *real* ⇒ 'a *set*
where *sphere x e* = {*y*. *dist x y* = *e*}

lemma *mem-ball* [*simp*]: *y* ∈ *ball x e* ⟷ *dist x y* < *e*
⟨*proof*⟩

lemma *mem-cball* [*simp*]: *y* ∈ *cball x e* ⟷ *dist x y* ≤ *e*
⟨*proof*⟩

lemma *mem-sphere* [*simp*]: *y* ∈ *sphere x e* ⟷ *dist x y* = *e*
⟨*proof*⟩

lemma *ball-trivial* [*simp*]: *ball x 0* = {}
⟨*proof*⟩

lemma *cball-trivial* [*simp*]: *cball x 0* = {*x*}
⟨*proof*⟩

lemma *mem-ball-0* [*simp*]:
fixes *x* :: 'a::*real-normed-vector*
shows *x* ∈ *ball 0 e* ⟷ *norm x* < *e*
⟨*proof*⟩

lemma *mem-cball-0* [*simp*]:
fixes *x* :: 'a::*real-normed-vector*
shows *x* ∈ *cball 0 e* ⟷ *norm x* ≤ *e*
⟨*proof*⟩

lemma *centre-in-ball* [*simp*]: *x* ∈ *ball x e* ⟷ *0* < *e*
⟨*proof*⟩

lemma *centre-in-cball* [*simp*]: $x \in cball\ x\ e \longleftrightarrow 0 \leq e$
⟨*proof*⟩

lemma *ball-subset-cball* [*simp*,*intro*]: $ball\ x\ e \subseteq cball\ x\ e$
⟨*proof*⟩

lemma *sphere-cball* [*simp*,*intro*]: $sphere\ z\ r \subseteq cball\ z\ r$
⟨*proof*⟩

lemma *subset-ball*[*intro*]: $d \leq e \Longrightarrow ball\ x\ d \subseteq ball\ x\ e$
⟨*proof*⟩

lemma *subset-cball*[*intro*]: $d \leq e \Longrightarrow cball\ x\ d \subseteq cball\ x\ e$
⟨*proof*⟩

lemma *ball-max-Un*: $ball\ a\ (max\ r\ s) = ball\ a\ r \cup ball\ a\ s$
⟨*proof*⟩

lemma *ball-min-Int*: $ball\ a\ (min\ r\ s) = ball\ a\ r \cap ball\ a\ s$
⟨*proof*⟩

lemma *cball-diff-eq-sphere*: $cball\ a\ r - ball\ a\ r = \{x.\ dist\ x\ a = r\}$
⟨*proof*⟩

lemma *image-add-ball* [*simp*]:
fixes $a :: 'a$::*real-normed-vector*
shows $op + b\ `\ ball\ a\ r = ball\ (a+b)\ r$
⟨*proof*⟩

lemma *image-add-cball* [*simp*]:
fixes $a :: 'a$::*real-normed-vector*
shows $op + b\ `\ cball\ a\ r = cball\ (a+b)\ r$
⟨*proof*⟩

lemma *open-ball* [*intro*, *simp*]: $open\ (ball\ x\ e)$
⟨*proof*⟩

lemma *open-contains-ball*: $open\ S \longleftrightarrow (\forall x{\in}S.\ \exists e{>}0.\ ball\ x\ e \subseteq S)$
⟨*proof*⟩

lemma *openI* [*intro?*]: $(\bigwedge x.\ x{\in}S \Longrightarrow \exists e{>}0.\ ball\ x\ e \subseteq S) \Longrightarrow open\ S$
⟨*proof*⟩

lemma *openE*[*elim?*]:
assumes $open\ S\ x{\in}S$
obtains e **where** $e{>}0\ ball\ x\ e \subseteq S$
⟨*proof*⟩

lemma *open-contains-ball-eq*: $open\ S \Longrightarrow x{\in}S \longleftrightarrow (\exists e{>}0.\ ball\ x\ e \subseteq S)$

⟨*proof*⟩

lemma *openin-contains-ball*:
  openin (*subtopology euclidean t*) *s* ⟷
   s ⊆ *t* ∧ (∀ *x* ∈ *s*. ∃ *e*. 0 < *e* ∧ *ball x e* ∩ *t* ⊆ *s*)
  (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *openin-contains-cball*:
  openin (*subtopology euclidean t*) *s* ⟷
    s ⊆ *t* ∧
    (∀ *x* ∈ *s*. ∃ *e*. 0 < *e* ∧ *cball x e* ∩ *t* ⊆ *s*)
⟨*proof*⟩

lemma *ball-eq-empty*[*simp*]: *ball x e* = {} ⟷ *e* ≤ 0
⟨*proof*⟩

lemma *ball-empty*: *e* ≤ 0 ⟹ *ball x e* = {} ⟨*proof*⟩

lemma *euclidean-dist-l2*:
fixes *x y* :: *'a* :: *euclidean-space*
shows *dist x y* = *setL2* (λ*i*. *dist* (*x* · *i*) (*y* · *i*)) *Basis*
⟨*proof*⟩

lemma *eventually-nhds-ball*: *d* > 0 ⟹ *eventually* (λ*x*. *x* ∈ *ball z d*) (*nhds z*)
⟨*proof*⟩

lemma *eventually-at-ball*: *d* > 0 ⟹ *eventually* (λ*t*. *t* ∈ *ball z d* ∧ *t* ∈ *A*) (*at z within A*)
⟨*proof*⟩

lemma *eventually-at-ball'*: *d* > 0 ⟹ *eventually* (λ*t*. *t* ∈ *ball z d* ∧ *t* ≠ *z* ∧ *t* ∈ *A*) (*at z within A*)
⟨*proof*⟩

16.6 Boxes

abbreviation *One* :: *'a*::*euclidean-space*
where *One* ≡ ∑ *Basis*

definition (**in** *euclidean-space*) *eucl-less* (**infix** <*e* 50)
where *eucl-less a b* ⟷ (∀ *i*∈*Basis*. *a* · *i* < *b* · *i*)

definition *box-eucl-less*: *box a b* = {*x*. *a* <*e x* ∧ *x* <*e b*}
definition *cbox a b* = {*x*. ∀ *i*∈*Basis*. *a* · *i* ≤ *x* · *i* ∧ *x* · *i* ≤ *b* · *i*}

lemma *box-def*: *box a b* = {*x*. ∀ *i*∈*Basis*. *a* · *i* < *x* · *i* ∧ *x* · *i* < *b* · *i*}
and *in-box-eucl-less*: *x* ∈ *box a b* ⟷ *a* <*e x* ∧ *x* <*e b*
and *mem-box*: *x* ∈ *box a b* ⟷ (∀ *i*∈*Basis*. *a* · *i* < *x* · *i* ∧ *x* · *i* < *b* · *i*)

$x \in cbox\ a\ b \longleftrightarrow (\forall i{\in}Basis.\ a \cdot i \leq x \cdot i \wedge x \cdot i \leq b \cdot i)$
⟨*proof*⟩

lemma *cbox-Pair-eq*: $cbox\ (a,\ c)\ (b,\ d) = cbox\ a\ b \times cbox\ c\ d$
⟨*proof*⟩

lemma *cbox-Pair-iff* [*iff*]: $(x,\ y) \in cbox\ (a,\ c)\ (b,\ d) \longleftrightarrow x \in cbox\ a\ b \wedge y \in cbox\ c\ d$
⟨*proof*⟩

lemma *cbox-Pair-eq-0*: $cbox\ (a,\ c)\ (b,\ d) = \{\} \longleftrightarrow cbox\ a\ b = \{\} \vee cbox\ c\ d = \{\}$
⟨*proof*⟩

lemma *swap-cbox-Pair* [*simp*]: $prod.swap\ `\ cbox\ (c,\ a)\ (d,\ b) = cbox\ (a,c)\ (b,d)$
⟨*proof*⟩

lemma *mem-box-real*[*simp*]:
$(x{::}real) \in box\ a\ b \longleftrightarrow a < x \wedge x < b$
$(x{::}real) \in cbox\ a\ b \longleftrightarrow a \leq x \wedge x \leq b$
⟨*proof*⟩

lemma *box-real*[*simp*]:
fixes *a b*:: *real*
shows $box\ a\ b = \{a <..< b\}$ $cbox\ a\ b = \{a\ ..\ b\}$
⟨*proof*⟩

lemma *box-Int-box*:
fixes $a :: 'a{::}euclidean\text{-}space$
shows $box\ a\ b \cap box\ c\ d =$
$box\ (\sum i{\in}Basis.\ max\ (a{\cdot}i)\ (c{\cdot}i) *_R i)\ (\sum i{\in}Basis.\ min\ (b{\cdot}i)\ (d{\cdot}i) *_R i)$
⟨*proof*⟩

lemma *rational-boxes*:
fixes $x :: 'a{::}euclidean\text{-}space$
assumes $e > 0$
shows $\exists a\ b.\ (\forall i{\in}Basis.\ a \cdot i \in \mathbb{Q} \wedge b \cdot i \in \mathbb{Q}\ ) \wedge x \in box\ a\ b \wedge box\ a\ b \subseteq ball\ x\ e$
⟨*proof*⟩

lemma *open-UNION-box*:
fixes $M :: 'a{::}euclidean\text{-}space\ set$
assumes *open M*
defines $a' \equiv \lambda f :: 'a \Rightarrow real \times real.\ (\sum (i{::}'a){\in}Basis.\ fst\ (f\ i) *_R i)$
defines $b' \equiv \lambda f :: 'a \Rightarrow real \times real.\ (\sum (i{::}'a){\in}Basis.\ snd\ (f\ i) *_R i)$
defines $I \equiv \{f{\in}Basis \to_E \mathbb{Q} \times \mathbb{Q}.\ box\ (a'\ f)\ (b'\ f) \subseteq M\}$
shows $M = (\bigcup f{\in}I.\ box\ (a'\ f)\ (b'\ f))$
⟨*proof*⟩

lemma *box-eq-empty*:
fixes $a :: 'a$::*euclidean-space*
shows $(box\ a\ b = \{\} \longleftrightarrow (\exists\, i{\in}Basis.\ b{\cdot}i \leq a{\cdot}i))$ (**is** *?th1*)
and $(cbox\ a\ b = \{\} \longleftrightarrow (\exists\, i{\in}Basis.\ b{\cdot}i < a{\cdot}i))$ (**is** *?th2*)
⟨*proof*⟩

lemma *box-ne-empty*:
fixes $a :: 'a$::*euclidean-space*
shows $cbox\ a\ b \neq \{\} \longleftrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i \leq b{\cdot}i)$
and $box\ a\ b \neq \{\} \longleftrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i < b{\cdot}i)$
⟨*proof*⟩

lemma
fixes $a :: 'a$::*euclidean-space*
shows *cbox-sing*: $cbox\ a\ a = \{a\}$
and *box-sing*: $box\ a\ a = \{\}$
⟨*proof*⟩

lemma *subset-box-imp*:
fixes $a :: 'a$::*euclidean-space*
shows $(\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i) \Longrightarrow cbox\ c\ d \subseteq cbox\ a\ b$
and $(\forall\, i{\in}Basis.\ a{\cdot}i < c{\cdot}i \wedge d{\cdot}i < b{\cdot}i) \Longrightarrow cbox\ c\ d \subseteq box\ a\ b$
and $(\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i) \Longrightarrow box\ c\ d \subseteq cbox\ a\ b$
and $(\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i) \Longrightarrow box\ c\ d \subseteq box\ a\ b$
⟨*proof*⟩

lemma *box-subset-cbox*:
fixes $a :: 'a$::*euclidean-space*
shows $box\ a\ b \subseteq cbox\ a\ b$
⟨*proof*⟩

lemma *subset-box*:
fixes $a :: 'a$::*euclidean-space*
shows $cbox\ c\ d \subseteq cbox\ a\ b \longleftrightarrow (\forall\, i{\in}Basis.\ c{\cdot}i \leq d{\cdot}i) \longrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i)$ (**is** *?th1*)
and $cbox\ c\ d \subseteq box\ a\ b \longleftrightarrow (\forall\, i{\in}Basis.\ c{\cdot}i \leq d{\cdot}i) \longrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i < c{\cdot}i \wedge d{\cdot}i < b{\cdot}i)$ (**is** *?th2*)
and $box\ c\ d \subseteq cbox\ a\ b \longleftrightarrow (\forall\, i{\in}Basis.\ c{\cdot}i < d{\cdot}i) \longrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i)$ (**is** *?th3*)
and $box\ c\ d \subseteq box\ a\ b \longleftrightarrow (\forall\, i{\in}Basis.\ c{\cdot}i < d{\cdot}i) \longrightarrow (\forall\, i{\in}Basis.\ a{\cdot}i \leq c{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i)$ (**is** *?th4*)
⟨*proof*⟩

lemma *inter-interval*:
fixes $a :: 'a$::*euclidean-space*
shows $cbox\ a\ b \cap cbox\ c\ d =$
$cbox\ (\sum i{\in}Basis.\ max\ (a{\cdot}i)\ (c{\cdot}i) *_R i)\ (\sum i{\in}Basis.\ min\ (b{\cdot}i)\ (d{\cdot}i) *_R i)$
⟨*proof*⟩

lemma *disjoint-interval*:
fixes *a*::*'a*::*euclidean-space*
shows *cbox a b* ∩ *cbox c d* = {} ⟷ (∃ *i*∈*Basis*. (*b*•*i* < *a*•*i* ∨ *d*•*i* < *c*•*i* ∨ *b*•*i* < *c*•*i* ∨ *d*•*i* < *a*•*i*)) (**is** *?th1*)
and *cbox a b* ∩ *box c d* = {} ⟷ (∃ *i*∈*Basis*. (*b*•*i* < *a*•*i* ∨ *d*•*i* ≤ *c*•*i* ∨ *b*•*i* ≤ *c*•*i* ∨ *d*•*i* ≤ *a*•*i*)) (**is** *?th2*)
and *box a b* ∩ *cbox c d* = {} ⟷ (∃ *i*∈*Basis*. (*b*•*i* ≤ *a*•*i* ∨ *d*•*i* < *c*•*i* ∨ *b*•*i* ≤ *c*•*i* ∨ *d*•*i* ≤ *a*•*i*)) (**is** *?th3*)
and *box a b* ∩ *box c d* = {} ⟷ (∃ *i*∈*Basis*. (*b*•*i* ≤ *a*•*i* ∨ *d*•*i* ≤ *c*•*i* ∨ *b*•*i* ≤ *c*•*i* ∨ *d*•*i* ≤ *a*•*i*)) (**is** *?th4*)
⟨*proof*⟩

lemma *UN-box-eq-UNIV*: (⋃ *i*::*nat*. *box* (− (*real i* ∗$_R$ *One*)) (*real i* ∗$_R$ *One*)) = *UNIV*
⟨*proof*⟩

Intervals in general, including infinite and mixtures of open and closed.

definition *is-interval* (*s*::(*'a*::*euclidean-space*) *set*) ⟷
(∀ *a*∈*s*. ∀ *b*∈*s*. ∀ *x*. (∀ *i*∈*Basis*. ((*a*•*i* ≤ *x*•*i* ∧ *x*•*i* ≤ *b*•*i*) ∨ (*b*•*i* ≤ *x*•*i* ∧ *x*•*i* ≤ *a*•*i*))) ⟶ *x* ∈ *s*)

lemma *is-interval-cbox*: *is-interval* (*cbox a* (*b*::*'a*::*euclidean-space*)) (**is** *?th1*)
and *is-interval-box*: *is-interval* (*box a b*) (**is** *?th2*)
⟨*proof*⟩

lemma *is-interval-empty* [*iff*]: *is-interval* {}
⟨*proof*⟩

lemma *is-interval-univ* [*iff*]: *is-interval UNIV*
⟨*proof*⟩

lemma *mem-is-intervalI*:
assumes *is-interval s*
assumes *a* ∈ *s b* ∈ *s*
assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* • *i* ≤ *x* • *i* ∧ *x* • *i* ≤ *b* • *i* ∨ *b* • *i* ≤ *x* • *i* ∧ *x* • *i* ≤ *a* • *i*
shows *x* ∈ *s*
⟨*proof*⟩

lemma *interval-subst*:
fixes *S*::*'a*::*euclidean-space set*
assumes *is-interval S*
assumes *x* ∈ *S y j* ∈ *S*
assumes *j* ∈ *Basis*
shows (∑ *i*∈*Basis*. (*if i* = *j then y i* • *i else x* • *i*) ∗$_R$ *i*) ∈ *S*
⟨*proof*⟩

lemma *mem-box-componentwiseI*:
fixes *S*::*'a*::*euclidean-space set*

assumes *is-interval S*
assumes $\bigwedge i.\ i \in Basis \Longrightarrow x \cdot i \in ((\lambda x.\ x \cdot i)\ `\ S)$
shows $x \in S$
⟨*proof*⟩

16.7 Connectedness

lemma *connected-local*:
connected S ⟷
¬ (∃ *e1 e2*.
openin (*subtopology euclidean S*) *e1* ∧
openin (*subtopology euclidean S*) *e2* ∧
$S \subseteq e1 \cup e2$ ∧
$e1 \cap e2 = \{\}$ ∧
$e1 \neq \{\}$ ∧
$e2 \neq \{\}$)
⟨*proof*⟩

lemma *exists-diff*:
fixes *P* :: *'a set* ⇒ *bool*
shows (∃ *S*. *P* (− *S*)) ⟷ (∃ *S*. *P S*) (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *connected-clopen*: *connected S* ⟷
(∀ *T*. *openin* (*subtopology euclidean S*) *T* ∧
closedin (*subtopology euclidean S*) *T* ⟶ $T = \{\} \vee T = S$) (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

16.8 Limit points

definition (**in** *topological-space*) *islimpt*:: *'a* ⇒ *'a set* ⇒ *bool* (**infixr** *islimpt 60*)
where *x islimpt S* ⟷ (∀ *T*. *x*∈*T* ⟶ *open T* ⟶ (∃ *y*∈*S*. *y*∈*T* ∧ *y*≠*x*))

lemma *islimptI*:
assumes $\bigwedge T.\ x \in T \Longrightarrow open\ T \Longrightarrow \exists y{\in}S.\ y \in T \wedge y \neq x$
shows *x islimpt S*
⟨*proof*⟩

lemma *islimptE*:
assumes *x islimpt S* **and** $x \in T$ **and** *open T*
obtains *y* **where** $y \in S$ **and** $y \in T$ **and** $y \neq x$
⟨*proof*⟩

lemma *islimpt-iff-eventually*: *x islimpt S* ⟷ ¬ *eventually* (λ*y*. *y* ∉ *S*) (*at x*)
⟨*proof*⟩

lemma *islimpt-subset*: *x islimpt S* ⟹ $S \subseteq T$ ⟹ *x islimpt T*
⟨*proof*⟩

lemma *islimpt-approachable*:

fixes x :: $'a$::*metric-space*
shows x *islimpt* $S \longleftrightarrow (\forall\, e{>}0.\ \exists\, x'{\in}S.\ x' \neq x \wedge$ *dist* $x'\ x < e)$
⟨*proof*⟩

lemma *islimpt-approachable-le*:
fixes x :: $'a$::*metric-space*
shows x *islimpt* $S \longleftrightarrow (\forall\, e{>}0.\ \exists\, x'{\in}\ S.\ x' \neq x \wedge$ *dist* $x'\ x \leq e)$
⟨*proof*⟩

lemma *islimpt-UNIV-iff*: x *islimpt UNIV* $\longleftrightarrow \neg$ *open* $\{x\}$
⟨*proof*⟩

lemma *islimpt-punctured*: x *islimpt* $S = x$ *islimpt* $(S{-}\{x\})$
⟨*proof*⟩

A perfect space has no isolated points.

lemma *islimpt-UNIV* [*simp*, *intro*]: $(x{::}'a{::}$*perfect-space*$)$ *islimpt UNIV*
⟨*proof*⟩

lemma *perfect-choose-dist*:
fixes x :: $'a{::}\{$*perfect-space*, *metric-space*$\}$
shows $0 < r \Longrightarrow \exists\, a.\ a \neq x \wedge$ *dist* $a\ x < r$
⟨*proof*⟩

lemma *closed-limpt*: *closed* $S \longleftrightarrow (\forall\, x.\ x$ *islimpt* $S \longrightarrow x \in S)$
⟨*proof*⟩

lemma *islimpt-EMPTY*[*simp*]: $\neg\ x$ *islimpt* $\{\}$
⟨*proof*⟩

lemma *finite-set-avoid*:
fixes a :: $'a$::*metric-space*
assumes *fS*: *finite* S
shows $\exists\, d{>}0.\ \forall\, x{\in}S.\ x \neq a \longrightarrow d \leq$ *dist* $a\ x$
⟨*proof*⟩

lemma *islimpt-Un*: x *islimpt* $(S \cup T) \longleftrightarrow x$ *islimpt* $S \vee x$ *islimpt* T
⟨*proof*⟩

lemma *discrete-imp-closed*:
fixes S :: $'a$::*metric-space set*
assumes e: $0 < e$
and d: $\forall\, x \in S.\ \forall\, y \in S.$ *dist* $y\ x < e \longrightarrow y = x$
shows *closed* S
⟨*proof*⟩

lemma *closed-of-nat-image*: *closed* (*of-nat* ' A :: $'a$ :: *real-normed-algebra-1 set*)
⟨*proof*⟩

lemma *closed-of-int-image*: *closed* (*of-int* ' *A* :: *'a* :: *real-normed-algebra-1 set*)
⟨*proof*⟩

lemma *closed-Nats* [*simp*]: *closed* (ℕ :: *'a* :: *real-normed-algebra-1 set*)
⟨*proof*⟩

lemma *closed-Ints* [*simp*]: *closed* (ℤ :: *'a* :: *real-normed-algebra-1 set*)
⟨*proof*⟩

16.9 Interior of a Set

definition *interior S* = ⋃{*T*. *open T* ∧ *T* ⊆ *S*}

lemma *interiorI* [*intro?*]:
assumes *open T* **and** *x* ∈ *T* **and** *T* ⊆ *S*
shows *x* ∈ *interior S*
⟨*proof*⟩

lemma *interiorE* [*elim?*]:
assumes *x* ∈ *interior S*
obtains *T* **where** *open T* **and** *x* ∈ *T* **and** *T* ⊆ *S*
⟨*proof*⟩

lemma *open-interior* [*simp*, *intro*]: *open* (*interior S*)
⟨*proof*⟩

lemma *interior-subset*: *interior S* ⊆ *S*
⟨*proof*⟩

lemma *interior-maximal*: *T* ⊆ *S* ⟹ *open T* ⟹ *T* ⊆ *interior S*
⟨*proof*⟩

lemma *interior-open*: *open S* ⟹ *interior S* = *S*
⟨*proof*⟩

lemma *interior-eq*: *interior S* = *S* ⟷ *open S*
⟨*proof*⟩

lemma *open-subset-interior*: *open S* ⟹ *S* ⊆ *interior T* ⟷ *S* ⊆ *T*
⟨*proof*⟩

lemma *interior-empty* [*simp*]: *interior* {} = {}
⟨*proof*⟩

lemma *interior-UNIV* [*simp*]: *interior UNIV* = *UNIV*
⟨*proof*⟩

lemma *interior-interior* [*simp*]: *interior* (*interior S*) = *interior S*
⟨*proof*⟩

lemma *interior-mono*: $S \subseteq T \Longrightarrow$ *interior* $S \subseteq$ *interior* T
⟨*proof*⟩

lemma *interior-unique*:
assumes $T \subseteq S$ **and** *open* T
assumes $\bigwedge T'.\ T' \subseteq S \Longrightarrow$ *open* $T' \Longrightarrow T' \subseteq T$
shows *interior* $S = T$
⟨*proof*⟩

lemma *interior-singleton* [*simp*]:
fixes a :: *'a*::*perfect-space* **shows** *interior* $\{a\} = \{\}$
⟨*proof*⟩

lemma *interior-Int* [*simp*]: *interior* $(S \cap T) =$ *interior* $S \cap$ *interior* T
⟨*proof*⟩

lemma *mem-interior*: $x \in$ *interior* $S \longleftrightarrow (\exists\, e{>}0.$ *ball* $x\ e \subseteq S)$
⟨*proof*⟩

lemma *eventually-nhds-in-nhd*: $x \in$ *interior* $s \Longrightarrow$ *eventually* $(\lambda y.\ y \in s)$ (*nhds* x)
⟨*proof*⟩

lemma *interior-limit-point* [*intro*]:
fixes x :: *'a*::*perfect-space*
assumes x: $x \in$ *interior* S
shows x *islimpt* S
⟨*proof*⟩

lemma *interior-closed-Un-empty-interior*:
assumes *cS*: *closed* S
and *iT*: *interior* $T = \{\}$
shows *interior* $(S \cup T) =$ *interior* S
⟨*proof*⟩

lemma *interior-Times*: *interior* $(A \times B) =$ *interior* $A \times$ *interior* B
⟨*proof*⟩

lemma *interior-Ici*:
fixes x :: *'a* :: {*dense-linorder*, *linorder-topology*}
assumes $b < x$
shows *interior* $\{\ x\ ..\} = \{\ x\ <..\}$
⟨*proof*⟩

lemma *interior-Iic*:
fixes x :: *'a* :: {*dense-linorder*, *linorder-topology*}
assumes $x < b$
shows *interior* $\{..\ x\} = \{..<\ x\}$

⟨*proof*⟩

16.10 Closure of a Set

definition *closure S* = *S* ∪ {*x* | *x*. *x islimpt S*}

lemma *interior-closure*: *interior S* = − (*closure* (− *S*))
⟨*proof*⟩

lemma *closure-interior*: *closure S* = − *interior* (− *S*)
⟨*proof*⟩

lemma *closed-closure*[*simp*, *intro*]: *closed* (*closure S*)
⟨*proof*⟩

lemma *closure-subset*: *S* ⊆ *closure S*
⟨*proof*⟩

lemma *closure-hull*: *closure S* = *closed hull S*
⟨*proof*⟩

lemma *closure-eq*: *closure S* = *S* ⟷ *closed S*
⟨*proof*⟩

lemma *closure-closed* [*simp*]: *closed S* ⟹ *closure S* = *S*
⟨*proof*⟩

lemma *closure-closure* [*simp*]: *closure* (*closure S*) = *closure S*
⟨*proof*⟩

lemma *closure-mono*: *S* ⊆ *T* ⟹ *closure S* ⊆ *closure T*
⟨*proof*⟩

lemma *closure-minimal*: *S* ⊆ *T* ⟹ *closed T* ⟹ *closure S* ⊆ *T*
⟨*proof*⟩

lemma *closure-unique*:
assumes *S* ⊆ *T*
and *closed T*
and ⋀*T'*. *S* ⊆ *T'* ⟹ *closed T'* ⟹ *T* ⊆ *T'*
shows *closure S* = *T*
⟨*proof*⟩

lemma *closure-empty* [*simp*]: *closure* {} = {}
⟨*proof*⟩

lemma *closure-UNIV* [*simp*]: *closure UNIV* = *UNIV*
⟨*proof*⟩

lemma *closure-union* [*simp*]: *closure* $(S \cup T) = \mathit{closure}\ S \cup \mathit{closure}\ T$
⟨*proof*⟩

lemma *closure-eq-empty* [*iff*]: *closure* $S = \{\} \longleftrightarrow S = \{\}$
⟨*proof*⟩

lemma *closure-subset-eq*: *closure* $S \subseteq S \longleftrightarrow \mathit{closed}\ S$
⟨*proof*⟩

lemma *open-Int-closure-eq-empty*:
open $S \Longrightarrow (S \cap \mathit{closure}\ T) = \{\} \longleftrightarrow S \cap T = \{\}$
⟨*proof*⟩

lemma *open-inter-closure-subset*:
open $S \Longrightarrow (S \cap (\mathit{closure}\ T)) \subseteq \mathit{closure}(S \cap T)$
⟨*proof*⟩

lemma *closure-complement*: *closure* $(-S) = -\ \mathit{interior}\ S$
⟨*proof*⟩

lemma *interior-complement*: *interior* $(-S) = -\ \mathit{closure}\ S$
⟨*proof*⟩

lemma *closure-Times*: *closure* $(A \times B) = \mathit{closure}\ A \times \mathit{closure}\ B$
⟨*proof*⟩

lemma *islimpt-in-closure*: $(x\ \mathit{islimpt}\ S) = (x{:}\mathit{closure}(S - \{x\}))$
⟨*proof*⟩

lemma *connected-imp-connected-closure*: *connected* $s \Longrightarrow \mathit{connected}\ (\mathit{closure}\ s)$
⟨*proof*⟩

lemma *limpt-of-limpts*:
fixes $x :: 'a{::}\mathit{metric\text{-}space}$
shows $x\ \mathit{islimpt}\ \{y.\ y\ \mathit{islimpt}\ s\} \Longrightarrow x\ \mathit{islimpt}\ s$
⟨*proof*⟩

lemma *closed-limpts*: *closed* $\{x{::}'a{::}\mathit{metric\text{-}space}.\ x\ \mathit{islimpt}\ s\}$
⟨*proof*⟩

lemma *limpt-of-closure*:
fixes $x :: 'a{::}\mathit{metric\text{-}space}$
shows $x\ \mathit{islimpt}\ \mathit{closure}\ s \longleftrightarrow x\ \mathit{islimpt}\ s$
⟨*proof*⟩

lemma *closedin-limpt*:
closedin $(\mathit{subtopology}\ \mathit{euclidean}\ t)\ s \longleftrightarrow s \subseteq t \wedge (\forall x.\ x\ \mathit{islimpt}\ s \wedge x \in t \longrightarrow x \in s)$
⟨*proof*⟩

lemma *closedin-closed-eq*:
closed s $\Longrightarrow$ (*closedin* (*subtopology euclidean s*) *t* $\longleftrightarrow$ *closed t* $\wedge$ *t* $\subseteq$ *s*)
⟨*proof*⟩

lemma *bdd-below-closure*:
fixes *A* :: *real set*
assumes *bdd-below A*
shows *bdd-below* (*closure A*)
⟨*proof*⟩

16.11 Connected components, considered as a connectedness relation or a set

definition
connected-component s x y $\equiv \exists t.$ *connected t* $\wedge$ *t* $\subseteq$ *s* $\wedge$ *x* $\in$ *t* $\wedge$ *y* $\in$ *t*

abbreviation
connected-component-set s x $\equiv$ *Collect* (*connected-component s x*)

lemma *connected-componentI*:
⟦*connected t*; *t* $\subseteq$ *s*; *x* $\in$ *t*; *y* $\in$ *t*⟧ $\Longrightarrow$ *connected-component s x y*
⟨*proof*⟩

lemma *connected-component-in*: *connected-component s x y* $\Longrightarrow$ *x* $\in$ *s* $\wedge$ *y* $\in$ *s*
⟨*proof*⟩

lemma *connected-component-refl*: *x* $\in$ *s* $\Longrightarrow$ *connected-component s x x*
⟨*proof*⟩

lemma *connected-component-refl-eq* [*simp*]: *connected-component s x x* $\longleftrightarrow$ *x* $\in$ *s*
⟨*proof*⟩

lemma *connected-component-sym*: *connected-component s x y* $\Longrightarrow$ *connected-component s y x*
⟨*proof*⟩

lemma *connected-component-trans*:
⟦*connected-component s x y*; *connected-component s y z*⟧ $\Longrightarrow$ *connected-component s x z*
⟨*proof*⟩

lemma *connected-component-of-subset*: ⟦*connected-component s x y*; *s* $\subseteq$ *t*⟧ $\Longrightarrow$ *connected-component t x y*
⟨*proof*⟩

lemma *connected-component-Union*: *connected-component-set s x* = $\bigcup$ {*t*. *connected t* $\wedge$ *x* $\in$ *t* $\wedge$ *t* $\subseteq$ *s*}
⟨*proof*⟩

lemma *connected-connected-component* [*iff*]: *connected* (*connected-component-set s x*)
⟨*proof*⟩

lemma *connected-iff-eq-connected-component-set*: *connected s* ⟷ (∀ *x* ∈ *s*. *connected-component-set s x* = *s*)
⟨*proof*⟩

lemma *connected-component-subset*: *connected-component-set s x* ⊆ *s*
⟨*proof*⟩

lemma *connected-component-eq-self*: ⟦*connected s*; *x* ∈ *s*⟧ ⟹ *connected-component-set s x* = *s*
⟨*proof*⟩

lemma *connected-iff-connected-component*:
connected s ⟷ (∀ *x* ∈ *s*. ∀ *y* ∈ *s*. *connected-component s x y*)
⟨*proof*⟩

lemma *connected-component-maximal*:
⟦*x* ∈ *t*; *connected t*; *t* ⊆ *s*⟧ ⟹ *t* ⊆ (*connected-component-set s x*)
⟨*proof*⟩

lemma *connected-component-mono*:
s ⊆ *t* ⟹ (*connected-component-set s x*) ⊆ (*connected-component-set t x*)
⟨*proof*⟩

lemma *connected-component-eq-empty* [*simp*]: *connected-component-set s x* = {} ⟷ (*x* ∉ *s*)
⟨*proof*⟩

lemma *connected-component-set-empty* [*simp*]: *connected-component-set* {} *x* = {}
⟨*proof*⟩

lemma *connected-component-eq*:
y ∈ *connected-component-set s x*
⟹ (*connected-component-set s y* = *connected-component-set s x*)
⟨*proof*⟩

lemma *closed-connected-component*:
assumes *s*: *closed s* **shows** *closed* (*connected-component-set s x*)
⟨*proof*⟩

lemma *connected-component-disjoint*:
(*connected-component-set s a*) ∩ (*connected-component-set s b*) = {} ⟷
a ∉ *connected-component-set s b*
⟨*proof*⟩

lemma *connected-component-nonoverlap*:
(*connected-component-set s a*) ∩ (*connected-component-set s b*) = {} ⟷
(*a* ∉ *s* ∨ *b* ∉ *s* ∨ *connected-component-set s a* ≠ *connected-component-set s b*)
⟨*proof*⟩

lemma *connected-component-overlap*:
(*connected-component-set s a* ∩ *connected-component-set s b* ≠ {}) =
(*a* ∈ *s* ∧ *b* ∈ *s* ∧ *connected-component-set s a* = *connected-component-set s b*)
⟨*proof*⟩

lemma *connected-component-sym-eq*: *connected-component s x y* ⟷ *connected-component s y x*
⟨*proof*⟩

lemma *connected-component-eq-eq*:
connected-component-set s x = *connected-component-set s y* ⟷
x ∉ *s* ∧ *y* ∉ *s* ∨ *x* ∈ *s* ∧ *y* ∈ *s* ∧ *connected-component s x y*
⟨*proof*⟩

lemma *connected-iff-connected-component-eq*:
connected s ⟷
(∀ *x* ∈ *s*. ∀ *y* ∈ *s*. *connected-component-set s x* = *connected-component-set s y*)
⟨*proof*⟩

lemma *connected-component-idemp*:
connected-component-set (*connected-component-set s x*) *x* = *connected-component-set s x*
⟨*proof*⟩

lemma *connected-component-unique*:
⟦*x* ∈ *c*; *c* ⊆ *s*; *connected c*;
⋀*c'*. *x* ∈ *c'* ∧ *c'* ⊆ *s* ∧ *connected c'*
⟹ *c'* ⊆ *c*⟧
⟹ *connected-component-set s x* = *c*
⟨*proof*⟩

lemma *joinable-connected-component-eq*:
⟦*connected t*; *t* ⊆ *s*;
connected-component-set s x ∩ *t* ≠ {};
connected-component-set s y ∩ *t* ≠ {}⟧
⟹ *connected-component-set s x* = *connected-component-set s y*
⟨*proof*⟩

lemma *Union-connected-component*: ⋃(*connected-component-set s* ` *s*) = *s*
⟨*proof*⟩

lemma *complement-connected-component-unions*:
s − connected-component-set s x =
⋃(*connected-component-set s ' s − {connected-component-set s x}*)
⟨*proof*⟩

lemma *connected-component-intermediate-subset*:
⟦*connected-component-set u a* ⊆ *t*; *t* ⊆ *u*⟧
⟹ *connected-component-set t a* = *connected-component-set u a*
⟨*proof*⟩

16.12 The set of connected components of a set

definition *components*:: *'a*::*topological-space set* ⇒ *'a set set* **where**
components s ≡ *connected-component-set s ' s*

lemma *components-iff*: *s* ∈ *components u* ⟷ (∃ *x*. *x* ∈ *u* ∧ *s* = *connected-component-set u x*)
⟨*proof*⟩

lemma *Union-components* [*simp*]: ⋃(*components u*) = *u*
⟨*proof*⟩

lemma *pairwise-disjoint-components*: *pairwise* (λ*X Y*. *X* ∩ *Y* = {}) (*components u*)
⟨*proof*⟩

lemma *in-components-nonempty*: *c* ∈ *components s* ⟹ *c* ≠ {}
⟨*proof*⟩

lemma *in-components-subset*: *c* ∈ *components s* ⟹ *c* ⊆ *s*
⟨*proof*⟩

lemma *in-components-connected*: *c* ∈ *components s* ⟹ *connected c*
⟨*proof*⟩

lemma *in-components-maximal*:
c ∈ *components s* ⟷
(*c* ≠ {} ∧ *c* ⊆ *s* ∧ *connected c* ∧ (∀ *d*. *d* ≠ {} ∧ *c* ⊆ *d* ∧ *d* ⊆ *s* ∧ *connected d* ⟶ *d* = *c*))
⟨*proof*⟩

lemma *joinable-components-eq*:
connected t ∧ *t* ⊆ *s* ∧ *c1* ∈ *components s* ∧ *c2* ∈ *components s* ∧ *c1* ∩ *t* ≠ {} ∧ *c2* ∩ *t* ≠ {} ⟹ *c1* = *c2*
⟨*proof*⟩

lemma *closed-components*: ⟦*closed s*; *c* ∈ *components s*⟧ ⟹ *closed c*
⟨*proof*⟩

lemma *components-nonoverlap*:
$[\![c \in \mathit{components}\ s;\ c' \in \mathit{components}\ s]\!] \Longrightarrow (c \cap c' = \{\}) \longleftrightarrow (c \neq c')$
⟨*proof*⟩

lemma *components-eq*: $[\![c \in \mathit{components}\ s;\ c' \in \mathit{components}\ s]\!] \Longrightarrow (c = c' \longleftrightarrow c \cap c' \neq \{\})$
⟨*proof*⟩

lemma *components-eq-empty* [*simp*]: $\mathit{components}\ s = \{\} \longleftrightarrow s = \{\}$
⟨*proof*⟩

lemma *components-empty* [*simp*]: $\mathit{components}\ \{\} = \{\}$
⟨*proof*⟩

lemma *connected-eq-connected-components-eq*: $\mathit{connected}\ s \longleftrightarrow (\forall\, c \in \mathit{components}\ s.\ \forall\, c' \in \mathit{components}\ s.\ c = c')$
⟨*proof*⟩

lemma *components-eq-sing-iff*: $\mathit{components}\ s = \{s\} \longleftrightarrow \mathit{connected}\ s \wedge s \neq \{\}$
⟨*proof*⟩

lemma *components-eq-sing-exists*: $(\exists\, a.\ \mathit{components}\ s = \{a\}) \longleftrightarrow \mathit{connected}\ s \wedge s \neq \{\}$
⟨*proof*⟩

lemma *connected-eq-components-subset-sing*: $\mathit{connected}\ s \longleftrightarrow \mathit{components}\ s \subseteq \{s\}$
⟨*proof*⟩

lemma *connected-eq-components-subset-sing-exists*: $\mathit{connected}\ s \longleftrightarrow (\exists\, a.\ \mathit{components}\ s \subseteq \{a\})$
⟨*proof*⟩

lemma *in-components-self*: $s \in \mathit{components}\ s \longleftrightarrow \mathit{connected}\ s \wedge s \neq \{\}$
⟨*proof*⟩

lemma *components-maximal*: $[\![c \in \mathit{components}\ s;\ \mathit{connected}\ t;\ t \subseteq s;\ c \cap t \neq \{\}]\!] \Longrightarrow t \subseteq c$
⟨*proof*⟩

lemma *exists-component-superset*: $[\![t \subseteq s;\ s \neq \{\};\ \mathit{connected}\ t]\!] \Longrightarrow \exists\, c.\ c \in \mathit{components}\ s \wedge t \subseteq c$
⟨*proof*⟩

lemma *components-intermediate-subset*: $[\![s \in \mathit{components}\ u;\ s \subseteq t;\ t \subseteq u]\!] \Longrightarrow s \in \mathit{components}\ t$
⟨*proof*⟩

lemma *in-components-unions-complement*: $c \in \mathit{components}\ s \Longrightarrow s - c = \bigcup(\mathit{components}$

$s - \{c\}$)
⟨*proof*⟩

lemma *connected-intermediate-closure*:
assumes *cs*: *connected s* **and** *st*: $s \subseteq t$ **and** *ts*: $t \subseteq$ *closure s*
shows *connected t*
⟨*proof*⟩

lemma *closedin-connected-component*: *closedin* (*subtopology euclidean s*) (*connected-component-set s x*)
⟨*proof*⟩

16.13 Frontier (aka boundary)

definition *frontier S* = *closure S* − *interior S*

lemma *frontier-closed* [*iff*]: *closed* (*frontier S*)
⟨*proof*⟩

lemma *frontier-closures*: *frontier S* = (*closure S*) ∩ (*closure*(− *S*))
⟨*proof*⟩

lemma *frontier-straddle*:
fixes *a* :: *'a*::*metric-space*
shows *a* ∈ *frontier S* ⟷ (∀ *e*>0. (∃ *x*∈*S*. *dist a x* < *e*) ∧ (∃ *x*. *x* ∉ *S* ∧ *dist a x* < *e*))
⟨*proof*⟩

lemma *frontier-subset-closed*: *closed S* ⟹ *frontier S* ⊆ *S*
⟨*proof*⟩

lemma *frontier-empty* [*simp*]: *frontier* {} = {}
⟨*proof*⟩

lemma *frontier-subset-eq*: *frontier S* ⊆ *S* ⟷ *closed S*
⟨*proof*⟩

lemma *frontier-complement* [*simp*]: *frontier* (− *S*) = *frontier S*
⟨*proof*⟩

lemma *frontier-disjoint-eq*: *frontier S* ∩ *S* = {} ⟷ *open S*
⟨*proof*⟩

lemma *frontier-UNIV* [*simp*]: *frontier UNIV* = {}
⟨*proof*⟩

16.14 Filters and the "eventually true" quantifier

definition *indirection* :: *'a*::*real-normed-vector* ⇒ *'a* ⇒ *'a filter*
(**infixr** *indirection 70*)

where *a indirection v = at a within {b. ∃ c≥0. b − a = scaleR c v}*

Identify Trivial limits, where we can't approach arbitrarily closely.

lemma *trivial-limit-within*: *trivial-limit (at a within S) ⟷ ¬ a islimpt S*
⟨*proof*⟩

lemma *trivial-limit-at-iff*: *trivial-limit (at a) ⟷ ¬ a islimpt UNIV*
⟨*proof*⟩

lemma *trivial-limit-at*:
fixes *a* :: *'a*::*perfect-space*
shows ¬ *trivial-limit (at a)*
⟨*proof*⟩

lemma *trivial-limit-at-infinity*:
¬ *trivial-limit (at-infinity* :: (*'a*::{*real-normed-vector*,*perfect-space*}) *filter*)
⟨*proof*⟩

lemma *not-trivial-limit-within*: ¬ *trivial-limit (at x within S) = (x ∈ closure (S − {x}))*
⟨*proof*⟩

lemma *at-within-eq-bot-iff*: *(at c within A = bot) ⟷ (c ∉ closure (A − {c}))*
⟨*proof*⟩

Some property holds "sufficiently close" to the limit point.

lemma *trivial-limit-eventually*: *trivial-limit net ⟹ eventually P net*
⟨*proof*⟩

lemma *trivial-limit-eq*: *trivial-limit net ⟷ (∀ P. eventually P net)*
⟨*proof*⟩

16.15 Limits

lemma *Lim*:
(f ⟶ l) net ⟷
trivial-limit net ∨
(∀ e>0. eventually (λx. dist (f x) l < e) net)
⟨*proof*⟩

Show that they yield usual definitions in the various cases.

lemma *Lim-within-le*: *(f ⟶ l)(at a within S) ⟷*
(∀ e>0. ∃ d>0. ∀ x∈S. 0 < dist x a ∧ dist x a ≤ d ⟶ dist (f x) l < e)
⟨*proof*⟩

lemma *Lim-within*: *(f ⟶ l) (at a within S) ⟷*
(∀ e >0. ∃ d>0. ∀ x ∈ S. 0 < dist x a ∧ dist x a < d ⟶ dist (f x) l < e)
⟨*proof*⟩

corollary *Lim-withinI* [*intro?*]:
assumes $\bigwedge e.\ e > 0 \Longrightarrow \exists\, d{>}0.\ \forall\, x \in S.\ 0 < dist\ x\ a \wedge dist\ x\ a < d \longrightarrow dist\ (f\ x)\ l \leq e$
shows $(f \longrightarrow l)\ (at\ a\ within\ S)$
⟨*proof*⟩

lemma *Lim-at*: $(f \longrightarrow l)\ (at\ a) \longleftrightarrow$
$(\forall\, e > 0.\ \exists\, d{>}0.\ \forall\, x.\ 0 < dist\ x\ a \wedge dist\ x\ a < d \longrightarrow dist\ (f\ x)\ l < e)$
⟨*proof*⟩

lemma *Lim-at-infinity*:
$(f \longrightarrow l)\ at\text{-}infinity \longleftrightarrow (\forall\, e{>}0.\ \exists\, b.\ \forall\, x.\ norm\ x \geq b \longrightarrow dist\ (f\ x)\ l < e)$
⟨*proof*⟩

corollary *Lim-at-infinityI* [*intro?*]:
assumes $\bigwedge e.\ e > 0 \Longrightarrow \exists\, B.\ \forall\, x.\ norm\ x \geq B \longrightarrow dist\ (f\ x)\ l \leq e$
shows $(f \longrightarrow l)\ at\text{-}infinity$
⟨*proof*⟩

lemma *Lim-eventually*: $eventually\ (\lambda x.\ f\ x = l)\ net \Longrightarrow (f \longrightarrow l)\ net$
⟨*proof*⟩

lemma *Lim-transform-within-set*:
fixes $a\ l :: 'a{::}real\text{-}normed\text{-}vector$
shows $[\![(f \longrightarrow l)\ (at\ a\ within\ s);\ eventually\ (\lambda x.\ x \in s \longleftrightarrow x \in t)\ (at\ a)]\!]$
$\Longrightarrow (f \longrightarrow l)\ (at\ a\ within\ t)$
⟨*proof*⟩

lemma *Lim-transform-within-set-eq*:
fixes $a\ l :: 'a{::}real\text{-}normed\text{-}vector$
shows $eventually\ (\lambda x.\ x \in s \longleftrightarrow x \in t)\ (at\ a)$
$\Longrightarrow ((f \longrightarrow l)\ (at\ a\ within\ s) \longleftrightarrow (f \longrightarrow l)\ (at\ a\ within\ t))$
⟨*proof*⟩

The expected monotonicity property.

lemma *Lim-Un*:
assumes $(f \longrightarrow l)\ (at\ x\ within\ S)\ (f \longrightarrow l)\ (at\ x\ within\ T)$
shows $(f \longrightarrow l)\ (at\ x\ within\ (S \cup T))$
⟨*proof*⟩

lemma *Lim-Un-univ*:
$(f \longrightarrow l)\ (at\ x\ within\ S) \Longrightarrow (f \longrightarrow l)\ (at\ x\ within\ T) \Longrightarrow$
$S \cup T = UNIV \Longrightarrow (f \longrightarrow l)\ (at\ x)$
⟨*proof*⟩

Interrelations between restricted and unrestricted limits.

lemma *Lim-at-imp-Lim-at-within*:
$(f \longrightarrow l)\ (at\ x) \Longrightarrow (f \longrightarrow l)\ (at\ x\ within\ S)$
⟨*proof*⟩

lemma *eventually-within-interior*:
assumes $x \in \mathit{interior}\ S$
shows $\mathit{eventually}\ P\ (\mathit{at}\ x\ \mathit{within}\ S) \longleftrightarrow \mathit{eventually}\ P\ (\mathit{at}\ x)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *at-within-interior*:
$x \in \mathit{interior}\ S \Longrightarrow \mathit{at}\ x\ \mathit{within}\ S = \mathit{at}\ x$
⟨*proof*⟩

lemma *Lim-within-LIMSEQ*:
fixes $a :: 'a$::*first-countable-topology*
assumes $\forall S.\ (\forall n.\ S\ n \neq a \wedge S\ n \in T) \wedge S \longrightarrow a \longrightarrow (\lambda n.\ X\ (S\ n)) \longrightarrow L$
shows $(X \longrightarrow L)\ (\mathit{at}\ a\ \mathit{within}\ T)$
⟨*proof*⟩

lemma *Lim-right-bound*:
fixes $f :: 'a :: \{\mathit{linorder\text{-}topology}, \mathit{conditionally\text{-}complete\text{-}linorder}, \mathit{no\text{-}top}\} \Rightarrow 'b$::$\{\mathit{linorder\text{-}topology}, \mathit{conditionally\text{-}complete\text{-}linorder}\}$
assumes *mono*: $\bigwedge a\ b.\ a \in I \Longrightarrow b \in I \Longrightarrow x < a \Longrightarrow a \leq b \Longrightarrow f\ a \leq f\ b$
and *bnd*: $\bigwedge a.\ a \in I \Longrightarrow x < a \Longrightarrow K \leq f\ a$
shows $(f \longrightarrow \mathit{Inf}\ (f\ `\ (\{x<..\} \cap I)))\ (\mathit{at}\ x\ \mathit{within}\ (\{x<..\} \cap I))$
⟨*proof*⟩

Another limit point characterization.

lemma *islimpt-sequential*:
fixes $x :: 'a$::*first-countable-topology*
shows $x\ \mathit{islimpt}\ S \longleftrightarrow (\exists f.\ (\forall n$::$\mathit{nat}.\ f\ n \in S - \{x\}) \wedge (f \longrightarrow x)\ \mathit{sequentially})$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *Lim-null*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
shows $(f \longrightarrow l)\ \mathit{net} \longleftrightarrow ((\lambda x.\ f(x) - l) \longrightarrow 0)\ \mathit{net}$
⟨*proof*⟩

lemma *Lim-null-comparison*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
assumes $\mathit{eventually}\ (\lambda x.\ \mathit{norm}\ (f\ x) \leq g\ x)\ \mathit{net}\ (g \longrightarrow 0)\ \mathit{net}$
shows $(f \longrightarrow 0)\ \mathit{net}$
⟨*proof*⟩

lemma *Lim-transform-bound*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
and $g :: 'a \Rightarrow 'c$::*real-normed-vector*
assumes $\mathit{eventually}\ (\lambda n.\ \mathit{norm}\ (f\ n) \leq \mathit{norm}\ (g\ n))\ \mathit{net}$
and $(g \longrightarrow 0)\ \mathit{net}$

shows $(f \longrightarrow 0)$ *net*
⟨*proof*⟩

lemma *lim-null-mult-right-bounded*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-div-algebra*
assumes *f*: $(f \longrightarrow 0)$ *F* **and** *g*: *eventually* $(\lambda x.\ norm(g\ x) \leq B)$ *F*
shows $((\lambda z.\ f\ z * g\ z) \longrightarrow 0)$ *F*
⟨*proof*⟩

lemma *lim-null-mult-left-bounded*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-div-algebra*
assumes *g*: *eventually* $(\lambda x.\ norm(g\ x) \leq B)$ *F* **and** *f*: $(f \longrightarrow 0)$ *F*
shows $((\lambda z.\ g\ z * f\ z) \longrightarrow 0)$ *F*
⟨*proof*⟩

Deducing things about the limit from the elements.

lemma *Lim-in-closed-set*:
assumes *closed S*
and *eventually* $(\lambda x.\ f(x) \in S)$ *net*
and ¬ *trivial-limit net* $(f \longrightarrow l)$ *net*
shows $l \in S$
⟨*proof*⟩

Need to prove closed(cball(x,e)) before deducing this as a corollary.

lemma *Lim-dist-ubound*:
assumes ¬(*trivial-limit net*)
and $(f \longrightarrow l)$ *net*
and *eventually* $(\lambda x.\ dist\ a\ (f\ x) \leq e)$ *net*
shows $dist\ a\ l \leq e$
⟨*proof*⟩

lemma *Lim-norm-ubound*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
assumes ¬(*trivial-limit net*) $(f \longrightarrow l)$ *net eventually* $(\lambda x.\ norm(f\ x) \leq e)$ *net*
shows $norm(l) \leq e$
⟨*proof*⟩

lemma *Lim-norm-lbound*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
assumes ¬ *trivial-limit net*
and $(f \longrightarrow l)$ *net*
and *eventually* $(\lambda x.\ e \leq norm\ (f\ x))$ *net*
shows $e \leq norm\ l$
⟨*proof*⟩

Limit under bilinear function

lemma *Lim-bilinear*:
assumes $(f \longrightarrow l)$ *net*
and $(g \longrightarrow m)$ *net*

and *bounded-bilinear h*
shows $((\lambda x.\ h\ (f\ x)\ (g\ x)) \longrightarrow (h\ l\ m))\ net$
⟨*proof*⟩

These are special for limits out of the same vector space.

lemma *Lim-within-id*: $(id \longrightarrow a)\ (at\ a\ within\ s)$
⟨*proof*⟩

lemma *Lim-at-id*: $(id \longrightarrow a)\ (at\ a)$
⟨*proof*⟩

lemma *Lim-at-zero*:
fixes $a :: 'a$::*real-normed-vector*
and $l :: 'b$::*topological-space*
shows $(f \longrightarrow l)\ (at\ a) \longleftrightarrow ((\lambda x.\ f(a + x)) \longrightarrow l)\ (at\ 0)$
⟨*proof*⟩

It's also sometimes useful to extract the limit point from the filter.

abbreviation *netlimit* :: $'a$::*t2-space filter* ⇒ $'a$
where $netlimit\ F \equiv Lim\ F\ (\lambda x.\ x)$

lemma *netlimit-within*: ¬ *trivial-limit* $(at\ a\ within\ S) \Longrightarrow netlimit\ (at\ a\ within\ S) = a$
⟨*proof*⟩

lemma *netlimit-at*:
fixes $a :: 'a$::{*perfect-space*,*t2-space*}
shows $netlimit\ (at\ a) = a$
⟨*proof*⟩

lemma *lim-within-interior*:
$x \in interior\ S \Longrightarrow (f \longrightarrow l)\ (at\ x\ within\ S) \longleftrightarrow (f \longrightarrow l)\ (at\ x)$
⟨*proof*⟩

lemma *netlimit-within-interior*:
fixes $x :: 'a$::{*t2-space*,*perfect-space*}
assumes $x \in interior\ S$
shows $netlimit\ (at\ x\ within\ S) = x$
⟨*proof*⟩

lemma *netlimit-at-vector*:
fixes $a :: 'a$::*real-normed-vector*
shows $netlimit\ (at\ a) = a$
⟨*proof*⟩

Useful lemmas on closure and set of possible sequential limits.

lemma *closure-sequential*:
fixes $l :: 'a$::*first-countable-topology*
shows $l \in closure\ S \longleftrightarrow (\exists x.\ (\forall n.\ x\ n \in S) \wedge (x \longrightarrow l)\ sequentially)$

(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *closed-sequential-limits*:
fixes *S* :: *'a*::*first-countable-topology set*
shows *closed S* ⟷ (∀ *x l*. (∀ *n*. *x n* ∈ *S*) ∧ (*x* ⟶ *l*) *sequentially* ⟶ *l* ∈ *S*)
⟨*proof*⟩

lemma *closure-approachable*:
fixes *S* :: *'a*::*metric-space set*
shows *x* ∈ *closure S* ⟷ (∀ *e*>*0*. ∃ *y*∈*S*. *dist y x* < *e*)
⟨*proof*⟩

lemma *closed-approachable*:
fixes *S* :: *'a*::*metric-space set*
shows *closed S* ⟹ (∀ *e*>*0*. ∃ *y*∈*S*. *dist y x* < *e*) ⟷ *x* ∈ *S*
⟨*proof*⟩

lemma *closure-contains-Inf*:
fixes *S* :: *real set*
assumes *S* ≠ {} *bdd-below S*
shows *Inf S* ∈ *closure S*
⟨*proof*⟩

lemma *closed-contains-Inf*:
fixes *S* :: *real set*
shows *S* ≠ {} ⟹ *bdd-below S* ⟹ *closed S* ⟹ *Inf S* ∈ *S*
⟨*proof*⟩

lemma *closed-subset-contains-Inf*:
fixes *A C* :: *real set*
shows *closed C* ⟹ *A* ⊆ *C* ⟹ *A* ≠ {} ⟹ *bdd-below A* ⟹ *Inf A* ∈ *C*
⟨*proof*⟩

lemma *atLeastAtMost-subset-contains-Inf*:
fixes *A* :: *real set* **and** *a b* :: *real*
shows *A* ≠ {} ⟹ *a* ≤ *b* ⟹ *A* ⊆ {*a*..*b*} ⟹ *Inf A* ∈ {*a*..*b*}
⟨*proof*⟩

lemma *not-trivial-limit-within-ball*:
¬ *trivial-limit* (*at x within S*) ⟷ (∀ *e*>*0*. *S* ∩ *ball x e* − {*x*} ≠ {})
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

16.16 Infimum Distance

definition *infdist x A* = (*if A* = {} *then 0 else INF a*:*A*. *dist x a*)

lemma *bdd-below-infdist*[*intro*, *simp*]: *bdd-below* (*dist x*‘*A*)

⟨*proof*⟩

lemma *infdist-notempty*: $A \neq \{\} \Longrightarrow$ *infdist x A* = (*INF a*:*A*. *dist x a*)
⟨*proof*⟩

lemma *infdist-nonneg*: $0 \leq$ *infdist x A*
⟨*proof*⟩

lemma *infdist-le*: $a \in A \Longrightarrow$ *infdist x A* $\leq$ *dist x a*
⟨*proof*⟩

lemma *infdist-le2*: $a \in A \Longrightarrow$ *dist x a* $\leq d \Longrightarrow$ *infdist x A* $\leq d$
⟨*proof*⟩

lemma *infdist-zero*[*simp*]: $a \in A \Longrightarrow$ *infdist a A* = 0
⟨*proof*⟩

lemma *infdist-triangle*: *infdist x A* $\leq$ *infdist y A* + *dist x y*
⟨*proof*⟩

lemma *in-closure-iff-infdist-zero*:
assumes $A \neq \{\}$
shows $x \in$ *closure A* $\longleftrightarrow$ *infdist x A* = 0
⟨*proof*⟩

lemma *in-closed-iff-infdist-zero*:
assumes *closed A* $A \neq \{\}$
shows $x \in A \longleftrightarrow$ *infdist x A* = 0
⟨*proof*⟩

lemma *tendsto-infdist* [*tendsto-intros*]:
assumes *f*: (*f* $\longrightarrow$ *l*) *F*
shows ((λ*x*. *infdist* (*f x*) *A*) $\longrightarrow$ *infdist l A*) *F*
⟨*proof*⟩

Some other lemmas about sequences.

lemma *sequentially-offset*:
assumes *eventually* (λ*i*. *P i*) *sequentially*
shows *eventually* (λ*i*. *P* (*i* + *k*)) *sequentially*
⟨*proof*⟩

lemma *seq-offset-neg*:
(*f* $\longrightarrow$ *l*) *sequentially* $\Longrightarrow$ ((λ*i*. *f*(*i* − *k*)) $\longrightarrow$ *l*) *sequentially*
⟨*proof*⟩

lemma *seq-harmonic*: ((λ*n*. *inverse* (*real n*)) $\longrightarrow$ 0) *sequentially*
⟨*proof*⟩

16.17 More properties of closed balls

lemma *closed-cball* [*iff*]: *closed* (*cball x e*)
⟨*proof*⟩

lemma *open-contains-cball*: *open S* ⟷ (∀ *x*∈*S*. ∃ *e*>*0*. *cball x e* ⊆ *S*)
⟨*proof*⟩

lemma *open-contains-cball-eq*: *open S* ⟹ (∀ *x*. *x* ∈ *S* ⟷ (∃ *e*>*0*. *cball x e* ⊆ *S*))
⟨*proof*⟩

lemma *mem-interior-cball*: *x* ∈ *interior S* ⟷ (∃ *e*>*0*. *cball x e* ⊆ *S*)
⟨*proof*⟩

lemma *islimpt-ball*:
fixes *x y* :: *'a*::{*real-normed-vector*,*perfect-space*}
shows *y islimpt ball x e* ⟷ *0* < *e* ∧ *y* ∈ *cball x e*
(**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *closure-ball-lemma*:
fixes *x y* :: *'a*::*real-normed-vector*
assumes *x* ≠ *y*
shows *y islimpt ball x* (*dist x y*)
⟨*proof*⟩

lemma *closure-ball* [*simp*]:
fixes *x* :: *'a*::*real-normed-vector*
shows *0* < *e* ⟹ *closure* (*ball x e*) = *cball x e*
⟨*proof*⟩

lemma *interior-cball* [*simp*]:
fixes *x* :: *'a*::{*real-normed-vector*, *perfect-space*}
shows *interior* (*cball x e*) = *ball x e*
⟨*proof*⟩

lemma *interior-ball* [*simp*]: *interior* (*ball x e*) = *ball x e*
⟨*proof*⟩

lemma *frontier-ball* [*simp*]:
fixes *a* :: *'a*::*real-normed-vector*
shows *0* < *e* ⟹ *frontier* (*ball a e*) = *sphere a e*
⟨*proof*⟩

lemma *frontier-cball* [*simp*]:
fixes *a* :: *'a*::{*real-normed-vector*, *perfect-space*}
shows *frontier* (*cball a e*) = *sphere a e*
⟨*proof*⟩

lemma *cball-eq-empty* [*simp*]: $cball\ x\ e = \{\} \longleftrightarrow e < 0$
⟨*proof*⟩

lemma *cball-empty* [*simp*]: $e < 0 \Longrightarrow cball\ x\ e = \{\}$
⟨*proof*⟩

lemma *cball-eq-sing*:
fixes $x :: 'a::\{metric\text{-}space, perfect\text{-}space\}$
shows $cball\ x\ e = \{x\} \longleftrightarrow e = 0$
⟨*proof*⟩

lemma *cball-sing*:
fixes $x :: 'a::metric\text{-}space$
shows $e = 0 \Longrightarrow cball\ x\ e = \{x\}$
⟨*proof*⟩

lemma *ball-divide-subset*: $d \geq 1 \Longrightarrow ball\ x\ (e/d) \subseteq ball\ x\ e$
⟨*proof*⟩

lemma *ball-divide-subset-numeral*: $ball\ x\ (e\ /\ numeral\ w) \subseteq ball\ x\ e$
⟨*proof*⟩

lemma *cball-divide-subset*: $d \geq 1 \Longrightarrow cball\ x\ (e/d) \subseteq cball\ x\ e$
⟨*proof*⟩

lemma *cball-divide-subset-numeral*: $cball\ x\ (e\ /\ numeral\ w) \subseteq cball\ x\ e$
⟨*proof*⟩

16.18 Boundedness

definition (**in** *metric-space*) *bounded* :: $'a\ set \Rightarrow bool$
where $bounded\ S \longleftrightarrow (\exists x\ e.\ \forall y{\in}S.\ dist\ x\ y \leq e)$

lemma *bounded-subset-cball*: $bounded\ S \longleftrightarrow (\exists e\ x.\ S \subseteq cball\ x\ e \wedge 0 \leq e)$
⟨*proof*⟩

lemma *bounded-subset-ballD*:
assumes *bounded S* **shows** $\exists r.\ 0 < r \wedge S \subseteq ball\ x\ r$
⟨*proof*⟩

lemma *bounded-any-center*: $bounded\ S \longleftrightarrow (\exists e.\ \forall y{\in}S.\ dist\ a\ y \leq e)$
⟨*proof*⟩

lemma *bounded-iff*: $bounded\ S \longleftrightarrow (\exists a.\ \forall x{\in}S.\ norm\ x \leq a)$
⟨*proof*⟩

lemma *bdd-above-norm*: $bdd\text{-}above\ (norm\ `\ X) \longleftrightarrow bounded\ X$
⟨*proof*⟩

lemma *bounded-realI*:
assumes $\forall x{\in}s.\ |x{::}real| \leq B$
shows *bounded s*
⟨*proof*⟩

lemma *bounded-empty* [*simp*]: *bounded* {}
⟨*proof*⟩

lemma *bounded-subset*: *bounded T* $\Longrightarrow$ $S \subseteq T$ $\Longrightarrow$ *bounded S*
⟨*proof*⟩

lemma *bounded-interior*[*intro*]: *bounded S* $\Longrightarrow$ *bounded*(*interior S*)
⟨*proof*⟩

lemma *bounded-closure*[*intro*]:
assumes *bounded S*
shows *bounded* (*closure S*)
⟨*proof*⟩

lemma *bounded-cball*[*simp*,*intro*]: *bounded* (*cball x e*)
⟨*proof*⟩

lemma *bounded-ball*[*simp*,*intro*]: *bounded* (*ball x e*)
⟨*proof*⟩

lemma *bounded-Un*[*simp*]: *bounded* ($S \cup T$) $\longleftrightarrow$ *bounded S* $\wedge$ *bounded T*
⟨*proof*⟩

lemma *bounded-Union*[*intro*]: *finite F* $\Longrightarrow$ $\forall S{\in}F.$ *bounded S* $\Longrightarrow$ *bounded* ($\bigcup F$)
⟨*proof*⟩

lemma *bounded-UN* [*intro*]: *finite A* $\Longrightarrow$ $\forall x{\in}A.$ *bounded* (*B x*) $\Longrightarrow$ *bounded* ($\bigcup x{\in}A.$ *B x*)
⟨*proof*⟩

lemma *bounded-insert* [*simp*]: *bounded* (*insert x S*) $\longleftrightarrow$ *bounded S*
⟨*proof*⟩

lemma *finite-imp-bounded* [*intro*]: *finite S* $\Longrightarrow$ *bounded S*
⟨*proof*⟩

lemma *bounded-pos*: *bounded S* $\longleftrightarrow$ ($\exists b{>}0.\ \forall x{\in}\ S.$ *norm x* $\leq b$)
⟨*proof*⟩

lemma *bounded-pos-less*: *bounded S* $\longleftrightarrow$ ($\exists b{>}0.\ \forall x{\in}\ S.$ *norm x* $< b$)
⟨*proof*⟩

lemma *Bseq-eq-bounded*:

fixes f :: $nat \Rightarrow 'a$::*real-normed-vector*
shows *Bseq* $f \longleftrightarrow$ *bounded* (*range* f)
⟨*proof*⟩

lemma *bounded-Int*[*intro*]: *bounded* $S \vee$ *bounded* $T \Longrightarrow$ *bounded* $(S \cap T)$
⟨*proof*⟩

lemma *bounded-diff*[*intro*]: *bounded* $S \Longrightarrow$ *bounded* $(S - T)$
⟨*proof*⟩

lemma *not-bounded-UNIV*[*simp*]:
$\neg$ *bounded* (*UNIV* :: $'a$::{*real-normed-vector*, *perfect-space*} *set*)
⟨*proof*⟩

corollary *cobounded-imp-unbounded*:
fixes S :: $'a$::{*real-normed-vector*, *perfect-space*} *set*
shows *bounded* $(-S) \Longrightarrow \sim$ (*bounded* S)
⟨*proof*⟩

lemma *bounded-linear-image*:
assumes *bounded* S
and *bounded-linear* f
shows *bounded* $(f$ ‘ $S)$
⟨*proof*⟩

lemma *bounded-scaling*:
fixes S :: $'a$::*real-normed-vector set*
shows *bounded* $S \Longrightarrow$ *bounded* $((\lambda x.\ c *_R x)$ ‘ $S)$
⟨*proof*⟩

lemma *bounded-translation*:
fixes S :: $'a$::*real-normed-vector set*
assumes *bounded* S
shows *bounded* $((\lambda x.\ a + x)$ ‘ $S)$
⟨*proof*⟩

lemma *bounded-translation-minus*:
fixes S :: $'a$::*real-normed-vector set*
shows *bounded* $S \Longrightarrow$ *bounded* $((\lambda x.\ x - a)$ ‘ $S)$
⟨*proof*⟩

lemma *bounded-uminus* [*simp*]:
fixes X :: $'a$::*real-normed-vector set*
shows *bounded* $(uminus$ ‘ $X) \longleftrightarrow$ *bounded* X
⟨*proof*⟩

Some theorems on sups and infs using the notion "bounded".

lemma *bounded-real*: *bounded* $(S$::*real set*$) \longleftrightarrow (\exists a.\ \forall x{\in}S.\ |x| \leq a)$
⟨*proof*⟩

lemma *bounded-imp-bdd-above*: *bounded S* $\Longrightarrow$ *bdd-above* (*S* :: *real set*)
⟨*proof*⟩

lemma *bounded-imp-bdd-below*: *bounded S* $\Longrightarrow$ *bdd-below* (*S* :: *real set*)
⟨*proof*⟩

lemma *bounded-inner-imp-bdd-above*:
assumes *bounded s*
shows *bdd-above* ((λ*x*. *x* • *a*) ‘ *s*)
⟨*proof*⟩

lemma *bounded-inner-imp-bdd-below*:
assumes *bounded s*
shows *bdd-below* ((λ*x*. *x* • *a*) ‘ *s*)
⟨*proof*⟩

lemma *bounded-has-Sup*:
fixes *S* :: *real set*
assumes *bounded S*
and *S* ≠ {}
shows ∀ *x*∈*S*. *x* ≤ *Sup S*
and ∀ *b*. (∀ *x*∈*S*. *x* ≤ *b*) ⟶ *Sup S* ≤ *b*
⟨*proof*⟩

lemma *Sup-insert*:
fixes *S* :: *real set*
shows *bounded S* $\Longrightarrow$ *Sup* (*insert x S*) = (*if S* = {} *then x else max x* (*Sup S*))
⟨*proof*⟩

lemma *Sup-insert-finite*:
fixes *S* :: *'a*::*conditionally-complete-linorder set*
shows *finite S* $\Longrightarrow$ *Sup* (*insert x S*) = (*if S* = {} *then x else max x* (*Sup S*))
⟨*proof*⟩

lemma *bounded-has-Inf*:
fixes *S* :: *real set*
assumes *bounded S*
and *S* ≠ {}
shows ∀ *x*∈*S*. *x* ≥ *Inf S*
and ∀ *b*. (∀ *x*∈*S*. *x* ≥ *b*) ⟶ *Inf S* ≥ *b*
⟨*proof*⟩

lemma *Inf-insert*:
fixes *S* :: *real set*
shows *bounded S* $\Longrightarrow$ *Inf* (*insert x S*) = (*if S* = {} *then x else min x* (*Inf S*))
⟨*proof*⟩

lemma *Inf-insert-finite*:

fixes $S :: 'a$::*conditionally-complete-linorder set*
shows *finite* $S \Longrightarrow$ *Inf* (*insert* x S) = (*if* $S = \{\}$ *then* x *else min* x (*Inf* S))
⟨*proof*⟩

lemma *finite-imp-less-Inf*:
fixes $a :: 'a$::*conditionally-complete-linorder*
shows $[\![$*finite* X; $x \in X$; $\bigwedge x.\ x \in X \Longrightarrow a < x]\!] \Longrightarrow a <$ *Inf* X
⟨*proof*⟩

lemma *finite-less-Inf-iff*:
fixes $a :: 'a ::$ *conditionally-complete-linorder*
shows $[\![$*finite* X; $X \neq \{\}]\!] \Longrightarrow a <$ *Inf* $X \longleftrightarrow (\forall x \in X.\ a < x)$
⟨*proof*⟩

lemma *finite-imp-Sup-less*:
fixes $a :: 'a$::*conditionally-complete-linorder*
shows $[\![$*finite* X; $x \in X$; $\bigwedge x.\ x \in X \Longrightarrow a > x]\!] \Longrightarrow a >$ *Sup* X
⟨*proof*⟩

lemma *finite-Sup-less-iff*:
fixes $a :: 'a ::$ *conditionally-complete-linorder*
shows $[\![$*finite* X; $X \neq \{\}]\!] \Longrightarrow a >$ *Sup* $X \longleftrightarrow (\forall x \in X.\ a > x)$
⟨*proof*⟩

16.19 Compactness

16.19.1 Bolzano-Weierstrass property

lemma *heine-borel-imp-bolzano-weierstrass*:
assumes *compact* s
and *infinite* t
and $t \subseteq s$
shows $\exists x \in s.$ x *islimpt* t
⟨*proof*⟩

lemma *acc-point-range-imp-convergent-subsequence*:
fixes $l :: 'a ::$ *first-countable-topology*
assumes l: $\forall U.\ l \in U \longrightarrow$ *open* $U \longrightarrow$ *infinite* $(U \cap$ *range* $f)$
shows $\exists r.$ *subseq* $r \wedge (f \circ r) \longrightarrow l$
⟨*proof*⟩

lemma *sequence-infinite-lemma*:
fixes $f ::$ *nat* $\Rightarrow 'a$::*t1-space*
assumes $\forall n.\ f\ n \neq l$
and $(f \longrightarrow l)$ *sequentially*
shows *infinite* (*range* f)
⟨*proof*⟩

lemma *closure-insert*:
fixes $x :: 'a$::*t1-space*

shows *closure* (*insert x s*) = *insert x* (*closure s*)
⟨*proof*⟩

lemma *islimpt-insert*:
fixes *x* :: *'a*::*t1-space*
shows *x islimpt* (*insert a s*) ⟷ *x islimpt s*
⟨*proof*⟩

lemma *islimpt-finite*:
fixes *x* :: *'a*::*t1-space*
shows *finite s* ⟹ ¬ *x islimpt s*
⟨*proof*⟩

lemma *islimpt-Un-finite*:
fixes *x* :: *'a*::*t1-space*
shows *finite s* ⟹ *x islimpt* (*s* ∪ *t*) ⟷ *x islimpt t*
⟨*proof*⟩

lemma *islimpt-eq-acc-point*:
fixes *l* :: *'a* :: *t1-space*
shows *l islimpt S* ⟷ (∀ *U*. *l*∈*U* ⟶ *open U* ⟶ *infinite* (*U* ∩ *S*))
⟨*proof*⟩

lemma *islimpt-range-imp-convergent-subsequence*:
fixes *l* :: *'a* :: {*t1-space*, *first-countable-topology*}
assumes *l*: *l islimpt* (*range f*)
shows ∃ *r*. *subseq r* ∧ (*f* ∘ *r*) ⟶⟶ *l*
⟨*proof*⟩

lemma *sequence-unique-limpt*:
fixes *f* :: *nat* ⇒ *'a*::*t2-space*
assumes (*f* ⟶⟶ *l*) *sequentially*
and *l'* *islimpt* (*range f*)
shows *l'* = *l*
⟨*proof*⟩

lemma *bolzano-weierstrass-imp-closed*:
fixes *s* :: *'a*::{*first-countable-topology*,*t2-space*} *set*
assumes ∀ *t*. *infinite t* ∧ *t* ⊆ *s* −−> (∃ *x* ∈ *s*. *x islimpt t*)
shows *closed s*
⟨*proof*⟩

lemma *compact-imp-bounded*:
assumes *compact U*
shows *bounded U*
⟨*proof*⟩

In particular, some common special cases.

lemma *compact-Un* [*intro*]:

assumes *compact s*
and *compact t*
shows *compact* $(s \cup t)$
⟨*proof*⟩

lemma *compact-Union* [*intro*]: *finite* $S \Longrightarrow (\bigwedge T.\ T \in S \Longrightarrow \mathit{compact}\ T) \Longrightarrow \mathit{compact}\ (\bigcup S)$
⟨*proof*⟩

lemma *compact-UN* [*intro*]:
finite $A \Longrightarrow (\bigwedge x.\ x \in A \Longrightarrow \mathit{compact}\ (B\ x)) \Longrightarrow \mathit{compact}\ (\bigcup x{\in}A.\ B\ x)$
⟨*proof*⟩

lemma *closed-Int-compact* [*intro*]:
assumes *closed s*
and *compact t*
shows *compact* $(s \cap t)$
⟨*proof*⟩

lemma *compact-Int* [*intro*]:
fixes $s\ t :: 'a :: \mathit{t2\text{-}space\ set}$
assumes *compact s*
and *compact t*
shows *compact* $(s \cap t)$
⟨*proof*⟩

lemma *compact-sing* [*simp*]: *compact* $\{a\}$
⟨*proof*⟩

lemma *compact-insert* [*simp*]:
assumes *compact s*
shows *compact* $(\mathit{insert}\ x\ s)$
⟨*proof*⟩

lemma *finite-imp-compact*: *finite* $s \Longrightarrow \mathit{compact}\ s$
⟨*proof*⟩

lemma *open-delete*:
fixes $s :: 'a{::}\mathit{t1\text{-}space\ set}$
shows *open* $s \Longrightarrow \mathit{open}\ (s - \{x\})$
⟨*proof*⟩

lemma *openin-delete*:
fixes $a :: 'a :: \mathit{t1\text{-}space}$
shows *openin* (*subtopology euclidean u*) *s*
$\Longrightarrow$ *openin* (*subtopology euclidean u*) $(s - \{a\})$
⟨*proof*⟩

Compactness expressed with filters

lemma *closure-iff-nhds-not-empty*:
$x \in \mathit{closure}\ X \longleftrightarrow (\forall A.\ \forall S{\subseteq}A.\ \mathit{open}\ S \longrightarrow x \in S \longrightarrow X \cap A \neq \{\})$
⟨*proof*⟩

lemma *compact-filter*:
$\mathit{compact}\ U \longleftrightarrow (\forall F.\ F \neq \mathit{bot} \longrightarrow \mathit{eventually}\ (\lambda x.\ x \in U)\ F \longrightarrow (\exists x{\in}U.\ \mathit{inf}\ (\mathit{nhds}\ x)\ F \neq \mathit{bot}))$
⟨*proof*⟩

definition *countably-compact U* $\longleftrightarrow$
$(\forall A.\ \mathit{countable}\ A \longrightarrow (\forall a{\in}A.\ \mathit{open}\ a) \longrightarrow U \subseteq \bigcup A \longrightarrow (\exists T{\subseteq}A.\ \mathit{finite}\ T \wedge U \subseteq \bigcup T))$

lemma *countably-compactE*:
assumes *countably-compact s* **and** $\forall t{\in}C.\ \mathit{open}\ t$ **and** $s \subseteq \bigcup C$ *countable C*
obtains C' **where** $C' \subseteq C$ **and** *finite* C' **and** $s \subseteq \bigcup C'$
⟨*proof*⟩

lemma *countably-compactI*:
assumes $\bigwedge C.\ \forall t{\in}C.\ \mathit{open}\ t \Longrightarrow s \subseteq \bigcup C \Longrightarrow \mathit{countable}\ C \Longrightarrow (\exists C'{\subseteq}C.\ \mathit{finite}\ C' \wedge s \subseteq \bigcup C')$
shows *countably-compact s*
⟨*proof*⟩

lemma *compact-imp-countably-compact*: *compact U* $\Longrightarrow$ *countably-compact U*
⟨*proof*⟩

lemma *countably-compact-imp-compact*:
assumes *countably-compact U*
and *ccover*: *countable B* $\forall b{\in}B.\ \mathit{open}\ b$
and *basis*: $\bigwedge T\ x.\ \mathit{open}\ T \Longrightarrow x \in T \Longrightarrow x \in U \Longrightarrow \exists b{\in}B.\ x \in b \wedge b \cap U \subseteq T$
shows *compact U*
⟨*proof*⟩

lemma *countably-compact-imp-compact-second-countable*:
countably-compact U $\Longrightarrow$ *compact* (*U* :: *'a* :: *second-countable-topology set*)
⟨*proof*⟩

lemma *countably-compact-eq-compact*:
countably-compact U $\longleftrightarrow$ *compact* (*U* :: *'a* :: *second-countable-topology set*)
⟨*proof*⟩

16.19.2 Sequential compactness

definition *seq-compact* :: *'a*::*topological-space set* $\Rightarrow$ *bool*
where *seq-compact S* $\longleftrightarrow$
$(\forall f.\ (\forall n.\ f\ n \in S) \longrightarrow (\exists l{\in}S.\ \exists r.\ \mathit{subseq}\ r \wedge ((f \circ r) \longrightarrow l)\ \mathit{sequentially}))$

lemma *seq-compactI*:
assumes $\bigwedge f.\ \forall n.\ f\,n \in S \Longrightarrow \exists\, l{\in}S.\ \exists\, r.\ subseq\ r \wedge ((f \circ r) \longrightarrow l)\ sequentially$
shows *seq-compact S*
⟨*proof*⟩

lemma *seq-compactE*:
assumes *seq-compact S* $\forall n.\ f\,n \in S$
obtains $l\ r$ **where** $l \in S$ *subseq r* $((f \circ r) \longrightarrow l)$ *sequentially*
⟨*proof*⟩

lemma *closed-sequentially*:
assumes *closed s* **and** $\forall n.\ f\,n \in s$ **and** $f \longrightarrow l$
shows $l \in s$
⟨*proof*⟩

lemma *seq-compact-Int-closed*:
assumes *seq-compact s* **and** *closed t*
shows *seq-compact* $(s \cap t)$
⟨*proof*⟩

lemma *seq-compact-closed-subset*:
assumes *closed s* **and** $s \subseteq t$ **and** *seq-compact t*
shows *seq-compact s*
⟨*proof*⟩

lemma *seq-compact-imp-countably-compact*:
fixes $U :: 'a ::$ *first-countable-topology set*
assumes *seq-compact U*
shows *countably-compact U*
⟨*proof*⟩

lemma *compact-imp-seq-compact*:
fixes $U :: 'a ::$ *first-countable-topology set*
assumes *compact U*
shows *seq-compact U*
⟨*proof*⟩

lemma *countably-compact-imp-acc-point*:
assumes *countably-compact s*
and *countable t*
and *infinite t*
and $t \subseteq s$
shows $\exists\, x{\in}s.\ \forall\, U.\ x{\in}U \wedge open\ U \longrightarrow infinite\ (U \cap t)$
⟨*proof*⟩

lemma *countable-acc-point-imp-seq-compact*:
fixes $s :: 'a::$*first-countable-topology set*
assumes $\forall\, t.\ infinite\ t \wedge countable\ t \wedge t \subseteq s \longrightarrow$
$(\exists\, x{\in}s.\ \forall\, U.\ x{\in}U \wedge open\ U \longrightarrow infinite\ (U \cap t))$

shows *seq-compact s*
⟨*proof*⟩

lemma *seq-compact-eq-countably-compact*:
fixes $U :: 'a :: first\text{-}countable\text{-}topology\ set$
shows *seq-compact U* $\longleftrightarrow$ *countably-compact U*
⟨*proof*⟩

lemma *seq-compact-eq-acc-point*:
fixes $s :: 'a :: first\text{-}countable\text{-}topology\ set$
shows *seq-compact s* $\longleftrightarrow$
$(\forall t.\ infinite\ t \wedge countable\ t \wedge t \subseteq s \longrightarrow (\exists x{\in}s.\ \forall U.\ x{\in}U \wedge open\ U \longrightarrow infinite\ (U \cap t)))$
⟨*proof*⟩

lemma *seq-compact-eq-compact*:
fixes $U :: 'a :: second\text{-}countable\text{-}topology\ set$
shows *seq-compact U* $\longleftrightarrow$ *compact U*
⟨*proof*⟩

lemma *bolzano-weierstrass-imp-seq-compact*:
fixes $s :: 'a{::}\{t1\text{-}space, first\text{-}countable\text{-}topology\}\ set$
shows $\forall t.\ infinite\ t \wedge t \subseteq s \longrightarrow (\exists x \in s.\ x\ islimpt\ t) \Longrightarrow$ *seq-compact s*
⟨*proof*⟩

16.19.3 Totally bounded

lemma *cauchy-def*: $Cauchy\ s \longleftrightarrow (\forall e{>}0.\ \exists N.\ \forall m\ n.\ m \geq N \wedge n \geq N \longrightarrow dist(s\ m)(s\ n) < e)$
⟨*proof*⟩

lemma *seq-compact-imp-totally-bounded*:
assumes *seq-compact s*
shows $\forall e{>}0.\ \exists k.\ finite\ k \wedge k \subseteq s \wedge s \subseteq (\bigcup x{\in}k.\ ball\ x\ e)$
⟨*proof*⟩

16.19.4 Heine-Borel theorem

lemma *seq-compact-imp-heine-borel*:
fixes $s :: 'a :: metric\text{-}space\ set$
assumes *seq-compact s*
shows *compact s*
⟨*proof*⟩

lemma *compact-eq-seq-compact-metric*:
compact $(s :: 'a{::}metric\text{-}space\ set)$ $\longleftrightarrow$ *seq-compact s*
⟨*proof*⟩

lemma *compact-def*:
compact $(S :: 'a{::}metric\text{-}space\ set) \longleftrightarrow$

$(\forall f.\ (\forall n.\ f\ n \in S) \longrightarrow (\exists l{\in}S.\ \exists r.\ subseq\ r \wedge (f \circ r) \longrightarrow l))$
⟨*proof*⟩

16.19.5 Complete the chain of compactness variants

lemma *compact-eq-bolzano-weierstrass*:
fixes $s :: 'a{::}metric\text{-}space\ set$
shows $compact\ s \longleftrightarrow (\forall t.\ infinite\ t \wedge t \subseteq s \longrightarrow (\exists x \in s.\ x\ islimpt\ t))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *bolzano-weierstrass-imp-bounded*:
$\forall t.\ infinite\ t \wedge t \subseteq s \longrightarrow (\exists x \in s.\ x\ islimpt\ t) \Longrightarrow bounded\ s$
⟨*proof*⟩

16.20 Metric spaces with the Heine-Borel property

A metric space (or topological vector space) is said to have the Heine-Borel property if every closed and bounded subset is compact.

class *heine-borel* = *metric-space* +
assumes *bounded-imp-convergent-subsequence*:
$bounded\ (range\ f) \Longrightarrow \exists l\ r.\ subseq\ r \wedge ((f \circ r) \longrightarrow l)\ sequentially$

lemma *bounded-closed-imp-seq-compact*:
fixes $s{::}'a{::}heine\text{-}borel\ set$
assumes *bounded s*
and *closed s*
shows *seq-compact s*
⟨*proof*⟩

lemma *compact-eq-bounded-closed*:
fixes $s :: 'a{::}heine\text{-}borel\ set$
shows $compact\ s \longleftrightarrow bounded\ s \wedge closed\ s$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *compact-closure* [*simp*]:
fixes $S :: 'a{::}heine\text{-}borel\ set$
shows $compact(closure\ S) \longleftrightarrow bounded\ S$
⟨*proof*⟩

lemma *compact-components*:
fixes $s :: 'a{::}heine\text{-}borel\ set$
shows $[\![compact\ s;\ c \in components\ s]\!] \Longrightarrow compact\ c$
⟨*proof*⟩

lemma *not-compact-UNIV* [*simp*]:
fixes $s :: 'a{::}\{real\text{-}normed\text{-}vector,perfect\text{-}space,heine\text{-}borel\}\ set$
shows ~ $compact\ (UNIV{::}'a\ set)$

⟨*proof*⟩

lemma *bounded-increasing-convergent*:
fixes $s :: nat \Rightarrow real$
shows $bounded\ \{s\ n|\ n.\ True\} \Longrightarrow \forall n.\ s\ n \leq s\ (Suc\ n) \Longrightarrow \exists l.\ s \longrightarrow l$
⟨*proof*⟩

instance *real* :: *heine-borel*
⟨*proof*⟩

lemma *compact-lemma-general*:
fixes $f :: nat \Rightarrow 'a$
fixes $proj :: 'a \Rightarrow 'b \Rightarrow 'c :: heine\text{-}borel$ (**infixl** *proj 60*)
fixes $unproj :: ('b \Rightarrow 'c) \Rightarrow 'a$
assumes *finite-basis*: *finite basis*
assumes *bounded-proj*: $\bigwedge k.\ k \in basis \Longrightarrow bounded\ ((\lambda x.\ x\ proj\ k)\ `\ range\ f)$
assumes *proj-unproj*: $\bigwedge e\ k.\ k \in basis \Longrightarrow (unproj\ e)\ proj\ k = e\ k$
assumes *unproj-proj*: $\bigwedge x.\ unproj\ (\lambda k.\ x\ proj\ k) = x$
shows $\forall d \subseteq basis.\ \exists l :: 'a.\ \exists\ r.$
$subseq\ r \wedge (\forall e > 0.\ eventually\ (\lambda n.\ \forall i \in d.\ dist\ (f\ (r\ n)\ proj\ i)\ (l\ proj\ i) < e)\ sequentially)$
⟨*proof*⟩

lemma *compact-lemma*:
fixes $f :: nat \Rightarrow 'a :: euclidean\text{-}space$
assumes *bounded* (*range f*)
shows $\forall d \subseteq Basis.\ \exists l :: 'a.\ \exists\ r.$
$subseq\ r \wedge (\forall e > 0.\ eventually\ (\lambda n.\ \forall i \in d.\ dist\ (f\ (r\ n) \cdot i)\ (l \cdot i) < e)\ sequentially)$
⟨*proof*⟩

instance *euclidean-space* $\subseteq$ *heine-borel*
⟨*proof*⟩

lemma *bounded-fst*: *bounded s* $\Longrightarrow$ *bounded* (*fst* ` *s*)
⟨*proof*⟩

lemma *bounded-snd*: *bounded s* $\Longrightarrow$ *bounded* (*snd* ` *s*)
⟨*proof*⟩

instance *prod* :: (*heine-borel*, *heine-borel*) *heine-borel*
⟨*proof*⟩

16.20.1 Completeness

lemma (**in** *metric-space*) *completeI*:
assumes $\bigwedge f.\ \forall n.\ f\ n \in s \Longrightarrow Cauchy\ f \Longrightarrow \exists l \in s.\ f \longrightarrow l$
shows *complete s*

⟨*proof*⟩

lemma (**in** *metric-space*) *completeE*:
assumes *complete s* **and** ∀ *n*. *f n* ∈ *s* **and** *Cauchy f*
obtains *l* **where** *l* ∈ *s* **and** *f* ⟶ *l*
⟨*proof*⟩

lemma *compact-imp-complete*:
fixes *s* :: *'a*::*metric-space set*
assumes *compact s*
shows *complete s*
⟨*proof*⟩

lemma *nat-approx-posE*:
fixes *e*::*real*
assumes *0* < *e*
obtains *n* :: *nat* **where** *1* / (*Suc n*) < *e*
⟨*proof*⟩

lemma *compact-eq-totally-bounded*:
compact s ⟷ *complete s* ∧ (∀ *e*>*0*. ∃ *k*. *finite k* ∧ *s* ⊆ (⋃ *x*∈*k*. *ball x e*))
(**is** - ⟷ *?rhs*)
⟨*proof*⟩

lemma *cauchy*: *Cauchy s* ⟷ (∀ *e*>*0*.∃ *N*::*nat*. ∀ *n*≥*N*. *dist*(*s n*)(*s N*) < *e*) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *cauchy-imp-bounded*:
assumes *Cauchy s*
shows *bounded* (*range s*)
⟨*proof*⟩

instance *heine-borel* < *complete-space*
⟨*proof*⟩

instance *euclidean-space* ⊆ *banach* ⟨*proof*⟩

lemma *complete-UNIV*: *complete* (*UNIV* :: (*'a*::*complete-space*) *set*)
⟨*proof*⟩

lemma *complete-imp-closed*:
fixes *s* :: *'a*::*metric-space set*
assumes *complete s*
shows *closed s*
⟨*proof*⟩

lemma *complete-Int-closed*:

fixes $s :: 'a::metric\text{-}space\ set$
assumes *complete* s **and** *closed* t
shows *complete* $(s \cap t)$
⟨*proof*⟩

lemma *complete-closed-subset*:
fixes $s :: 'a::metric\text{-}space\ set$
assumes *closed* s **and** $s \subseteq t$ **and** *complete* t
shows *complete* s
⟨*proof*⟩

lemma *complete-eq-closed*:
fixes $s :: ('a::complete\text{-}space)\ set$
shows *complete* $s \longleftrightarrow$ *closed* s
⟨*proof*⟩

lemma *convergent-eq-cauchy*:
fixes $s :: nat \Rightarrow 'a::complete\text{-}space$
shows $(\exists\, l.\ (s \longrightarrow l)\ sequentially) \longleftrightarrow Cauchy\ s$
⟨*proof*⟩

lemma *convergent-imp-bounded*:
fixes $s :: nat \Rightarrow 'a::metric\text{-}space$
shows $(s \longrightarrow l)\ sequentially \Longrightarrow bounded\ (range\ s)$
⟨*proof*⟩

lemma *compact-cball*[*simp*]:
fixes $x :: 'a::heine\text{-}borel$
shows *compact* $(cball\ x\ e)$
⟨*proof*⟩

lemma *compact-frontier-bounded*[*intro*]:
fixes $s :: 'a::heine\text{-}borel\ set$
shows *bounded* $s \Longrightarrow$ *compact* $(frontier\ s)$
⟨*proof*⟩

lemma *compact-frontier*[*intro*]:
fixes $s :: 'a::heine\text{-}borel\ set$
shows *compact* $s \Longrightarrow$ *compact* $(frontier\ s)$
⟨*proof*⟩

corollary *compact-sphere*:
fixes $a :: 'a::\{real\text{-}normed\text{-}vector, perfect\text{-}space, heine\text{-}borel\}$
shows *compact* $(sphere\ a\ r)$
⟨*proof*⟩

lemma *frontier-subset-compact*:
fixes $s :: 'a::heine\text{-}borel\ set$
shows *compact* $s \Longrightarrow$ *frontier* $s \subseteq s$

⟨*proof*⟩

16.21 Relations among convergence and absolute convergence for power series.

lemma *summable-imp-bounded*:
fixes f :: *nat* ⇒ $'a$::*real-normed-vector*
shows *summable* f ⟹ *bounded* (*range* f)
⟨*proof*⟩

lemma *summable-imp-sums-bounded*:
summable f ⟹ *bounded* (*range* (λ*n*. *setsum* f {..<*n*}))
⟨*proof*⟩

lemma *power-series-conv-imp-absconv-weak*:
fixes *a*:: *nat* ⇒ $'a$::{*real-normed-div-algebra*,*banach*} **and** w :: $'a$
assumes *sum*: *summable* (λ*n*. *a n* ∗ *z* ^ *n*) **and** *no*: *norm w* < *norm z*
shows *summable* (λ*n*. *of-real*(*norm*(*a n*)) ∗ *w* ^ *n*)
⟨*proof*⟩

16.22 Bounded closed nest property (proof does not use Heine-Borel)

lemma *bounded-closed-nest*:
fixes *s* :: *nat* ⇒ ($'a$::*heine-borel*) *set*
assumes ∀ *n*. *closed* (*s n*)
and ∀ *n*. *s n* ≠ {}
and ∀ *m n*. *m* ≤ *n* ⟶ *s n* ⊆ *s m*
and *bounded* (*s* 0)
shows ∃ *a*. ∀ *n*. *a* ∈ *s n*
⟨*proof*⟩

Decreasing case does not even need compactness, just completeness.

lemma *decreasing-closed-nest*:
fixes *s* :: *nat* ⇒ ($'a$::*complete-space*) *set*
assumes
∀ *n*. *closed* (*s n*)
∀ *n*. *s n* ≠ {}
∀ *m n*. *m* ≤ *n* ⟶ *s n* ⊆ *s m*
∀ *e*>0. ∃ *n*. ∀ *x*∈*s n*. ∀ *y*∈*s n*. *dist x y* < *e*
shows ∃ *a*. ∀ *n*. *a* ∈ *s n*
⟨*proof*⟩

Strengthen it to the intersection actually being a singleton.

lemma *decreasing-closed-nest-sing*:
fixes *s* :: *nat* ⇒ $'a$::*complete-space set*
assumes
∀ *n*. *closed*(*s n*)
∀ *n*. *s n* ≠ {}

$\forall m\ n.\ m \leq n \longrightarrow s\ n \subseteq s\ m$
$\forall e{>}0.\ \exists n.\ \forall x \in (s\ n).\ \forall\ y{\in}(s\ n).\ dist\ x\ y < e$
shows $\exists a.\ \bigcap(range\ s) = \{a\}$
⟨*proof*⟩

Cauchy-type criteria for uniform convergence.

lemma *uniformly-convergent-eq-cauchy*:
fixes *s*::*nat* ⇒ *'b* ⇒ *'a*::*complete-space*
shows
$(\exists l.\ \forall e{>}0.\ \exists N.\ \forall n\ x.\ N \leq n \wedge P\ x \longrightarrow dist(s\ n\ x)(l\ x) < e) \longleftrightarrow$
$(\forall e{>}0.\ \exists N.\ \forall m\ n\ x.\ N \leq m \wedge N \leq n \wedge P\ x \longrightarrow dist\ (s\ m\ x)\ (s\ n\ x) < e)$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *uniformly-cauchy-imp-uniformly-convergent*:
fixes *s* :: *nat* ⇒ *'a* ⇒ *'b*::*complete-space*
assumes $\forall e{>}0.\exists N.\ \forall m\ (n{::}nat)\ x.\ N \leq m \wedge N \leq n \wedge P\ x \dashrightarrow dist(s\ m\ x)(s\ n\ x) < e$
and $\forall x.\ P\ x \dashrightarrow (\forall e{>}0.\ \exists N.\ \forall n.\ N \leq n \longrightarrow dist(s\ n\ x)(l\ x) < e)$
shows $\forall e{>}0.\ \exists N.\ \forall n\ x.\ N \leq n \wedge P\ x \longrightarrow dist(s\ n\ x)(l\ x) < e$
⟨*proof*⟩

16.23 Continuity

Derive the epsilon-delta forms, which we often use as "definitions"

lemma *continuous-within-eps-delta*:
continuous (*at x within s*) $f \longleftrightarrow (\forall e{>}0.\ \exists d{>}0.\ \forall x'{\in}\ s.\ \ dist\ x'\ x < d \dashrightarrow dist\ (f\ x')\ (f\ x) < e)$
⟨*proof*⟩

corollary *continuous-at-eps-delta*:
continuous (*at x*) $f \longleftrightarrow (\forall e > 0.\ \exists d > 0.\ \forall x'.\ dist\ x'\ x < d \longrightarrow dist\ (f\ x')\ (f\ x) < e)$
⟨*proof*⟩

lemma *continuous-at-right-real-increasing*:
fixes *f* :: *real* ⇒ *real*
assumes *nondecF*: $\bigwedge x\ y.\ x \leq y \Longrightarrow f\ x \leq f\ y$
shows *continuous* (*at-right a*) $f \longleftrightarrow (\forall e{>}0.\ \exists d{>}0.\ f\ (a + d) - f\ a < e)$
⟨*proof*⟩

lemma *continuous-at-left-real-increasing*:
assumes *nondecF*: $\bigwedge\ x\ y.\ x \leq y \Longrightarrow f\ x \leq ((f\ y) :: real)$
shows (*continuous* (*at-left* (*a* :: *real*)) *f*) = $(\forall e > 0.\ \exists delta > 0.\ f\ a - f\ (a - delta) < e)$
⟨*proof*⟩

Versions in terms of open balls.

lemma *continuous-within-ball*:

continuous (*at x within s*) *f* ⟷
(∀ *e* > *0*. ∃ *d* > *0*. *f* ‘ (*ball x d* ∩ *s*) ⊆ *ball* (*f x*) *e*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *continuous-at-ball*:
continuous (*at x*) *f* ⟷ (∀ *e*>*0*. ∃ *d*>*0*. *f* ‘ (*ball x d*) ⊆ *ball* (*f x*) *e*) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

Define setwise continuity in terms of limits within the set.

lemma *continuous-on-iff*:
continuous-on s f ⟷
(∀ *x*∈*s*. ∀ *e*>*0*. ∃ *d*>*0*. ∀ *x′*∈*s*. *dist x′ x* < *d* ⟶ *dist* (*f x′*) (*f x*) < *e*)
⟨*proof*⟩

lemma *continuous-within-E*:
assumes *continuous* (*at x within s*) *f e*>*0*
obtains *d* **where** *d*>*0* ⋀*x′*. ⟦*x′*∈ *s*; *dist x′ x* ≤ *d*⟧ ⟹ *dist* (*f x′*) (*f x*) < *e*
⟨*proof*⟩

lemma *continuous-onI* [*intro?*]:
assumes ⋀*x e*. ⟦*e* > *0*; *x* ∈ *s*⟧ ⟹ ∃ *d*>*0*. ∀ *x′*∈*s*. *dist x′ x* < *d* ⟶ *dist* (*f x′*) (*f x*) ≤ *e*
shows *continuous-on s f*
⟨*proof*⟩

lemma *uniformly-continuous-on-def*:
fixes *f* :: *′a*::*metric-space* ⇒ *′b*::*metric-space*
shows *uniformly-continuous-on s f* ⟷
(∀ *e*>*0*. ∃ *d*>*0*. ∀ *x*∈*s*. ∀ *x′*∈*s*. *dist x′ x* < *d* ⟶ *dist* (*f x′*) (*f x*) < *e*)
⟨*proof*⟩

Some simple consequential lemmas.

lemma *continuous-onE*:
assumes *continuous-on s f x*∈*s e*>*0*
obtains *d* **where** *d*>*0* ⋀*x′*. ⟦*x′* ∈ *s*; *dist x′ x* ≤ *d*⟧ ⟹ *dist* (*f x′*) (*f x*) < *e*
⟨*proof*⟩

lemma *uniformly-continuous-onE*:
assumes *uniformly-continuous-on s f* 0 < *e*
obtains *d* **where** *d*>*0* ⋀*x x′*. ⟦*x*∈*s*; *x′*∈*s*; *dist x′ x* < *d*⟧ ⟹ *dist* (*f x′*) (*f x*) < *e*
⟨*proof*⟩

lemma *continuous-at-imp-continuous-within*:
continuous (*at x*) *f* ⟹ *continuous* (*at x within s*) *f*
⟨*proof*⟩

lemma *Lim-trivial-limit*: *trivial-limit net* $\Longrightarrow$ $(f \longrightarrow l)$ *net*
⟨*proof*⟩

lemmas *continuous-on* = *continuous-on-def* — legacy theorem name

lemma *continuous-within-subset*:
continuous (*at x within s*) *f* $\Longrightarrow$ $t \subseteq s$ $\Longrightarrow$ *continuous* (*at x within t*) *f*
⟨*proof*⟩

lemma *continuous-on-interior*:
continuous-on s f $\Longrightarrow$ $x \in$ *interior s* $\Longrightarrow$ *continuous* (*at x*) *f*
⟨*proof*⟩

lemma *continuous-on-eq*:
$[\![$*continuous-on s f*; $\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x]\!] \Longrightarrow$ *continuous-on s g*
⟨*proof*⟩

Characterization of various kinds of continuity in terms of sequences.

lemma *continuous-within-sequentially*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*topological-space*
shows *continuous* (*at a within s*) *f* $\longleftrightarrow$
$(\forall x.\ (\forall n$::*nat*. *x n* $\in s) \wedge (x \longrightarrow a)$ *sequentially*
$\longrightarrow ((f \circ x) \longrightarrow f\ a)$ *sequentially*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *continuous-at-sequentially*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*topological-space*
shows *continuous* (*at a*) *f* $\longleftrightarrow$
$(\forall x.\ (x \longrightarrow a)$ *sequentially* $-->$ $((f \circ x) \longrightarrow f\ a)$ *sequentially*)
⟨*proof*⟩

lemma *continuous-on-sequentially*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*topological-space*
shows *continuous-on s f* $\longleftrightarrow$
$(\forall x.\ \forall a \in s.\ (\forall n.\ x(n) \in s) \wedge (x \longrightarrow a)$ *sequentially*
$-->$ $((f \circ x) \longrightarrow f\ a)$ *sequentially*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *uniformly-continuous-on-sequentially*:
uniformly-continuous-on s f $\longleftrightarrow$ $(\forall x\ y.\ (\forall n.\ x\ n \in s) \wedge (\forall n.\ y\ n \in s) \wedge$
$(\lambda n.\ dist\ (x\ n)\ (y\ n)) \longrightarrow 0 \longrightarrow (\lambda n.\ dist\ (f(x\ n))\ (f(y\ n))) \longrightarrow 0)$ (**is**
?lhs = *?rhs*)
⟨*proof*⟩

The usual transformation theorems.

lemma *continuous-transform-within*:
fixes *f g* :: *'a*::*metric-space* ⇒ *'b*::*topological-space*

assumes *continuous* (*at x within s*) *f*
and $0 < d$
and $x \in s$
and $\bigwedge x'.\ [\![x' \in s;\ dist\ x'\ x < d]\!] \Longrightarrow f\ x' = g\ x'$
shows *continuous* (*at x within s*) *g*
⟨*proof*⟩

16.23.1 Structural rules for pointwise continuity

lemma *continuous-infdist*[*continuous-intros*]:
assumes *continuous F f*
shows *continuous F* ($\lambda x.$ *infdist* (*f x*) *A*)
⟨*proof*⟩

lemma *continuous-infnorm*[*continuous-intros*]:
continuous F f $\Longrightarrow$ *continuous F* ($\lambda x.$ *infnorm* (*f x*))
⟨*proof*⟩

lemma *continuous-inner*[*continuous-intros*]:
assumes *continuous F f*
and *continuous F g*
shows *continuous F* ($\lambda x.$ *inner* (*f x*) (*g x*))
⟨*proof*⟩

lemmas *continuous-at-inverse* = *isCont-inverse*

16.23.2 Structural rules for setwise continuity

lemma *continuous-on-infnorm*[*continuous-intros*]:
continuous-on s f $\Longrightarrow$ *continuous-on s* ($\lambda x.$ *infnorm* (*f x*))
⟨*proof*⟩

lemma *continuous-on-inner*[*continuous-intros*]:
fixes *g* :: *'a*::*topological-space* $\Rightarrow$ *'b*::*real-inner*
assumes *continuous-on s f*
and *continuous-on s g*
shows *continuous-on s* ($\lambda x.$ *inner* (*f x*) (*g x*))
⟨*proof*⟩

16.23.3 Structural rules for uniform continuity

lemma *uniformly-continuous-on-dist*[*continuous-intros*]:
fixes *f g* :: *'a*::*metric-space* $\Rightarrow$ *'b*::*metric-space*
assumes *uniformly-continuous-on s f*
and *uniformly-continuous-on s g*
shows *uniformly-continuous-on s* ($\lambda x.$ *dist* (*f x*) (*g x*))
⟨*proof*⟩

lemma *uniformly-continuous-on-norm*[*continuous-intros*]:
fixes *f* :: *'a* :: *metric-space* $\Rightarrow$ *'b* :: *real-normed-vector*

assumes *uniformly-continuous-on s f*
shows *uniformly-continuous-on s* (λ*x*. *norm* (*f x*))
⟨*proof*⟩

lemma (**in** *bounded-linear*) *uniformly-continuous-on*[*continuous-intros*]:
fixes *g* :: -::*metric-space* ⇒ -
assumes *uniformly-continuous-on s g*
shows *uniformly-continuous-on s* (λ*x*. *f* (*g x*))
⟨*proof*⟩

lemma *uniformly-continuous-on-cmul*[*continuous-intros*]:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*real-normed-vector*
assumes *uniformly-continuous-on s f*
shows *uniformly-continuous-on s* (λ*x*. *c* ∗$_R$ *f*(*x*))
⟨*proof*⟩

lemma *dist-minus*:
fixes *x y* :: *'a*::*real-normed-vector*
shows *dist* (− *x*) (− *y*) = *dist x y*
⟨*proof*⟩

lemma *uniformly-continuous-on-minus*[*continuous-intros*]:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*real-normed-vector*
shows *uniformly-continuous-on s f* ⟹ *uniformly-continuous-on s* (λ*x*. − *f x*)
⟨*proof*⟩

lemma *uniformly-continuous-on-add*[*continuous-intros*]:
fixes *f g* :: *'a*::*metric-space* ⇒ *'b*::*real-normed-vector*
assumes *uniformly-continuous-on s f*
and *uniformly-continuous-on s g*
shows *uniformly-continuous-on s* (λ*x*. *f x* + *g x*)
⟨*proof*⟩

lemma *uniformly-continuous-on-diff*[*continuous-intros*]:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*real-normed-vector*
assumes *uniformly-continuous-on s f*
and *uniformly-continuous-on s g*
shows *uniformly-continuous-on s* (λ*x*. *f x* − *g x*)
⟨*proof*⟩

lemmas *continuous-at-compose* = *isCont-o*

Continuity in terms of open preimages.

lemma *continuous-at-open*:
continuous (*at x*) *f* ⟷ (∀ *t*. *open t* ∧ *f x* ∈ *t* −−> (∃ *s*. *open s* ∧ *x* ∈ *s* ∧ (∀ *x'* ∈ *s*. (*f x'*) ∈ *t*)))
⟨*proof*⟩

lemma *continuous-imp-tendsto*:

assumes *continuous* (*at x0*) *f*
and $x \longrightarrow x0$
shows $(f \circ x) \longrightarrow (f\ x0)$
⟨*proof*⟩

lemma *continuous-on-open*:
continuous-on s f ⟷
$(\forall t.$ *openin* (*subtopology euclidean* (*f* ' *s*)) *t* ⟶
openin (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\}$)
⟨*proof*⟩

Similarly in terms of closed sets.

lemma *continuous-on-closed*:
continuous-on s f ⟷
$(\forall t.$ *closedin* (*subtopology euclidean* (*f* ' *s*)) *t* ⟶
closedin (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\}$)
⟨*proof*⟩

Half-global and completely global cases.

lemma *continuous-openin-preimage*:
assumes *continuous-on s f open t*
shows *openin* (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\}$
⟨*proof*⟩

lemma *continuous-closedin-preimage*:
assumes *continuous-on s f* **and** *closed t*
shows *closedin* (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\}$
⟨*proof*⟩

lemma *continuous-open-preimage*:
assumes *continuous-on s f*
and *open s*
and *open t*
shows *open* $\{x \in s.\ f\ x \in t\}$
⟨*proof*⟩

lemma *continuous-closed-preimage*:
assumes *continuous-on s f*
and *closed s*
and *closed t*
shows *closed* $\{x \in s.\ f\ x \in t\}$
⟨*proof*⟩

lemma *continuous-open-preimage-univ*:
$\forall x.$ *continuous* (*at x*) *f* ⟹ *open s* ⟹ *open* $\{x.\ f\ x \in s\}$
⟨*proof*⟩

lemma *continuous-closed-preimage-univ*:
$(\forall x.$ *continuous* (*at x*) *f*) ⟹ *closed s* ⟹ *closed* $\{x.\ f\ x \in s\}$

⟨*proof*⟩

lemma *continuous-open-vimage*: ∀ *x*. *continuous* (*at x*) *f* ⟹ *open s* ⟹ *open* (*f* −‘ *s*)
⟨*proof*⟩

lemma *continuous-closed-vimage*: ∀ *x*. *continuous* (*at x*) *f* ⟹ *closed s* ⟹ *closed* (*f* −‘ *s*)
⟨*proof*⟩

lemma *interior-image-subset*:
assumes ∀ *x*. *continuous* (*at x*) *f*
and *inj f*
shows *interior* (*f* ‘ *s*) ⊆ *f* ‘ (*interior s*)
⟨*proof*⟩

Equality of continuous functions on closure and related results.

lemma *continuous-closedin-preimage-constant*:
fixes *f* :: - ⇒ *'b*::*t1-space*
shows *continuous-on s f* ⟹ *closedin* (*subtopology euclidean s*) {*x* ∈ *s*. *f x* = *a*}
⟨*proof*⟩

lemma *continuous-closed-preimage-constant*:
fixes *f* :: - ⇒ *'b*::*t1-space*
shows *continuous-on s f* ⟹ *closed s* ⟹ *closed* {*x* ∈ *s*. *f x* = *a*}
⟨*proof*⟩

lemma *continuous-constant-on-closure*:
fixes *f* :: - ⇒ *'b*::*t1-space*
assumes *continuous-on* (*closure S*) *f*
and ⋀*x*. *x* ∈ *S* ⟹ *f x* = *a*
and *x* ∈ *closure S*
shows *f x* = *a*
⟨*proof*⟩

lemma *image-closure-subset*:
assumes *continuous-on* (*closure s*) *f*
and *closed t*
and (*f* ‘ *s*) ⊆ *t*
shows *f* ‘ (*closure s*) ⊆ *t*
⟨*proof*⟩

lemma *continuous-on-closure-norm-le*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*real-normed-vector*
assumes *continuous-on* (*closure s*) *f*
and ∀ *y* ∈ *s*. *norm*(*f y*) ≤ *b*
and *x* ∈ (*closure s*)
shows *norm* (*f x*) ≤ *b*
⟨*proof*⟩

lemma *isCont-indicator*:
fixes $x :: 'a::t2\text{-}space$
shows $isCont\ (indicator\ A :: 'a \Rightarrow real)\ x = (x \notin frontier\ A)$
⟨*proof*⟩

16.24 Theorems relating continuity and uniform continuity to closures

lemma *continuous-on-closure*:
$continuous\text{-}on\ (closure\ S)\ f \longleftrightarrow$
$(\forall x\ e.\ x \in closure\ S \wedge 0 < e \longrightarrow (\exists d.\ 0 < d \wedge (\forall y.\ y \in S \wedge dist\ y\ x < d \longrightarrow dist\ (f\ y)\ (f\ x) < e)))$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *continuous-on-closure-sequentially*:
fixes $f :: 'a::metric\text{-}space \Rightarrow 'b :: metric\text{-}space$
shows
$continuous\text{-}on\ (closure\ S)\ f \longleftrightarrow$
$(\forall x\ a.\ a \in closure\ S \wedge (\forall n.\ x\ n \in S) \wedge x \longrightarrow a \longrightarrow (f \circ x) \longrightarrow f\ a)$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *uniformly-continuous-on-closure*:
fixes $f :: 'a::metric\text{-}space \Rightarrow 'b::metric\text{-}space$
assumes *ucont*: $uniformly\text{-}continuous\text{-}on\ S\ f$
and *cont*: $continuous\text{-}on\ (closure\ S)\ f$
shows $uniformly\text{-}continuous\text{-}on\ (closure\ S)\ f$
⟨*proof*⟩

16.25 Quotient maps

lemma *quotient-map-imp-continuous-open*:
assumes *t*: $f\ `\ s \subseteq t$
and *ope*: $\bigwedge u.\ u \subseteq t \Longrightarrow (openin\ (subtopology\ euclidean\ s)\ \{x.\ x \in s \wedge f\ x \in u\} \longleftrightarrow openin\ (subtopology\ euclidean\ t)\ u)$
shows $continuous\text{-}on\ s\ f$
⟨*proof*⟩

lemma *quotient-map-imp-continuous-closed*:
assumes *t*: $f\ `\ s \subseteq t$
and *ope*: $\bigwedge u.\ u \subseteq t \Longrightarrow (closedin\ (subtopology\ euclidean\ s)\ \{x.\ x \in s \wedge f\ x \in u\} \longleftrightarrow closedin\ (subtopology\ euclidean\ t)\ u)$
shows $continuous\text{-}on\ s\ f$
⟨*proof*⟩

lemma *open-map-imp-quotient-map*:
assumes *contf*: *continuous-on s f*
and *t*: $t \subseteq f \mathbin{`} s$
and *ope*: $\bigwedge t.$ *openin* (*subtopology euclidean s*) *t*
$\Longrightarrow$ *openin* (*subtopology euclidean* ($f \mathbin{`} s$)) ($f \mathbin{`} t$)
shows *openin* (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\} =$
openin (*subtopology euclidean* ($f \mathbin{`} s$)) *t*
⟨*proof*⟩

lemma *closed-map-imp-quotient-map*:
assumes *contf*: *continuous-on s f*
and *t*: $t \subseteq f \mathbin{`} s$
and *ope*: $\bigwedge t.$ *closedin* (*subtopology euclidean s*) *t*
$\Longrightarrow$ *closedin* (*subtopology euclidean* ($f \mathbin{`} s$)) ($f \mathbin{`} t$)
shows *openin* (*subtopology euclidean s*) $\{x \in s.\ f\ x \in t\} \longleftrightarrow$
openin (*subtopology euclidean* ($f \mathbin{`} s$)) *t*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *continuous-right-inverse-imp-quotient-map*:
assumes *contf*: *continuous-on s f* **and** *imf*: $f \mathbin{`} s \subseteq t$
and *contg*: *continuous-on t g* **and** *img*: $g \mathbin{`} t \subseteq s$
and *fg* [*simp*]: $\bigwedge y.\ y \in t \Longrightarrow f(g\ y) = y$
and *u*: $u \subseteq t$
shows *openin* (*subtopology euclidean s*) $\{x.\ x \in s \wedge f\ x \in u\} \longleftrightarrow$
openin (*subtopology euclidean t*) *u*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *continuous-left-inverse-imp-quotient-map*:
assumes *continuous-on s f*
and *continuous-on* ($f \mathbin{`} s$) *g*
and $\bigwedge x.\ x \in s \Longrightarrow g(f\ x) = x$
and $u \subseteq f \mathbin{`} s$
shows *openin* (*subtopology euclidean s*) $\{x.\ x \in s \wedge f\ x \in u\} \longleftrightarrow$
openin (*subtopology euclidean* ($f \mathbin{`} s$)) *u*
⟨*proof*⟩

16.26 A function constant on a set

definition *constant-on* (**infixl** (*constant'-on*) *50*)
where *f constant-on A* $\equiv \exists y.\ \forall x{\in}A.\ f\ x = y$

lemma *constant-on-subset*: $[\![f$ *constant-on A*; $B \subseteq A]\!] \Longrightarrow f$ *constant-on B*
⟨*proof*⟩

lemma *injective-not-constant*:
fixes *S* :: *'a*::{*perfect-space*} *set*
shows $[\![$*open S*; *inj-on f S*; *f constant-on S*$]\!] \Longrightarrow S = \{\}$

⟨*proof*⟩

lemma *constant-on-closureI*:
fixes *f* :: - ⇒ *'b*::*t1-space*
assumes *cof*: *f constant-on S* **and** *contf*: *continuous-on* (*closure S*) *f*
shows *f constant-on* (*closure S*)
⟨*proof*⟩

Making a continuous function avoid some value in a neighbourhood.

lemma *continuous-within-avoid*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*t1-space*
assumes *continuous* (*at x within s*) *f*
and *f x* ≠ *a*
shows ∃ *e*>*0*. ∀ *y* ∈ *s*. *dist x y* < *e* −−> *f y* ≠ *a*
⟨*proof*⟩

lemma *continuous-at-avoid*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*t1-space*
assumes *continuous* (*at x*) *f*
and *f x* ≠ *a*
shows ∃ *e*>*0*. ∀ *y*. *dist x y* < *e* ⟶ *f y* ≠ *a*
⟨*proof*⟩

lemma *continuous-on-avoid*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*t1-space*
assumes *continuous-on s f*
and *x* ∈ *s*
and *f x* ≠ *a*
shows ∃ *e*>*0*. ∀ *y* ∈ *s*. *dist x y* < *e* ⟶ *f y* ≠ *a*
⟨*proof*⟩

lemma *continuous-on-open-avoid*:
fixes *f* :: *'a*::*metric-space* ⇒ *'b*::*t1-space*
assumes *continuous-on s f*
and *open s*
and *x* ∈ *s*
and *f x* ≠ *a*
shows ∃ *e*>*0*. ∀ *y*. *dist x y* < *e* ⟶ *f y* ≠ *a*
⟨*proof*⟩

Proving a function is constant by proving open-ness of level set.

lemma *continuous-levelset-openin-cases*:
fixes *f* :: - ⇒ *'b*::*t1-space*
shows *connected s* ⟹ *continuous-on s f* ⟹
openin (*subtopology euclidean s*) {*x* ∈ *s*. *f x* = *a*}
⟹ (∀ *x* ∈ *s*. *f x* ≠ *a*) ∨ (∀ *x* ∈ *s*. *f x* = *a*)
⟨*proof*⟩

lemma *continuous-levelset-openin*:

fixes f :: - $\Rightarrow$ *'b::t1-space*
shows *connected* $s \Longrightarrow$ *continuous-on* $s\ f \Longrightarrow$
$\quad$ *openin* (*subtopology euclidean* s) $\{x \in s.\ f\ x = a\} \Longrightarrow$
$\quad$ $(\exists\, x \in s.\ f\ x = a) \Longrightarrow (\forall\, x \in s.\ f\ x = a)$
⟨*proof*⟩

lemma *continuous-levelset-open*:
fixes f :: - $\Rightarrow$ *'b::t1-space*
assumes *connected* s
and *continuous-on* $s\ f$
and *open* $\{x \in s.\ f\ x = a\}$
and $\exists\, x \in s.\ \ f\ x = a$
shows $\forall\, x \in s.\ f\ x = a$
⟨*proof*⟩

Some arithmetical combinations (more to prove).

lemma *open-scaling*[*intro*]:
fixes s :: *'a::real-normed-vector set*
assumes $c \neq 0$
and *open* s
shows *open*$((\lambda x.\ c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *minus-image-eq-vimage*:
fixes A :: *'a::ab-group-add set*
shows $(\lambda x.\ -\ x)\ `\ A = (\lambda x.\ -\ x) -`\ A$
⟨*proof*⟩

lemma *open-negations*:
fixes s :: *'a::real-normed-vector set*
shows *open* $s \Longrightarrow$ *open* $((\lambda x.\ -\ x)\ `\ s)$
⟨*proof*⟩

lemma *open-translation*:
fixes s :: *'a::real-normed-vector set*
assumes *open* s
shows *open*$((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *open-affinity*:
fixes s :: *'a::real-normed-vector set*
assumes *open* $s\ \ c \neq 0$
shows *open* $((\lambda x.\ a + c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *interior-translation*:
fixes s :: *'a::real-normed-vector set*
shows *interior* $((\lambda x.\ a + x)\ `\ s) = (\lambda x.\ a + x)\ `\ (\mathit{interior}\ s)$
⟨*proof*⟩

Topological properties of linear functions.

lemma *linear-lim-0*:
assumes *bounded-linear f*
shows $(f \longrightarrow 0)\ (at\ (0))$
⟨*proof*⟩

lemma *linear-continuous-at*:
assumes *bounded-linear f*
shows *continuous* $(at\ a)\ f$
⟨*proof*⟩

lemma *linear-continuous-within*:
bounded-linear f $\Longrightarrow$ *continuous* $(at\ x\ within\ s)\ f$
⟨*proof*⟩

lemma *linear-continuous-on*:
bounded-linear f $\Longrightarrow$ *continuous-on s f*
⟨*proof*⟩

Also bilinear functions, in composition form.

lemma *bilinear-continuous-at-compose*:
continuous $(at\ x)\ f \Longrightarrow$ *continuous* $(at\ x)\ g \Longrightarrow$ *bounded-bilinear* $h \Longrightarrow$
continuous $(at\ x)\ (\lambda x.\ h\ (f\ x)\ (g\ x))$
⟨*proof*⟩

lemma *bilinear-continuous-within-compose*:
continuous $(at\ x\ within\ s)\ f \Longrightarrow$ *continuous* $(at\ x\ within\ s)\ g \Longrightarrow$ *bounded-bilinear* $h \Longrightarrow$
continuous $(at\ x\ within\ s)\ (\lambda x.\ h\ (f\ x)\ (g\ x))$
⟨*proof*⟩

lemma *bilinear-continuous-on-compose*:
continuous-on s f $\Longrightarrow$ *continuous-on s g* $\Longrightarrow$ *bounded-bilinear h* $\Longrightarrow$
continuous-on s $(\lambda x.\ h\ (f\ x)\ (g\ x))$
⟨*proof*⟩

Preservation of compactness and connectedness under continuous function.

lemma *compact-eq-openin-cover*:
compact S $\longleftrightarrow$
$(\forall C.\ (\forall c{\in}C.\ openin\ (subtopology\ euclidean\ S)\ c) \wedge S \subseteq \bigcup C \longrightarrow$
$(\exists D{\subseteq}C.\ finite\ D \wedge S \subseteq \bigcup D))$
⟨*proof*⟩

lemma *connected-continuous-image*:
assumes *continuous-on s f*
and *connected s*
shows *connected*$(f\ `\ s)$
⟨*proof*⟩

lemma *connected-linear-image*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes *linear* f **and** *connected* s
shows *connected* $(f \mathbin{`} s)$
⟨*proof*⟩

Continuity implies uniform continuity on a compact domain.

lemma *compact-uniformly-continuous*:
fixes $f :: 'a :: metric\text{-}space \Rightarrow 'b :: metric\text{-}space$
assumes f: *continuous-on* s f
and s: *compact* s
shows *uniformly-continuous-on* s f
⟨*proof*⟩

A uniformly convergent limit of continuous functions is continuous.

lemma *continuous-uniform-limit*:
fixes $f :: 'a \Rightarrow 'b::metric\text{-}space \Rightarrow 'c::metric\text{-}space$
assumes $\neg$ *trivial-limit* F
and *eventually* $(\lambda n.\ continuous\text{-}on\ s\ (f\ n))\ F$
and $\forall e{>}0.$ *eventually* $(\lambda n.\ \forall x{\in}s.\ dist\ (f\ n\ x)\ (g\ x) < e)\ F$
shows *continuous-on* s g
⟨*proof*⟩

16.27 Topological stuff lifted from and dropped to R

lemma *open-real*:
fixes $s :: real\ set$
shows $open\ s \longleftrightarrow (\forall x \in s.\ \exists e{>}0.\ \forall x'.\ |x' - x| < e \longrightarrow x' \in s)$
⟨*proof*⟩

lemma *islimpt-approachable-real*:
fixes $s :: real\ set$
shows $x\ islimpt\ s \longleftrightarrow (\forall e{>}0.\ \exists x'{\in}\ s.\ x' \neq x \wedge |x' - x| < e)$
⟨*proof*⟩

lemma *closed-real*:
fixes $s :: real\ set$
shows $closed\ s \longleftrightarrow (\forall x.\ (\forall e{>}0.\ \exists x' \in s.\ x' \neq x \wedge |x' - x| < e) \longrightarrow x \in s)$
⟨*proof*⟩

lemma *continuous-at-real-range*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow real$
shows $continuous\ (at\ x)\ f \longleftrightarrow (\forall e{>}0.\ \exists d{>}0.\ \forall x'.\ norm(x' - x) < d \longrightarrow |f\ x' - f\ x| < e)$
⟨*proof*⟩

lemma *continuous-on-real-range*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow real$
shows $continuous\text{-}on\ s\ f \longleftrightarrow$

$(\forall\, x \in s.\ \forall\, e{>}0.\ \exists\, d{>}0.\ (\forall\, x' \in s.\ norm(x' - x) < d \longrightarrow |f\, x' - f\, x| < e))$
⟨*proof*⟩

Hence some handy theorems on distance, diameter etc. of/from a set.

lemma *distance-attains-sup*:
assumes *compact s* $s \neq \{\}$
shows $\exists\, x{\in}s.\ \forall\, y{\in}s.$ *dist a y* $\leq$ *dist a x*
⟨*proof*⟩

For *minimal* distance, we only need closure, not compactness.

lemma *distance-attains-inf*:
fixes *a* :: *'a*::*heine-borel*
assumes *closed s* **and** $s \neq \{\}$
obtains *x* **where** $x{\in}s$ $\bigwedge y.\ y \in s \Longrightarrow$ *dist a x* $\leq$ *dist a y*
⟨*proof*⟩

16.28 Cartesian products

lemma *bounded-Times*:
assumes *bounded s bounded t*
shows *bounded* $(s \times t)$
⟨*proof*⟩

lemma *mem-Times-iff*: $x \in A \times B \longleftrightarrow$ *fst* $x \in A \wedge$ *snd* $x \in B$
⟨*proof*⟩

lemma *seq-compact-Times*: *seq-compact s* $\Longrightarrow$ *seq-compact t* $\Longrightarrow$ *seq-compact* $(s \times t)$
⟨*proof*⟩

lemma *compact-Times*:
assumes *compact s compact t*
shows *compact* $(s \times t)$
⟨*proof*⟩

Hence some useful properties follow quite easily.

lemma *compact-scaling*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *compact s*
shows *compact* $((\lambda x.\ c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *compact-negations*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *compact s*
shows *compact* $((\lambda x.\ -\ x)\ `\ s)$
⟨*proof*⟩

lemma *compact-sums*:

fixes $s\ t :: 'a$::*real-normed-vector set*
assumes *compact* s
and *compact* t
shows *compact* $\{x + y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *compact-differences*:
fixes $s\ t :: 'a$::*real-normed-vector set*
assumes *compact* s
and *compact* t
shows *compact* $\{x - y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *compact-translation*:
fixes $s :: 'a$::*real-normed-vector set*
assumes *compact* s
shows *compact* $((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *compact-affinity*:
fixes $s :: 'a$::*real-normed-vector set*
assumes *compact* s
shows *compact* $((\lambda x.\ a + c *_R x)\ `\ s)$
⟨*proof*⟩

Hence we get the following.

lemma *compact-sup-maxdistance*:
fixes $s :: 'a$::*metric-space set*
assumes *compact* s
and $s \neq \{\}$
shows $\exists x{\in}s.\ \exists y{\in}s.\ \forall u{\in}s.\ \forall v{\in}s.$ *dist* $u\ v \leq$ *dist* $x\ y$
⟨*proof*⟩

We can state this in terms of diameter of a set.

definition *diameter* $:: 'a$::*metric-space set* $\Rightarrow$ *real* **where**
diameter $S =$ (*if* $S = \{\}$ *then 0 else SUP* $(x,y){:}S{\times}S.$ *dist* $x\ y$)

lemma *diameter-bounded-bound*:
fixes $s :: 'a ::$ *metric-space set*
assumes s: *bounded* $s\ x \in s\ y \in s$
shows *dist* $x\ y \leq$ *diameter* s
⟨*proof*⟩

lemma *diameter-lower-bounded*:
fixes $s :: 'a ::$ *metric-space set*
assumes s: *bounded* s
and d: $0 < d\ d <$ *diameter* s
shows $\exists x{\in}s.\ \exists y{\in}s.\ d <$ *dist* $x\ y$
⟨*proof*⟩

lemma *diameter-bounded*:
assumes *bounded s*
shows $\forall\, x{\in}s.\ \forall\, y{\in}s.\ dist\ x\ y \leq diameter\ s$
and $\forall\, d{>}0.\ d < diameter\ s \longrightarrow (\exists\, x{\in}s.\ \exists\, y{\in}s.\ dist\ x\ y > d)$
⟨*proof*⟩

lemma *diameter-compact-attained*:
assumes *compact s*
and $s \neq \{\}$
shows $\exists\, x{\in}s.\ \exists\, y{\in}s.\ dist\ x\ y = diameter\ s$
⟨*proof*⟩

Related results with closure as the conclusion.

lemma *closed-scaling*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *closed* $((\lambda x.\ c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *closed-negations*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *closed* $((\lambda x.\ -x)\ `\ s)$
⟨*proof*⟩

lemma *compact-closed-sums*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *compact s* **and** *closed t*
shows *closed* $\{x + y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *closed-compact-sums*:
fixes *s t* :: *'a*::*real-normed-vector set*
assumes *closed s*
and *compact t*
shows *closed* $\{x + y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *compact-closed-differences*:
fixes *s t* :: *'a*::*real-normed-vector set*
assumes *compact s*
and *closed t*
shows *closed* $\{x - y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *closed-compact-differences*:
fixes *s t* :: *'a*::*real-normed-vector set*
assumes *closed s*

and *compact t*
shows *closed* $\{x - y \mid x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *closed-translation*:
fixes a :: *'a*::*real-normed-vector*
assumes *closed s*
shows *closed* $((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *translation-Compl*:
fixes a :: *'a*::*ab-group-add*
shows $(\lambda x.\ a + x)\ `\ (-\ t) = -\ ((\lambda x.\ a + x)\ `\ t)$
⟨*proof*⟩

lemma *translation-UNIV*:
fixes a :: *'a*::*ab-group-add*
shows *range* $(\lambda x.\ a + x) = UNIV$
⟨*proof*⟩

lemma *translation-diff*:
fixes a :: *'a*::*ab-group-add*
shows $(\lambda x.\ a + x)\ `\ (s - t) = ((\lambda x.\ a + x)\ `\ s) - ((\lambda x.\ a + x)\ `\ t)$
⟨*proof*⟩

lemma *closure-translation*:
fixes a :: *'a*::*real-normed-vector*
shows *closure* $((\lambda x.\ a + x)\ `\ s) = (\lambda x.\ a + x)\ `\ (closure\ s)$
⟨*proof*⟩

lemma *frontier-translation*:
fixes a :: *'a*::*real-normed-vector*
shows *frontier*$((\lambda x.\ a + x)\ `\ s) = (\lambda x.\ a + x)\ `\ (frontier\ s)$
⟨*proof*⟩

16.29 Separation between points and sets

lemma *separate-point-closed*:
fixes s :: *'a*::*heine-borel set*
assumes *closed s* **and** $a \notin s$
shows $\exists\, d{>}0.\ \forall\, x{\in}s.\ d \leq dist\ a\ x$
⟨*proof*⟩

lemma *separate-compact-closed*:
fixes $s\ t$ :: *'a*::*heine-borel set*
assumes *compact s*
and *t*: *closed t* $s \cap t = \{\}$
shows $\exists\, d{>}0.\ \forall\, x{\in}s.\ \forall\, y{\in}t.\ d \leq dist\ x\ y$
⟨*proof*⟩

lemma *separate-closed-compact*:
fixes *s t* :: *'a*::*heine-borel set*
assumes *closed s*
and *compact t*
and $s \cap t = \{\}$
shows $\exists\, d{>}0.\ \forall\, x{\in}s.\ \forall\, y{\in}t.\ d \leq dist\ x\ y$
⟨*proof*⟩

16.30 Closure of halfspaces and hyperplanes

lemma *isCont-open-vimage*:
assumes $\bigwedge x.\ isCont\ f\ x$
and *open s*
shows *open* $(f\ -`\ s)$
⟨*proof*⟩

lemma *isCont-closed-vimage*:
assumes $\bigwedge x.\ isCont\ f\ x$
and *closed s*
shows *closed* $(f\ -`\ s)$
⟨*proof*⟩

lemma *open-Collect-less*:
fixes *f g* :: *'a*::*t2-space* ⇒ *real*
assumes *f*: $\bigwedge x.\ isCont\ f\ x$
and *g*: $\bigwedge x.\ isCont\ g\ x$
shows *open* $\{x.\ f\ x < g\ x\}$
⟨*proof*⟩

lemma *closed-Collect-le*:
fixes *f g* :: *'a*::*t2-space* ⇒ *real*
assumes *f*: $\bigwedge x.\ isCont\ f\ x$
and *g*: $\bigwedge x.\ isCont\ g\ x$
shows *closed* $\{x.\ f\ x \leq g\ x\}$
⟨*proof*⟩

lemma *closed-Collect-eq*:
fixes *f g* :: *'a*::*t2-space* ⇒ *'b*::*t2-space*
assumes *f*: $\bigwedge x.\ isCont\ f\ x$
and *g*: $\bigwedge x.\ isCont\ g\ x$
shows *closed* $\{x.\ f\ x = g\ x\}$
⟨*proof*⟩

lemma *continuous-on-closed-Collect-le*:
fixes *f g* :: *'a*::*t2-space* ⇒ *real*
assumes *f*: *continuous-on s f* **and** *g*: *continuous-on s g* **and** *s*: *closed s*
shows *closed* $\{x \in s.\ f\ x \leq g\ x\}$
⟨*proof*⟩

lemma *continuous-at-inner*: *continuous* (*at x*) (*inner a*)
⟨*proof*⟩

lemma *closed-halfspace-le*: *closed* {*x*. *inner a x* ≤ *b*}
⟨*proof*⟩

lemma *closed-halfspace-ge*: *closed* {*x*. *inner a x* ≥ *b*}
⟨*proof*⟩

lemma *closed-hyperplane*: *closed* {*x*. *inner a x* = *b*}
⟨*proof*⟩

lemma *closed-halfspace-component-le*: *closed* {*x*::'*a*::*euclidean-space*. *x*•*i* ≤ *a*}
⟨*proof*⟩

lemma *closed-halfspace-component-ge*: *closed* {*x*::'*a*::*euclidean-space*. *x*•*i* ≥ *a*}
⟨*proof*⟩

lemma *closed-interval-left*:
fixes *b* :: '*a*::*euclidean-space*
shows *closed* {*x*::'*a*. ∀ *i*∈*Basis*. *x*•*i* ≤ *b*•*i*}
⟨*proof*⟩

lemma *closed-interval-right*:
fixes *a* :: '*a*::*euclidean-space*
shows *closed* {*x*::'*a*. ∀ *i*∈*Basis*. *a*•*i* ≤ *x*•*i*}
⟨*proof*⟩

lemma *continuous-le-on-closure*:
fixes *a*::*real*
assumes *f*: *continuous-on* (*closure s*) *f*
and *x*: *x* ∈ *closure*(*s*)
and *xlo*: ⋀*x*. *x* ∈ *s* ==> *f*(*x*) ≤ *a*
shows *f*(*x*) ≤ *a*
⟨*proof*⟩

lemma *continuous-ge-on-closure*:
fixes *a*::*real*
assumes *f*: *continuous-on* (*closure s*) *f*
and *x*: *x* ∈ *closure*(*s*)
and *xlo*: ⋀*x*. *x* ∈ *s* ==> *f*(*x*) ≥ *a*
shows *f*(*x*) ≥ *a*
⟨*proof*⟩

Openness of halfspaces.

lemma *open-halfspace-lt*: *open* {*x*. *inner a x* < *b*}
⟨*proof*⟩

lemma *open-halfspace-gt*: *open* $\{x.\ inner\ a\ x > b\}$
⟨*proof*⟩

lemma *open-halfspace-component-lt*: *open* $\{x::'a::euclidean\text{-}space.\ x\cdot i < a\}$
⟨*proof*⟩

lemma *open-halfspace-component-gt*: *open* $\{x::'a::euclidean\text{-}space.\ x\cdot i > a\}$
⟨*proof*⟩

This gives a simple derivation of limit component bounds.

lemma *Lim-component-le*:
fixes $f :: 'a \Rightarrow 'b::euclidean\text{-}space$
assumes $(f \longrightarrow l)\ net$
and $\neg\ (trivial\text{-}limit\ net)$
and $eventually\ (\lambda x.\ f(x)\cdot i \leq b)\ net$
shows $l\cdot i \leq b$
⟨*proof*⟩

lemma *Lim-component-ge*:
fixes $f :: 'a \Rightarrow 'b::euclidean\text{-}space$
assumes $(f \longrightarrow l)\ net$
and $\neg\ (trivial\text{-}limit\ net)$
and $eventually\ (\lambda x.\ b \leq (f\ x)\cdot i)\ net$
shows $b \leq l\cdot i$
⟨*proof*⟩

lemma *Lim-component-eq*:
fixes $f :: 'a \Rightarrow 'b::euclidean\text{-}space$
assumes *net*: $(f \longrightarrow l)\ net\ \neg\ trivial\text{-}limit\ net$
and *ev*:*eventually* $(\lambda x.\ f(x)\cdot i = b)\ net$
shows $l\cdot i = b$
⟨*proof*⟩

Limits relative to a union.

lemma *eventually-within-Un*:
$eventually\ P\ (at\ x\ within\ (s \cup t)) \longleftrightarrow$
$eventually\ P\ (at\ x\ within\ s) \wedge eventually\ P\ (at\ x\ within\ t)$
⟨*proof*⟩

lemma *Lim-within-union*:
$(f \longrightarrow l)\ (at\ x\ within\ (s \cup t)) \longleftrightarrow$
$(f \longrightarrow l)\ (at\ x\ within\ s) \wedge (f \longrightarrow l)\ (at\ x\ within\ t)$
⟨*proof*⟩

lemma *Lim-topological*:
$(f \longrightarrow l)\ net \longleftrightarrow$
$trivial\text{-}limit\ net \vee (\forall S.\ open\ S \longrightarrow l \in S \longrightarrow eventually\ (\lambda x.\ f\ x \in S)\ net)$
⟨*proof*⟩

Continuity relative to a union.

lemma *continuous-on-Un-local*:
⟦*closedin* (*subtopology euclidean* (*s* ∪ *t*)) *s*; *closedin* (*subtopology euclidean* (*s* ∪ *t*)) *t*;
continuous-on s f; *continuous-on t f*⟧
⟹ *continuous-on* (*s* ∪ *t*) *f*
⟨*proof*⟩

lemma *continuous-on-cases-local*:
⟦*closedin* (*subtopology euclidean* (*s* ∪ *t*)) *s*; *closedin* (*subtopology euclidean* (*s* ∪ *t*)) *t*;
continuous-on s f; *continuous-on t g*;
⋀*x*. ⟦*x* ∈ *s* ∧ ~*P x* ∨ *x* ∈ *t* ∧ *P x*⟧ ⟹ *f x* = *g x*⟧
⟹ *continuous-on* (*s* ∪ *t*) (λ*x*. *if P x then f x else g x*)
⟨*proof*⟩

lemma *continuous-on-cases-le*:
fixes *h* :: *'a* :: *topological-space* ⇒ *real*
assumes *continuous-on* {*t* ∈ *s*. *h t* ≤ *a*} *f*
and *continuous-on* {*t* ∈ *s*. *a* ≤ *h t*} *g*
and *h*: *continuous-on s h*
and ⋀*t*. ⟦*t* ∈ *s*; *h t* = *a*⟧ ⟹ *f t* = *g t*
shows *continuous-on s* (λ*t*. *if h t* ≤ *a then f*(*t*) *else g*(*t*))
⟨*proof*⟩

lemma *continuous-on-cases-1*:
fixes *s* :: *real set*
assumes *continuous-on* {*t* ∈ *s*. *t* ≤ *a*} *f*
and *continuous-on* {*t* ∈ *s*. *a* ≤ *t*} *g*
and *a* ∈ *s* ⟹ *f a* = *g a*
shows *continuous-on s* (λ*t*. *if t* ≤ *a then f*(*t*) *else g*(*t*))
⟨*proof*⟩

Some more convenient intermediate-value theorem formulations.

lemma *connected-ivt-hyperplane*:
assumes *connected s*
and *x* ∈ *s*
and *y* ∈ *s*
and *inner a x* ≤ *b*
and *b* ≤ *inner a y*
shows ∃ *z* ∈ *s*. *inner a z* = *b*
⟨*proof*⟩

lemma *connected-ivt-component*:
fixes *x*::*'a*::*euclidean-space*
shows *connected s* ⟹
x ∈ *s* ⟹ *y* ∈ *s* ⟹
x•*k* ≤ *a* ⟹ *a* ≤ *y*•*k* ⟹ (∃ *z*∈*s*. *z*•*k* = *a*)
⟨*proof*⟩

16.31 Intervals

lemma *open-box*[*intro*]: *open* (*box a b*)
⟨*proof*⟩

instance *euclidean-space* ⊆ *second-countable-topology*
⟨*proof*⟩

instance *euclidean-space* ⊆ *polish-space* ⟨*proof*⟩

lemma *closed-cbox*[*intro*]:
fixes *a b* :: *'a*::*euclidean-space*
shows *closed* (*cbox a b*)
⟨*proof*⟩

lemma *interior-cbox* [*simp*]:
fixes *a b* :: *'a*::*euclidean-space*
shows *interior* (*cbox a b*) = *box a b* (**is** *?L* = *?R*)
⟨*proof*⟩

lemma *bounded-cbox*:
fixes *a* :: *'a*::*euclidean-space*
shows *bounded* (*cbox a b*)
⟨*proof*⟩

lemma *bounded-box* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *bounded* (*box a b*)
⟨*proof*⟩

lemma *not-interval-UNIV* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *cbox a b* ≠ *UNIV box a b* ≠ *UNIV*
⟨*proof*⟩

lemma *compact-cbox* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *compact* (*cbox a b*)
⟨*proof*⟩

lemma *box-midpoint*:
fixes *a* :: *'a*::*euclidean-space*
assumes *box a b* ≠ {}
shows ((*1*/*2*) *∗*$_R$ (*a* + *b*)) ∈ *box a b*
⟨*proof*⟩

lemma *open-cbox-convex*:
fixes *x* :: *'a*::*euclidean-space*
assumes *x*: *x* ∈ *box a b*
and *y*: *y* ∈ *cbox a b*

and *e*: *0* < *e e* ≤ *1*
shows (*e* *$_R$ *x* + (*1* − *e*) *$_R$ *y*) ∈ *box a b*
⟨*proof*⟩

lemma *closure-box*:
fixes *a* :: *'a*::*euclidean-space*
assumes *box a b* ≠ {}
shows *closure* (*box a b*) = *cbox a b*
⟨*proof*⟩

lemma *bounded-subset-box-symmetric*:
fixes *s*::(*'a*::*euclidean-space*) *set*
assumes *bounded s*
shows ∃ *a*. *s* ⊆ *box* (−*a*) *a*
⟨*proof*⟩

lemma *bounded-subset-open-interval*:
fixes *s* :: (*'a*::*euclidean-space*) *set*
shows *bounded s* ⟹ (∃ *a b*. *s* ⊆ *box a b*)
⟨*proof*⟩

lemma *bounded-subset-cbox-symmetric*:
fixes *s* :: (*'a*::*euclidean-space*) *set*
assumes *bounded s*
shows ∃ *a*. *s* ⊆ *cbox* (−*a*) *a*
⟨*proof*⟩

lemma *bounded-subset-cbox*:
fixes *s* :: (*'a*::*euclidean-space*) *set*
shows *bounded s* ⟹ ∃ *a b*. *s* ⊆ *cbox a b*
⟨*proof*⟩

lemma *frontier-cbox*:
fixes *a b* :: *'a*::*euclidean-space*
shows *frontier* (*cbox a b*) = *cbox a b* − *box a b*
⟨*proof*⟩

lemma *frontier-box*:
fixes *a b* :: *'a*::*euclidean-space*
shows *frontier* (*box a b*) = (*if box a b* = {} *then* {} *else cbox a b* − *box a b*)
⟨*proof*⟩

lemma *inter-interval-mixed-eq-empty*:
fixes *a* :: *'a*::*euclidean-space*
assumes *box c d* ≠ {}
shows *box a b* ∩ *cbox c d* = {} ⟷ *box a b* ∩ *box c d* = {}
⟨*proof*⟩

lemma *diameter-cbox*:

fixes $a\ b$::$'a$::*euclidean-space*
shows $(\forall i \in Basis.\ a \cdot i \leq b \cdot i) \Longrightarrow diameter\ (cbox\ a\ b) = dist\ a\ b$
⟨*proof*⟩

lemma *eucl-less-eq-halfspaces*:
fixes a :: $'a$::*euclidean-space*
shows $\{x.\ x <e\ a\} = (\bigcap i{\in}Basis.\ \{x.\ x \cdot i < a \cdot i\})$
$\{x.\ a <e\ x\} = (\bigcap i{\in}Basis.\ \{x.\ a \cdot i < x \cdot i\})$
⟨*proof*⟩

lemma *eucl-le-eq-halfspaces*:
fixes a :: $'a$::*euclidean-space*
shows $\{x.\ \forall i{\in}Basis.\ x \cdot i \leq a \cdot i\} = (\bigcap i{\in}Basis.\ \{x.\ x \cdot i \leq a \cdot i\})$
$\{x.\ \forall i{\in}Basis.\ a \cdot i \leq x \cdot i\} = (\bigcap i{\in}Basis.\ \{x.\ a \cdot i \leq x \cdot i\})$
⟨*proof*⟩

lemma *open-Collect-eucl-less*[*simp*, *intro*]:
fixes a :: $'a$::*euclidean-space*
shows *open* $\{x.\ x <e\ a\}$
open $\{x.\ a <e\ x\}$
⟨*proof*⟩

lemma *closed-Collect-eucl-le*[*simp*, *intro*]:
fixes a :: $'a$::*euclidean-space*
shows *closed* $\{x.\ \forall i{\in}Basis.\ a \cdot i \leq x \cdot i\}$
closed $\{x.\ \forall i{\in}Basis.\ x \cdot i \leq a \cdot i\}$
⟨*proof*⟩

lemma *image-affinity-cbox*: **fixes** m::*real*
fixes $a\ b\ c$ :: $'a$::*euclidean-space*
shows $(\lambda x.\ m *_R x + c)$ ‘ $cbox\ a\ b =$
(*if* $cbox\ a\ b = \{\}$ *then* $\{\}$
else (*if* $0 \leq m$ *then* $cbox\ (m *_R a + c)\ (m *_R b + c)$
else $cbox\ (m *_R b + c)\ (m *_R a + c)$)))
⟨*proof*⟩

lemma *image-smult-cbox*:$(\lambda x.\ m *_R (x$::-::*euclidean-space*$))$ ‘ $cbox\ a\ b =$
(*if* $cbox\ a\ b = \{\}$ *then* $\{\}$ *else if* $0 \leq m$ *then* $cbox\ (m *_R a)\ (m *_R b)$ *else* $cbox\ (m *_R b)\ (m *_R a)$)
⟨*proof*⟩

lemma *islimpt-greaterThanLessThan1*:
fixes $a\ b$::$'a$::{*linorder-topology*, *dense-order*}
assumes $a < b$
shows a *islimpt* $\{a<..<b\}$
⟨*proof*⟩

lemma *islimpt-greaterThanLessThan2*:
fixes $a\ b$::$'a$::{*linorder-topology*, *dense-order*}

assumes $a < b$
shows b *islimpt* $\{a<..<b\}$
⟨*proof*⟩

lemma *closure-greaterThanLessThan*[*simp*]:
fixes a b::$'a$::$\{$*linorder-topology*, *dense-order*$\}$
shows $a < b \Longrightarrow$ *closure* $\{a <..< b\} = \{a\ ..\ b\}$ (**is** - $\Longrightarrow$ *?l* = *?r*)
⟨*proof*⟩

lemma *closure-greaterThan*[*simp*]:
fixes a b::$'a$::$\{$*no-top*, *linorder-topology*, *dense-order*$\}$
shows *closure* $\{a<..\} = \{a..\}$
⟨*proof*⟩

lemma *closure-lessThan*[*simp*]:
fixes b::$'a$::$\{$*no-bot*, *linorder-topology*, *dense-order*$\}$
shows *closure* $\{..<b\} = \{..b\}$
⟨*proof*⟩

lemma *closure-atLeastLessThan*[*simp*]:
fixes a b::$'a$::$\{$*linorder-topology*, *dense-order*$\}$
assumes $a < b$
shows *closure* $\{a\ ..<\ b\} = \{a\ ..\ b\}$
⟨*proof*⟩

lemma *closure-greaterThanAtMost*[*simp*]:
fixes a b::$'a$::$\{$*linorder-topology*, *dense-order*$\}$
assumes $a < b$
shows *closure* $\{a <..\ b\} = \{a\ ..\ b\}$
⟨*proof*⟩

16.32 Homeomorphisms

definition *homeomorphism* s t f g $\longleftrightarrow$
$(\forall x{\in}s.\ (g(f\ x) = x)) \wedge (f\ `\ s = t) \wedge$ *continuous-on* s f $\wedge$
$(\forall y{\in}t.\ (f(g\ y) = y)) \wedge (g\ `\ t = s) \wedge$ *continuous-on* t g

lemma *homeomorphism-translation*:
fixes a :: $'a$ :: *real-normed-vector*
shows *homeomorphism* $(op\ +\ a\ `\ S)\ S\ (op\ +\ (-\ a))\ (op\ +\ a)$
⟨*proof*⟩

lemma *homeomorphism-symD*: *homeomorphism* S t f $g \Longrightarrow$ *homeomorphism* t S g f
⟨*proof*⟩

lemma *homeomorphism-sym*: *homeomorphism* S t f g = *homeomorphism* t S g f
⟨*proof*⟩

definition *homeomorphic* :: *'a::topological-space set* ⇒ *'b::topological-space set* ⇒ *bool*
(**infixr** *homeomorphic 60*)
where *s homeomorphic t* ≡ (∃*f g*. *homeomorphism s t f g*)

lemma *homeomorphic-refl*: *s homeomorphic s*
⟨*proof*⟩

lemma *homeomorphic-sym*: *s homeomorphic t* ⟷ *t homeomorphic s*
⟨*proof*⟩

lemma *homeomorphic-trans* [*trans*]:
assumes *s homeomorphic t*
and *t homeomorphic u*
shows *s homeomorphic u*
⟨*proof*⟩

lemma *homeomorphic-minimal*:
s homeomorphic t ⟷
(∃*f g*. (∀ *x*∈*s*. *f*(*x*) ∈ *t* ∧ (*g*(*f*(*x*)) = *x*)) ∧
(∀ *y*∈*t*. *g*(*y*) ∈ *s* ∧ (*f*(*g*(*y*)) = *y*)) ∧
continuous-on s f ∧ *continuous-on t g*)
⟨*proof*⟩

Relatively weak hypotheses if a set is compact.

lemma *homeomorphism-compact*:
fixes *f* :: *'a::topological-space* ⇒ *'b::t2-space*
assumes *compact s continuous-on s f f* ‘ *s* = *t inj-on f s*
shows ∃*g*. *homeomorphism s t f g*
⟨*proof*⟩

lemma *homeomorphic-compact*:
fixes *f* :: *'a::topological-space* ⇒ *'b::t2-space*
shows *compact s* ⟹ *continuous-on s f* ⟹ (*f* ‘ *s* = *t*) ⟹ *inj-on f s* ⟹ *s homeomorphic t*
⟨*proof*⟩

Preservation of topological properties.

lemma *homeomorphic-compactness*: *s homeomorphic t* ⟹ (*compact s* ⟷ *compact t*)
⟨*proof*⟩

Results on translation, scaling etc.

lemma *homeomorphic-scaling*:
fixes *s* :: *'a::real-normed-vector set*
assumes *c* ≠ *0*
shows *s homeomorphic* ((λ*x*. *c* ∗$_R$ *x*) ‘ *s*)
⟨*proof*⟩

lemma *homeomorphic-translation*:
fixes s :: *'a::real-normed-vector set*
shows s *homeomorphic* $((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *homeomorphic-affinity*:
fixes s :: *'a::real-normed-vector set*
assumes $c \neq 0$
shows s *homeomorphic* $((\lambda x.\ a + c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *homeomorphic-balls*:
fixes $a\ b$ ::*'a::real-normed-vector*
assumes $0 < d\ \ 0 < e$
shows $(ball\ a\ d)$ *homeomorphic* $(ball\ b\ e)$ (**is** *?th*)
and $(cball\ a\ d)$ *homeomorphic* $(cball\ b\ e)$ (**is** *?cth*)
⟨*proof*⟩

"Isometry" (up to constant bounds) of injective linear map etc.

lemma *cauchy-isometric*:
assumes e: $e > 0$
and s: *subspace s*
and f: *bounded-linear f*
and *normf*: $\forall x{\in}s.\ norm\ (f\ x) \geq e * norm\ x$
and *xs*: $\forall n.\ x\ n \in s$
and *cf*: *Cauchy* $(f \circ x)$
shows *Cauchy x*
⟨*proof*⟩

lemma *complete-isometric-image*:
assumes $0 < e$
and s: *subspace s*
and f: *bounded-linear f*
and *normf*: $\forall x{\in}s.\ norm(f\ x) \geq e * norm(x)$
and *cs*: *complete s*
shows *complete* $(f\ `\ s)$
⟨*proof*⟩

lemma *injective-imp-isometric*:
fixes f :: *'a::euclidean-space* ⇒ *'b::euclidean-space*
assumes s: *closed s subspace s*
and f: *bounded-linear f* $\forall x{\in}s.\ f\ x = 0 \longrightarrow x = 0$
shows $\exists\ e{>}0.\ \forall x{\in}s.\ norm\ (f\ x) \geq e * norm\ x$
⟨*proof*⟩

lemma *closed-injective-image-subspace*:
fixes f :: *'a::euclidean-space* ⇒ *'b::euclidean-space*
assumes *subspace s bounded-linear f* $\forall x{\in}s.\ f\ x = 0 \longrightarrow x = 0$ *closed s*
shows *closed*$(f\ `\ s)$

⟨*proof*⟩

16.33 Some properties of a canonical subspace

lemma *subspace-substandard*:
subspace {*x*::'*a*::*euclidean-space*. (∀ *i*∈*Basis*. *P i* ⟶ *x*•*i* = *0*)}
⟨*proof*⟩

lemma *closed-substandard*:
closed {*x*::'*a*::*euclidean-space*. ∀ *i*∈*Basis*. *P i* −−> *x*•*i* = *0*} (**is** *closed ?A*)
⟨*proof*⟩

lemma *dim-substandard*:
assumes *d*: *d* ⊆ *Basis*
shows *dim* {*x*::'*a*::*euclidean-space*. ∀ *i*∈*Basis*. *i* ∉ *d* ⟶ *x*•*i* = *0*} = *card d* (**is** *dim ?A* = -)
⟨*proof*⟩

Hence closure and completeness of all subspaces.

lemma *ex-card*:
assumes *n* ≤ *card A*
shows ∃ *S*⊆*A*. *card S* = *n*
⟨*proof*⟩

lemma *closed-subspace*:
fixes *s* :: '*a*::*euclidean-space set*
assumes *subspace s*
shows *closed s*
⟨*proof*⟩

lemma *complete-subspace*:
fixes *s* :: ('*a*::*euclidean-space*) *set*
shows *subspace s* ⟹ *complete s*
⟨*proof*⟩

lemma *dim-closure*:
fixes *s* :: ('*a*::*euclidean-space*) *set*
shows *dim*(*closure s*) = *dim s* (**is** *?dc* = *?d*)
⟨*proof*⟩

16.34 Affine transformations of intervals

lemma *real-affinity-le*:
0 < (*m*::'*a*::*linordered-field*) ⟹ (*m* ∗ *x* + *c* ≤ *y* ⟷ *x* ≤ *inverse*(*m*) ∗ *y* + −(*c* / *m*))
⟨*proof*⟩

lemma *real-le-affinity*:
0 < (*m*::'*a*::*linordered-field*) ⟹ (*y* ≤ *m* ∗ *x* + *c* ⟷ *inverse*(*m*) ∗ *y* + −(*c* / *m*) ≤ *x*)

⟨*proof*⟩

lemma *real-affinity-lt*:
$0 < (m::'a::linordered\text{-}field) \Longrightarrow (m * x + c < y \longleftrightarrow x < inverse(m) * y + -(c / m))$
⟨*proof*⟩

lemma *real-lt-affinity*:
$0 < (m::'a::linordered\text{-}field) \Longrightarrow (y < m * x + c \longleftrightarrow inverse(m) * y + -(c / m) < x)$
⟨*proof*⟩

lemma *real-affinity-eq*:
$(m::'a::linordered\text{-}field) \neq 0 \Longrightarrow (m * x + c = y \longleftrightarrow x = inverse(m) * y + -(c / m))$
⟨*proof*⟩

lemma *real-eq-affinity*:
$(m::'a::linordered\text{-}field) \neq 0 \Longrightarrow (y = m * x + c \longleftrightarrow inverse(m) * y + -(c / m) = x)$
⟨*proof*⟩

16.35 Banach fixed point theorem (not really topological...)

theorem *banach-fix*:
assumes *s*: *complete s* $s \neq \{\}$
and *c*: $0 \leq c$ $c < 1$
and *f*: $(f \text{ ` } s) \subseteq s$
and *lipschitz*: $\forall x{\in}s.\ \forall y{\in}s.\ dist\ (f\ x)\ (f\ y) \leq c * dist\ x\ y$
shows $\exists!x{\in}s.\ f\ x = x$
⟨*proof*⟩

16.36 Edelstein fixed point theorem

theorem *edelstein-fix*:
fixes *s* :: *'a::metric-space set*
assumes *s*: *compact s* $s \neq \{\}$
and *gs*: $(g \text{ ` } s) \subseteq s$
and *dist*: $\forall x{\in}s.\ \forall y{\in}s.\ x \neq y \longrightarrow dist\ (g\ x)\ (g\ y) < dist\ x\ y$
shows $\exists!x{\in}s.\ g\ x = x$
⟨*proof*⟩

lemma *cball-subset-cball-iff*:
fixes *a* :: *'a* :: *euclidean-space*
shows *cball a r* $\subseteq$ *cball a' r'* $\longleftrightarrow$ *dist a a'* $+ r \leq r' \vee r < 0$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *cball-subset-ball-iff*:

fixes $a :: 'a :: euclidean\text{-}space$
shows $cball\ a\ r \subseteq ball\ a'\ r' \longleftrightarrow dist\ a\ a' + r < r' \vee r < 0$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *ball-subset-cball-iff*:
fixes $a :: 'a :: euclidean\text{-}space$
shows $ball\ a\ r \subseteq cball\ a'\ r' \longleftrightarrow dist\ a\ a' + r \leq r' \vee r \leq 0$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *ball-subset-ball-iff*:
fixes $a :: 'a :: euclidean\text{-}space$
shows $ball\ a\ r \subseteq ball\ a'\ r' \longleftrightarrow dist\ a\ a' + r \leq r' \vee r \leq 0$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *ball-eq-ball-iff*:
fixes $x :: 'a :: euclidean\text{-}space$
shows $ball\ x\ d = ball\ y\ e \longleftrightarrow d \leq 0 \wedge e \leq 0 \vee x{=}y \wedge d{=}e$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *cball-eq-cball-iff*:
fixes $x :: 'a :: euclidean\text{-}space$
shows $cball\ x\ d = cball\ y\ e \longleftrightarrow d < 0 \wedge e < 0 \vee x{=}y \wedge d{=}e$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *ball-eq-cball-iff*:
fixes $x :: 'a :: euclidean\text{-}space$
shows $ball\ x\ d = cball\ y\ e \longleftrightarrow d \leq 0 \wedge e < 0$ (**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *cball-eq-ball-iff*:
fixes $x :: 'a :: euclidean\text{-}space$
shows $cball\ x\ d = ball\ y\ e \longleftrightarrow d < 0 \wedge e \leq 0$
⟨*proof*⟩

no-notation
eucl-less (**infix** $<e$ *50*)

end

17 Convexity in real vector spaces

theory *Convex*
imports *Product-Vector*

begin

17.1 Convexity

definition *convex* :: *'a::real-vector set* ⇒ *bool*
  where *convex s* ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. ∀ *u*≥*0*. ∀ *v*≥*0*. *u* + *v* = *1* ⟶ *u* *$_R$ *x* + *v* *$_R$ *y* ∈ *s*)

lemma *convexI*:
  assumes ⋀*x y u v*. *x* ∈ *s* ⟹ *y* ∈ *s* ⟹ *0* ≤ *u* ⟹ *0* ≤ *v* ⟹ *u* + *v* = *1* ⟹ *u* *$_R$ *x* + *v* *$_R$ *y* ∈ *s*
  shows *convex s*
  ⟨*proof*⟩

lemma *convexD*:
  assumes *convex s* **and** *x* ∈ *s* **and** *y* ∈ *s* **and** *0* ≤ *u* **and** *0* ≤ *v* **and** *u* + *v* = *1*
  shows *u* *$_R$ *x* + *v* *$_R$ *y* ∈ *s*
  ⟨*proof*⟩

lemma *convex-alt*:
  convex s ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. ∀ *u*. *0* ≤ *u* ∧ *u* ≤ *1* ⟶ ((*1* − *u*) *$_R$ *x* + *u* *$_R$ *y*) ∈ *s*)
  (**is** - ⟷ *?alt*)
⟨*proof*⟩

lemma *convexD-alt*:
  assumes *convex s a* ∈ *s b* ∈ *s 0* ≤ *u u* ≤ *1*
  shows ((*1* − *u*) *$_R$ *a* + *u* *$_R$ *b*) ∈ *s*
  ⟨*proof*⟩

lemma *mem-convex-alt*:
  assumes *convex S x* ∈ *S y* ∈ *S u* ≥ *0 v* ≥ *0 u* + *v* > *0*
  shows ((*u*/(*u*+*v*)) *$_R$ *x* + (*v*/(*u*+*v*)) *$_R$ *y*) ∈ *S*
  ⟨*proof*⟩

lemma *convex-empty*[*intro*,*simp*]: *convex* {}
  ⟨*proof*⟩

lemma *convex-singleton*[*intro*,*simp*]: *convex* {*a*}
  ⟨*proof*⟩

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

lemma *convex-Inter*: (∀ *s*∈*f*. *convex s*) ⟹ *convex*(⋂*f*)
  ⟨*proof*⟩

lemma *convex-Int*: *convex s* ⟹ *convex t* ⟹ *convex* (*s* ∩ *t*)
  ⟨*proof*⟩

lemma *convex-INT*: $\forall\, i{\in}A.\ convex\ (B\ i) \Longrightarrow convex\ (\bigcap i{\in}A.\ B\ i)$
⟨*proof*⟩

lemma *convex-Times*: $convex\ s \Longrightarrow convex\ t \Longrightarrow convex\ (s \times t)$
⟨*proof*⟩

lemma *convex-halfspace-le*: $convex\ \{x.\ inner\ a\ x \leq b\}$
⟨*proof*⟩

lemma *convex-halfspace-ge*: $convex\ \{x.\ inner\ a\ x \geq b\}$
⟨*proof*⟩

lemma *convex-hyperplane*: $convex\ \{x.\ inner\ a\ x = b\}$
⟨*proof*⟩

lemma *convex-halfspace-lt*: $convex\ \{x.\ inner\ a\ x < b\}$
⟨*proof*⟩

lemma *convex-halfspace-gt*: $convex\ \{x.\ inner\ a\ x > b\}$
⟨*proof*⟩

lemma *convex-real-interval* [*iff*]:
fixes $a\ b :: real$
shows $convex\ \{a..\}$ **and** $convex\ \{..b\}$
and $convex\ \{a<..\}$ **and** $convex\ \{..<b\}$
and $convex\ \{a..b\}$ **and** $convex\ \{a<..b\}$
and $convex\ \{a..<b\}$ **and** $convex\ \{a<..<b\}$
⟨*proof*⟩

lemma *convex-Reals*: $convex\ \mathbb{R}$
⟨*proof*⟩

17.2 Explicit expressions for convexity in terms of arbitrary sums

lemma *convex-setsum*:
fixes $C :: 'a{::}real\text{-}vector\ set$
assumes *finite s*
and *convex C*
and $(\sum\ i \in s.\ a\ i) = 1$
assumes $\bigwedge i.\ i \in s \Longrightarrow a\ i \geq 0$
and $\bigwedge i.\ i \in s \Longrightarrow y\ i \in C$
shows $(\sum\ j \in s.\ a\ j *_R y\ j) \in C$
⟨*proof*⟩

lemma *convex*:
$convex\ s \longleftrightarrow (\forall\, (k{::}nat)\ u\ x.\ (\forall\, i.\ 1{\leq}i \wedge i{\leq}k \longrightarrow 0 \leq u\ i \wedge x\ i \in s) \wedge (setsum\ u\ \{1..k\} = 1)$

$\longrightarrow$ *setsum* $(\lambda i.\ u\ i *_R x\ i)\ \{1..k\} \in s)$
⟨*proof*⟩

lemma *convex-explicit*:
fixes s :: *'a::real-vector set*
shows *convex* $s \longleftrightarrow$
$(\forall t\ u.$ *finite* $t \wedge t \subseteq s \wedge (\forall x{\in}t.\ 0 \leq u\ x) \wedge$ *setsum* $u\ t = 1 \longrightarrow$ *setsum* $(\lambda x.\ u\ x *_R x)\ t \in s)$
⟨*proof*⟩

lemma *convex-finite*:
assumes *finite* s
shows *convex* $s \longleftrightarrow (\forall u.\ (\forall x{\in}s.\ 0 \leq u\ x) \wedge$ *setsum* $u\ s = 1 \longrightarrow$ *setsum* $(\lambda x.\ u\ x *_R x)\ s \in s)$
⟨*proof*⟩

17.3 Functions that are convex on a set

definition *convex-on* :: *'a::real-vector set* ⇒ *('a* ⇒ *real)* ⇒ *bool*
where *convex-on* $s\ f \longleftrightarrow$
$(\forall x{\in}s.\ \forall y{\in}s.\ \forall u{\geq}0.\ \forall v{\geq}0.\ u + v = 1 \longrightarrow f\ (u *_R x + v *_R y) \leq u * f\ x + v * f\ y)$

lemma *convex-onI* [*intro?*]:
assumes $\bigwedge t\ x\ y.\ t > 0 \Longrightarrow t < 1 \Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow$
$f\ ((1 - t) *_R x + t *_R y) \leq (1 - t) * f\ x + t * f\ y$
shows *convex-on* $A\ f$
⟨*proof*⟩

lemma *convex-on-linorderI* [*intro?*]:
fixes A :: *('a::{linorder,real-vector}) set*
assumes $\bigwedge t\ x\ y.\ t > 0 \Longrightarrow t < 1 \Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow x < y \Longrightarrow$
$f\ ((1 - t) *_R x + t *_R y) \leq (1 - t) * f\ x + t * f\ y$
shows *convex-on* $A\ f$
⟨*proof*⟩

lemma *convex-onD*:
assumes *convex-on* $A\ f$
shows $\bigwedge t\ x\ y.\ t \geq 0 \Longrightarrow t \leq 1 \Longrightarrow x \in A \Longrightarrow y \in A \Longrightarrow$
$f\ ((1 - t) *_R x + t *_R y) \leq (1 - t) * f\ x + t * f\ y$
⟨*proof*⟩

lemma *convex-onD-Icc*:
assumes *convex-on* $\{x..y\}\ f\ x \leq (y$ :: - :: *{real-vector,preorder}*$)$
shows $\bigwedge t.\ t \geq 0 \Longrightarrow t \leq 1 \Longrightarrow$
$f\ ((1 - t) *_R x + t *_R y) \leq (1 - t) * f\ x + t * f\ y$
⟨*proof*⟩

lemma *convex-on-subset*: *convex-on t f* $\Longrightarrow$ $s \subseteq t$ $\Longrightarrow$ *convex-on s f*
⟨*proof*⟩

lemma *convex-on-add* [*intro*]:
assumes *convex-on s f*
and *convex-on s g*
shows *convex-on s* ($\lambda x.$ *f x* + *g x*)
⟨*proof*⟩

lemma *convex-on-cmul* [*intro*]:
fixes *c* :: *real*
assumes $0 \leq c$
and *convex-on s f*
shows *convex-on s* ($\lambda x.$ *c* ∗ *f x*)
⟨*proof*⟩

lemma *convex-lower*:
assumes *convex-on s f*
and $x \in s$
and $y \in s$
and $0 \leq u$
and $0 \leq v$
and $u + v = 1$
shows *f* (*u* $*_R$ *x* + *v* $*_R$ *y*) ≤ *max* (*f x*) (*f y*)
⟨*proof*⟩

lemma *convex-on-dist* [*intro*]:
fixes *s* :: *'a*::*real-normed-vector set*
shows *convex-on s* ($\lambda x.$ *dist a x*)
⟨*proof*⟩

17.4 Arithmetic operations on sets preserve convexity

lemma *convex-linear-image*:
assumes *linear f*
and *convex s*
shows *convex* (*f* ‘ *s*)
⟨*proof*⟩

lemma *convex-linear-vimage*:
assumes *linear f*
and *convex s*
shows *convex* (*f* −‘ *s*)
⟨*proof*⟩

lemma *convex-scaling*:
assumes *convex s*
shows *convex* (($\lambda x.$ *c* $*_R$ *x*) ‘ *s*)
⟨*proof*⟩

lemma *convex-scaled*:
assumes *convex s*
shows *convex* $((\lambda x.\ x *_R c)\ `\ s)$
⟨*proof*⟩

lemma *convex-negations*:
assumes *convex s*
shows *convex* $((\lambda x.\ -\ x)\ `\ s)$
⟨*proof*⟩

lemma *convex-sums*:
assumes *convex s*
and *convex t*
shows *convex* $\{x + y |\ x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *convex-differences*:
assumes *convex s convex t*
shows *convex* $\{x - y |\ x\ y.\ x \in s \wedge y \in t\}$
⟨*proof*⟩

lemma *convex-translation*:
assumes *convex s*
shows *convex* $((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *convex-affinity*:
assumes *convex s*
shows *convex* $((\lambda x.\ a + c *_R x)\ `\ s)$
⟨*proof*⟩

lemma *pos-is-convex*: *convex* $\{0 :: real <..\}$
⟨*proof*⟩

lemma *convex-on-setsum*:
fixes $a :: 'a \Rightarrow real$
and $y :: 'a \Rightarrow 'b$::*real-vector*
and $f :: 'b \Rightarrow real$
assumes *finite s* $s \neq \{\}$
and *convex-on C f*
and *convex C*
and $(\sum\ i \in s.\ a\ i) = 1$
and $\bigwedge i.\ i \in s \Longrightarrow a\ i \geq 0$
and $\bigwedge i.\ i \in s \Longrightarrow y\ i \in C$
shows $f\ (\sum\ i \in s.\ a\ i *_R y\ i) \leq (\sum\ i \in s.\ a\ i * f\ (y\ i))$
⟨*proof*⟩

lemma *convex-on-alt*:

fixes C :: *'a::real-vector set*
assumes *convex* C
shows *convex-on* $C\ f \longleftrightarrow$
$(\forall\, x \in C.\ \forall\ y \in C.\ \forall\ \mu :: real.\ \mu \geq 0 \wedge \mu \leq 1 \longrightarrow$
$f\ (\mu *_R x + (1 - \mu) *_R y) \leq \mu * f\ x + (1 - \mu) * f\ y)$
⟨*proof*⟩

lemma *convex-on-diff*:
fixes f :: *real* ⇒ *real*
assumes f: *convex-on* $I\ f$
and I: $x \in I\ y \in I$
and t: $x < t\ t < y$
shows $(f\ x - f\ t) / (x - t) \leq (f\ x - f\ y) / (x - y)$
and $(f\ x - f\ y) / (x - y) \leq (f\ t - f\ y) / (t - y)$
⟨*proof*⟩

lemma *pos-convex-function*:
fixes f :: *real* ⇒ *real*
assumes *convex* C
and *leq*: $\bigwedge x\ y.\ x \in C \Longrightarrow y \in C \Longrightarrow f'\ x * (y - x) \leq f\ y - f\ x$
shows *convex-on* $C\ f$
⟨*proof*⟩

lemma *atMostAtLeast-subset-convex*:
fixes C :: *real set*
assumes *convex* C
and $x \in C\ y \in C\ x < y$
shows $\{x\ ..\ y\} \subseteq C$
⟨*proof*⟩

lemma *f''-imp-f'*:
fixes f :: *real* ⇒ *real*
assumes *convex* C
and f': $\bigwedge x.\ x \in C \Longrightarrow DERIV\ f\ x :> (f'\ x)$
and f'': $\bigwedge x.\ x \in C \Longrightarrow DERIV\ f'\ x :> (f''\ x)$
and *pos*: $\bigwedge x.\ x \in C \Longrightarrow f''\ x \geq 0$
and $x \in C\ y \in C$
shows $f'\ x * (y - x) \leq f\ y - f\ x$
⟨*proof*⟩

lemma *f''-ge0-imp-convex*:
fixes f :: *real* ⇒ *real*
assumes *conv*: *convex* C
and f': $\bigwedge x.\ x \in C \Longrightarrow DERIV\ f\ x :> (f'\ x)$
and f'': $\bigwedge x.\ x \in C \Longrightarrow DERIV\ f'\ x :> (f''\ x)$
and *pos*: $\bigwedge x.\ x \in C \Longrightarrow f''\ x \geq 0$
shows *convex-on* $C\ f$
⟨*proof*⟩

lemma *minus-log-convex*:
fixes $b :: real$
assumes $b > 1$
shows *convex-on* $\{0 <..\}$ $(\lambda\ x.\ -\ log\ b\ x)$
⟨*proof*⟩

17.5 Convexity of real functions

lemma *convex-on-realI*:
assumes *connected A*
assumes $\bigwedge x.\ x \in A \Longrightarrow (f\ has\text{-}real\text{-}derivative\ f'\ x)\ (at\ x)$
assumes $\bigwedge x\ y.\ x \in A \Longrightarrow y \in A \Longrightarrow x \leq y \Longrightarrow f'\ x \leq f'\ y$
shows *convex-on A f*
⟨*proof*⟩

lemma *convex-on-inverse*:
assumes $A \subseteq \{0<..\}$
shows *convex-on A* (*inverse* :: *real* ⇒ *real*)
⟨*proof*⟩

lemma *convex-onD-Icc'*:
assumes *convex-on* $\{x..y\}$ *f* $c \in \{x..y\}$
defines $d \equiv y - x$
shows $f\ c \leq (f\ y - f\ x)\ /\ d * (c - x) + f\ x$
⟨*proof*⟩

lemma *convex-onD-Icc''*:
assumes *convex-on* $\{x..y\}$ *f* $c \in \{x..y\}$
defines $d \equiv y - x$
shows $f\ c \leq (f\ x - f\ y)\ /\ d * (y - c) + f\ y$
⟨*proof*⟩

end

18 Algebraic operations on sets

theory *Set-Algebras*
imports *Main*
begin

This library lifts operations like addition and multiplication to sets. It was designed to support asymptotic calculations. See the comments at the top of theory *BigO*.

instantiation *set* :: (*plus*) *plus*
begin

definition *plus-set* :: *'a*::*plus set* ⇒ *'a set* ⇒ *'a set* **where**

set-plus-def: $A + B = \{c.\ \exists\, a{\in}A.\ \exists\, b{\in}B.\ c = a + b\}$

instance ⟨*proof*⟩

end

instantiation *set* :: (*times*) *times*
begin

definition *times-set* :: $'a$::*times set* ⇒ $'a$ *set* ⇒ $'a$ *set* **where**
set-times-def: $A * B = \{c.\ \exists\, a{\in}A.\ \exists\, b{\in}B.\ c = a * b\}$

instance ⟨*proof*⟩

end

instantiation *set* :: (*zero*) *zero*
begin

definition
set-zero[*simp*]: $(0{::}'a{::}zero\ set) = \{0\}$

instance ⟨*proof*⟩

end

instantiation *set* :: (*one*) *one*
begin

definition
set-one[*simp*]: $(1{::}'a{::}one\ set) = \{1\}$

instance ⟨*proof*⟩

end

definition *elt-set-plus* :: $'a$::*plus* ⇒ $'a$ *set* ⇒ $'a$ *set* (**infixl** $+o$ *70*) **where**
$a +o B = \{c.\ \exists\, b{\in}B.\ c = a + b\}$

definition *elt-set-times* :: $'a$::*times* ⇒ $'a$ *set* ⇒ $'a$ *set* (**infixl** $*o$ *80*) **where**
$a *o B = \{c.\ \exists\, b{\in}B.\ c = a * b\}$

abbreviation (*input*) *elt-set-eq* :: $'a$ ⇒ $'a$ *set* ⇒ *bool* (**infix** $=o$ *50*) **where**
$x =o A \equiv x \in A$

instance *set* :: (*semigroup-add*) *semigroup-add*
⟨*proof*⟩

instance *set* :: (*ab-semigroup-add*) *ab-semigroup-add*

⟨*proof*⟩

instance *set* :: (*monoid-add*) *monoid-add*
⟨*proof*⟩

instance *set* :: (*comm-monoid-add*) *comm-monoid-add*
⟨*proof*⟩

instance *set* :: (*semigroup-mult*) *semigroup-mult*
⟨*proof*⟩

instance *set* :: (*ab-semigroup-mult*) *ab-semigroup-mult*
⟨*proof*⟩

instance *set* :: (*monoid-mult*) *monoid-mult*
⟨*proof*⟩

instance *set* :: (*comm-monoid-mult*) *comm-monoid-mult*
⟨*proof*⟩

lemma *set-plus-intro* [*intro*]: $a \in C \Longrightarrow b \in D \Longrightarrow a + b \in C + D$
⟨*proof*⟩

lemma *set-plus-elim*:
assumes $x \in A + B$
obtains a b **where** $x = a + b$ **and** $a \in A$ **and** $b \in B$
⟨*proof*⟩

lemma *set-plus-intro2* [*intro*]: $b \in C \Longrightarrow a + b \in a +o C$
⟨*proof*⟩

lemma *set-plus-rearrange*:
$((a::'a::comm\text{-}monoid\text{-}add) +o C) + (b +o D) = (a + b) +o (C + D)$
⟨*proof*⟩

lemma *set-plus-rearrange2*: $(a::'a::semigroup\text{-}add) +o (b +o C) = (a + b) +o C$
⟨*proof*⟩

lemma *set-plus-rearrange3*: $((a::'a::semigroup\text{-}add) +o B) + C = a +o (B + C)$
⟨*proof*⟩

theorem *set-plus-rearrange4*: $C + ((a::'a::comm\text{-}monoid\text{-}add) +o D) = a +o (C + D)$
⟨*proof*⟩

lemmas *set-plus-rearranges* = *set-plus-rearrange set-plus-rearrange2 set-plus-rearrange3 set-plus-rearrange4*

lemma *set-plus-mono* [*intro*!]: $C \subseteq D \Longrightarrow a +o C \subseteq a +o D$

⟨*proof*⟩

lemma *set-plus-mono2* [*intro*]: $(C::'a::plus\ set) \subseteq D \Longrightarrow E \subseteq F \Longrightarrow C + E \subseteq D + F$
⟨*proof*⟩

lemma *set-plus-mono3* [*intro*]: $a \in C \Longrightarrow a +o D \subseteq C + D$
⟨*proof*⟩

lemma *set-plus-mono4* [*intro*]: $(a::'a::comm\text{-}monoid\text{-}add) \in C \Longrightarrow a +o D \subseteq D + C$
⟨*proof*⟩

lemma *set-plus-mono5*: $a \in C \Longrightarrow B \subseteq D \Longrightarrow a +o B \subseteq C + D$
⟨*proof*⟩

lemma *set-plus-mono-b*: $C \subseteq D \Longrightarrow x \in a +o C \Longrightarrow x \in a +o D$
⟨*proof*⟩

lemma *set-plus-mono2-b*: $C \subseteq D \Longrightarrow E \subseteq F \Longrightarrow x \in C + E \Longrightarrow x \in D + F$
⟨*proof*⟩

lemma *set-plus-mono3-b*: $a \in C \Longrightarrow x \in a +o D \Longrightarrow x \in C + D$
⟨*proof*⟩

lemma *set-plus-mono4-b*: $(a::'a::comm\text{-}monoid\text{-}add) : C \Longrightarrow x \in a +o D \Longrightarrow x \in D + C$
⟨*proof*⟩

lemma *set-zero-plus* [*simp*]: $(0::'a::comm\text{-}monoid\text{-}add) +o C = C$
⟨*proof*⟩

lemma *set-zero-plus2*: $(0::'a::comm\text{-}monoid\text{-}add) \in A \Longrightarrow B \subseteq A + B$
⟨*proof*⟩

lemma *set-plus-imp-minus*: $(a::'a::ab\text{-}group\text{-}add) : b +o C \Longrightarrow (a - b) \in C$
⟨*proof*⟩

lemma *set-minus-imp-plus*: $(a::'a::ab\text{-}group\text{-}add) - b : C \Longrightarrow a \in b +o C$
⟨*proof*⟩

lemma *set-minus-plus*: $(a::'a::ab\text{-}group\text{-}add) - b \in C \longleftrightarrow a \in b +o C$
⟨*proof*⟩

lemma *set-times-intro* [*intro*]: $a \in C \Longrightarrow b \in D \Longrightarrow a * b \in C * D$
⟨*proof*⟩

lemma *set-times-elim*:
assumes $x \in A * B$

obtains a b **where** $x = a * b$ **and** $a \in A$ **and** $b \in B$
⟨*proof*⟩

lemma *set-times-intro2* [*intro*!]: $b \in C \Longrightarrow a * b \in a *o C$
⟨*proof*⟩

lemma *set-times-rearrange*:
$((a::'a::comm\text{-}monoid\text{-}mult) *o C) * (b *o D) = (a * b) *o (C * D)$
⟨*proof*⟩

lemma *set-times-rearrange2*:
$(a::'a::semigroup\text{-}mult) *o (b *o C) = (a * b) *o C$
⟨*proof*⟩

lemma *set-times-rearrange3*:
$((a::'a::semigroup\text{-}mult) *o B) * C = a *o (B * C)$
⟨*proof*⟩

theorem *set-times-rearrange4*:
$C * ((a::'a::comm\text{-}monoid\text{-}mult) *o D) = a *o (C * D)$
⟨*proof*⟩

lemmas *set-times-rearranges = set-times-rearrange set-times-rearrange2 set-times-rearrange3 set-times-rearrange4*

lemma *set-times-mono* [*intro*]: $C \subseteq D \Longrightarrow a *o C \subseteq a *o D$
⟨*proof*⟩

lemma *set-times-mono2* [*intro*]: $(C::'a::times\ set) \subseteq D \Longrightarrow E \subseteq F \Longrightarrow C * E \subseteq D * F$
⟨*proof*⟩

lemma *set-times-mono3* [*intro*]: $a \in C \Longrightarrow a *o D \subseteq C * D$
⟨*proof*⟩

lemma *set-times-mono4* [*intro*]: $(a::'a::comm\text{-}monoid\text{-}mult) : C \Longrightarrow a *o D \subseteq D * C$
⟨*proof*⟩

lemma *set-times-mono5*: $a \in C \Longrightarrow B \subseteq D \Longrightarrow a *o B \subseteq C * D$
⟨*proof*⟩

lemma *set-times-mono-b*: $C \subseteq D \Longrightarrow x \in a *o C \Longrightarrow x \in a *o D$
⟨*proof*⟩

lemma *set-times-mono2-b*: $C \subseteq D \Longrightarrow E \subseteq F \Longrightarrow x \in C * E \Longrightarrow x \in D * F$
⟨*proof*⟩

lemma *set-times-mono3-b*: $a \in C \Longrightarrow x \in a *o D \Longrightarrow x \in C * D$

⟨*proof*⟩

lemma *set-times-mono4-b*: $(a::'a::comm\text{-}monoid\text{-}mult) \in C \Longrightarrow x \in a *o D \Longrightarrow x \in D * C$
⟨*proof*⟩

lemma *set-one-times* [*simp*]: $(1::'a::comm\text{-}monoid\text{-}mult) *o C = C$
⟨*proof*⟩

lemma *set-times-plus-distrib*:
$(a::'a::semiring) *o (b +o C) = (a * b) +o (a *o C)$
⟨*proof*⟩

lemma *set-times-plus-distrib2*:
$(a::'a::semiring) *o (B + C) = (a *o B) + (a *o C)$
⟨*proof*⟩

lemma *set-times-plus-distrib3*: $((a::'a::semiring) +o C) * D \subseteq a *o D + C * D$
⟨*proof*⟩

lemmas *set-times-plus-distribs* =
set-times-plus-distrib
set-times-plus-distrib2

lemma *set-neg-intro*: $(a::'a::ring\text{-}1) \in (-1) *o C \Longrightarrow - a \in C$
⟨*proof*⟩

lemma *set-neg-intro2*: $(a::'a::ring\text{-}1) \in C \Longrightarrow - a \in (-1) *o C$
⟨*proof*⟩

lemma *set-plus-image*: $S + T = (\lambda(x, y).\ x + y)\ `\ (S \times T)$
⟨*proof*⟩

lemma *set-times-image*: $S * T = (\lambda(x, y).\ x * y)\ `\ (S \times T)$
⟨*proof*⟩

lemma *finite-set-plus*: *finite* $s \Longrightarrow$ *finite* $t \Longrightarrow$ *finite* $(s + t)$
⟨*proof*⟩

lemma *finite-set-times*: *finite* $s \Longrightarrow$ *finite* $t \Longrightarrow$ *finite* $(s * t)$
⟨*proof*⟩

lemma *set-setsum-alt*:
assumes *fin*: *finite* I
shows *setsum* $S\ I = \{setsum\ s\ I\ |s.\ \forall i{\in}I.\ s\ i \in S\ i\}$
(**is** - = *?setsum* I)
⟨*proof*⟩

lemma *setsum-set-cond-linear*:

fixes f :: $'a$::*comm-monoid-add set* $\Rightarrow$ $'b$::*comm-monoid-add set*
assumes [*intro*!]: $\bigwedge A\ B.\ P\ A \implies P\ B \implies P\ (A + B)\ P\ \{0\}$
and f: $\bigwedge A\ B.\ P\ A \implies P\ B \Longrightarrow f\ (A + B) = f\ A + f\ B\ f\ \{0\} = \{0\}$
assumes *all*: $\bigwedge i.\ i \in I \Longrightarrow P\ (S\ i)$
shows $f\ (setsum\ S\ I) = setsum\ (f \circ S)\ I$
⟨*proof*⟩

lemma *setsum-set-linear*:
fixes f :: $'a$::*comm-monoid-add set* $\Rightarrow$ $'b$::*comm-monoid-add set*
assumes $\bigwedge A\ B.\ f(A) + f(B) = f(A + B)\ f\ \{0\} = \{0\}$
shows $f\ (setsum\ S\ I) = setsum\ (f \circ S)\ I$
⟨*proof*⟩

lemma *set-times-Un-distrib*:
$A * (B \cup C) = A * B \cup A * C$
$(A \cup B) * C = A * C \cup B * C$
⟨*proof*⟩

lemma *set-times-UNION-distrib*:
$A * UNION\ I\ M = (\bigcup i{\in}I.\ A * M\ i)$
$UNION\ I\ M * A = (\bigcup i{\in}I.\ M\ i * A)$
⟨*proof*⟩

end

19 Convex sets, functions and related things.

theory *Convex-Euclidean-Space*
imports
Topology-Euclidean-Space
~~/*src*/*HOL*/*Library*/*Convex*
~~/*src*/*HOL*/*Library*/*Set-Algebras*
begin

lemma *independent-injective-on-span-image*:
assumes *iS*: *independent S*
and *lf*: *linear f*
and *fi*: *inj-on f* (*span S*)
shows *independent* (*f* ‘ *S*)
⟨*proof*⟩

lemma *dim-image-eq*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'m$::*euclidean-space*
assumes *lf*: *linear f*
and *fi*: *inj-on f* (*span S*)
shows *dim* (*f* ‘ *S*) = *dim* (*S*::$'n$::*euclidean-space set*)
⟨*proof*⟩

lemma *linear-injective-on-subspace-0*:

assumes *lf*: *linear f*
and *subspace S*
shows *inj-on f S* $\longleftrightarrow$ $(\forall x \in S.\ f\ x = 0 \longrightarrow x = 0)$
⟨*proof*⟩

lemma *subspace-Inter*: $\forall s \in f.$ *subspace s* $\Longrightarrow$ *subspace* $(\bigcap f)$
⟨*proof*⟩

lemma *span-eq*[*simp*]: *span s* $=$ *s* $\longleftrightarrow$ *subspace s*
⟨*proof*⟩

lemma *substdbasis-expansion-unique*:
assumes *d*: $d \subseteq$ *Basis*
shows $(\sum i{\in}d.\ f\ i *_R i) = (x::'a::euclidean\text{-}space) \longleftrightarrow$
$(\forall i{\in}Basis.\ (i \in d \longrightarrow f\ i = x \cdot i) \wedge (i \notin d \longrightarrow x \cdot i = 0))$
⟨*proof*⟩

lemma *independent-substdbasis*: $d \subseteq$ *Basis* $\Longrightarrow$ *independent d*
⟨*proof*⟩

lemma *dim-cball*:
assumes $e > 0$
shows *dim* (*cball* $(0 :: 'n::euclidean\text{-}space)$ *e*) $=$ $DIM('n)$
⟨*proof*⟩

lemma *indep-card-eq-dim-span*:
fixes $B :: 'n::euclidean\text{-}space\ set$
assumes *independent B*
shows *finite B* $\wedge$ *card B* $=$ *dim* (*span B*)
⟨*proof*⟩

lemma *setsum-not-0*: *setsum f A* $\neq 0 \Longrightarrow \exists a \in A.\ f\ a \neq 0$
⟨*proof*⟩

lemma *subset-translation-eq* [*simp*]:
fixes $a :: 'a::real\text{-}vector$ **shows** *op* $+$ *a* ‘ *s* $\subseteq$ *op* $+$ *a* ‘ *t* $\longleftrightarrow$ *s* $\subseteq$ *t*
⟨*proof*⟩

lemma *translate-inj-on*:
fixes $A :: 'a::ab\text{-}group\text{-}add\ set$
shows *inj-on* $(\lambda x.\ a + x)$ *A*
⟨*proof*⟩

lemma *translation-assoc*:
fixes *a b* $:: 'a::ab\text{-}group\text{-}add$
shows $(\lambda x.\ b + x)$ ‘ $((\lambda x.\ a + x)$ ‘ *S*$) = (\lambda x.\ (a + b) + x)$ ‘ *S*
⟨*proof*⟩

lemma *translation-invert*:

fixes $a :: 'a::ab\text{-}group\text{-}add$
assumes $(\lambda x.\ a + x)\ `\ A = (\lambda x.\ a + x)\ `\ B$
shows $A = B$
⟨*proof*⟩

lemma *translation-galois*:
fixes $a :: 'a::ab\text{-}group\text{-}add$
shows $T = ((\lambda x.\ a + x)\ `\ S) \longleftrightarrow S = ((\lambda x.\ (-\ a) + x)\ `\ T)$
⟨*proof*⟩

lemma *convex-translation-eq* [*simp*]: $convex\ ((\lambda x.\ a + x)\ `\ s) \longleftrightarrow convex\ s$
⟨*proof*⟩

lemma *translation-inverse-subset*:
assumes $((\lambda x.\ -\ a + x)\ `\ V) \leq (S :: 'n::ab\text{-}group\text{-}add\ set)$
shows $V \leq ((\lambda x.\ a + x)\ `\ S)$
⟨*proof*⟩

lemma *convex-linear-image-eq* [*simp*]:
fixes $f :: 'a::real\text{-}vector \Rightarrow 'b::real\text{-}vector$
shows $[\![linear\ f;\ inj\ f]\!] \Longrightarrow convex\ (f\ `\ s) \longleftrightarrow convex\ s$
⟨*proof*⟩

lemma *basis-to-basis-subspace-isomorphism*:
assumes s: $subspace\ (S :: ('n::euclidean\text{-}space)\ set)$
and t: $subspace\ (T :: ('m::euclidean\text{-}space)\ set)$
and d: $dim\ S = dim\ T$
and B: $B \subseteq S\ independent\ B\ S \subseteq span\ B\ card\ B = dim\ S$
and C: $C \subseteq T\ independent\ C\ T \subseteq span\ C\ card\ C = dim\ T$
shows $\exists f.\ linear\ f \wedge f\ `\ B = C \wedge f\ `\ S = T \wedge inj\text{-}on\ f\ S$
⟨*proof*⟩

lemma *closure-bounded-linear-image-subset*:
assumes f: $bounded\text{-}linear\ f$
shows $f\ `\ closure\ S \subseteq closure\ (f\ `\ S)$
⟨*proof*⟩

lemma *closure-linear-image-subset*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::real\text{-}normed\text{-}vector$
assumes $linear\ f$
shows $f\ `\ (closure\ S) \subseteq closure\ (f\ `\ S)$
⟨*proof*⟩

lemma *closed-injective-linear-image*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes S: $closed\ S$ **and** f: $linear\ f\ inj\ f$
shows $closed\ (f\ `\ S)$
⟨*proof*⟩

lemma *closed-injective-linear-image-eq*:
fixes f :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes f: *linear f inj f*
shows (*closed*(*image f s*) ⟷ *closed s*)
⟨*proof*⟩

lemma *closure-injective-linear-image*:
fixes f :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
shows ⟦*linear f*; *inj f*⟧ ⟹ *f* ‘ (*closure S*) = *closure* (*f* ‘ *S*)
⟨*proof*⟩

lemma *closure-bounded-linear-image*:
fixes f :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
shows ⟦*linear f*; *bounded S*⟧ ⟹ *f* ‘ (*closure S*) = *closure* (*f* ‘ *S*)
⟨*proof*⟩

lemma *closure-scaleR*:
fixes S :: *'a*::*real-normed-vector set*
shows (*op* ∗$_R$ *c*) ‘ (*closure S*) = *closure* ((*op* ∗$_R$ *c*) ‘ *S*)
⟨*proof*⟩

lemma *fst-linear*: *linear fst*
⟨*proof*⟩

lemma *snd-linear*: *linear snd*
⟨*proof*⟩

lemma *fst-snd-linear*: *linear* (λ(*x*,*y*). *x* + *y*)
⟨*proof*⟩

lemma *scaleR-2*:
fixes x :: *'a*::*real-vector*
shows *scaleR 2 x* = *x* + *x*
⟨*proof*⟩

lemma *scaleR-half-double* [*simp*]:
fixes a :: *'a*::*real-normed-vector*
shows (*1* / *2*) ∗$_R$ (*a* + *a*) = *a*
⟨*proof*⟩

lemma *vector-choose-size*:
assumes *0* ≤ *c*
obtains x :: *'a*::{*real-normed-vector*, *perfect-space*} **where** *norm x* = *c*
⟨*proof*⟩

lemma *vector-choose-dist*:
assumes *0* ≤ *c*
obtains y :: *'a*::{*real-normed-vector*, *perfect-space*} **where** *dist x y* = *c*
⟨*proof*⟩

lemma *sphere-eq-empty* [*simp*]:
fixes *a* :: *'a*::{*real-normed-vector*, *perfect-space*}
shows *sphere a r* = {} ⟷ *r* < *0*
⟨*proof*⟩

lemma *setsum-delta-notmem*:
assumes *x* ∉ *s*
shows *setsum* (λ*y*. *if* (*y* = *x*) *then P x else Q y*) *s* = *setsum Q s*
and *setsum* (λ*y*. *if* (*x* = *y*) *then P x else Q y*) *s* = *setsum Q s*
and *setsum* (λ*y*. *if* (*y* = *x*) *then P y else Q y*) *s* = *setsum Q s*
and *setsum* (λ*y*. *if* (*x* = *y*) *then P y else Q y*) *s* = *setsum Q s*
⟨*proof*⟩

lemma *setsum-delta''*:
fixes *s*::*'a*::*real-vector set*
assumes *finite s*
shows (∑ *x*∈*s*. (*if y* = *x then f x else 0*) ∗*R x*) = (*if y*∈*s then* (*f y*) ∗*R y else 0*)
⟨*proof*⟩

lemma *if-smult*: (*if P then x else* (*y*::*real*)) ∗*R v* = (*if P then x* ∗*R v else y* ∗*R v*)
⟨*proof*⟩

lemma *dist-triangle-eq*:
fixes *x y z* :: *'a*::*real-inner*
shows *dist x z* = *dist x y* + *dist y z* ⟷
norm (*x* − *y*) ∗*R* (*y* − *z*) = *norm* (*y* − *z*) ∗*R* (*x* − *y*)
⟨*proof*⟩

lemma *norm-minus-eqI*: *x* = − *y* ⟹ *norm x* = *norm y* ⟨*proof*⟩

lemma *Min-grI*:
assumes *finite A A* ≠ {} ∀ *a*∈*A*. *x* < *a*
shows *x* < *Min A*
⟨*proof*⟩

lemma *norm-lt*: *norm x* < *norm y* ⟷ *inner x x* < *inner y y*
⟨*proof*⟩

lemma *norm-le*: *norm x* ≤ *norm y* ⟷ *inner x x* ≤ *inner y y*
⟨*proof*⟩

19.1 Affine set and affine hull

definition *affine* :: *'a*::*real-vector set* ⇒ *bool*
where *affine s* ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. ∀ *u v*. *u* + *v* = *1* ⟶ *u* ∗*R x* + *v* ∗*R y* ∈ *s*)

lemma *affine-alt*: *affine s* ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. ∀ *u*::*real*. (*1* − *u*) ∗*R x* + *u* ∗*R y* ∈ *s*)

⟨*proof*⟩

lemma *affine-empty* [*iff*]: *affine* {}
⟨*proof*⟩

lemma *affine-sing* [*iff*]: *affine* {*x*}
⟨*proof*⟩

lemma *affine-UNIV* [*iff*]: *affine UNIV*
⟨*proof*⟩

lemma *affine-Inter*[*intro*]: (∀ *s*∈*f*. *affine s*) ⟹ *affine* (⋂*f*)
⟨*proof*⟩

lemma *affine-Int*[*intro*]: *affine s* ⟹ *affine t* ⟹ *affine* (*s* ∩ *t*)
⟨*proof*⟩

lemma *affine-affine-hull* [*simp*]: *affine*(*affine hull s*)
⟨*proof*⟩

lemma *affine-hull-eq*[*simp*]: (*affine hull s* = *s*) ⟷ *affine s*
⟨*proof*⟩

lemma *affine-hyperplane*: *affine* {*x*. *a* ∙ *x* = *b*}
⟨*proof*⟩

19.1.1 Some explicit formulations (from Lars Schewe)

lemma *affine*:
fixes *V*::*'a*::*real-vector set*
shows *affine V* ⟷
(∀ *s u*. *finite s* ∧ *s* ≠ {} ∧ *s* ⊆ *V* ∧ *setsum u s* = *1* ⟶ (*setsum* (λ*x*. (*u x*) ∗*R* *x*)) *s* ∈ *V*)
⟨*proof*⟩

lemma *affine-hull-explicit*:
affine hull p =
{*y*. ∃ *s u*. *finite s* ∧ *s* ≠ {} ∧ *s* ⊆ *p* ∧ *setsum u s* = *1* ∧ *setsum* (λ*v*. (*u v*) ∗*R* *v*) *s* = *y*}
⟨*proof*⟩

lemma *affine-hull-finite*:
assumes *finite s*
shows *affine hull s* = {*y*. ∃ *u*. *setsum u s* = *1* ∧ *setsum* (λ*v*. *u v* ∗*R* *v*) *s* = *y*}
⟨*proof*⟩

19.1.2 Stepping theorems and hence small special cases

lemma *affine-hull-empty*[*simp*]: *affine hull* {} = {}
⟨*proof*⟩

lemma *affine-hull-finite-step*:
fixes y :: *'a*::*real-vector*
shows
$(\exists u.\ setsum\ u\ \{\} = w \wedge setsum\ (\lambda x.\ u\ x *_R x)\ \{\} = y) \longleftrightarrow w = 0 \wedge y = 0$
(**is** *?th1*)
and
finite $s \Longrightarrow$
$(\exists u.\ setsum\ u\ (insert\ a\ s) = w \wedge setsum\ (\lambda x.\ u\ x *_R x)\ (insert\ a\ s) = y)$
$\longleftrightarrow$
$(\exists v\ u.\ setsum\ u\ s = w - v \wedge setsum\ (\lambda x.\ u\ x *_R x)\ s = y - v *_R a)$ (**is** *?lhs* $\Longrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *affine-hull-2*:
fixes $a\ b$ :: *'a*::*real-vector*
shows *affine hull* $\{a,b\} = \{u *_R a + v *_R b |\ u\ v.\ (u + v = 1)\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *affine-hull-3*:
fixes $a\ b\ c$ :: *'a*::*real-vector*
shows *affine hull* $\{a,b,c\} = \{ u *_R a + v *_R b + w *_R c |\ u\ v\ w.\ u + v + w = 1\}$
⟨*proof*⟩

lemma *mem-affine*:
assumes *affine* $S\ x \in S\ y \in S\ u + v = 1$
shows $u *_R x + v *_R y \in S$
⟨*proof*⟩

lemma *mem-affine-3*:
assumes *affine* $S\ x \in S\ y \in S\ z \in S\ u + v + w = 1$
shows $u *_R x + v *_R y + w *_R z \in S$
⟨*proof*⟩

lemma *mem-affine-3-minus*:
assumes *affine* $S\ x \in S\ y \in S\ z \in S$
shows $x + v *_R (y-z) \in S$
⟨*proof*⟩

corollary *mem-affine-3-minus2*:
$[\![affine\ S;\ x \in S;\ y \in S;\ z \in S]\!] \Longrightarrow x - v *_R (y-z) \in S$
⟨*proof*⟩

19.1.3 Some relations between affine hull and subspaces

lemma *affine-hull-insert-subset-span*:
affine hull $(insert\ a\ s) \subseteq \{a + v |\ v\ .\ v \in span\ \{x - a\ |\ x\ .\ x \in s\}\}$

⟨*proof*⟩

lemma *affine-hull-insert-span*:
assumes $a \notin s$
shows *affine hull* (*insert a s*) = $\{a + v \mid v\ .\ v \in$ *span* $\{x - a \mid x.\ x \in s\}\}$
⟨*proof*⟩

lemma *affine-hull-span*:
assumes $a \in s$
shows *affine hull* $s = \{a + v \mid v.\ v \in$ *span* $\{x - a \mid x.\ x \in s - \{a\}\}\}$
⟨*proof*⟩

19.1.4 Parallel affine sets

definition *affine-parallel* :: *'a*::*real-vector set* ⇒ *'a*::*real-vector set* ⇒ *bool*
where *affine-parallel S T* ⟷ $(\exists a.\ T = (\lambda x.\ a + x)$ ‘ $S)$

lemma *affine-parallel-expl-aux*:
fixes *S T* :: *'a*::*real-vector set*
assumes $\forall x.\ x \in S \longleftrightarrow a + x \in T$
shows $T = (\lambda x.\ a + x)$ ‘ S
⟨*proof*⟩

lemma *affine-parallel-expl*: *affine-parallel S T* ⟷ $(\exists a.\ \forall x.\ x \in S \longleftrightarrow a + x \in T)$
⟨*proof*⟩

lemma *affine-parallel-reflex*: *affine-parallel S S*
⟨*proof*⟩

lemma *affine-parallel-commut*:
assumes *affine-parallel A B*
shows *affine-parallel B A*
⟨*proof*⟩

lemma *affine-parallel-assoc*:
assumes *affine-parallel A B*
and *affine-parallel B C*
shows *affine-parallel A C*
⟨*proof*⟩

lemma *affine-translation-aux*:
fixes *a* :: *'a*::*real-vector*
assumes *affine* $((\lambda x.\ a + x)$ ‘ $S)$
shows *affine S*
⟨*proof*⟩

lemma *affine-translation*:
fixes *a* :: *'a*::*real-vector*

shows *affine* $S \longleftrightarrow$ *affine* $((\lambda x.\ a + x)\ `\ S)$
⟨*proof*⟩

lemma *parallel-is-affine*:
fixes $S\ T :: 'a$::*real-vector set*
assumes *affine* S *affine-parallel* $S\ T$
shows *affine* T
⟨*proof*⟩

lemma *subspace-imp-affine*: *subspace* $s \Longrightarrow$ *affine* s
⟨*proof*⟩

19.1.5 Subspace parallel to an affine set

lemma *subspace-affine*: *subspace* $S \longleftrightarrow$ *affine* $S \wedge 0 \in S$
⟨*proof*⟩

lemma *affine-diffs-subspace*:
assumes *affine* $S\ a \in S$
shows *subspace* $((\lambda x.\ (-a)+x)\ `\ S)$
⟨*proof*⟩

lemma *parallel-subspace-explicit*:
assumes *affine* S
and $a \in S$
assumes $L \equiv \{y.\ \exists x \in S.\ (-a) + x = y\}$
shows *subspace* $L \wedge$ *affine-parallel* $S\ L$
⟨*proof*⟩

lemma *parallel-subspace-aux*:
assumes *subspace* A
and *subspace* B
and *affine-parallel* $A\ B$
shows $A \supseteq B$
⟨*proof*⟩

lemma *parallel-subspace*:
assumes *subspace* A
and *subspace* B
and *affine-parallel* $A\ B$
shows $A = B$
⟨*proof*⟩

lemma *affine-parallel-subspace*:
assumes *affine* $S\ S \neq \{\}$
shows $\exists!L.$ *subspace* $L \wedge$ *affine-parallel* $S\ L$
⟨*proof*⟩

19.2 Cones

definition *cone* :: *'a::real-vector set* ⇒ *bool*
  where *cone s* ⟷ (∀ *x*∈*s*. ∀ *c*≥*0*. *c* ∗$_R$ *x* ∈ *s*)

lemma *cone-empty*[*intro*, *simp*]: *cone* {}
  ⟨*proof*⟩

lemma *cone-univ*[*intro*, *simp*]: *cone UNIV*
  ⟨*proof*⟩

lemma *cone-Inter*[*intro*]: ∀ *s*∈*f*. *cone s* ⟹ *cone* (⋂ *f*)
  ⟨*proof*⟩

19.2.1 Conic hull

lemma *cone-cone-hull*: *cone* (*cone hull s*)
  ⟨*proof*⟩

lemma *cone-hull-eq*: *cone hull s* = *s* ⟷ *cone s*
  ⟨*proof*⟩

lemma *mem-cone*:
  assumes *cone S x* ∈ *S c* ≥ *0*
  shows *c* ∗$_R$ *x* : *S*
  ⟨*proof*⟩

lemma *cone-contains-0*:
  assumes *cone S*
  shows *S* ≠ {} ⟷ *0* ∈ *S*
⟨*proof*⟩

lemma *cone-0*: *cone* {*0*}
  ⟨*proof*⟩

lemma *cone-Union*[*intro*]: (∀ *s*∈*f*. *cone s*) ⟶ *cone* (⋃ *f*)
  ⟨*proof*⟩

lemma *cone-iff*:
  assumes *S* ≠ {}
  shows *cone S* ⟷ *0* ∈ *S* ∧ (∀ *c*. *c* > *0* ⟶ (*op* ∗$_R$ *c*) ' *S* = *S*)
⟨*proof*⟩

lemma *cone-hull-empty*: *cone hull* {} = {}
  ⟨*proof*⟩

lemma *cone-hull-empty-iff*: *S* = {} ⟷ *cone hull S* = {}
  ⟨*proof*⟩

lemma *cone-hull-contains-0*: *S* ≠ {} ⟷ *0* ∈ *cone hull S*

⟨*proof*⟩

lemma *mem-cone-hull*:
assumes $x \in S$ $c \geq 0$
shows $c *_R x \in$ *cone hull* S
⟨*proof*⟩

lemma *cone-hull-expl*: *cone hull* $S = \{c *_R x \mid c\ x.\ c \geq 0 \wedge x \in S\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *cone-closure*:
fixes S :: *'a*::*real-normed-vector set*
assumes *cone* S
shows *cone* (*closure* S)
⟨*proof*⟩

19.3 Affine dependence and consequential theorems (from Lars Schewe)

definition *affine-dependent* :: *'a*::*real-vector set* ⇒ *bool*
where *affine-dependent* $s \longleftrightarrow (\exists x{\in}s.\ x \in$ *affine hull* $(s - \{x\}))$

lemma *affine-dependent-explicit*:
affine-dependent $p \longleftrightarrow$
$(\exists s\ u.$ *finite* $s \wedge s \subseteq p \wedge$ *setsum* $u\ s = 0 \wedge$
$(\exists v{\in}s.\ u\ v \neq 0) \wedge$ *setsum* $(\lambda v.\ u\ v *_R v)\ s = 0)$
⟨*proof*⟩

lemma *affine-dependent-explicit-finite*:
fixes s :: *'a*::*real-vector set*
assumes *finite* s
shows *affine-dependent* $s \longleftrightarrow$
$(\exists u.$ *setsum* $u\ s = 0 \wedge (\exists v{\in}s.\ u\ v \neq 0) \wedge$ *setsum* $(\lambda v.\ u\ v *_R v)\ s = 0)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

19.4 Connectedness of convex sets

lemma *connectedD*:
connected $S \Longrightarrow$ *open* $A \Longrightarrow$ *open* $B \Longrightarrow S \subseteq A \cup B \Longrightarrow A \cap B \cap S = \{\} \Longrightarrow$
$A \cap S = \{\} \vee B \cap S = \{\}$
⟨*proof*⟩

lemma *convex-connected*:
fixes s :: *'a*::*real-normed-vector set*
assumes *convex* s
shows *connected* s
⟨*proof*⟩

corollary *connected-UNIV*[*intro*]: *connected* (*UNIV* :: *'a*::*real-normed-vector set*)
⟨*proof*⟩

proposition *clopen*:
fixes *s* :: *'a* :: *real-normed-vector set*
shows *closed s* ∧ *open s* ⟷ *s* = {} ∨ *s* = *UNIV*
⟨*proof*⟩

corollary *compact-open*:
fixes *s* :: *'a* :: *euclidean-space set*
shows *compact s* ∧ *open s* ⟷ *s* = {}
⟨*proof*⟩

corollary *finite-imp-not-open*:
fixes *S* :: *'a*::{*real-normed-vector*, *perfect-space*} *set*
shows ⟦*finite S*; *open S*⟧ ⟹ *S*={}
⟨*proof*⟩

Balls, being convex, are connected.

lemma *convex-prod*:
assumes ⋀*i*. *i* ∈ *Basis* ⟹ *convex* {*x*. *P i x*}
shows *convex* {*x*. ∀ *i*∈*Basis*. *P i* (*x*•*i*)}
⟨*proof*⟩

lemma *convex-positive-orthant*: *convex* {*x*::*'a*::*euclidean-space*. (∀ *i*∈*Basis*. *0* ≤ *x*•*i*)}
⟨*proof*⟩

lemma *convex-local-global-minimum*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *e* > *0*
and *convex-on s f*
and *ball x e* ⊆ *s*
and ∀ *y*∈*ball x e*. *f x* ≤ *f y*
shows ∀ *y*∈*s*. *f x* ≤ *f y*
⟨*proof*⟩

lemma *convex-ball* [*iff*]:
fixes *x* :: *'a*::*real-normed-vector*
shows *convex* (*ball x e*)
⟨*proof*⟩

lemma *convex-cball* [*iff*]:
fixes *x* :: *'a*::*real-normed-vector*
shows *convex* (*cball x e*)
⟨*proof*⟩

lemma *connected-ball* [*iff*]:

fixes x :: $'a$::*real-normed-vector*
shows *connected* (*ball* x e)
⟨*proof*⟩

lemma *connected-cball* [*iff*]:
fixes x :: $'a$::*real-normed-vector*
shows *connected* (*cball* x e)
⟨*proof*⟩

19.5 Convex hull

lemma *convex-convex-hull* [*iff*]: *convex* (*convex hull* s)
⟨*proof*⟩

lemma *convex-hull-eq*: *convex hull* $s = s \longleftrightarrow$ *convex* s
⟨*proof*⟩

lemma *bounded-convex-hull*:
fixes s :: $'a$::*real-normed-vector set*
assumes *bounded* s
shows *bounded* (*convex hull* s)
⟨*proof*⟩

lemma *finite-imp-bounded-convex-hull*:
fixes s :: $'a$::*real-normed-vector set*
shows *finite* $s \Longrightarrow$ *bounded* (*convex hull* s)
⟨*proof*⟩

19.5.1 Convex hull is "preserved" by a linear function

lemma *convex-hull-linear-image*:
assumes f: *linear* f
shows f ‘ (*convex hull* s) = *convex hull* (f ‘ s)
⟨*proof*⟩

lemma *in-convex-hull-linear-image*:
assumes *linear* f
and $x \in$ *convex hull* s
shows $f\ x \in$ *convex hull* (f ‘ s)
⟨*proof*⟩

lemma *convex-hull-Times*:
convex hull ($s \times t$) = (*convex hull* s) × (*convex hull* t)
⟨*proof*⟩

19.5.2 Stepping theorems for convex hulls of finite sets

lemma *convex-hull-empty*[*simp*]: *convex hull* {} = {}
⟨*proof*⟩

lemma *convex-hull-singleton*[*simp*]: *convex hull* $\{a\} = \{a\}$
⟨*proof*⟩

lemma *convex-hull-insert*:
fixes s :: *'a*::*real-vector set*
assumes $s \neq \{\}$
shows *convex hull* (*insert a s*) =
$\{x.\ \exists u{\geq}0.\ \exists v{\geq}0.\ \exists b.\ (u + v = 1) \wedge b \in (\textit{convex hull } s) \wedge (x = u *_R a + v *_R b)\}$
(**is** - = *?hull*)
⟨*proof*⟩

19.5.3 Explicit expression for convex hull

lemma *convex-hull-indexed*:
fixes s :: *'a*::*real-vector set*
shows *convex hull s* =
$\{y.\ \exists k\ u\ x.$
$(\forall i{\in}\{1{::}nat\ ..\ k\}.\ 0 \leq u\ i \wedge x\ i \in s) \wedge$
$(setsum\ u\ \{1..k\} = 1) \wedge (setsum\ (\lambda i.\ u\ i *_R x\ i)\ \{1..k\} = y)\}$
(**is** *?xyz* = *?hull*)
⟨*proof*⟩

lemma *convex-hull-finite*:
fixes s :: *'a*::*real-vector set*
assumes *finite s*
shows *convex hull* $s = \{y.\ \exists u.\ (\forall x{\in}s.\ 0 \leq u\ x) \wedge$
$setsum\ u\ s = 1 \wedge setsum\ (\lambda x.\ u\ x *_R x)\ s = y\}$
(**is** *?HULL* = *?set*)
⟨*proof*⟩

19.5.4 Another formulation from Lars Schewe

lemma *convex-hull-explicit*:
fixes p :: *'a*::*real-vector set*
shows *convex hull p* =
$\{y.\ \exists s\ u.\ finite\ s \wedge s \subseteq p \wedge (\forall x{\in}s.\ 0 \leq u\ x) \wedge setsum\ u\ s = 1 \wedge setsum\ (\lambda v.\ u\ v *_R v)\ s = y\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

19.5.5 A stepping theorem for that expansion

lemma *convex-hull-finite-step*:
fixes s :: *'a*::*real-vector set*
assumes *finite s*
shows
$(\exists u.\ (\forall x{\in}insert\ a\ s.\ 0 \leq u\ x) \wedge setsum\ u\ (insert\ a\ s) = w \wedge setsum\ (\lambda x.\ u\ x *_R x)\ (insert\ a\ s) = y)$

$\longleftrightarrow (\exists\, v{\geq}0.\ \exists\, u.\ (\forall\, x{\in}s.\ 0 \leq u\ x) \wedge \mathit{setsum}\ u\ s = w - v \wedge \mathit{setsum}\ (\lambda x.\ u\ x *_R x)\ s = y - v *_R a)$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

19.5.6 Hence some special cases

lemma *convex-hull-2*:
convex hull $\{a,b\} = \{u *_R a + v *_R b \mid u\ v.\ 0 \leq u \wedge 0 \leq v \wedge u + v = 1\}$
⟨*proof*⟩

lemma *convex-hull-2-alt*: *convex hull* $\{a,b\} = \{a + u *_R (b - a) \mid u.\ 0 \leq u \wedge u \leq 1\}$
⟨*proof*⟩

lemma *convex-hull-3*:
convex hull $\{a,b,c\} = \{ u *_R a + v *_R b + w *_R c \mid u\ v\ w.\ 0 \leq u \wedge 0 \leq v \wedge 0 \leq w \wedge u + v + w = 1\}$
⟨*proof*⟩

lemma *convex-hull-3-alt*:
convex hull $\{a,b,c\} = \{a + u *_R (b - a) + v *_R (c - a) \mid u\ v.\ 0 \leq u \wedge 0 \leq v \wedge u + v \leq 1\}$
⟨*proof*⟩

19.6 Relations among closure notions and corresponding hulls

lemma *affine-imp-convex*: *affine s* $\Longrightarrow$ *convex s*
⟨*proof*⟩

lemma *subspace-imp-convex*: *subspace s* $\Longrightarrow$ *convex s*
⟨*proof*⟩

lemma *affine-hull-subset-span*: (*affine hull s*) $\subseteq$ (*span s*)
⟨*proof*⟩

lemma *convex-hull-subset-span*: (*convex hull s*) $\subseteq$ (*span s*)
⟨*proof*⟩

lemma *convex-hull-subset-affine-hull*: (*convex hull s*) $\subseteq$ (*affine hull s*)
⟨*proof*⟩

lemma *affine-dependent-imp-dependent*: *affine-dependent s* $\Longrightarrow$ *dependent s*
⟨*proof*⟩

lemma *dependent-imp-affine-dependent*:
assumes *dependent* $\{x - a \mid x\ .\ x \in s\}$
and $a \notin s$
shows *affine-dependent* (*insert a s*)

⟨*proof*⟩

lemma *convex-cone*:
convex s ∧ *cone s* ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. (*x* + *y*) ∈ *s*) ∧ (∀ *x*∈*s*. ∀ *c*≥*0*. (*c* *$_R$ *x*) ∈ *s*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *affine-dependent-biggerset*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *finite s card s* ≥ *DIM*(*'a*) + *2*
shows *affine-dependent s*
⟨*proof*⟩

lemma *affine-dependent-biggerset-general*:
assumes *finite* (*s* :: *'a*::*euclidean-space set*)
and *card s* ≥ *dim s* + *2*
shows *affine-dependent s*
⟨*proof*⟩

19.7 Some Properties of Affine Dependent Sets

lemma *affine-independent-empty*: ¬ *affine-dependent* {}
⟨*proof*⟩

lemma *affine-independent-sing*: ¬ *affine-dependent* {*a*}
⟨*proof*⟩

lemma *affine-hull-translation*: *affine hull* ((λ*x*. *a* + *x*) ‘ *S*) = (λ*x*. *a* + *x*) ‘ (*affine hull S*)
⟨*proof*⟩

lemma *affine-dependent-translation*:
assumes *affine-dependent S*
shows *affine-dependent* ((λ*x*. *a* + *x*) ‘ *S*)
⟨*proof*⟩

lemma *affine-dependent-translation-eq*:
affine-dependent S ⟷ *affine-dependent* ((λ*x*. *a* + *x*) ‘ *S*)
⟨*proof*⟩

lemma *affine-hull-0-dependent*:
assumes *0* ∈ *affine hull S*
shows *dependent S*
⟨*proof*⟩

lemma *affine-dependent-imp-dependent2*:
assumes *affine-dependent* (*insert 0 S*)
shows *dependent S*

⟨*proof*⟩

lemma *affine-dependent-iff-dependent*:
assumes $a \notin S$
shows *affine-dependent* (*insert a S*) $\longleftrightarrow$ *dependent* $((\lambda x.\ -a + x)$ ` $S)$
⟨*proof*⟩

lemma *affine-dependent-iff-dependent2*:
assumes $a \in S$
shows *affine-dependent S* $\longleftrightarrow$ *dependent* $((\lambda x.\ -a + x)$ ` $(S - \{a\}))$
⟨*proof*⟩

lemma *affine-hull-insert-span-gen*:
affine hull (*insert a s*) = $(\lambda x.\ a + x)$ ` *span* $((\lambda x.\ -a + x)$ ` *s*)
⟨*proof*⟩

lemma *affine-hull-span2*:
assumes $a \in s$
shows *affine hull s* = $(\lambda x.\ a+x)$ ` *span* $((\lambda x.\ -a+x)$ ` $(s-\{a\}))$
⟨*proof*⟩

lemma *affine-hull-span-gen*:
assumes $a \in$ *affine hull s*
shows *affine hull s* = $(\lambda x.\ a+x)$ ` *span* $((\lambda x.\ -a+x)$ ` *s*)
⟨*proof*⟩

lemma *affine-hull-span-0*:
assumes $0 \in$ *affine hull S*
shows *affine hull S* = *span S*
⟨*proof*⟩

lemma *extend-to-affine-basis*:
fixes *S V* :: *'n*::*euclidean-space set*
assumes ¬ *affine-dependent S* $S \subseteq V$ $S \neq \{\}$
shows $\exists\, T.$ ¬ *affine-dependent T* $\wedge$ $S \subseteq T \wedge T \subseteq V \wedge$ *affine hull T* = *affine hull V*
⟨*proof*⟩

lemma *affine-basis-exists*:
fixes *V* :: *'n*::*euclidean-space set*
shows $\exists\, B.\ B \subseteq V \wedge$ ¬ *affine-dependent B* $\wedge$ *affine hull V* = *affine hull B*
⟨*proof*⟩

19.8 Affine Dimension of a Set

definition *aff-dim* :: (*'a*::*euclidean-space*) *set* ⇒ *int*
where *aff-dim V* =
(*SOME d* :: *int*.

$\exists B.$ *affine hull* $B =$ *affine hull* $V \wedge \neg$ *affine-dependent* $B \wedge$ *of-nat* (*card* B) = $d + 1$)

lemma *aff-dim-basis-exists*:
fixes $V :: ('n::euclidean\text{-}space)$ *set*
shows $\exists B.$ *affine hull* $B =$ *affine hull* $V \wedge \neg$ *affine-dependent* $B \wedge$ *of-nat* (*card* B) = *aff-dim* $V + 1$
⟨*proof*⟩

lemma *affine-hull-nonempty*: $S \neq \{\} \longleftrightarrow$ *affine hull* $S \neq \{\}$
⟨*proof*⟩

lemma *aff-dim-parallel-subspace-aux*:
fixes $B :: 'n::euclidean\text{-}space$ *set*
assumes $\neg$ *affine-dependent* B $a \in B$
shows *finite* $B \wedge ((card\ B) - 1 = dim\ (span\ ((\lambda x.\ -a+x)\ `\ (B-\{a\}))))$
⟨*proof*⟩

lemma *aff-dim-parallel-subspace*:
fixes $V\ L :: 'n::euclidean\text{-}space$ *set*
assumes $V \neq \{\}$
and *subspace* L
and *affine-parallel* (*affine hull* V) L
shows *aff-dim* $V = int\ (dim\ L)$
⟨*proof*⟩

lemma *aff-independent-finite*:
fixes $B :: 'n::euclidean\text{-}space$ *set*
assumes $\neg$ *affine-dependent* B
shows *finite* B
⟨*proof*⟩

lemma *independent-finite*:
fixes $B :: 'n::euclidean\text{-}space$ *set*
assumes *independent* B
shows *finite* B
⟨*proof*⟩

lemma *subspace-dim-equal*:
assumes *subspace* $(S :: ('n::euclidean\text{-}space)\ set)$
and *subspace* T
and $S \subseteq T$
and $dim\ S \geq dim\ T$
shows $S = T$
⟨*proof*⟩

lemma *span-substd-basis*:
assumes d: $d \subseteq Basis$
shows *span* $d = \{x.\ \forall i{\in}Basis.\ i \notin d \longrightarrow x{\cdot}i = 0\}$

(**is** - = *?B*)
⟨*proof*⟩

lemma *basis-to-substdbasis-subspace-isomorphism*:
fixes *B* :: *'a*::*euclidean-space set*
assumes *independent B*
shows ∃*f d*::*'a set*. *card d* = *card B* ∧ *linear f* ∧ *f* ' *B* = *d* ∧
f ' *span B* = {*x*. ∀ *i*∈*Basis*. *i* ∉ *d* ⟶ *x* • *i* = *0*} ∧ *inj-on f* (*span B*) ∧ *d* ⊆ *Basis*
⟨*proof*⟩

lemma *aff-dim-empty*:
fixes *S* :: *'n*::*euclidean-space set*
shows *S* = {} ⟷ *aff-dim S* = *−1*
⟨*proof*⟩

lemma *aff-dim-empty-eq* [*simp*]: *aff-dim* ({}::*'a*::*euclidean-space set*) = *−1*
⟨*proof*⟩

lemma *aff-dim-affine-hull*: *aff-dim* (*affine hull S*) = *aff-dim S*
⟨*proof*⟩

lemma *aff-dim-affine-hull2*:
assumes *affine hull S* = *affine hull T*
shows *aff-dim S* = *aff-dim T*
⟨*proof*⟩

lemma *aff-dim-unique*:
fixes *B V* :: *'n*::*euclidean-space set*
assumes *affine hull B* = *affine hull V* ∧ ¬ *affine-dependent B*
shows *of-nat* (*card B*) = *aff-dim V* + *1*
⟨*proof*⟩

lemma *aff-dim-affine-independent*:
fixes *B* :: *'n*::*euclidean-space set*
assumes ¬ *affine-dependent B*
shows *of-nat* (*card B*) = *aff-dim B* + *1*
⟨*proof*⟩

lemma *affine-independent-iff-card*:
fixes *s* :: *'a*::*euclidean-space set*
shows ~ *affine-dependent s* ⟷ *finite s* ∧ *aff-dim s* = *int*(*card s*) − *1*
⟨*proof*⟩

lemma *aff-dim-sing* [*simp*]:
fixes *a* :: *'n*::*euclidean-space*
shows *aff-dim* {*a*} = *0*
⟨*proof*⟩

lemma *aff-dim-inner-basis-exists*:
fixes $V :: ('n::euclidean\text{-}space)\ set$
shows $\exists B.\ B \subseteq V \wedge affine\ hull\ B = affine\ hull\ V \wedge$
$\neg\ affine\text{-}dependent\ B \wedge of\text{-}nat\ (card\ B) = aff\text{-}dim\ V + 1$
⟨*proof*⟩

lemma *aff-dim-le-card*:
fixes $V :: 'n::euclidean\text{-}space\ set$
assumes *finite V*
shows $aff\text{-}dim\ V \leq of\text{-}nat\ (card\ V) - 1$
⟨*proof*⟩

lemma *aff-dim-parallel-eq*:
fixes $S\ T :: 'n::euclidean\text{-}space\ set$
assumes $affine\text{-}parallel\ (affine\ hull\ S)\ (affine\ hull\ T)$
shows $aff\text{-}dim\ S = aff\text{-}dim\ T$
⟨*proof*⟩

lemma *aff-dim-translation-eq*:
fixes $a :: 'n::euclidean\text{-}space$
shows $aff\text{-}dim\ ((\lambda x.\ a + x)\ `\ S) = aff\text{-}dim\ S$
⟨*proof*⟩

lemma *aff-dim-affine*:
fixes $S\ L :: 'n::euclidean\text{-}space\ set$
assumes $S \neq \{\}$
and *affine S*
and *subspace L*
and $affine\text{-}parallel\ S\ L$
shows $aff\text{-}dim\ S = int\ (dim\ L)$
⟨*proof*⟩

lemma *dim-affine-hull*:
fixes $S :: 'n::euclidean\text{-}space\ set$
shows $dim\ (affine\ hull\ S) = dim\ S$
⟨*proof*⟩

lemma *aff-dim-subspace*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes $S \neq \{\}$
and *subspace S*
shows $aff\text{-}dim\ S = int\ (dim\ S)$
⟨*proof*⟩

lemma *aff-dim-zero*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes $0 \in affine\ hull\ S$
shows $aff\text{-}dim\ S = int\ (dim\ S)$
⟨*proof*⟩

lemma *aff-dim-univ*: *aff-dim* (*UNIV* :: *'n*::*euclidean-space set*) = *int*(*DIM*(*'n*))
⟨*proof*⟩

lemma *aff-dim-geq*:
fixes *V* :: *'n*::*euclidean-space set*
shows *aff-dim V* ≥ −*1*
⟨*proof*⟩

lemma *independent-card-le-aff-dim*:
fixes *B* :: *'n*::*euclidean-space set*
assumes *B* ⊆ *V*
assumes ¬ *affine-dependent B*
shows *int* (*card B*) ≤ *aff-dim V* + *1*
⟨*proof*⟩

lemma *aff-dim-subset*:
fixes *S T* :: *'n*::*euclidean-space set*
assumes *S* ⊆ *T*
shows *aff-dim S* ≤ *aff-dim T*
⟨*proof*⟩

lemma *aff-dim-subset-univ*:
fixes *S* :: *'n*::*euclidean-space set*
shows *aff-dim S* ≤ *int* (*DIM*(*'n*))
⟨*proof*⟩

lemma *affine-dim-equal*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *affine S affine T S* ≠ {} *S* ⊆ *T aff-dim S* = *aff-dim T*
shows *S* = *T*
⟨*proof*⟩

lemma *affine-hull-univ*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *aff-dim S* = *int*(*DIM*(*'n*))
shows *affine hull S* = (*UNIV* :: (*'n*::*euclidean-space*) *set*)
⟨*proof*⟩

lemma *aff-dim-convex-hull*:
fixes *S* :: *'n*::*euclidean-space set*
shows *aff-dim* (*convex hull S*) = *aff-dim S*
⟨*proof*⟩

lemma *aff-dim-cball*:
fixes *a* :: *'n*::*euclidean-space*
assumes *e* > *0*
shows *aff-dim* (*cball a e*) = *int* (*DIM*(*'n*))
⟨*proof*⟩

lemma *aff-dim-open*:
fixes S :: *'n::euclidean-space set*
assumes *open* S
and $S \neq \{\}$
shows *aff-dim* S = *int* (*DIM*(*'n*))
⟨*proof*⟩

lemma *low-dim-interior*:
fixes S :: *'n::euclidean-space set*
assumes ¬ *aff-dim* S = *int* (*DIM*(*'n*))
shows *interior* $S = \{\}$
⟨*proof*⟩

corollary *empty-interior-lowdim*:
fixes S :: *'n::euclidean-space set*
shows *dim* S < *DIM* (*'n*) ⟹ *interior* $S = \{\}$
⟨*proof*⟩

19.9 Caratheodory's theorem.

lemma *convex-hull-caratheodory-aff-dim*:
fixes p :: (*'a::euclidean-space*) *set*
shows *convex hull* p =
{y. ∃ s u. *finite* s ∧ $s \subseteq p$ ∧ *card* s ≤ *aff-dim* p + 1 ∧
(∀ x∈s. 0 ≤ u x) ∧ *setsum* u s = 1 ∧ *setsum* (λv. u v ∗$_R$ v) s = y}
⟨*proof*⟩

lemma *caratheodory-aff-dim*:
fixes p :: (*'a::euclidean-space*) *set*
shows *convex hull* p = {x. ∃ s. *finite* s ∧ $s \subseteq p$ ∧ *card* s ≤ *aff-dim* p + 1 ∧ x ∈ *convex hull* s}
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *convex-hull-caratheodory*:
fixes p :: (*'a::euclidean-space*) *set*
shows *convex hull* p =
{y. ∃ s u. *finite* s ∧ $s \subseteq p$ ∧ *card* s ≤ *DIM*(*'a*) + 1 ∧
(∀ x∈s. 0 ≤ u x) ∧ *setsum* u s = 1 ∧ *setsum* (λv. u v ∗$_R$ v) s = y}
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

theorem *caratheodory*:
convex hull p =
{x::*'a::euclidean-space*. ∃ s. *finite* s ∧ $s \subseteq p$ ∧
card s ≤ *DIM*(*'a*) + 1 ∧ x ∈ *convex hull* s}
⟨*proof*⟩

19.10 Relative interior of a set

definition *rel-interior S* =
$\{x.\ \exists\, T.\ openin\ (subtopology\ euclidean\ (affine\ hull\ S))\ T \wedge x \in T \wedge T \subseteq S\}$

lemma *rel-interior*:
rel-interior S = $\{x \in S.\ \exists\, T.\ open\ T \wedge x \in T \wedge T \cap affine\ hull\ S \subseteq S\}$
⟨*proof*⟩

lemma *mem-rel-interior*: $x \in rel\text{-}interior\ S \longleftrightarrow (\exists\, T.\ open\ T \wedge x \in T \cap S \wedge T \cap affine\ hull\ S \subseteq S)$
⟨*proof*⟩

lemma *mem-rel-interior-ball*:
$x \in rel\text{-}interior\ S \longleftrightarrow x \in S \wedge (\exists\, e.\ e > 0 \wedge ball\ x\ e \cap affine\ hull\ S \subseteq S)$
⟨*proof*⟩

lemma *rel-interior-ball*:
rel-interior S = $\{x \in S.\ \exists\, e.\ e > 0 \wedge ball\ x\ e \cap affine\ hull\ S \subseteq S\}$
⟨*proof*⟩

lemma *mem-rel-interior-cball*:
$x \in rel\text{-}interior\ S \longleftrightarrow x \in S \wedge (\exists\, e.\ e > 0 \wedge cball\ x\ e \cap affine\ hull\ S \subseteq S)$
⟨*proof*⟩

lemma *rel-interior-cball*:
rel-interior S = $\{x \in S.\ \exists\, e.\ e > 0 \wedge cball\ x\ e \cap affine\ hull\ S \subseteq S\}$
⟨*proof*⟩

lemma *rel-interior-empty* [*simp*]: *rel-interior* $\{\}$ = $\{\}$
⟨*proof*⟩

lemma *affine-hull-sing* [*simp*]: *affine hull* $\{a :: 'n{::}euclidean\text{-}space\}$ = $\{a\}$
⟨*proof*⟩

lemma *rel-interior-sing* [*simp*]: *rel-interior* $\{a :: 'n{::}euclidean\text{-}space\}$ = $\{a\}$
⟨*proof*⟩

lemma *subset-rel-interior*:
fixes *S T* :: $'n{::}euclidean\text{-}space\ set$
assumes $S \subseteq T$
and *affine hull S* = *affine hull T*
shows *rel-interior S* ⊆ *rel-interior T*
⟨*proof*⟩

lemma *rel-interior-subset*: *rel-interior S* ⊆ *S*
⟨*proof*⟩

lemma *rel-interior-subset-closure*: *rel-interior S* ⊆ *closure S*
⟨*proof*⟩

lemma *interior-subset-rel-interior*: *interior S* $\subseteq$ *rel-interior S*
⟨*proof*⟩

lemma *interior-rel-interior*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *aff-dim S* = *int*(*DIM*(*'n*))
shows *rel-interior S* = *interior S*
⟨*proof*⟩

lemma *rel-interior-interior*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *affine hull S* = *UNIV*
shows *rel-interior S* = *interior S*
⟨*proof*⟩

lemma *rel-interior-open*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *open S*
shows *rel-interior S* = *S*
⟨*proof*⟩

lemma *interior-ball* [*simp*]: *interior* (*ball x e*) = *ball x e*
⟨*proof*⟩

lemma *interior-rel-interior-gen*:
fixes *S* :: *'n*::*euclidean-space set*
shows *interior S* = (*if aff-dim S* = *int*(*DIM*(*'n*)) *then rel-interior S else* {})
⟨*proof*⟩

lemma *rel-interior-univ*:
fixes *S* :: *'n*::*euclidean-space set*
shows *rel-interior* (*affine hull S*) = *affine hull S*
⟨*proof*⟩

lemma *rel-interior-univ2*: *rel-interior* (*UNIV* :: (*'n*::*euclidean-space*) *set*) = *UNIV*
⟨*proof*⟩

lemma *rel-interior-convex-shrink*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *convex S*
and *c* $\in$ *rel-interior S*
and *x* $\in$ *S*
and *0* < *e*
and *e* $\leq$ *1*
shows *x* − *e* $*_R$ (*x* − *c*) $\in$ *rel-interior S*
⟨*proof*⟩

lemma *interior-real-semiline*:

fixes a :: *real*
shows *interior* $\{a..\} = \{a<..\}$
⟨*proof*⟩

lemma *continuous-ge-on-Ioo*:
assumes *continuous-on* $\{c..d\}$ g $\bigwedge x.\ x \in \{c<..<d\} \Longrightarrow g\ x \geq a\ c < d\ x \in \{c..d\}$
shows g (x::*real*) $\geq$ (a::*real*)
⟨*proof*⟩

lemma *interior-real-semiline'*:
fixes a :: *real*
shows *interior* $\{..a\} = \{..<a\}$
⟨*proof*⟩

lemma *interior-atLeastAtMost-real*: *interior* $\{a..b\} = \{a<..<b$ :: *real*$\}$
⟨*proof*⟩

lemma *frontier-real-Iic*:
fixes a :: *real*
shows *frontier* $\{..a\} = \{a\}$
⟨*proof*⟩

lemma *rel-interior-real-box*:
fixes a b :: *real*
assumes $a < b$
shows *rel-interior* $\{a\ ..\ b\} = \{a\ <..<\ b\}$
⟨*proof*⟩

lemma *rel-interior-real-semiline*:
fixes a :: *real*
shows *rel-interior* $\{a..\} = \{a<..\}$
⟨*proof*⟩

19.10.1 Relative open sets

definition *rel-open* $S \longleftrightarrow$ *rel-interior* $S = S$

lemma *rel-open*: *rel-open* $S \longleftrightarrow$ *openin* (*subtopology euclidean* (*affine hull* S)) S
⟨*proof*⟩

lemma *opein-rel-interior*: *openin* (*subtopology euclidean* (*affine hull* S)) (*rel-interior* S)
⟨*proof*⟩

lemma *affine-rel-open*:
fixes S :: *'n*::*euclidean-space set*
assumes *affine* S
shows *rel-open* S

⟨*proof*⟩

lemma *affine-closed*:
fixes S :: *'n*::*euclidean-space set*
assumes *affine S*
shows *closed S*
⟨*proof*⟩

lemma *closure-affine-hull*:
fixes S :: *'n*::*euclidean-space set*
shows *closure S* ⊆ *affine hull S*
⟨*proof*⟩

lemma *closure-same-affine-hull* [*simp*]:
fixes S :: *'n*::*euclidean-space set*
shows *affine hull* (*closure S*) = *affine hull S*
⟨*proof*⟩

lemma *closure-aff-dim*:
fixes S :: *'n*::*euclidean-space set*
shows *aff-dim* (*closure S*) = *aff-dim S*
⟨*proof*⟩

lemma *rel-interior-closure-convex-shrink*:
fixes S :: -::*euclidean-space set*
assumes *convex S*
and $c \in$ *rel-interior S*
and $x \in$ *closure S*
and $e > 0$
and $e \leq 1$
shows $x - e *_R (x - c) \in$ *rel-interior S*
⟨*proof*⟩

lemma *rel-interior-eq*:
rel-interior s = *s* ⟷ *openin*(*subtopology euclidean* (*affine hull s*)) *s*
⟨*proof*⟩

lemma *rel-interior-openin*:
openin(*subtopology euclidean* (*affine hull s*)) *s* ⟹ *rel-interior s* = *s*
⟨*proof*⟩

19.10.2 Relative interior preserves under linear transformations

lemma *rel-interior-translation-aux*:
fixes a :: *'n*::*euclidean-space*
shows ((λ*x*. *a* + *x*) ‘ *rel-interior S*) ⊆ *rel-interior* ((λ*x*. *a* + *x*) ‘ *S*)
⟨*proof*⟩

lemma *rel-interior-translation*:

fixes $a :: 'n::euclidean\text{-}space$
shows *rel-interior* $((\lambda x.\ a + x)\ `\ S) = (\lambda x.\ a + x)\ `$ *rel-interior* S
⟨*proof*⟩

lemma *affine-hull-linear-image*:
assumes *bounded-linear* f
shows $f\ `$ (*affine hull* $s$) = *affine hull* $f\ `\ s$
⟨*proof*⟩

lemma *rel-interior-injective-on-span-linear-image*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::euclidean\text{-}space$
and $S :: 'm::euclidean\text{-}space\ set$
assumes *bounded-linear* f
and *inj-on* f (*span* S)
shows *rel-interior* $(f\ `\ S) = f\ `$ (*rel-interior* S)
⟨*proof*⟩

lemma *rel-interior-injective-linear-image*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::euclidean\text{-}space$
assumes *bounded-linear* f
and *inj* f
shows *rel-interior* $(f\ `\ S) = f\ `$ (*rel-interior* S)
⟨*proof*⟩

19.11 Some Properties of subset of standard basis

lemma *affine-hull-substd-basis*:
assumes $d \subseteq Basis$
shows *affine hull* (*insert* 0 d) = $\{x::'a::euclidean\text{-}space.\ \forall i{\in}Basis.\ i \notin d \longrightarrow x{\cdot}i = 0\}$
(**is** *affine hull* (*insert* 0 *?A*) = *?B*)
⟨*proof*⟩

lemma *affine-hull-convex-hull* [*simp*]: *affine hull* (*convex hull* S) = *affine hull* S
⟨*proof*⟩

19.12 Openness and compactness are preserved by convex hull operation.

lemma *open-convex-hull*[*intro*]:
fixes $s :: 'a::real\text{-}normed\text{-}vector\ set$
assumes *open* s
shows *open* (*convex hull* s)
⟨*proof*⟩

lemma *compact-convex-combinations*:
fixes $s\ t :: 'a::real\text{-}normed\text{-}vector\ set$

assumes *compact s compact t*
shows *compact* { $(1 - u) *_R x + u *_R y \mid x\ y\ u.\ 0 \leq u \wedge u \leq 1 \wedge x \in s \wedge y \in t$}
⟨*proof*⟩

lemma *finite-imp-compact-convex-hull*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *finite s*
shows *compact* (*convex hull s*)
⟨*proof*⟩

lemma *compact-convex-hull*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *compact s*
shows *compact* (*convex hull s*)
⟨*proof*⟩

19.13 Extremal points of a simplex are some vertices.

lemma *dist-increases-online*:
fixes *a b d* :: *'a*::*real-inner*
assumes $d \neq 0$
shows *dist a* $(b + d) >$ *dist a b* $\vee$ *dist a* $(b - d) >$ *dist a b*
⟨*proof*⟩

lemma *norm-increases-online*:
fixes *d* :: *'a*::*real-inner*
shows $d \neq 0 \Longrightarrow$ *norm* $(a + d) >$ *norm a* $\vee$ $norm(a - d) >$ *norm a*
⟨*proof*⟩

lemma *simplex-furthest-lt*:
fixes *s* :: *'a*::*real-inner set*
assumes *finite s*
shows $\forall x \in$ *convex hull s*. $x \notin s \longrightarrow (\exists y \in$ *convex hull s*. *norm* $(x - a) <$ $norm(y - a))$
⟨*proof*⟩

lemma *simplex-furthest-le*:
fixes *s* :: *'a*::*real-inner set*
assumes *finite s*
and $s \neq \{\}$
shows $\exists y \in s.\ \forall x \in$ *convex hull s*. *norm* $(x - a) \leq$ *norm* $(y - a)$
⟨*proof*⟩

lemma *simplex-furthest-le-exists*:
fixes *s* :: (*'a*::*real-inner*) *set*
shows *finite s* $\Longrightarrow \forall x \in$(*convex hull s*). $\exists y \in s.$ *norm* $(x - a) \leq$ *norm* $(y - a)$
⟨*proof*⟩

lemma *simplex-extremal-le*:
fixes $s :: 'a::real\text{-}inner\ set$
assumes *finite s*
and $s \neq \{\}$
shows $\exists u{\in}s.\ \exists v{\in}s.\ \forall x{\in}convex\ hull\ s.\ \forall y \in convex\ hull\ s.\ norm\ (x - y) \leq norm\ (u - v)$
⟨*proof*⟩

lemma *simplex-extremal-le-exists*:
fixes $s :: 'a::real\text{-}inner\ set$
shows $finite\ s \Longrightarrow x \in convex\ hull\ s \Longrightarrow y \in convex\ hull\ s \Longrightarrow$
$\exists u{\in}s.\ \exists v{\in}s.\ norm\ (x - y) \leq norm\ (u - v)$
⟨*proof*⟩

19.14 Closest point of a convex set is unique, with a continuous projection.

definition $closest\text{-}point :: 'a::\{real\text{-}inner,heine\text{-}borel\}\ set \Rightarrow 'a \Rightarrow 'a$
where $closest\text{-}point\ s\ a = (SOME\ x.\ x \in s \wedge (\forall y{\in}s.\ dist\ a\ x \leq dist\ a\ y))$

lemma *closest-point-exists*:
assumes *closed s*
and $s \neq \{\}$
shows $closest\text{-}point\ s\ a \in s$
and $\forall y{\in}s.\ dist\ a\ (closest\text{-}point\ s\ a) \leq dist\ a\ y$
⟨*proof*⟩

lemma *closest-point-in-set*: $closed\ s \Longrightarrow s \neq \{\} \Longrightarrow closest\text{-}point\ s\ a \in s$
⟨*proof*⟩

lemma *closest-point-le*: $closed\ s \Longrightarrow x \in s \Longrightarrow dist\ a\ (closest\text{-}point\ s\ a) \leq dist\ a\ x$
⟨*proof*⟩

lemma *closest-point-self*:
assumes $x \in s$
shows $closest\text{-}point\ s\ x = x$
⟨*proof*⟩

lemma *closest-point-refl*: $closed\ s \Longrightarrow s \neq \{\} \Longrightarrow closest\text{-}point\ s\ x = x \longleftrightarrow x \in s$
⟨*proof*⟩

lemma *closer-points-lemma*:
assumes $inner\ y\ z > 0$
shows $\exists u{>}0.\ \forall v{>}0.\ v \leq u \longrightarrow norm(v *_R z - y) < norm\ y$
⟨*proof*⟩

lemma *closer-point-lemma*:

assumes *inner* $(y - x)$ $(z - x) > 0$
shows $\exists u{>}0.\ u \leq 1 \wedge$ *dist* $(x + u *_R (z - x))$ $y <$ *dist* x y
⟨*proof*⟩

lemma *any-closest-point-dot*:
assumes *convex s closed s* $x \in s$ $y \in s$ $\forall z{\in}s.$ *dist* a $x \leq$ *dist* a z
shows *inner* $(a - x)$ $(y - x) \leq 0$
⟨*proof*⟩

lemma *any-closest-point-unique*:
fixes x :: *'a*::*real-inner*
assumes *convex s closed s* $x \in s$ $y \in s$
$\forall z{\in}s.$ *dist* a $x \leq$ *dist* a z $\forall z{\in}s.$ *dist* a $y \leq$ *dist* a z
shows $x = y$
⟨*proof*⟩

lemma *closest-point-unique*:
assumes *convex s closed s* $x \in s$ $\forall z{\in}s.$ *dist* a $x \leq$ *dist* a z
shows $x =$ *closest-point s a*
⟨*proof*⟩

lemma *closest-point-dot*:
assumes *convex s closed s* $x \in s$
shows *inner* $(a -$ *closest-point s a*$)$ $(x -$ *closest-point s a*$) \leq 0$
⟨*proof*⟩

lemma *closest-point-lt*:
assumes *convex s closed s* $x \in s$ $x \neq$ *closest-point s a*
shows *dist a* (*closest-point s a*) $<$ *dist a x*
⟨*proof*⟩

lemma *closest-point-lipschitz*:
assumes *convex s*
and *closed s* $s \neq \{\}$
shows *dist* (*closest-point s x*) (*closest-point s y*) $\leq$ *dist x y*
⟨*proof*⟩

lemma *continuous-at-closest-point*:
assumes *convex s*
and *closed s*
and $s \neq \{\}$
shows *continuous* (*at x*) (*closest-point s*)
⟨*proof*⟩

lemma *continuous-on-closest-point*:
assumes *convex s*
and *closed s*
and $s \neq \{\}$
shows *continuous-on t* (*closest-point s*)

⟨*proof*⟩

19.14.1 Various point-to-set separating/supporting hyperplane theorems.

lemma *supporting-hyperplane-closed-point*:
fixes $z :: 'a::\{real\text{-}inner,heine\text{-}borel\}$
assumes *convex s*
and *closed s*
and $s \neq \{\}$
and $z \notin s$
shows $\exists\, a\ b.\ \exists\, y{\in}s.\ inner\ a\ z < b \wedge inner\ a\ y = b \wedge (\forall\, x{\in}s.\ inner\ a\ x \geq b)$
⟨*proof*⟩

lemma *separating-hyperplane-closed-point*:
fixes $z :: 'a::\{real\text{-}inner,heine\text{-}borel\}$
assumes *convex s*
and *closed s*
and $z \notin s$
shows $\exists\, a\ b.\ inner\ a\ z < b \wedge (\forall\, x{\in}s.\ inner\ a\ x > b)$
⟨*proof*⟩

lemma *separating-hyperplane-closed-0*:
assumes *convex* $(s::('a::euclidean\text{-}space)\ set)$
and *closed s*
and $0 \notin s$
shows $\exists\, a\ b.\ a \neq 0 \wedge 0 < b \wedge (\forall\, x{\in}s.\ inner\ a\ x > b)$
⟨*proof*⟩

19.14.2 Now set-to-set for closed/compact sets

lemma *separating-hyperplane-closed-compact*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *convex s*
and *closed s*
and *convex t*
and *compact t*
and $t \neq \{\}$
and $s \cap t = \{\}$
shows $\exists\, a\ b.\ (\forall\, x{\in}s.\ inner\ a\ x < b) \wedge (\forall\, x{\in}t.\ inner\ a\ x > b)$
⟨*proof*⟩

lemma *separating-hyperplane-compact-closed*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *convex s*
and *compact s*
and $s \neq \{\}$
and *convex t*
and *closed t*
and $s \cap t = \{\}$

shows $\exists\, a\ b.\ (\forall\, x{\in}s.\ inner\ a\ x < b) \wedge (\forall\, x{\in}t.\ inner\ a\ x > b)$
⟨*proof*⟩

19.14.3 General case without assuming closure and getting non-strict separation

lemma *separating-hyperplane-set-0*:
assumes *convex s* $(0{::}'a{::}euclidean\text{-}space) \notin s$
shows $\exists\, a.\ a \neq 0 \wedge (\forall\, x{\in}s.\ 0 \leq inner\ a\ x)$
⟨*proof*⟩

lemma *separating-hyperplane-sets*:
fixes *s t* :: *'a*::*euclidean-space set*
assumes *convex s*
and *convex t*
and $s \neq \{\}$
and $t \neq \{\}$
and $s \cap t = \{\}$
shows $\exists\, a\ b.\ a \neq 0 \wedge (\forall\, x{\in}s.\ inner\ a\ x \leq b) \wedge (\forall\, x{\in}t.\ inner\ a\ x \geq b)$
⟨*proof*⟩

19.15 More convexity generalities

lemma *convex-closure* [*intro*,*simp*]:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *convex s*
shows *convex* (*closure s*)
⟨*proof*⟩

lemma *convex-interior* [*intro*,*simp*]:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *convex s*
shows *convex* (*interior s*)
⟨*proof*⟩

lemma *convex-hull-eq-empty*[*simp*]: *convex hull s* $= \{\} \longleftrightarrow s = \{\}$
⟨*proof*⟩

19.16 Moving and scaling convex hulls.

lemma *convex-hull-set-plus*:
convex hull $(s + t) =$ *convex hull s* + *convex hull t*
⟨*proof*⟩

lemma *translation-eq-singleton-plus*: $(\lambda x.\ a + x)\ `\ t = \{a\} + t$
⟨*proof*⟩

lemma *convex-hull-translation*:
convex hull $((\lambda x.\ a + x)\ `\ s) = (\lambda x.\ a + x)\ `$ (*convex hull s*)
⟨*proof*⟩

lemma *convex-hull-scaling*:
convex hull ((λ*x*. *c* ∗$_R$ *x*) ‘ *s*) = (λ*x*. *c* ∗$_R$ *x*) ‘ (*convex hull s*)
⟨*proof*⟩

lemma *convex-hull-affinity*:
convex hull ((λ*x*. *a* + *c* ∗$_R$ *x*) ‘ *s*) = (λ*x*. *a* + *c* ∗$_R$ *x*) ‘ (*convex hull s*)
⟨*proof*⟩

19.17 Convexity of cone hulls

lemma *convex-cone-hull*:
assumes *convex S*
shows *convex* (*cone hull S*)
⟨*proof*⟩

lemma *cone-convex-hull*:
assumes *cone S*
shows *cone* (*convex hull S*)
⟨*proof*⟩

19.18 Convex set as intersection of halfspaces

lemma *convex-halfspace-intersection*:
fixes *s* :: ('*a*::*euclidean-space*) *set*
assumes *closed s convex s*
shows *s* = ⋂{*h*. *s* ⊆ *h* ∧ (∃ *a b*. *h* = {*x*. *inner a x* ≤ *b*})}
⟨*proof*⟩

19.19 Radon's theorem (from Lars Schewe)

lemma *radon-ex-lemma*:
assumes *finite c affine-dependent c*
shows ∃ *u*. *setsum u c* = *0* ∧ (∃ *v*∈*c*. *u v* ≠ *0*) ∧ *setsum* (λ*v*. *u v* ∗$_R$ *v*) *c* = *0*
⟨*proof*⟩

lemma *radon-s-lemma*:
assumes *finite s*
and *setsum f s* = (*0*::*real*)
shows *setsum f* {*x*∈*s*. *0* < *f x*} = − *setsum f* {*x*∈*s*. *f x* < *0*}
⟨*proof*⟩

lemma *radon-v-lemma*:
assumes *finite s*
and *setsum f s* = *0*
and ∀ *x*. *g x* = (*0*::*real*) ⟶ *f x* = (*0*::'*a*::*euclidean-space*)
shows (*setsum f* {*x*∈*s*. *0* < *g x*}) = − *setsum f* {*x*∈*s*. *g x* < *0*}
⟨*proof*⟩

lemma *radon-partition*:

assumes *finite c affine-dependent c*
shows $\exists m\ p.\ m \cap p = \{\} \wedge m \cup p = c \wedge (\textit{convex hull } m) \cap (\textit{convex hull } p) \neq \{\}$
⟨*proof*⟩

lemma *radon*:
assumes *affine-dependent c*
obtains *m p* **where** $m \subseteq c\ p \subseteq c\ m \cap p = \{\}\ (\textit{convex hull } m) \cap (\textit{convex hull } p) \neq \{\}$
⟨*proof*⟩

19.20 Helly's theorem

lemma *helly-induct*:
fixes *f* :: *'a*::*euclidean-space set set*
assumes *card f* = *n*
and $n \geq \textit{DIM}('a) + 1$
and $\forall s{\in}f.\ \textit{convex } s\ \forall t{\subseteq}f.\ \textit{card } t = \textit{DIM}('a) + 1 \longrightarrow \bigcap t \neq \{\}$
shows $\bigcap f \neq \{\}$
⟨*proof*⟩

lemma *helly*:
fixes *f* :: *'a*::*euclidean-space set set*
assumes $\textit{card } f \geq \textit{DIM}('a) + 1\ \forall s{\in}f.\ \textit{convex } s$
and $\forall t{\subseteq}f.\ \textit{card } t = \textit{DIM}('a) + 1 \longrightarrow \bigcap t \neq \{\}$
shows $\bigcap f \neq \{\}$
⟨*proof*⟩

19.21 Homeomorphism of all convex compact sets with nonempty interior

lemma *compact-frontier-line-lemma*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *compact s*
and *0* ∈ *s*
and *x* ≠ *0*
obtains *u* **where** $0 \leq u$ **and** $(u *_R x) \in \textit{frontier } s\ \forall v{>}u.\ (v *_R x) \notin s$
⟨*proof*⟩

lemma *starlike-compact-projective*:
assumes *compact s*
and *cball* (*0*::*'a*::*euclidean-space*) *1* ⊆ *s*
and $\forall x{\in}s.\ \forall u.\ 0 \leq u \wedge u < 1 \longrightarrow u *_R x \in s - \textit{frontier } s$
shows *s homeomorphic* (*cball* (*0*::*'a*::*euclidean-space*) *1*)
⟨*proof*⟩

lemma *homeomorphic-convex-compact-lemma*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *convex s*

and *compact s*
and *cball 0 1* $\subseteq$ *s*
shows *s homeomorphic* (*cball* (*0*::*'a*) *1*)
⟨*proof*⟩

lemma *homeomorphic-convex-compact-cball*:
fixes *e* :: *real*
and *s* :: *'a*::*euclidean-space set*
assumes *convex s*
and *compact s*
and *interior s* $\neq$ {}
and *e* > *0*
shows *s homeomorphic* (*cball* (*b*::*'a*) *e*)
⟨*proof*⟩

lemma *homeomorphic-convex-compact*:
fixes *s* :: *'a*::*euclidean-space set*
and *t* :: *'a set*
assumes *convex s compact s interior s* $\neq$ {}
and *convex t compact t interior t* $\neq$ {}
shows *s homeomorphic t*
⟨*proof*⟩

19.22 Epigraphs of convex functions

definition *epigraph s* (*f* :: - $\Rightarrow$ *real*) = {*xy*. *fst xy* $\in$ *s* $\wedge$ *f* (*fst xy*) $\leq$ *snd xy*}

lemma *mem-epigraph*: (*x*, *y*) $\in$ *epigraph s f* $\longleftrightarrow$ *x* $\in$ *s* $\wedge$ *f x* $\leq$ *y*
⟨*proof*⟩

lemma *convex-epigraph*: *convex* (*epigraph s f*) $\longleftrightarrow$ *convex-on s f* $\wedge$ *convex s*
⟨*proof*⟩

lemma *convex-epigraphI*: *convex-on s f* $\Longrightarrow$ *convex s* $\Longrightarrow$ *convex* (*epigraph s f*)
⟨*proof*⟩

lemma *convex-epigraph-convex*: *convex s* $\Longrightarrow$ *convex-on s f* $\longleftrightarrow$ *convex*(*epigraph s f*)
⟨*proof*⟩

19.22.1 Use this to derive general bound property of convex function

lemma *convex-on*:
assumes *convex s*
shows *convex-on s f* $\longleftrightarrow$
(∀ *k u x*. (∀ *i*∈{*1*..*k*::*nat*}. *0* $\leq$ *u i* $\wedge$ *x i* $\in$ *s*) $\wedge$ *setsum u* {*1*..*k*} = *1* $\longrightarrow$
f (*setsum* (λ*i*. *u i* $*_R$ *x i*) {*1*..*k*}) $\leq$ *setsum* (λ*i*. *u i* * *f*(*x i*)) {*1*..*k*})
⟨*proof*⟩

19.23 Convexity of general and special intervals

lemma *is-interval-convex*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *is-interval s*
shows *convex s*
⟨*proof*⟩

lemma *is-interval-connected*:
fixes $s :: 'a::euclidean\text{-}space\ set$
shows *is-interval s* $\Longrightarrow$ *connected s*
⟨*proof*⟩

lemma *convex-box* [*simp*]: *convex* (*cbox a b*) *convex* (*box a* (*b*::′*a*::*euclidean-space*))
⟨*proof*⟩

19.24 On *real*, *is-interval*, *convex* and *connected* are all equivalent.

lemma *is-interval-1*:
is-interval (*s*::*real set*) $\longleftrightarrow$ $(\forall\, a{\in}s.\ \forall\, b{\in}s.\ \forall\ x.\ a \le x \wedge x \le b \longrightarrow x \in s)$
⟨*proof*⟩

lemma *is-interval-connected-1*:
fixes $s :: real\ set$
shows *is-interval s* $\longleftrightarrow$ *connected s*
⟨*proof*⟩

lemma *is-interval-convex-1*:
fixes $s :: real\ set$
shows *is-interval s* $\longleftrightarrow$ *convex s*
⟨*proof*⟩

lemma *connected-convex-1*:
fixes $s :: real\ set$
shows *connected s* $\longleftrightarrow$ *convex s*
⟨*proof*⟩

lemma *connected-convex-1-gen*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
assumes $DIM('a) = 1$
shows *connected s* $\longleftrightarrow$ *convex s*
⟨*proof*⟩

19.25 Another intermediate value theorem formulation

lemma *ivt-increasing-component-on-1*:
fixes $f :: real \Rightarrow 'a::euclidean\text{-}space$
assumes $a \le b$
and *continuous-on* $\{a..b\}$ *f*

and $(f\ a)\cdot k \leq y\ y \leq (f\ b)\cdot k$
shows $\exists\, x{\in}\{a..b\}.\ (f\ x)\cdot k = y$
⟨*proof*⟩

lemma *ivt-increasing-component-1*:
fixes f :: *real* ⇒ $'a$::*euclidean-space*
shows $a \leq b \Longrightarrow \forall\, x{\in}\{a..b\}.$ *continuous* (*at* x) $f \Longrightarrow$
$f\ a\cdot k \leq y \Longrightarrow y \leq f\ b\cdot k \Longrightarrow \exists\, x{\in}\{a..b\}.\ (f\ x)\cdot k = y$
⟨*proof*⟩

lemma *ivt-decreasing-component-on-1*:
fixes f :: *real* ⇒ $'a$::*euclidean-space*
assumes $a \leq b$
and *continuous-on* $\{a..b\}$ f
and $(f\ b)\cdot k \leq y$
and $y \leq (f\ a)\cdot k$
shows $\exists\, x{\in}\{a..b\}.\ (f\ x)\cdot k = y$
⟨*proof*⟩

lemma *ivt-decreasing-component-1*:
fixes f :: *real* ⇒ $'a$::*euclidean-space*
shows $a \leq b \Longrightarrow \forall\, x{\in}\{a..b\}.$ *continuous* (*at* x) $f \Longrightarrow$
$f\ b\cdot k \leq y \Longrightarrow y \leq f\ a\cdot k \Longrightarrow \exists\, x{\in}\{a..b\}.\ (f\ x)\cdot k = y$
⟨*proof*⟩

19.26 A bound within a convex hull, and so an interval

lemma *convex-on-convex-hull-bound*:
assumes *convex-on* (*convex hull* s) f
and $\forall\, x{\in}s.\ f\ x \leq b$
shows $\forall\, x{\in}$ *convex hull* $s.\ f\ x \leq b$
⟨*proof*⟩

lemma *inner-setsum-Basis*[*simp*]: $i \in Basis \Longrightarrow (\sum Basis) \cdot i = 1$
⟨*proof*⟩

lemma *convex-set-plus*:
assumes *convex* s **and** *convex* t **shows** *convex* $(s + t)$
⟨*proof*⟩

lemma *convex-set-setsum*:
assumes $\bigwedge i.\ i \in A \Longrightarrow$ *convex* $(B\ i)$
shows *convex* $(\sum i{\in}A.\ B\ i)$
⟨*proof*⟩

lemma *finite-set-setsum*:
assumes *finite* A **and** $\forall\, i{\in}A.$ *finite* $(B\ i)$ **shows** *finite* $(\sum i{\in}A.\ B\ i)$
⟨*proof*⟩

lemma *set-setsum-eq*:
finite A $\Longrightarrow$ $(\sum i{\in}A.\ B\ i) = \{\sum i{\in}A.\ f\ i\ |f.\ \forall i{\in}A.\ f\ i \in B\ i\}$
⟨*proof*⟩

lemma *box-eq-set-setsum-Basis*:
shows $\{x.\ \forall i{\in}Basis.\ x{\cdot}i \in B\ i\} = (\sum i{\in}Basis.\ image\ (\lambda x.\ x *_R i)\ (B\ i))$
⟨*proof*⟩

lemma *convex-hull-set-setsum*:
convex hull $(\sum i{\in}A.\ B\ i) = (\sum i{\in}A.\ convex\ hull\ (B\ i))$
⟨*proof*⟩

lemma *convex-hull-eq-real-cbox*:
fixes *x y* :: *real* **assumes** $x \le y$
shows *convex hull* $\{x, y\} = cbox\ x\ y$
⟨*proof*⟩

lemma *unit-interval-convex-hull*:
cbox $(0{::}'a{::}euclidean\text{-}space)$ *One* = *convex hull* $\{x.\ \forall i{\in}Basis.\ (x{\cdot}i = 0) \vee (x{\cdot}i = 1)\}$
(**is** *?int* = *convex hull ?points*)
⟨*proof*⟩

And this is a finite set of vertices.

lemma *unit-cube-convex-hull*:
obtains *s* :: $'a{::}euclidean\text{-}space\ set$
where *finite s* **and** *cbox 0* $(\sum Basis)$ = *convex hull s*
⟨*proof*⟩

Hence any cube (could do any nonempty interval).

lemma *cube-convex-hull*:
assumes $d > 0$
obtains *s* :: $'a{::}euclidean\text{-}space\ set$ **where**
finite s **and** *cbox* $(x - (\sum i{\in}Basis.\ d{*_R}i))\ (x + (\sum i{\in}Basis.\ d{*_R}i))$ = *convex hull s*
⟨*proof*⟩

19.27 Bounded convex function on open set is continuous

lemma *convex-on-bounded-continuous*:
fixes *s* :: $('a{::}real\text{-}normed\text{-}vector)\ set$
assumes *open s*
and *convex-on s f*
and $\forall x{\in}s.\ |f\ x| \le b$
shows *continuous-on s f*
⟨*proof*⟩

19.28 Upper bound on a ball implies upper and lower bounds

lemma *convex-bounds-lemma*:

fixes $x :: 'a$::*real-normed-vector*
assumes *convex-on* (*cball x e*) *f*
and $\forall y \in$ *cball x e*. *f y* $\leq b$
shows $\forall y \in$ *cball x e*. $|f\ y| \leq b + 2 * |f\ x|$
⟨*proof*⟩

19.28.1 Hence a convex function on an open set is continuous

lemma *real-of-nat-ge-one-iff*: $1 \leq$ *real* (n::*nat*) $\longleftrightarrow 1 \leq n$
⟨*proof*⟩

lemma *convex-on-continuous*:
assumes *open* (s::($'a$::*euclidean-space*) *set*) *convex-on s f*
shows *continuous-on s f*
⟨*proof*⟩

19.29 Line segments, Starlike Sets, etc.

definition *midpoint* :: $'a$::*real-vector* $\Rightarrow$ $'a \Rightarrow$ $'a$
where *midpoint a b* = (*inverse* (2::*real*)) $*_R$ (*a* + *b*)

definition *closed-segment* :: $'a$::*real-vector* $\Rightarrow$ $'a \Rightarrow$ $'a$ *set*
where *closed-segment a b* = $\{(1 - u) *_R a + u *_R b \mid u$::*real*. $0 \leq u \wedge u \leq 1\}$

definition *open-segment* :: $'a$::*real-vector* $\Rightarrow$ $'a \Rightarrow$ $'a$ *set* **where**
open-segment a b ≡ *closed-segment a b* − {*a*,*b*}

lemmas *segment* = *open-segment-def closed-segment-def*

lemma *in-segment*:
x ∈ *closed-segment a b* ⟷ (∃ *u*. 0 ≤ *u* ∧ *u* ≤ 1 ∧ *x* = (1 − *u*) $*_R$ *a* + *u* $*_R$ *b*)
x ∈ *open-segment a b* ⟷ *a* ≠ *b* ∧ (∃ *u*. 0 < *u* ∧ *u* < 1 ∧ *x* = (1 − *u*) $*_R$ *a* + *u* $*_R$ *b*)
⟨*proof*⟩

definition *between* = (λ(*a*,*b*) *x*. *x* ∈ *closed-segment a b*)

definition *starlike s* ⟷ (∃ *a*∈*s*. ∀ *x*∈*s*. *closed-segment a x* ⊆ *s*)

lemma *starlike-UNIV* [*simp*]: *starlike UNIV*
⟨*proof*⟩

lemma *midpoint-refl*: *midpoint x x* = *x*
⟨*proof*⟩

lemma *midpoint-sym*: *midpoint a b* = *midpoint b a*
⟨*proof*⟩

lemma *midpoint-eq-iff*: *midpoint a b* = *c* ⟷ *a* + *b* = *c* + *c*

⟨*proof*⟩

lemma *dist-midpoint*:
fixes *a b* :: *'a*::*real-normed-vector* **shows**
dist a (*midpoint a b*) = (*dist a b*) / *2* (**is** *?t1*)
dist b (*midpoint a b*) = (*dist a b*) / *2* (**is** *?t2*)
dist (*midpoint a b*) *a* = (*dist a b*) / *2* (**is** *?t3*)
dist (*midpoint a b*) *b* = (*dist a b*) / *2* (**is** *?t4*)
⟨*proof*⟩

lemma *midpoint-eq-endpoint*:
midpoint a b = *a* ⟷ *a* = *b*
midpoint a b = *b* ⟷ *a* = *b*
⟨*proof*⟩

lemma *convex-contains-segment*:
convex s ⟷ (∀ *a*∈*s*. ∀ *b*∈*s*. *closed-segment a b* ⊆ *s*)
⟨*proof*⟩

lemma *closed-segment-subset*: ⟦*x* ∈ *s*; *y* ∈ *s*; *convex s*⟧ ⟹ *closed-segment x y* ⊆ *s*
⟨*proof*⟩

lemma *closed-segment-subset-convex-hull*:
⟦*x* ∈ *convex hull s*; *y* ∈ *convex hull s*⟧ ⟹ *closed-segment x y* ⊆ *convex hull s*
⟨*proof*⟩

lemma *convex-imp-starlike*:
convex s ⟹ *s* ≠ {} ⟹ *starlike s*
⟨*proof*⟩

lemma *segment-convex-hull*:
closed-segment a b = *convex hull* {*a*,*b*}
⟨*proof*⟩

lemma *open-closed-segment*: *u* ∈ *open-segment w z* ⟹ *u* ∈ *closed-segment w z*
⟨*proof*⟩

lemma *segment-open-subset-closed*:
open-segment a b ⊆ *closed-segment a b*
⟨*proof*⟩

lemma *bounded-closed-segment*:
fixes *a* :: *'a*::*euclidean-space* **shows** *bounded* (*closed-segment a b*)
⟨*proof*⟩

lemma *bounded-open-segment*:
fixes *a* :: *'a*::*euclidean-space* **shows** *bounded* (*open-segment a b*)
⟨*proof*⟩

lemmas *bounded-segment = bounded-closed-segment open-closed-segment*

lemma *ends-in-segment* [*iff*]: $a \in$ *closed-segment a b* $b \in$ *closed-segment a b*
⟨*proof*⟩

lemma *segment-furthest-le*:
fixes *a b x y* :: *'a*::*euclidean-space*
assumes $x \in$ *closed-segment a b*
shows *norm* $(y - x) \leq$ *norm* $(y - a) \vee$ *norm* $(y - x) \leq$ *norm* $(y - b)$
⟨*proof*⟩

lemma *closed-segment-commute*: *closed-segment a b = closed-segment b a*
⟨*proof*⟩

lemma *segment-bound1*:
assumes $x \in$ *closed-segment a b*
shows *norm* $(x - a) \leq$ *norm* $(b - a)$
⟨*proof*⟩

lemma *segment-bound*:
assumes $x \in$ *closed-segment a b*
shows *norm* $(x - a) \leq$ *norm* $(b - a)$ *norm* $(x - b) \leq$ *norm* $(b - a)$
⟨*proof*⟩

lemma *open-segment-commute*: *open-segment a b = open-segment b a*
⟨*proof*⟩

lemma *closed-segment-idem* [*simp*]: *closed-segment a a* $= \{a\}$
⟨*proof*⟩

lemma *open-segment-idem* [*simp*]: *open-segment a a* $= \{\}$
⟨*proof*⟩

lemma *closed-segment-eq-open*: *closed-segment a b = open-segment a b* $\cup \{a,b\}$
⟨*proof*⟩

lemma *closed-segment-eq-real-ivl*:
fixes *a b*::*real*
shows *closed-segment a b* = (*if a* $\leq$ *b then* $\{a$ .. $b\}$ *else* $\{b$ .. $a\}$)
⟨*proof*⟩

lemma *closed-segment-real-eq*:
fixes *u*::*real* **shows** *closed-segment u v* $= (\lambda x.\ (v - u) * x + u)$ ' $\{0..1\}$
⟨*proof*⟩

19.29.1 More lemmas, especially for working with the underlying formula

lemma *segment-eq-compose*:
fixes $a :: 'a :: real\text{-}vector$
shows $(\lambda u.\ (1 - u) *_R a + u *_R b) = (\lambda x.\ a + x)\ o\ (\lambda u.\ u *_R (b - a))$
⟨*proof*⟩

lemma *segment-degen-1*:
fixes $a :: 'a :: real\text{-}vector$
shows $(1 - u) *_R a + u *_R b = b \longleftrightarrow a{=}b \vee u{=}1$
⟨*proof*⟩

lemma *segment-degen-0*:
fixes $a :: 'a :: real\text{-}vector$
shows $(1 - u) *_R a + u *_R b = a \longleftrightarrow a{=}b \vee u{=}0$
⟨*proof*⟩

lemma *closed-segment-image-interval*:
closed-segment a b $= (\lambda u.\ (1 - u) *_R a + u *_R b)$ ` $\{0..1\}$
⟨*proof*⟩

lemma *open-segment-image-interval*:
open-segment a b $= (if\ a{=}b\ then\ \{\}\ else\ (\lambda u.\ (1 - u) *_R a + u *_R b)$ ` $\{0{<}..{<}1\})$
⟨*proof*⟩

lemmas *segment-image-interval* = *closed-segment-image-interval open-segment-image-interval*

lemma *open-segment-bound1*:
assumes $x \in$ *open-segment a b*
shows *norm* $(x - a) <$ *norm* $(b - a)$
⟨*proof*⟩

lemma *compact-segment* [*simp*]:
fixes $a :: 'a{::}real\text{-}normed\text{-}vector$
shows *compact* (*closed-segment a b*)
⟨*proof*⟩

lemma *closed-segment* [*simp*]:
fixes $a :: 'a{::}real\text{-}normed\text{-}vector$
shows *closed* (*closed-segment a b*)
⟨*proof*⟩

lemma *closure-closed-segment* [*simp*]:
fixes $a :: 'a{::}real\text{-}normed\text{-}vector$
shows *closure*(*closed-segment a b*) = *closed-segment a b*
⟨*proof*⟩

lemma *open-segment-bound*:

assumes $x \in$ *open-segment a b*
shows *norm* $(x - a) <$ *norm* $(b - a)$ *norm* $(x - b) <$ *norm* $(b - a)$
⟨*proof*⟩

lemma *closure-open-segment* [*simp*]:
fixes a :: $'a$::*euclidean-space*
shows *closure*(*open-segment a b*) = (*if a* = *b then* {} *else closed-segment a b*)
⟨*proof*⟩

lemma *closed-open-segment-iff* [*simp*]:
fixes a :: $'a$::*euclidean-space* **shows** *closed*(*open-segment a b*) ⟷ $a = b$
⟨*proof*⟩

lemma *compact-open-segment-iff* [*simp*]:
fixes a :: $'a$::*euclidean-space* **shows** *compact*(*open-segment a b*) ⟷ $a = b$
⟨*proof*⟩

lemma *convex-closed-segment* [*iff*]: *convex* (*closed-segment a b*)
⟨*proof*⟩

lemma *convex-open-segment* [*iff*]: *convex*(*open-segment a b*)
⟨*proof*⟩

lemmas *convex-segment* = *convex-closed-segment convex-open-segment*

lemma *connected-segment* [*iff*]:
fixes x :: $'a$ :: *real-normed-vector*
shows *connected* (*closed-segment x y*)
⟨*proof*⟩

lemma *affine-hull-closed-segment* [*simp*]:
affine hull (*closed-segment a b*) = *affine hull* {*a*,*b*}
⟨*proof*⟩

lemma *affine-hull-open-segment* [*simp*]:
fixes a :: $'a$::*euclidean-space*
shows *affine hull* (*open-segment a b*) = (*if a* = *b then* {} *else affine hull* {*a*,*b*})
⟨*proof*⟩

lemma *rel-interior-closure-convex-segment*:
fixes S :: -::*euclidean-space set*
assumes *convex S a* ∈ *rel-interior S b* ∈ *closure S*
shows *open-segment a b* ⊆ *rel-interior S*
⟨*proof*⟩

19.30 More results about segments

lemma *dist-half-times2*:
fixes a :: $'a$ :: *real-normed-vector*

shows *dist* ((*1* / *2*) $*_R$ (*a* + *b*)) *x* * *2* = *dist* (*a*+*b*) (*2* $*_R$ *x*)
⟨*proof*⟩

lemma *closed-segment-as-ball*:
closed-segment a b = *affine hull* {*a*,*b*} ∩ *cball*(*inverse 2* $*_R$ (*a* + *b*))(*norm*(*b* − *a*) / *2*)
⟨*proof*⟩

lemma *open-segment-as-ball*:
open-segment a b =
affine hull {*a*,*b*} ∩ *ball*(*inverse 2* $*_R$ (*a* + *b*))(*norm*(*b* − *a*) / *2*)
⟨*proof*⟩

lemmas *segment-as-ball* = *closed-segment-as-ball open-segment-as-ball*

lemma *closed-segment-neq-empty* [*simp*]: *closed-segment a b* ≠ {}
⟨*proof*⟩

lemma *open-segment-eq-empty* [*simp*]: *open-segment a b* = {} ⟷ *a* = *b*
⟨*proof*⟩

lemma *open-segment-eq-empty′* [*simp*]: {} = *open-segment a b* ⟷ *a* = *b*
⟨*proof*⟩

lemmas *segment-eq-empty* = *closed-segment-neq-empty open-segment-eq-empty*

lemma *inj-segment*:
fixes *a* :: *′a* :: *real-vector*
assumes *a* ≠ *b*
shows *inj-on* (λ*u*. (*1* − *u*) $*_R$ *a* + *u* $*_R$ *b*) *I*
⟨*proof*⟩

lemma *finite-closed-segment* [*simp*]: *finite*(*closed-segment a b*) ⟷ *a* = *b*
⟨*proof*⟩

lemma *finite-open-segment* [*simp*]: *finite*(*open-segment a b*) ⟷ *a* = *b*
⟨*proof*⟩

lemmas *finite-segment* = *finite-closed-segment finite-open-segment*

lemma *closed-segment-eq-sing*: *closed-segment a b* = {*c*} ⟷ *a* = *c* ∧ *b* = *c*
⟨*proof*⟩

lemma *open-segment-eq-sing*: *open-segment a b* ≠ {*c*}
⟨*proof*⟩

lemmas *segment-eq-sing* = *closed-segment-eq-sing open-segment-eq-sing*

lemma *subset-closed-segment*:

closed-segment a b ⊆ *closed-segment c d* ⟷
a ∈ *closed-segment c d* ∧ *b* ∈ *closed-segment c d*
⟨*proof*⟩

lemma *subset-co-segment*:
closed-segment a b ⊆ *open-segment c d* ⟷
a ∈ *open-segment c d* ∧ *b* ∈ *open-segment c d*
⟨*proof*⟩

lemma *subset-open-segment*:
fixes *a* :: *'a*::*euclidean-space*
shows *open-segment a b* ⊆ *open-segment c d* ⟷
a = *b* ∨ *a* ∈ *closed-segment c d* ∧ *b* ∈ *closed-segment c d*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *subset-oc-segment*:
fixes *a* :: *'a*::*euclidean-space*
shows *open-segment a b* ⊆ *closed-segment c d* ⟷
a = *b* ∨ *a* ∈ *closed-segment c d* ∧ *b* ∈ *closed-segment c d*
⟨*proof*⟩

lemmas *subset-segment* = *subset-closed-segment subset-co-segment subset-oc-segment subset-open-segment*

19.31 Betweenness

lemma *between-mem-segment*: *between* (*a*,*b*) *x* ⟷ *x* ∈ *closed-segment a b*
⟨*proof*⟩

lemma *between*: *between* (*a*, *b*) (*x*::*'a*::*euclidean-space*) ⟷ *dist a b* = (*dist a x*) + (*dist x b*)
⟨*proof*⟩

lemma *between-midpoint*:
fixes *a* :: *'a*::*euclidean-space*
shows *between* (*a*,*b*) (*midpoint a b*) (**is** *?t1*)
and *between* (*b*,*a*) (*midpoint a b*) (**is** *?t2*)
⟨*proof*⟩

lemma *between-mem-convex-hull*:
between (*a*,*b*) *x* ⟷ *x* ∈ *convex hull* {*a*,*b*}
⟨*proof*⟩

19.32 Shrinking towards the interior of a convex set

lemma *mem-interior-convex-shrink*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *convex s*
and *c* ∈ *interior s*

and $x \in s$
and $0 < e$
and $e \leq 1$
shows $x - e *_R (x - c) \in \mathit{interior}\ s$
⟨*proof*⟩

lemma *mem-interior-closure-convex-shrink*:
fixes $s :: 'a::\mathit{euclidean\text{-}space}\ \mathit{set}$
assumes *convex s*
and $c \in \mathit{interior}\ s$
and $x \in \mathit{closure}\ s$
and $0 < e$
and $e \leq 1$
shows $x - e *_R (x - c) \in \mathit{interior}\ s$
⟨*proof*⟩

19.33 Some obvious but surprisingly hard simplex lemmas

lemma *simplex*:
assumes *finite s*
and $0 \notin s$
shows *convex hull* (*insert 0 s*) =
$\{y.\ (\exists u.\ (\forall x{\in}s.\ 0 \leq u\ x) \wedge \mathit{setsum}\ u\ s \leq 1 \wedge \mathit{setsum}\ (\lambda x.\ u\ x *_R x)\ s = y)\}$
⟨*proof*⟩

lemma *substd-simplex*:
assumes *d*: $d \subseteq \mathit{Basis}$
shows *convex hull* (*insert 0 d*) =
$\{x.\ (\forall i{\in}\mathit{Basis}.\ 0 \leq x{\cdot}i) \wedge (\sum i{\in}d.\ x{\cdot}i) \leq 1 \wedge (\forall i{\in}\mathit{Basis}.\ i \notin d \longrightarrow x{\cdot}i = 0)\}$
(**is** *convex hull* (*insert 0 ?p*) = *?s*)
⟨*proof*⟩

lemma *std-simplex*:
convex hull (*insert 0 Basis*) =
$\{x::'a::\mathit{euclidean\text{-}space}.\ (\forall i{\in}\mathit{Basis}.\ 0 \leq x{\cdot}i) \wedge \mathit{setsum}\ (\lambda i.\ x{\cdot}i)\ \mathit{Basis} \leq 1\}$
⟨*proof*⟩

lemma *interior-std-simplex*:
interior (*convex hull* (*insert 0 Basis*)) =
$\{x::'a::\mathit{euclidean\text{-}space}.\ (\forall i{\in}\mathit{Basis}.\ 0 < x{\cdot}i) \wedge \mathit{setsum}\ (\lambda i.\ x{\cdot}i)\ \mathit{Basis} < 1\}$
⟨*proof*⟩

lemma *interior-std-simplex-nonempty*:
obtains $a :: 'a::\mathit{euclidean\text{-}space}$ **where**
$a \in \mathit{interior}(\mathit{convex\ hull}\ (\mathit{insert}\ 0\ \mathit{Basis}))$
⟨*proof*⟩

lemma *rel-interior-substd-simplex*:

assumes d: $d \subseteq Basis$
shows *rel-interior* (*convex hull* (*insert 0 d*)) =
$\{x::'a::euclidean\text{-}space.\ (\forall i{\in}d.\ 0 < x{\cdot}i) \wedge (\sum i{\in}d.\ x{\cdot}i) < 1 \wedge (\forall i{\in}Basis.\ i \notin d \longrightarrow x{\cdot}i = 0)\}$
(**is** *rel-interior* (*convex hull* (*insert 0 ?p*)) = *?s*)
⟨*proof*⟩

lemma *rel-interior-substd-simplex-nonempty*:
assumes $d \neq \{\}$
and $d \subseteq Basis$
obtains $a :: 'a::euclidean\text{-}space$
where $a \in$ *rel-interior* (*convex hull* (*insert 0 d*))
⟨*proof*⟩

19.34 Relative interior of convex set

lemma *rel-interior-convex-nonempty-aux*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes *convex S*
and $0 \in S$
shows *rel-interior S* $\neq \{\}$
⟨*proof*⟩

lemma *rel-interior-eq-empty*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes *convex S*
shows *rel-interior S* $= \{\} \longleftrightarrow S = \{\}$
⟨*proof*⟩

lemma *convex-rel-interior*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes *convex S*
shows *convex* (*rel-interior S*)
⟨*proof*⟩

lemma *convex-closure-rel-interior*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes *convex S*
shows *closure* (*rel-interior S*) = *closure S*
⟨*proof*⟩

lemma *rel-interior-same-affine-hull*:
fixes $S :: 'n::euclidean\text{-}space\ set$
assumes *convex S*
shows *affine hull* (*rel-interior S*) = *affine hull S*
⟨*proof*⟩

lemma *rel-interior-aff-dim*:
fixes $S :: 'n::euclidean\text{-}space\ set$

assumes *convex S*
shows *aff-dim* (*rel-interior S*) = *aff-dim S*
⟨*proof*⟩

lemma *rel-interior-rel-interior*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *convex S*
shows *rel-interior* (*rel-interior S*) = *rel-interior S*
⟨*proof*⟩

lemma *rel-interior-rel-open*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *convex S*
shows *rel-open* (*rel-interior S*)
⟨*proof*⟩

lemma *convex-rel-interior-closure-aux*:
fixes *x y z* :: *'n*::*euclidean-space*
assumes *0 < a 0 < b* (*a + b*) *∗R z* = *a ∗R x* + *b ∗R y*
obtains *e* **where** *0 < e e ≤ 1 z* = *y* − *e ∗R* (*y* − *x*)
⟨*proof*⟩

lemma *convex-rel-interior-closure*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *convex S*
shows *rel-interior* (*closure S*) = *rel-interior S*
⟨*proof*⟩

lemma *convex-interior-closure*:
fixes *S* :: *'n*::*euclidean-space set*
assumes *convex S*
shows *interior* (*closure S*) = *interior S*
⟨*proof*⟩

lemma *closure-eq-rel-interior-eq*:
fixes *S1 S2* :: *'n*::*euclidean-space set*
assumes *convex S1*
and *convex S2*
shows *closure S1* = *closure S2* ⟷ *rel-interior S1* = *rel-interior S2*
⟨*proof*⟩

lemma *closure-eq-between*:
fixes *S1 S2* :: *'n*::*euclidean-space set*
assumes *convex S1*
and *convex S2*
shows *closure S1* = *closure S2* ⟷ *rel-interior S1* ≤ *S2* ∧ *S2* ⊆ *closure S1*
(**is** *?A* ⟷ *?B*)
⟨*proof*⟩

lemma *open-inter-closure-rel-interior*:
fixes *S A* :: *'n*::*euclidean-space set*
assumes *convex S*
and *open A*
shows *A* ∩ *closure S* = {} ⟷ *A* ∩ *rel-interior S* = {}
⟨*proof*⟩

lemma *rel-interior-open-segment*:
fixes *a* :: *'a* :: *euclidean-space*
shows *rel-interior*(*open-segment a b*) = *open-segment a b*
⟨*proof*⟩

lemma *rel-interior-closed-segment*:
fixes *a* :: *'a* :: *euclidean-space*
shows *rel-interior*(*closed-segment a b*) =
(*if a* = *b then* {*a*} *else open-segment a b*)
⟨*proof*⟩

lemmas *rel-interior-segment* = *rel-interior-closed-segment rel-interior-open-segment*

lemma *starlike-convex-tweak-boundary-points*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *convex S S* ≠ {} **and** *ST*: *rel-interior S* ⊆ *T* **and** *TS*: *T* ⊆ *closure S*
shows *starlike T*
⟨*proof*⟩

19.35 The relative frontier of a set

definition *rel-frontier S* = *closure S* − *rel-interior S*

lemma *closed-affine-hull*:
fixes *S* :: *'n*::*euclidean-space set*
shows *closed* (*affine hull S*)
⟨*proof*⟩

lemma *closed-rel-frontier*:
fixes *S* :: *'n*::*euclidean-space set*
shows *closed* (*rel-frontier S*)
⟨*proof*⟩

lemma *convex-rel-frontier-aff-dim*:
fixes *S1 S2* :: *'n*::*euclidean-space set*
assumes *convex S1*
and *convex S2*
and *S2* ≠ {}
and *S1* ≤ *rel-frontier S2*
shows *aff-dim S1* < *aff-dim S2*
⟨*proof*⟩

lemma *convex-rel-interior-if*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
and $z \in$ *rel-interior* S
shows $\forall x \in$ *affine hull* S. $\exists m$. $m > 1 \wedge (\forall e.\ e > 1 \wedge e \leq m \longrightarrow (1 - e) *_R x + e *_R z \in S)$
⟨*proof*⟩

lemma *convex-rel-interior-if2*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
assumes $z \in$ *rel-interior* S
shows $\forall x \in$ *affine hull* S. $\exists e.\ e > 1 \wedge (1 - e) *_R x + e *_R z \in S$
⟨*proof*⟩

lemma *convex-rel-interior-only-if*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
and $S \neq \{\}$
assumes $\forall x \in S.\ \exists e.\ e > 1 \wedge (1 - e) *_R x + e *_R z \in S$
shows $z \in$ *rel-interior* S
⟨*proof*⟩

lemma *convex-rel-interior-iff*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
and $S \neq \{\}$
shows $z \in$ *rel-interior* $S \longleftrightarrow (\forall x \in S.\ \exists e.\ e > 1 \wedge (1 - e) *_R x + e *_R z \in S)$
⟨*proof*⟩

lemma *convex-rel-interior-iff2*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
and $S \neq \{\}$
shows $z \in$ *rel-interior* $S \longleftrightarrow (\forall x \in$ *affine hull* S. $\exists e.\ e > 1 \wedge (1 - e) *_R x + e *_R z \in S)$
⟨*proof*⟩

lemma *convex-interior-iff*:
fixes S :: *'n::euclidean-space set*
assumes *convex* S
shows $z \in$ *interior* $S \longleftrightarrow (\forall x.\ \exists e.\ e > 0 \wedge z + e *_R x \in S)$
⟨*proof*⟩

19.35.1 Relative interior and closure under common operations

lemma *rel-interior-inter-aux*: $\bigcap \{$*rel-interior* S $|S.\ S : I\} \subseteq \bigcap I$

⟨*proof*⟩

lemma *closure-inter*: *closure* $(\bigcap I) \leq \bigcap \{closure\ S\ |S.\ S \in I\}$
⟨*proof*⟩

lemma *convex-closure-rel-interior-inter*:
assumes $\forall S{\in}I.\ convex\ (S :: 'n{::}euclidean\text{-}space\ set)$
and $\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\} \neq \{\}$
shows $\bigcap \{closure\ S\ |S.\ S \in I\} \leq closure\ (\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\})$
⟨*proof*⟩

lemma *convex-closure-inter*:
assumes $\forall S{\in}I.\ convex\ (S :: 'n{::}euclidean\text{-}space\ set)$
and $\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\} \neq \{\}$
shows $closure\ (\bigcap I) = \bigcap \{closure\ S\ |S.\ S \in I\}$
⟨*proof*⟩

lemma *convex-inter-rel-interior-same-closure*:
assumes $\forall S{\in}I.\ convex\ (S :: 'n{::}euclidean\text{-}space\ set)$
and $\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\} \neq \{\}$
shows $closure\ (\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\}) = closure\ (\bigcap I)$
⟨*proof*⟩

lemma *convex-rel-interior-inter*:
assumes $\forall S{\in}I.\ convex\ (S :: 'n{::}euclidean\text{-}space\ set)$
and $\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\} \neq \{\}$
shows $rel\text{-}interior\ (\bigcap I) \subseteq \bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\}$
⟨*proof*⟩

lemma *convex-rel-interior-finite-inter*:
assumes $\forall S{\in}I.\ convex\ (S :: 'n{::}euclidean\text{-}space\ set)$
and $\bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\} \neq \{\}$
and *finite I*
shows $rel\text{-}interior\ (\bigcap I) = \bigcap \{rel\text{-}interior\ S\ |S.\ S \in I\}$
⟨*proof*⟩

lemma *convex-closure-inter-two*:
fixes $S\ T :: 'n{::}euclidean\text{-}space\ set$
assumes *convex S*
and *convex T*
assumes $rel\text{-}interior\ S \cap rel\text{-}interior\ T \neq \{\}$
shows $closure\ (S \cap T) = closure\ S \cap closure\ T$
⟨*proof*⟩

lemma *convex-rel-interior-inter-two*:
fixes $S\ T :: 'n{::}euclidean\text{-}space\ set$
assumes *convex S*
and *convex T*
and $rel\text{-}interior\ S \cap rel\text{-}interior\ T \neq \{\}$

shows *rel-interior* $(S \cap T) = \textit{rel-interior}\ S \cap \textit{rel-interior}\ T$
⟨*proof*⟩

lemma *convex-affine-closure-inter*:
fixes $S\ T :: 'n{::}\textit{euclidean-space set}$
assumes *convex S*
and *affine T*
and $\textit{rel-interior}\ S \cap T \neq \{\}$
shows $\textit{closure}\ (S \cap T) = \textit{closure}\ S \cap T$
⟨*proof*⟩

lemma *connected-component-1-gen*:
fixes $S :: 'a :: \textit{euclidean-space set}$
assumes $\textit{DIM}('a) = 1$
shows $\textit{connected-component}\ S\ a\ b \longleftrightarrow \textit{closed-segment}\ a\ b \subseteq S$
⟨*proof*⟩

lemma *connected-component-1*:
fixes $S :: \textit{real set}$
shows $\textit{connected-component}\ S\ a\ b \longleftrightarrow \textit{closed-segment}\ a\ b \subseteq S$
⟨*proof*⟩

lemma *convex-affine-rel-interior-inter*:
fixes $S\ T :: 'n{::}\textit{euclidean-space set}$
assumes *convex S*
and *affine T*
and $\textit{rel-interior}\ S \cap T \neq \{\}$
shows $\textit{rel-interior}\ (S \cap T) = \textit{rel-interior}\ S \cap T$
⟨*proof*⟩

lemma *subset-rel-interior-convex*:
fixes $S\ T :: 'n{::}\textit{euclidean-space set}$
assumes *convex S*
and *convex T*
and $S \leq \textit{closure}\ T$
and $\neg\ S \subseteq \textit{rel-frontier}\ T$
shows $\textit{rel-interior}\ S \subseteq \textit{rel-interior}\ T$
⟨*proof*⟩

lemma *rel-interior-convex-linear-image*:
fixes $f :: 'm{::}\textit{euclidean-space} \Rightarrow 'n{::}\textit{euclidean-space}$
assumes *linear f*
and *convex S*
shows $f\ `\ (\textit{rel-interior}\ S) = \textit{rel-interior}\ (f\ `\ S)$
⟨*proof*⟩

lemma *rel-interior-convex-linear-preimage*:
fixes $f :: 'm{::}\textit{euclidean-space} \Rightarrow 'n{::}\textit{euclidean-space}$
assumes *linear f*

and *convex S*
and $f -` (\textit{rel-interior}\ S) \neq \{\}$
shows $\textit{rel-interior}\ (f -` S) = f -` (\textit{rel-interior}\ S)$
⟨*proof*⟩

lemma *rel-interior-direct-sum*:
fixes $S :: 'n{::}\textit{euclidean-space set}$
and $T :: 'm{::}\textit{euclidean-space set}$
assumes *convex S*
and *convex T*
shows $\textit{rel-interior}\ (S \times T) = \textit{rel-interior}\ S \times \textit{rel-interior}\ T$
⟨*proof*⟩

lemma *rel-interior-scaleR*:
fixes $S :: 'n{::}\textit{euclidean-space set}$
assumes $c \neq 0$
shows $(op *_R\ c)\ `\ (\textit{rel-interior}\ S) = \textit{rel-interior}\ ((op *_R\ c)\ `\ S)$
⟨*proof*⟩

lemma *rel-interior-convex-scaleR*:
fixes $S :: 'n{::}\textit{euclidean-space set}$
assumes *convex S*
shows $(op *_R\ c)\ `\ (\textit{rel-interior}\ S) = \textit{rel-interior}\ ((op *_R\ c)\ `\ S)$
⟨*proof*⟩

lemma *convex-rel-open-scaleR*:
fixes $S :: 'n{::}\textit{euclidean-space set}$
assumes *convex S*
and *rel-open S*
shows $\textit{convex}\ ((op *_R\ c)\ `\ S) \wedge \textit{rel-open}\ ((op *_R\ c)\ `\ S)$
⟨*proof*⟩

lemma *convex-rel-open-finite-inter*:
assumes $\forall S{\in}I.\ \textit{convex}\ (S :: 'n{::}\textit{euclidean-space set}) \wedge \textit{rel-open}\ S$
and *finite I*
shows $\textit{convex}\ (\bigcap I) \wedge \textit{rel-open}\ (\bigcap I)$
⟨*proof*⟩

lemma *convex-rel-open-linear-image*:
fixes $f :: 'm{::}\textit{euclidean-space} \Rightarrow 'n{::}\textit{euclidean-space}$
assumes *linear f*
and *convex S*
and *rel-open S*
shows $\textit{convex}\ (f\ `\ S) \wedge \textit{rel-open}\ (f\ `\ S)$
⟨*proof*⟩

lemma *convex-rel-open-linear-preimage*:
fixes $f :: 'm{::}\textit{euclidean-space} \Rightarrow 'n{::}\textit{euclidean-space}$
assumes *linear f*

and *convex S*
and *rel-open S*
shows *convex* $(f\ -`\ S) \wedge$ *rel-open* $(f\ -`\ S)$
⟨*proof*⟩

lemma *rel-interior-projection*:
fixes S :: $('m$::*euclidean-space* $\times$ $'n$::*euclidean-space*$)$ *set*
and f :: $'m$::*euclidean-space* $\Rightarrow$ $'n$::*euclidean-space set*
assumes *convex S*
and $f = (\lambda y.\ \{z.\ (y,\ z) \in S\})$
shows $(y,\ z) \in$ *rel-interior S* $\longleftrightarrow$ $(y \in$ *rel-interior* $\{y.\ (f\ y \neq \{\})\} \wedge z \in$ *rel-interior* $(f\ y))$
⟨*proof*⟩

19.35.2 Relative interior of convex cone

lemma *cone-rel-interior*:
fixes S :: $'m$::*euclidean-space set*
assumes *cone S*
shows *cone* $(\{0\} \cup$ *rel-interior S*$)$
⟨*proof*⟩

lemma *rel-interior-convex-cone-aux*:
fixes S :: $'m$::*euclidean-space set*
assumes *convex S*
shows $(c,\ x) \in$ *rel-interior* $($*cone hull* $(\{(1$ :: *real*$)\} \times S)) \longleftrightarrow$
$c > 0 \wedge x \in ((op\ *_R\ c)\ `$ *(rel-interior S)*$)$
⟨*proof*⟩

lemma *rel-interior-convex-cone*:
fixes S :: $'m$::*euclidean-space set*
assumes *convex S*
shows *rel-interior* $($*cone hull* $(\{1$ :: *real*$\} \times S)) =$
$\{(c,\ c\ *_R\ x) \mid c\ x.\ c > 0 \wedge x \in$ *rel-interior S*$\}$
(is *?lhs = ?rhs*$)$
⟨*proof*⟩

lemma *convex-hull-finite-union*:
assumes *finite I*
assumes $\forall i{\in}I.$ *convex* $(S\ i) \wedge (S\ i) \neq \{\}$
shows *convex hull* $(\bigcup (S\ `\ I)) =$
$\{$*setsum* $(\lambda i.\ c\ i\ *_R\ s\ i)\ I \mid c\ s.\ (\forall i{\in}I.\ c\ i \geq 0) \wedge$ *setsum c I* $= 1 \wedge (\forall i{\in}I.\ s\ i \in S\ i)\}$
(is *?lhs = ?rhs*$)$
⟨*proof*⟩

lemma *convex-hull-union-two*:
fixes $S\ T$ :: $'m$::*euclidean-space set*
assumes *convex S*

and $S \neq \{\}$
and *convex* T
and $T \neq \{\}$
shows *convex hull* $(S \cup T) =$
$\{u *_R s + v *_R t \mid u\ v\ s\ t.\ u \geq 0 \wedge v \geq 0 \wedge u + v = 1 \wedge s \in S \wedge t \in T\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

19.36 Convexity on direct sums

lemma *closure-sum*:
fixes $S\ T :: 'a$::*real-normed-vector set*
shows *closure* S + *closure* $T \subseteq$ *closure* $(S + T)$
⟨*proof*⟩

lemma *rel-interior-sum*:
fixes $S\ T :: 'n$::*euclidean-space set*
assumes *convex* S
and *convex* T
shows *rel-interior* $(S + T)$ = *rel-interior* S + *rel-interior* T
⟨*proof*⟩

lemma *rel-interior-sum-gen*:
fixes $S :: 'a \Rightarrow 'n$::*euclidean-space set*
assumes $\forall i \in I.$ *convex* $(S\ i)$
shows *rel-interior* $(setsum\ S\ I)$ = $setsum\ (\lambda i.$ *rel-interior* $(S\ i))\ I$
⟨*proof*⟩

lemma *convex-rel-open-direct-sum*:
fixes $S\ T :: 'n$::*euclidean-space set*
assumes *convex* S
and *rel-open* S
and *convex* T
and *rel-open* T
shows *convex* $(S \times T) \wedge$ *rel-open* $(S \times T)$
⟨*proof*⟩

lemma *convex-rel-open-sum*:
fixes $S\ T :: 'n$::*euclidean-space set*
assumes *convex* S
and *rel-open* S
and *convex* T
and *rel-open* T
shows *convex* $(S + T) \wedge$ *rel-open* $(S + T)$
⟨*proof*⟩

lemma *convex-hull-finite-union-cones*:
assumes *finite* I
and $I \neq \{\}$

assumes $\forall\, i{\in}I.\ convex\ (S\ i) \wedge cone\ (S\ i) \wedge S\ i \neq \{\}$
shows $convex\ hull\ (\bigcup (S\ `\ I)) = setsum\ S\ I$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *convex-hull-union-cones-two*:
fixes $S\ T :: 'm{::}euclidean\text{-}space\ set$
assumes *convex S*
and *cone S*
and $S \neq \{\}$
assumes *convex T*
and *cone T*
and $T \neq \{\}$
shows $convex\ hull\ (S \cup T) = S + T$
⟨*proof*⟩

lemma *rel-interior-convex-hull-union*:
fixes $S :: 'a \Rightarrow 'n{::}euclidean\text{-}space\ set$
assumes *finite I*
and $\forall\, i{\in}I.\ convex\ (S\ i) \wedge S\ i \neq \{\}$
shows $rel\text{-}interior\ (convex\ hull\ (\bigcup (S\ `\ I))) =$
$\{setsum\ (\lambda i.\ c\ i *_R s\ i)\ I \mid c\ s.\ (\forall\, i{\in}I.\ c\ i > 0) \wedge setsum\ c\ I = 1 \wedge$
$(\forall\, i{\in}I.\ s\ i \in rel\text{-}interior(S\ i))\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *convex-le-Inf-differential*:
fixes $f :: real \Rightarrow real$
assumes *convex-on I f*
and $x \in interior\ I$
and $y \in I$
shows $f\ y \geq f\ x + Inf\ ((\lambda t.\ (f\ x - f\ t)\ /\ (x - t))\ `\ (\{x{<}..\} \cap I)) * (y - x)$
(**is** - ≥ - + *Inf* (*?F x*) ∗ (*y* − *x*))
⟨*proof*⟩

19.37 Explicit formulas for interior and relative interior of convex hull

lemma *interior-atLeastAtMost* [*simp*]:
fixes *a*::*real* **shows** $interior\ \{a..b\} = \{a{<}..{<}b\}$
⟨*proof*⟩

lemma *interior-atLeastLessThan* [*simp*]:
fixes *a*::*real* **shows** $interior\ \{a..{<}b\} = \{a{<}..{<}b\}$
⟨*proof*⟩

lemma *interior-lessThanAtMost* [*simp*]:
fixes *a*::*real* **shows** $interior\ \{a{<}..b\} = \{a{<}..{<}b\}$

⟨*proof*⟩

lemma *at-within-closed-interval*:
fixes *x*::*real*
shows $a < x \Longrightarrow x < b \Longrightarrow (at\ x\ within\ \{a..b\}) = at\ x$
⟨*proof*⟩

lemma *affine-independent-convex-affine-hull*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~*affine-dependent s t* ⊆ *s*
shows *convex hull t* = *affine hull t* ∩ *convex hull s*
⟨*proof*⟩

lemma *affine-independent-span-eq*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~*affine-dependent s card s* = *Suc* (*DIM* (*'a*))
shows *affine hull s* = *UNIV*
⟨*proof*⟩

lemma *affine-independent-span-gt*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *ind*: ~ *affine-dependent s* **and** *dim*: *DIM* (*'a*) < *card s*
shows *affine hull s* = *UNIV*
⟨*proof*⟩

lemma *empty-interior-affine-hull*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *finite s* **and** *dim*: *card s* ≤ *DIM* (*'a*)
shows *interior*(*affine hull s*) = {}
⟨*proof*⟩

lemma *empty-interior-convex-hull*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *finite s* **and** *dim*: *card s* ≤ *DIM* (*'a*)
shows *interior*(*convex hull s*) = {}
⟨*proof*⟩

lemma *explicit-subset-rel-interior-convex-hull*:
fixes *s* :: *'a*::*euclidean-space set*
shows *finite s*
$\Longrightarrow \{y.\ \exists u.\ (\forall x \in s.\ 0 < u\ x \wedge u\ x < 1) \wedge setsum\ u\ s = 1 \wedge setsum\ (\lambda x.\ u\ x *_R x)\ s = y\}$
⊆ *rel-interior* (*convex hull s*)
⟨*proof*⟩

lemma *explicit-subset-rel-interior-convex-hull-minimal*:
fixes *s* :: *'a*::*euclidean-space set*
shows *finite s*
$\Longrightarrow \{y.\ \exists u.\ (\forall x \in s.\ 0 < u\ x) \wedge setsum\ u\ s = 1 \wedge setsum\ (\lambda x.\ u\ x *_R$

$x)\ s = y\}$
$\subseteq$ *rel-interior* (*convex hull s*)
⟨*proof*⟩

lemma *rel-interior-convex-hull-explicit*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~ *affine-dependent s*
shows *rel-interior*(*convex hull s*) =
$\{y.\ \exists u.\ (\forall x \in s.\ 0 < u\ x) \wedge$ *setsum u s* $= 1 \wedge$ *setsum* $(\lambda x.\ u\ x *_R x)\ s = y\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *interior-convex-hull-explicit-minimal*:
fixes *s* :: *'a*::*euclidean-space set*
shows
~ *affine-dependent s*
==> *interior*(*convex hull s*) =
(*if card*(*s*) $\leq$ *DIM*(*'a*) *then* {}
else $\{y.\ \exists u.\ (\forall x \in s.\ 0 < u\ x) \wedge$ *setsum u s* $= 1 \wedge (\sum x \in s.\ u\ x *_R x) = y\}$)
⟨*proof*⟩

lemma *interior-convex-hull-explicit*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~ *affine-dependent s*
shows
interior(*convex hull s*) =
(*if card*(*s*) $\leq$ *DIM*(*'a*) *then* {}
else $\{y.\ \exists u.\ (\forall x \in s.\ 0 < u\ x \wedge u\ x < 1) \wedge$ *setsum u s* $= 1 \wedge (\sum x \in s.\ u\ x *_R x) = y\}$)
⟨*proof*⟩

lemma *interior-closed-segment-ge2*:
fixes *a* :: *'a*::*euclidean-space*
assumes *2* $\leq$ *DIM*(*'a*)
shows *interior*(*closed-segment a b*) = {}
⟨*proof*⟩

lemma *interior-open-segment*:
fixes *a* :: *'a*::*euclidean-space*
shows *interior*(*open-segment a b*) =
(*if 2* $\leq$ *DIM*(*'a*) *then* {} *else open-segment a b*)
⟨*proof*⟩

lemma *interior-closed-segment*:
fixes *a* :: *'a*::*euclidean-space*
shows *interior*(*closed-segment a b*) =
(*if 2* $\leq$ *DIM*(*'a*) *then* {} *else open-segment a b*)

⟨*proof*⟩

lemmas *interior-segment* = *interior-closed-segment interior-open-segment*

lemma *closed-segment-eq* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *closed-segment a b* = *closed-segment c d* ⟷ {*a*,*b*} = {*c*,*d*}
⟨*proof*⟩

lemma *closed-open-segment-eq* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *closed-segment a b* ≠ *open-segment c d*
⟨*proof*⟩

lemma *open-closed-segment-eq* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *open-segment a b* ≠ *closed-segment c d*
⟨*proof*⟩

lemma *open-segment-eq* [*simp*]:
fixes *a* :: *'a*::*euclidean-space*
shows *open-segment a b* = *open-segment c d* ⟷ *a* = *b* ∧ *c* = *d* ∨ {*a*,*b*} = {*c*,*d*}
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

19.38 Similar results for closure and (relative or absolute) frontier.

lemma *closure-convex-hull* [*simp*]:
fixes *s* :: *'a*::*euclidean-space set*
shows *compact s* ==> *closure*(*convex hull s*) = *convex hull s*
⟨*proof*⟩

lemma *rel-frontier-convex-hull-explicit*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~ *affine-dependent s*
shows *rel-frontier*(*convex hull s*) =
{*y*. ∃ *u*. (∀ *x* ∈ *s*. 0 ≤ *u x*) ∧ (∃ *x* ∈ *s*. *u x* = 0) ∧ *setsum u s* = 1 ∧ *setsum* (λ*x*. *u x* ∗*R x*) *s* = *y*}
⟨*proof*⟩

lemma *frontier-convex-hull-explicit*:
fixes *s* :: *'a*::*euclidean-space set*
assumes ~ *affine-dependent s*
shows *frontier*(*convex hull s*) =
{*y*. ∃ *u*. (∀ *x* ∈ *s*. 0 ≤ *u x*) ∧ (*DIM* (*'a*) < *card s* ⟶ (∃ *x* ∈ *s*. *u x* = 0)) ∧
setsum u s = 1 ∧ *setsum* (λ*x*. *u x* ∗*R x*) *s* = *y*}

⟨*proof*⟩

lemma *rel-frontier-convex-hull-cases*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes ~ *affine-dependent s*
shows *rel-frontier*(*convex hull s*) = $\bigcup\{convex\ hull\ (s - \{x\})\ |x.\ x \in s\}$
⟨*proof*⟩

lemma *frontier-convex-hull-eq-rel-frontier*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes ~ *affine-dependent s*
shows *frontier*(*convex hull s*) =
(*if card s* ≤ *DIM* ('*a*) *then convex hull s else rel-frontier*(*convex hull s*))
⟨*proof*⟩

lemma *frontier-convex-hull-cases*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes ~ *affine-dependent s*
shows *frontier*(*convex hull s*) =
(*if card s* ≤ *DIM* ('*a*) *then convex hull s else* $\bigcup\{convex\ hull\ (s - \{x\})$
$|x.\ x \in s\}$)
⟨*proof*⟩

lemma *in-frontier-convex-hull*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *finite s card s* ≤ *Suc* (*DIM* ('*a*)) $x \in s$
shows $x \in frontier(convex\ hull\ s)$
⟨*proof*⟩

lemma *not-in-interior-convex-hull*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *finite s card s* ≤ *Suc* (*DIM* ('*a*)) $x \in s$
shows $x \notin interior(convex\ hull\ s)$
⟨*proof*⟩

lemma *interior-convex-hull-eq-empty*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *card s* = *Suc* (*DIM* ('*a*))
shows $interior(convex\ hull\ s) = \{\} \longleftrightarrow$ *affine-dependent s*
⟨*proof*⟩

19.39 Coplanarity, and collinearity in terms of affine hull

definition *coplanar* **where**
$coplanar\ s \equiv \exists u\ v\ w.\ s \subseteq affine\ hull\ \{u,v,w\}$

lemma *collinear-affine-hull*:
$collinear\ s \longleftrightarrow (\exists u\ v.\ s \subseteq affine\ hull\ \{u,v\})$
⟨*proof*⟩

lemma *collinear-closed-segment* [*simp*]: *collinear* (*closed-segment a b*)
⟨*proof*⟩

lemma *collinear-open-segment* [*simp*]: *collinear* (*open-segment a b*)
⟨*proof*⟩

lemma *subset-continuous-image-segment-1*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *real*
assumes *continuous-on* (*closed-segment a b*) *f*
shows *closed-segment* (*f a*) (*f b*) ⊆ *image f* (*closed-segment a b*)
⟨*proof*⟩

lemma *collinear-imp-coplanar*:
collinear s ==> *coplanar s*
⟨*proof*⟩

lemma *collinear-small*:
assumes *finite s card s* ≤ *2*
shows *collinear s*
⟨*proof*⟩

lemma *coplanar-small*:
assumes *finite s card s* ≤ *3*
shows *coplanar s*
⟨*proof*⟩

lemma *coplanar-empty*: *coplanar* {}
⟨*proof*⟩

lemma *coplanar-sing*: *coplanar* {*a*}
⟨*proof*⟩

lemma *coplanar-2*: *coplanar* {*a*,*b*}
⟨*proof*⟩

lemma *coplanar-3*: *coplanar* {*a*,*b*,*c*}
⟨*proof*⟩

lemma *collinear-affine-hull-collinear*: *collinear*(*affine hull s*) ⟷ *collinear s*
⟨*proof*⟩

lemma *coplanar-affine-hull-coplanar*: *coplanar*(*affine hull s*) ⟷ *coplanar s*
⟨*proof*⟩

lemma *coplanar-linear-image*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*real-normed-vector*
assumes *coplanar s linear f* **shows** *coplanar*(*f* ‘ *s*)
⟨*proof*⟩

lemma *coplanar-translation-imp*: *coplanar s* $\Longrightarrow$ *coplanar* $((\lambda x.\ a + x)\ `\ s)$
⟨*proof*⟩

lemma *coplanar-translation-eq*: *coplanar*$((\lambda x.\ a + x)\ `\ s) \longleftrightarrow$ *coplanar s*
⟨*proof*⟩

lemma *coplanar-linear-image-eq*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'b*::*euclidean-space*
assumes *linear f inj f* **shows** *coplanar*$(f\ `\ s)$ = *coplanar s*
⟨*proof*⟩

lemma *coplanar-subset*: $[\![$*coplanar t*; *s* $\subseteq$ *t*$]\!] \Longrightarrow$ *coplanar s*
⟨*proof*⟩

lemma *affine-hull-3-imp-collinear*: *c* $\in$ *affine hull* {*a*,*b*} $\Longrightarrow$ *collinear* {*a*,*b*,*c*}
⟨*proof*⟩

lemma *collinear-3-imp-in-affine-hull*: $[\![$*collinear* {*a*,*b*,*c*}; *a* $\neq$ *b*$]\!] \Longrightarrow$ *c* $\in$ *affine hull* {*a*,*b*}
⟨*proof*⟩

lemma *collinear-3-affine-hull*:
assumes *a* $\neq$ *b*
shows *collinear* {*a*,*b*,*c*} $\longleftrightarrow$ *c* $\in$ *affine hull* {*a*,*b*}
⟨*proof*⟩

lemma *collinear-3-eq-affine-dependent*:
collinear{*a*,*b*,*c*} $\longleftrightarrow$ *a* = *b* $\vee$ *a* = *c* $\vee$ *b* = *c* $\vee$ *affine-dependent* {*a*,*b*,*c*}
⟨*proof*⟩

lemma *affine-dependent-imp-collinear-3*:
affine-dependent {*a*,*b*,*c*} $\Longrightarrow$ *collinear*{*a*,*b*,*c*}
⟨*proof*⟩

lemma *collinear-3*: *NO-MATCH 0 x* $\Longrightarrow$ *collinear* {*x*,*y*,*z*} $\longleftrightarrow$ *collinear* {*0*, *x*−*y*, *z*−*y*}
⟨*proof*⟩

thm *affine-hull-nonempty*
corollary *affine-hull-eq-empty* [*simp*]: *affine hull S* = {} $\longleftrightarrow$ *S* = {}
⟨*proof*⟩

lemma *affine-hull-2-alt*:
fixes *a b* :: *'a*::*real-vector*
shows *affine hull* {*a*,*b*} = *range* $(\lambda u.\ a + u *_R (b - a))$
⟨*proof*⟩

lemma *interior-convex-hull-3-minimal*:
fixes a :: *'a*::*euclidean-space*
shows $[\![\sim$ *collinear*$\{a,b,c\}$; *DIM*$('a) = 2]\!]$
$\Longrightarrow$ *interior*(*convex hull* $\{a,b,c\}) =$
$\{v.\ \exists x\ y\ z.\ 0 < x \wedge 0 < y \wedge 0 < z \wedge x + y + z = 1 \wedge$
$x *_R a + y *_R b + z *_R c = v\}$
⟨*proof*⟩

19.40 The infimum of the distance between two sets

definition *setdist* :: *'a*::*metric-space set* $\Rightarrow$ *'a set* $\Rightarrow$ *real* **where**
setdist s t $\equiv$
(*if* $s = \{\} \vee t = \{\}$ *then 0*
else Inf $\{$*dist x y*$|\ x\ y.\ x \in s \wedge y \in t\}$)

lemma *setdist-empty1* [*simp*]: *setdist* $\{\}$ *t* = *0*
⟨*proof*⟩

lemma *setdist-empty2* [*simp*]: *setdist t* $\{\}$ = *0*
⟨*proof*⟩

lemma *setdist-pos-le*: *0* $\leq$ *setdist s t*
⟨*proof*⟩

lemma *le-setdistI*:
assumes $s \neq \{\}$ $t \neq \{\}$ $\bigwedge x\ y.\ [\![x \in s;\ y \in t]\!] \Longrightarrow d \leq$ *dist x y*
shows $d \leq$ *setdist s t*
⟨*proof*⟩

lemma *setdist-le-dist*: $[\![x \in s;\ y \in t]\!] \Longrightarrow$ *setdist s t* $\leq$ *dist x y*
⟨*proof*⟩

lemma *le-setdist-iff*:
$d \leq$ *setdist s t* $\longleftrightarrow$
$(\forall x \in s.\ \forall y \in t.\ d \leq$ *dist x y*$) \wedge (s = \{\} \vee t = \{\} \longrightarrow d \leq 0)$
⟨*proof*⟩

lemma *setdist-ltE*:
assumes *setdist s t* $< b$ $s \neq \{\}$ $t \neq \{\}$
obtains *x y* **where** $x \in s$ $y \in t$ *dist x y* $< b$
⟨*proof*⟩

lemma *setdist-refl*: *setdist s s* = *0*
⟨*proof*⟩

lemma *setdist-sym*: *setdist s t* = *setdist t s*
⟨*proof*⟩

lemma *setdist-triangle*: *setdist s t* ≤ *setdist s* {*a*} + *setdist* {*a*} *t*
⟨*proof*⟩

lemma *setdist-singletons* [*simp*]: *setdist* {*x*} {*y*} = *dist x y*
⟨*proof*⟩

lemma *setdist-Lipschitz*: |*setdist* {*x*} *s* − *setdist* {*y*} *s*| ≤ *dist x y*
⟨*proof*⟩

lemma *continuous-at-setdist*: *continuous* (*at x*) (λ*y*. (*setdist* {*y*} *s*))
⟨*proof*⟩

lemma *continuous-on-setdist*: *continuous-on t* (λ*y*. (*setdist* {*y*} *s*))
⟨*proof*⟩

lemma *uniformly-continuous-on-setdist*: *uniformly-continuous-on t* (λ*y*. (*setdist* {*y*} *s*))
⟨*proof*⟩

lemma *setdist-subset-right*: ⟦*t* ≠ {}; *t* ⊆ *u*⟧ ⟹ *setdist s u* ≤ *setdist s t*
⟨*proof*⟩

lemma *setdist-subset-left*: ⟦*s* ≠ {}; *s* ⊆ *t*⟧ ⟹ *setdist t u* ≤ *setdist s u*
⟨*proof*⟩

lemma *setdist-closure-1* [*simp*]: *setdist* (*closure s*) *t* = *setdist s t*
⟨*proof*⟩

lemma *setdist-closure-2* [*simp*]: *setdist t* (*closure s*) = *setdist t s*
⟨*proof*⟩

lemma *setdist-compact-closed*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *s*: *compact s* **and** *t*: *closed t*
and *s* ≠ {} *t* ≠ {}
shows ∃*x* ∈ *s*. ∃*y* ∈ *t*. *dist x y* = *setdist s t*
⟨*proof*⟩

lemma *setdist-closed-compact*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *s*: *closed s* **and** *t*: *compact t*
and *s* ≠ {} *t* ≠ {}
shows ∃*x* ∈ *s*. ∃*y* ∈ *t*. *dist x y* = *setdist s t*
⟨*proof*⟩

lemma *setdist-eq-0I*: ⟦*x* ∈ *s*; *x* ∈ *t*⟧ ⟹ *setdist s t* = *0*
⟨*proof*⟩

lemma *setdist-eq-0-compact-closed*:

fixes s :: $'a$::*euclidean-space set*
assumes s: *compact s* **and** t: *closed t*
shows *setdist s t* = *0* $\longleftrightarrow$ $s = \{\} \vee t = \{\} \vee s \cap t \neq \{\}$
⟨*proof*⟩

corollary *setdist-gt-0-compact-closed*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *compact s* **and** t: *closed t*
shows *setdist s t* > *0* $\longleftrightarrow$ $(s \neq \{\} \wedge t \neq \{\} \wedge s \cap t = \{\})$
⟨*proof*⟩

lemma *setdist-eq-0-closed-compact*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *closed s* **and** t: *compact t*
shows *setdist s t* = *0* $\longleftrightarrow$ $s = \{\} \vee t = \{\} \vee s \cap t \neq \{\}$
⟨*proof*⟩

lemma *setdist-eq-0-bounded*:
fixes s :: $'a$::*euclidean-space set*
assumes *bounded s* $\vee$ *bounded t*
shows *setdist s t* = *0* $\longleftrightarrow$ $s = \{\} \vee t = \{\} \vee$ *closure s* $\cap$ *closure t* $\neq \{\}$
⟨*proof*⟩

lemma *setdist-unique*:
$[\![a \in s;\ b \in t;\ \bigwedge x\ y.\ x \in s \wedge y \in t$ ==> *dist a b* $\leq$ *dist x y*$]\!]$
$\Longrightarrow$ *setdist s t* = *dist a b*
⟨*proof*⟩

lemma *setdist-closest-point*:
$[\![$*closed s*; $s \neq \{\}]\!] \Longrightarrow$ *setdist* $\{a\}$ *s* = *dist a* (*closest-point s a*)
⟨*proof*⟩

lemma *setdist-eq-0-sing-1* [*simp*]:
fixes s :: $'a$::*euclidean-space set*
shows *setdist* $\{x\}$ *s* = *0* $\longleftrightarrow$ $s = \{\} \vee x \in$ *closure s*
⟨*proof*⟩

lemma *setdist-eq-0-sing-2* [*simp*]:
fixes s :: $'a$::*euclidean-space set*
shows *setdist s* $\{x\}$ = *0* $\longleftrightarrow$ $s = \{\} \vee x \in$ *closure s*
⟨*proof*⟩

lemma *setdist-sing-in-set*:
fixes s :: $'a$::*euclidean-space set*
shows $x \in s \Longrightarrow$ *setdist* $\{x\}$ *s* = *0*
⟨*proof*⟩

lemma *setdist-le-sing*: $x \in s$ ==> *setdist s t* $\leq$ *setdist* $\{x\}$ *t*
⟨*proof*⟩

end

20 Continuous paths and path-connected sets

theory *Path-Connected*
imports *Convex-Euclidean-Space*
begin

20.1 Paths and Arcs

definition *path* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *bool*
  where *path g* ⟷ *continuous-on* {*0..1*} *g*

definition *pathstart* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *'a*
  where *pathstart g* = *g 0*

definition *pathfinish* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *'a*
  where *pathfinish g* = *g 1*

definition *path-image* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *'a set*
  where *path-image g* = *g* ' {*0* .. *1*}

definition *reversepath* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *real* ⇒ *'a*
  where *reversepath g* = (λ*x*. *g*(*1* − *x*))

definition *joinpaths* :: (*real* ⇒ *'a*::*topological-space*) ⇒ (*real* ⇒ *'a*) ⇒ *real* ⇒ *'a*
    (**infixr** +++ *75*)
  where *g1* +++ *g2* = (λ*x*. *if x* ≤ *1/2 then g1* (*2* ∗ *x*) *else g2* (*2* ∗ *x* − *1*))

definition *simple-path* :: (*real* ⇒ *'a*::*topological-space*) ⇒ *bool*
  where *simple-path g* ⟷
    path g ∧ (∀ *x*∈{*0..1*}. ∀ *y*∈{*0..1*}. *g x* = *g y* ⟶ *x* = *y* ∨ *x* = *0* ∧ *y* = *1* ∨ *x* = *1* ∧ *y* = *0*)

definition *arc* :: (*real* ⇒ *'a* :: *topological-space*) ⇒ *bool*
  where *arc g* ⟷ *path g* ∧ *inj-on g* {*0..1*}

20.2 Invariance theorems

lemma *path-eq*: *path p* ⟹ (⋀*t*. *t* ∈ {*0..1*} ⟹ *p t* = *q t*) ⟹ *path q*
  ⟨*proof*⟩

lemma *path-continuous-image*: *path g* ⟹ *continuous-on* (*path-image g*) *f* ⟹ *path*(*f o g*)
  ⟨*proof*⟩

lemma *path-translation-eq*:
  fixes *g* :: *real* ⇒ *'a* :: *real-normed-vector*

shows *path*((λ*x*. *a* + *x*) *o* *g*) = *path* *g*
⟨*proof*⟩

lemma *path-linear-image-eq*:
fixes *f* :: 'a::*euclidean-space* ⇒ 'b::*euclidean-space*
assumes *linear* *f* *inj* *f*
shows *path*(*f* *o* *g*) = *path* *g*
⟨*proof*⟩

lemma *pathstart-translation*: *pathstart*((λ*x*. *a* + *x*) *o* *g*) = *a* + *pathstart* *g*
⟨*proof*⟩

lemma *pathstart-linear-image-eq*: *linear* *f* ⟹ *pathstart*(*f* *o* *g*) = *f*(*pathstart* *g*)
⟨*proof*⟩

lemma *pathfinish-translation*: *pathfinish*((λ*x*. *a* + *x*) *o* *g*) = *a* + *pathfinish* *g*
⟨*proof*⟩

lemma *pathfinish-linear-image*: *linear* *f* ⟹ *pathfinish*(*f* *o* *g*) = *f*(*pathfinish* *g*)
⟨*proof*⟩

lemma *path-image-translation*: *path-image*((λ*x*. *a* + *x*) *o* *g*) = (λ*x*. *a* + *x*) ‘ (*path-image* *g*)
⟨*proof*⟩

lemma *path-image-linear-image*: *linear* *f* ⟹ *path-image*(*f* *o* *g*) = *f* ‘ (*path-image* *g*)
⟨*proof*⟩

lemma *reversepath-translation*: *reversepath*((λ*x*. *a* + *x*) *o* *g*) = (λ*x*. *a* + *x*) *o* *reversepath* *g*
⟨*proof*⟩

lemma *reversepath-linear-image*: *linear* *f* ⟹ *reversepath*(*f* *o* *g*) = *f* *o* *reversepath* *g*
⟨*proof*⟩

lemma *joinpaths-translation*:
((λ*x*. *a* + *x*) *o* *g1*) +++ ((λ*x*. *a* + *x*) *o* *g2*) = (λ*x*. *a* + *x*) *o* (*g1* +++ *g2*)
⟨*proof*⟩

lemma *joinpaths-linear-image*: *linear* *f* ⟹ (*f* *o* *g1*) +++ (*f* *o* *g2*) = *f* *o* (*g1* +++ *g2*)
⟨*proof*⟩

lemma *simple-path-translation-eq*:
fixes *g* :: *real* ⇒ 'a::*euclidean-space*
shows *simple-path*((λ*x*. *a* + *x*) *o* *g*) = *simple-path* *g*
⟨*proof*⟩

lemma *simple-path-linear-image-eq*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes *linear f inj f*
shows $simple\text{-}path(f\ o\ g) = simple\text{-}path\ g$
⟨*proof*⟩

lemma *arc-translation-eq*:
fixes $g :: real \Rightarrow 'a::euclidean\text{-}space$
shows $arc((\lambda x.\ a + x)\ o\ g) = arc\ g$
⟨*proof*⟩

lemma *arc-linear-image-eq*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes *linear f inj f*
shows $arc(f\ o\ g) = arc\ g$
⟨*proof*⟩

20.3 Basic lemmas about paths

lemma *arc-imp-simple-path*: $arc\ g \Longrightarrow simple\text{-}path\ g$
⟨*proof*⟩

lemma *arc-imp-path*: $arc\ g \Longrightarrow path\ g$
⟨*proof*⟩

lemma *simple-path-imp-path*: $simple\text{-}path\ g \Longrightarrow path\ g$
⟨*proof*⟩

lemma *simple-path-cases*: $simple\text{-}path\ g \Longrightarrow arc\ g \vee pathfinish\ g = pathstart\ g$
⟨*proof*⟩

lemma *simple-path-imp-arc*: $simple\text{-}path\ g \Longrightarrow pathfinish\ g \neq pathstart\ g \Longrightarrow arc\ g$
⟨*proof*⟩

lemma *arc-distinct-ends*: $arc\ g \Longrightarrow pathfinish\ g \neq pathstart\ g$
⟨*proof*⟩

lemma *arc-simple-path*: $arc\ g \longleftrightarrow simple\text{-}path\ g \wedge pathfinish\ g \neq pathstart\ g$
⟨*proof*⟩

lemma *simple-path-eq-arc*: $pathfinish\ g \neq pathstart\ g \Longrightarrow (simple\text{-}path\ g = arc\ g)$
⟨*proof*⟩

lemma *path-image-nonempty* [*simp*]: $path\text{-}image\ g \neq \{\}$
⟨*proof*⟩

lemma *pathstart-in-path-image*[*intro*]: $pathstart\ g \in path\text{-}image\ g$

⟨*proof*⟩

lemma *pathfinish-in-path-image*[*intro*]: *pathfinish g* ∈ *path-image g*
⟨*proof*⟩

lemma *connected-path-image*[*intro*]: *path g* ⟹ *connected* (*path-image g*)
⟨*proof*⟩

lemma *compact-path-image*[*intro*]: *path g* ⟹ *compact* (*path-image g*)
⟨*proof*⟩

lemma *reversepath-reversepath*[*simp*]: *reversepath* (*reversepath g*) = *g*
⟨*proof*⟩

lemma *pathstart-reversepath*[*simp*]: *pathstart* (*reversepath g*) = *pathfinish g*
⟨*proof*⟩

lemma *pathfinish-reversepath*[*simp*]: *pathfinish* (*reversepath g*) = *pathstart g*
⟨*proof*⟩

lemma *pathstart-join*[*simp*]: *pathstart* (*g1* +++ *g2*) = *pathstart g1*
⟨*proof*⟩

lemma *pathfinish-join*[*simp*]: *pathfinish* (*g1* +++ *g2*) = *pathfinish g2*
⟨*proof*⟩

lemma *path-image-reversepath*[*simp*]: *path-image* (*reversepath g*) = *path-image g*
⟨*proof*⟩

lemma *path-reversepath* [*simp*]: *path* (*reversepath g*) ⟷ *path g*
⟨*proof*⟩

lemma *arc-reversepath*:
assumes *arc g* **shows** *arc*(*reversepath g*)
⟨*proof*⟩

lemma *simple-path-reversepath*: *simple-path g* ⟹ *simple-path* (*reversepath g*)
⟨*proof*⟩

lemmas *reversepath-simps* =
path-reversepath path-image-reversepath pathstart-reversepath pathfinish-reversepath

lemma *path-join*[*simp*]:
assumes *pathfinish g1* = *pathstart g2*
shows *path* (*g1* +++ *g2*) ⟷ *path g1* ∧ *path g2*
⟨*proof*⟩

21 Path Images

lemma *bounded-path-image*: *path g* $\Longrightarrow$ *bounded*(*path-image g*)
⟨*proof*⟩

lemma *closed-path-image*:
fixes *g* :: *real* $\Rightarrow$ *'a*::*t2-space*
shows *path g* $\Longrightarrow$ *closed*(*path-image g*)
⟨*proof*⟩

lemma *connected-simple-path-image*: *simple-path g* $\Longrightarrow$ *connected*(*path-image g*)
⟨*proof*⟩

lemma *compact-simple-path-image*: *simple-path g* $\Longrightarrow$ *compact*(*path-image g*)
⟨*proof*⟩

lemma *bounded-simple-path-image*: *simple-path g* $\Longrightarrow$ *bounded*(*path-image g*)
⟨*proof*⟩

lemma *closed-simple-path-image*:
fixes *g* :: *real* $\Rightarrow$ *'a*::*t2-space*
shows *simple-path g* $\Longrightarrow$ *closed*(*path-image g*)
⟨*proof*⟩

lemma *connected-arc-image*: *arc g* $\Longrightarrow$ *connected*(*path-image g*)
⟨*proof*⟩

lemma *compact-arc-image*: *arc g* $\Longrightarrow$ *compact*(*path-image g*)
⟨*proof*⟩

lemma *bounded-arc-image*: *arc g* $\Longrightarrow$ *bounded*(*path-image g*)
⟨*proof*⟩

lemma *closed-arc-image*:
fixes *g* :: *real* $\Rightarrow$ *'a*::*t2-space*
shows *arc g* $\Longrightarrow$ *closed*(*path-image g*)
⟨*proof*⟩

lemma *path-image-join-subset*: *path-image* (*g1* +++ *g2*) $\subseteq$ *path-image g1* $\cup$ *path-image g2*
⟨*proof*⟩

lemma *subset-path-image-join*:
assumes *path-image g1* $\subseteq$ *s*
and *path-image g2* $\subseteq$ *s*
shows *path-image* (*g1* +++ *g2*) $\subseteq$ *s*
⟨*proof*⟩

lemma *path-image-join*:

pathfinish g1 = pathstart g2 ⟹ path-image(g1 +++ g2) = path-image g1 ∪ path-image g2
⟨*proof*⟩

lemma *not-in-path-image-join*:
assumes *x ∉ path-image g1*
and *x ∉ path-image g2*
shows *x ∉ path-image (g1 +++ g2)*
⟨*proof*⟩

lemma *pathstart-compose*: *pathstart(f o p) = f(pathstart p)*
⟨*proof*⟩

lemma *pathfinish-compose*: *pathfinish(f o p) = f(pathfinish p)*
⟨*proof*⟩

lemma *path-image-compose*: *path-image (f o p) = f ‘ (path-image p)*
⟨*proof*⟩

lemma *path-compose-join*: *f o (p +++ q) = (f o p) +++ (f o q)*
⟨*proof*⟩

lemma *path-compose-reversepath*: *f o reversepath p = reversepath(f o p)*
⟨*proof*⟩

lemma *joinpaths-eq*:
(⋀t. t ∈ {0..1} ⟹ p t = p′ t) ⟹
(⋀t. t ∈ {0..1} ⟹ q t = q′ t)
⟹ t ∈ {0..1} ⟹ (p +++ q) t = (p′ +++ q′) t
⟨*proof*⟩

lemma *simple-path-inj-on*: *simple-path g ⟹ inj-on g {0<..<1}*
⟨*proof*⟩

21.1 Simple paths with the endpoints removed

lemma *simple-path-endless*:
simple-path c ⟹ path-image c − {pathstart c,pathfinish c} = c ‘ {0<..<1}
⟨*proof*⟩

lemma *connected-simple-path-endless*:
simple-path c ⟹ connected(path-image c − {pathstart c,pathfinish c})
⟨*proof*⟩

lemma *nonempty-simple-path-endless*:
simple-path c ⟹ path-image c − {pathstart c,pathfinish c} ≠ {}
⟨*proof*⟩

21.2 The operations on paths

lemma *path-image-subset-reversepath*: *path-image*(*reversepath g*) ≤ *path-image g*
⟨*proof*⟩

lemma *path-imp-reversepath*: *path g* ⟹ *path*(*reversepath g*)
⟨*proof*⟩

lemma *half-bounded-equal*: *1* ≤ *x* ∗ *2* ⟹ *x* ∗ *2* ≤ *1* ⟷ *x* = (*1*/*2*::*real*)
⟨*proof*⟩

lemma *continuous-on-joinpaths*:
assumes *continuous-on* {*0..1*} *g1 continuous-on* {*0..1*} *g2 pathfinish g1* = *pathstart g2*
shows *continuous-on* {*0..1*} (*g1* +++ *g2*)
⟨*proof*⟩

lemma *path-join-imp*: ⟦*path g1*; *path g2*; *pathfinish g1* = *pathstart g2*⟧ ⟹ *path*(*g1* +++ *g2*)
⟨*proof*⟩

lemma *simple-path-join-loop*:
assumes *arc g1 arc g2*
pathfinish g1 = *pathstart g2* *pathfinish g2* = *pathstart g1*
path-image g1 ∩ *path-image g2* ⊆ {*pathstart g1*, *pathstart g2*}
shows *simple-path*(*g1* +++ *g2*)
⟨*proof*⟩

lemma *arc-join*:
assumes *arc g1 arc g2*
pathfinish g1 = *pathstart g2*
path-image g1 ∩ *path-image g2* ⊆ {*pathstart g2*}
shows *arc*(*g1* +++ *g2*)
⟨*proof*⟩

lemma *reversepath-joinpaths*:
pathfinish g1 = *pathstart g2* ⟹ *reversepath*(*g1* +++ *g2*) = *reversepath g2* +++ *reversepath g1*
⟨*proof*⟩

21.3 Some reversed and "if and only if" versions of joining theorems

lemma *path-join-path-ends*:
fixes *g1* :: *real* ⇒ *'a*::*metric-space*
assumes *path*(*g1* +++ *g2*) *path g2*
shows *pathfinish g1* = *pathstart g2*
⟨*proof*⟩

lemma *path-join-eq* [*simp*]:

fixes *g1* :: *real* ⇒ *'a*::*metric-space*
assumes *path g1 path g2*
shows *path*(*g1* +++ *g2*) ⟷ *pathfinish g1* = *pathstart g2*
⟨*proof*⟩

lemma *simple-path-joinE*:
assumes *simple-path*(*g1* +++ *g2*) **and** *pathfinish g1* = *pathstart g2*
obtains *arc g1 arc g2*
path-image g1 ∩ *path-image g2* ⊆ {*pathstart g1*, *pathstart g2*}
⟨*proof*⟩

lemma *simple-path-join-loop-eq*:
assumes *pathfinish g2* = *pathstart g1 pathfinish g1* = *pathstart g2*
shows *simple-path*(*g1* +++ *g2*) ⟷
arc g1 ∧ *arc g2* ∧ *path-image g1* ∩ *path-image g2* ⊆ {*pathstart g1*, *pathstart g2*}
⟨*proof*⟩

lemma *arc-join-eq*:
assumes *pathfinish g1* = *pathstart g2*
shows *arc*(*g1* +++ *g2*) ⟷
arc g1 ∧ *arc g2* ∧ *path-image g1* ∩ *path-image g2* ⊆ {*pathstart g2*}
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *arc-join-eq-alt*:
pathfinish g1 = *pathstart g2*
⟹ (*arc*(*g1* +++ *g2*) ⟷
arc g1 ∧ *arc g2* ∧
path-image g1 ∩ *path-image g2* = {*pathstart g2*})
⟨*proof*⟩

21.4 The joining of paths is associative

lemma *path-assoc*:
⟦*pathfinish p* = *pathstart q*; *pathfinish q* = *pathstart r*⟧
⟹ *path*(*p* +++ (*q* +++ *r*)) ⟷ *path*((*p* +++ *q*) +++ *r*)
⟨*proof*⟩

lemma *simple-path-assoc*:
assumes *pathfinish p* = *pathstart q pathfinish q* = *pathstart r*
shows *simple-path* (*p* +++ (*q* +++ *r*)) ⟷ *simple-path* ((*p* +++ *q*) +++ *r*)
⟨*proof*⟩

lemma *arc-assoc*:
⟦*pathfinish p* = *pathstart q*; *pathfinish q* = *pathstart r*⟧
⟹ *arc*(*p* +++ (*q* +++ *r*)) ⟷ *arc*((*p* +++ *q*) +++ *r*)
⟨*proof*⟩

21.4.1 Symmetry and loops

lemma *path-sym*:
⟦*pathfinish p = pathstart q*; *pathfinish q = pathstart p*⟧ ⟹ *path*(*p +++ q*) ⟷ *path*(*q +++ p*)
⟨*proof*⟩

lemma *simple-path-sym*:
⟦*pathfinish p = pathstart q*; *pathfinish q = pathstart p*⟧
⟹ *simple-path*(*p +++ q*) ⟷ *simple-path*(*q +++ p*)
⟨*proof*⟩

lemma *path-image-sym*:
⟦*pathfinish p = pathstart q*; *pathfinish q = pathstart p*⟧
⟹ *path-image*(*p +++ q*) = *path-image*(*q +++ p*)
⟨*proof*⟩

22 Choosing a subpath of an existing path

definition *subpath* :: *real* ⇒ *real* ⇒ (*real* ⇒ *'a*) ⇒ *real* ⇒ *'a*::*real-normed-vector*
where *subpath a b g* ≡ λ*x*. *g*((*b* − *a*) ∗ *x* + *a*)

lemma *path-image-subpath-gen*:
fixes *g* :: - ⇒ *'a*::*real-normed-vector*
shows *path-image*(*subpath u v g*) = *g* ‘ (*closed-segment u v*)
⟨*proof*⟩

lemma *path-image-subpath*:
fixes *g* :: *real* ⇒ *'a*::*real-normed-vector*
shows *path-image*(*subpath u v g*) = (*if u* ≤ *v then g* ‘ {*u*..*v*} *else g* ‘ {*v*..*u*})
⟨*proof*⟩

lemma *path-subpath* [*simp*]:
fixes *g* :: *real* ⇒ *'a*::*real-normed-vector*
assumes *path g u* ∈ {*0*..*1*} *v* ∈ {*0*..*1*}
shows *path*(*subpath u v g*)
⟨*proof*⟩

lemma *pathstart-subpath* [*simp*]: *pathstart*(*subpath u v g*) = *g*(*u*)
⟨*proof*⟩

lemma *pathfinish-subpath* [*simp*]: *pathfinish*(*subpath u v g*) = *g*(*v*)
⟨*proof*⟩

lemma *subpath-trivial* [*simp*]: *subpath 0 1 g* = *g*
⟨*proof*⟩

lemma *subpath-reversepath*: *subpath 1 0 g* = *reversepath g*
⟨*proof*⟩

lemma *reversepath-subpath*: *reversepath*(*subpath u v g*) = *subpath v u g*
⟨*proof*⟩

lemma *subpath-translation*: *subpath u v* ((λ*x*. *a* + *x*) *o g*) = (λ*x*. *a* + *x*) *o subpath u v g*
⟨*proof*⟩

lemma *subpath-linear-image*: *linear f* ⟹ *subpath u v* (*f o g*) = *f o subpath u v g*
⟨*proof*⟩

lemma *affine-ineq*:
fixes *x* :: *'a*::*linordered-idom*
assumes *x* ≤ *1 v* ≤ *u*
shows *v* + *x* ∗ *u* ≤ *u* + *x* ∗ *v*
⟨*proof*⟩

lemma *sum-le-prod1*:
fixes *a*::*real* **shows** ⟦*a* ≤ *1*; *b* ≤ *1*⟧ ⟹ *a* + *b* ≤ *1* + *a* ∗ *b*
⟨*proof*⟩

lemma *simple-path-subpath-eq*:
simple-path(*subpath u v g*) ⟷
path(*subpath u v g*) ∧ *u*≠*v* ∧
(∀ *x y*. *x* ∈ *closed-segment u v* ∧ *y* ∈ *closed-segment u v* ∧ *g x* = *g y*
⟶ *x* = *y* ∨ *x* = *u* ∧ *y* = *v* ∨ *x* = *v* ∧ *y* = *u*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *arc-subpath-eq*:
arc(*subpath u v g*) ⟷ *path*(*subpath u v g*) ∧ *u*≠*v* ∧ *inj-on g* (*closed-segment u v*)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *simple-path-subpath*:
assumes *simple-path g u* ∈ {*0..1*} *v* ∈ {*0..1*} *u* ≠ *v*
shows *simple-path*(*subpath u v g*)
⟨*proof*⟩

lemma *arc-simple-path-subpath*:
⟦*simple-path g*; *u* ∈ {*0..1*}; *v* ∈ {*0..1*}; *g u* ≠ *g v*⟧ ⟹ *arc*(*subpath u v g*)
⟨*proof*⟩

lemma *arc-subpath-arc*:
⟦*arc g*; *u* ∈ {*0..1*}; *v* ∈ {*0..1*}; *u* ≠ *v*⟧ ⟹ *arc*(*subpath u v g*)
⟨*proof*⟩

lemma *arc-simple-path-subpath-interior*:
⟦*simple-path g*; *u* ∈ {*0..1*}; *v* ∈ {*0..1*}; *u* ≠ *v*; |*u*−*v*| < *1*⟧ ⟹ *arc*(*subpath u v g*)
⟨*proof*⟩

lemma *path-image-subpath-subset*:
⟦*path g*; *u* ∈ {*0..1*}; *v* ∈ {*0..1*}⟧ ⟹ *path-image*(*subpath u v g*) ⊆ *path-image g*
⟨*proof*⟩

lemma *join-subpaths-middle*: *subpath* (*0*) ((*1* / *2*)) *p* +++ *subpath* ((*1* / *2*)) *1 p* = *p*
⟨*proof*⟩

22.1 There is a subpath to the frontier

lemma *subpath-to-frontier-explicit*:
fixes *S* :: *'a*::*metric-space set*
assumes *g*: *path g* **and** *pathfinish g* ∉ *S*
obtains *u* **where** *0* ≤ *u u* ≤ *1*
⋀*x*. *0* ≤ *x* ∧ *x* < *u* ⟹ *g x* ∈ *interior S*
(*g u* ∉ *interior S*) (*u* = *0* ∨ *g u* ∈ *closure S*)
⟨*proof*⟩

lemma *subpath-to-frontier-strong*:
assumes *g*: *path g* **and** *pathfinish g* ∉ *S*
obtains *u* **where** *0* ≤ *u u* ≤ *1 g u* ∉ *interior S*
u = *0* ∨ (∀*x*. *0* ≤ *x* ∧ *x* < *1* ⟶ *subpath 0 u g x* ∈ *interior S*)
∧ *g u* ∈ *closure S*
⟨*proof*⟩

lemma *subpath-to-frontier*:
assumes *g*: *path g* **and** *g0*: *pathstart g* ∈ *closure S* **and** *g1*: *pathfinish g* ∉ *S*
obtains *u* **where** *0* ≤ *u u* ≤ *1 g u* ∈ *frontier S* (*path-image*(*subpath 0 u g*) − {*g u*}) ⊆ *interior S*
⟨*proof*⟩

lemma *exists-path-subpath-to-frontier*:
fixes *S* :: *'a*::*real-normed-vector set*
assumes *path g pathstart g* ∈ *closure S pathfinish g* ∉ *S*
obtains *h* **where** *path h pathstart h* = *pathstart g path-image h* ⊆ *path-image g*
path-image h − {*pathfinish h*} ⊆ *interior S*
pathfinish h ∈ *frontier S*
⟨*proof*⟩

lemma *exists-path-subpath-to-frontier-closed*:
fixes *S* :: *'a*::*real-normed-vector set*
assumes *S*: *closed S* **and** *g*: *path g* **and** *g0*: *pathstart g* ∈ *S* **and** *g1*: *pathfinish g* ∉ *S*

obtains h **where** *path* h *pathstart* h = *pathstart* g *path-image* $h \subseteq$ *path-image* $g \cap S$
pathfinish $h \in$ *frontier* S
⟨*proof*⟩

22.2 Reparametrizing a closed curve to start at some chosen point

definition *shiftpath* :: *real* ⇒ (*real* ⇒ 'a::*topological-space*) ⇒ *real* ⇒ 'a
where *shiftpath* a f = (λx. *if* $(a + x) \leq 1$ *then* f $(a + x)$ *else* f $(a + x - 1)$)

lemma *pathstart-shiftpath*: $a \leq 1 \Longrightarrow$ *pathstart* (*shiftpath* a g) = g a
⟨*proof*⟩

lemma *pathfinish-shiftpath*:
assumes $0 \leq a$
and *pathfinish* g = *pathstart* g
shows *pathfinish* (*shiftpath* a g) = g a
⟨*proof*⟩

lemma *endpoints-shiftpath*:
assumes *pathfinish* g = *pathstart* g
and $a \in \{0 \,..\, 1\}$
shows *pathfinish* (*shiftpath* a g) = g a
and *pathstart* (*shiftpath* a g) = g a
⟨*proof*⟩

lemma *closed-shiftpath*:
assumes *pathfinish* g = *pathstart* g
and $a \in \{0..1\}$
shows *pathfinish* (*shiftpath* a g) = *pathstart* (*shiftpath* a g)
⟨*proof*⟩

lemma *path-shiftpath*:
assumes *path* g
and *pathfinish* g = *pathstart* g
and $a \in \{0..1\}$
shows *path* (*shiftpath* a g)
⟨*proof*⟩

lemma *shiftpath-shiftpath*:
assumes *pathfinish* g = *pathstart* g
and $a \in \{0..1\}$
and $x \in \{0..1\}$
shows *shiftpath* $(1 - a)$ (*shiftpath* a g) $x = g\ x$
⟨*proof*⟩

lemma *path-image-shiftpath*:
assumes $a \in \{0..1\}$

and *pathfinish g* = *pathstart g*
shows *path-image* (*shiftpath a g*) = *path-image g*
⟨*proof*⟩

22.3 Special case of straight-line paths

definition *linepath* :: *'a*::*real-normed-vector* ⇒ *'a* ⇒ *real* ⇒ *'a*
where *linepath a b* = (λ*x*. (*1* − *x*) *∗*$_R$ *a* + *x* *∗*$_R$ *b*)

lemma *pathstart-linepath*[*simp*]: *pathstart* (*linepath a b*) = *a*
⟨*proof*⟩

lemma *pathfinish-linepath*[*simp*]: *pathfinish* (*linepath a b*) = *b*
⟨*proof*⟩

lemma *continuous-linepath-at*[*intro*]: *continuous* (*at x*) (*linepath a b*)
⟨*proof*⟩

lemma *continuous-on-linepath* [*intro*,*continuous-intros*]: *continuous-on s* (*linepath a b*)
⟨*proof*⟩

lemma *path-linepath*[*iff*]: *path* (*linepath a b*)
⟨*proof*⟩

lemma *path-image-linepath*[*simp*]: *path-image* (*linepath a b*) = *closed-segment a b*
⟨*proof*⟩

lemma *reversepath-linepath*[*simp*]: *reversepath* (*linepath a b*) = *linepath b a*
⟨*proof*⟩

lemma *linepath-0* [*simp*]: *linepath 0 b x* = *x* *∗*$_R$ *b*
⟨*proof*⟩

lemma *arc-linepath*:
assumes *a* ≠ *b* **shows** [*simp*]: *arc* (*linepath a b*)
⟨*proof*⟩

lemma *simple-path-linepath*[*intro*]: *a* ≠ *b* ⟹ *simple-path* (*linepath a b*)
⟨*proof*⟩

lemma *linepath-trivial* [*simp*]: *linepath a a x* = *a*
⟨*proof*⟩

lemma *subpath-refl*: *subpath a a g* = *linepath* (*g a*) (*g a*)
⟨*proof*⟩

lemma *linepath-of-real*: (*linepath* (*of-real a*) (*of-real b*) *x*) = *of-real* ((*1* − *x*)*∗a* + *x∗b*)

⟨*proof*⟩

lemma *of-real-linepath*: *of-real* (*linepath a b x*) = *linepath* (*of-real a*) (*of-real b*) *x*
⟨*proof*⟩

22.4 Segments via convex hulls

lemma *segments-subset-convex-hull*:
closed-segment a b ⊆ (*convex hull* {*a*,*b*,*c*})
closed-segment a c ⊆ (*convex hull* {*a*,*b*,*c*})
closed-segment b c ⊆ (*convex hull* {*a*,*b*,*c*})
closed-segment b a ⊆ (*convex hull* {*a*,*b*,*c*})
closed-segment c a ⊆ (*convex hull* {*a*,*b*,*c*})
closed-segment c b ⊆ (*convex hull* {*a*,*b*,*c*})
⟨*proof*⟩

lemma *midpoints-in-convex-hull*:
assumes *x* ∈ *convex hull s y* ∈ *convex hull s*
shows *midpoint x y* ∈ *convex hull s*
⟨*proof*⟩

lemma *convex-hull-subset*:
s ⊆ *convex hull t* ⟹ *convex hull s* ⊆ *convex hull t*
⟨*proof*⟩

lemma *not-in-interior-convex-hull-3*:
fixes *a* :: *complex*
shows *a* ∉ *interior*(*convex hull* {*a*,*b*,*c*})
b ∉ *interior*(*convex hull* {*a*,*b*,*c*})
c ∉ *interior*(*convex hull* {*a*,*b*,*c*})
⟨*proof*⟩

lemma *midpoint-in-closed-segment* [*simp*]: *midpoint a b* ∈ *closed-segment a b*
⟨*proof*⟩

lemma *midpoint-in-open-segment* [*simp*]: *midpoint a b* ∈ *open-segment a b* ⟷ *a* ≠ *b*
⟨*proof*⟩

22.5 Bounding a point away from a path

lemma *not-on-path-ball*:
fixes *g* :: *real* ⇒ *'a*::*heine-borel*
assumes *path g*
and *z* ∉ *path-image g*
shows ∃ *e* > *0*. *ball z e* ∩ *path-image g* = {}
⟨*proof*⟩

lemma *not-on-path-cball*:
fixes *g* :: *real* ⇒ *'a*::*heine-borel*

assumes *path g*
and $z \notin$ *path-image g*
shows $\exists e{>}0.$ *cball z e* $\cap$ (*path-image g*) $= \{\}$
⟨*proof*⟩

23 Path component, considered as a "joinability" relation (from Tom Hales)

definition *path-component s x y* $\longleftrightarrow$
$(\exists g.$ *path g* $\wedge$ *path-image g* $\subseteq$ *s* $\wedge$ *pathstart g* $=$ *x* $\wedge$ *pathfinish g* $=$ *y*$)$

abbreviation
path-component-set s x $\equiv$ *Collect* (*path-component s x*)

lemmas *path-defs* $=$ *path-def pathstart-def pathfinish-def path-image-def path-component-def*

lemma *path-component-mem*:
assumes *path-component s x y*
shows $x \in s$ **and** $y \in s$
⟨*proof*⟩

lemma *path-component-refl*:
assumes $x \in s$
shows *path-component s x x*
⟨*proof*⟩

lemma *path-component-refl-eq*: *path-component s x x* $\longleftrightarrow$ $x \in s$
⟨*proof*⟩

lemma *path-component-sym*: *path-component s x y* $\Longrightarrow$ *path-component s y x*
⟨*proof*⟩

lemma *path-component-trans*:
assumes *path-component s x y* **and** *path-component s y z*
shows *path-component s x z*
⟨*proof*⟩

lemma *path-component-of-subset*: $s \subseteq t \Longrightarrow$ *path-component s x y* $\Longrightarrow$ *path-component t x y*
⟨*proof*⟩

lemma *path-connected-linepath*:
fixes *s* :: *'a*::*real-normed-vector set*
shows *closed-segment a b* $\subseteq$ *s* $\Longrightarrow$ *path-component s a b*
⟨*proof*⟩

23.0.1 Path components as sets

lemma *path-component-set*:
path-component-set s x =
{*y*. (∃ *g*. *path g* ∧ *path-image g* ⊆ *s* ∧ *pathstart g* = *x* ∧ *pathfinish g* = *y*)}
⟨*proof*⟩

lemma *path-component-subset*: *path-component-set s x* ⊆ *s*
⟨*proof*⟩

lemma *path-component-eq-empty*: *path-component-set s x* = {} ⟷ *x* ∉ *s*
⟨*proof*⟩

lemma *path-component-mono*:
s ⊆ *t* ⟹ (*path-component-set s x*) ⊆ (*path-component-set t x*)
⟨*proof*⟩

lemma *path-component-eq*:
y ∈ *path-component-set s x* ⟹ *path-component-set s y* = *path-component-set s x*
⟨*proof*⟩

23.1 Path connectedness of a space

definition *path-connected s* ⟷
(∀ *x*∈*s*. ∀ *y*∈*s*. ∃ *g*. *path g* ∧ *path-image g* ⊆ *s* ∧ *pathstart g* = *x* ∧ *pathfinish g* = *y*)

lemma *path-connected-component*: *path-connected s* ⟷ (∀ *x*∈*s*. ∀ *y*∈*s*. *path-component s x y*)
⟨*proof*⟩

lemma *path-connected-component-set*: *path-connected s* ⟷ (∀ *x*∈*s*. *path-component-set s x* = *s*)
⟨*proof*⟩

lemma *path-component-maximal*:
⟦*x* ∈ *t*; *path-connected t*; *t* ⊆ *s*⟧ ⟹ *t* ⊆ (*path-component-set s x*)
⟨*proof*⟩

lemma *convex-imp-path-connected*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *convex s*
shows *path-connected s*
⟨*proof*⟩

lemma *path-connected-UNIV* [*iff*]: *path-connected* (*UNIV* :: *'a*::*real-normed-vector set*)
⟨*proof*⟩

lemma *path-component-UNIV*: *path-component-set UNIV x* = (*UNIV* :: *'a*::*real-normed-vector set*)
⟨*proof*⟩

lemma *path-connected-imp-connected*:
assumes *path-connected s*
shows *connected s*
⟨*proof*⟩

lemma *open-path-component*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *open s*
shows *open* (*path-component-set s x*)
⟨*proof*⟩

lemma *open-non-path-component*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *open s*
shows *open* (*s* − *path-component-set s x*)
⟨*proof*⟩

lemma *connected-open-path-connected*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *open s*
and *connected s*
shows *path-connected s*
⟨*proof*⟩

lemma *path-connected-continuous-image*:
assumes *continuous-on s f*
and *path-connected s*
shows *path-connected* (*f* ‘ *s*)
⟨*proof*⟩

lemma *path-connected-segment*:
fixes *a* :: *'a*::*real-normed-vector*
shows *path-connected* (*closed-segment a b*)
⟨*proof*⟩

lemma *path-connected-open-segment*:
fixes *a* :: *'a*::*real-normed-vector*
shows *path-connected* (*open-segment a b*)
⟨*proof*⟩

lemma *homeomorphic-path-connectedness*:
s homeomorphic t ⟹ *path-connected s* ⟷ *path-connected t*
⟨*proof*⟩

lemma *path-connected-empty*: *path-connected* {}

⟨*proof*⟩

lemma *path-connected-singleton*: *path-connected* {*a*}
⟨*proof*⟩

lemma *path-connected-Un*:
assumes *path-connected s*
and *path-connected t*
and $s \cap t \neq \{\}$
shows *path-connected* $(s \cup t)$
⟨*proof*⟩

lemma *path-connected-UNION*:
assumes $\bigwedge i.\ i \in A \Longrightarrow$ *path-connected* $(S\ i)$
and $\bigwedge i.\ i \in A \Longrightarrow z \in S\ i$
shows *path-connected* $(\bigcup i{\in}A.\ S\ i)$
⟨*proof*⟩

lemma *path-component-path-image-pathstart*:
assumes *p*: *path p* **and** *x*: $x \in$ *path-image p*
shows *path-component* (*path-image p*) (*pathstart p*) *x*
⟨*proof*⟩

lemma *path-connected-path-image*: *path p* $\Longrightarrow$ *path-connected*(*path-image p*)
⟨*proof*⟩

lemma *path-connected-path-component*:
path-connected (*path-component-set s x*)
⟨*proof*⟩

lemma *path-component*: *path-component s x y* $\longleftrightarrow$ $(\exists t.$ *path-connected t* $\wedge$ $t \subseteq s$ $\wedge$ $x \in t \wedge y \in t)$
⟨*proof*⟩

lemma *path-component-path-component* [*simp*]:
path-component-set (*path-component-set s x*) *x* = *path-component-set s x*
⟨*proof*⟩

lemma *path-component-subset-connected-component*:
(*path-component-set s x*) $\subseteq$ (*connected-component-set s x*)
⟨*proof*⟩

23.2 Lemmas about path-connectedness

lemma *path-connected-linear-image*:
fixes *f* :: *'a*::*real-normed-vector* $\Rightarrow$ *'b*::*real-normed-vector*
assumes *path-connected s bounded-linear f*
shows *path-connected*(*f* ‘ *s*)
⟨*proof*⟩

lemma *is-interval-path-connected*: *is-interval s* $\Longrightarrow$ *path-connected s*
⟨*proof*⟩

lemma *linear-homeomorphism-image*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'b*::*euclidean-space*
assumes *linear f inj f*
obtains *g* **where** *homeomorphism* (*f* ' *S*) *S g f*
⟨*proof*⟩

lemma *linear-homeomorphic-image*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'b*::*euclidean-space*
assumes *linear f inj f*
shows *S homeomorphic f* ' *S*
⟨*proof*⟩

lemma *path-connected-Times*:
assumes *path-connected s path-connected t*
shows *path-connected* (*s* × *t*)
⟨*proof*⟩

lemma *is-interval-path-connected-1*:
fixes *s* :: *real set*
shows *is-interval s* $\longleftrightarrow$ *path-connected s*
⟨*proof*⟩

lemma *Union-path-component* [*simp*]:
Union {*path-component-set S x* |*x*. *x* ∈ *S*} = *S*
⟨*proof*⟩

lemma *path-component-disjoint*:
disjnt (*path-component-set S a*) (*path-component-set S b*) $\longleftrightarrow$
(*a* ∉ *path-component-set S b*)
⟨*proof*⟩

lemma *path-component-eq-eq*:
path-component S x = *path-component S y* $\longleftrightarrow$
(*x* ∉ *S*) ∧ (*y* ∉ *S*) ∨ *x* ∈ *S* ∧ *y* ∈ *S* ∧ *path-component S x y*
⟨*proof*⟩

lemma *path-component-unique*:
assumes *x* ∈ *c c* ⊆ *S path-connected c*
⋀*c'*. ⟦*x* ∈ *c'*; *c'* ⊆ *S*; *path-connected c'*⟧ $\Longrightarrow$ *c'* ⊆ *c*
shows *path-component-set S x* = *c*
⟨*proof*⟩

lemma *path-component-intermediate-subset*:
path-component-set u a ⊆ *t* ∧ *t* ⊆ *u*

$\Longrightarrow$ *path-component-set t a = path-component-set u a*
⟨*proof*⟩

lemma *complement-path-component-Union*:
fixes *x* :: *'a* :: *topological-space*
shows *S − path-component-set S x* =
$\bigcup$({*path-component-set S y*| *y*. *y* ∈ *S*} − {*path-component-set S x*})
⟨*proof*⟩

23.3 Sphere is path-connected

lemma *path-connected-punctured-universe*:
assumes *2* ≤ *DIM*(*'a*::*euclidean-space*)
shows *path-connected* (− {*a*::*'a*})
⟨*proof*⟩

lemma *path-connected-sphere*:
assumes *2* ≤ *DIM*(*'a*::*euclidean-space*)
shows *path-connected* {*x*::*'a*. *norm* (*x* − *a*) = *r*}
⟨*proof*⟩

corollary *connected-sphere*: *2* ≤ *DIM*(*'a*::*euclidean-space*) $\Longrightarrow$ *connected* {*x*::*'a*. *norm* (*x* − *a*) = *r*}
⟨*proof*⟩

corollary *path-connected-complement-bounded-convex*:
fixes *s* :: *'a* :: *euclidean-space set*
assumes *bounded s convex s* **and** *2*: *2* ≤ *DIM*(*'a*)
shows *path-connected* (− *s*)
⟨*proof*⟩

lemma *connected-complement-bounded-convex*:
fixes *s* :: *'a* :: *euclidean-space set*
assumes *bounded s convex s 2* ≤ *DIM*(*'a*)
shows *connected* (− *s*)
⟨*proof*⟩

lemma *connected-diff-ball*:
fixes *s* :: *'a* :: *euclidean-space set*
assumes *connected s cball a r* ⊆ *s 2* ≤ *DIM*(*'a*)
shows *connected* (*s* − *ball a r*)
⟨*proof*⟩

proposition *connected-open-delete*:
assumes *open S connected S* **and** *2*: *2* ≤ *DIM*(*'N*::*euclidean-space*)
shows *connected*(*S* − {*a*::*'N*})
⟨*proof*⟩

corollary *path-connected-open-delete*:
assumes *open S connected S* **and** *2*: $2 \leq DIM('N::euclidean\text{-}space)$
shows *path-connected*$(S - \{a::'N\})$
⟨*proof*⟩

corollary *path-connected-punctured-ball*:
$2 \leq DIM('N::euclidean\text{-}space) \Longrightarrow path\text{-}connected(ball\ a\ r - \{a::'N\})$
⟨*proof*⟩

lemma *connected-punctured-ball*:
$2 \leq DIM('N::euclidean\text{-}space) \Longrightarrow connected(ball\ a\ r - \{a::'N\})$
⟨*proof*⟩

23.4 Relations between components and path components

lemma *open-connected-component*:
fixes $s :: 'a::real\text{-}normed\text{-}vector\ set$
shows $open\ s \Longrightarrow open\ (connected\text{-}component\text{-}set\ s\ x)$
⟨*proof*⟩

corollary *open-components*:
fixes $s :: 'a::real\text{-}normed\text{-}vector\ set$
shows $[\![open\ u;\ s \in components\ u]\!] \Longrightarrow open\ s$
⟨*proof*⟩

lemma *in-closure-connected-component*:
fixes $s :: 'a::real\text{-}normed\text{-}vector\ set$
assumes *x*: $x \in s$ **and** *s*: *open s*
shows $x \in closure\ (connected\text{-}component\text{-}set\ s\ y) \longleftrightarrow x \in connected\text{-}component\text{-}set\ s\ y$
⟨*proof*⟩

23.5 Existence of unbounded components

lemma *cobounded-unbounded-component*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
assumes *bounded* $(-s)$
shows $\exists x.\ x \in s \wedge \sim bounded\ (connected\text{-}component\text{-}set\ s\ x)$
⟨*proof*⟩

lemma *cobounded-unique-unbounded-component*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
assumes *bs*: *bounded* $(-s)$ **and** $2 \leq DIM('a)$
and *bo*: $\sim bounded(connected\text{-}component\text{-}set\ s\ x)$
$\sim bounded(connected\text{-}component\text{-}set\ s\ y)$
shows $connected\text{-}component\text{-}set\ s\ x = connected\text{-}component\text{-}set\ s\ y$
⟨*proof*⟩

lemma *cobounded-unbounded-components*:
fixes $s :: 'a :: euclidean\text{-}space\ set$

shows *bounded* $(-s) \Longrightarrow \exists\, c.\ c \in$ *components* $s \wedge \sim$*bounded* c
⟨*proof*⟩

lemma *cobounded-unique-unbounded-components*:
fixes $s :: 'a ::$ *euclidean-space set*
shows $[\![$*bounded* $(-\ s)$; $c \in$ *components* s; $\neg$ *bounded* c; $c' \in$ *components* s; $\neg$ *bounded* c'; $2 \leq DIM('a)]\!] \Longrightarrow c' = c$
⟨*proof*⟩

lemma *cobounded-has-bounded-component*:
fixes $s :: 'a ::$ *euclidean-space set*
shows $[\![$*bounded* $(-\ s)$; $\sim$*connected* s; $2 \leq DIM('a)]\!] \Longrightarrow \exists\, c.\ c \in$ *components* $s \wedge$ *bounded* c
⟨*proof*⟩

24 The "inside" and "outside" of a set

The inside comprises the points in a bounded connected component of the set's complement. The outside comprises the points in unbounded connected component of the complement.

definition *inside* **where**
inside $s \equiv \{x.\ (x \notin s) \wedge$ *bounded*(*connected-component-set* $(-\ s)\ x)\}$

definition *outside* **where**
outside $s \equiv -s \cap \{x.\ \sim$ *bounded*(*connected-component-set* $(-\ s)\ x)\}$

lemma *outside*: *outside* $s = \{x.\ \sim$ *bounded*(*connected-component-set* $(-\ s)\ x)\}$
⟨*proof*⟩

lemma *inside-no-overlap* [*simp*]: *inside* $s \cap s = \{\}$
⟨*proof*⟩

lemma *outside-no-overlap* [*simp*]:
outside $s \cap s = \{\}$
⟨*proof*⟩

lemma *inside-inter-outside* [*simp*]: *inside* $s \cap$ *outside* $s = \{\}$
⟨*proof*⟩

lemma *inside-union-outside* [*simp*]: *inside* $s \cup$ *outside* $s = (-\ s)$
⟨*proof*⟩

lemma *inside-eq-outside*:
inside $s =$ *outside* $s \longleftrightarrow s =$ *UNIV*
⟨*proof*⟩

lemma *inside-outside*: *inside* $s = (-\ (s \cup$ *outside* $s))$
⟨*proof*⟩

lemma *outside-inside*: $outside\ s = (-\ (s \cup inside\ s))$
⟨*proof*⟩

lemma *union-with-inside*: $s \cup inside\ s = -\ outside\ s$
⟨*proof*⟩

lemma *union-with-outside*: $s \cup outside\ s = -\ inside\ s$
⟨*proof*⟩

lemma *outside-mono*: $s \subseteq t \Longrightarrow outside\ t \subseteq outside\ s$
⟨*proof*⟩

lemma *inside-mono*: $s \subseteq t \Longrightarrow inside\ s - t \subseteq inside\ t$
⟨*proof*⟩

lemma *segment-bound-lemma*:
fixes *u*::*real*
assumes $x \geq B\ y \geq B\ 0 \leq u\ u \leq 1$
shows $(1 - u) * x + u * y \geq B$
⟨*proof*⟩

lemma *cobounded-outside*:
fixes $s :: 'a :: real\text{-}normed\text{-}vector\ set$
assumes *bounded s* **shows** *bounded* $(-\ outside\ s)$
⟨*proof*⟩

lemma *unbounded-outside*:
fixes $s :: 'a::\{real\text{-}normed\text{-}vector, perfect\text{-}space\}\ set$
shows $bounded\ s \Longrightarrow \sim\ bounded(outside\ s)$
⟨*proof*⟩

lemma *bounded-inside*:
fixes $s :: 'a::\{real\text{-}normed\text{-}vector, perfect\text{-}space\}\ set$
shows $bounded\ s \Longrightarrow bounded(inside\ s)$
⟨*proof*⟩

lemma *connected-outside*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes $bounded\ s\ 2 \leq DIM('a)$
shows $connected(outside\ s)$
⟨*proof*⟩

lemma *outside-connected-component-lt*:
$outside\ s = \{x.\ \forall B.\ \exists y.\ B < norm(y) \wedge connected\text{-}component\ (-\ s)\ x\ y\}$
⟨*proof*⟩

lemma *outside-connected-component-le*:
$outside\ s =$

$\{x.\ \forall B.\ \exists y.\ B \leq norm(y) \wedge$
connected-component $(-\ s)\ x\ y\}$
⟨*proof*⟩

lemma *not-outside-connected-component-lt*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *bounded* s **and** $2 \leq DIM('a)$
shows $-$ (*outside* s) = $\{x.\ \forall B.\ \exists y.\ B < norm(y) \wedge \sim$ (*connected-component* $(-\ s)\ x\ y)\}$
⟨*proof*⟩

lemma *not-outside-connected-component-le*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *bounded* s $2 \leq DIM('a)$
shows $-$ (*outside* s) = $\{x.\ \forall B.\ \exists y.\ B \leq norm(y) \wedge \sim$ (*connected-component* $(-\ s)\ x\ y)\}$
⟨*proof*⟩

lemma *inside-connected-component-lt*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *bounded* s $2 \leq DIM('a)$
shows *inside* s = $\{x.\ (x \notin s) \wedge (\forall B.\ \exists y.\ B < norm(y) \wedge \sim$(*connected-component* $(-\ s)\ x\ y))\}$
⟨*proof*⟩

lemma *inside-connected-component-le*:
fixes s :: $'a$::*euclidean-space set*
assumes s: *bounded* s $2 \leq DIM('a)$
shows *inside* s = $\{x.\ (x \notin s) \wedge (\forall B.\ \exists y.\ B \leq norm(y) \wedge \sim$(*connected-component* $(-\ s)\ x\ y))\}$
⟨*proof*⟩

lemma *inside-subset*:
assumes *connected* u **and** $\sim$*bounded* u **and** $t \cup u = -\ s$
shows *inside* $s \subseteq t$
⟨*proof*⟩

lemma *frontier-interiors*: *frontier* $s = -$ *interior*$(s) -$ *interior*$(-s)$
⟨*proof*⟩

lemma *frontier-interior-subset*: *frontier*(*interior* S) $\subseteq$ *frontier* S
⟨*proof*⟩

lemma *connected-Int-frontier*:
⟦*connected* s; $s \cap t \neq \{\}$; $s - t \neq \{\}$⟧ $\Longrightarrow$ $(s \cap$ *frontier* $t \neq \{\})$
⟨*proof*⟩

lemma *frontier-not-empty*:
fixes S :: $'a$ :: *real-normed-vector set*

shows $[\![S \neq \{\}; S \neq UNIV]\!] \Longrightarrow frontier\ S \neq \{\}$
⟨*proof*⟩

lemma *frontier-eq-empty*:
fixes $S :: 'a :: real\text{-}normed\text{-}vector\ set$
shows $frontier\ S = \{\} \longleftrightarrow S = \{\} \vee S = UNIV$
⟨*proof*⟩

lemma *frontier-of-connected-component-subset*:
fixes $S :: 'a{::}real\text{-}normed\text{-}vector\ set$
shows $frontier(connected\text{-}component\text{-}set\ S\ x) \subseteq frontier\ S$
⟨*proof*⟩

lemma *frontier-Union-subset-closure*:
fixes $F :: 'a{::}real\text{-}normed\text{-}vector\ set\ set$
shows $frontier(\bigcup F) \subseteq closure(\bigcup t \in F.\ frontier\ t)$
⟨*proof*⟩

lemma *frontier-Union-subset*:
fixes $F :: 'a{::}real\text{-}normed\text{-}vector\ set\ set$
shows $finite\ F \Longrightarrow frontier(\bigcup F) \subseteq (\bigcup t \in F.\ frontier\ t)$
⟨*proof*⟩

lemma *connected-component-UNIV* [*simp*]:
fixes $x :: 'a{::}real\text{-}normed\text{-}vector$
shows $connected\text{-}component\text{-}set\ UNIV\ x = UNIV$
⟨*proof*⟩

lemma *connected-component-eq-UNIV*:
fixes $x :: 'a{::}real\text{-}normed\text{-}vector$
shows $connected\text{-}component\text{-}set\ s\ x = UNIV \longleftrightarrow s = UNIV$
⟨*proof*⟩

lemma *components-univ* [*simp*]: $components\ UNIV = \{UNIV :: 'a{::}real\text{-}normed\text{-}vector\ set\}$
⟨*proof*⟩

lemma *interior-inside-frontier*:
fixes $s :: 'a{::}real\text{-}normed\text{-}vector\ set$
assumes *bounded s*
shows $interior\ s \subseteq inside\ (frontier\ s)$
⟨*proof*⟩

lemma *inside-empty* [*simp*]: $inside\ \{\} = (\{\} :: 'a :: \{real\text{-}normed\text{-}vector, perfect\text{-}space\}\ set)$
⟨*proof*⟩

lemma *outside-empty* [*simp*]: $outside\ \{\} = (UNIV :: 'a :: \{real\text{-}normed\text{-}vector, perfect\text{-}space\}\ set)$

⟨*proof*⟩

lemma *inside-same-component*:
⟦*connected-component* (− *s*) *x y*; *x* ∈ *inside s*⟧ ⟹ *y* ∈ *inside s*
⟨*proof*⟩

lemma *outside-same-component*:
⟦*connected-component* (− *s*) *x y*; *x* ∈ *outside s*⟧ ⟹ *y* ∈ *outside s*
⟨*proof*⟩

lemma *convex-in-outside*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
assumes *s*: *convex s* **and** *z*: *z* ∉ *s*
shows *z* ∈ *outside s*
⟨*proof*⟩

lemma *outside-convex*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
assumes *convex s*
shows *outside s* = − *s*
⟨*proof*⟩

lemma *inside-convex*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
shows *convex s* ⟹ *inside s* = {}
⟨*proof*⟩

lemma *outside-subset-convex*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
shows ⟦*convex t*; *s* ⊆ *t*⟧ ⟹ − *t* ⊆ *outside s*
⟨*proof*⟩

lemma *outside-frontier-misses-closure*:
fixes *s* :: *'a*::*real-normed-vector set*
assumes *bounded s*
shows *outside*(*frontier s*) ⊆ − *closure s*
⟨*proof*⟩

lemma *outside-frontier-eq-complement-closure*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
assumes *bounded s convex s*
shows *outside*(*frontier s*) = − *closure s*
⟨*proof*⟩

lemma *inside-frontier-eq-interior*:
fixes *s* :: *'a* :: {*real-normed-vector*, *perfect-space*} *set*
shows ⟦*bounded s*; *convex s*⟧ ⟹ *inside*(*frontier s*) = *interior s*
⟨*proof*⟩

lemma *open-inside*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *open* (*inside s*)
⟨*proof*⟩

lemma *open-outside*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *open* (*outside s*)
⟨*proof*⟩

lemma *closure-inside-subset*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *closure*(*inside s*) ⊆ s ∪ *inside s*
⟨*proof*⟩

lemma *frontier-inside-subset*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *frontier*(*inside s*) ⊆ s
⟨*proof*⟩

lemma *closure-outside-subset*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *closure*(*outside s*) ⊆ s ∪ *outside s*
⟨*proof*⟩

lemma *frontier-outside-subset*:
fixes s :: *'a*::*real-normed-vector set*
assumes *closed s*
shows *frontier*(*outside s*) ⊆ s
⟨*proof*⟩

lemma *inside-complement-unbounded-connected-empty*:
⟦*connected* (− s); ¬ *bounded* (− s)⟧ ⟹ *inside s* = {}
⟨*proof*⟩

lemma *inside-bounded-complement-connected-empty*:
fixes s :: *'a*::{*real-normed-vector*, *perfect-space*} *set*
shows ⟦*connected* (− s); *bounded s*⟧ ⟹ *inside s* = {}
⟨*proof*⟩

lemma *inside-inside*:
assumes s ⊆ *inside t*
shows *inside s* − t ⊆ *inside t*
⟨*proof*⟩

lemma *inside-inside-subset*: $inside(inside\ s) \subseteq s$
⟨*proof*⟩

lemma *inside-outside-intersect-connected*:
$[\![connected\ t;\ inside\ s \cap t \neq \{\};\ outside\ s \cap t \neq \{\}]\!] \Longrightarrow s \cap t \neq \{\}$
⟨*proof*⟩

lemma *outside-bounded-nonempty*:
fixes $s :: 'a :: \{real\text{-}normed\text{-}vector,\ perfect\text{-}space\}\ set$
assumes *bounded s* **shows** $outside\ s \neq \{\}$
⟨*proof*⟩

lemma *outside-compact-in-open*:
fixes $s :: 'a :: \{real\text{-}normed\text{-}vector,perfect\text{-}space\}\ set$
assumes *s*: *compact s* **and** *t*: *open t* **and** $s \subseteq t\ t \neq \{\}$
shows $outside\ s \cap t \neq \{\}$
⟨*proof*⟩

lemma *inside-inside-compact-connected*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
assumes *s*: *closed s* **and** *t*: *compact t* **and** $connected\ t\ s \subseteq inside\ t$
shows $inside\ s \subseteq inside\ t$
⟨*proof*⟩

lemma *connected-with-inside*:
fixes $s :: 'a :: real\text{-}normed\text{-}vector\ set$
assumes *s*: *closed s* **and** *cons*: *connected s*
shows $connected(s \cup inside\ s)$
⟨*proof*⟩

The proof is virtually the same as that above.

lemma *connected-with-outside*:
fixes $s :: 'a :: real\text{-}normed\text{-}vector\ set$
assumes *s*: *closed s* **and** *cons*: *connected s*
shows $connected(s \cup outside\ s)$
⟨*proof*⟩

lemma *inside-inside-eq-empty* [*simp*]:
fixes $s :: 'a :: \{real\text{-}normed\text{-}vector,\ perfect\text{-}space\}\ set$
assumes *s*: *closed s* **and** *cons*: *connected s*
shows $inside\ (inside\ s) = \{\}$
⟨*proof*⟩

lemma *inside-in-components*:
$inside\ s \in components\ (-\ s) \longleftrightarrow connected(inside\ s) \wedge inside\ s \neq \{\}$
⟨*proof*⟩

The proof is virtually the same as that above.

lemma *outside-in-components*:
$outside\ s \in components\ (-\ s) \longleftrightarrow connected(outside\ s) \wedge outside\ s \neq \{\}$
⟨*proof*⟩

lemma *bounded-unique-outside*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
shows $[\![bounded\ s;\ DIM('a) \geq 2]\!] \Longrightarrow (c \in components\ (-\ s) \wedge \sim bounded\ c \longleftrightarrow c = outside\ s)$
⟨*proof*⟩

24.1 Condition for an open map's image to contain a ball

lemma *ball-subset-open-map-image*:
fixes $f :: 'a::heine\text{-}borel \Rightarrow 'b :: \{real\text{-}normed\text{-}vector, heine\text{-}borel\}$
assumes *contf*: *continuous-on* (*closure S*) *f*
and *oint*: *open* (*f ' interior S*)
and *le-no*: $\bigwedge z.\ z \in frontier\ S \Longrightarrow r \leq norm(f\ z - f\ a)$
and *bounded S a* ∈ *S 0* < *r*
shows *ball* (*f a*) *r* ⊆ *f ' S*
⟨*proof*⟩

25 Homotopy of maps p,q : X=¿Y with property P of all intermediate maps.

We often just want to require that it fixes some subset, but to take in the case of a loop homotopy, it's convenient to have a general property P.

definition *homotopic-with* ::
$[('a::topological\text{-}space \Rightarrow 'b::topological\text{-}space) \Rightarrow bool, 'a\ set, 'b\ set, 'a \Rightarrow 'b, 'a \Rightarrow 'b] \Rightarrow bool$
where
homotopic-with P X Y p q ≡
(∃ *h*:: *real* × *'a* ⇒ *'b*.
continuous-on ({*0..1*} × *X*) *h* ∧
h ' ({0..1} × X) ⊆ Y ∧
(∀ *x*. *h*(*0*, *x*) = *p x*) ∧
(∀ *x*. *h*(*1*, *x*) = *q x*) ∧
(∀ *t* ∈ {*0..1*}. *P*(λ*x*. *h*(*t*, *x*))))

We often want to just localize the ending function equality or whatever.

proposition *homotopic-with*:
fixes *X* :: *'a::topological-space set* **and** *Y* :: *'b::topological-space set*
assumes $\bigwedge h\ k.\ (\bigwedge x.\ x \in X \Longrightarrow h\ x = k\ x) \Longrightarrow (P\ h \longleftrightarrow P\ k)$
shows *homotopic-with P X Y p q* ⟷
(∃ *h* :: *real* × *'a* ⇒ *'b*.
continuous-on ({*0..1*} × *X*) *h* ∧
h ' ({0..1} × X) ⊆ Y ∧
(∀ *x* ∈ *X*. *h*(*0*,*x*) = *p x*) ∧

$(\forall\, x \in X.\ h(1,x) = q\ x) \wedge$
$(\forall\, t \in \{0..1\}.\ P(\lambda x.\ h(t,\ x))))$
⟨*proof*⟩

proposition *homotopic-with-eq*:
assumes *h*: *homotopic-with P X Y f g*
and *f'*: $\bigwedge x.\ x \in X \Longrightarrow f'\ x = f\ x$
and *g'*: $\bigwedge x.\ x \in X \Longrightarrow g'\ x = g\ x$
and *P*: $(\bigwedge h\ k.\ (\bigwedge x.\ x \in X \Longrightarrow h\ x = k\ x) \Longrightarrow (P\ h \longleftrightarrow P\ k))$
shows *homotopic-with P X Y f' g'*
⟨*proof*⟩

proposition *homotopic-with-equal*:
assumes *contf*: *continuous-on X f* **and** *fXY*: $f\ `\ X \subseteq Y$
and *gf*: $\bigwedge x.\ x \in X \Longrightarrow g\ x = f\ x$
and *P*: *P f P g*
shows *homotopic-with P X Y f g*
⟨*proof*⟩

lemma *image-Pair-const*: $(\lambda x.\ (x,\ c))\ `\ A = A \times \{c\}$
⟨*proof*⟩

lemma *homotopic-constant-maps*:
homotopic-with $(\lambda x.\ True)\ s\ t\ (\lambda x.\ a)\ (\lambda x.\ b) \longleftrightarrow s = \{\} \vee$ *path-component t a b*
⟨*proof*⟩

25.1 Trivial properties.

lemma *homotopic-with-imp-property*: *homotopic-with P X Y f g* $\Longrightarrow$ *P f* $\wedge$ *P g*
⟨*proof*⟩

lemma *continuous-on-o-Pair*: $[\![$*continuous-on* $(T \times X)\ h$; $t \in T]\!] \Longrightarrow$ *continuous-on X* $(h$ *o Pair t*$)$
⟨*proof*⟩

lemma *homotopic-with-imp-continuous*:
assumes *homotopic-with P X Y f g*
shows *continuous-on X f* $\wedge$ *continuous-on X g*
⟨*proof*⟩

proposition *homotopic-with-imp-subset1*:
homotopic-with P X Y f g $\Longrightarrow f\ `\ X \subseteq Y$
⟨*proof*⟩

proposition *homotopic-with-imp-subset2*:
homotopic-with P X Y f g $\Longrightarrow g\ `\ X \subseteq Y$
⟨*proof*⟩

proposition *homotopic-with-mono*:
assumes *hom*: *homotopic-with P X Y f g*
and *Q*: $\bigwedge h.$ ⟦*continuous-on X h*; *image h X* ⊆ *Y* ∧ *P h*⟧ ⟹ *Q h*
shows *homotopic-with Q X Y f g*
⟨*proof*⟩

proposition *homotopic-with-subset-left*:
⟦*homotopic-with P X Y f g*; *Z* ⊆ *X*⟧ ⟹ *homotopic-with P Z Y f g*
⟨*proof*⟩

proposition *homotopic-with-subset-right*:
⟦*homotopic-with P X Y f g*; *Y* ⊆ *Z*⟧ ⟹ *homotopic-with P X Z f g*
⟨*proof*⟩

proposition *homotopic-with-compose-continuous-right*:
⟦*homotopic-with* (λ*f*. *p* (*f* ∘ *h*)) *X Y f g*; *continuous-on W h*; *h* ‘ *W* ⊆ *X*⟧
⟹ *homotopic-with p W Y* (*f o h*) (*g o h*)
⟨*proof*⟩

proposition *homotopic-compose-continuous-right*:
⟦*homotopic-with* (λ*f*. *True*) *X Y f g*; *continuous-on W h*; *h* ‘ *W* ⊆ *X*⟧
⟹ *homotopic-with* (λ*f*. *True*) *W Y* (*f o h*) (*g o h*)
⟨*proof*⟩

proposition *homotopic-with-compose-continuous-left*:
⟦*homotopic-with* (λ*f*. *p* (*h* ∘ *f*)) *X Y f g*; *continuous-on Y h*; *h* ‘ *Y* ⊆ *Z*⟧
⟹ *homotopic-with p X Z* (*h o f*) (*h o g*)
⟨*proof*⟩

proposition *homotopic-compose-continuous-left*:
⟦*homotopic-with* (λ-. *True*) *X Y f g*;
continuous-on Y h; *h* ‘ *Y* ⊆ *Z*⟧
⟹ *homotopic-with* (λ*f*. *True*) *X Z* (*h o f*) (*h o g*)
⟨*proof*⟩

proposition *homotopic-with-Pair*:
assumes *hom*: *homotopic-with p s t f g homotopic-with p′ s′ t′ f′ g′*
and *q*: $\bigwedge f\ g.$ ⟦*p f*; *p′ g*⟧ ⟹ *q*(λ(*x*,*y*). (*f x*, *g y*))
shows *homotopic-with q* (*s* × *s′*) (*t* × *t′*)
(λ(*x*,*y*). (*f x*, *f′ y*)) (λ(*x*,*y*). (*g x*, *g′ y*))
⟨*proof*⟩

lemma *homotopic-on-empty* [*simp*]: *homotopic-with* (λ*x*. *True*) {} *t f g*
⟨*proof*⟩

Homotopy with P is an equivalence relation (on continuous functions mapping X into Y that satisfy P, though this only affects reflexivity.

proposition *homotopic-with-refl*:

homotopic-with P X Y f f ⟷ *continuous-on X f* ∧ *image f X* ⊆ *Y* ∧ *P f*
⟨*proof*⟩

lemma *homotopic-with-symD*:
fixes *X* :: *'a*::*real-normed-vector set*
assumes *homotopic-with P X Y f g*
shows *homotopic-with P X Y g f*
⟨*proof*⟩

proposition *homotopic-with-sym*:
fixes *X* :: *'a*::*real-normed-vector set*
shows *homotopic-with P X Y f g* ⟷ *homotopic-with P X Y g f*
⟨*proof*⟩

lemma *split-01*: {*0..1*::*real*} = {*0..1/2*} ∪ {*1/2..1*}
⟨*proof*⟩

lemma *split-01-prod*: {*0..1*::*real*} × *X* = ({*0..1/2*} × *X*) ∪ ({*1/2..1*} × *X*)
⟨*proof*⟩

proposition *homotopic-with-trans*:
fixes *X* :: *'a*::*real-normed-vector set*
assumes *homotopic-with P X Y f g* **and** *homotopic-with P X Y g h*
shows *homotopic-with P X Y f h*
⟨*proof*⟩

proposition *homotopic-compose*:
fixes *s* :: *'a*::*real-normed-vector set*
shows ⟦*homotopic-with* (λ*x*. *True*) *s t f f'*; *homotopic-with* (λ*x*. *True*) *t u g g'*⟧
⟹ *homotopic-with* (λ*x*. *True*) *s u* (*g o f*) (*g' o f'*)
⟨*proof*⟩

25.2 Homotopy of paths, maintaining the same endpoints.

definition *homotopic-paths* :: [*'a set*, *real* ⇒ *'a*, *real* ⇒ *'a*::*topological-space*] ⇒ *bool*
where
homotopic-paths s p q ≡
homotopic-with (λ*r*. *pathstart r* = *pathstart p* ∧ *pathfinish r* = *pathfinish p*) {*0..1*} *s p q*

lemma *homotopic-paths*:
homotopic-paths s p q ⟷
(∃ *h*. *continuous-on* ({*0..1*} × {*0..1*}) *h* ∧
h ' ({*0..1*} × {*0..1*}) ⊆ *s* ∧
(∀ *x* ∈ {*0..1*}. *h*(*0*,*x*) = *p x*) ∧
(∀ *x* ∈ {*0..1*}. *h*(*1*,*x*) = *q x*) ∧
(∀ *t* ∈ {*0..1*::*real*}. *pathstart*(*h o Pair t*) = *pathstart p* ∧

pathfinish(*h o Pair t*) = *pathfinish p*))
⟨*proof*⟩

proposition *homotopic-paths-imp-pathstart*:
homotopic-paths s p q ⟹ *pathstart p* = *pathstart q*
⟨*proof*⟩

proposition *homotopic-paths-imp-pathfinish*:
homotopic-paths s p q ⟹ *pathfinish p* = *pathfinish q*
⟨*proof*⟩

lemma *homotopic-paths-imp-path*:
homotopic-paths s p q ⟹ *path p* ∧ *path q*
⟨*proof*⟩

lemma *homotopic-paths-imp-subset*:
homotopic-paths s p q ⟹ *path-image p* ⊆ *s* ∧ *path-image q* ⊆ *s*
⟨*proof*⟩

proposition *homotopic-paths-refl* [*simp*]: *homotopic-paths s p p* ⟷ *path p* ∧ *path-image p* ⊆ *s*
⟨*proof*⟩

proposition *homotopic-paths-sym*: *homotopic-paths s p q* ⟹ *homotopic-paths s q p*
⟨*proof*⟩

proposition *homotopic-paths-sym-eq*: *homotopic-paths s p q* ⟷ *homotopic-paths s q p*
⟨*proof*⟩

proposition *homotopic-paths-trans* [*trans*]:
⟦*homotopic-paths s p q*; *homotopic-paths s q r*⟧ ⟹ *homotopic-paths s p r*
⟨*proof*⟩

proposition *homotopic-paths-eq*:
⟦*path p*; *path-image p* ⊆ *s*; ⋀*t*. *t* ∈ {*0..1*} ⟹ *p t* = *q t*⟧ ⟹ *homotopic-paths s p q*
⟨*proof*⟩

proposition *homotopic-paths-reparametrize*:
assumes *path p*
and *pips*: *path-image p* ⊆ *s*
and *contf*: *continuous-on* {*0..1*} *f*
and *f01*:*f* ‘ {*0..1*} ⊆ {*0..1*}
and [*simp*]: *f*(*0*) = *0 f*(*1*) = *1*
and *q*: ⋀*t*. *t* ∈ {*0..1*} ⟹ *q*(*t*) = *p*(*f t*)
shows *homotopic-paths s p q*
⟨*proof*⟩

lemma *homotopic-paths-subset*: $[\![homotopic\text{-}paths\ s\ p\ q;\ s \subseteq t]\!] \Longrightarrow homotopic\text{-}paths\ t\ p\ q$
⟨*proof*⟩

A slightly ad-hoc but useful lemma in constructing homotopies.

lemma *homotopic-join-lemma*:
fixes $q :: [real,real] \Rightarrow 'a::topological\text{-}space$
assumes p: *continuous-on* $(\{0..1\} \times \{0..1\})\ (\lambda y.\ p\ (fst\ y)\ (snd\ y))$
and q: *continuous-on* $(\{0..1\} \times \{0..1\})\ (\lambda y.\ q\ (fst\ y)\ (snd\ y))$
and pf: $\bigwedge t.\ t \in \{0..1\} \Longrightarrow pathfinish(p\ t) = pathstart(q\ t)$
shows *continuous-on* $(\{0..1\} \times \{0..1\})\ (\lambda y.\ (p(fst\ y) +\!\!+\!\!+ q(fst\ y))\ (snd\ y))$
⟨*proof*⟩

Congruence properties of homotopy w.r.t. path-combining operations.

lemma *homotopic-paths-reversepath-D*:
assumes *homotopic-paths s p q*
shows *homotopic-paths s* (*reversepath p*) (*reversepath q*)
⟨*proof*⟩

proposition *homotopic-paths-reversepath*:
homotopic-paths s (*reversepath p*) (*reversepath q*) $\longleftrightarrow$ *homotopic-paths s p q*
⟨*proof*⟩

proposition *homotopic-paths-join*:
$[\![homotopic\text{-}paths\ s\ p\ p';\ homotopic\text{-}paths\ s\ q\ q';\ pathfinish\ p = pathstart\ q]\!] \Longrightarrow homotopic\text{-}paths\ s\ (p +\!\!+\!\!+ q)\ (p' +\!\!+\!\!+ q')$
⟨*proof*⟩

proposition *homotopic-paths-continuous-image*:
$[\![homotopic\text{-}paths\ s\ f\ g;\ continuous\text{-}on\ s\ h;\ h\ `\ s \subseteq t]\!] \Longrightarrow homotopic\text{-}paths\ t\ (h\ o\ f)\ (h\ o\ g)$
⟨*proof*⟩

25.3 Group properties for homotopy of paths

So taking equivalence classes under homotopy would give the fundamental group

proposition *homotopic-paths-rid*:
$[\![path\ p;\ path\text{-}image\ p \subseteq s]\!] \Longrightarrow homotopic\text{-}paths\ s\ (p +\!\!+\!\!+ linepath\ (pathfinish\ p)\ (pathfinish\ p))\ p$
⟨*proof*⟩

proposition *homotopic-paths-lid*:
$[\![path\ p;\ path\text{-}image\ p \subseteq s]\!] \Longrightarrow homotopic\text{-}paths\ s\ (linepath\ (pathstart\ p)\ (pathstart\ p) +\!\!+\!\!+ p)\ p$
⟨*proof*⟩

proposition *homotopic-paths-assoc*:
⟦*path p*; *path-image p* ⊆ *s*; *path q*; *path-image q* ⊆ *s*; *path r*; *path-image r* ⊆ *s*; *pathfinish p* = *pathstart q*;
pathfinish q = *pathstart r*⟧
⟹ *homotopic-paths s* (*p* +++ (*q* +++ *r*)) ((*p* +++ *q*) +++ *r*)
⟨*proof*⟩

proposition *homotopic-paths-rinv*:
assumes *path p path-image p* ⊆ *s*
shows *homotopic-paths s* (*p* +++ *reversepath p*) (*linepath* (*pathstart p*) (*pathstart p*))
⟨*proof*⟩

proposition *homotopic-paths-linv*:
assumes *path p path-image p* ⊆ *s*
shows *homotopic-paths s* (*reversepath p* +++ *p*) (*linepath* (*pathfinish p*) (*pathfinish p*))
⟨*proof*⟩

25.4 Homotopy of loops without requiring preservation of endpoints.

definition *homotopic-loops* :: *'a*::*topological-space set* ⇒ (*real* ⇒ *'a*) ⇒ (*real* ⇒ *'a*) ⇒ *bool* **where**
homotopic-loops s p q ≡
homotopic-with (λ*r*. *pathfinish r* = *pathstart r*) {*0..1*} *s p q*

lemma *homotopic-loops*:
homotopic-loops s p q ⟷
(∃*h*. *continuous-on* ({*0..1*::*real*} × {*0..1*}) *h* ∧
image h ({*0..1*} × {*0..1*}) ⊆ *s* ∧
(∀*x* ∈ {*0..1*}. *h*(*0*,*x*) = *p x*) ∧
(∀*x* ∈ {*0..1*}. *h*(*1*,*x*) = *q x*) ∧
(∀*t* ∈ {*0..1*}. *pathfinish*(*h o Pair t*) = *pathstart*(*h o Pair t*)))
⟨*proof*⟩

proposition *homotopic-loops-imp-loop*:
homotopic-loops s p q ⟹ *pathfinish p* = *pathstart p* ∧ *pathfinish q* = *pathstart q*
⟨*proof*⟩

proposition *homotopic-loops-imp-path*:
homotopic-loops s p q ⟹ *path p* ∧ *path q*
⟨*proof*⟩

proposition *homotopic-loops-imp-subset*:
homotopic-loops s p q ⟹ *path-image p* ⊆ *s* ∧ *path-image q* ⊆ *s*
⟨*proof*⟩

proposition *homotopic-loops-refl*:
  homotopic-loops s p p ⟷
   path p ∧ *path-image p* ⊆ *s* ∧ *pathfinish p* = *pathstart p*
⟨*proof*⟩

proposition *homotopic-loops-sym*: *homotopic-loops s p q* ⟹ *homotopic-loops s q p*
⟨*proof*⟩

proposition *homotopic-loops-sym-eq*: *homotopic-loops s p q* ⟷ *homotopic-loops s q p*
⟨*proof*⟩

proposition *homotopic-loops-trans*:
  ⟦*homotopic-loops s p q*; *homotopic-loops s q r*⟧ ⟹ *homotopic-loops s p r*
⟨*proof*⟩

proposition *homotopic-loops-subset*:
  ⟦*homotopic-loops s p q*; *s* ⊆ *t*⟧ ⟹ *homotopic-loops t p q*
⟨*proof*⟩

proposition *homotopic-loops-eq*:
  ⟦*path p*; *path-image p* ⊆ *s*; *pathfinish p* = *pathstart p*; ⋀*t*. *t* ∈ {*0..1*} ⟹ *p*(*t*) = *q*(*t*)⟧
    ⟹ *homotopic-loops s p q*
⟨*proof*⟩

proposition *homotopic-loops-continuous-image*:
  ⟦*homotopic-loops s f g*; *continuous-on s h*; *h* ‘ *s* ⊆ *t*⟧ ⟹ *homotopic-loops t* (*h* ∘ *f*) (*h* ∘ *g*)
⟨*proof*⟩

25.5 Relations between the two variants of homotopy

proposition *homotopic-paths-imp-homotopic-loops*:
  ⟦*homotopic-paths s p q*; *pathfinish p* = *pathstart p*; *pathfinish q* = *pathstart p*⟧
⟹ *homotopic-loops s p q*
⟨*proof*⟩

proposition *homotopic-loops-imp-homotopic-paths-null*:
 assumes *homotopic-loops s p* (*linepath a a*)
  shows *homotopic-paths s p* (*linepath* (*pathstart p*) (*pathstart p*))
⟨*proof*⟩

proposition *homotopic-loops-conjugate*:
 fixes *s* :: *'a*::*real-normed-vector set*
 assumes *path p path q* **and** *pip*: *path-image p* ⊆ *s* **and** *piq*: *path-image q* ⊆ *s*
    and *papp*: *pathfinish p* = *pathstart q* **and** *qloop*: *pathfinish q* = *pathstart q*

shows *homotopic-loops s* (*p +++ q +++ reversepath p*) *q*
⟨*proof*⟩

25.6 Homotopy of "nearby" function, paths and loops.

lemma *homotopic-with-linear*:
fixes *f g* :: - ⇒ *'b*::*real-normed-vector*
assumes *contf*: *continuous-on s f*
and *contg*:*continuous-on s g*
and *sub*: ⋀*x*. *x* ∈ *s* ⟹ *closed-segment* (*f x*) (*g x*) ⊆ *t*
shows *homotopic-with* (λ*z*. *True*) *s t f g*
⟨*proof*⟩

lemma *homotopic-paths-linear*:
fixes *g h* :: *real* ⇒ *'a*::*real-normed-vector*
assumes *path g path h pathstart h* = *pathstart g pathfinish h* = *pathfinish g*
⋀*t x*. *t* ∈ {*0..1*} ⟹ *closed-segment* (*g t*) (*h t*) ⊆ *s*
shows *homotopic-paths s g h*
⟨*proof*⟩

lemma *homotopic-loops-linear*:
fixes *g h* :: *real* ⇒ *'a*::*real-normed-vector*
assumes *path g path h pathfinish g* = *pathstart g pathfinish h* = *pathstart h*
⋀*t x*. *t* ∈ {*0..1*} ⟹ *closed-segment* (*g t*) (*h t*) ⊆ *s*
shows *homotopic-loops s g h*
⟨*proof*⟩

lemma *homotopic-paths-nearby-explicit*:
assumes *path g path h pathstart h* = *pathstart g pathfinish h* = *pathfinish g*
and *no*: ⋀*t x*. ⟦*t* ∈ {*0..1*}; *x* ∉ *s*⟧ ⟹ *norm*(*h t* − *g t*) < *norm*(*g t* − *x*)
shows *homotopic-paths s g h*
⟨*proof*⟩

lemma *homotopic-loops-nearby-explicit*:
assumes *path g path h pathfinish g* = *pathstart g pathfinish h* = *pathstart h*
and *no*: ⋀*t x*. ⟦*t* ∈ {*0..1*}; *x* ∉ *s*⟧ ⟹ *norm*(*h t* − *g t*) < *norm*(*g t* − *x*)
shows *homotopic-loops s g h*
⟨*proof*⟩

lemma *homotopic-nearby-paths*:
fixes *g h* :: *real* ⇒ *'a*::*euclidean-space*
assumes *path g open s path-image g* ⊆ *s*
shows ∃ *e*. *0* < *e* ∧
(∀ *h*. *path h* ∧
pathstart h = *pathstart g* ∧ *pathfinish h* = *pathfinish g* ∧
(∀ *t* ∈ {*0..1*}. *norm*(*h t* − *g t*) < *e*) ⟶ *homotopic-paths s g h*)
⟨*proof*⟩

lemma *homotopic-nearby-loops*:

fixes *g h* :: *real* ⇒ *'a*::*euclidean-space*
assumes *path g open s path-image g* ⊆ *s pathfinish g* = *pathstart g*
shows ∃ *e*. *0* < *e* ∧
(∀ *h*. *path h* ∧ *pathfinish h* = *pathstart h* ∧
(∀ *t* ∈ {*0..1*}. *norm*(*h t* − *g t*) < *e*) ⟶ *homotopic-loops s g h*)
⟨*proof*⟩

25.7 Homotopy and subpaths

lemma *homotopic-join-subpaths1*:
assumes *path g* **and** *pag*: *path-image g* ⊆ *s*
and *u*: *u* ∈ {*0..1*} **and** *v*: *v* ∈ {*0..1*} **and** *w*: *w* ∈ {*0..1*} *u* ≤ *v v* ≤ *w*
shows *homotopic-paths s* (*subpath u v g* +++ *subpath v w g*) (*subpath u w g*)
⟨*proof*⟩

lemma *homotopic-join-subpaths2*:
assumes *homotopic-paths s* (*subpath u v g* +++ *subpath v w g*) (*subpath u w g*)
shows *homotopic-paths s* (*subpath w v g* +++ *subpath v u g*) (*subpath w u g*)
⟨*proof*⟩

lemma *homotopic-join-subpaths3*:
assumes *hom*: *homotopic-paths s* (*subpath u v g* +++ *subpath v w g*) (*subpath u w g*)
and *path g* **and** *pag*: *path-image g* ⊆ *s*
and *u*: *u* ∈ {*0..1*} **and** *v*: *v* ∈ {*0..1*} **and** *w*: *w* ∈ {*0..1*}
shows *homotopic-paths s* (*subpath v w g* +++ *subpath w u g*) (*subpath v u g*)
⟨*proof*⟩

proposition *homotopic-join-subpaths*:
⟦*path g*; *path-image g* ⊆ *s*; *u* ∈ {*0..1*}; *v* ∈ {*0..1*}; *w* ∈ {*0..1*}⟧
⟹ *homotopic-paths s* (*subpath u v g* +++ *subpath v w g*) (*subpath u w g*)
⟨*proof*⟩

Relating homotopy of trivial loops to path-connectedness.

lemma *path-component-imp-homotopic-points*:
path-component S a b ⟹ *homotopic-loops S* (*linepath a a*) (*linepath b b*)
⟨*proof*⟩

lemma *homotopic-loops-imp-path-component-value*:
⟦*homotopic-loops S p q*; *0* ≤ *t*; *t* ≤ *1*⟧
⟹ *path-component S* (*p t*) (*q t*)
⟨*proof*⟩

lemma *homotopic-points-eq-path-component*:
homotopic-loops S (*linepath a a*) (*linepath b b*) ⟷
path-component S a b
⟨*proof*⟩

lemma *path-connected-eq-homotopic-points*:

path-connected S ⟷
(∀ *a b*. *a* ∈ *S* ∧ *b* ∈ *S* ⟶ *homotopic-loops S* (*linepath a a*) (*linepath b b*))
⟨*proof*⟩

25.8 Simply connected sets

defined as "all loops are homotopic (as loops)

definition *simply-connected* **where**
simply-connected S ≡
∀ *p q*. *path p* ∧ *pathfinish p* = *pathstart p* ∧ *path-image p* ⊆ *S* ∧
path q ∧ *pathfinish q* = *pathstart q* ∧ *path-image q* ⊆ *S*
⟶ *homotopic-loops S p q*

lemma *simply-connected-empty* [*iff*]: *simply-connected* {}
⟨*proof*⟩

lemma *simply-connected-imp-path-connected*:
fixes *S* :: -::*real-normed-vector set*
shows *simply-connected S* ⟹ *path-connected S*
⟨*proof*⟩

lemma *simply-connected-imp-connected*:
fixes *S* :: -::*real-normed-vector set*
shows *simply-connected S* ⟹ *connected S*
⟨*proof*⟩

lemma *simply-connected-eq-contractible-loop-any*:
fixes *S* :: -::*real-normed-vector set*
shows *simply-connected S* ⟷
(∀ *p a*. *path p* ∧ *path-image p* ⊆ *S* ∧
pathfinish p = *pathstart p* ∧ *a* ∈ *S*
⟶ *homotopic-loops S p* (*linepath a a*))
⟨*proof*⟩

lemma *simply-connected-eq-contractible-loop-some*:
fixes *S* :: -::*real-normed-vector set*
shows *simply-connected S* ⟷
path-connected S ∧
(∀ *p*. *path p* ∧ *path-image p* ⊆ *S* ∧ *pathfinish p* = *pathstart p*
⟶ (∃ *a*. *a* ∈ *S* ∧ *homotopic-loops S p* (*linepath a a*)))
⟨*proof*⟩

lemma *simply-connected-eq-contractible-loop-all*:
fixes *S* :: -::*real-normed-vector set*
shows *simply-connected S* ⟷
S = {} ∨
(∃ *a* ∈ *S*. ∀ *p*. *path p* ∧ *path-image p* ⊆ *S* ∧ *pathfinish p* = *pathstart p*
⟶ *homotopic-loops S p* (*linepath a a*))
(**is** *?lhs* = *?rhs*)

⟨*proof*⟩

lemma *simply-connected-eq-contractible-path*:
fixes S :: -::*real-normed-vector set*
shows *simply-connected* $S \longleftrightarrow$
path-connected $S \wedge$
$(\forall p.$ *path* $p \wedge$ *path-image* $p \subseteq S \wedge$ *pathfinish* $p =$ *pathstart* p
$\longrightarrow$ *homotopic-paths* S p (*linepath* (*pathstart* p) (*pathstart* p)))
⟨*proof*⟩

lemma *simply-connected-eq-homotopic-paths*:
fixes S :: -::*real-normed-vector set*
shows *simply-connected* $S \longleftrightarrow$
path-connected $S \wedge$
$(\forall p\ q.$ *path* $p \wedge$ *path-image* $p \subseteq S \wedge$
path $q \wedge$ *path-image* $q \subseteq S \wedge$
pathstart $q =$ *pathstart* $p \wedge$ *pathfinish* $q =$ *pathfinish* p
$\longrightarrow$ *homotopic-paths* S p q)
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

proposition *simply-connected-Times*:
fixes S :: *'a*::*real-normed-vector set* **and** T :: *'b*::*real-normed-vector set*
assumes S: *simply-connected* S **and** T: *simply-connected* T
shows *simply-connected*($S \times T$)
⟨*proof*⟩

25.9 Contractible sets

definition *contractible* **where**
contractible $S \equiv \exists a.$ *homotopic-with* $(\lambda x.\ True)$ S S *id* $(\lambda x.\ a)$

proposition *contractible-imp-simply-connected*:
fixes S :: -::*real-normed-vector set*
assumes *contractible* S **shows** *simply-connected* S
⟨*proof*⟩

corollary *contractible-imp-connected*:
fixes S :: -::*real-normed-vector set*
shows *contractible* $S \Longrightarrow$ *connected* S
⟨*proof*⟩

lemma *contractible-imp-path-connected*:
fixes S :: -::*real-normed-vector set*
shows *contractible* $S \Longrightarrow$ *path-connected* S
⟨*proof*⟩

lemma *nullhomotopic-through-contractible*:
fixes S :: -::*topological-space set*

assumes *f*: *continuous-on S f f ' S ⊆ T*
and *g*: *continuous-on T g g ' T ⊆ U*
and *T*: *contractible T*
obtains *c* **where** *homotopic-with* (λ*h*. *True*) *S U* (*g o f*) (λ*x*. *c*)
⟨*proof*⟩

lemma *nullhomotopic-into-contractible*:
assumes *f*: *continuous-on S f f ' S ⊆ T*
and *T*: *contractible T*
obtains *c* **where** *homotopic-with* (λ*h*. *True*) *S T f* (λ*x*. *c*)
⟨*proof*⟩

lemma *nullhomotopic-from-contractible*:
assumes *f*: *continuous-on S f f ' S ⊆ T*
and *S*: *contractible S*
obtains *c* **where** *homotopic-with* (λ*h*. *True*) *S T f* (λ*x*. *c*)
⟨*proof*⟩

lemma *homotopic-through-contractible*:
fixes *S* :: -::*real-normed-vector set*
assumes *continuous-on S f1 f1 ' S ⊆ T*
continuous-on T g1 g1 ' T ⊆ U
continuous-on S f2 f2 ' S ⊆ T
continuous-on T g2 g2 ' T ⊆ U
contractible T path-connected U
shows *homotopic-with* (λ*h*. *True*) *S U* (*g1 o f1*) (*g2 o f2*)
⟨*proof*⟩

lemma *homotopic-into-contractible*:
fixes *S* :: *'a*::*real-normed-vector set* **and** *T*:: *'b*::*real-normed-vector set*
assumes *f*: *continuous-on S f f ' S ⊆ T*
and *g*: *continuous-on S g g ' S ⊆ T*
and *T*: *contractible T*
shows *homotopic-with* (λ*h*. *True*) *S T f g*
⟨*proof*⟩

lemma *homotopic-from-contractible*:
fixes *S* :: *'a*::*real-normed-vector set* **and** *T*:: *'b*::*real-normed-vector set*
assumes *f*: *continuous-on S f f ' S ⊆ T*
and *g*: *continuous-on S g g ' S ⊆ T*
and *contractible S path-connected T*
shows *homotopic-with* (λ*h*. *True*) *S T f g*
⟨*proof*⟩

lemma *starlike-imp-contractible-gen*:
fixes *S* :: *'a*::*real-normed-vector set*
assumes *S*: *starlike S*
and *P*: ⋀*a T*. ⟦*a* ∈ *S*; *0* ≤ *T*; *T* ≤ *1*⟧ ⟹ *P*(λ*x*. (*1* − *T*) *∗R x* + *T ∗R a*)
obtains *a* **where** *homotopic-with P S S* (λ*x*. *x*) (λ*x*. *a*)

⟨*proof*⟩

lemma *starlike-imp-contractible*:
fixes S :: *'a*::*real-normed-vector set*
shows *starlike* $S \Longrightarrow$ *contractible* S
⟨*proof*⟩

lemma *contractible-UNIV*: *contractible* (*UNIV* :: *'a*::*real-normed-vector set*)
⟨*proof*⟩

lemma *starlike-imp-simply-connected*:
fixes S :: *'a*::*real-normed-vector set*
shows *starlike* $S \Longrightarrow$ *simply-connected* S
⟨*proof*⟩

lemma *convex-imp-simply-connected*:
fixes S :: *'a*::*real-normed-vector set*
shows *convex* $S \Longrightarrow$ *simply-connected* S
⟨*proof*⟩

lemma *starlike-imp-path-connected*:
fixes S :: *'a*::*real-normed-vector set*
shows *starlike* $S \Longrightarrow$ *path-connected* S
⟨*proof*⟩

lemma *starlike-imp-connected*:
fixes S :: *'a*::*real-normed-vector set*
shows *starlike* $S \Longrightarrow$ *connected* S
⟨*proof*⟩

lemma *is-interval-simply-connected-1*:
fixes S :: *real set*
shows *is-interval* $S \longleftrightarrow$ *simply-connected* S
⟨*proof*⟩

lemma *contractible-empty*: *contractible* {}
⟨*proof*⟩

lemma *contractible-convex-tweak-boundary-points*:
fixes S :: *'a*::*euclidean-space set*
assumes *convex* S **and** *TS*: *rel-interior* $S \subseteq T$ $T \subseteq$ *closure* S
shows *contractible* T
⟨*proof*⟩

lemma *convex-imp-contractible*:
fixes S :: *'a*::*real-normed-vector set*
shows *convex* $S \Longrightarrow$ *contractible* S
⟨*proof*⟩

lemma *contractible-sing*:
fixes a :: *'a::real-normed-vector*
shows *contractible* $\{a\}$
⟨*proof*⟩

lemma *is-interval-contractible-1*:
fixes S :: *real set*
shows *is-interval* $S \longleftrightarrow$ *contractible* S
⟨*proof*⟩

lemma *contractible-times*:
fixes S :: *'a::euclidean-space set* **and** T :: *'b::euclidean-space set*
assumes S: *contractible* S **and** T: *contractible* T
shows *contractible* $(S \times T)$
⟨*proof*⟩

lemma *homotopy-dominated-contractibility*:
fixes S :: *'a::real-normed-vector set* **and** T :: *'b::real-normed-vector set*
assumes S: *contractible* S
and f: *continuous-on* S f *image* f $S \subseteq T$
and g: *continuous-on* T g *image* g $T \subseteq S$
and *hom*: *homotopic-with* $(\lambda x.\ True)$ T T $(f\ o\ g)$ *id*
shows *contractible* T
⟨*proof*⟩

25.10 Local versions of topological properties in general

definition *locally* :: $('a$::*topological-space set* $\Rightarrow$ *bool*$) \Rightarrow$ $'a$ *set* $\Rightarrow$ *bool*
where
locally P $S \equiv$
$\forall w\ x.$ *openin* (*subtopology euclidean* S) $w \wedge x \in w$
$\longrightarrow (\exists u\ v.$ *openin* (*subtopology euclidean* S) $u \wedge P\ v \wedge$
$x \in u \wedge u \subseteq v \wedge v \subseteq w)$

lemma *locallyI*:
assumes $\bigwedge w\ x.\ [\![$*openin* (*subtopology euclidean* S) w; $x \in w]\!]$
$\Longrightarrow \exists u\ v.$ *openin* (*subtopology euclidean* S) $u \wedge P\ v \wedge$
$x \in u \wedge u \subseteq v \wedge v \subseteq w$
shows *locally* P S
⟨*proof*⟩

lemma *locallyE*:
assumes *locally* P S *openin* (*subtopology euclidean* S) w $x \in w$
obtains u v **where** *openin* (*subtopology euclidean* S) u
$P\ v$ $x \in u$ $u \subseteq v$ $v \subseteq w$
⟨*proof*⟩

lemma *locally-mono*:
assumes *locally* P S $\bigwedge t.\ P\ t \Longrightarrow Q\ t$

shows *locally Q S*
⟨*proof*⟩

lemma *locally-open-subset*:
assumes *locally P S openin* (*subtopology euclidean S*) *t*
shows *locally P t*
⟨*proof*⟩

lemma *locally-diff-closed*:
⟦*locally P S*; *closedin* (*subtopology euclidean S*) *t*⟧ ⟹ *locally P* (*S* − *t*)
⟨*proof*⟩

lemma *locally-empty* [*iff*]: *locally P* {}
⟨*proof*⟩

lemma *locally-singleton* [*iff*]:
fixes *a* :: *'a*::*metric-space*
shows *locally P* {*a*} ⟷ *P* {*a*}
⟨*proof*⟩

lemma *locally-iff*:
locally P S ⟷
(∀ *T x*. *open T* ∧ *x* ∈ *S* ∩ *T* ⟶ (∃ *U*. *open U* ∧ (∃ *v*. *P v* ∧ *x* ∈ *S* ∩ *U* ∧ *S* ∩ *U* ⊆ *v* ∧ *v* ⊆ *S* ∩ *T*)))
⟨*proof*⟩

lemma *locally-Int*:
assumes *S*: *locally P S* **and** *t*: *locally P t*
and *P*: ⋀*S t*. *P S* ∧ *P t* ⟹ *P*(*S* ∩ *t*)
shows *locally P* (*S* ∩ *t*)
⟨*proof*⟩

proposition *homeomorphism-locally-imp*:
fixes *S* :: *'a*::*metric-space set* **and** *t* :: *'b*::*t2-space set*
assumes *S*: *locally P S* **and** *hom*: *homeomorphism S t f g*
and *Q*: ⋀*S t*. ⟦*P S*; *homeomorphism S t f g*⟧ ⟹ *Q t*
shows *locally Q t*
⟨*proof*⟩

lemma *homeomorphism-locally*:
fixes *f*:: *'a*::*metric-space* ⇒ *'b*::*metric-space*
assumes *hom*: *homeomorphism S t f g*
and *eq*: ⋀*S t*. *homeomorphism S t f g* ⟹ (*P S* ⟷ *Q t*)
shows *locally P S* ⟷ *locally Q t*
⟨*proof*⟩

lemma *locally-translation*:
fixes *P* :: *'a* :: *real-normed-vector set* ⇒ *bool*

shows
$(\bigwedge S.\ P\ (image\ (\lambda x.\ a + x)\ S) \longleftrightarrow P\ S)$
$\Longrightarrow locally\ P\ (image\ (\lambda x.\ a + x)\ S) \longleftrightarrow locally\ P\ S$
⟨*proof*⟩

lemma *locally-injective-linear-image*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes *f*: *linear f inj f* **and** *iff*: $\bigwedge S.\ P\ (f\ `\ S) \longleftrightarrow Q\ S$
shows $locally\ P\ (f\ `\ S) \longleftrightarrow locally\ Q\ S$
⟨*proof*⟩

lemma *locally-open-map-image*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes *P*: *locally P S*
and *f*: *continuous-on S f*
and *oo*: $\bigwedge t.\ openin\ (subtopology\ euclidean\ S)\ t$
$\Longrightarrow openin\ (subtopology\ euclidean\ (f\ `\ S))\ (f\ `\ t)$
and *Q*: $\bigwedge t.\ [\![t \subseteq S; P\ t]\!] \Longrightarrow Q(f\ `\ t)$
shows $locally\ Q\ (f\ `\ S)$
⟨*proof*⟩

25.11 Basic properties of local compactness

lemma *locally-compact*:
fixes $s :: 'a :: metric\text{-}space\ set$
shows
$locally\ compact\ s \longleftrightarrow$
$(\forall x \in s.\ \exists u\ v.\ x \in u \wedge u \subseteq v \wedge v \subseteq s \wedge$
$openin\ (subtopology\ euclidean\ s)\ u \wedge compact\ v)$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *locally-compactE*:
fixes $s :: 'a :: metric\text{-}space\ set$
assumes *locally compact s*
obtains *u v* **where** $\bigwedge x.\ x \in s \Longrightarrow x \in u\ x \wedge u\ x \subseteq v\ x \wedge v\ x \subseteq s \wedge$
$openin\ (subtopology\ euclidean\ s)\ (u\ x) \wedge compact\ (v\ x)$
⟨*proof*⟩

lemma *locally-compact-alt*:
fixes $s :: 'a :: heine\text{-}borel\ set$
shows $locally\ compact\ s \longleftrightarrow$
$(\forall x \in s.\ \exists u.\ x \in u \wedge$
$openin\ (subtopology\ euclidean\ s)\ u \wedge compact(closure\ u) \wedge closure\ u \subseteq s)$
⟨*proof*⟩

lemma *locally-compact-Int-cball*:
fixes $s :: 'a :: heine\text{-}borel\ set$

shows *locally compact s* $\longleftrightarrow$ $(\forall x \in s.\ \exists e.\ 0 < e \wedge closed(cball\ x\ e \cap s))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *locally-compact-compact*:
fixes *s* :: *'a* :: *heine-borel set*
shows *locally compact s* $\longleftrightarrow$
$(\forall k.\ k \subseteq s \wedge compact\ k \longrightarrow (\exists u\ v.\ k \subseteq u \wedge u \subseteq v \wedge v \subseteq s \wedge openin\ (subtopology\ euclidean\ s)\ u \wedge compact\ v))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *open-imp-locally-compact*:
fixes *s* :: *'a* :: *heine-borel set*
assumes *open s*
shows *locally compact s*
⟨*proof*⟩

lemma *closed-imp-locally-compact*:
fixes *s* :: *'a* :: *heine-borel set*
assumes *closed s*
shows *locally compact s*
⟨*proof*⟩

lemma *locally-compact-UNIV*: *locally compact* (*UNIV* :: *'a* :: *heine-borel set*)
⟨*proof*⟩

lemma *locally-compact-Int*:
fixes *s* :: *'a* :: *t2-space set*
shows ⟦*locally compact s*; *locally compact t*⟧ ⟹ *locally compact* $(s \cap t)$
⟨*proof*⟩

lemma *locally-compact-closedin*:
fixes *s* :: *'a* :: *heine-borel set*
shows ⟦*closedin* (*subtopology euclidean s*) *t*; *locally compact s*⟧
⟹ *locally compact t*
⟨*proof*⟩

lemma *locally-compact-delete*:
fixes *s* :: *'a* :: *t1-space set*
shows *locally compact s* ⟹ *locally compact* $(s - \{a\})$
⟨*proof*⟩

lemma *locally-closed*:
fixes *s* :: *'a* :: *heine-borel set*
shows *locally closed s* $\longleftrightarrow$ *locally compact s*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

25.12 Important special cases of local connectedness and path connectedness

lemma *locally-connected-1*:
assumes
$\bigwedge v\ x.\ [\![\mathit{openin}\ (\mathit{subtopology}\ \mathit{euclidean}\ S)\ v;\ x \in v]\!]$
$\Longrightarrow \exists u.\ \mathit{openin}\ (\mathit{subtopology}\ \mathit{euclidean}\ S)\ u \wedge$
$\mathit{connected}\ u \wedge x \in u \wedge u \subseteq v$
shows *locally connected S*
⟨*proof*⟩

lemma *locally-connected-2*:
assumes *locally connected S*
openin (*subtopology euclidean S*) *t*
$x \in t$
shows *openin* (*subtopology euclidean S*) (*connected-component-set t x*)
⟨*proof*⟩

lemma *locally-connected-3*:
assumes $\bigwedge t\ x.\ [\![\mathit{openin}\ (\mathit{subtopology}\ \mathit{euclidean}\ S)\ t;\ x \in t]\!]$
$\Longrightarrow$ *openin* (*subtopology euclidean S*)
(*connected-component-set t x*)
openin (*subtopology euclidean S*) *v* $x \in v$
shows $\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$ *connected* $u \wedge x \in u \wedge u \subseteq v$
⟨*proof*⟩

lemma *locally-connected*:
locally connected S $\longleftrightarrow$
$(\forall v\ x.$ *openin* (*subtopology euclidean S*) $v \wedge x \in v$
$\longrightarrow (\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$ *connected* $u \wedge x \in u \wedge u \subseteq v))$
⟨*proof*⟩

lemma *locally-connected-open-connected-component*:
locally connected S $\longleftrightarrow$
$(\forall t\ x.$ *openin* (*subtopology euclidean S*) $t \wedge x \in t$
$\longrightarrow$ *openin* (*subtopology euclidean S*) (*connected-component-set t x*))
⟨*proof*⟩

lemma *locally-path-connected-1*:
assumes
$\bigwedge v\ x.\ [\![\mathit{openin}\ (\mathit{subtopology}\ \mathit{euclidean}\ S)\ v;\ x \in v]\!]$
$\Longrightarrow \exists u.\ \mathit{openin}\ (\mathit{subtopology}\ \mathit{euclidean}\ S)\ u \wedge \mathit{path\text{-}connected}\ u \wedge x \in u \wedge u \subseteq v$
shows *locally path-connected S*
⟨*proof*⟩

lemma *locally-path-connected-2*:
assumes *locally path-connected S*
openin (*subtopology euclidean S*) *t*

$x \in t$
shows *openin* (*subtopology euclidean S*) (*path-component-set t x*)
⟨*proof*⟩

lemma *locally-path-connected-3*:
assumes $\bigwedge t\ x.\ [\![$*openin* (*subtopology euclidean S*) *t*; $x \in t]\!]$
$\Longrightarrow$ *openin* (*subtopology euclidean S*) (*path-component-set t x*)
openin (*subtopology euclidean S*) *v* $x \in v$
shows $\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$ *path-connected* $u \wedge x \in u \wedge u \subseteq v$
⟨*proof*⟩

proposition *locally-path-connected*:
locally path-connected S $\longleftrightarrow$
$(\forall v\ x.$ *openin* (*subtopology euclidean S*) $v \wedge x \in v$
$\longrightarrow (\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$ *path-connected* $u \wedge x \in u \wedge u \subseteq v))$
⟨*proof*⟩

proposition *locally-path-connected-open-path-connected-component*:
locally path-connected S $\longleftrightarrow$
$(\forall t\ x.$ *openin* (*subtopology euclidean S*) $t \wedge x \in t$
$\longrightarrow$ *openin* (*subtopology euclidean S*) (*path-component-set t x*))
⟨*proof*⟩

lemma *locally-connected-open-component*:
locally connected S $\longleftrightarrow$
$(\forall t\ c.$ *openin* (*subtopology euclidean S*) $t \wedge c \in$ *components t*
$\longrightarrow$ *openin* (*subtopology euclidean S*) *c*)
⟨*proof*⟩

proposition *locally-connected-im-kleinen*:
locally connected S $\longleftrightarrow$
$(\forall v\ x.$ *openin* (*subtopology euclidean S*) $v \wedge x \in v$
$\longrightarrow (\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$
$x \in u \wedge u \subseteq v \wedge$
$(\forall y.\ y \in u \longrightarrow (\exists c.$ *connected* $c \wedge c \subseteq v \wedge x \in c \wedge y \in c))))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

proposition *locally-path-connected-im-kleinen*:
locally path-connected S $\longleftrightarrow$
$(\forall v\ x.$ *openin* (*subtopology euclidean S*) $v \wedge x \in v$
$\longrightarrow (\exists u.$ *openin* (*subtopology euclidean S*) $u \wedge$
$x \in u \wedge u \subseteq v \wedge$
$(\forall y.\ y \in u \longrightarrow (\exists p.$ *path* $p \wedge$ *path-image* $p \subseteq v \wedge$
pathstart $p = x \wedge$ *pathfinish* $p = y))))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *locally-path-connected-imp-locally-connected*:
locally path-connected S $\Longrightarrow$ *locally connected S*
⟨*proof*⟩

lemma *locally-connected-components*:
⟦*locally connected S*; *c* ∈ *components S*⟧ $\Longrightarrow$ *locally connected c*
⟨*proof*⟩

lemma *locally-path-connected-components*:
⟦*locally path-connected S*; *c* ∈ *components S*⟧ $\Longrightarrow$ *locally path-connected c*
⟨*proof*⟩

lemma *locally-path-connected-connected-component*:
locally path-connected S $\Longrightarrow$ *locally path-connected* (*connected-component-set S x*)
⟨*proof*⟩

lemma *open-imp-locally-path-connected*:
fixes *S* :: *'a* :: *real-normed-vector set*
shows *open S* $\Longrightarrow$ *locally path-connected S*
⟨*proof*⟩

lemma *open-imp-locally-connected*:
fixes *S* :: *'a* :: *real-normed-vector set*
shows *open S* $\Longrightarrow$ *locally connected S*
⟨*proof*⟩

lemma *locally-path-connected-UNIV*: *locally path-connected* (*UNIV*::*'a* :: *real-normed-vector set*)
⟨*proof*⟩

lemma *locally-connected-UNIV*: *locally connected* (*UNIV*::*'a* :: *real-normed-vector set*)
⟨*proof*⟩

lemma *openin-connected-component-locally-connected*:
locally connected S
$\Longrightarrow$ *openin* (*subtopology euclidean S*) (*connected-component-set S x*)
⟨*proof*⟩

lemma *openin-components-locally-connected*:
⟦*locally connected S*; *c* ∈ *components S*⟧ $\Longrightarrow$ *openin* (*subtopology euclidean S*) *c*
⟨*proof*⟩

lemma *openin-path-component-locally-path-connected*:
locally path-connected S
$\Longrightarrow$ *openin* (*subtopology euclidean S*) (*path-component-set S x*)
⟨*proof*⟩

lemma *closedin-path-component-locally-path-connected*:
  locally path-connected S
    $\Longrightarrow$ *closedin* (*subtopology euclidean S*) (*path-component-set S x*)
⟨*proof*⟩

lemma *convex-imp-locally-path-connected*:
  fixes *S* :: *'a*:: *real-normed-vector set*
  shows *convex S* $\Longrightarrow$ *locally path-connected S*
⟨*proof*⟩

25.13 Retracts, in a general sense, preserve (co)homotopic triviality)

locale *Retracts* =
  fixes *s h t k*
  assumes *conth*: *continuous-on s h*
    and *imh*: *h ' s = t*
    and *contk*: *continuous-on t k*
    and *imk*: *k ' t* $\subseteq$ *s*
    and *idhk*: $\bigwedge y.\ y \in t \Longrightarrow h(k\ y) = y$

begin

lemma *homotopically-trivial-retraction-gen*:
  assumes *P*: $\bigwedge f.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *t*; *Q f*$]\!] \Longrightarrow P(k\ o\ f)$
    and *Q*: $\bigwedge f.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *s*; *P f*$]\!] \Longrightarrow Q(h\ o\ f)$
    and *Qeq*: $\bigwedge h\ k.\ (\bigwedge x.\ x \in u \Longrightarrow h\ x = k\ x) \Longrightarrow Q\ h = Q\ k$
    and *hom*: $\bigwedge f\ g.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *s*; *P f*;
                continuous-on u g; *g ' u* $\subseteq$ *s*; *P g*$]\!]$
                $\Longrightarrow$ *homotopic-with P u s f g*
    and *contf*: *continuous-on u f* **and** *imf*: *f ' u* $\subseteq$ *t* **and** *Qf*: *Q f*
    and *contg*: *continuous-on u g* **and** *img*: *g ' u* $\subseteq$ *t* **and** *Qg*: *Q g*
  shows *homotopic-with Q u t f g*
⟨*proof*⟩

lemma *homotopically-trivial-retraction-null-gen*:
  assumes *P*: $\bigwedge f.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *t*; *Q f*$]\!] \Longrightarrow P(k\ o\ f)$
    and *Q*: $\bigwedge f.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *s*; *P f*$]\!] \Longrightarrow Q(h\ o\ f)$
    and *Qeq*: $\bigwedge h\ k.\ (\bigwedge x.\ x \in u \Longrightarrow h\ x = k\ x) \Longrightarrow Q\ h = Q\ k$
    and *hom*: $\bigwedge f.\ [\![$*continuous-on u f*; *f ' u* $\subseteq$ *s*; *P f*$]\!]$
                $\Longrightarrow \exists c.$ *homotopic-with P u s f* $(\lambda x.\ c)$
    and *contf*: *continuous-on u f* **and** *imf*:*f ' u* $\subseteq$ *t* **and** *Qf*: *Q f*
  obtains *c* **where** *homotopic-with Q u t f* $(\lambda x.\ c)$
⟨*proof*⟩

lemma *cohomotopically-trivial-retraction-gen*:
  assumes *P*: $\bigwedge f.\ [\![$*continuous-on t f*; *f ' t* $\subseteq$ *u*; *Q f*$]\!] \Longrightarrow P(f\ o\ h)$
    and *Q*: $\bigwedge f.\ [\![$*continuous-on s f*; *f ' s* $\subseteq$ *u*; *P f*$]\!] \Longrightarrow Q(f\ o\ k)$

and *Qeq*: $\bigwedge h\ k.\ (\bigwedge x.\ x \in t \Longrightarrow h\ x = k\ x) \Longrightarrow Q\ h = Q\ k$
and *hom*: $\bigwedge f\ g.\ [\![$*continuous-on* $s\ f$; f ‘ $s \subseteq u$; $P\ f$;
continuous-on $s\ g$; g ‘ $s \subseteq u$; $P\ g]\!] \Longrightarrow$ *homotopic-with* $P\ s\ u\ f\ g$
and *contf*: *continuous-on* $t\ f$ **and** *imf*: f ‘ $t \subseteq u$ **and** *Qf*: $Q\ f$
and *contg*: *continuous-on* $t\ g$ **and** *img*: g ‘ $t \subseteq u$ **and** *Qg*: $Q\ g$
shows *homotopic-with* $Q\ t\ u\ f\ g$
⟨*proof*⟩

lemma *cohomotopically-trivial-retraction-null-gen*:
assumes *P*: $\bigwedge f.\ [\![$*continuous-on* $t\ f$; f ‘ $t \subseteq u$; $Q\ f]\!] \Longrightarrow P(f\ o\ h)$
and *Q*: $\bigwedge f.\ [\![$*continuous-on* $s\ f$; f ‘ $s \subseteq u$; $P\ f]\!] \Longrightarrow Q(f\ o\ k)$
and *Qeq*: $\bigwedge h\ k.\ (\bigwedge x.\ x \in t \Longrightarrow h\ x = k\ x) \Longrightarrow Q\ h = Q\ k$
and *hom*: $\bigwedge f\ g.\ [\![$*continuous-on* $s\ f$; f ‘ $s \subseteq u$; $P\ f]\!]$
$\Longrightarrow \exists c.$ *homotopic-with* $P\ s\ u\ f\ (\lambda x.\ c)$
and *contf*: *continuous-on* $t\ f$ **and** *imf*: f ‘ $t \subseteq u$ **and** *Qf*: $Q\ f$
obtains c **where** *homotopic-with* $Q\ t\ u\ f\ (\lambda x.\ c)$
⟨*proof*⟩

end

lemma *simply-connected-retraction-gen*:
shows $[\![$*simply-connected* S; *continuous-on* $S\ h$; h ‘ $S = T$;
continuous-on $T\ k$; k ‘ $T \subseteq S$; $\bigwedge y.\ y \in T \Longrightarrow h(k\ y) = y]\!]$
$\Longrightarrow$ *simply-connected* T
⟨*proof*⟩

lemma *homeomorphic-simply-connected*:
$[\![S$ *homeomorphic* T; *simply-connected* $S]\!] \Longrightarrow$ *simply-connected* T
⟨*proof*⟩

lemma *homeomorphic-simply-connected-eq*:
S *homeomorphic* $T \Longrightarrow$ (*simply-connected* $S \longleftrightarrow$ *simply-connected* T)
⟨*proof*⟩

end

26 Results connected with topological dimension.

theory *Brouwer-Fixpoint*
imports *Path-Connected*
begin

lemma *bij-betw-singleton-eq*:
assumes *f*: *bij-betw* $f\ A\ B$ **and** *g*: *bij-betw* $g\ A\ B$ **and** *a*: $a \in A$
assumes *eq*: $(\bigwedge x.\ x \in A \Longrightarrow x \neq a \Longrightarrow f\ x = g\ x)$
shows $f\ a = g\ a$
⟨*proof*⟩

lemma *swap-image*:
Fun.swap i j f ‘ A = (if i ∈ A then (if j ∈ A then f ‘ A else f ‘ ((A − {i}) ∪ {j}))
else (if j ∈ A then f ‘ ((A − {j}) ∪ {i}) else f ‘ A))
⟨*proof*⟩

lemma *swap-apply1*: *Fun.swap x y f x = f y*
⟨*proof*⟩

lemma *swap-apply2*: *Fun.swap x y f y = f x*
⟨*proof*⟩

lemma *lessThan-empty-iff*: *{..< n::nat} = {} ⟷ n = 0*
⟨*proof*⟩

lemma *Zero-notin-Suc*: *0 ∉ Suc ‘ A*
⟨*proof*⟩

lemma *atMost-Suc-eq-insert-0*: *{.. Suc n} = insert 0 (Suc ‘ {.. n})*
⟨*proof*⟩

lemma *setsum-union-disjoint'*:
assumes *finite A*
and *finite B*
and *A ∩ B = {}*
and *A ∪ B = C*
shows *setsum g C = setsum g A + setsum g B*
⟨*proof*⟩

lemma *pointwise-minimal-pointwise-maximal*:
fixes *s :: (nat ⇒ nat) set*
assumes *finite s*
and *s ≠ {}*
and *∀ x∈s. ∀ y∈s. x ≤ y ∨ y ≤ x*
shows *∃ a∈s. ∀ x∈s. a ≤ x*
and *∃ a∈s. ∀ x∈s. x ≤ a*
⟨*proof*⟩

lemma *brouwer-compactness-lemma*:
fixes *f :: 'a::metric-space ⇒ 'b::real-normed-vector*
assumes *compact s*
and *continuous-on s f*
and *¬ (∃ x∈s. f x = 0)*
obtains *d* **where** *0 < d* **and** *∀ x∈s. d ≤ norm (f x)*
⟨*proof*⟩

lemma *kuhn-labelling-lemma*:
fixes *P Q :: 'a::euclidean-space ⇒ bool*
assumes *∀ x. P x ⟶ P (f x)*

and $\forall x.\ P\ x \longrightarrow (\forall i{\in}Basis.\ Q\ i \longrightarrow 0 \leq x{\cdot}i \wedge x{\cdot}i \leq 1)$
shows $\exists l.\ (\forall x.\forall i{\in}Basis.\ l\ x\ i \leq (1{::}nat)) \wedge$
$(\forall x.\forall i{\in}Basis.\ P\ x \wedge Q\ i \wedge (x{\cdot}i = 0) \longrightarrow (l\ x\ i = 0)) \wedge$
$(\forall x.\forall i{\in}Basis.\ P\ x \wedge Q\ i \wedge (x{\cdot}i = 1) \longrightarrow (l\ x\ i = 1)) \wedge$
$(\forall x.\forall i{\in}Basis.\ P\ x \wedge Q\ i \wedge (l\ x\ i = 0) \longrightarrow x{\cdot}i \leq f\ x{\cdot}i) \wedge$
$(\forall x.\forall i{\in}Basis.\ P\ x \wedge Q\ i \wedge (l\ x\ i = 1) \longrightarrow f\ x{\cdot}i \leq x{\cdot}i)$
⟨*proof*⟩

26.1 The key "counting" observation, somewhat abstracted.

lemma *kuhn-counting-lemma*:
fixes *bnd compo compo' face S F*
defines $nF\ s == card\ \{f{\in}F.\ face\ f\ s \wedge compo'\ f\}$
assumes [*simp*, *intro*]: *finite F* — faces **and** [*simp*, *intro*]: *finite S* — simplices
and $\bigwedge f.\ f \in F \Longrightarrow bnd\ f \Longrightarrow card\ \{s{\in}S.\ face\ f\ s\} = 1$
and $\bigwedge f.\ f \in F \Longrightarrow \neg\ bnd\ f \Longrightarrow card\ \{s{\in}S.\ face\ f\ s\} = 2$
and $\bigwedge s.\ s \in S \Longrightarrow compo\ s \Longrightarrow nF\ s = 1$
and $\bigwedge s.\ s \in S \Longrightarrow \neg\ compo\ s \Longrightarrow nF\ s = 0 \vee nF\ s = 2$
and $odd\ (card\ \{f{\in}F.\ compo'\ f \wedge bnd\ f\})$
shows $odd\ (card\ \{s{\in}S.\ compo\ s\})$
⟨*proof*⟩

26.2 The odd/even result for faces of complete vertices, generalized.

lemma *kuhn-complete-lemma*:
assumes [*simp*]: *finite simplices*
and *face*: $\bigwedge f\ s.\ face\ f\ s \longleftrightarrow (\exists a{\in}s.\ f = s - \{a\})$
and *card-s*[*simp*]: $\bigwedge s.\ s \in simplices \Longrightarrow card\ s = n + 2$
and *rl-bd*: $\bigwedge s.\ s \in simplices \Longrightarrow rl\ `\ s \subseteq \{..Suc\ n\}$
and *bnd*: $\bigwedge f\ s.\ s \in simplices \Longrightarrow face\ f\ s \Longrightarrow bnd\ f \Longrightarrow card\ \{s{\in}simplices.\ face\ f\ s\} = 1$
and *nbnd*: $\bigwedge f\ s.\ s \in simplices \Longrightarrow face\ f\ s \Longrightarrow \neg\ bnd\ f \Longrightarrow card\ \{s{\in}simplices.\ face\ f\ s\} = 2$
and *odd-card*: $odd\ (card\ \{f.\ (\exists s{\in}simplices.\ face\ f\ s) \wedge rl\ `\ f = \{..n\} \wedge bnd\ f\})$
shows $odd\ (card\ \{s{\in}simplices.\ (rl\ `\ s = \{..Suc\ n\})\})$
⟨*proof*⟩

locale *kuhn-simplex* =
fixes *p n* **and** *base upd* **and** *s* :: $(nat \Rightarrow nat)\ set$
assumes *base*: $base \in \{..<n\} \to \{..<p\}$
assumes *base-out*: $\bigwedge i.\ n \leq i \Longrightarrow base\ i = p$
assumes *upd*: *bij-betw upd* $\{..<n\}\ \{..<n\}$
assumes *s-pre*: $s = (\lambda i\ j.\ if\ j \in upd`\{..<i\}\ then\ Suc\ (base\ j)\ else\ base\ j)\ `\ \{..n\}$
begin

definition $enum\ i\ j = (if\ j \in upd`\{..<i\}\ then\ Suc\ (base\ j)\ else\ base\ j)$

lemma *s-eq*: $s = enum\ `\ \{..\ n\}$
⟨*proof*⟩

lemma *upd-space*: $i < n \Longrightarrow upd\ i < n$
⟨*proof*⟩

lemma *s-space*: $s \subseteq \{..< n\} \to \{..\ p\}$
⟨*proof*⟩

lemma *inj-upd*: *inj-on upd* $\{..< n\}$
⟨*proof*⟩

lemma *inj-enum*: *inj-on enum* $\{..\ n\}$
⟨*proof*⟩

lemma *enum-0*: $enum\ 0 = base$
⟨*proof*⟩

lemma *base-in-s*: $base \in s$
⟨*proof*⟩

lemma *enum-in*: $i \leq n \Longrightarrow enum\ i \in s$
⟨*proof*⟩

lemma *one-step*:
assumes *a*: $a \in s\ j < n$
assumes $*$: $\bigwedge a'.\ a' \in s \Longrightarrow a' \neq a \Longrightarrow a'\ j = p'$
shows $a\ j \neq p'$
⟨*proof*⟩

lemma *upd-inj*: $i < n \Longrightarrow j < n \Longrightarrow upd\ i = upd\ j \longleftrightarrow i = j$
⟨*proof*⟩

lemma *upd-surj*: $upd\ `\ \{..< n\} = \{..< n\}$
⟨*proof*⟩

lemma *in-upd-image*: $A \subseteq \{..< n\} \Longrightarrow i < n \Longrightarrow upd\ i \in upd\ `\ A \longleftrightarrow i \in A$
⟨*proof*⟩

lemma *enum-inj*: $i \leq n \Longrightarrow j \leq n \Longrightarrow enum\ i = enum\ j \longleftrightarrow i = j$
⟨*proof*⟩

lemma *in-enum-image*: $A \subseteq \{..\ n\} \Longrightarrow i \leq n \Longrightarrow enum\ i \in enum\ `\ A \longleftrightarrow i \in A$
⟨*proof*⟩

lemma *enum-mono*: $i \leq n \Longrightarrow j \leq n \Longrightarrow enum\ i \leq enum\ j \longleftrightarrow i \leq j$
⟨*proof*⟩

lemma *enum-strict-mono*: $i \leq n \Longrightarrow j \leq n \Longrightarrow enum\ i < enum\ j \longleftrightarrow i < j$

⟨*proof*⟩

lemma *chain*: $a \in s \Longrightarrow b \in s \Longrightarrow a \leq b \vee b \leq a$
⟨*proof*⟩

lemma *less*: $a \in s \Longrightarrow b \in s \Longrightarrow a\ i < b\ i \Longrightarrow a < b$
⟨*proof*⟩

lemma *enum-0-bot*: $a \in s \Longrightarrow a = enum\ 0 \longleftrightarrow (\forall a' \in s.\ a \leq a')$
⟨*proof*⟩

lemma *enum-n-top*: $a \in s \Longrightarrow a = enum\ n \longleftrightarrow (\forall a' \in s.\ a' \leq a)$
⟨*proof*⟩

lemma *enum-Suc*: $i < n \Longrightarrow enum\ (Suc\ i) = (enum\ i)(upd\ i := Suc\ (enum\ i\ (upd\ i)))$
⟨*proof*⟩

lemma *enum-eq-p*: $i \leq n \Longrightarrow n \leq j \Longrightarrow enum\ i\ j = p$
⟨*proof*⟩

lemma *out-eq-p*: $a \in s \Longrightarrow n \leq j \Longrightarrow a\ j = p$
⟨*proof*⟩

lemma *s-le-p*: $a \in s \Longrightarrow a\ j \leq p$
⟨*proof*⟩

lemma *le-Suc-base*: $a \in s \Longrightarrow a\ j \leq Suc\ (base\ j)$
⟨*proof*⟩

lemma *base-le*: $a \in s \Longrightarrow base\ j \leq a\ j$
⟨*proof*⟩

lemma *enum-le-p*: $i \leq n \Longrightarrow j < n \Longrightarrow enum\ i\ j \leq p$
⟨*proof*⟩

lemma *enum-less*: $a \in s \Longrightarrow i < n \Longrightarrow enum\ i < a \longleftrightarrow enum\ (Suc\ i) \leq a$
⟨*proof*⟩

lemma *ksimplex-0*:
$n = 0 \Longrightarrow s = \{(\lambda x.\ p)\}$
⟨*proof*⟩

lemma *replace-0*:
assumes $j < n\ a \in s$ **and** p: $\forall x \in s - \{a\}.\ x\ j = 0$ **and** $x \in s$
shows $x \leq a$
⟨*proof*⟩

lemma *replace-1*:

assumes $j < n$ $a \in s$ **and** p: $\forall x{\in}s - \{a\}.\ x\ j = p$ **and** $x \in s$
shows $a \leq x$
⟨*proof*⟩

end

locale *kuhn-simplex-pair* = s: *kuhn-simplex p n b-s u-s s* + t: *kuhn-simplex p n b-t u-t t*
for *p n b-s u-s s b-t u-t t*
begin

lemma *enum-eq*:
assumes l: $i \leq l$ $l \leq j$ **and** $j + d \leq n$
assumes *eq*: *s.enum* ' $\{i\ ..\ j\} = t.enum$ ' $\{i + d\ ..\ j + d\}$
shows $s.enum\ l = t.enum\ (l + d)$
⟨*proof*⟩

lemma *ksimplex-eq-bot*:
assumes a: $a \in s$ $\bigwedge a'.\ a' \in s \Longrightarrow a \leq a'$
assumes b: $b \in t$ $\bigwedge b'.\ b' \in t \Longrightarrow b \leq b'$
assumes *eq*: $s - \{a\} = t - \{b\}$
shows $s = t$
⟨*proof*⟩

lemma *ksimplex-eq-top*:
assumes a: $a \in s$ $\bigwedge a'.\ a' \in s \Longrightarrow a' \leq a$
assumes b: $b \in t$ $\bigwedge b'.\ b' \in t \Longrightarrow b' \leq b$
assumes *eq*: $s - \{a\} = t - \{b\}$
shows $s = t$
⟨*proof*⟩

end

inductive *ksimplex* **for** *p n* :: *nat* **where**
ksimplex: *kuhn-simplex p n base upd s* $\Longrightarrow$ *ksimplex p n s*

lemma *finite-ksimplexes*: *finite* $\{s.\ ksimplex\ p\ n\ s\}$
⟨*proof*⟩

lemma *ksimplex-card*:
assumes *ksimplex p n s* **shows** *card s* = *Suc n*
⟨*proof*⟩

lemma *simplex-top-face*:
assumes $0 < p$ $\forall x{\in}s'.\ x\ n = p$
shows *ksimplex p n s'* $\longleftrightarrow$ $(\exists s\ a.\ ksimplex\ p\ (Suc\ n)\ s \wedge a \in s \wedge s' = s - \{a\})$
⟨*proof*⟩

lemma *ksimplex-replace-0*:

assumes *s*: *ksimplex p n s* **and** *a*: $a \in s$
assumes *j*: $j < n$ **and** *p*: $\forall x{\in}s - \{a\}.\ x\ j = 0$
shows *card* $\{s'.\ ksimplex\ p\ n\ s' \wedge (\exists b{\in}s'.\ s' - \{b\} = s - \{a\})\} = 1$
⟨*proof*⟩

lemma *ksimplex-replace-1*:
assumes *s*: *ksimplex p n s* **and** *a*: $a \in s$
assumes *j*: $j < n$ **and** *p*: $\forall x{\in}s - \{a\}.\ x\ j = p$
shows *card* $\{s'.\ ksimplex\ p\ n\ s' \wedge (\exists b{\in}s'.\ s' - \{b\} = s - \{a\})\} = 1$
⟨*proof*⟩

lemma *card-2-exists*: *card* $s = 2 \longleftrightarrow (\exists x{\in}s.\ \exists y{\in}s.\ x \neq y \wedge (\forall z{\in}s.\ z = x \vee z = y))$
⟨*proof*⟩

lemma *ksimplex-replace-2*:
assumes *s*: *ksimplex p n s* **and** $a \in s$ **and** $n \neq 0$
and *lb*: $\forall j{<}n.\ \exists x{\in}s - \{a\}.\ x\ j \neq 0$
and *ub*: $\forall j{<}n.\ \exists x{\in}s - \{a\}.\ x\ j \neq p$
shows *card* $\{s'.\ ksimplex\ p\ n\ s' \wedge (\exists b{\in}s'.\ s' - \{b\} = s - \{a\})\} = 2$
⟨*proof*⟩

Hence another step towards concreteness.

lemma *kuhn-simplex-lemma*:
assumes $\forall s.\ ksimplex\ p\ (Suc\ n)\ s \longrightarrow rl\ `\ s \subseteq \{..\ Suc\ n\}$
and *odd* $(card\ \{f.\ \exists s\ a.\ ksimplex\ p\ (Suc\ n)\ s \wedge a \in s \wedge (f = s - \{a\}) \wedge rl\ `\ f = \{..n\} \wedge ((\exists j{\leq}n.\ \forall x{\in}f.\ x\ j = 0) \vee (\exists j{\leq}n.\ \forall x{\in}f.\ x\ j = p))\})$
shows *odd* $(card\ \{s.\ ksimplex\ p\ (Suc\ n)\ s \wedge rl\ `\ s = \{..Suc\ n\}\})$
⟨*proof*⟩

26.3 Reduced labelling

definition *reduced* :: $nat \Rightarrow (nat \Rightarrow nat) \Rightarrow nat$ **where** $reduced\ n\ x = (LEAST\ k.\ k = n \vee x\ k \neq 0)$

lemma *reduced-labelling*:
shows $reduced\ n\ x \leq n$
and $\forall i{<}reduced\ n\ x.\ x\ i = 0$
and $reduced\ n\ x = n \vee x\ (reduced\ n\ x) \neq 0$
⟨*proof*⟩

lemma *reduced-labelling-unique*:
$r \leq n \Longrightarrow \forall i{<}r.\ x\ i = 0 \Longrightarrow r = n \vee x\ r \neq 0 \Longrightarrow reduced\ n\ x = r$
⟨*proof*⟩

lemma *reduced-labelling-zero*: $j < n \Longrightarrow x\ j = 0 \Longrightarrow reduced\ n\ x \neq j$
⟨*proof*⟩

lemma *reduce-labelling-zero*[*simp*]: $reduced\ 0\ x = 0$

⟨*proof*⟩

lemma *reduced-labelling-nonzero*: $j < n \Longrightarrow x\ j \neq 0 \Longrightarrow reduced\ n\ x \leq j$
⟨*proof*⟩

lemma *reduced-labelling-Suc*: $reduced\ (Suc\ n)\ x \neq Suc\ n \Longrightarrow reduced\ (Suc\ n)\ x = reduced\ n\ x$
⟨*proof*⟩

lemma *complete-face-top*:
assumes $\forall x{\in}f.\ \forall j{\leq}n.\ x\ j = 0 \longrightarrow lab\ x\ j = 0$
and $\forall x{\in}f.\ \forall j{\leq}n.\ x\ j = p \longrightarrow lab\ x\ j = 1$
and *eq*: $(reduced\ (Suc\ n) \circ lab)\ `\ f = \{..n\}$
shows $((\exists j{\leq}n.\ \forall x{\in}f.\ x\ j = 0) \vee (\exists j{\leq}n.\ \forall x{\in}f.\ x\ j = p)) \longleftrightarrow (\forall x{\in}f.\ x\ n = p)$
⟨*proof*⟩

Hence we get just about the nice induction.

lemma *kuhn-induction*:
assumes $0 < p$
and *lab-0*: $\forall x.\ \forall j{\leq}n.\ (\forall j.\ x\ j \leq p) \wedge x\ j = 0 \longrightarrow lab\ x\ j = 0$
and *lab-1*: $\forall x.\ \forall j{\leq}n.\ (\forall j.\ x\ j \leq p) \wedge x\ j = p \longrightarrow lab\ x\ j = 1$
and *odd*: $odd\ (card\ \{s.\ ksimplex\ p\ n\ s \wedge (reduced\ n{\circ}lab)\ `\ s = \{..n\}\})$
shows $odd\ (card\ \{s.\ ksimplex\ p\ (Suc\ n)\ s \wedge (reduced\ (Suc\ n){\circ}lab)\ `\ s = \{..Suc\ n\}\})$
⟨*proof*⟩

And so we get the final combinatorial result.

lemma *ksimplex-0*: $ksimplex\ p\ 0\ s \longleftrightarrow s = \{(\lambda x.\ p)\}$
⟨*proof*⟩

lemma *kuhn-combinatorial*:
assumes $0 < p$
and $\forall x\ j.\ (\forall j.\ x\ j \leq p) \wedge j < n \wedge x\ j = 0 \longrightarrow lab\ x\ j = 0$
and $\forall x\ j.\ (\forall j.\ x\ j \leq p) \wedge j < n\ \wedge x\ j = p \longrightarrow lab\ x\ j = 1$
shows $odd\ (card\ \{s.\ ksimplex\ p\ n\ s \wedge (reduced\ n{\circ}lab)\ `\ s = \{..n\}\})$
(**is** $odd\ (card\ (?M\ n))$)
⟨*proof*⟩

lemma *kuhn-lemma*:
fixes $n\ p :: nat$
assumes $0 < p$
and $\forall x.\ (\forall i{<}n.\ x\ i \leq p) \longrightarrow (\forall i{<}n.\ label\ x\ i = (0{::}nat) \vee label\ x\ i = 1)$
and $\forall x.\ (\forall i{<}n.\ x\ i \leq p) \longrightarrow (\forall i{<}n.\ x\ i = 0 \longrightarrow label\ x\ i = 0)$
and $\forall x.\ (\forall i{<}n.\ x\ i \leq p) \longrightarrow (\forall i{<}n.\ x\ i = p \longrightarrow label\ x\ i = 1)$
obtains q **where** $\forall i{<}n.\ q\ i < p$
and $\forall i{<}n.\ \exists r\ s.\ (\forall j{<}n.\ q\ j \leq r\ j \wedge r\ j \leq q\ j + 1) \wedge (\forall j{<}n.\ q\ j \leq s\ j \wedge s\ j \leq q\ j + 1) \wedge label\ r\ i \neq label\ s\ i$
⟨*proof*⟩

26.4 The main result for the unit cube

lemma *kuhn-labelling-lemma'*:
assumes $(\forall x{::}nat \Rightarrow real.\ P\ x \longrightarrow P\ (f\ x))$
and $\forall x.\ P\ x \longrightarrow (\forall i{::}nat.\ Q\ i \longrightarrow 0 \leq x\ i \wedge x\ i \leq 1)$
shows $\exists l.\ (\forall x\ i.\ l\ x\ i \leq (1{::}nat)) \wedge$
$(\forall x\ i.\ P\ x \wedge Q\ i \wedge x\ i = 0 \longrightarrow l\ x\ i = 0) \wedge$
$(\forall x\ i.\ P\ x \wedge Q\ i \wedge x\ i = 1 \longrightarrow l\ x\ i = 1) \wedge$
$(\forall x\ i.\ P\ x \wedge Q\ i \wedge l\ x\ i = 0 \longrightarrow x\ i \leq f\ x\ i) \wedge$
$(\forall x\ i.\ P\ x \wedge Q\ i \wedge l\ x\ i = 1 \longrightarrow f\ x\ i \leq x\ i)$
⟨*proof*⟩

definition *unit-cube* :: *'a::euclidean-space set*
where $unit\text{-}cube = \{x.\ \forall i \in Basis.\ 0 \leq x \cdot i \wedge x \cdot i \leq 1\}$

lemma *mem-unit-cube*: $x \in unit\text{-}cube \longleftrightarrow (\forall i \in Basis.\ 0 \leq x \cdot i \wedge x \cdot i \leq 1)$
⟨*proof*⟩

lemma *bounded-unit-cube*: *bounded unit-cube*
⟨*proof*⟩

lemma *closed-unit-cube*: *closed unit-cube*
⟨*proof*⟩

lemma *compact-unit-cube*: *compact unit-cube* (**is** *compact ?C*)
⟨*proof*⟩

lemma *brouwer-cube*:
fixes *f* :: *'a::euclidean-space* ⇒ *'a*
assumes *continuous-on unit-cube f*
and *f ' unit-cube* ⊆ *unit-cube*
shows $\exists x \in unit\text{-}cube.\ f\ x = x$
⟨*proof*⟩

26.5 Retractions

definition *retraction s t r* ⟷ $t \subseteq s \wedge$ *continuous-on s r* $\wedge$ *r ' s* $\subseteq t \wedge (\forall x \in t.\ r\ x = x)$

definition *retract-of* (**infixl** *retract'-of 50*)
where $(t\ retract\text{-}of\ s) \longleftrightarrow (\exists r.\ retraction\ s\ t\ r)$

lemma *retraction-idempotent*: *retraction s t r* ⟹ $x \in s \Longrightarrow\ r\ (r\ x) = r\ x$
⟨*proof*⟩

26.6 Preservation of fixpoints under (more general notion of) retraction

lemma *invertible-fixpoint-property*:
fixes *s* :: *'a::euclidean-space set*

and $t :: 'b$::*euclidean-space set*
assumes *continuous-on t i*
and $i \; ` \; t \subseteq s$
and *continuous-on s r*
and $r \; ` \; s \subseteq t$
and $\forall y{\in}t.\; r\; (i\; y) = y$
and $\forall f.$ *continuous-on s f* $\wedge\; f \; ` \; s \subseteq s \longrightarrow (\exists x{\in}s.\; f\; x = x)$
and *continuous-on t g*
and $g \; ` \; t \subseteq t$
obtains y **where** $y \in t$ **and** $g\; y = y$
⟨*proof*⟩

lemma *homeomorphic-fixpoint-property*:
fixes $s :: 'a$::*euclidean-space set*
and $t :: 'b$::*euclidean-space set*
assumes *s homeomorphic t*
shows $(\forall f.$ *continuous-on s f* $\wedge\; f \; ` \; s \subseteq s \longrightarrow (\exists x{\in}s.\; f\; x = x)) \longleftrightarrow$
$(\forall g.$ *continuous-on t g* $\wedge\; g \; ` \; t \subseteq t \longrightarrow (\exists y{\in}t.\; g\; y = y))$
⟨*proof*⟩

lemma *retract-fixpoint-property*:
fixes $f :: 'a$::*euclidean-space* $\Rightarrow$ $'b$::*euclidean-space*
and $s :: 'a$ *set*
assumes *t retract-of s*
and $\forall f.$ *continuous-on s f* $\wedge\; f \; ` \; s \subseteq s \longrightarrow (\exists x{\in}s.\; f\; x = x)$
and *continuous-on t g*
and $g \; ` \; t \subseteq t$
obtains y **where** $y \in t$ **and** $g\; y = y$
⟨*proof*⟩

26.7 The Brouwer theorem for any set with nonempty interior

lemma *convex-unit-cube*: *convex unit-cube*
⟨*proof*⟩

lemma *brouwer-weak*:
fixes $f :: 'a$::*euclidean-space* $\Rightarrow$ $'a$
assumes *compact s*
and *convex s*
and *interior s* $\neq \{\}$
and *continuous-on s f*
and $f \; ` \; s \subseteq s$
obtains x **where** $x \in s$ **and** $f\; x = x$
⟨*proof*⟩

And in particular for a closed ball.

lemma *brouwer-ball*:
fixes $f :: 'a$::*euclidean-space* $\Rightarrow$ $'a$

assumes $e > 0$
and *continuous-on* (*cball a e*) *f*
and f ‘ *cball a e* $\subseteq$ *cball a e*
obtains x **where** $x \in$ *cball a e* **and** $f\,x = x$
⟨*proof*⟩

Still more general form; could derive this directly without using the rather involved *HOMEOMORPHIC-CONVEX-COMPACT* theorem, just using a scaling and translation to put the set inside the unit cube.

lemma *brouwer*:
fixes $f :: 'a$::*euclidean-space* $\Rightarrow 'a$
assumes *compact s*
and *convex s*
and $s \neq \{\}$
and *continuous-on s f*
and f ‘ $s \subseteq s$
obtains x **where** $x \in s$ **and** $f\,x = x$
⟨*proof*⟩

So we get the no-retraction theorem.

lemma *no-retraction-cball*:
fixes $a :: 'a$::*euclidean-space*
assumes $e > 0$
shows ¬ (*frontier* (*cball a e*) *retract-of* (*cball a e*))
⟨*proof*⟩

26.8 Retractions

lemma *retraction*:
retraction s t r ⟷
$t \subseteq s \wedge$ *continuous-on s r* $\wedge$ r ‘ $s = t \wedge (\forall x \in t.\ r\,x = x)$
⟨*proof*⟩

lemma *retract-of-imp-extensible*:
assumes *s retract-of t* **and** *continuous-on s f* **and** f ‘ $s \subseteq u$
obtains *g* **where** *continuous-on t g* g ‘ $t \subseteq u$ $\bigwedge x.\ x \in s \Longrightarrow g\,x = f\,x$
⟨*proof*⟩

lemma *idempotent-imp-retraction*:
assumes *continuous-on s f* **and** f ‘ $s \subseteq s$ **and** $\bigwedge x.\ x \in s \Longrightarrow f(f\,x) = f\,x$
shows *retraction s* (f ‘ s) *f*
⟨*proof*⟩

lemma *retraction-subset*:
assumes *retraction s t r* **and** $t \subseteq s'$ **and** $s' \subseteq s$
shows *retraction* s' *t r*
⟨*proof*⟩

lemma *retract-of-subset*:

assumes *t retract-of s* **and** $t \subseteq s'$ **and** $s' \subseteq s$
shows *t retract-of s'*
⟨*proof*⟩

lemma *retraction-refl* [*simp*]: *retraction s s* ($\lambda x.\ x$)
⟨*proof*⟩

lemma *retract-of-refl* [*iff*]: *s retract-of s*
⟨*proof*⟩

lemma *retract-of-imp-subset*:
s retract-of t $\Longrightarrow$ $s \subseteq t$
⟨*proof*⟩

lemma *retract-of-empty* [*simp*]:
($\{\}$ *retract-of s*) $\longleftrightarrow$ $s = \{\}$ (*s retract-of* $\{\}$) $\longleftrightarrow$ $s = \{\}$
⟨*proof*⟩

lemma *retract-of-singleton* [*iff*]: ($\{x\}$ *retract-of s*) $\longleftrightarrow$ $x \in s$
⟨*proof*⟩

lemma *retraction-comp*:
⟦*retraction s t f*; *retraction t u g*⟧
$\Longrightarrow$ *retraction s u* (*g o f*)
⟨*proof*⟩

lemma *retract-of-trans*:
assumes *s retract-of t* **and** *t retract-of u*
shows *s retract-of u*
⟨*proof*⟩

lemma *closedin-retract*:
fixes $s :: 'a :: real\text{-}normed\text{-}vector\ set$
assumes *s retract-of t*
shows *closedin* (*subtopology euclidean t*) *s*
⟨*proof*⟩

lemma *retract-of-contractible*:
assumes *contractible t s retract-of t*
shows *contractible s*
⟨*proof*⟩

lemma *retract-of-compact*:
⟦*compact t*; *s retract-of t*⟧ $\Longrightarrow$ *compact s*
⟨*proof*⟩

lemma *retract-of-closed*:
fixes $s :: 'a :: real\text{-}normed\text{-}vector\ set$
shows ⟦*closed t*; *s retract-of t*⟧ $\Longrightarrow$ *closed s*

⟨*proof*⟩

lemma *retract-of-connected*:
⟦*connected t*; *s retract-of t*⟧ ⟹ *connected s*
⟨*proof*⟩

lemma *retract-of-path-connected*:
⟦*path-connected t*; *s retract-of t*⟧ ⟹ *path-connected s*
⟨*proof*⟩

lemma *retract-of-simply-connected*:
⟦*simply-connected t*; *s retract-of t*⟧ ⟹ *simply-connected s*
⟨*proof*⟩

lemma *retract-of-homotopically-trivial*:
assumes *ts*: *t retract-of s*
and *hom*: ⋀*f g*. ⟦*continuous-on u f*; *f ' u* ⊆ *s*;
continuous-on u g; *g ' u* ⊆ *s*⟧
⟹ *homotopic-with* (λ*x*. *True*) *u s f g*
and *continuous-on u f f ' u* ⊆ *t*
and *continuous-on u g g ' u* ⊆ *t*
shows *homotopic-with* (λ*x*. *True*) *u t f g*
⟨*proof*⟩

lemma *retract-of-homotopically-trivial-null*:
assumes *ts*: *t retract-of s*
and *hom*: ⋀*f*. ⟦*continuous-on u f*; *f ' u* ⊆ *s*⟧
⟹ ∃ *c*. *homotopic-with* (λ*x*. *True*) *u s f* (λ*x*. *c*)
and *continuous-on u f f ' u* ⊆ *t*
obtains *c* **where** *homotopic-with* (λ*x*. *True*) *u t f* (λ*x*. *c*)
⟨*proof*⟩

lemma *retraction-imp-quotient-map*:
retraction s t r
⟹ *u* ⊆ *t*
⟹ (*openin* (*subtopology euclidean s*) {*x*. *x* ∈ *s* ∧ *r x* ∈ *u*} ⟷
openin (*subtopology euclidean t*) *u*)
⟨*proof*⟩

lemma *retract-of-locally-compact*:
fixes *s* :: *'a* :: {*heine-borel*,*real-normed-vector*} *set*
shows ⟦ *locally compact s*; *t retract-of s*⟧ ⟹ *locally compact t*
⟨*proof*⟩

lemma *retract-of-times*:
⟦*s retract-of s'*; *t retract-of t'*⟧ ⟹ (*s* × *t*) *retract-of* (*s'* × *t'*)
⟨*proof*⟩

lemma *homotopic-into-retract*:

$[\![f \text{ ` } s \subseteq t;\ g \text{ ` } s \subseteq t;\ t \textit{ retract-of } u;$
$\quad \textit{homotopic-with } (\lambda x.\ \textit{True})\ s\ u\ f\ g]\!]$
$\quad \Longrightarrow \textit{homotopic-with } (\lambda x.\ \textit{True})\ s\ t\ f\ g$
⟨*proof*⟩

end

27 A generic phantom type

theory *Phantom-Type*
imports *Main*
begin

datatype $('a, 'b)$ *phantom* = *phantom* (*of-phantom*: $'b$)

lemma *type-definition-phantom'*: *type-definition of-phantom phantom UNIV*
⟨*proof*⟩

lemma *phantom-comp-of-phantom* [*simp*]: *phantom* ∘ *of-phantom* = *id*
and *of-phantom-comp-phantom* [*simp*]: *of-phantom* ∘ *phantom* = *id*
⟨*proof*⟩

syntax *-Phantom* :: *type* ⇒ *logic* (*(1Phantom/(1'(-')))*)
translations
Phantom('*t*) => *CONST phantom* :: - ⇒ ('*t*, -) *phantom*

⟨*ML*⟩

lemma *of-phantom-inject* [*simp*]:
of-phantom x = *of-phantom y* ⟷ *x* = *y*
⟨*proof*⟩

end

28 Cardinality of types

theory *Cardinality*
imports *Phantom-Type*
begin

28.1 Preliminary lemmas

lemma (**in** *type-definition*) *univ*:
UNIV = *Abs* ` *A*
⟨*proof*⟩

lemma (**in** *type-definition*) *card*: *card* (*UNIV* :: $'b$ *set*) = *card A*
⟨*proof*⟩

lemma *finite-range-Some*: *finite* (*range* (*Some* :: *'a* ⇒ *'a option*)) = *finite* (*UNIV* :: *'a set*)
⟨*proof*⟩

lemma *infinite-literal*: ¬ *finite* (*UNIV* :: *String.literal set*)
⟨*proof*⟩

28.2 Cardinalities of types

syntax *-type-card* :: *type* => *nat* ((*1CARD/(1'(-')*)))

translations *CARD('t)* => *CONST card* (*CONST UNIV* :: *'t set*)

⟨*ML*⟩

lemma *card-prod* [*simp*]: *CARD('a* × *'b)* = *CARD('a)* ∗ *CARD('b)*
⟨*proof*⟩

lemma *card-UNIV-sum*: *CARD('a* + *'b)* = (*if CARD('a)* ≠ *0* ∧ *CARD('b)* ≠ *0 then CARD('a)* + *CARD('b) else 0*)
⟨*proof*⟩

lemma *card-sum* [*simp*]: *CARD('a* + *'b)* = *CARD('a*::*finite)* + *CARD('b*::*finite)*
⟨*proof*⟩

lemma *card-UNIV-option*: *CARD('a option)* = (*if CARD('a)* = *0 then 0 else CARD('a)* + *1*)
⟨*proof*⟩

lemma *card-option* [*simp*]: *CARD('a option)* = *Suc CARD('a*::*finite)*
⟨*proof*⟩

lemma *card-UNIV-set*: *CARD('a set)* = (*if CARD('a)* = *0 then 0 else 2* ^ *CARD('a)*)
⟨*proof*⟩

lemma *card-set* [*simp*]: *CARD('a set)* = *2* ^ *CARD('a*::*finite)*
⟨*proof*⟩

lemma *card-nat* [*simp*]: *CARD(nat)* = *0*
⟨*proof*⟩

lemma *card-fun*: *CARD('a* ⇒ *'b)* = (*if CARD('a)* ≠ *0* ∧ *CARD('b)* ≠ *0* ∨ *CARD('b)* = *1 then CARD('b)* ^ *CARD('a) else 0*)
⟨*proof*⟩

corollary *finite-UNIV-fun*:
finite (*UNIV* :: (*'a* ⇒ *'b*) *set*) ⟷
finite (*UNIV* :: *'a set*) ∧ *finite* (*UNIV* :: *'b set*) ∨ *CARD('b)* = *1*

(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *card-literal*: $CARD(String.literal) = 0$
⟨*proof*⟩

28.3 Classes with at least 1 and 2

Class finite already captures "at least 1"

lemma *zero-less-card-finite* [*simp*]: $0 < CARD('a::finite)$
⟨*proof*⟩

lemma *one-le-card-finite* [*simp*]: $Suc\ 0 \leq CARD('a::finite)$
⟨*proof*⟩

Class for cardinality "at least 2"

class *card2* = *finite* +
assumes *two-le-card*: $2 \leq CARD('a)$

lemma *one-less-card*: $Suc\ 0 < CARD('a::card2)$
⟨*proof*⟩

lemma *one-less-int-card*: $1 < int\ CARD('a::card2)$
⟨*proof*⟩

28.4 A type class for deciding finiteness of types

type-synonym *'a finite-UNIV* = *('a, bool) phantom*

class *finite-UNIV* =
fixes *finite-UNIV* :: *('a, bool) phantom*
assumes *finite-UNIV*: *finite-UNIV* = *Phantom('a)* (*finite* (*UNIV* :: *'a set*))

lemma *finite-UNIV-code* [*code-unfold*]:
finite (*UNIV* :: *'a* :: *finite-UNIV set*)
$\longleftrightarrow$ *of-phantom* (*finite-UNIV* :: *'a finite-UNIV*)
⟨*proof*⟩

28.5 A type class for computing the cardinality of types

definition *is-list-UNIV* :: *'a list* $\Rightarrow$ *bool*
where *is-list-UNIV xs* = (*let c* = *CARD('a) in if c* = *0 then False else size* (*remdups xs*) = *c*)

lemma *is-list-UNIV-iff*: *is-list-UNIV xs* $\longleftrightarrow$ *set xs* = *UNIV*
⟨*proof*⟩

type-synonym *'a card-UNIV* = *('a, nat) phantom*

class *card-UNIV* = *finite-UNIV* +
  fixes *card-UNIV* :: *'a card-UNIV*
  assumes *card-UNIV*: *card-UNIV* = *Phantom*(*'a*) *CARD*(*'a*)

28.6 Instantiations for *card-UNIV*

instantiation *nat* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*nat*) *False*
definition *card-UNIV* = *Phantom*(*nat*) *0*
instance ⟨*proof*⟩
end

instantiation *int* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*int*) *False*
definition *card-UNIV* = *Phantom*(*int*) *0*
instance ⟨*proof*⟩
end

instantiation *natural* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*natural*) *False*
definition *card-UNIV* = *Phantom*(*natural*) *0*
instance
  ⟨*proof*⟩
end

instantiation *integer* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*integer*) *False*
definition *card-UNIV* = *Phantom*(*integer*) *0*
instance
  ⟨*proof*⟩
end

instantiation *list* :: (*type*) *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*'a list*) *False*
definition *card-UNIV* = *Phantom*(*'a list*) *0*
instance ⟨*proof*⟩
end

instantiation *unit* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*unit*) *True*
definition *card-UNIV* = *Phantom*(*unit*) *1*
instance ⟨*proof*⟩
end

instantiation *bool* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*bool*) *True*
definition *card-UNIV* = *Phantom*(*bool*) *2*
instance ⟨*proof*⟩
end

instantiation *char* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*char*) *True*
definition *card-UNIV* = *Phantom*(*char*) *256*
instance ⟨*proof*⟩
end

instantiation *prod* :: (*finite-UNIV*, *finite-UNIV*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*('*a* × '*b*)
(*of-phantom* (*finite-UNIV* :: '*a finite-UNIV*) ∧ *of-phantom* (*finite-UNIV* :: '*b finite-UNIV*))
instance ⟨*proof*⟩
end

instantiation *prod* :: (*card-UNIV*, *card-UNIV*) *card-UNIV* **begin**
definition *card-UNIV* = *Phantom*('*a* × '*b*)
(*of-phantom* (*card-UNIV* :: '*a card-UNIV*) ∗ *of-phantom* (*card-UNIV* :: '*b card-UNIV*))
instance ⟨*proof*⟩
end

instantiation *sum* :: (*finite-UNIV*, *finite-UNIV*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*('*a* + '*b*)
(*of-phantom* (*finite-UNIV* :: '*a finite-UNIV*) ∧ *of-phantom* (*finite-UNIV* :: '*b finite-UNIV*))
instance
⟨*proof*⟩
end

instantiation *sum* :: (*card-UNIV*, *card-UNIV*) *card-UNIV* **begin**
definition *card-UNIV* = *Phantom*('*a* + '*b*)
(*let ca* = *of-phantom* (*card-UNIV* :: '*a card-UNIV*);
cb = *of-phantom* (*card-UNIV* :: '*b card-UNIV*)
in if ca ≠ *0* ∧ *cb* ≠ *0 then ca* + *cb else 0*)
instance ⟨*proof*⟩
end

instantiation *fun* :: (*finite-UNIV*, *card-UNIV*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*('*a* ⇒ '*b*)
(*let cb* = *of-phantom* (*card-UNIV* :: '*b card-UNIV*)
in cb = *1* ∨ *of-phantom* (*finite-UNIV* :: '*a finite-UNIV*) ∧ *cb* ≠ *0*)
instance
⟨*proof*⟩
end

instantiation *fun* :: (*card-UNIV*, *card-UNIV*) *card-UNIV* **begin**
definition *card-UNIV* = *Phantom*('*a* ⇒ '*b*)
(*let ca* = *of-phantom* (*card-UNIV* :: '*a card-UNIV*);
cb = *of-phantom* (*card-UNIV* :: '*b card-UNIV*)
in if ca ≠ *0* ∧ *cb* ≠ *0* ∨ *cb* = *1 then cb* ^ *ca else 0*)

instance ⟨*proof*⟩
end

instantiation *option* :: (*finite-UNIV*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*('*a option*) (*of-phantom* (*finite-UNIV* :: '*a finite-UNIV*))
instance ⟨*proof*⟩
end

instantiation *option* :: (*card-UNIV*) *card-UNIV* **begin**
definition *card-UNIV* = *Phantom*('*a option*)
(*let c* = *of-phantom* (*card-UNIV* :: '*a card-UNIV*) *in if c* ≠ *0 then Suc c else 0*)
instance ⟨*proof*⟩
end

instantiation *String.literal* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*String.literal*) *False*
definition *card-UNIV* = *Phantom*(*String.literal*) *0*
instance
⟨*proof*⟩
end

instantiation *set* :: (*finite-UNIV*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*('*a set*) (*of-phantom* (*finite-UNIV* :: '*a finite-UNIV*))
instance ⟨*proof*⟩
end

instantiation *set* :: (*card-UNIV*) *card-UNIV* **begin**
definition *card-UNIV* = *Phantom*('*a set*)
(*let c* = *of-phantom* (*card-UNIV* :: '*a card-UNIV*) *in if c* = *0 then 0 else 2* ^ *c*)
instance ⟨*proof*⟩
end

lemma *UNIV-finite-1*: *UNIV* = *set* [*finite-1*.a_1]
⟨*proof*⟩

lemma *UNIV-finite-2*: *UNIV* = *set* [*finite-2*.a_1, *finite-2*.a_2]
⟨*proof*⟩

lemma *UNIV-finite-3*: *UNIV* = *set* [*finite-3*.a_1, *finite-3*.a_2, *finite-3*.a_3]
⟨*proof*⟩

lemma *UNIV-finite-4*: *UNIV* = *set* [*finite-4*.a_1, *finite-4*.a_2, *finite-4*.a_3, *finite-4*.a_4]
⟨*proof*⟩

lemma *UNIV-finite-5*:
UNIV = *set* [*finite-5*.a_1, *finite-5*.a_2, *finite-5*.a_3, *finite-5*.a_4, *finite-5*.a_5]
⟨*proof*⟩

instantiation *Enum.finite-1* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*Enum.finite-1*) *True*
definition *card-UNIV* = *Phantom*(*Enum.finite-1*) *1*
instance
⟨*proof*⟩
end

instantiation *Enum.finite-2* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*Enum.finite-2*) *True*
definition *card-UNIV* = *Phantom*(*Enum.finite-2*) *2*
instance
⟨*proof*⟩
end

instantiation *Enum.finite-3* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*Enum.finite-3*) *True*
definition *card-UNIV* = *Phantom*(*Enum.finite-3*) *3*
instance
⟨*proof*⟩
end

instantiation *Enum.finite-4* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*Enum.finite-4*) *True*
definition *card-UNIV* = *Phantom*(*Enum.finite-4*) *4*
instance
⟨*proof*⟩
end

instantiation *Enum.finite-5* :: *card-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*Enum.finite-5*) *True*
definition *card-UNIV* = *Phantom*(*Enum.finite-5*) *5*
instance
⟨*proof*⟩
end

28.7 Code setup for sets

Implement *CARD*('*a*) via *card-UNIV-class.card-UNIV* and provide implementations for *finite*, *card*, *op* ⊆, and *op* =if the calling context already provides *finite-UNIV* and *card-UNIV* instances. If we implemented the latter always via *card-UNIV-class.card-UNIV*, we would require instances of essentially all element types, i.e., a lot of instantiation proofs and – at run time – possibly slow dictionary constructions.

context
begin

qualified definition *card-UNIV'* :: '*a card-UNIV*
where [*code del*]: *card-UNIV'* = *Phantom*('*a*) *CARD*('*a*)

lemma *CARD-code* [*code-unfold*]:
CARD('*a*) = *of-phantom* (*card-UNIV'* :: '*a card-UNIV*)
⟨*proof*⟩

lemma *card-UNIV'-code* [*code*]:
card-UNIV' = *card-UNIV*
⟨*proof*⟩

end

lemma *card-Compl*:
finite A ⟹ *card* (− *A*) = *card* (*UNIV* :: '*a set*) − *card* (*A* :: '*a set*)
⟨*proof*⟩

context fixes *xs* :: '*a* :: *finite-UNIV list*
begin

qualified definition *finite'* :: '*a set* ⇒ *bool*
where [*simp*, *code del*, *code-abbrev*]: *finite'* = *finite*

lemma *finite'-code* [*code*]:
finite' (*set xs*) ⟷ *True*
finite' (*List.coset xs*) ⟷ *of-phantom* (*finite-UNIV* :: '*a finite-UNIV*)
⟨*proof*⟩

end

context fixes *xs* :: '*a* :: *card-UNIV list*
begin

qualified definition *card'* :: '*a set* ⇒ *nat*
where [*simp*, *code del*, *code-abbrev*]: *card'* = *card*

lemma *card'-code* [*code*]:
card' (*set xs*) = *length* (*remdups xs*)
card' (*List.coset xs*) = *of-phantom* (*card-UNIV* :: '*a card-UNIV*) − *length* (*remdups xs*)
⟨*proof*⟩ **definition** *subset'* :: '*a set* ⇒ '*a set* ⇒ *bool*
where [*simp*, *code del*, *code-abbrev*]: *subset'* = *op* ⊆

lemma *subset'-code* [*code*]:
subset' *A* (*List.coset ys*) ⟷ (∀ *y* ∈ *set ys*. *y* ∉ *A*)
subset' (*set ys*) *B* ⟷ (∀ *y* ∈ *set ys*. *y* ∈ *B*)
subset' (*List.coset xs*) (*set ys*) ⟷ (*let n* = *CARD*('*a*) *in n* > *0* ∧ *card*(*set* (*xs* @ *ys*)) = *n*)
⟨*proof*⟩ **definition** *eq-set* :: '*a set* ⇒ '*a set* ⇒ *bool*
where [*simp*, *code del*, *code-abbrev*]: *eq-set* = *op* =

lemma *eq-set-code* [*code*]:
fixes *ys*
defines *rhs* ≡
let n = CARD('a)
in if n = 0 then False else
let xs' = remdups xs; ys' = remdups ys
in length xs' + length ys' = n ∧ (∀ x ∈ set xs'. x ∉ set ys') ∧ (∀ y ∈ set ys'. y ∉ set xs')
shows *eq-set (List.coset xs) (set ys) ⟷ rhs*
and *eq-set (set ys) (List.coset xs) ⟷ rhs*
and *eq-set (set xs) (set ys) ⟷ (∀ x ∈ set xs. x ∈ set ys) ∧ (∀ y ∈ set ys. y ∈ set xs)*
and *eq-set (List.coset xs) (List.coset ys) ⟷ (∀ x ∈ set xs. x ∈ set ys) ∧ (∀ y ∈ set ys. y ∈ set xs)*
⟨*proof*⟩

end

Provide more informative exceptions than Match for non-rewritten cases. If generated code raises one these exceptions, then a code equation calls the mentioned operator for an element type that is not an instance of *card-UNIV* and is therefore not implemented via *card-UNIV-class.card-UNIV*. Constrain the element type with sort *card-UNIV* to change this.

lemma *card-coset-error* [*code*]:
card (List.coset xs) =
Code.abort (STR ''card (List.coset -) requires type class instance card-UNIV'')
(λ-. card (List.coset xs))
⟨*proof*⟩

lemma *coset-subseteq-set-code* [*code*]:
List.coset xs ⊆ set ys ⟷
(if xs = [] ∧ ys = [] then False
else Code.abort
(STR ''subset-eq (List.coset -) (List.set -) requires type class instance card-UNIV'')
(λ-. List.coset xs ⊆ set ys))
⟨*proof*⟩

notepad begin — test code setup
⟨*proof*⟩
end

end

29 Numeral Syntax for Types

theory *Numeral-Type*
imports *Cardinality*
begin

29.1 Numeral Types

typedef *num0* = *UNIV* :: *nat set* ⟨*proof*⟩
typedef *num1* = *UNIV* :: *unit set* ⟨*proof*⟩

typedef *'a bit0* = {*0* ..< *2* ∗ *int CARD*('a::*finite*)}
⟨*proof*⟩

typedef *'a bit1* = {*0* ..< *1* + *2* ∗ *int CARD*('a::*finite*)}
⟨*proof*⟩

lemma *card-num0* [*simp*]: *CARD* (*num0*) = *0*
⟨*proof*⟩

lemma *infinite-num0*: ¬ *finite* (*UNIV* :: *num0 set*)
⟨*proof*⟩

lemma *card-num1* [*simp*]: *CARD*(*num1*) = *1*
⟨*proof*⟩

lemma *card-bit0* [*simp*]: *CARD*('a *bit0*) = *2* ∗ *CARD*('a::*finite*)
⟨*proof*⟩

lemma *card-bit1* [*simp*]: *CARD*('a *bit1*) = *Suc* (*2* ∗ *CARD*('a::*finite*))
⟨*proof*⟩

instance *num1* :: *finite*
⟨*proof*⟩

instance *bit0* :: (*finite*) *card2*
⟨*proof*⟩

instance *bit1* :: (*finite*) *card2*
⟨*proof*⟩

29.2 Locales for for modular arithmetic subtypes

locale *mod-type* =
fixes *n* :: *int*
and *Rep* :: *'a*::{*zero*,*one*,*plus*,*times*,*uminus*,*minus*} ⇒ *int*
and *Abs* :: *int* ⇒ *'a*::{*zero*,*one*,*plus*,*times*,*uminus*,*minus*}
assumes *type*: *type-definition Rep Abs* {*0*..<*n*}
and *size1*: *1* < *n*
and *zero-def*: *0* = *Abs 0*
and *one-def*: *1* = *Abs 1*
and *add-def*: *x* + *y* = *Abs* ((*Rep x* + *Rep y*) *mod n*)
and *mult-def*: *x* ∗ *y* = *Abs* ((*Rep x* ∗ *Rep y*) *mod n*)
and *diff-def*: *x* − *y* = *Abs* ((*Rep x* − *Rep y*) *mod n*)
and *minus-def*: − *x* = *Abs* ((− *Rep x*) *mod n*)
begin

lemma *size0*: $0 < n$
⟨*proof*⟩

lemmas *definitions* =
zero-def one-def add-def mult-def minus-def diff-def

lemma *Rep-less-n*: $Rep\ x < n$
⟨*proof*⟩

lemma *Rep-le-n*: $Rep\ x \leq n$
⟨*proof*⟩

lemma *Rep-inject-sym*: $x = y \longleftrightarrow Rep\ x = Rep\ y$
⟨*proof*⟩

lemma *Rep-inverse*: $Abs\ (Rep\ x) = x$
⟨*proof*⟩

lemma *Abs-inverse*: $m \in \{0..<n\} \Longrightarrow Rep\ (Abs\ m) = m$
⟨*proof*⟩

lemma *Rep-Abs-mod*: $Rep\ (Abs\ (m\ mod\ n)) = m\ mod\ n$
⟨*proof*⟩

lemma *Rep-Abs-0*: $Rep\ (Abs\ 0) = 0$
⟨*proof*⟩

lemma *Rep-0*: $Rep\ 0 = 0$
⟨*proof*⟩

lemma *Rep-Abs-1*: $Rep\ (Abs\ 1) = 1$
⟨*proof*⟩

lemma *Rep-1*: $Rep\ 1 = 1$
⟨*proof*⟩

lemma *Rep-mod*: $Rep\ x\ mod\ n = Rep\ x$
⟨*proof*⟩

lemmas *Rep-simps* =
Rep-inject-sym Rep-inverse Rep-Abs-mod Rep-mod Rep-Abs-0 Rep-Abs-1

lemma *comm-ring-1*: $OFCLASS('a, comm\text{-}ring\text{-}1\text{-}class)$
⟨*proof*⟩

end

locale *mod-ring* = *mod-type n Rep Abs*

for $n :: int$
and $Rep :: 'a::\{comm\text{-}ring\text{-}1\} \Rightarrow int$
and $Abs :: int \Rightarrow 'a::\{comm\text{-}ring\text{-}1\}$
begin

lemma *of-nat-eq*: $of\text{-}nat\ k = Abs\ (int\ k\ mod\ n)$
⟨*proof*⟩

lemma *of-int-eq*: $of\text{-}int\ z = Abs\ (z\ mod\ n)$
⟨*proof*⟩

lemma *Rep-numeral*:
$Rep\ (numeral\ w) = numeral\ w\ mod\ n$
⟨*proof*⟩

lemma *iszero-numeral*:
$iszero\ (numeral\ w::'a) \longleftrightarrow numeral\ w\ mod\ n = 0$
⟨*proof*⟩

lemma *cases*:
assumes *1*: $\bigwedge z.\ [\![(x::'a) = of\text{-}int\ z;\ 0 \le z;\ z < n]\!] \Longrightarrow P$
shows P
⟨*proof*⟩

lemma *induct*:
$(\bigwedge z.\ [\![0 \le z;\ z < n]\!] \Longrightarrow P\ (of\text{-}int\ z)) \Longrightarrow P\ (x::'a)$
⟨*proof*⟩

end

29.3 Ring class instances

Unfortunately *ring-1* instance is not possible for *num1*, since 0 and 1 are not distinct.

instantiation *num1* :: {*comm-ring*,*comm-monoid-mult*,*numeral*}
begin

lemma *num1-eq-iff*: $(x::num1) = (y::num1) \longleftrightarrow True$
⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation
bit0 **and** *bit1* :: (*finite*) {*zero*,*one*,*plus*,*times*,*uminus*,*minus*}
begin

definition *Abs-bit0'* :: *int* ⇒ *'a bit0* **where**
Abs-bit0' x = *Abs-bit0* (*x mod int CARD*(*'a bit0*))

definition *Abs-bit1'* :: *int* ⇒ *'a bit1* **where**
Abs-bit1' x = *Abs-bit1* (*x mod int CARD*(*'a bit1*))

definition *0* = *Abs-bit0 0*
definition *1* = *Abs-bit0 1*
definition *x* + *y* = *Abs-bit0'* (*Rep-bit0 x* + *Rep-bit0 y*)
definition *x* ∗ *y* = *Abs-bit0'* (*Rep-bit0 x* ∗ *Rep-bit0 y*)
definition *x* − *y* = *Abs-bit0'* (*Rep-bit0 x* − *Rep-bit0 y*)
definition − *x* = *Abs-bit0'* (− *Rep-bit0 x*)

definition *0* = *Abs-bit1 0*
definition *1* = *Abs-bit1 1*
definition *x* + *y* = *Abs-bit1'* (*Rep-bit1 x* + *Rep-bit1 y*)
definition *x* ∗ *y* = *Abs-bit1'* (*Rep-bit1 x* ∗ *Rep-bit1 y*)
definition *x* − *y* = *Abs-bit1'* (*Rep-bit1 x* − *Rep-bit1 y*)
definition − *x* = *Abs-bit1'* (− *Rep-bit1 x*)

instance ⟨*proof*⟩

end

interpretation *bit0*:
mod-type int CARD(*'a*::*finite bit0*)
Rep-bit0 :: *'a*::*finite bit0* ⇒ *int*
Abs-bit0 :: *int* ⇒ *'a*::*finite bit0*
⟨*proof*⟩

interpretation *bit1*:
mod-type int CARD(*'a*::*finite bit1*)
Rep-bit1 :: *'a*::*finite bit1* ⇒ *int*
Abs-bit1 :: *int* ⇒ *'a*::*finite bit1*
⟨*proof*⟩

instance *bit0* :: (*finite*) *comm-ring-1*
⟨*proof*⟩

instance *bit1* :: (*finite*) *comm-ring-1*
⟨*proof*⟩

interpretation *bit0*:
mod-ring int CARD(*'a*::*finite bit0*)
Rep-bit0 :: *'a*::*finite bit0* ⇒ *int*
Abs-bit0 :: *int* ⇒ *'a*::*finite bit0*
⟨*proof*⟩

interpretation *bit1*:

mod-ring int CARD('a::finite bit1)
Rep-bit1 :: 'a::finite bit1 ⇒ int
Abs-bit1 :: int ⇒ 'a::finite bit1
⟨*proof*⟩

Set up cases, induction, and arithmetic

lemmas *bit0-cases* [*case-names of-int*, *cases type*: *bit0*] = *bit0.cases*
lemmas *bit1-cases* [*case-names of-int*, *cases type*: *bit1*] = *bit1.cases*

lemmas *bit0-induct* [*case-names of-int*, *induct type*: *bit0*] = *bit0.induct*
lemmas *bit1-induct* [*case-names of-int*, *induct type*: *bit1*] = *bit1.induct*

lemmas *bit0-iszero-numeral* [*simp*] = *bit0.iszero-numeral*
lemmas *bit1-iszero-numeral* [*simp*] = *bit1.iszero-numeral*

lemmas [*simp*] = *eq-numeral-iff-iszero* [**where** *'a='a bit0*] **for** *dummy :: 'a::finite*
lemmas [*simp*] = *eq-numeral-iff-iszero* [**where** *'a='a bit1*] **for** *dummy :: 'a::finite*

29.4 Order instances

instantiation *bit0* **and** *bit1* :: (*finite*) *linorder* **begin**
definition $a < b \longleftrightarrow$ *Rep-bit0 a < Rep-bit0 b*
definition $a \leq b \longleftrightarrow$ *Rep-bit0 a ≤ Rep-bit0 b*
definition $a < b \longleftrightarrow$ *Rep-bit1 a < Rep-bit1 b*
definition $a \leq b \longleftrightarrow$ *Rep-bit1 a ≤ Rep-bit1 b*

instance
⟨*proof*⟩
end

lemma (**in** *preorder*) *tranclp-less*: *op* $<^{++}$ = *op* <
⟨*proof*⟩

instance *bit0* **and** *bit1* :: (*finite*) *wellorder*
⟨*proof*⟩

29.5 Code setup and type classes for code generation

Code setup for *num0* and *num1*

definition *Num0* :: *num0* **where** *Num0* = *Abs-num0 0*
code-datatype *Num0*

instantiation *num0* :: *equal* **begin**
definition *equal-num0* :: *num0* ⇒ *num0* ⇒ *bool*
where *equal-num0* = *op* =
instance ⟨*proof*⟩
end

lemma *equal-num0-code* [*code*]:

equal-class.equal Num0 Num0 = True
⟨*proof*⟩

code-datatype *1* :: *num1*

instantiation *num1* :: *equal* **begin**
definition *equal-num1* :: *num1* ⇒ *num1* ⇒ *bool*
where *equal-num1 = op =*
instance ⟨*proof*⟩
end

lemma *equal-num1-code* [*code*]:
equal-class.equal (*1* :: *num1*) *1 = True*
⟨*proof*⟩

instantiation *num1* :: *enum* **begin**
definition *enum-class.enum* = [*1* :: *num1*]
definition *enum-class.enum-all P = P* (*1* :: *num1*)
definition *enum-class.enum-ex P = P* (*1* :: *num1*)
instance
⟨*proof*⟩
end

instantiation *num0* **and** *num1* :: *card-UNIV* **begin**
definition *finite-UNIV = Phantom*(*num0*) *False*
definition *card-UNIV = Phantom*(*num0*) *0*
definition *finite-UNIV = Phantom*(*num1*) *True*
definition *card-UNIV = Phantom*(*num1*) *1*
instance
⟨*proof*⟩
end

Code setup for *'a bit0* and *'a bit1*

declare
bit0.Rep-inverse[*code abstype*]
bit0.Rep-0[*code abstract*]
bit0.Rep-1[*code abstract*]

lemma *Abs-bit0'-code* [*code abstract*]:
Rep-bit0 (*Abs-bit0' x* :: *'a* :: *finite bit0*) = *x mod int* (*CARD*(*'a bit0*))
⟨*proof*⟩

lemma *inj-on-Abs-bit0*:
inj-on (*Abs-bit0* :: *int* ⇒ *'a bit0*) {*0..<2* ∗ *int CARD*(*'a* :: *finite*)}
⟨*proof*⟩

declare
bit1.Rep-inverse[*code abstype*]
bit1.Rep-0[*code abstract*]

bit1.Rep-1[*code abstract*]

lemma *Abs-bit1'-code* [*code abstract*]:
Rep-bit1 (*Abs-bit1'* *x* :: *'a* :: *finite bit1*) = *x mod int* (*CARD*(*'a bit1*))
⟨*proof*⟩

lemma *inj-on-Abs-bit1*:
inj-on (*Abs-bit1* :: *int* ⇒ *'a bit1*) {*0..<1 + 2 * int CARD*(*'a* :: *finite*)}
⟨*proof*⟩

instantiation *bit0* **and** *bit1* :: (*finite*) *equal* **begin**

definition *equal-class.equal x y* ⟷ *Rep-bit0 x* = *Rep-bit0 y*
definition *equal-class.equal x y* ⟷ *Rep-bit1 x* = *Rep-bit1 y*

instance
⟨*proof*⟩

end

instantiation *bit0* :: (*finite*) *enum* **begin**
definition (*enum-class.enum* :: *'a bit0 list*) = *map* (*Abs-bit0'* ∘ *int*) (*upt 0* (*CARD*(*'a bit0*)))
definition *enum-class.enum-all P* = (∀ *b* :: *'a bit0* ∈ *set enum-class.enum*. *P b*)
definition *enum-class.enum-ex P* = (∃ *b* :: *'a bit0* ∈ *set enum-class.enum*. *P b*)

instance
⟨*proof*⟩

end

instantiation *bit1* :: (*finite*) *enum* **begin**
definition (*enum-class.enum* :: *'a bit1 list*) = *map* (*Abs-bit1'* ∘ *int*) (*upt 0* (*CARD*(*'a bit1*)))
definition *enum-class.enum-all P* = (∀ *b* :: *'a bit1* ∈ *set enum-class.enum*. *P b*)
definition *enum-class.enum-ex P* = (∃ *b* :: *'a bit1* ∈ *set enum-class.enum*. *P b*)

instance
⟨*proof*⟩

end

instantiation *bit0* **and** *bit1* :: (*finite*) *finite-UNIV* **begin**
definition *finite-UNIV* = *Phantom*(*'a bit0*) *True*
definition *finite-UNIV* = *Phantom*(*'a bit1*) *True*
instance ⟨*proof*⟩
end

instantiation *bit0* **and** *bit1* :: ({*finite*,*card-UNIV*}) *card-UNIV* **begin**

definition *card-UNIV* = *Phantom*('*a bit0*) (*2* ∗ *of-phantom* (*card-UNIV* :: '*a card-UNIV*))
definition *card-UNIV* = *Phantom*('*a bit1*) (*1* + *2* ∗ *of-phantom* (*card-UNIV* :: '*a card-UNIV*))
instance ⟨*proof*⟩
end

29.6 Syntax

syntax
-NumeralType :: *num-token* => *type* (-)
-NumeralType0 :: *type* (*0*)
-NumeralType1 :: *type* (*1*)

translations
(*type*) *1* == (*type*) *num1*
(*type*) *0* == (*type*) *num0*

⟨*ML*⟩

29.7 Examples

lemma *CARD*(*0*) = *0* ⟨*proof*⟩
lemma *CARD*(*17*) = *17* ⟨*proof*⟩
lemma *8* ∗ *11* ^ *3* − *6* = (*2*::*5*) ⟨*proof*⟩

end

30 Definition of finite Cartesian product types.

theory *Finite-Cartesian-Product*
imports
Euclidean-Space
L2-Norm
~~/*src*/*HOL*/*Library*/*Numeral-Type*
begin

30.1 Finite Cartesian products, with indexing and lambdas.

typedef ('*a*, '*b*) *vec* = *UNIV* :: (('*b*::*finite*) ⇒ '*a*) *set*
morphisms *vec-nth* *vec-lambda* ⟨*proof*⟩

notation
vec-nth (**infixl** $ *90*) **and**
vec-lambda (**binder** χ *10*)

syntax *-finite-vec* :: *type* ⇒ *type* ⇒ *type* ((- ^/ -) [*15*, *16*] *15*)

⟨*ML*⟩

lemma *vec-eq-iff*: $(x = y) \longleftrightarrow (\forall\, i.\ x\$i = y\$i)$
⟨*proof*⟩

lemma *vec-lambda-beta* [*simp*]: *vec-lambda g* $ *i* = *g i*
⟨*proof*⟩

lemma *vec-lambda-unique*: $(\forall\, i.\ f\$i = g\ i) \longleftrightarrow$ *vec-lambda g* = *f*
⟨*proof*⟩

lemma *vec-lambda-eta*: $(\chi\ i.\ (g\$i)) = g$
⟨*proof*⟩

30.2 Group operations and class instances

instantiation *vec* :: (*zero*, *finite*) *zero*
begin
definition *0* ≡ $(\chi\ i.\ 0)$
instance ⟨*proof*⟩
end

instantiation *vec* :: (*plus*, *finite*) *plus*
begin
definition *op* + ≡ $(\lambda\ x\ y.\ (\chi\ i.\ x\$i + y\$i))$
instance ⟨*proof*⟩
end

instantiation *vec* :: (*minus*, *finite*) *minus*
begin
definition *op* − ≡ $(\lambda\ x\ y.\ (\chi\ i.\ x\$i - y\$i))$
instance ⟨*proof*⟩
end

instantiation *vec* :: (*uminus*, *finite*) *uminus*
begin
definition *uminus* ≡ $(\lambda\ x.\ (\chi\ i.\ -\ (x\$i)))$
instance ⟨*proof*⟩
end

lemma *zero-index* [*simp*]: *0* $ *i* = *0*
⟨*proof*⟩

lemma *vector-add-component* [*simp*]: $(x + y)\$i = x\$i + y\$i$
⟨*proof*⟩

lemma *vector-minus-component* [*simp*]: $(x - y)\$i = x\$i - y\$i$
⟨*proof*⟩

lemma *vector-uminus-component* [*simp*]: $(-x)\$i = -(x\$i)$
⟨*proof*⟩

instance *vec* :: (*semigroup-add*, *finite*) *semigroup-add*
⟨*proof*⟩

instance *vec* :: (*ab-semigroup-add*, *finite*) *ab-semigroup-add*
⟨*proof*⟩

instance *vec* :: (*monoid-add*, *finite*) *monoid-add*
⟨*proof*⟩

instance *vec* :: (*comm-monoid-add*, *finite*) *comm-monoid-add*
⟨*proof*⟩

instance *vec* :: (*cancel-semigroup-add*, *finite*) *cancel-semigroup-add*
⟨*proof*⟩

instance *vec* :: (*cancel-ab-semigroup-add*, *finite*) *cancel-ab-semigroup-add*
⟨*proof*⟩

instance *vec* :: (*cancel-comm-monoid-add*, *finite*) *cancel-comm-monoid-add* ⟨*proof*⟩

instance *vec* :: (*group-add*, *finite*) *group-add*
⟨*proof*⟩

instance *vec* :: (*ab-group-add*, *finite*) *ab-group-add*
⟨*proof*⟩

30.3 Real vector space

instantiation *vec* :: (*real-vector*, *finite*) *real-vector*
begin

definition $scaleR \equiv (\lambda\ r\ x.\ (\chi\ i.\ scaleR\ r\ (x\$i)))$

lemma *vector-scaleR-component* [*simp*]: $(scaleR\ r\ x)\$i = scaleR\ r\ (x\$i)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

30.4 Topological space

instantiation *vec* :: (*topological-space*, *finite*) *topological-space*
begin

definition [*code del*]:
open (*S* :: ('*a* ^ '*b*) *set*) ⟷
(∀ *x*∈*S*. ∃ *A*. (∀ *i*. *open* (*A i*) ∧ *x*$*i* ∈ *A i*) ∧
(∀ *y*. (∀ *i*. *y*$*i* ∈ *A i*) ⟶ *y* ∈ *S*))

instance ⟨*proof*⟩

end

lemma *open-vector-box*: ∀ *i*. *open* (*S i*) ⟹ *open* {*x*. ∀ *i*. *x* $ *i* ∈ *S i*}
⟨*proof*⟩

lemma *open-vimage-vec-nth*: *open S* ⟹ *open* ((λ*x*. *x* $ *i*) −‘ *S*)
⟨*proof*⟩

lemma *closed-vimage-vec-nth*: *closed S* ⟹ *closed* ((λ*x*. *x* $ *i*) −‘ *S*)
⟨*proof*⟩

lemma *closed-vector-box*: ∀ *i*. *closed* (*S i*) ⟹ *closed* {*x*. ∀ *i*. *x* $ *i* ∈ *S i*}
⟨*proof*⟩

lemma *tendsto-vec-nth* [*tendsto-intros*]:
assumes ((λ*x*. *f x*) ⟶ *a*) *net*
shows ((λ*x*. *f x* $ *i*) ⟶ *a* $ *i*) *net*
⟨*proof*⟩

lemma *isCont-vec-nth* [*simp*]: *isCont f a* ⟹ *isCont* (λ*x*. *f x* $ *i*) *a*
⟨*proof*⟩

lemma *vec-tendstoI*:
assumes ⋀*i*. ((λ*x*. *f x* $ *i*) ⟶ *a* $ *i*) *net*
shows ((λ*x*. *f x*) ⟶ *a*) *net*
⟨*proof*⟩

lemma *tendsto-vec-lambda* [*tendsto-intros*]:
assumes ⋀*i*. ((λ*x*. *f x i*) ⟶ *a i*) *net*
shows ((λ*x*. χ *i*. *f x i*) ⟶ (χ *i*. *a i*)) *net*
⟨*proof*⟩

lemma *open-image-vec-nth*: **assumes** *open S* **shows** *open* ((λ*x*. *x* $ *i*) ‘ *S*)
⟨*proof*⟩

instance *vec* :: (*perfect-space*, *finite*) *perfect-space*
⟨*proof*⟩

30.5 Metric space

instantiation *vec* :: (*metric-space*, *finite*) *dist*
begin

definition
dist x y = setL2 (λi. dist (x$i) (y$i)) UNIV

instance ⟨*proof*⟩
end

instantiation *vec* :: (*metric-space*, *finite*) *uniformity-dist*
begin

definition [*code del*]:
(*uniformity* :: (('*a*, '*b*) *vec* × ('*a*, '*b*) *vec*) *filter*) =
(*INF e*:{*0* <..}. *principal* {(*x*, *y*). *dist x y* < *e*})

instance
⟨*proof*⟩
end

declare *uniformity-Abort*[**where** '*a*='*a* :: *metric-space* ^ '*b* :: *finite*, *code*]

instantiation *vec* :: (*metric-space*, *finite*) *metric-space*
begin

lemma *dist-vec-nth-le*: *dist* (*x* $ *i*) (*y* $ *i*) ≤ *dist x y*
⟨*proof*⟩

instance ⟨*proof*⟩

end

lemma *Cauchy-vec-nth*:
Cauchy (λ*n*. *X n*) ⟹ *Cauchy* (λ*n*. *X n* $ *i*)
⟨*proof*⟩

lemma *vec-CauchyI*:
fixes *X* :: *nat* ⇒ '*a*::*metric-space* ^ '*n*
assumes *X*: ⋀*i*. *Cauchy* (λ*n*. *X n* $ *i*)
shows *Cauchy* (λ*n*. *X n*)
⟨*proof*⟩

instance *vec* :: (*complete-space*, *finite*) *complete-space*
⟨*proof*⟩

30.6 Normed vector space

instantiation *vec* :: (*real-normed-vector*, *finite*) *real-normed-vector*
begin

definition *norm x = setL2* (λ*i*. *norm* (*x*$*i*)) *UNIV*

definition *sgn* $(x::'a\hat{}'b) = scaleR\ (inverse\ (norm\ x))\ x$

instance ⟨*proof*⟩

end

lemma *norm-nth-le*: $norm\ (x\ \$\ i) \leq norm\ x$
⟨*proof*⟩

lemma *bounded-linear-vec-nth*: *bounded-linear* $(\lambda x.\ x\ \$\ i)$
⟨*proof*⟩

instance *vec* :: (*banach*, *finite*) *banach* ⟨*proof*⟩

30.7 Inner product space

instantiation *vec* :: (*real-inner*, *finite*) *real-inner*
begin

definition $inner\ x\ y = setsum\ (\lambda i.\ inner\ (x\$i)\ (y\$i))\ UNIV$

instance ⟨*proof*⟩

end

30.8 Euclidean space

Vectors pointing along a single axis.

definition $axis\ k\ x = (\chi\ i.\ if\ i = k\ then\ x\ else\ 0)$

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

lemma *axis-eq-axis*: $axis\ i\ x = axis\ j\ y \longleftrightarrow x = y \wedge i = j \vee x = 0 \wedge y = 0$
⟨*proof*⟩

lemma *inner-axis-axis*:
$inner\ (axis\ i\ x)\ (axis\ j\ y) = (if\ i = j\ then\ inner\ x\ y\ else\ 0)$
⟨*proof*⟩

lemma *setsum-single*:
assumes *finite* A **and** $k \in A$ **and** $f\ k = y$
assumes $\bigwedge i.\ i \in A \Longrightarrow i \neq k \Longrightarrow f\ i = 0$
shows $(\sum i{\in}A.\ f\ i) = y$
⟨*proof*⟩

lemma *inner-axis*: $inner\ x\ (axis\ i\ y) = inner\ (x\ \$\ i)\ y$
⟨*proof*⟩

instantiation *vec* :: (*euclidean-space*, *finite*) *euclidean-space*
begin

definition *Basis* = $(\bigcup i.\ \bigcup u{\in}Basis.\ \{axis\ i\ u\})$

instance ⟨*proof*⟩

lemma *DIM-cart*[*simp*]: $DIM('a\hat{}'b) = CARD('b) * DIM('a)$
⟨*proof*⟩

end

lemma *cart-eq-inner-axis*: $a\ \$\ i = inner\ a\ (axis\ i\ 1)$
⟨*proof*⟩

lemma *axis-in-Basis*: $a \in Basis \Longrightarrow axis\ i\ a \in Basis$
⟨*proof*⟩

end

31 Operator Norm

theory *Operator-Norm*
imports *Complex-Main*
begin

This formulation yields zero if $'a$ is the trivial vector space.

definition *onorm* :: ($'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*) ⇒ *real*
where $onorm\ f = (SUP\ x.\ norm\ (f\ x)\ /\ norm\ x)$

lemma *onorm-bound*:
assumes $0 \le b$ **and** $\bigwedge x.\ norm\ (f\ x) \le b * norm\ x$
shows $onorm\ f \le b$
⟨*proof*⟩

In non-trivial vector spaces, the first assumption is redundant.

lemma *onorm-le*:
fixes f :: $'a$::{*real-normed-vector*, *perfect-space*} ⇒ $'b$::*real-normed-vector*
assumes $\bigwedge x.\ norm\ (f\ x) \le b * norm\ x$
shows $onorm\ f \le b$
⟨*proof*⟩

lemma *le-onorm*:
assumes *bounded-linear f*
shows $norm\ (f\ x)\ /\ norm\ x \le onorm\ f$
⟨*proof*⟩

lemma *onorm*:
assumes *bounded-linear f*
shows *norm* $(f\,x) \leq$ *onorm* $f *$ *norm* x
⟨*proof*⟩

lemma *onorm-pos-le*:
assumes f: *bounded-linear f*
shows $0 \leq$ *onorm f*
⟨*proof*⟩

lemma *onorm-zero*: *onorm* $(\lambda x.\ 0) = 0$
⟨*proof*⟩

lemma *onorm-eq-0*:
assumes f: *bounded-linear f*
shows *onorm* $f = 0 \longleftrightarrow (\forall x.\ f\,x = 0)$
⟨*proof*⟩

lemma *onorm-pos-lt*:
assumes f: *bounded-linear f*
shows $0 <$ *onorm* $f \longleftrightarrow \neg\ (\forall x.\ f\,x = 0)$
⟨*proof*⟩

lemma *onorm-id-le*: *onorm* $(\lambda x.\ x) \leq 1$
⟨*proof*⟩

lemma *onorm-id*: *onorm* $(\lambda x.\ x{::}'a{::}\{$*real-normed-vector*, *perfect-space*$\}) = 1$
⟨*proof*⟩

lemma *onorm-compose*:
assumes f: *bounded-linear f*
assumes g: *bounded-linear g*
shows *onorm* $(f \circ g) \leq$ *onorm* $f *$ *onorm* g
⟨*proof*⟩

lemma *onorm-scaleR-lemma*:
assumes f: *bounded-linear f*
shows *onorm* $(\lambda x.\ r *_R f\,x) \leq |r| *$ *onorm f*
⟨*proof*⟩

lemma *onorm-scaleR*:
assumes f: *bounded-linear f*
shows *onorm* $(\lambda x.\ r *_R f\,x) = |r| *$ *onorm f*
⟨*proof*⟩

lemma *onorm-scaleR-left-lemma*:
assumes r: *bounded-linear r*
shows *onorm* $(\lambda x.\ r\,x *_R f) \leq$ *onorm* $r *$ *norm f*
⟨*proof*⟩

lemma *onorm-scaleR-left*:
assumes *f*: *bounded-linear r*
shows *onorm* $(\lambda x.\ r\ x *_R f) = onorm\ r * norm\ f$
⟨*proof*⟩

lemma *onorm-neg*:
shows *onorm* $(\lambda x.\ -f\ x) = onorm\ f$
⟨*proof*⟩

lemma *onorm-triangle*:
assumes *f*: *bounded-linear f*
assumes *g*: *bounded-linear g*
shows *onorm* $(\lambda x.\ f\ x + g\ x) \leq onorm\ f + onorm\ g$
⟨*proof*⟩

lemma *onorm-triangle-le*:
assumes *bounded-linear f*
assumes *bounded-linear g*
assumes $onorm\ f + onorm\ g \leq e$
shows *onorm* $(\lambda x.\ f\ x + g\ x) \leq e$
⟨*proof*⟩

lemma *onorm-triangle-lt*:
assumes *bounded-linear f*
assumes *bounded-linear g*
assumes $onorm\ f + onorm\ g < e$
shows *onorm* $(\lambda x.\ f\ x + g\ x) < e$
⟨*proof*⟩

end

32 Countable Complete Lattices

theory *Countable-Complete-Lattices*
imports *Main Countable-Set*
begin

lemma *UNIV-nat-eq*: $UNIV = insert\ 0\ (range\ Suc)$
⟨*proof*⟩

class *countable-complete-lattice* = *lattice* + *Inf* + *Sup* + *bot* + *top* +
assumes *ccInf-lower*: $countable\ A \Longrightarrow x \in A \Longrightarrow Inf\ A \leq x$
assumes *ccInf-greatest*: $countable\ A \Longrightarrow (\bigwedge x.\ x \in A \Longrightarrow z \leq x) \Longrightarrow z \leq Inf\ A$
assumes *ccSup-upper*: $countable\ A \Longrightarrow x \in A \Longrightarrow x \leq Sup\ A$
assumes *ccSup-least*: $countable\ A \Longrightarrow (\bigwedge x.\ x \in A \Longrightarrow x \leq z) \Longrightarrow Sup\ A \leq z$
assumes *ccInf-empty* [*simp*]: $Inf\ \{\} = top$
assumes *ccSup-empty* [*simp*]: $Sup\ \{\} = bot$
begin

subclass *bounded-lattice*
⟨*proof*⟩

lemma *ccINF-lower*: *countable A* ⟹ *i* ∈ *A* ⟹ (*INF i* :*A*. *f i*) ≤ *f i*
⟨*proof*⟩

lemma *ccINF-greatest*: *countable A* ⟹ (⋀*i*. *i* ∈ *A* ⟹ *u* ≤ *f i*) ⟹ *u* ≤ (*INF i* :*A*. *f i*)
⟨*proof*⟩

lemma *ccSUP-upper*: *countable A* ⟹ *i* ∈ *A* ⟹ *f i* ≤ (*SUP i* :*A*. *f i*)
⟨*proof*⟩

lemma *ccSUP-least*: *countable A* ⟹ (⋀*i*. *i* ∈ *A* ⟹ *f i* ≤ *u*) ⟹ (*SUP i* :*A*. *f i*) ≤ *u*
⟨*proof*⟩

lemma *ccInf-lower2*: *countable A* ⟹ *u* ∈ *A* ⟹ *u* ≤ *v* ⟹ *Inf A* ≤ *v*
⟨*proof*⟩

lemma *ccINF-lower2*: *countable A* ⟹ *i* ∈ *A* ⟹ *f i* ≤ *u* ⟹ (*INF i* :*A*. *f i*) ≤ *u*
⟨*proof*⟩

lemma *ccSup-upper2*: *countable A* ⟹ *u* ∈ *A* ⟹ *v* ≤ *u* ⟹ *v* ≤ *Sup A*
⟨*proof*⟩

lemma *ccSUP-upper2*: *countable A* ⟹ *i* ∈ *A* ⟹ *u* ≤ *f i* ⟹ *u* ≤ (*SUP i* :*A*. *f i*)
⟨*proof*⟩

lemma *le-ccInf-iff*: *countable A* ⟹ *b* ≤ *Inf A* ⟷ (∀ *a*∈*A*. *b* ≤ *a*)
⟨*proof*⟩

lemma *le-ccINF-iff*: *countable A* ⟹ *u* ≤ (*INF i* :*A*. *f i*) ⟷ (∀ *i*∈*A*. *u* ≤ *f i*)
⟨*proof*⟩

lemma *ccSup-le-iff*: *countable A* ⟹ *Sup A* ≤ *b* ⟷ (∀ *a*∈*A*. *a* ≤ *b*)
⟨*proof*⟩

lemma *ccSUP-le-iff*: *countable A* ⟹ (*SUP i* :*A*. *f i*) ≤ *u* ⟷ (∀ *i*∈*A*. *f i* ≤ *u*)
⟨*proof*⟩

lemma *ccInf-insert* [*simp*]: *countable A* ⟹ *Inf* (*insert a A*) = *inf a* (*Inf A*)
⟨*proof*⟩

lemma *ccINF-insert* [*simp*]: *countable A* ⟹ (*INF x*:*insert a A*. *f x*) = *inf* (*f a*) (*INFIMUM A f*)
⟨*proof*⟩

lemma *ccSup-insert* [*simp*]: *countable A* $\Longrightarrow$ *Sup* (*insert a A*) = *sup a* (*Sup A*)
⟨*proof*⟩

lemma *ccSUP-insert* [*simp*]: *countable A* $\Longrightarrow$ (*SUP x*:*insert a A*. *f x*) = *sup* (*f a*) (*SUPREMUM A f*)
⟨*proof*⟩

lemma *ccINF-empty* [*simp*]: (*INF x*:{}. *f x*) = *top*
⟨*proof*⟩

lemma *ccSUP-empty* [*simp*]: (*SUP x*:{}. *f x*) = *bot*
⟨*proof*⟩

lemma *ccInf-superset-mono*: *countable A* $\Longrightarrow$ *B* ⊆ *A* $\Longrightarrow$ *Inf A* ≤ *Inf B*
⟨*proof*⟩

lemma *ccSup-subset-mono*: *countable B* $\Longrightarrow$ *A* ⊆ *B* $\Longrightarrow$ *Sup A* ≤ *Sup B*
⟨*proof*⟩

lemma *ccInf-mono*:
assumes [*intro*]: *countable B countable A*
assumes $\bigwedge b.\ b \in B \Longrightarrow \exists\, a{\in}A.\ a \leq b$
shows *Inf A* ≤ *Inf B*
⟨*proof*⟩

lemma *ccINF-mono*:
countable A $\Longrightarrow$ *countable B* $\Longrightarrow$ ($\bigwedge m.\ m \in B \Longrightarrow \exists\, n{\in}A.\ f\ n \leq g\ m$) $\Longrightarrow$ (*INF n*:*A*. *f n*) ≤ (*INF n*:*B*. *g n*)
⟨*proof*⟩

lemma *ccSup-mono*:
assumes [*intro*]: *countable B countable A*
assumes $\bigwedge a.\ a \in A \Longrightarrow \exists\, b{\in}B.\ a \leq b$
shows *Sup A* ≤ *Sup B*
⟨*proof*⟩

lemma *ccSUP-mono*:
countable A $\Longrightarrow$ *countable B* $\Longrightarrow$ ($\bigwedge n.\ n \in A \Longrightarrow \exists\, m{\in}B.\ f\ n \leq g\ m$) $\Longrightarrow$ (*SUP n*:*A*. *f n*) ≤ (*SUP n*:*B*. *g n*)
⟨*proof*⟩

lemma *ccINF-superset-mono*:
countable A $\Longrightarrow$ *B* ⊆ *A* $\Longrightarrow$ ($\bigwedge x.\ x \in B \Longrightarrow f\ x \leq g\ x$) $\Longrightarrow$ (*INF x*:*A*. *f x*) ≤ (*INF x*:*B*. *g x*)
⟨*proof*⟩

lemma *ccSUP-subset-mono*:
countable B $\Longrightarrow$ *A* ⊆ *B* $\Longrightarrow$ ($\bigwedge x.\ x \in A \Longrightarrow f\ x \leq g\ x$) $\Longrightarrow$ (*SUP x*:*A*. *f x*) ≤

(*SUP* x:B. $g\ x$)
⟨*proof*⟩

lemma *less-eq-ccInf-inter*: *countable* $A \Longrightarrow$ *countable* $B \Longrightarrow$ *sup* (*Inf* A) (*Inf* B) $\leq$ *Inf* ($A \cap B$)
⟨*proof*⟩

lemma *ccSup-inter-less-eq*: *countable* $A \Longrightarrow$ *countable* $B \Longrightarrow$ *Sup* ($A \cap B$) $\leq$ *inf* (*Sup* A) (*Sup* B)
⟨*proof*⟩

lemma *ccInf-union-distrib*: *countable* $A \Longrightarrow$ *countable* $B \Longrightarrow$ *Inf* ($A \cup B$) = *inf* (*Inf* A) (*Inf* B)
⟨*proof*⟩

lemma *ccINF-union*:
countable $A \Longrightarrow$ *countable* $B \Longrightarrow$ (*INF* i:$A \cup B$. $M\ i$) = *inf* (*INF* i:A. $M\ i$) (*INF* i:B. $M\ i$)
⟨*proof*⟩

lemma *ccSup-union-distrib*: *countable* $A \Longrightarrow$ *countable* $B \Longrightarrow$ *Sup* ($A \cup B$) = *sup* (*Sup* A) (*Sup* B)
⟨*proof*⟩

lemma *ccSUP-union*:
countable $A \Longrightarrow$ *countable* $B \Longrightarrow$ (*SUP* i:$A \cup B$. $M\ i$) = *sup* (*SUP* i:A. $M\ i$) (*SUP* i:B. $M\ i$)
⟨*proof*⟩

lemma *ccINF-inf-distrib*: *countable* $A \Longrightarrow$ *inf* (*INF* a:A. $f\ a$) (*INF* a:A. $g\ a$) = (*INF* a:A. *inf* ($f\ a$) ($g\ a$))
⟨*proof*⟩

lemma *ccSUP-sup-distrib*: *countable* $A \Longrightarrow$ *sup* (*SUP* a:A. $f\ a$) (*SUP* a:A. $g\ a$) = (*SUP* a:A. *sup* ($f\ a$) ($g\ a$))
⟨*proof*⟩

lemma *ccINF-const* [*simp*]: $A \neq \{\} \Longrightarrow$ (*INF* i :A. f) = f
⟨*proof*⟩

lemma *ccSUP-const* [*simp*]: $A \neq \{\} \Longrightarrow$ (*SUP* i :A. f) = f
⟨*proof*⟩

lemma *ccINF-top* [*simp*]: (*INF* x:A. *top*) = *top*
⟨*proof*⟩

lemma *ccSUP-bot* [*simp*]: (*SUP* x:A. *bot*) = *bot*
⟨*proof*⟩

lemma *ccINF-commute*: *countable A* $\Longrightarrow$ *countable B* $\Longrightarrow$ (*INF i*:*A*. *INF j*:*B*. *f i j*) = (*INF j*:*B*. *INF i*:*A*. *f i j*)
⟨*proof*⟩

lemma *ccSUP-commute*: *countable A* $\Longrightarrow$ *countable B* $\Longrightarrow$ (*SUP i*:*A*. *SUP j*:*B*. *f i j*) = (*SUP j*:*B*. *SUP i*:*A*. *f i j*)
⟨*proof*⟩

end

context
fixes *a* :: *'a*::{*countable-complete-lattice*, *linorder*}
begin

lemma *less-ccSup-iff*: *countable S* $\Longrightarrow$ *a* < *Sup S* $\longleftrightarrow$ ($\exists$ *x*∈*S*. *a* < *x*)
⟨*proof*⟩

lemma *less-ccSUP-iff*: *countable A* $\Longrightarrow$ *a* < (*SUP i*:*A*. *f i*) $\longleftrightarrow$ ($\exists$ *x*∈*A*. *a* < *f x*)
⟨*proof*⟩

lemma *ccInf-less-iff*: *countable S* $\Longrightarrow$ *Inf S* < *a* $\longleftrightarrow$ ($\exists$ *x*∈*S*. *x* < *a*)
⟨*proof*⟩

lemma *ccINF-less-iff*: *countable A* $\Longrightarrow$ (*INF i*:*A*. *f i*) < *a* $\longleftrightarrow$ ($\exists$ *x*∈*A*. *f x* < *a*)
⟨*proof*⟩

end

class *countable-complete-distrib-lattice* = *countable-complete-lattice* +
assumes *sup-ccInf*: *countable B* $\Longrightarrow$ *sup a* (*Inf B*) = (*INF b*:*B*. *sup a b*)
assumes *inf-ccSup*: *countable B* $\Longrightarrow$ *inf a* (*Sup B*) = (*SUP b*:*B*. *inf a b*)
begin

lemma *sup-ccINF*:
countable B $\Longrightarrow$ *sup a* (*INF b*:*B*. *f b*) = (*INF b*:*B*. *sup a* (*f b*))
⟨*proof*⟩

lemma *inf-ccSUP*:
countable B $\Longrightarrow$ *inf a* (*SUP b*:*B*. *f b*) = (*SUP b*:*B*. *inf a* (*f b*))
⟨*proof*⟩

subclass *distrib-lattice*
⟨*proof*⟩

lemma *ccInf-sup*:
countable B $\Longrightarrow$ *sup* (*Inf B*) *a* = (*INF b*:*B*. *sup b a*)
⟨*proof*⟩

lemma *ccSup-inf*:
countable B $\Longrightarrow$ *inf* (*Sup B*) *a* = (*SUP b*:*B*. *inf b a*)
⟨*proof*⟩

lemma *ccINF-sup*:
countable B $\Longrightarrow$ *sup* (*INF b*:*B*. *f b*) *a* = (*INF b*:*B*. *sup* (*f b*) *a*)
⟨*proof*⟩

lemma *ccSUP-inf*:
countable B $\Longrightarrow$ *inf* (*SUP b*:*B*. *f b*) *a* = (*SUP b*:*B*. *inf* (*f b*) *a*)
⟨*proof*⟩

lemma *ccINF-sup-distrib2*:
countable A $\Longrightarrow$ *countable B* $\Longrightarrow$ *sup* (*INF a*:*A*. *f a*) (*INF b*:*B*. *g b*) = (*INF a*:*A*. *INF b*:*B*. *sup* (*f a*) (*g b*))
⟨*proof*⟩

lemma *ccSUP-inf-distrib2*:
countable A $\Longrightarrow$ *countable B* $\Longrightarrow$ *inf* (*SUP a*:*A*. *f a*) (*SUP b*:*B*. *g b*) = (*SUP a*:*A*. *SUP b*:*B*. *inf* (*f a*) (*g b*))
⟨*proof*⟩

context
fixes *f* :: *'a* $\Rightarrow$ *'b*::*countable-complete-lattice*
assumes *mono f*
begin

lemma *mono-ccInf*:
countable A $\Longrightarrow$ *f* (*Inf A*) $\leq$ (*INF x*:*A*. *f x*)
⟨*proof*⟩

lemma *mono-ccSup*:
countable A $\Longrightarrow$ (*SUP x*:*A*. *f x*) $\leq$ *f* (*Sup A*)
⟨*proof*⟩

lemma *mono-ccINF*:
countable I $\Longrightarrow$ *f* (*INF i* : *I*. *A i*) $\leq$ (*INF x* : *I*. *f* (*A x*))
⟨*proof*⟩

lemma *mono-ccSUP*:
countable I $\Longrightarrow$ (*SUP x* : *I*. *f* (*A x*)) $\leq$ *f* (*SUP i* : *I*. *A i*)
⟨*proof*⟩

end

end

32.0.1 Instances of countable complete lattices

instance *fun* :: (*type*, *countable-complete-lattice*) *countable-complete-lattice*
⟨*proof*⟩

subclass (**in** *complete-lattice*) *countable-complete-lattice*
⟨*proof*⟩

subclass (**in** *complete-distrib-lattice*) *countable-complete-distrib-lattice*
⟨*proof*⟩

end

33 Continuity and iterations

theory *Order-Continuity*
imports *Complex-Main Countable-Complete-Lattices*
begin

lemma *SUP-nat-binary*:
(*SUP n*::*nat*. *if n* = *0 then A else B*) = (*sup A B*::′*a*::*countable-complete-lattice*)
⟨*proof*⟩

lemma *INF-nat-binary*:
(*INF n*::*nat*. *if n* = *0 then A else B*) = (*inf A B*::′*a*::*countable-complete-lattice*)
⟨*proof*⟩

The name *continuous* is already taken in *Complex-Main*, so we use *sup-continuous* and *inf-continuous*. These names appear sometimes in literature and have the advantage that these names are duals.

named-theorems *order-continuous-intros*

33.1 Continuity for complete lattices

definition
sup-continuous :: (′*a*::*countable-complete-lattice* ⇒ ′*b*::*countable-complete-lattice*)
⇒ *bool*
where
sup-continuous F ⟷ (∀ *M*::*nat* ⇒ ′*a*. *mono M* ⟶ *F* (*SUP i*. *M i*) = (*SUP i*. *F* (*M i*)))

lemma *sup-continuousD*: *sup-continuous F* ⟹ *mono M* ⟹ *F* (*SUP i*::*nat*. *M i*) = (*SUP i*. *F* (*M i*))
⟨*proof*⟩

lemma *sup-continuous-mono*:
assumes [*simp*]: *sup-continuous F* **shows** *mono F*

⟨*proof*⟩

lemma [*order-continuous-intros*]:
shows *sup-continuous-const*: *sup-continuous* (λ*x*. *c*)
and *sup-continuous-id*: *sup-continuous* (λ*x*. *x*)
and *sup-continuous-apply*: *sup-continuous* (λ*f*. *f x*)
and *sup-continuous-fun*: (⋀*s*. *sup-continuous* (λ*x*. *P x s*)) ⟹ *sup-continuous P*
and *sup-continuous-If*: *sup-continuous F* ⟹ *sup-continuous G* ⟹ *sup-continuous* (λ*f*. *if C then F f else G f*)
⟨*proof*⟩

lemma *sup-continuous-compose*:
assumes *f*: *sup-continuous f* **and** *g*: *sup-continuous g*
shows *sup-continuous* (λ*x*. *f* (*g x*))
⟨*proof*⟩

lemma *sup-continuous-sup*[*order-continuous-intros*]:
sup-continuous f ⟹ *sup-continuous g* ⟹ *sup-continuous* (λ*x*. *sup* (*f x*) (*g x*))
⟨*proof*⟩

lemma *sup-continuous-inf*[*order-continuous-intros*]:
fixes *P Q* :: *'a* :: *countable-complete-lattice* ⇒ *'b* :: *countable-complete-distrib-lattice*
assumes *P*: *sup-continuous P* **and** *Q*: *sup-continuous Q*
shows *sup-continuous* (λ*x*. *inf* (*P x*) (*Q x*))
⟨*proof*⟩

lemma *sup-continuous-and*[*order-continuous-intros*]:
sup-continuous P ⟹ *sup-continuous Q* ⟹ *sup-continuous* (λ*x*. *P x* ∧ *Q x*)
⟨*proof*⟩

lemma *sup-continuous-or*[*order-continuous-intros*]:
sup-continuous P ⟹ *sup-continuous Q* ⟹ *sup-continuous* (λ*x*. *P x* ∨ *Q x*)
⟨*proof*⟩

lemma *sup-continuous-lfp*:
assumes *sup-continuous F* **shows** *lfp F* = (*SUP i*. (*F* ^^ *i*) *bot*) (**is** *lfp F* = *?U*)
⟨*proof*⟩

lemma *lfp-transfer-bounded*:
assumes *P*: *P bot* ⋀*x*. *P x* ⟹ *P* (*f x*) ⋀*M*. (⋀*i*. *P* (*M i*)) ⟹ *P* (*SUP i*::*nat*. *M i*)
assumes α: ⋀*M*. *mono M* ⟹ (⋀*i*::*nat*. *P* (*M i*)) ⟹ α (*SUP i*. *M i*) = (*SUP i*. α (*M i*))
assumes *f*: *sup-continuous f* **and** *g*: *sup-continuous g*
assumes [*simp*]: ⋀*x*. *P x* ⟹ *x* ≤ *lfp f* ⟹ α (*f x*) = *g* (α *x*)
assumes *g-bound*: ⋀*x*. α *bot* ≤ *g x*
shows α (*lfp f*) = *lfp g*
⟨*proof*⟩

lemma *lfp-transfer*:
sup-continuous $\alpha \Longrightarrow$ *sup-continuous* $f \Longrightarrow$ *sup-continuous* $g \Longrightarrow$
$(\bigwedge x.\ \alpha\ bot \leq g\ x) \Longrightarrow (\bigwedge x.\ x \leq lfp\ f \Longrightarrow \alpha\ (f\ x) = g\ (\alpha\ x)) \Longrightarrow \alpha\ (lfp\ f) = lfp\ g$
⟨*proof*⟩

definition
inf-continuous :: ('*a*::*countable-complete-lattice* ⇒ '*b*::*countable-complete-lattice*) ⇒ *bool*
where
inf-continuous F ⟷ (∀ *M*::*nat* ⇒ '*a*. *antimono M* ⟶ *F* (*INF i*. *M i*) = (*INF i*. *F* (*M i*)))

lemma *inf-continuousD*: *inf-continuous F* ⟹ *antimono M* ⟹ *F* (*INF i*::*nat*. *M i*) = (*INF i*. *F* (*M i*))
⟨*proof*⟩

lemma *inf-continuous-mono*:
assumes [*simp*]: *inf-continuous F* **shows** *mono F*
⟨*proof*⟩

lemma [*order-continuous-intros*]:
shows *inf-continuous-const*: *inf-continuous* (λ*x*. *c*)
and *inf-continuous-id*: *inf-continuous* (λ*x*. *x*)
and *inf-continuous-apply*: *inf-continuous* (λ*f*. *f x*)
and *inf-continuous-fun*: $(\bigwedge s.$ *inf-continuous* (λ*x*. *P x s*)) ⟹ *inf-continuous P*
and *inf-continuous-If*: *inf-continuous F* ⟹ *inf-continuous G* ⟹ *inf-continuous* (λ*f*. *if C then F f else G f*)
⟨*proof*⟩

lemma *inf-continuous-inf*[*order-continuous-intros*]:
inf-continuous f ⟹ *inf-continuous g* ⟹ *inf-continuous* (λ*x*. *inf* (*f x*) (*g x*))
⟨*proof*⟩

lemma *inf-continuous-sup*[*order-continuous-intros*]:
fixes *P Q* :: '*a* :: *countable-complete-lattice* ⇒ '*b* :: *countable-complete-distrib-lattice*
assumes *P*: *inf-continuous P* **and** *Q*: *inf-continuous Q*
shows *inf-continuous* (λ*x*. *sup* (*P x*) (*Q x*))
⟨*proof*⟩

lemma *inf-continuous-and*[*order-continuous-intros*]:
inf-continuous P ⟹ *inf-continuous Q* ⟹ *inf-continuous* (λ*x*. *P x* ∧ *Q x*)
⟨*proof*⟩

lemma *inf-continuous-or*[*order-continuous-intros*]:
inf-continuous P ⟹ *inf-continuous Q* ⟹ *inf-continuous* (λ*x*. *P x* ∨ *Q x*)
⟨*proof*⟩

lemma *inf-continuous-compose*:
assumes f: *inf-continuous* f **and** g: *inf-continuous* g
shows *inf-continuous* $(\lambda x.\ f\ (g\ x))$
⟨*proof*⟩

lemma *inf-continuous-gfp*:
assumes *inf-continuous* F **shows** *gfp* $F = (INF\ i.\ (F\ \hat{}\hat{}\ i)\ top)$ (**is** *gfp* $F = ?U$)
⟨*proof*⟩

lemma *gfp-transfer*:
assumes α: *inf-continuous* α **and** f: *inf-continuous* f **and** g: *inf-continuous* g
assumes [*simp*]: $\alpha\ top = top\ \bigwedge x.\ \alpha\ (f\ x) = g\ (\alpha\ x)$
shows $\alpha\ (gfp\ f) = gfp\ g$
⟨*proof*⟩

lemma *gfp-transfer-bounded*:
assumes P: $P\ (f\ top)\ \bigwedge x.\ P\ x \Longrightarrow P\ (f\ x)\ \bigwedge M.\ antimono\ M \Longrightarrow (\bigwedge i.\ P\ (M\ i)) \Longrightarrow P\ (INF\ i{::}nat.\ M\ i)$
assumes α: $\bigwedge M.\ antimono\ M \Longrightarrow (\bigwedge i{::}nat.\ P\ (M\ i)) \Longrightarrow \alpha\ (INF\ i.\ M\ i) = (INF\ i.\ \alpha\ (M\ i))$
assumes f: *inf-continuous* f **and** g: *inf-continuous* g
assumes [*simp*]: $\bigwedge x.\ P\ x \Longrightarrow \alpha\ (f\ x) = g\ (\alpha\ x)$
assumes *g-bound*: $\bigwedge x.\ g\ x \leq \alpha\ (f\ top)$
shows $\alpha\ (gfp\ f) = gfp\ g$
⟨*proof*⟩

33.1.1 Least fixed points in countable complete lattices

definition (**in** *countable-complete-lattice*) *cclfp* :: $('a \Rightarrow 'a) \Rightarrow 'a$
where $cclfp\ f = (SUP\ i.\ (f\ \hat{}\hat{}\ i)\ bot)$

lemma *cclfp-unfold*:
assumes *sup-continuous* F **shows** *cclfp* $F = F\ (cclfp\ F)$
⟨*proof*⟩

lemma *cclfp-lowerbound*: **assumes** f: *mono* f **and** A: $f\ A \leq A$ **shows** *cclfp* $f \leq A$
⟨*proof*⟩

lemma *cclfp-transfer*:
assumes *sup-continuous* α *mono* f
assumes $\alpha\ bot = bot\ \bigwedge x.\ \alpha\ (f\ x) = g\ (\alpha\ x)$
shows $\alpha\ (cclfp\ f) = cclfp\ g$
⟨*proof*⟩

end

34 Extended natural numbers (i.e. with infinity)

theory *Extended-Nat*
imports *Main Countable Order-Continuity*
begin

class *infinity* =
  fixes *infinity* :: *'a* (∞)

context
  fixes *f* :: *nat* ⇒ *'a*::{*canonically-ordered-monoid-add*, *linorder-topology*, *complete-linorder*}
begin

lemma *sums-SUP*[*simp*, *intro*]: *f sums* (*SUP n.* $\sum i<n.$ *f i*)
  ⟨*proof*⟩

lemma *suminf-eq-SUP*: *suminf f* = (*SUP n.* $\sum i<n.$ *f i*)
  ⟨*proof*⟩

end

34.1 Type definition

We extend the standard natural numbers by a special value indicating infinity.

typedef *enat* = *UNIV* :: *nat option set* ⟨*proof*⟩

TODO: introduce enat as coinductive datatype, enat is just *of-nat*

definition *enat* :: *nat* ⇒ *enat* **where**
  enat n = *Abs-enat* (*Some n*)

instantiation *enat* :: *infinity*
begin

definition ∞ = *Abs-enat None*
instance ⟨*proof*⟩

end

instance *enat* :: *countable*
⟨*proof*⟩

old-rep-datatype *enat* ∞ :: *enat*
⟨*proof*⟩

declare [[*coercion enat*::*nat*⇒*enat*]]

lemmas *enat2-cases* = *enat.exhaust*[*case-product enat.exhaust*]
lemmas *enat3-cases* = *enat.exhaust*[*case-product enat.exhaust enat.exhaust*]

lemma *not-infinity-eq* [*iff*]: $(x \neq \infty) = (\exists\, i.\ x = enat\ i)$
⟨*proof*⟩

lemma *not-enat-eq* [*iff*]: $(\forall\, y.\ x \neq enat\ y) = (x = \infty)$
⟨*proof*⟩

lemma *enat-ex-split*: $(\exists\, c{::}enat.\ P\ c) \longleftrightarrow P\ \infty \vee (\exists\, c{::}nat.\ P\ c)$
⟨*proof*⟩

primrec *the-enat* :: *enat* ⇒ *nat*
where *the-enat* (*enat n*) = *n*

34.2 Constructors and numbers

instantiation *enat* :: *zero-neq-one*
begin

definition
0 = *enat 0*

definition
1 = *enat 1*

instance
⟨*proof*⟩

end

definition *eSuc* :: *enat* ⇒ *enat* **where**
eSuc i = (*case i of enat n* ⇒ *enat* (*Suc n*) | ∞ ⇒ ∞)

lemma *enat-0* [*code-post*]: *enat 0* = *0*
⟨*proof*⟩

lemma *enat-1* [*code-post*]: *enat 1* = *1*
⟨*proof*⟩

lemma *enat-0-iff*: *enat x* = *0* ⟷ *x* = *0* *0* = *enat x* ⟷ *x* = *0*
⟨*proof*⟩

lemma *enat-1-iff*: *enat x* = *1* ⟷ *x* = *1* *1* = *enat x* ⟷ *x* = *1*
⟨*proof*⟩

lemma *one-eSuc*: *1* = *eSuc 0*
⟨*proof*⟩

lemma *infinity-ne-i0* [*simp*]: (∞::*enat*) ≠ *0*
⟨*proof*⟩

lemma *i0-ne-infinity* [*simp*]: $0 \neq (\infty::enat)$
⟨*proof*⟩

lemma *zero-one-enat-neq*:
$\neg\ 0 = (1::enat)$
$\neg\ 1 = (0::enat)$
⟨*proof*⟩

lemma *infinity-ne-i1* [*simp*]: $(\infty::enat) \neq 1$
⟨*proof*⟩

lemma *i1-ne-infinity* [*simp*]: $1 \neq (\infty::enat)$
⟨*proof*⟩

lemma *eSuc-enat*: $eSuc\ (enat\ n) = enat\ (Suc\ n)$
⟨*proof*⟩

lemma *eSuc-infinity* [*simp*]: $eSuc\ \infty = \infty$
⟨*proof*⟩

lemma *eSuc-ne-0* [*simp*]: $eSuc\ n \neq 0$
⟨*proof*⟩

lemma *zero-ne-eSuc* [*simp*]: $0 \neq eSuc\ n$
⟨*proof*⟩

lemma *eSuc-inject* [*simp*]: $eSuc\ m = eSuc\ n \longleftrightarrow m = n$
⟨*proof*⟩

lemma *eSuc-enat-iff*: $eSuc\ x = enat\ y \longleftrightarrow (\exists\, n.\ y = Suc\ n \wedge x = enat\ n)$
⟨*proof*⟩

lemma *enat-eSuc-iff*: $enat\ y = eSuc\ x \longleftrightarrow (\exists\, n.\ y = Suc\ n \wedge enat\ n = x)$
⟨*proof*⟩

34.3 Addition

instantiation *enat* :: *comm-monoid-add*
begin

definition [*nitpick-simp*]:
$m + n = (case\ m\ of\ \infty \Rightarrow \infty \mid enat\ m \Rightarrow (case\ n\ of\ \infty \Rightarrow \infty \mid enat\ n \Rightarrow enat\ (m + n)))$

lemma *plus-enat-simps* [*simp*, *code*]:
fixes q :: *enat*
shows $enat\ m + enat\ n = enat\ (m + n)$
and $\infty + q = \infty$

and $q + \infty = \infty$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *eSuc-plus-1*:
$eSuc\ n = n + 1$
⟨*proof*⟩

lemma *plus-1-eSuc*:
$1 + q = eSuc\ q$
$q + 1 = eSuc\ q$
⟨*proof*⟩

lemma *iadd-Suc*: $eSuc\ m + n = eSuc\ (m + n)$
⟨*proof*⟩

lemma *iadd-Suc-right*: $m + eSuc\ n = eSuc\ (m + n)$
⟨*proof*⟩

34.4 Multiplication

instantiation *enat* :: {*comm-semiring-1*, *semiring-no-zero-divisors*}
begin

definition *times-enat-def* [*nitpick-simp*]:
$m * n = (case\ m\ of\ \infty \Rightarrow if\ n = 0\ then\ 0\ else\ \infty \mid enat\ m \Rightarrow$
$(case\ n\ of\ \infty \Rightarrow if\ m = 0\ then\ 0\ else\ \infty \mid enat\ n \Rightarrow enat\ (m * n)))$

lemma *times-enat-simps* [*simp*, *code*]:
$enat\ m * enat\ n = enat\ (m * n)$
$\infty * \infty = (\infty :: enat)$
$\infty * enat\ n = (if\ n = 0\ then\ 0\ else\ \infty)$
$enat\ m * \infty = (if\ m = 0\ then\ 0\ else\ \infty)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *mult-eSuc*: $eSuc\ m * n = n + m * n$
⟨*proof*⟩

lemma *mult-eSuc-right*: $m * eSuc\ n = m + m * n$
⟨*proof*⟩

lemma *of-nat-eq-enat*: *of-nat n = enat n*
⟨*proof*⟩

instance *enat* :: *semiring-char-0*
⟨*proof*⟩

lemma *imult-is-infinity*: $((a::enat) * b = \infty) = (a = \infty \wedge b \neq 0 \vee b = \infty \wedge a \neq 0)$
⟨*proof*⟩

34.5 Numerals

lemma *numeral-eq-enat*:
numeral k = enat (numeral k)
⟨*proof*⟩

lemma *enat-numeral* [*code-abbrev*]:
enat (numeral k) = numeral k
⟨*proof*⟩

lemma *infinity-ne-numeral* [*simp*]: $(\infty::enat) \neq numeral\ k$
⟨*proof*⟩

lemma *numeral-ne-infinity* [*simp*]: $numeral\ k \neq (\infty::enat)$
⟨*proof*⟩

lemma *eSuc-numeral* [*simp*]: *eSuc (numeral k) = numeral (k + Num.One)*
⟨*proof*⟩

34.6 Subtraction

instantiation *enat* :: *minus*
begin

definition *diff-enat-def*:
a − b = (case a of (enat x) ⇒ (case b of (enat y) ⇒ enat (x − y) | ∞ ⇒ 0)
| ∞ ⇒ ∞)

instance ⟨*proof*⟩

end

lemma *idiff-enat-enat* [*simp*, *code*]: *enat a − enat b = enat (a − b)*
⟨*proof*⟩

lemma *idiff-infinity* [*simp*, *code*]: $\infty - n = (\infty::enat)$
⟨*proof*⟩

lemma *idiff-infinity-right* [*simp*, *code*]: *enat a − ∞ = 0*

⟨*proof*⟩

lemma *idiff-0* [*simp*]: (*0*::*enat*) − *n* = *0*
⟨*proof*⟩

lemmas *idiff-enat-0* [*simp*] = *idiff-0* [*unfolded zero-enat-def*]

lemma *idiff-0-right* [*simp*]: (*n*::*enat*) − *0* = *n*
⟨*proof*⟩

lemmas *idiff-enat-0-right* [*simp*] = *idiff-0-right* [*unfolded zero-enat-def*]

lemma *idiff-self* [*simp*]: *n* ≠ ∞ ⟹ (*n*::*enat*) − *n* = *0*
⟨*proof*⟩

lemma *eSuc-minus-eSuc* [*simp*]: *eSuc n* − *eSuc m* = *n* − *m*
⟨*proof*⟩

lemma *eSuc-minus-1* [*simp*]: *eSuc n* − *1* = *n*
⟨*proof*⟩

34.7 Ordering

instantiation *enat* :: *linordered-ab-semigroup-add*
begin

definition [*nitpick-simp*]:
m ≤ *n* = (*case n of enat n1* ⇒ (*case m of enat m1* ⇒ *m1* ≤ *n1* | ∞ ⇒ *False*)
| ∞ ⇒ *True*)

definition [*nitpick-simp*]:
m < *n* = (*case m of enat m1* ⇒ (*case n of enat n1* ⇒ *m1* < *n1* | ∞ ⇒ *True*)
| ∞ ⇒ *False*)

lemma *enat-ord-simps* [*simp*]:
enat m ≤ *enat n* ⟷ *m* ≤ *n*
enat m < *enat n* ⟷ *m* < *n*
q ≤ (∞::*enat*)
q < (∞::*enat*) ⟷ *q* ≠ ∞
(∞::*enat*) ≤ *q* ⟷ *q* = ∞
(∞::*enat*) < *q* ⟷ *False*
⟨*proof*⟩

lemma *numeral-le-enat-iff* [*simp*]:
shows *numeral m* ≤ *enat n* ⟷ *numeral m* ≤ *n*
⟨*proof*⟩

lemma *numeral-less-enat-iff* [*simp*]:
shows *numeral m* < *enat n* ⟷ *numeral m* < *n*

⟨*proof*⟩

lemma *enat-ord-code* [*code*]:
enat m ≤ *enat n* ⟷ *m* ≤ *n*
enat m < *enat n* ⟷ *m* < *n*
q ≤ (∞::*enat*) ⟷ *True*
enat m < ∞ ⟷ *True*
∞ ≤ *enat n* ⟷ *False*
(∞::*enat*) < *q* ⟷ *False*
⟨*proof*⟩

instance
⟨*proof*⟩

end

instance *enat* :: *dioid*
⟨*proof*⟩

instance *enat* :: {*linordered-nonzero-semiring*, *strict-ordered-comm-monoid-add*}
⟨*proof*⟩

lemma *enat-ord-number* [*simp*]:
(*numeral m* :: *enat*) ≤ *numeral n* ⟷ (*numeral m* :: *nat*) ≤ *numeral n*
(*numeral m* :: *enat*) < *numeral n* ⟷ (*numeral m* :: *nat*) < *numeral n*
⟨*proof*⟩

lemma *infinity-ileE* [*elim*!]: ∞ ≤ *enat m* ⟹ *R*
⟨*proof*⟩

lemma *infinity-ilessE* [*elim*!]: ∞ < *enat m* ⟹ *R*
⟨*proof*⟩

lemma *eSuc-ile-mono* [*simp*]: *eSuc n* ≤ *eSuc m* ⟷ *n* ≤ *m*
⟨*proof*⟩

lemma *eSuc-mono* [*simp*]: *eSuc n* < *eSuc m* ⟷ *n* < *m*
⟨*proof*⟩

lemma *ile-eSuc* [*simp*]: *n* ≤ *eSuc n*
⟨*proof*⟩

lemma *not-eSuc-ilei0* [*simp*]: ¬ *eSuc n* ≤ *0*
⟨*proof*⟩

lemma *i0-iless-eSuc* [*simp*]: *0* < *eSuc n*
⟨*proof*⟩

lemma *iless-eSuc0*[*simp*]: $(n < eSuc\ 0) = (n = 0)$
⟨*proof*⟩

lemma *ileI1*: $m < n \Longrightarrow eSuc\ m \leq n$
⟨*proof*⟩

lemma *Suc-ile-eq*: $enat\ (Suc\ m) \leq n \longleftrightarrow enat\ m < n$
⟨*proof*⟩

lemma *iless-Suc-eq* [*simp*]: $enat\ m < eSuc\ n \longleftrightarrow enat\ m \leq n$
⟨*proof*⟩

lemma *imult-infinity*: $(0::enat) < n \Longrightarrow \infty * n = \infty$
⟨*proof*⟩

lemma *imult-infinity-right*: $(0::enat) < n \Longrightarrow n * \infty = \infty$
⟨*proof*⟩

lemma *enat-0-less-mult-iff*: $(0 < (m::enat) * n) = (0 < m \wedge 0 < n)$
⟨*proof*⟩

lemma *mono-eSuc*: *mono eSuc*
⟨*proof*⟩

lemma *min-enat-simps* [*simp*]:
$min\ (enat\ m)\ (enat\ n) = enat\ (min\ m\ n)$
$min\ q\ 0 = 0$
$min\ 0\ q = 0$
$min\ q\ (\infty::enat) = q$
$min\ (\infty::enat)\ q = q$
⟨*proof*⟩

lemma *max-enat-simps* [*simp*]:
$max\ (enat\ m)\ (enat\ n) = enat\ (max\ m\ n)$
$max\ q\ 0 = q$
$max\ 0\ q = q$
$max\ q\ \infty = (\infty::enat)$
$max\ \infty\ q = (\infty::enat)$
⟨*proof*⟩

lemma *enat-ile*: $n \leq enat\ m \Longrightarrow \exists\, k.\ n = enat\ k$
⟨*proof*⟩

lemma *enat-iless*: $n < enat\ m \Longrightarrow \exists\, k.\ n = enat\ k$
⟨*proof*⟩

lemma *iadd-le-enat-iff*:
$x + y \leq enat\ n \longleftrightarrow (\exists\, y'\ x'.\ x = enat\ x' \wedge y = enat\ y' \wedge x' + y' \leq n)$

⟨*proof*⟩

lemma *chain-incr*: $\forall i.\ \exists j.\ Y\ i < Y\ j \Longrightarrow \exists j.\ enat\ k < Y\ j$
⟨*proof*⟩

lemma *eSuc-max*: $eSuc\ (max\ x\ y) = max\ (eSuc\ x)\ (eSuc\ y)$
⟨*proof*⟩

lemma *eSuc-Max*:
assumes *finite A* $A \neq \{\}$
shows $eSuc\ (Max\ A) = Max\ (eSuc\ `\ A)$
⟨*proof*⟩

instantiation *enat* :: {*order-bot*, *order-top*}
begin

definition *bot-enat* :: *enat* **where** *bot-enat* = *0*
definition *top-enat* :: *enat* **where** *top-enat* = ∞

instance
⟨*proof*⟩

end

lemma *finite-enat-bounded*:
assumes *le-fin*: $\bigwedge y.\ y \in A \Longrightarrow y \leq enat\ n$
shows *finite A*
⟨*proof*⟩

34.8 Cancellation simprocs

lemma *enat-add-left-cancel*: $a + b = a + c \longleftrightarrow a = (\infty::enat) \vee b = c$
⟨*proof*⟩

lemma *enat-add-left-cancel-le*: $a + b \leq a + c \longleftrightarrow a = (\infty::enat) \vee b \leq c$
⟨*proof*⟩

lemma *enat-add-left-cancel-less*: $a + b < a + c \longleftrightarrow a \neq (\infty::enat) \wedge b < c$
⟨*proof*⟩

⟨*ML*⟩

TODO: add regression tests for these simprocs

TODO: add simprocs for combining and cancelling numerals

34.9 Well-ordering

lemma *less-enatE*:
[| $n < enat\ m$; !!k. $n = enat\ k$ ==> $k < m$ ==> P |] ==> P

⟨*proof*⟩

lemma *less-infinityE*:
[| $n < \infty$; !!k. n = *enat* k ==> P |] ==> P
⟨*proof*⟩

lemma *enat-less-induct*:
assumes *prem*: !!n. $\forall$ m::*enat*. $m < n$ −−> P m ==> P n **shows** P n
⟨*proof*⟩

instance *enat* :: *wellorder*
⟨*proof*⟩

34.10 Complete Lattice

instantiation *enat* :: *complete-lattice*
begin

definition *inf-enat* :: *enat* ⇒ *enat* ⇒ *enat* **where**
inf-enat = *min*

definition *sup-enat* :: *enat* ⇒ *enat* ⇒ *enat* **where**
sup-enat = *max*

definition *Inf-enat* :: *enat set* ⇒ *enat* **where**
Inf-enat A = (*if A* = {} *then* ∞ *else* (*LEAST x*. *x* ∈ *A*))

definition *Sup-enat* :: *enat set* ⇒ *enat* **where**
Sup-enat A = (*if A* = {} *then 0 else if finite A then Max A else* ∞)
instance
⟨*proof*⟩
end

instance *enat* :: *complete-linorder* ⟨*proof*⟩

lemma *eSuc-Sup*: $A \neq \{\} \Longrightarrow$ *eSuc* (*Sup A*) = *Sup* (*eSuc* ‘ *A*)
⟨*proof*⟩

lemma *sup-continuous-eSuc*: *sup-continuous f* ⟹ *sup-continuous* (λ*x*. *eSuc* (*f x*))
⟨*proof*⟩

34.11 Traditional theorem names

lemmas *enat-defs* = *zero-enat-def one-enat-def eSuc-def*
plus-enat-def less-eq-enat-def less-enat-def

lemma *iadd-is-0*: (*m* + *n* = (*0*::*enat*)) = (*m* = *0* ∧ *n* = *0*)
⟨*proof*⟩

lemma *i0-lb* : $(0::enat) \leq n$
⟨*proof*⟩

lemma *ile0-eq*: $n \leq (0::enat) \longleftrightarrow n = 0$
⟨*proof*⟩

lemma *not-iless0*: $\neg\ n < (0::enat)$
⟨*proof*⟩

lemma *i0-less*[*simp*]: $(0::enat) < n \longleftrightarrow n \neq 0$
⟨*proof*⟩

lemma *imult-is-0*: $((m::enat) * n = 0) = (m = 0 \vee n = 0)$
⟨*proof*⟩

end

35 Liminf and Limsup on conditionally complete lattices

theory *Liminf-Limsup*
imports *Complex-Main*
begin

lemma (**in** *conditionally-complete-linorder*) *le-cSup-iff*:
assumes $A \neq \{\}$ *bdd-above* A
shows $x \leq Sup\ A \longleftrightarrow (\forall\, y{<}x.\ \exists\, a{\in}A.\ y < a)$
⟨*proof*⟩

lemma (**in** *conditionally-complete-linorder*) *le-cSUP-iff*:
$A \neq \{\} \Longrightarrow$ *bdd-above* $(f`A) \Longrightarrow x \leq SUPREMUM\ A\ f \longleftrightarrow (\forall\, y{<}x.\ \exists\, i{\in}A.\ y < f\ i)$
⟨*proof*⟩

lemma *le-cSup-iff-less*:
fixes $x :: 'a :: \{$*conditionally-complete-linorder*, *dense-linorder*$\}$
shows $A \neq \{\} \Longrightarrow$ *bdd-above* $(f`A) \Longrightarrow x \leq (SUP\ i{:}A.\ f\ i) \longleftrightarrow (\forall\, y{<}x.\ \exists\, i{\in}A.\ y \leq f\ i)$
⟨*proof*⟩

lemma *le-Sup-iff-less*:
fixes $x :: 'a :: \{$*complete-linorder*, *dense-linorder*$\}$
shows $x \leq (SUP\ i{:}A.\ f\ i) \longleftrightarrow (\forall\, y{<}x.\ \exists\, i{\in}A.\ y \leq f\ i)$ (**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma (**in** *conditionally-complete-linorder*) *cInf-le-iff*:
assumes $A \neq \{\}$ *bdd-below* A
shows $Inf\ A \leq x \longleftrightarrow (\forall\, y{>}x.\ \exists\, a{\in}A.\ y > a)$

⟨*proof*⟩

lemma (**in** *conditionally-complete-linorder*) *cINF-le-iff*:
  $A \neq \{\} \Longrightarrow$ *bdd-below* (f`A) $\Longrightarrow$ *INFIMUM* A $f \leq x \longleftrightarrow (\forall y{>}x.\ \exists i{\in}A.\ y > f\ i)$
  ⟨*proof*⟩

lemma *cInf-le-iff-less*:
  fixes $x :: 'a :: \{$*conditionally-complete-linorder*, *dense-linorder*$\}$
  shows $A \neq \{\} \Longrightarrow$ *bdd-below* (f`A) $\Longrightarrow$ (*INF* $i{:}A.\ f\ i) \leq x \longleftrightarrow (\forall y{>}x.\ \exists i{\in}A.\ f\ i \leq y)$
  ⟨*proof*⟩

lemma *Inf-le-iff-less*:
  fixes $x :: 'a :: \{$*complete-linorder*, *dense-linorder*$\}$
  shows (*INF* $i{:}A.\ f\ i) \leq x \longleftrightarrow (\forall y{>}x.\ \exists i{\in}A.\ f\ i \leq y)$
  ⟨*proof*⟩

lemma *SUP-pair*:
  fixes f :: - ⇒ - ⇒ - :: *complete-lattice*
  shows (*SUP* $i : A.$ *SUP* $j : B.\ f\ i\ j) =$ (*SUP* $p : A \times B.\ f$ (*fst* p) (*snd* p))
  ⟨*proof*⟩

lemma *INF-pair*:
  fixes f :: - ⇒ - ⇒ - :: *complete-lattice*
  shows (*INF* $i : A.$ *INF* $j : B.\ f\ i\ j) =$ (*INF* $p : A \times B.\ f$ (*fst* p) (*snd* p))
  ⟨*proof*⟩

35.0.1 *Liminf* and *Limsup*

definition *Liminf* :: $'a$ *filter* ⇒ ($'a$ ⇒ $'b$) ⇒ $'b$ :: *complete-lattice* **where**
  Liminf $F\ f =$ (*SUP* $P{:}\{P.$ *eventually* $P\ F\}.$ *INF* $x{:}\{x.\ P\ x\}.\ f\ x$)

definition *Limsup* :: $'a$ *filter* ⇒ ($'a$ ⇒ $'b$) ⇒ $'b$ :: *complete-lattice* **where**
  Limsup $F\ f =$ (*INF* $P{:}\{P.$ *eventually* $P\ F\}.$ *SUP* $x{:}\{x.\ P\ x\}.\ f\ x$)

abbreviation *liminf* ≡ *Liminf sequentially*

abbreviation *limsup* ≡ *Limsup sequentially*

lemma *Liminf-eqI*:
  ($\bigwedge P.$ *eventually* $P\ F \Longrightarrow$ *INFIMUM* (*Collect* P) $f \leq x) \Longrightarrow$
    ($\bigwedge y.$ ($\bigwedge P.$ *eventually* $P\ F \Longrightarrow$ *INFIMUM* (*Collect* P) $f \leq y) \Longrightarrow x \leq y) \Longrightarrow$ *Liminf* $F\ f = x$
  ⟨*proof*⟩

lemma *Limsup-eqI*:
  ($\bigwedge P.$ *eventually* $P\ F \Longrightarrow x \leq$ *SUPREMUM* (*Collect* P) $f) \Longrightarrow$
    ($\bigwedge y.$ ($\bigwedge P.$ *eventually* $P\ F \Longrightarrow y \leq$ *SUPREMUM* (*Collect* P) $f) \Longrightarrow y \leq x$)

$\Longrightarrow$ *Limsup F f = x*
⟨*proof*⟩

lemma *liminf-SUP-INF*: *liminf f = (SUP n. INF m:{n..}. f m)*
⟨*proof*⟩

lemma *limsup-INF-SUP*: *limsup f = (INF n. SUP m:{n..}. f m)*
⟨*proof*⟩

lemma *Limsup-const*:
assumes *ntriv*: ¬ *trivial-limit F*
shows *Limsup F (λx. c) = c*
⟨*proof*⟩

lemma *Liminf-const*:
assumes *ntriv*: ¬ *trivial-limit F*
shows *Liminf F (λx. c) = c*
⟨*proof*⟩

lemma *Liminf-mono*:
assumes *ev*: *eventually (λx. f x ≤ g x) F*
shows *Liminf F f ≤ Liminf F g*
⟨*proof*⟩

lemma *Liminf-eq*:
assumes *eventually (λx. f x = g x) F*
shows *Liminf F f = Liminf F g*
⟨*proof*⟩

lemma *Limsup-mono*:
assumes *ev*: *eventually (λx. f x ≤ g x) F*
shows *Limsup F f ≤ Limsup F g*
⟨*proof*⟩

lemma *Limsup-eq*:
assumes *eventually (λx. f x = g x) net*
shows *Limsup net f = Limsup net g*
⟨*proof*⟩

lemma *Liminf-le-Limsup*:
assumes *ntriv*: ¬ *trivial-limit F*
shows *Liminf F f ≤ Limsup F f*
⟨*proof*⟩

lemma *Liminf-bounded*:
assumes *ntriv*: ¬ *trivial-limit F*
assumes *le*: *eventually (λn. C ≤ X n) F*
shows *C ≤ Liminf F X*
⟨*proof*⟩

lemma *Limsup-bounded*:
assumes *ntriv*: ¬ *trivial-limit F*
assumes *le*: *eventually* (λ*n*. *X n* ≤ *C*) *F*
shows *Limsup F X* ≤ *C*
⟨*proof*⟩

lemma *le-Limsup*:
assumes *F*: *F* ≠ *bot* **and** *x*: ∀ *x in F*. *l* ≤ *f x*
shows *l* ≤ *Limsup F f*
⟨*proof*⟩

lemma *le-Liminf-iff*:
fixes *X* :: - ⇒ - :: *complete-linorder*
shows *C* ≤ *Liminf F X* ⟷ (∀ *y*<*C*. *eventually* (λ*x*. *y* < *X x*) *F*)
⟨*proof*⟩

lemma *Limsup-le-iff*:
fixes *X* :: - ⇒ - :: *complete-linorder*
shows *C* ≥ *Limsup F X* ⟷ (∀ *y*>*C*. *eventually* (λ*x*. *y* > *X x*) *F*)
⟨*proof*⟩

lemma *less-LiminfD*:
y < *Liminf F* (*f* :: - ⇒ *'a* :: *complete-linorder*) ⟹ *eventually* (λ*x*. *f x* > *y*) *F*
⟨*proof*⟩

lemma *Limsup-lessD*:
y > *Limsup F* (*f* :: - ⇒ *'a* :: *complete-linorder*) ⟹ *eventually* (λ*x*. *f x* < *y*) *F*
⟨*proof*⟩

lemma *lim-imp-Liminf*:
fixes *f* :: *'a* ⇒ - :: {*complete-linorder*,*linorder-topology*}
assumes *ntriv*: ¬ *trivial-limit F*
assumes *lim*: (*f* ⟶ *f0*) *F*
shows *Liminf F f* = *f0*
⟨*proof*⟩

lemma *lim-imp-Limsup*:
fixes *f* :: *'a* ⇒ - :: {*complete-linorder*,*linorder-topology*}
assumes *ntriv*: ¬ *trivial-limit F*
assumes *lim*: (*f* ⟶ *f0*) *F*
shows *Limsup F f* = *f0*
⟨*proof*⟩

lemma *Liminf-eq-Limsup*:
fixes *f0* :: *'a* :: {*complete-linorder*,*linorder-topology*}
assumes *ntriv*: ¬ *trivial-limit F*
and *lim*: *Liminf F f* = *f0 Limsup F f* = *f0*
shows (*f* ⟶ *f0*) *F*

⟨*proof*⟩

lemma *tendsto-iff-Liminf-eq-Limsup*:
fixes *f0* :: *'a* :: {*complete-linorder*,*linorder-topology*}
shows ¬ *trivial-limit F* ⟹ (*f* ⟶ *f0*) *F* ⟷ (*Liminf F f* = *f0* ∧ *Limsup F f* = *f0*)
⟨*proof*⟩

lemma *liminf-subseq-mono*:
fixes *X* :: *nat* ⇒ *'a* :: *complete-linorder*
assumes *subseq r*
shows *liminf X* ≤ *liminf* (*X* ∘ *r*)
⟨*proof*⟩

lemma *limsup-subseq-mono*:
fixes *X* :: *nat* ⇒ *'a* :: *complete-linorder*
assumes *subseq r*
shows *limsup* (*X* ∘ *r*) ≤ *limsup X*
⟨*proof*⟩

lemma *continuous-on-imp-continuous-within*:
continuous-on s f ⟹ *t* ⊆ *s* ⟹ *x* ∈ *s* ⟹ *continuous* (*at x within t*) *f*
⟨*proof*⟩

lemma *Liminf-compose-continuous-mono*:
fixes *f* :: *'a*::{*complete-linorder*, *linorder-topology*} ⇒ *'b*::{*complete-linorder*, *linorder-topology*}
assumes *c*: *continuous-on UNIV f* **and** *am*: *mono f* **and** *F*: *F* ≠ *bot*
shows *Liminf F* (λ*n*. *f* (*g n*)) = *f* (*Liminf F g*)
⟨*proof*⟩

lemma *Limsup-compose-continuous-mono*:
fixes *f* :: *'a*::{*complete-linorder*, *linorder-topology*} ⇒ *'b*::{*complete-linorder*, *linorder-topology*}
assumes *c*: *continuous-on UNIV f* **and** *am*: *mono f* **and** *F*: *F* ≠ *bot*
shows *Limsup F* (λ*n*. *f* (*g n*)) = *f* (*Limsup F g*)
⟨*proof*⟩

lemma *Liminf-compose-continuous-antimono*:
fixes *f* :: *'a*::{*complete-linorder*,*linorder-topology*} ⇒ *'b*::{*complete-linorder*,*linorder-topology*}
assumes *c*: *continuous-on UNIV f*
and *am*: *antimono f*
and *F*: *F* ≠ *bot*
shows *Liminf F* (λ*n*. *f* (*g n*)) = *f* (*Limsup F g*)
⟨*proof*⟩

lemma *Limsup-compose-continuous-antimono*:
fixes *f* :: *'a*::{*complete-linorder*, *linorder-topology*} ⇒ *'b*::{*complete-linorder*, *linorder-topology*}

assumes *c*: *continuous-on UNIV f* **and** *am*: *antimono f* **and** *F*: $F \neq bot$
shows *Limsup F* $(\lambda n.\ f\ (g\ n)) = f\ (Liminf\ F\ g)$
⟨*proof*⟩

35.1 More Limits

lemma *convergent-limsup-cl*:
fixes $X :: nat \Rightarrow 'a$::{*complete-linorder*,*linorder-topology*}
shows *convergent X* $\Longrightarrow$ *limsup X* = *lim X*
⟨*proof*⟩

lemma *convergent-liminf-cl*:
fixes $X :: nat \Rightarrow 'a$::{*complete-linorder*,*linorder-topology*}
shows *convergent X* $\Longrightarrow$ *liminf X* = *lim X*
⟨*proof*⟩

lemma *lim-increasing-cl*:
assumes $\bigwedge n\ m.\ n \geq m \Longrightarrow f\ n \geq f\ m$
obtains *l* **where** $f \longrightarrow (l::'a::\{complete\text{-}linorder,linorder\text{-}topology\})$
⟨*proof*⟩

lemma *lim-decreasing-cl*:
assumes $\bigwedge n\ m.\ n \geq m \Longrightarrow f\ n \leq f\ m$
obtains *l* **where** $f \longrightarrow (l::'a::\{complete\text{-}linorder,linorder\text{-}topology\})$
⟨*proof*⟩

lemma *compact-complete-linorder*:
fixes $X :: nat \Rightarrow 'a$::{*complete-linorder*,*linorder-topology*}
shows $\exists\, l\ r.\ subseq\ r \wedge (X \circ r) \longrightarrow l$
⟨*proof*⟩

lemma *tendsto-Limsup*:
fixes $f :: - \Rightarrow 'a :: \{complete\text{-}linorder,linorder\text{-}topology\}$
shows $F \neq bot \Longrightarrow Limsup\ F\ f = Liminf\ F\ f \Longrightarrow (f \longrightarrow Limsup\ F\ f)\ F$
⟨*proof*⟩

lemma *tendsto-Liminf*:
fixes $f :: - \Rightarrow 'a :: \{complete\text{-}linorder,linorder\text{-}topology\}$
shows $F \neq bot \Longrightarrow Limsup\ F\ f = Liminf\ F\ f \Longrightarrow (f \longrightarrow Liminf\ F\ f)\ F$
⟨*proof*⟩

end

36 Extended real number line

theory *Extended-Real*
imports *Complex-Main Extended-Nat Liminf-Limsup*
begin

This should be part of *Extended-Nat* or *Order-Continuity*, but then the AFP-entry *Jinja-Thread* fails, as it does overload certain named from *Complex-Main*.

lemma *incseq-setsumI2*:
fixes $f :: 'i \Rightarrow nat \Rightarrow 'a$::*ordered-comm-monoid-add*
shows $(\bigwedge n.\ n \in A \Longrightarrow mono\ (f\ n)) \Longrightarrow mono\ (\lambda i.\ \sum n{\in}A.\ f\ n\ i)$
⟨*proof*⟩

lemma *incseq-setsumI*:
fixes $f :: nat \Rightarrow 'a$::*ordered-comm-monoid-add*
assumes $\bigwedge i.\ 0 \leq f\ i$
shows *incseq* $(\lambda i.\ setsum\ f\ \{..< i\})$
⟨*proof*⟩

lemma *continuous-at-left-imp-sup-continuous*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*} $\Rightarrow$ $'b$::{*complete-linorder*, *linorder-topology*}
assumes *mono* f $\bigwedge x.$ *continuous* (*at-left* x) f
shows *sup-continuous* f
⟨*proof*⟩

lemma *sup-continuous-at-left*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*, *first-countable-topology*} $\Rightarrow$
$'b$::{*complete-linorder*, *linorder-topology*}
assumes f: *sup-continuous* f
shows *continuous* (*at-left* x) f
⟨*proof*⟩

lemma *sup-continuous-iff-at-left*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*, *first-countable-topology*} $\Rightarrow$
$'b$::{*complete-linorder*, *linorder-topology*}
shows *sup-continuous* $f \longleftrightarrow (\forall x.$ *continuous* (*at-left* x) $f) \wedge$ *mono* f
⟨*proof*⟩

lemma *continuous-at-right-imp-inf-continuous*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*} $\Rightarrow$ $'b$::{*complete-linorder*, *linorder-topology*}
assumes *mono* f $\bigwedge x.$ *continuous* (*at-right* x) f
shows *inf-continuous* f
⟨*proof*⟩

lemma *inf-continuous-at-right*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*, *first-countable-topology*} $\Rightarrow$
$'b$::{*complete-linorder*, *linorder-topology*}
assumes f: *inf-continuous* f
shows *continuous* (*at-right* x) f
⟨*proof*⟩

lemma *inf-continuous-iff-at-right*:
fixes $f :: 'a$::{*complete-linorder*, *linorder-topology*, *first-countable-topology*} $\Rightarrow$

'b::{*complete-linorder*, *linorder-topology*}
shows *inf-continuous f* ⟷ (∀ *x*. *continuous* (*at-right x*) *f*) ∧ *mono f*
⟨*proof*⟩

instantiation *enat* :: *linorder-topology*
begin

definition *open-enat* :: *enat set* ⇒ *bool* **where**
open-enat = *generate-topology* (*range lessThan* ∪ *range greaterThan*)

instance
⟨*proof*⟩

end

lemma *open-enat*: *open* {*enat n*}
⟨*proof*⟩

lemma *open-enat-iff*:
fixes *A* :: *enat set*
shows *open A* ⟷ (∞ ∈ *A* ⟶ (∃ *n*::*nat*. {*n* <..} ⊆ *A*))
⟨*proof*⟩

lemma *nhds-enat*: *nhds x* = (*if x* = ∞ *then INF i*. *principal* {*enat i*..} *else principal* {*x*})
⟨*proof*⟩

instance *enat* :: *topological-comm-monoid-add*
⟨*proof*⟩

For more lemmas about the extended real numbers go to `~~/src/HOL/Multivariate_Analysis/Extended_Real_Limits.thy`

36.1 Definition and basic properties

datatype *ereal* = *ereal real* | *PInfty* | *MInfty*

instantiation *ereal* :: *uminus*
begin

fun *uminus-ereal* **where**
− (*ereal r*) = *ereal* (− *r*)
| − *PInfty* = *MInfty*
| − *MInfty* = *PInfty*

instance ⟨*proof*⟩

end

instantiation *ereal* :: *infinity*
begin

definition (∞::*ereal*) = *PInfty*
instance ⟨*proof*⟩

end

declare [[*coercion ereal* :: *real* ⇒ *ereal*]]

lemma *ereal-uminus-uminus*[*simp*]:
fixes *a* :: *ereal*
shows − (− *a*) = *a*
⟨*proof*⟩

lemma
shows *PInfty-eq-infinity*[*simp*]: *PInfty* = ∞
and *MInfty-eq-minfinity*[*simp*]: *MInfty* = − ∞
and *MInfty-neq-PInfty*[*simp*]: ∞ ≠ − (∞::*ereal*) − ∞ ≠ (∞::*ereal*)
and *MInfty-neq-ereal*[*simp*]: *ereal r* ≠ − ∞ − ∞ ≠ *ereal r*
and *PInfty-neq-ereal*[*simp*]: *ereal r* ≠ ∞ ∞ ≠ *ereal r*
and *PInfty-cases*[*simp*]: (*case* ∞ *of ereal r* ⇒ *f r* | *PInfty* ⇒ *y* | *MInfty* ⇒ *z*) = *y*
and *MInfty-cases*[*simp*]: (*case* − ∞ *of ereal r* ⇒ *f r* | *PInfty* ⇒ *y* | *MInfty* ⇒ *z*) = *z*
⟨*proof*⟩

declare
PInfty-eq-infinity[*code-post*]
MInfty-eq-minfinity[*code-post*]

lemma [*code-unfold*]:
∞ = *PInfty*
− *PInfty* = *MInfty*
⟨*proof*⟩

lemma *inj-ereal*[*simp*]: *inj-on ereal A*
⟨*proof*⟩

lemma *ereal-cases*[*cases type*: *ereal*]:
obtains (*real*) *r* **where** *x* = *ereal r*
| (*PInf*) *x* = ∞
| (*MInf*) *x* = −∞
⟨*proof*⟩

lemmas *ereal2-cases* = *ereal-cases*[*case-product ereal-cases*]
lemmas *ereal3-cases* = *ereal2-cases*[*case-product ereal-cases*]

lemma *ereal-all-split*: ⋀*P*. (∀ *x*::*ereal*. *P x*) ⟷ *P* ∞ ∧ (∀ *x*. *P* (*ereal x*)) ∧ *P*

$(-\infty)$
⟨*proof*⟩

lemma *ereal-ex-split*: $\bigwedge P.\ (\exists\, x{::}ereal.\ P\ x) \longleftrightarrow P\ \infty \vee (\exists\, x.\ P\ (ereal\ x)) \vee P\ (-\infty)$
⟨*proof*⟩

lemma *ereal-uminus-eq-iff*[*simp*]:
fixes *a b* :: *ereal*
shows $-a = -b \longleftrightarrow a = b$
⟨*proof*⟩

function *real-of-ereal* :: *ereal* ⇒ *real* **where**
real-of-ereal (*ereal r*) = *r*
| *real-of-ereal* ∞ = *0*
| *real-of-ereal* (−∞) = *0*
⟨*proof*⟩
termination ⟨*proof*⟩

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

lemma *range-ereal*[*simp*]: *range ereal* = *UNIV* − {∞, −∞}
⟨*proof*⟩

lemma *ereal-range-uminus*[*simp*]: *range uminus* = (*UNIV*::*ereal set*)
⟨*proof*⟩

instantiation *ereal* :: *abs*
begin

function *abs-ereal* **where**
|*ereal r*| = *ereal* |*r*|
| |−∞| = (∞::*ereal*)
| |∞| = (∞::*ereal*)
⟨*proof*⟩
termination ⟨*proof*⟩

instance ⟨*proof*⟩

end

lemma *abs-eq-infinity-cases*[*elim*!]:
fixes *x* :: *ereal*
assumes |*x*| = ∞
obtains *x* = ∞ | *x* = −∞
⟨*proof*⟩

lemma *abs-neq-infinity-cases*[*elim*!]:
fixes x :: *ereal*
assumes $|x| \neq \infty$
obtains r **where** $x = ereal\ r$
⟨*proof*⟩

lemma *abs-ereal-uminus*[*simp*]:
fixes x :: *ereal*
shows $|-x| = |x|$
⟨*proof*⟩

lemma *ereal-infinity-cases*:
fixes a :: *ereal*
shows $a \neq \infty \Longrightarrow a \neq -\infty \Longrightarrow |a| \neq \infty$
⟨*proof*⟩

36.1.1 Addition

instantiation *ereal* :: {*one*,*comm-monoid-add*,*zero-neq-one*}
begin

definition $0 = ereal\ 0$
definition $1 = ereal\ 1$

function *plus-ereal* **where**
$ereal\ r + ereal\ p = ereal\ (r + p)$
$| \infty + a = (\infty::ereal)$
$| a + \infty = (\infty::ereal)$
$| ereal\ r + -\infty = -\infty$
$| -\infty + ereal\ p = -(\infty::ereal)$
$| -\infty + -\infty = -(\infty::ereal)$
⟨*proof*⟩
termination ⟨*proof*⟩

lemma *Infty-neq-0*[*simp*]:
$(\infty::ereal) \neq 0\ 0 \neq (\infty::ereal)$
$-(\infty::ereal) \neq 0\ 0 \neq -(\infty::ereal)$
⟨*proof*⟩

lemma *ereal-eq-0*[*simp*]:
$ereal\ r = 0 \longleftrightarrow r = 0$
$0 = ereal\ r \longleftrightarrow r = 0$
⟨*proof*⟩

lemma *ereal-eq-1*[*simp*]:
$ereal\ r = 1 \longleftrightarrow r = 1$
$1 = ereal\ r \longleftrightarrow r = 1$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *ereal-0-plus* [*simp*]: *ereal 0 + x = x*
and *plus-ereal-0* [*simp*]: *x + ereal 0 = x*
⟨*proof*⟩

instance *ereal* :: *numeral* ⟨*proof*⟩

lemma *real-of-ereal-0*[*simp*]: *real-of-ereal* (*0*::*ereal*) = *0*
⟨*proof*⟩

lemma *abs-ereal-zero*[*simp*]: |*0*| = (*0*::*ereal*)
⟨*proof*⟩

lemma *ereal-uminus-zero*[*simp*]: − *0* = (*0*::*ereal*)
⟨*proof*⟩

lemma *ereal-uminus-zero-iff*[*simp*]:
fixes *a* :: *ereal*
shows −*a* = *0* ⟷ *a* = *0*
⟨*proof*⟩

lemma *ereal-plus-eq-PInfty*[*simp*]:
fixes *a b* :: *ereal*
shows *a* + *b* = ∞ ⟷ *a* = ∞ ∨ *b* = ∞
⟨*proof*⟩

lemma *ereal-plus-eq-MInfty*[*simp*]:
fixes *a b* :: *ereal*
shows *a* + *b* = −∞ ⟷ (*a* = −∞ ∨ *b* = −∞) ∧ *a* ≠ ∞ ∧ *b* ≠ ∞
⟨*proof*⟩

lemma *ereal-add-cancel-left*:
fixes *a b* :: *ereal*
assumes *a* ≠ −∞
shows *a* + *b* = *a* + *c* ⟷ *a* = ∞ ∨ *b* = *c*
⟨*proof*⟩

lemma *ereal-add-cancel-right*:
fixes *a b* :: *ereal*
assumes *a* ≠ −∞
shows *b* + *a* = *c* + *a* ⟷ *a* = ∞ ∨ *b* = *c*
⟨*proof*⟩

lemma *ereal-real*: *ereal* (*real-of-ereal x*) = (*if* |*x*| = ∞ *then 0 else x*)
⟨*proof*⟩

lemma *real-of-ereal-add*:
fixes $a\ b :: ereal$
shows *real-of-ereal* $(a + b) =$
$(if\ (|a| = \infty) \wedge (|b| = \infty) \vee (|a| \neq \infty) \wedge (|b| \neq \infty)\ then\ real\text{-}of\text{-}ereal\ a +$
real-of-ereal b else 0)
⟨*proof*⟩

36.1.2 Linear order on *ereal*

instantiation *ereal* :: *linorder*
begin

function *less-ereal*
where
$ereal\ x < ereal\ y \longleftrightarrow x < y$
$| (\infty::ereal) < a \longleftrightarrow False$
$| a < -(\infty::ereal) \longleftrightarrow False$
$| ereal\ x < \infty \longleftrightarrow True$
$| -\infty < ereal\ r \longleftrightarrow True$
$| -\infty < (\infty::ereal) \longleftrightarrow True$
⟨*proof*⟩
termination ⟨*proof*⟩

definition $x \leq (y::ereal) \longleftrightarrow x < y \vee x = y$

lemma *ereal-infty-less*[*simp*]:
fixes $x :: ereal$
shows $x < \infty \longleftrightarrow (x \neq \infty)$
$-\infty < x \longleftrightarrow (x \neq -\infty)$
⟨*proof*⟩

lemma *ereal-infty-less-eq*[*simp*]:
fixes $x :: ereal$
shows $\infty \leq x \longleftrightarrow x = \infty$
and $x \leq -\infty \longleftrightarrow x = -\infty$
⟨*proof*⟩

lemma *ereal-less*[*simp*]:
$ereal\ r < 0 \longleftrightarrow (r < 0)$
$0 < ereal\ r \longleftrightarrow (0 < r)$
$ereal\ r < 1 \longleftrightarrow (r < 1)$
$1 < ereal\ r \longleftrightarrow (1 < r)$
$0 < (\infty::ereal)$
$-(\infty::ereal) < 0$
⟨*proof*⟩

lemma *ereal-less-eq*[*simp*]:
$x \leq (\infty::ereal)$

$-(\infty::ereal) \leq x$
$ereal\ r \leq ereal\ p \longleftrightarrow r \leq p$
$ereal\ r \leq 0 \longleftrightarrow r \leq 0$
$0 \leq ereal\ r \longleftrightarrow 0 \leq r$
$ereal\ r \leq 1 \longleftrightarrow r \leq 1$
$1 \leq ereal\ r \longleftrightarrow 1 \leq r$
⟨*proof*⟩

lemma *ereal-infty-less-eq2*:
$a \leq b \Longrightarrow a = \infty \Longrightarrow b = (\infty::ereal)$
$a \leq b \Longrightarrow b = -\infty \Longrightarrow a = -(\infty::ereal)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *ereal-dense2*: $x < y \Longrightarrow \exists\, z.\ x < ereal\ z \wedge ereal\ z < y$
⟨*proof*⟩

instance *ereal* :: *dense-linorder*
⟨*proof*⟩

instance *ereal* :: *ordered-comm-monoid-add*
⟨*proof*⟩

lemma *ereal-one-not-less-zero-ereal*[*simp*]: $\neg\ 1 < (0::ereal)$
⟨*proof*⟩

lemma *real-of-ereal-positive-mono*:
fixes $x\ y$ :: *ereal*
shows $0 \leq x \Longrightarrow x \leq y \Longrightarrow y \neq \infty \Longrightarrow real\text{-}of\text{-}ereal\ x \leq real\text{-}of\text{-}ereal\ y$
⟨*proof*⟩

lemma *ereal-MInfty-lessI*[*intro*, *simp*]:
fixes a :: *ereal*
shows $a \neq -\infty \Longrightarrow -\infty < a$
⟨*proof*⟩

lemma *ereal-less-PInfty*[*intro*, *simp*]:
fixes a :: *ereal*
shows $a \neq \infty \Longrightarrow a < \infty$
⟨*proof*⟩

lemma *ereal-less-ereal-Ex*:
fixes $a\ b$ :: *ereal*
shows $x < ereal\ r \longleftrightarrow x = -\infty \vee (\exists\, p.\ p < r \wedge x = ereal\ p)$
⟨*proof*⟩

lemma *less-PInf-Ex-of-nat*: $x \neq \infty \longleftrightarrow (\exists\, n{::}nat.\ x < ereal\ (real\ n))$
⟨*proof*⟩

lemma *ereal-add-mono*:
fixes $a\ b\ c\ d :: ereal$
assumes $a \leq b$
and $c \leq d$
shows $a + c \leq b + d$
⟨*proof*⟩

lemma *ereal-minus-le-minus*[*simp*]:
fixes $a\ b :: ereal$
shows $- a \leq - b \longleftrightarrow b \leq a$
⟨*proof*⟩

lemma *ereal-minus-less-minus*[*simp*]:
fixes $a\ b :: ereal$
shows $- a < - b \longleftrightarrow b < a$
⟨*proof*⟩

lemma *ereal-le-real-iff*:
$x \leq real\text{-}of\text{-}ereal\ y \longleftrightarrow (|y| \neq \infty \longrightarrow ereal\ x \leq y) \wedge (|y| = \infty \longrightarrow x \leq 0)$
⟨*proof*⟩

lemma *real-le-ereal-iff*:
$real\text{-}of\text{-}ereal\ y \leq x \longleftrightarrow (|y| \neq \infty \longrightarrow y \leq ereal\ x) \wedge (|y| = \infty \longrightarrow 0 \leq x)$
⟨*proof*⟩

lemma *ereal-less-real-iff*:
$x < real\text{-}of\text{-}ereal\ y \longleftrightarrow (|y| \neq \infty \longrightarrow ereal\ x < y) \wedge (|y| = \infty \longrightarrow x < 0)$
⟨*proof*⟩

lemma *real-less-ereal-iff*:
$real\text{-}of\text{-}ereal\ y < x \longleftrightarrow (|y| \neq \infty \longrightarrow y < ereal\ x) \wedge (|y| = \infty \longrightarrow 0 < x)$
⟨*proof*⟩

lemma *real-of-ereal-pos*:
fixes $x :: ereal$
shows $0 \leq x \Longrightarrow 0 \leq real\text{-}of\text{-}ereal\ x$ ⟨*proof*⟩

lemmas *real-of-ereal-ord-simps* =
ereal-le-real-iff real-le-ereal-iff ereal-less-real-iff real-less-ereal-iff

lemma *abs-ereal-ge0*[*simp*]: $0 \leq x \Longrightarrow |x :: ereal| = x$
⟨*proof*⟩

lemma *abs-ereal-less0*[*simp*]: $x < 0 \Longrightarrow |x :: ereal| = -x$
⟨*proof*⟩

lemma *abs-ereal-pos*[*simp*]: $0 \leq |x :: ereal|$
⟨*proof*⟩

lemma *ereal-abs-leI*:
fixes $x\ y :: ereal$
shows $[\![x \leq y; -x \leq y]\!] \Longrightarrow |x| \leq y$
⟨*proof*⟩

lemma *real-of-ereal-le-0*[*simp*]: *real-of-ereal* $(x :: ereal) \leq 0 \longleftrightarrow x \leq 0 \vee x = \infty$
⟨*proof*⟩

lemma *abs-real-of-ereal*[*simp*]: $|$*real-of-ereal* $(x :: ereal)| =$ *real-of-ereal* $|x|$
⟨*proof*⟩

lemma *zero-less-real-of-ereal*:
fixes $x :: ereal$
shows $0 <$ *real-of-ereal* $x \longleftrightarrow 0 < x \wedge x \neq \infty$
⟨*proof*⟩

lemma *ereal-0-le-uminus-iff*[*simp*]:
fixes $a :: ereal$
shows $0 \leq -\ a \longleftrightarrow a \leq 0$
⟨*proof*⟩

lemma *ereal-uminus-le-0-iff*[*simp*]:
fixes $a :: ereal$
shows $-\ a \leq 0 \longleftrightarrow 0 \leq a$
⟨*proof*⟩

lemma *ereal-add-strict-mono*:
fixes $a\ b\ c\ d :: ereal$
assumes $a \leq b$
and $0 \leq a$
and $a \neq \infty$
and $c < d$
shows $a + c < b + d$
⟨*proof*⟩

lemma *ereal-less-add*:
fixes $a\ b\ c :: ereal$
shows $|a| \neq \infty \Longrightarrow c < b \Longrightarrow a + c < a + b$
⟨*proof*⟩

lemma *ereal-add-nonneg-eq-0-iff*:
fixes $a\ b :: ereal$
shows $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow a + b = 0 \longleftrightarrow a = 0 \wedge b = 0$
⟨*proof*⟩

lemma *ereal-uminus-eq-reorder*: $- a = b \longleftrightarrow a = (-b::ereal)$
⟨*proof*⟩

lemma *ereal-uminus-less-reorder*: $- a < b \longleftrightarrow -b < (a::ereal)$
⟨*proof*⟩

lemma *ereal-less-uminus-reorder*: $a < - b \longleftrightarrow b < - (a::ereal)$
⟨*proof*⟩

lemma *ereal-uminus-le-reorder*: $- a \leq b \longleftrightarrow -b \leq (a::ereal)$
⟨*proof*⟩

lemmas *ereal-uminus-reorder* =
ereal-uminus-eq-reorder ereal-uminus-less-reorder ereal-uminus-le-reorder

lemma *ereal-bot*:
fixes $x :: ereal$
assumes $\bigwedge B.\ x \leq ereal\ B$
shows $x = - \infty$
⟨*proof*⟩

lemma *ereal-top*:
fixes $x :: ereal$
assumes $\bigwedge B.\ x \geq ereal\ B$
shows $x = \infty$
⟨*proof*⟩

lemma
shows *ereal-max*[*simp*]: $ereal\ (max\ x\ y) = max\ (ereal\ x)\ (ereal\ y)$
and *ereal-min*[*simp*]: $ereal\ (min\ x\ y) = min\ (ereal\ x)\ (ereal\ y)$
⟨*proof*⟩

lemma *ereal-max-0*: $max\ 0\ (ereal\ r) = ereal\ (max\ 0\ r)$
⟨*proof*⟩

lemma
fixes $f :: nat \Rightarrow ereal$
shows *ereal-incseq-uminus*[*simp*]: $incseq\ (\lambda x.\ - f\ x) \longleftrightarrow decseq\ f$
and *ereal-decseq-uminus*[*simp*]: $decseq\ (\lambda x.\ - f\ x) \longleftrightarrow incseq\ f$
⟨*proof*⟩

lemma *incseq-ereal*: $incseq\ f \Longrightarrow incseq\ (\lambda x.\ ereal\ (f\ x))$
⟨*proof*⟩

lemma *ereal-add-nonneg-nonneg*[*simp*]:
fixes $a\ b :: ereal$
shows $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow 0 \leq a + b$
⟨*proof*⟩

lemma *setsum-ereal*[*simp*]: $(\sum x{\in}A.\ ereal\ (f\ x)) = ereal\ (\sum x{\in}A.\ f\ x)$
⟨*proof*⟩

lemma *setsum-Pinfty*:
fixes $f :: 'a \Rightarrow ereal$
shows $(\sum x{\in}P.\ f\ x) = \infty \longleftrightarrow finite\ P \wedge (\exists\, i{\in}P.\ f\ i = \infty)$
⟨*proof*⟩

lemma *setsum-Inf*:
fixes $f :: 'a \Rightarrow ereal$
shows $|setsum\ f\ A| = \infty \longleftrightarrow finite\ A \wedge (\exists\, i{\in}A.\ |f\ i| = \infty)$
⟨*proof*⟩

lemma *setsum-real-of-ereal*:
fixes $f :: 'i \Rightarrow ereal$
assumes $\bigwedge x.\ x \in S \Longrightarrow |f\ x| \neq \infty$
shows $(\sum x{\in}S.\ real\text{-}of\text{-}ereal\ (f\ x)) = real\text{-}of\text{-}ereal\ (setsum\ f\ S)$
⟨*proof*⟩

lemma *setsum-ereal-0*:
fixes $f :: 'a \Rightarrow ereal$
assumes *finite A*
and $\bigwedge i.\ i \in A \Longrightarrow 0 \leq f\ i$
shows $(\sum x{\in}A.\ f\ x) = 0 \longleftrightarrow (\forall\, i{\in}A.\ f\ i = 0)$
⟨*proof*⟩

36.1.3 Multiplication

instantiation *ereal* :: {*comm-monoid-mult*,*sgn*}
begin

function *sgn-ereal* :: *ereal* ⇒ *ereal* **where**
$sgn\ (ereal\ r) = ereal\ (sgn\ r)$
| $sgn\ (\infty{::}ereal) = 1$
| $sgn\ (-\infty{::}ereal) = -1$
⟨*proof*⟩
termination ⟨*proof*⟩

function *times-ereal* **where**
$ereal\ r * ereal\ p = ereal\ (r * p)$
| $ereal\ r * \infty = (if\ r = 0\ then\ 0\ else\ if\ r > 0\ then\ \infty\ else\ -\infty)$
| $\infty * ereal\ r = (if\ r = 0\ then\ 0\ else\ if\ r > 0\ then\ \infty\ else\ -\infty)$
| $ereal\ r * -\infty = (if\ r = 0\ then\ 0\ else\ if\ r > 0\ then\ -\infty\ else\ \infty)$
| $-\infty * ereal\ r = (if\ r = 0\ then\ 0\ else\ if\ r > 0\ then\ -\infty\ else\ \infty)$
| $(\infty{::}ereal) * \infty = \infty$
| $-(\infty{::}ereal) * \infty = -\infty$
| $(\infty{::}ereal) * -\infty = -\infty$
| $-(\infty{::}ereal) * -\infty = \infty$
⟨*proof*⟩

termination ⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma [*simp*]:
shows *ereal-1-times*: *ereal 1 * x = x*
and *times-ereal-1*: *x * ereal 1 = x*
⟨*proof*⟩

lemma *one-not-le-zero-ereal*[*simp*]: ¬ (*1 ≤ (0*::*ereal*))
⟨*proof*⟩

lemma *real-ereal-1*[*simp*]: *real-of-ereal (1*::*ereal*) = *1*
⟨*proof*⟩

lemma *real-of-ereal-le-1*:
fixes *a* :: *ereal*
shows *a ≤ 1 ⟹ real-of-ereal a ≤ 1*
⟨*proof*⟩

lemma *abs-ereal-one*[*simp*]: *|1| = (1*::*ereal*)
⟨*proof*⟩

lemma *ereal-mult-zero*[*simp*]:
fixes *a* :: *ereal*
shows *a * 0 = 0*
⟨*proof*⟩

lemma *ereal-zero-mult*[*simp*]:
fixes *a* :: *ereal*
shows *0 * a = 0*
⟨*proof*⟩

lemma *ereal-m1-less-0*[*simp*]: *−(1*::*ereal*) < *0*
⟨*proof*⟩

lemma *ereal-times*[*simp*]:
1 ≠ (∞::*ereal*) *(∞*::*ereal*) ≠ *1*
1 ≠ −(∞::*ereal*) *−(∞*::*ereal*) ≠ *1*
⟨*proof*⟩

lemma *ereal-plus-1*[*simp*]:
1 + ereal r = ereal (r + 1)
ereal r + 1 = ereal (r + 1)
1 + −(∞::*ereal*) = *−∞*
−(∞::*ereal*) + *1* = *−∞*

⟨*proof*⟩

lemma *ereal-zero-times*[*simp*]:
fixes *a b* :: *ereal*
shows $a * b = 0 \longleftrightarrow a = 0 \vee b = 0$
⟨*proof*⟩

lemma *ereal-mult-eq-PInfty*[*simp*]:
$a * b = (\infty::ereal) \longleftrightarrow$
$(a = \infty \wedge b > 0) \vee (a > 0 \wedge b = \infty) \vee (a = -\infty \wedge b < 0) \vee (a < 0 \wedge b = -\infty)$
⟨*proof*⟩

lemma *ereal-mult-eq-MInfty*[*simp*]:
$a * b = -(\infty::ereal) \longleftrightarrow$
$(a = \infty \wedge b < 0) \vee (a < 0 \wedge b = \infty) \vee (a = -\infty \wedge b > 0) \vee (a > 0 \wedge b = -\infty)$
⟨*proof*⟩

lemma *ereal-abs-mult*: $|x * y :: ereal| = |x| * |y|$
⟨*proof*⟩

lemma *ereal-0-less-1*[*simp*]: $0 < (1::ereal)$
⟨*proof*⟩

lemma *ereal-mult-minus-left*[*simp*]:
fixes *a b* :: *ereal*
shows $-a * b = -(a * b)$
⟨*proof*⟩

lemma *ereal-mult-minus-right*[*simp*]:
fixes *a b* :: *ereal*
shows $a * -b = -(a * b)$
⟨*proof*⟩

lemma *ereal-mult-infty*[*simp*]:
$a * (\infty::ereal) = (if\ a = 0\ then\ 0\ else\ if\ 0 < a\ then\ \infty\ else\ -\infty)$
⟨*proof*⟩

lemma *ereal-infty-mult*[*simp*]:
$(\infty::ereal) * a = (if\ a = 0\ then\ 0\ else\ if\ 0 < a\ then\ \infty\ else\ -\infty)$
⟨*proof*⟩

lemma *ereal-mult-strict-right-mono*:
assumes $a < b$
and $0 < c$
and $c < (\infty::ereal)$
shows $a * c < b * c$
⟨*proof*⟩

lemma *ereal-mult-strict-left-mono*:
$a < b \Longrightarrow 0 < c \Longrightarrow c < (\infty::ereal) \Longrightarrow c * a < c * b$
⟨*proof*⟩

lemma *ereal-mult-right-mono*:
fixes $a\ b\ c :: ereal$
shows $a \leq b \Longrightarrow 0 \leq c \Longrightarrow a * c \leq b * c$
⟨*proof*⟩

lemma *ereal-mult-left-mono*:
fixes $a\ b\ c :: ereal$
shows $a \leq b \Longrightarrow 0 \leq c \Longrightarrow c * a \leq c * b$
⟨*proof*⟩

lemma *zero-less-one-ereal*[*simp*]: $0 \leq (1::ereal)$
⟨*proof*⟩

lemma *ereal-0-le-mult*[*simp*]: $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow 0 \leq a * (b :: ereal)$
⟨*proof*⟩

lemma *ereal-right-distrib*:
fixes $r\ a\ b :: ereal$
shows $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow r * (a + b) = r * a + r * b$
⟨*proof*⟩

lemma *ereal-left-distrib*:
fixes $r\ a\ b :: ereal$
shows $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow (a + b) * r = a * r + b * r$
⟨*proof*⟩

lemma *ereal-mult-le-0-iff*:
fixes $a\ b :: ereal$
shows $a * b \leq 0 \longleftrightarrow (0 \leq a \wedge b \leq 0) \vee (a \leq 0 \wedge 0 \leq b)$
⟨*proof*⟩

lemma *ereal-zero-le-0-iff*:
fixes $a\ b :: ereal$
shows $0 \leq a * b \longleftrightarrow (0 \leq a \wedge 0 \leq b) \vee (a \leq 0 \wedge b \leq 0)$
⟨*proof*⟩

lemma *ereal-mult-less-0-iff*:
fixes $a\ b :: ereal$
shows $a * b < 0 \longleftrightarrow (0 < a \wedge b < 0) \vee (a < 0 \wedge 0 < b)$
⟨*proof*⟩

lemma *ereal-zero-less-0-iff*:
fixes $a\ b :: ereal$
shows $0 < a * b \longleftrightarrow (0 < a \wedge 0 < b) \vee (a < 0 \wedge b < 0)$

⟨*proof*⟩

lemma *ereal-left-mult-cong*:
fixes *a b c* :: *ereal*
shows $c = d \Longrightarrow (d \neq 0 \Longrightarrow a = b) \Longrightarrow a * c = b * d$
⟨*proof*⟩

lemma *ereal-right-mult-cong*:
fixes *a b c* :: *ereal*
shows $c = d \Longrightarrow (d \neq 0 \Longrightarrow a = b) \Longrightarrow c * a = d * b$
⟨*proof*⟩

lemma *ereal-distrib*:
fixes *a b c* :: *ereal*
assumes $a \neq \infty \vee b \neq -\infty$
and $a \neq -\infty \vee b \neq \infty$
and $|c| \neq \infty$
shows $(a + b) * c = a * c + b * c$
⟨*proof*⟩

lemma *numeral-eq-ereal* [*simp*]: *numeral w* = *ereal* (*numeral w*)
⟨*proof*⟩

lemma *distrib-left-ereal-nn*:
$c \geq 0 \Longrightarrow (x + y) * ereal\ c = x * ereal\ c + y * ereal\ c$
⟨*proof*⟩

lemma *setsum-ereal-right-distrib*:
fixes *f* :: *'a* ⇒ *ereal*
shows $(\bigwedge i.\ i \in A \Longrightarrow 0 \leq f\ i) \Longrightarrow r * setsum\ f\ A = (\sum n{\in}A.\ r * f\ n)$
⟨*proof*⟩

lemma *setsum-ereal-left-distrib*:
$(\bigwedge i.\ i \in A \Longrightarrow 0 \leq f\ i) \Longrightarrow setsum\ f\ A * r = (\sum n{\in}A.\ f\ n * r :: ereal)$
⟨*proof*⟩

lemma *setsum-left-distrib-ereal*:
$c \geq 0 \Longrightarrow setsum\ f\ A * ereal\ c = (\sum x{\in}A.\ f\ x * c :: ereal)$
⟨*proof*⟩

lemma *ereal-le-epsilon*:
fixes *x y* :: *ereal*
assumes $\forall e.\ 0 < e \longrightarrow x \leq y + e$
shows $x \leq y$
⟨*proof*⟩

lemma *ereal-le-epsilon2*:
fixes *x y* :: *ereal*
assumes $\forall e.\ 0 < e \longrightarrow x \leq y + ereal\ e$

shows $x \leq y$
⟨*proof*⟩

lemma *ereal-le-real*:
fixes $x\ y$ :: *ereal*
assumes $\forall z.\ x \leq ereal\ z \longrightarrow y \leq ereal\ z$
shows $y \leq x$
⟨*proof*⟩

lemma *setprod-ereal-0*:
fixes f :: $'a \Rightarrow ereal$
shows $(\prod i{\in}A.\ f\ i) = 0 \longleftrightarrow finite\ A \wedge (\exists i{\in}A.\ f\ i = 0)$
⟨*proof*⟩

lemma *setprod-ereal-pos*:
fixes f :: $'a \Rightarrow ereal$
assumes *pos*: $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$
shows $0 \leq (\prod i{\in}I.\ f\ i)$
⟨*proof*⟩

lemma *setprod-PInf*:
fixes f :: $'a \Rightarrow ereal$
assumes $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$
shows $(\prod i{\in}I.\ f\ i) = \infty \longleftrightarrow finite\ I \wedge (\exists i{\in}I.\ f\ i = \infty) \wedge (\forall i{\in}I.\ f\ i \neq 0)$
⟨*proof*⟩

lemma *setprod-ereal*: $(\prod i{\in}A.\ ereal\ (f\ i)) = ereal\ (setprod\ f\ A)$
⟨*proof*⟩

36.1.4 Power

lemma *ereal-power*[*simp*]: $(ereal\ x)\ \hat{}\ n = ereal\ (x\hat{}n)$
⟨*proof*⟩

lemma *ereal-power-PInf*[*simp*]: $(\infty{::}ereal)\ \hat{}\ n = (if\ n = 0\ then\ 1\ else\ \infty)$
⟨*proof*⟩

lemma *ereal-power-uminus*[*simp*]:
fixes x :: *ereal*
shows $(-x)\ \hat{}\ n = (if\ even\ n\ then\ x\ \hat{}\ n\ else\ -(x\hat{}n))$
⟨*proof*⟩

lemma *ereal-power-numeral*[*simp*]:
$(numeral\ num :: ereal)\ \hat{}\ n = ereal\ (numeral\ num\ \hat{}\ n)$
⟨*proof*⟩

lemma *zero-le-power-ereal*[*simp*]:
fixes a :: *ereal*
assumes $0 \leq a$

shows $0 \leq a \hat{} n$
⟨*proof*⟩

36.1.5 Subtraction

lemma *ereal-minus-minus-image*[*simp*]:
fixes S :: *ereal set*
shows *uminus* ‘ *uminus* ‘ $S = S$
⟨*proof*⟩

lemma *ereal-uminus-lessThan*[*simp*]:
fixes a :: *ereal*
shows *uminus* ‘ $\{..<a\} = \{-a<..\}$
⟨*proof*⟩

lemma *ereal-uminus-greaterThan*[*simp*]: *uminus* ‘ $\{(a::ereal)<..\} = \{..<-a\}$
⟨*proof*⟩

instantiation *ereal* :: *minus*
begin

definition $x - y = x + -(y::ereal)$
instance ⟨*proof*⟩

end

lemma *ereal-minus*[*simp*]:
ereal r − *ereal* p = *ereal* $(r - p)$
$-\infty$ − *ereal* $r = -\infty$
ereal $r - \infty = -\infty$
$(\infty::ereal) - x = \infty$
$-(\infty::ereal) - \infty = -\infty$
$x - -y = x + y$
$x - 0 = x$
$0 - x = -x$
⟨*proof*⟩

lemma *ereal-x-minus-x*[*simp*]: $x - x = (if\ |x| = \infty\ then\ \infty\ else\ 0::ereal)$
⟨*proof*⟩

lemma *ereal-eq-minus-iff*:
fixes $x\ y\ z$ :: *ereal*
shows $x = z - y \longleftrightarrow$
$(|y| \neq \infty \longrightarrow x + y = z) \wedge$
$(y = -\infty \longrightarrow x = \infty) \wedge$
$(y = \infty \longrightarrow z = \infty \longrightarrow x = \infty) \wedge$
$(y = \infty \longrightarrow z \neq \infty \longrightarrow x = -\infty)$
⟨*proof*⟩

lemma *ereal-eq-minus*:
fixes x y z :: *ereal*
shows $|y| \neq \infty \Longrightarrow x = z - y \longleftrightarrow x + y = z$
⟨*proof*⟩

lemma *ereal-less-minus-iff*:
fixes x y z :: *ereal*
shows $x < z - y \longleftrightarrow$
$(y = \infty \longrightarrow z = \infty \wedge x \neq \infty) \wedge$
$(y = -\infty \longrightarrow x \neq \infty) \wedge$
$(|y| \neq \infty \longrightarrow x + y < z)$
⟨*proof*⟩

lemma *ereal-less-minus*:
fixes x y z :: *ereal*
shows $|y| \neq \infty \Longrightarrow x < z - y \longleftrightarrow x + y < z$
⟨*proof*⟩

lemma *ereal-le-minus-iff*:
fixes x y z :: *ereal*
shows $x \leq z - y \longleftrightarrow (y = \infty \longrightarrow z \neq \infty \longrightarrow x = -\infty) \wedge (|y| \neq \infty \longrightarrow x + y \leq z)$
⟨*proof*⟩

lemma *ereal-le-minus*:
fixes x y z :: *ereal*
shows $|y| \neq \infty \Longrightarrow x \leq z - y \longleftrightarrow x + y \leq z$
⟨*proof*⟩

lemma *ereal-minus-less-iff*:
fixes x y z :: *ereal*
shows $x - y < z \longleftrightarrow y \neq -\infty \wedge (y = \infty \longrightarrow x \neq \infty \wedge z \neq -\infty) \wedge (y \neq \infty \longrightarrow x < z + y)$
⟨*proof*⟩

lemma *ereal-minus-less*:
fixes x y z :: *ereal*
shows $|y| \neq \infty \Longrightarrow x - y < z \longleftrightarrow x < z + y$
⟨*proof*⟩

lemma *ereal-minus-le-iff*:
fixes x y z :: *ereal*
shows $x - y \leq z \longleftrightarrow$
$(y = -\infty \longrightarrow z = \infty) \wedge$
$(y = \infty \longrightarrow x = \infty \longrightarrow z = \infty) \wedge$
$(|y| \neq \infty \longrightarrow x \leq z + y)$
⟨*proof*⟩

lemma *ereal-minus-le*:

fixes $x\ y\ z :: ereal$
shows $|y| \neq \infty \Longrightarrow x - y \leq z \longleftrightarrow x \leq z + y$
⟨*proof*⟩

lemma *ereal-minus-eq-minus-iff*:
fixes $a\ b\ c :: ereal$
shows $a - b = a - c \longleftrightarrow$
$b = c \vee a = \infty \vee (a = -\infty \wedge b \neq -\infty \wedge c \neq -\infty)$
⟨*proof*⟩

lemma *ereal-add-le-add-iff*:
fixes $a\ b\ c :: ereal$
shows $c + a \leq c + b \longleftrightarrow$
$a \leq b \vee c = \infty \vee (c = -\infty \wedge a \neq \infty \wedge b \neq \infty)$
⟨*proof*⟩

lemma *ereal-add-le-add-iff2*:
fixes $a\ b\ c :: ereal$
shows $a + c \leq b + c \longleftrightarrow a \leq b \vee c = \infty \vee (c = -\infty \wedge a \neq \infty \wedge b \neq \infty)$
⟨*proof*⟩

lemma *ereal-mult-le-mult-iff*:
fixes $a\ b\ c :: ereal$
shows $|c| \neq \infty \Longrightarrow c * a \leq c * b \longleftrightarrow (0 < c \longrightarrow a \leq b) \wedge (c < 0 \longrightarrow b \leq a)$
⟨*proof*⟩

lemma *ereal-minus-mono*:
fixes $A\ B\ C\ D :: ereal$ **assumes** $A \leq B\ D \leq C$
shows $A - C \leq B - D$
⟨*proof*⟩

lemma *ereal-mono-minus-cancel*:
fixes $a\ b\ c :: ereal$
shows $c - a \leq c - b \Longrightarrow 0 \leq c \Longrightarrow c < \infty \Longrightarrow b \leq a$
⟨*proof*⟩

lemma *real-of-ereal-minus*:
fixes $a\ b :: ereal$
shows *real-of-ereal* $(a - b) =$ (*if* $|a| = \infty \vee |b| = \infty$ *then* 0 *else real-of-ereal* a $-$ *real-of-ereal* b)
⟨*proof*⟩

lemma *real-of-ereal-minus'*: $|x| = \infty \longleftrightarrow |y| = \infty \Longrightarrow$ *real-of-ereal* $x -$ *real-of-ereal* $y =$ *real-of-ereal* $(x - y :: ereal)$
⟨*proof*⟩

lemma *ereal-diff-positive*:
fixes $a\ b :: ereal$ **shows** $a \leq b \Longrightarrow 0 \leq b - a$
⟨*proof*⟩

lemma *ereal-between*:
fixes x e :: *ereal*
assumes $|x| \neq \infty$
and $0 < e$
shows $x - e < x$
and $x < x + e$
⟨*proof*⟩

lemma *ereal-minus-eq-PInfty-iff*:
fixes x y :: *ereal*
shows $x - y = \infty \longleftrightarrow y = -\infty \vee x = \infty$
⟨*proof*⟩

lemma *ereal-diff-add-eq-diff-diff-swap*:
fixes x y z :: *ereal*
shows $|y| \neq \infty \Longrightarrow x - (y + z) = x - y - z$
⟨*proof*⟩

lemma *ereal-diff-add-assoc2*:
fixes x y z :: *ereal*
shows $x + y - z = x - z + y$
⟨*proof*⟩

lemma *ereal-add-uminus-conv-diff*: **fixes** x y z :: *ereal* **shows** $- x + y = y - x$
⟨*proof*⟩

lemma *ereal-minus-diff-eq*:
fixes x y :: *ereal*
shows $[\![x = \infty \longrightarrow y \neq \infty; x = -\infty \longrightarrow y \neq -\infty]\!] \Longrightarrow - (x - y) = y - x$
⟨*proof*⟩

lemma *ediff-le-self* [*simp*]: $x - y \leq (x :: enat)$
⟨*proof*⟩

36.1.6 Division

instantiation *ereal* :: *inverse*
begin

function *inverse-ereal* **where**
inverse (*ereal r*) = (*if r = 0 then* ∞ *else ereal* (*inverse r*))
| *inverse* (∞::*ereal*) = *0*
| *inverse* (−∞::*ereal*) = *0*
⟨*proof*⟩
termination ⟨*proof*⟩

definition x *div* $y = x * inverse\ (y :: ereal)$

instance ⟨*proof*⟩

end

lemma *real-of-ereal-inverse*[*simp*]:
fixes *a* :: *ereal*
shows *real-of-ereal* (*inverse a*) = *1* / *real-of-ereal a*
⟨*proof*⟩

lemma *ereal-inverse*[*simp*]:
inverse (*0*::*ereal*) = ∞
inverse (*1*::*ereal*) = *1*
⟨*proof*⟩

lemma *ereal-divide*[*simp*]:
ereal r / *ereal p* = (*if p* = *0 then ereal r* ∗ ∞ *else ereal* (*r* / *p*))
⟨*proof*⟩

lemma *ereal-divide-same*[*simp*]:
fixes *x* :: *ereal*
shows *x* / *x* = (*if* |*x*| = ∞ ∨ *x* = *0 then 0 else 1*)
⟨*proof*⟩

lemma *ereal-inv-inv*[*simp*]:
fixes *x* :: *ereal*
shows *inverse* (*inverse x*) = (*if x* ≠ −∞ *then x else* ∞)
⟨*proof*⟩

lemma *ereal-inverse-minus*[*simp*]:
fixes *x* :: *ereal*
shows *inverse* (− *x*) = (*if x* = *0 then* ∞ *else* −*inverse x*)
⟨*proof*⟩

lemma *ereal-uminus-divide*[*simp*]:
fixes *x y* :: *ereal*
shows − *x* / *y* = − (*x* / *y*)
⟨*proof*⟩

lemma *ereal-divide-Infty*[*simp*]:
fixes *x* :: *ereal*
shows *x* / ∞ = *0 x* / −∞ = *0*
⟨*proof*⟩

lemma *ereal-divide-one*[*simp*]: *x* / *1* = (*x*::*ereal*)
⟨*proof*⟩

lemma *ereal-divide-ereal*[*simp*]: ∞ / *ereal r* = (*if 0* ≤ *r then* ∞ *else* −∞)
⟨*proof*⟩

lemma *ereal-inverse-nonneg-iff*: $0 \leq inverse\ (x :: ereal) \longleftrightarrow 0 \leq x \vee x = -\infty$
⟨*proof*⟩

lemma *inverse-ereal-ge0I*: $0 \leq (x :: ereal) \Longrightarrow 0 \leq inverse\ x$
⟨*proof*⟩

lemma *zero-le-divide-ereal*[*simp*]:
fixes $a :: ereal$
assumes $0 \leq a$
and $0 \leq b$
shows $0 \leq a\ /\ b$
⟨*proof*⟩

lemma *ereal-le-divide-pos*:
fixes $x\ y\ z :: ereal$
shows $x > 0 \Longrightarrow x \neq \infty \Longrightarrow y \leq z\ /\ x \longleftrightarrow x * y \leq z$
⟨*proof*⟩

lemma *ereal-divide-le-pos*:
fixes $x\ y\ z :: ereal$
shows $x > 0 \Longrightarrow x \neq \infty \Longrightarrow z\ /\ x \leq y \longleftrightarrow z \leq x * y$
⟨*proof*⟩

lemma *ereal-le-divide-neg*:
fixes $x\ y\ z :: ereal$
shows $x < 0 \Longrightarrow x \neq -\infty \Longrightarrow y \leq z\ /\ x \longleftrightarrow z \leq x * y$
⟨*proof*⟩

lemma *ereal-divide-le-neg*:
fixes $x\ y\ z :: ereal$
shows $x < 0 \Longrightarrow x \neq -\infty \Longrightarrow z\ /\ x \leq y \longleftrightarrow x * y \leq z$
⟨*proof*⟩

lemma *ereal-inverse-antimono-strict*:
fixes $x\ y :: ereal$
shows $0 \leq x \Longrightarrow x < y \Longrightarrow inverse\ y < inverse\ x$
⟨*proof*⟩

lemma *ereal-inverse-antimono*:
fixes $x\ y :: ereal$
shows $0 \leq x \Longrightarrow x \leq y \Longrightarrow inverse\ y \leq inverse\ x$
⟨*proof*⟩

lemma *inverse-inverse-Pinfty-iff*[*simp*]:
fixes $x :: ereal$
shows $inverse\ x = \infty \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *ereal-inverse-eq-0*:

fixes x :: *ereal*
shows *inverse* $x = 0 \longleftrightarrow x = \infty \vee x = -\infty$
⟨*proof*⟩

lemma *ereal-0-gt-inverse*:
fixes x :: *ereal*
shows $0 <$ *inverse* $x \longleftrightarrow x \neq \infty \wedge 0 \leq x$
⟨*proof*⟩

lemma *ereal-inverse-le-0-iff*:
fixes x :: *ereal*
shows *inverse* $x \leq 0 \longleftrightarrow x < 0 \vee x = \infty$
⟨*proof*⟩

lemma *ereal-divide-eq-0-iff*: x / $y = 0 \longleftrightarrow x = 0 \vee |y :: ereal| = \infty$
⟨*proof*⟩

lemma *ereal-mult-less-right*:
fixes a b c :: *ereal*
assumes $b * a < c * a$
and $0 < a$
and $a < \infty$
shows $b < c$
⟨*proof*⟩

lemma *ereal-mult-divide*: **fixes** a b :: *ereal* **shows** $0 < b \Longrightarrow b < \infty \Longrightarrow b * (a$ / $b) = a$
⟨*proof*⟩

lemma *ereal-power-divide*:
fixes x y :: *ereal*
shows $y \neq 0 \Longrightarrow (x$ / $y)$ ^ $n = x$^n / y^n
⟨*proof*⟩

lemma *ereal-le-mult-one-interval*:
fixes x y :: *ereal*
assumes y: $y \neq -\infty$
assumes z: $\bigwedge z.\ 0 < z \Longrightarrow z < 1 \Longrightarrow z * x \leq y$
shows $x \leq y$
⟨*proof*⟩

lemma *ereal-divide-right-mono*[*simp*]:
fixes x y z :: *ereal*
assumes $x \leq y$
and $0 < z$
shows x / $z \leq y$ / z
⟨*proof*⟩

lemma *ereal-divide-left-mono*[*simp*]:

fixes $x\ y\ z :: ereal$
assumes $y \leq x$
and $0 < z$
and $0 < x * y$
shows $z\ /\ x \leq z\ /\ y$
⟨*proof*⟩

lemma *ereal-divide-zero-left*[*simp*]:
fixes $a :: ereal$
shows $0\ /\ a = 0$
⟨*proof*⟩

lemma *ereal-times-divide-eq-left*[*simp*]:
fixes $a\ b\ c :: ereal$
shows $b\ /\ c * a = b * a\ /\ c$
⟨*proof*⟩

lemma *ereal-times-divide-eq*: $a * (b\ /\ c :: ereal) = a * b\ /\ c$
⟨*proof*⟩

lemma *ereal-inverse-real*: $|z| \neq \infty \Longrightarrow z \neq 0 \Longrightarrow ereal\ (inverse\ (real\text{-}of\text{-}ereal\ z)) = inverse\ z$
⟨*proof*⟩

lemma *ereal-inverse-mult*:
$a \neq 0 \Longrightarrow b \neq 0 \Longrightarrow inverse\ (a * (b{::}ereal)) = inverse\ a * inverse\ b$
⟨*proof*⟩

36.2 Complete lattice

instantiation *ereal* :: *lattice*
begin

definition [*simp*]: $sup\ x\ y = (max\ x\ y :: ereal)$
definition [*simp*]: $inf\ x\ y = (min\ x\ y :: ereal)$
instance ⟨*proof*⟩

end

instantiation *ereal* :: *complete-lattice*
begin

definition $bot = (-\infty{::}ereal)$
definition $top = (\infty{::}ereal)$

definition $Sup\ S = (SOME\ x :: ereal.\ (\forall\, y{\in}S.\ y \leq x) \wedge (\forall\, z.\ (\forall\, y{\in}S.\ y \leq z) \longrightarrow x \leq z))$
definition $Inf\ S = (SOME\ x :: ereal.\ (\forall\, y{\in}S.\ x \leq y) \wedge (\forall\, z.\ (\forall\, y{\in}S.\ z \leq y) \longrightarrow z \leq x))$

lemma *ereal-complete-Sup*:
fixes S :: *ereal set*
shows $\exists x.\ (\forall y{\in}S.\ y \leq x) \wedge (\forall z.\ (\forall y{\in}S.\ y \leq z) \longrightarrow x \leq z)$
⟨*proof*⟩

lemma *ereal-complete-uminus-eq*:
fixes S :: *ereal set*
shows $(\forall y{\in}uminus`S.\ y \leq x) \wedge (\forall z.\ (\forall y{\in}uminus`S.\ y \leq z) \longrightarrow x \leq z)$
$\longleftrightarrow (\forall y{\in}S.\ -x \leq y) \wedge (\forall z.\ (\forall y{\in}S.\ z \leq y) \longrightarrow z \leq -x)$
⟨*proof*⟩

lemma *ereal-complete-Inf*:
$\exists x.\ (\forall y{\in}S{::}ereal\ set.\ x \leq y) \wedge (\forall z.\ (\forall y{\in}S.\ z \leq y) \longrightarrow z \leq x)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

instance *ereal* :: *complete-linorder* ⟨*proof*⟩

instance *ereal* :: *linear-continuum*
⟨*proof*⟩

36.2.1 Topological space

instantiation *ereal* :: *linear-continuum-topology*
begin

definition *open-ereal* :: *ereal set* ⇒ *bool* **where**
open-ereal-generated: *open-ereal* = *generate-topology* (*range lessThan* ∪ *range greaterThan*)

instance
⟨*proof*⟩

end

lemma *continuous-on-ereal*[*continuous-intros*]:
assumes f: *continuous-on s f* **shows** *continuous-on s* (λ*x*. *ereal* (*f x*))
⟨*proof*⟩

lemma *tendsto-ereal*[*tendsto-intros*, *simp*, *intro*]: $(f \longrightarrow x)\ F \Longrightarrow ((\lambda x.\ ereal\ (f\ x)) \longrightarrow ereal\ x)\ F$
⟨*proof*⟩

lemma *tendsto-uminus-ereal*[*tendsto-intros*, *simp*, *intro*]: $(f \longrightarrow x)\ F \Longrightarrow ((\lambda x.$

$- f\ x::ereal) \longrightarrow - x)\ F$
⟨*proof*⟩

lemma *at-infty-ereal-eq-at-top*: $at\ \infty = filtermap\ ereal\ at\text{-}top$
⟨*proof*⟩

lemma *ereal-Lim-uminus*: $(f \longrightarrow f0)\ net \longleftrightarrow ((\lambda x.\ - f\ x::ereal) \longrightarrow - f0)\ net$
⟨*proof*⟩

lemma *ereal-divide-less-iff*: $0 < (c::ereal) \Longrightarrow c < \infty \Longrightarrow a\ /\ c < b \longleftrightarrow a < b * c$
⟨*proof*⟩

lemma *ereal-less-divide-iff*: $0 < (c::ereal) \Longrightarrow c < \infty \Longrightarrow a < b\ /\ c \longleftrightarrow a * c < b$
⟨*proof*⟩

lemma *tendsto-cmult-ereal*[*tendsto-intros*, *simp*, *intro*]:
assumes c: $|c| \neq \infty$ **and** f: $(f \longrightarrow x)\ F$ **shows** $((\lambda x.\ c * f\ x::ereal) \longrightarrow c * x)\ F$
⟨*proof*⟩

lemma *tendsto-cmult-ereal-not-0*[*tendsto-intros*, *simp*, *intro*]:
assumes $x \neq 0$ **and** f: $(f \longrightarrow x)\ F$ **shows** $((\lambda x.\ c * f\ x::ereal) \longrightarrow c * x)\ F$
⟨*proof*⟩

lemma *tendsto-cadd-ereal*[*tendsto-intros*, *simp*, *intro*]:
assumes c: $y \neq - \infty\ x \neq - \infty$ **and** f: $(f \longrightarrow x)\ F$ **shows** $((\lambda x.\ f\ x + y::ereal) \longrightarrow x + y)\ F$
⟨*proof*⟩

lemma *tendsto-add-left-ereal*[*tendsto-intros*, *simp*, *intro*]:
assumes c: $|y| \neq \infty$ **and** f: $(f \longrightarrow x)\ F$ **shows** $((\lambda x.\ f\ x + y::ereal) \longrightarrow x + y)\ F$
⟨*proof*⟩

lemma *continuous-at-ereal*[*continuous-intros*]: $continuous\ F\ f \Longrightarrow continuous\ F\ (\lambda x.\ ereal\ (f\ x))$
⟨*proof*⟩

lemma *ereal-Sup*:
assumes $*$: $|SUP\ a{:}A.\ ereal\ a| \neq \infty$
shows $ereal\ (Sup\ A) = (SUP\ a{:}A.\ ereal\ a)$
⟨*proof*⟩

lemma *ereal-SUP*: $|SUP\ a{:}A.\ ereal\ (f\ a)| \neq \infty \Longrightarrow ereal\ (SUP\ a{:}A.\ f\ a) = (SUP\ a{:}A.\ ereal\ (f\ a))$

⟨*proof*⟩

lemma *ereal-Inf*:
assumes ∗: $|INF\ a{:}A.\ ereal\ a| \neq \infty$
shows $ereal\ (Inf\ A) = (INF\ a{:}A.\ ereal\ a)$
⟨*proof*⟩

lemma *ereal-Inf'*:
assumes ∗: *bdd-below A* $A \neq \{\}$
shows $ereal\ (Inf\ A) = (INF\ a{:}A.\ ereal\ a)$
⟨*proof*⟩

lemma *ereal-INF*: $|INF\ a{:}A.\ ereal\ (f\ a)| \neq \infty \Longrightarrow ereal\ (INF\ a{:}A.\ f\ a) = (INF\ a{:}A.\ ereal\ (f\ a))$
⟨*proof*⟩

lemma *ereal-Sup-uminus-image-eq*: $Sup\ (uminus\ `\ S{::}ereal\ set) = -\ Inf\ S$
⟨*proof*⟩

lemma *ereal-SUP-uminus-eq*:
fixes $f :: 'a \Rightarrow ereal$
shows $(SUP\ x{:}S.\ uminus\ (f\ x)) = -\ (INF\ x{:}S.\ f\ x)$
⟨*proof*⟩

lemma *ereal-inj-on-uminus*[*intro*, *simp*]: *inj-on uminus* $(A :: ereal\ set)$
⟨*proof*⟩

lemma *ereal-Inf-uminus-image-eq*: $Inf\ (uminus\ `\ S{::}ereal\ set) = -\ Sup\ S$
⟨*proof*⟩

lemma *ereal-INF-uminus-eq*:
fixes $f :: 'a \Rightarrow ereal$
shows $(INF\ x{:}S.\ -\ f\ x) = -\ (SUP\ x{:}S.\ f\ x)$
⟨*proof*⟩

lemma *ereal-SUP-uminus*:
fixes $f :: 'a \Rightarrow ereal$
shows $(SUP\ i : R.\ -\ f\ i) = -\ (INF\ i : R.\ f\ i)$
⟨*proof*⟩

lemma *ereal-SUP-not-infty*:
fixes $f :: - \Rightarrow ereal$
shows $A \neq \{\} \Longrightarrow l \neq -\infty \Longrightarrow u \neq \infty \Longrightarrow \forall a{\in}A.\ l \leq f\ a \wedge f\ a \leq u \Longrightarrow |SUPREMUM\ A\ f| \neq \infty$
⟨*proof*⟩

lemma *ereal-INF-not-infty*:
fixes $f :: - \Rightarrow ereal$
shows $A \neq \{\} \Longrightarrow l \neq -\infty \Longrightarrow u \neq \infty \Longrightarrow \forall a{\in}A.\ l \leq f\ a \wedge f\ a \leq u \Longrightarrow$

$|INFIMUM\ A\ f| \neq \infty$
⟨*proof*⟩

lemma *ereal-image-uminus-shift*:
fixes $X\ Y$ :: *ereal set*
shows $uminus\ `\ X = Y \longleftrightarrow X = uminus\ `\ Y$
⟨*proof*⟩

lemma *Sup-eq-MInfty*:
fixes S :: *ereal set*
shows $Sup\ S = -\infty \longleftrightarrow S = \{\} \vee S = \{-\infty\}$
⟨*proof*⟩

lemma *Inf-eq-PInfty*:
fixes S :: *ereal set*
shows $Inf\ S = \infty \longleftrightarrow S = \{\} \vee S = \{\infty\}$
⟨*proof*⟩

lemma *Inf-eq-MInfty*:
fixes S :: *ereal set*
shows $-\infty \in S \Longrightarrow Inf\ S = -\infty$
⟨*proof*⟩

lemma *Sup-eq-PInfty*:
fixes S :: *ereal set*
shows $\infty \in S \Longrightarrow Sup\ S = \infty$
⟨*proof*⟩

lemma *not-MInfty-nonneg*[*simp*]: $0 \leq (x::ereal) \Longrightarrow x \neq -\infty$
⟨*proof*⟩

lemma *Sup-ereal-close*:
fixes e :: *ereal*
assumes $0 < e$
and S: $|Sup\ S| \neq \infty\ S \neq \{\}$
shows $\exists\, x{\in}S.\ Sup\ S - e < x$
⟨*proof*⟩

lemma *Inf-ereal-close*:
fixes e :: *ereal*
assumes $|Inf\ X| \neq \infty$
and $0 < e$
shows $\exists\, x{\in}X.\ x < Inf\ X + e$
⟨*proof*⟩

lemma *SUP-PInfty*:
$(\bigwedge n::nat.\ \exists\, i{\in}A.\ ereal\ (real\ n) \leq f\ i) \Longrightarrow (SUP\ i{:}A.\ f\ i :: ereal) = \infty$
⟨*proof*⟩

lemma *SUP-nat-Infty*: $(SUP\ i{::}nat.\ ereal\ (real\ i)) = \infty$
⟨*proof*⟩

lemma *SUP-ereal-add-left*:
assumes $I \neq \{\}$ $c \neq -\infty$
shows $(SUP\ i{:}I.\ f\ i + c :: ereal) = (SUP\ i{:}I.\ f\ i) + c$
⟨*proof*⟩

lemma *SUP-ereal-add-right*:
fixes $c :: ereal$
shows $I \neq \{\} \Longrightarrow c \neq -\infty \Longrightarrow (SUP\ i{:}I.\ c + f\ i) = c + (SUP\ i{:}I.\ f\ i)$
⟨*proof*⟩

lemma *SUP-ereal-minus-right*:
assumes $I \neq \{\}$ $c \neq -\infty$
shows $(SUP\ i{:}I.\ c - f\ i :: ereal) = c - (INF\ i{:}I.\ f\ i)$
⟨*proof*⟩

lemma *SUP-ereal-minus-left*:
assumes $I \neq \{\}$ $c \neq \infty$
shows $(SUP\ i{:}I.\ f\ i - c{::}\ ereal) = (SUP\ i{:}I.\ f\ i) - c$
⟨*proof*⟩

lemma *INF-ereal-minus-right*:
assumes $I \neq \{\}$ **and** $|c| \neq \infty$
shows $(INF\ i{:}I.\ c - f\ i) = c - (SUP\ i{:}I.\ f\ i{::}ereal)$
⟨*proof*⟩

lemma *SUP-ereal-le-addI*:
fixes $f :: 'i \Rightarrow ereal$
assumes $\bigwedge i.\ f\ i + y \leq z$ **and** $y \neq -\infty$
shows $SUPREMUM\ UNIV\ f + y \leq z$
⟨*proof*⟩

lemma *SUP-combine*:
fixes $f :: 'a{::}semilattice\text{-}sup \Rightarrow 'a{::}semilattice\text{-}sup \Rightarrow 'b{::}complete\text{-}lattice$
assumes *mono*: $\bigwedge a\ b\ c\ d.\ a \leq b \Longrightarrow c \leq d \Longrightarrow f\ a\ c \leq f\ b\ d$
shows $(SUP\ i{:}UNIV.\ SUP\ j{:}UNIV.\ f\ i\ j) = (SUP\ i.\ f\ i\ i)$
⟨*proof*⟩

lemma *SUP-ereal-add*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes *inc*: *incseq f incseq g*
and *pos*: $\bigwedge i.\ f\ i \neq -\infty\ \bigwedge i.\ g\ i \neq -\infty$
shows $(SUP\ i.\ f\ i + g\ i) = SUPREMUM\ UNIV\ f + SUPREMUM\ UNIV\ g$
⟨*proof*⟩

lemma *INF-ereal-add*:
fixes $f :: nat \Rightarrow ereal$

assumes *decseq f decseq g*
and *fin*: $\bigwedge i.\ f\ i \neq \infty\ \bigwedge i.\ g\ i \neq \infty$
shows $(INF\ i.\ f\ i + g\ i) = INFIMUM\ UNIV\ f + INFIMUM\ UNIV\ g$
⟨*proof*⟩

lemma *SUP-ereal-add-pos*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes *inc*: *incseq f incseq g*
and *pos*: $\bigwedge i.\ 0 \leq f\ i\ \bigwedge i.\ 0 \leq g\ i$
shows $(SUP\ i.\ f\ i + g\ i) = SUPREMUM\ UNIV\ f + SUPREMUM\ UNIV\ g$
⟨*proof*⟩

lemma *SUP-ereal-setsum*:
fixes $f\ g :: 'a \Rightarrow nat \Rightarrow ereal$
assumes $\bigwedge n.\ n \in A \Longrightarrow incseq\ (f\ n)$
and *pos*: $\bigwedge n\ i.\ n \in A \Longrightarrow 0 \leq f\ n\ i$
shows $(SUP\ i.\ \sum n{\in}A.\ f\ n\ i) = (\sum n{\in}A.\ SUPREMUM\ UNIV\ (f\ n))$
⟨*proof*⟩

lemma *SUP-ereal-mult-left*:
fixes $f :: 'a \Rightarrow ereal$
assumes $I \neq \{\}$
assumes *f*: $\bigwedge i.\ i \in I \Longrightarrow 0 \leq f\ i$ **and** *c*: $0 \leq c$
shows $(SUP\ i{:}I.\ c * f\ i) = c * (SUP\ i{:}I.\ f\ i)$
⟨*proof*⟩

lemma *countable-approach*:
fixes $x :: ereal$
assumes $x \neq -\infty$
shows $\exists f.\ incseq\ f \wedge (\forall i{::}nat.\ f\ i < x) \wedge (f \longrightarrow x)$
⟨*proof*⟩

lemma *Sup-countable-SUP*:
assumes $A \neq \{\}$
shows $\exists f{::}nat \Rightarrow ereal.\ incseq\ f \wedge range\ f \subseteq A \wedge Sup\ A = (SUP\ i.\ f\ i)$
⟨*proof*⟩

lemma *SUP-countable-SUP*:
$A \neq \{\} \Longrightarrow \exists f{::}nat \Rightarrow ereal.\ range\ f \subseteq g`A \wedge SUPREMUM\ A\ g = SUPREMUM\ UNIV\ f$
⟨*proof*⟩

36.3 Relation to *enat*

definition $ereal\text{-}of\text{-}enat\ n = (case\ n\ of\ enat\ n \Rightarrow ereal\ (real\ n) \mid \infty \Rightarrow \infty)$

declare [[*coercion ereal-of-enat* :: *enat* ⇒ *ereal*]]
declare [[*coercion* (λ*n*. *ereal* (*real n*)) :: *nat* ⇒ *ereal*]]

lemma *ereal-of-enat-simps*[*simp*]:
ereal-of-enat (*enat n*) = *ereal n*
ereal-of-enat ∞ = ∞
⟨*proof*⟩

lemma *ereal-of-enat-le-iff*[*simp*]: *ereal-of-enat m* ≤ *ereal-of-enat n* ⟷ *m* ≤ *n*
⟨*proof*⟩

lemma *ereal-of-enat-less-iff*[*simp*]: *ereal-of-enat m* < *ereal-of-enat n* ⟷ *m* < *n*
⟨*proof*⟩

lemma *numeral-le-ereal-of-enat-iff*[*simp*]: *numeral m* ≤ *ereal-of-enat n* ⟷ *numeral m* ≤ *n*
⟨*proof*⟩

lemma *numeral-less-ereal-of-enat-iff*[*simp*]: *numeral m* < *ereal-of-enat n* ⟷ *numeral m* < *n*
⟨*proof*⟩

lemma *ereal-of-enat-ge-zero-cancel-iff*[*simp*]: *0* ≤ *ereal-of-enat n* ⟷ *0* ≤ *n*
⟨*proof*⟩

lemma *ereal-of-enat-gt-zero-cancel-iff*[*simp*]: *0* < *ereal-of-enat n* ⟷ *0* < *n*
⟨*proof*⟩

lemma *ereal-of-enat-zero*[*simp*]: *ereal-of-enat 0* = *0*
⟨*proof*⟩

lemma *ereal-of-enat-inf*[*simp*]: *ereal-of-enat n* = ∞ ⟷ *n* = ∞
⟨*proof*⟩

lemma *ereal-of-enat-add*: *ereal-of-enat* (*m* + *n*) = *ereal-of-enat m* + *ereal-of-enat n*
⟨*proof*⟩

lemma *ereal-of-enat-sub*:
assumes *n* ≤ *m*
shows *ereal-of-enat* (*m* − *n*) = *ereal-of-enat m* − *ereal-of-enat n*
⟨*proof*⟩

lemma *ereal-of-enat-mult*:
ereal-of-enat (*m* ∗ *n*) = *ereal-of-enat m* ∗ *ereal-of-enat n*
⟨*proof*⟩

lemmas *ereal-of-enat-pushin* = *ereal-of-enat-add ereal-of-enat-sub ereal-of-enat-mult*
lemmas *ereal-of-enat-pushout* = *ereal-of-enat-pushin*[*symmetric*]

lemma *ereal-of-enat-nonneg*: *ereal-of-enat n* ≥ *0*
⟨*proof*⟩

lemma *ereal-of-enat-Sup*:
assumes $A \neq \{\}$ **shows** *ereal-of-enat* (*Sup* A) = (*SUP* $a : A$. *ereal-of-enat* a)
⟨*proof*⟩

lemma *ereal-of-enat-SUP*:
$A \neq \{\} \Longrightarrow$ *ereal-of-enat* (*SUP* a:A. $f\ a$) = (*SUP* $a : A$. *ereal-of-enat* ($f\ a$))
⟨*proof*⟩

36.4 Limits on *ereal*

lemma *open-PInfty*: *open* $A \Longrightarrow \infty \in A \Longrightarrow (\exists x.\ \{ereal\ x<..\} \subseteq A)$
⟨*proof*⟩

lemma *open-MInfty*: *open* $A \Longrightarrow -\infty \in A \Longrightarrow (\exists x.\ \{..<ereal\ x\} \subseteq A)$
⟨*proof*⟩

lemma *open-ereal-vimage*: *open* $S \Longrightarrow$ *open* (*ereal* $-`S$)
⟨*proof*⟩

lemma *open-ereal*: *open* $S \Longrightarrow$ *open* (*ereal* $`S$)
⟨*proof*⟩

lemma *eventually-finite*:
fixes x :: *ereal*
assumes $|x| \neq \infty$ ($f \longrightarrow x$) F
shows *eventually* ($\lambda x.\ |f\ x| \neq \infty$) F
⟨*proof*⟩

lemma *open-ereal-def*:
open $A \longleftrightarrow$ *open* (*ereal* $-`A$) $\wedge$ ($\infty \in A \longrightarrow (\exists x.\ \{ereal\ x <..\} \subseteq A)$) $\wedge$ ($-\infty \in A \longrightarrow (\exists x.\ \{..<ereal\ x\} \subseteq A)$)
(**is** *open* $A \longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *open-PInfty2*:
assumes *open* A
and $\infty \in A$
obtains x **where** $\{ereal\ x<..\} \subseteq A$
⟨*proof*⟩

lemma *open-MInfty2*:
assumes *open* A
and $-\infty \in A$
obtains x **where** $\{..<ereal\ x\} \subseteq A$
⟨*proof*⟩

lemma *ereal-openE*:
assumes *open A*
obtains *x y* **where** *open* (*ereal* −‘ *A*)
and ∞ ∈ *A* ⟹ {*ereal x*<..} ⊆ *A*
and −∞ ∈ *A* ⟹ {..<*ereal y*} ⊆ *A*
⟨*proof*⟩

lemmas *open-ereal-lessThan* = *open-lessThan*[**where** *'a*=*ereal*]
lemmas *open-ereal-greaterThan* = *open-greaterThan*[**where** *'a*=*ereal*]
lemmas *ereal-open-greaterThanLessThan* = *open-greaterThanLessThan*[**where** *'a*=*ereal*]
lemmas *closed-ereal-atLeast* = *closed-atLeast*[**where** *'a*=*ereal*]
lemmas *closed-ereal-atMost* = *closed-atMost*[**where** *'a*=*ereal*]
lemmas *closed-ereal-atLeastAtMost* = *closed-atLeastAtMost*[**where** *'a*=*ereal*]
lemmas *closed-ereal-singleton* = *closed-singleton*[**where** *'a*=*ereal*]

lemma *ereal-open-cont-interval*:
fixes *S* :: *ereal set*
assumes *open S*
and *x* ∈ *S*
and |*x*| ≠ ∞
obtains *e* **where** *e* > *0* **and** {*x*−*e* <..< *x*+*e*} ⊆ *S*
⟨*proof*⟩

lemma *ereal-open-cont-interval2*:
fixes *S* :: *ereal set*
assumes *open S*
and *x* ∈ *S*
and *x*: |*x*| ≠ ∞
obtains *a b* **where** *a* < *x* **and** *x* < *b* **and** {*a* <..< *b*} ⊆ *S*
⟨*proof*⟩

36.4.1 Convergent sequences

lemma *lim-real-of-ereal*[*simp*]:
assumes *lim*: (*f* ⟶ *ereal x*) *net*
shows ((λ*x*. *real-of-ereal* (*f x*)) ⟶ *x*) *net*
⟨*proof*⟩

lemma *lim-ereal*[*simp*]: ((λ*n*. *ereal* (*f n*)) ⟶ *ereal x*) *net* ⟷ (*f* ⟶ *x*) *net*
⟨*proof*⟩

lemma *convergent-real-imp-convergent-ereal*:
assumes *convergent a*
shows *convergent* (λ*n*. *ereal* (*a n*)) **and** *lim* (λ*n*. *ereal* (*a n*)) = *ereal* (*lim a*)
⟨*proof*⟩

lemma *tendsto-PInfty*: (*f* ⟶ ∞) *F* ⟷ (∀ *r*. *eventually* (λ*x*. *ereal r* < *f x*) *F*)
⟨*proof*⟩

lemma *tendsto-PInfty'*: $(f \longrightarrow \infty)\ F = (\forall r{>}c.\ \mathit{eventually}\ (\lambda x.\ \mathit{ereal}\ r < f\ x)\ F)$
⟨*proof*⟩

lemma *tendsto-PInfty-eq-at-top*:
$((\lambda z.\ \mathit{ereal}\ (f\ z)) \longrightarrow \infty)\ F \longleftrightarrow (\mathit{LIM}\ z\ F.\ f\ z :> \mathit{at\text{-}top})$
⟨*proof*⟩

lemma *tendsto-MInfty*: $(f \longrightarrow -\infty)\ F \longleftrightarrow (\forall r.\ \mathit{eventually}\ (\lambda x.\ f\ x < \mathit{ereal}\ r)\ F)$
⟨*proof*⟩

lemma *tendsto-MInfty'*: $(f \longrightarrow -\infty)\ F = (\forall r{<}c.\ \mathit{eventually}\ (\lambda x.\ \mathit{ereal}\ r > f\ x)\ F)$
⟨*proof*⟩

lemma *Lim-PInfty*: $f \longrightarrow \infty \longleftrightarrow (\forall B.\ \exists N.\ \forall n{\geq}N.\ f\ n \geq \mathit{ereal}\ B)$
⟨*proof*⟩

lemma *Lim-MInfty*: $f \longrightarrow -\infty \longleftrightarrow (\forall B.\ \exists N.\ \forall n{\geq}N.\ \mathit{ereal}\ B \geq f\ n)$
⟨*proof*⟩

lemma *Lim-bounded-PInfty*: $f \longrightarrow l \Longrightarrow (\bigwedge n.\ f\ n \leq \mathit{ereal}\ B) \Longrightarrow l \neq \infty$
⟨*proof*⟩

lemma *Lim-bounded-MInfty*: $f \longrightarrow l \Longrightarrow (\bigwedge n.\ \mathit{ereal}\ B \leq f\ n) \Longrightarrow l \neq -\infty$
⟨*proof*⟩

lemma *tendsto-zero-erealI*:
assumes $\bigwedge e.\ e > 0 \Longrightarrow \mathit{eventually}\ (\lambda x.\ |f\ x| < \mathit{ereal}\ e)\ F$
shows $(f \longrightarrow 0)\ F$
⟨*proof*⟩

lemma *tendsto-explicit*:
$f \longrightarrow f0 \longleftrightarrow (\forall S.\ \mathit{open}\ S \longrightarrow f0 \in S \longrightarrow (\exists N.\ \forall n{\geq}N.\ f\ n \in S))$
⟨*proof*⟩

lemma *Lim-bounded-PInfty2*: $f \longrightarrow l \Longrightarrow \forall n{\geq}N.\ f\ n \leq \mathit{ereal}\ B \Longrightarrow l \neq \infty$
⟨*proof*⟩

lemma *Lim-bounded-ereal*: $f \longrightarrow (l :: 'a{::}\mathit{linorder\text{-}topology}) \Longrightarrow \forall n{\geq}M.\ f\ n \leq C \Longrightarrow l \leq C$
⟨*proof*⟩

lemma *Lim-bounded2-ereal*:
assumes *lim*:$f \longrightarrow (l :: 'a{::}\mathit{linorder\text{-}topology})$
and *ge*: $\forall n{\geq}N.\ f\ n \geq C$
shows $l \geq C$
⟨*proof*⟩

lemma *real-of-ereal-mult*[*simp*]:
fixes $a\ b$ :: *ereal*
shows *real-of-ereal* $(a * b) =$ *real-of-ereal* $a *$ *real-of-ereal* b
⟨*proof*⟩

lemma *real-of-ereal-eq-0*:
fixes x :: *ereal*
shows *real-of-ereal* $x = 0 \longleftrightarrow x = \infty \vee x = -\infty \vee x = 0$
⟨*proof*⟩

lemma *tendsto-ereal-realD*:
fixes f :: $'a \Rightarrow$ *ereal*
assumes $x \neq 0$
and *tendsto*: $((\lambda x.$ *ereal* (*real-of-ereal* $(f\ x))) \longrightarrow x)$ *net*
shows $(f \longrightarrow x)$ *net*
⟨*proof*⟩

lemma *tendsto-ereal-realI*:
fixes f :: $'a \Rightarrow$ *ereal*
assumes x: $|x| \neq \infty$ **and** *tendsto*: $(f \longrightarrow x)$ *net*
shows $((\lambda x.$ *ereal* (*real-of-ereal* $(f\ x))) \longrightarrow x)$ *net*
⟨*proof*⟩

lemma *ereal-mult-cancel-left*:
fixes $a\ b\ c$ :: *ereal*
shows $a * b = a * c \longleftrightarrow (|a| = \infty \wedge 0 < b * c) \vee a = 0 \vee b = c$
⟨*proof*⟩

lemma *tendsto-add-ereal*:
fixes $x\ y$ :: *ereal*
assumes x: $|x| \neq \infty$ **and** y: $|y| \neq \infty$
assumes f: $(f \longrightarrow x)\ F$ **and** g: $(g \longrightarrow y)\ F$
shows $((\lambda x.\ f\ x + g\ x) \longrightarrow x + y)\ F$
⟨*proof*⟩

lemma *tendsto-add-ereal-nonneg*:
fixes $x\ y$ :: *ereal*
assumes $x \neq -\infty$ $y \neq -\infty$ $(f \longrightarrow x)\ F$ $(g \longrightarrow y)\ F$
shows $((\lambda x.\ f\ x + g\ x) \longrightarrow x + y)\ F$
⟨*proof*⟩

lemma *ereal-inj-affinity*:
fixes $m\ t$ :: *ereal*
assumes $|m| \neq \infty$
and $m \neq 0$
and $|t| \neq \infty$
shows *inj-on* $(\lambda x.\ m * x + t)\ A$
⟨*proof*⟩

lemma *ereal-PInfty-eq-plus*[*simp*]:
fixes $a\ b$ :: *ereal*
shows $\infty = a + b \longleftrightarrow a = \infty \vee b = \infty$
⟨*proof*⟩

lemma *ereal-MInfty-eq-plus*[*simp*]:
fixes $a\ b$ :: *ereal*
shows $-\infty = a + b \longleftrightarrow (a = -\infty \wedge b \neq \infty) \vee (b = -\infty \wedge a \neq \infty)$
⟨*proof*⟩

lemma *ereal-less-divide-pos*:
fixes $x\ y$ :: *ereal*
shows $x > 0 \Longrightarrow x \neq \infty \Longrightarrow y < z\ /\ x \longleftrightarrow x * y < z$
⟨*proof*⟩

lemma *ereal-divide-less-pos*:
fixes $x\ y\ z$ :: *ereal*
shows $x > 0 \Longrightarrow x \neq \infty \Longrightarrow y\ /\ x < z \longleftrightarrow y < x * z$
⟨*proof*⟩

lemma *ereal-divide-eq*:
fixes $a\ b\ c$ :: *ereal*
shows $b \neq 0 \Longrightarrow |b| \neq \infty \Longrightarrow a\ /\ b = c \longleftrightarrow a = b * c$
⟨*proof*⟩

lemma *ereal-inverse-not-MInfty*[*simp*]: *inverse* (*a*::*ereal*) $\neq -\infty$
⟨*proof*⟩

lemma *ereal-mult-m1*[*simp*]: $x *$ *ereal* $(-1) = -x$
⟨*proof*⟩

lemma *ereal-real'*:
assumes $|x| \neq \infty$
shows *ereal* (*real-of-ereal* x) $= x$
⟨*proof*⟩

lemma *real-ereal-id*: *real-of-ereal* $\circ$ *ereal* $=$ *id*
⟨*proof*⟩

lemma *open-image-ereal*: *open*(*UNIV*−{ ∞ , ($-\infty$ :: *ereal*)})
⟨*proof*⟩

lemma *ereal-le-distrib*:
fixes $a\ b\ c$ :: *ereal*
shows $c * (a + b) \leq c * a + c * b$
⟨*proof*⟩

lemma *ereal-pos-distrib*:

fixes a b c :: *ereal*
assumes $0 \leq c$
and $c \neq \infty$
shows $c * (a + b) = c * a + c * b$
⟨*proof*⟩

lemma *ereal-max-mono*: $(a::ereal) \leq b \Longrightarrow c \leq d \Longrightarrow max\ a\ c \leq max\ b\ d$
⟨*proof*⟩

lemma *ereal-max-least*: $(a::ereal) \leq x \Longrightarrow c \leq x \Longrightarrow max\ a\ c \leq x$
⟨*proof*⟩

lemma *ereal-LimI-finite*:
fixes x :: *ereal*
assumes $|x| \neq \infty$
and $\bigwedge r.\ 0 < r \Longrightarrow \exists N.\ \forall n \geq N.\ u\ n < x + r \wedge x < u\ n + r$
shows $u \longrightarrow x$
⟨*proof*⟩

lemma *tendsto-obtains-N*:
assumes $f \longrightarrow f0$
assumes *open* S
and $f0 \in S$
obtains N **where** $\forall n \geq N.\ f\ n \in S$
⟨*proof*⟩

lemma *ereal-LimI-finite-iff*:
fixes x :: *ereal*
assumes $|x| \neq \infty$
shows $u \longrightarrow x \longleftrightarrow (\forall r.\ 0 < r \longrightarrow (\exists N.\ \forall n \geq N.\ u\ n < x + r \wedge x < u\ n + r))$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *ereal-Limsup-uminus*:
fixes f :: $'a \Rightarrow ereal$
shows *Limsup net* $(\lambda x.\ -\ (f\ x)) = -\ Liminf\ net\ f$
⟨*proof*⟩

lemma *liminf-bounded-iff*:
fixes x :: $nat \Rightarrow ereal$
shows $C \leq liminf\ x \longleftrightarrow (\forall B < C.\ \exists N.\ \forall n \geq N.\ B < x\ n)$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *Liminf-add-le*:
fixes $f\ g$ :: - $\Rightarrow$ *ereal*
assumes F: $F \neq bot$
assumes *ev*: *eventually* $(\lambda x.\ 0 \leq f\ x)\ F$ *eventually* $(\lambda x.\ 0 \leq g\ x)\ F$

shows *Liminf F f + Liminf F g ≤ Liminf F (λx. f x + g x)*
⟨*proof*⟩

lemma *Sup-ereal-mult-right'*:
assumes *nonempty*: *Y ≠ {}*
and *x*: *x ≥ 0*
shows *(SUP i:Y. f i) * ereal x = (SUP i:Y. f i * ereal x)* (**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *Sup-ereal-mult-left'*:
*⟦ Y ≠ {}; x ≥ 0 ⟧ ⟹ ereal x * (SUP i:Y. f i) = (SUP i:Y. ereal x * f i)*
⟨*proof*⟩

lemma *sup-continuous-add*[*order-continuous-intros*]:
fixes *f g :: 'a::complete-lattice ⇒ ereal*
assumes *nn*: *⋀x. 0 ≤ f x ⋀x. 0 ≤ g x* **and** *cont*: *sup-continuous f sup-continuous g*
shows *sup-continuous (λx. f x + g x)*
⟨*proof*⟩

lemma *sup-continuous-mult-right*[*order-continuous-intros*]:
*0 ≤ c ⟹ c < ∞ ⟹ sup-continuous f ⟹ sup-continuous (λx. f x * c :: ereal)*
⟨*proof*⟩

lemma *sup-continuous-mult-left*[*order-continuous-intros*]:
*0 ≤ c ⟹ c < ∞ ⟹ sup-continuous f ⟹ sup-continuous (λx. c * f x :: ereal)*
⟨*proof*⟩

lemma *sup-continuous-ereal-of-enat*[*order-continuous-intros*]:
assumes *f*: *sup-continuous f* **shows** *sup-continuous (λx. ereal-of-enat (f x))*
⟨*proof*⟩

36.4.2 Sums

lemma *sums-ereal-positive*:
fixes *f :: nat ⇒ ereal*
assumes *⋀i. 0 ≤ f i*
shows *f sums (SUP n. ∑ i<n. f i)*
⟨*proof*⟩

lemma *summable-ereal-pos*:
fixes *f :: nat ⇒ ereal*
assumes *⋀i. 0 ≤ f i*
shows *summable f*
⟨*proof*⟩

lemma *sums-ereal*: *(λx. ereal (f x)) sums ereal x ⟷ f sums x*
⟨*proof*⟩

lemma *suminf-ereal-eq-SUP*:
fixes $f :: nat \Rightarrow ereal$
assumes $\bigwedge i.\ 0 \le f\ i$
shows $(\sum x.\ f\ x) = (SUP\ n.\ \sum i{<}n.\ f\ i)$
⟨*proof*⟩

lemma *suminf-bound*:
fixes $f :: nat \Rightarrow ereal$
assumes $\forall N.\ (\sum n{<}N.\ f\ n) \le x$
and *pos*: $\bigwedge n.\ 0 \le f\ n$
shows $suminf\ f \le x$
⟨*proof*⟩

lemma *suminf-bound-add*:
fixes $f :: nat \Rightarrow ereal$
assumes $\forall N.\ (\sum n{<}N.\ f\ n) + y \le x$
and *pos*: $\bigwedge n.\ 0 \le f\ n$
and $y \neq -\infty$
shows $suminf\ f + y \le x$
⟨*proof*⟩

lemma *suminf-upper*:
fixes $f :: nat \Rightarrow ereal$
assumes $\bigwedge n.\ 0 \le f\ n$
shows $(\sum n{<}N.\ f\ n) \le (\sum n.\ f\ n)$
⟨*proof*⟩

lemma *suminf-0-le*:
fixes $f :: nat \Rightarrow ereal$
assumes $\bigwedge n.\ 0 \le f\ n$
shows $0 \le (\sum n.\ f\ n)$
⟨*proof*⟩

lemma *suminf-le-pos*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes $\bigwedge N.\ f\ N \le g\ N$
and $\bigwedge N.\ 0 \le f\ N$
shows $suminf\ f \le suminf\ g$
⟨*proof*⟩

lemma *suminf-half-series-ereal*: $(\sum n.\ (1/2 :: ereal)\ \hat{}\ Suc\ n) = 1$
⟨*proof*⟩

lemma *suminf-add-ereal*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes $\bigwedge i.\ 0 \le f\ i$
and $\bigwedge i.\ 0 \le g\ i$
shows $(\sum i.\ f\ i + g\ i) = suminf\ f + suminf\ g$
⟨*proof*⟩

lemma *suminf-cmult-ereal*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes $\bigwedge i.\ 0 \le f\ i$
and $0 \le a$
shows $(\sum i.\ a * f\ i) = a * suminf\ f$
⟨*proof*⟩

lemma *suminf-PInfty*:
fixes $f :: nat \Rightarrow ereal$
assumes $\bigwedge i.\ 0 \le f\ i$
and $suminf\ f \neq \infty$
shows $f\ i \neq \infty$
⟨*proof*⟩

lemma *suminf-PInfty-fun*:
assumes $\bigwedge i.\ 0 \le f\ i$
and $suminf\ f \neq \infty$
shows $\exists f'.\ f = (\lambda x.\ ereal\ (f'\ x))$
⟨*proof*⟩

lemma *summable-ereal*:
assumes $\bigwedge i.\ 0 \le f\ i$
and $(\sum i.\ ereal\ (f\ i)) \neq \infty$
shows *summable f*
⟨*proof*⟩

lemma *suminf-ereal*:
assumes $\bigwedge i.\ 0 \le f\ i$
and $(\sum i.\ ereal\ (f\ i)) \neq \infty$
shows $(\sum i.\ ereal\ (f\ i)) = ereal\ (suminf\ f)$
⟨*proof*⟩

lemma *suminf-ereal-minus*:
fixes $f\ g :: nat \Rightarrow ereal$
assumes *ord*: $\bigwedge i.\ g\ i \le f\ i\ \bigwedge i.\ 0 \le g\ i$
and *fin*: $suminf\ f \neq \infty\ suminf\ g \neq \infty$
shows $(\sum i.\ f\ i - g\ i) = suminf\ f - suminf\ g$
⟨*proof*⟩

lemma *suminf-ereal-PInf* [*simp*]: $(\sum x.\ \infty :: ereal) = \infty$
⟨*proof*⟩

lemma *summable-real-of-ereal*:
fixes $f :: nat \Rightarrow ereal$
assumes *f*: $\bigwedge i.\ 0 \le f\ i$
and *fin*: $(\sum i.\ f\ i) \neq \infty$
shows *summable* $(\lambda i.\ real\text{-}of\text{-}ereal\ (f\ i))$
⟨*proof*⟩

lemma *suminf-SUP-eq*:
fixes $f :: nat \Rightarrow nat \Rightarrow ereal$
assumes $\bigwedge i.\ incseq\ (\lambda n.\ f\ n\ i)$
and $\bigwedge n\ i.\ 0 \le f\ n\ i$
shows $(\sum i.\ SUP\ n.\ f\ n\ i) = (SUP\ n.\ \sum i.\ f\ n\ i)$
⟨*proof*⟩

lemma *suminf-setsum-ereal*:
fixes $f :: - \Rightarrow - \Rightarrow ereal$
assumes *nonneg*: $\bigwedge i\ a.\ a \in A \Longrightarrow 0 \le f\ i\ a$
shows $(\sum i.\ \sum a{\in}A.\ f\ i\ a) = (\sum a{\in}A.\ \sum i.\ f\ i\ a)$
⟨*proof*⟩

lemma *suminf-ereal-eq-0*:
fixes $f :: nat \Rightarrow ereal$
assumes *nneg*: $\bigwedge i.\ 0 \le f\ i$
shows $(\sum i.\ f\ i) = 0 \longleftrightarrow (\forall i.\ f\ i = 0)$
⟨*proof*⟩

lemma *suminf-ereal-offset-le*:
fixes $f :: nat \Rightarrow ereal$
assumes *f*: $\bigwedge i.\ 0 \le f\ i$
shows $(\sum i.\ f\ (i + k)) \le suminf\ f$
⟨*proof*⟩

lemma *sums-suminf-ereal*: $f\ sums\ x \Longrightarrow (\sum i.\ ereal\ (f\ i)) = ereal\ x$
⟨*proof*⟩

lemma *suminf-ereal'*: $summable\ f \Longrightarrow (\sum i.\ ereal\ (f\ i)) = ereal\ (\sum i.\ f\ i)$
⟨*proof*⟩

lemma *suminf-ereal-finite*: $summable\ f \Longrightarrow (\sum i.\ ereal\ (f\ i)) \neq \infty$
⟨*proof*⟩

lemma *suminf-ereal-finite-neg*:
assumes *summable f*
shows $(\sum x.\ ereal\ (f\ x)) \neq -\infty$
⟨*proof*⟩

lemma *SUP-ereal-add-directed*:
fixes $f\ g :: 'a \Rightarrow ereal$
assumes *nonneg*: $\bigwedge i.\ i \in I \Longrightarrow 0 \le f\ i\ \bigwedge i.\ i \in I \Longrightarrow 0 \le g\ i$
assumes *directed*: $\bigwedge i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow \exists k{\in}I.\ f\ i + g\ j \le f\ k + g\ k$
shows $(SUP\ i{:}I.\ f\ i + g\ i) = (SUP\ i{:}I.\ f\ i) + (SUP\ i{:}I.\ g\ i)$
⟨*proof*⟩

lemma *SUP-ereal-setsum-directed*:
fixes $f\ g :: 'a \Rightarrow 'b \Rightarrow ereal$

assumes $I \neq \{\}$
assumes *directed*: $\bigwedge N\ i\ j.\ N \subseteq A \Longrightarrow i \in I \Longrightarrow j \in I \Longrightarrow \exists k{\in}I.\ \forall n{\in}N.\ f\ n\ i \leq f\ n\ k \wedge f\ n\ j \leq f\ n\ k$
assumes *nonneg*: $\bigwedge n\ i.\ i \in I \Longrightarrow n \in A \Longrightarrow 0 \leq f\ n\ i$
shows $(SUP\ i{:}I.\ \sum n{\in}A.\ f\ n\ i) = (\sum n{\in}A.\ SUP\ i{:}I.\ f\ n\ i)$
⟨*proof*⟩

lemma *suminf-SUP-eq-directed*:
fixes f :: - ⇒ *nat* ⇒ *ereal*
assumes $I \neq \{\}$
assumes *directed*: $\bigwedge N\ i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow$ *finite* $N \Longrightarrow \exists k{\in}I.\ \forall n{\in}N.\ f\ i\ n \leq f\ k\ n \wedge f\ j\ n \leq f\ k\ n$
assumes *nonneg*: $\bigwedge n\ i.\ 0 \leq f\ n\ i$
shows $(\sum i.\ SUP\ n{:}I.\ f\ n\ i) = (SUP\ n{:}I.\ \sum i.\ f\ n\ i)$
⟨*proof*⟩

lemma *ereal-dense3*:
fixes $x\ y$ :: *ereal*
shows $x < y \Longrightarrow \exists r{::}rat.\ x <$ *real-of-rat* $r \wedge$ *real-of-rat* $r < y$
⟨*proof*⟩

lemma *continuous-within-ereal*[*intro*, *simp*]: $x \in A \Longrightarrow$ *continuous* (*at* x *within* A) *ereal*
⟨*proof*⟩

lemma *ereal-open-uminus*:
fixes S :: *ereal set*
assumes *open* S
shows *open* (*uminus* ‘ S)
⟨*proof*⟩

lemma *ereal-uminus-complement*:
fixes S :: *ereal set*
shows *uminus* ‘ $(-\ S) = -$ *uminus* ‘ S
⟨*proof*⟩

lemma *ereal-closed-uminus*:
fixes S :: *ereal set*
assumes *closed* S
shows *closed* (*uminus* ‘ S)
⟨*proof*⟩

lemma *ereal-open-affinity-pos*:
fixes S :: *ereal set*
assumes *open* S
and m: $m \neq \infty\ 0 < m$
and t: $|t| \neq \infty$
shows *open* $((\lambda x.\ m * x + t)$ ‘ $S)$
⟨*proof*⟩

lemma *ereal-open-affinity*:
fixes S :: *ereal set*
assumes *open* S
and m: $|m| \neq \infty$ $m \neq 0$
and t: $|t| \neq \infty$
shows *open* $((\lambda x.\ m * x + t)\ `\ S)$
⟨*proof*⟩

lemma *open-uminus-iff*:
fixes S :: *ereal set*
shows *open* $(uminus\ `\ S) \longleftrightarrow$ *open* S
⟨*proof*⟩

lemma *ereal-Liminf-uminus*:
fixes f :: $'a \Rightarrow$ *ereal*
shows *Liminf net* $(\lambda x.\ -\ (f\ x)) = -$ *Limsup net* f
⟨*proof*⟩

lemma *Liminf-PInfty*:
fixes f :: $'a \Rightarrow$ *ereal*
assumes ¬ *trivial-limit net*
shows $(f \longrightarrow \infty)$ *net* $\longleftrightarrow$ *Liminf net* $f = \infty$
⟨*proof*⟩

lemma *Limsup-MInfty*:
fixes f :: $'a \Rightarrow$ *ereal*
assumes ¬ *trivial-limit net*
shows $(f \longrightarrow -\infty)$ *net* $\longleftrightarrow$ *Limsup net* $f = -\infty$
⟨*proof*⟩

lemma *convergent-ereal*: — RENAME
fixes X :: *nat* $\Rightarrow$ $'a$ :: {*complete-linorder*,*linorder-topology*}
shows *convergent* $X \longleftrightarrow$ *limsup* $X =$ *liminf* X
⟨*proof*⟩

lemma *limsup-le-liminf-real*:
fixes X :: *nat* $\Rightarrow$ *real* **and** L :: *real*
assumes *1*: *limsup* $X \leq L$ **and** *2*: $L \leq$ *liminf* X
shows $X \longrightarrow L$
⟨*proof*⟩

lemma *liminf-PInfty*:
fixes X :: *nat* $\Rightarrow$ *ereal*
shows $X \longrightarrow \infty \longleftrightarrow$ *liminf* $X = \infty$
⟨*proof*⟩

lemma *limsup-MInfty*:
fixes X :: *nat* $\Rightarrow$ *ereal*

shows $X \longrightarrow -\infty \longleftrightarrow \mathit{limsup}\ X = -\infty$
⟨*proof*⟩

lemma *ereal-lim-mono*:
fixes $X\ Y :: \mathit{nat} \Rightarrow 'a$::*linorder-topology*
assumes $\bigwedge n.\ N \leq n \Longrightarrow X\ n \leq Y\ n$
and $X \longrightarrow x$
and $Y \longrightarrow y$
shows $x \leq y$
⟨*proof*⟩

lemma *incseq-le-ereal*:
fixes $X :: \mathit{nat} \Rightarrow 'a$::*linorder-topology*
assumes *inc*: *incseq* X
and *lim*: $X \longrightarrow L$
shows $X\ N \leq L$
⟨*proof*⟩

lemma *decseq-ge-ereal*:
assumes *dec*: *decseq* X
and *lim*: $X \longrightarrow (L::'a$::*linorder-topology*$)$
shows $X\ N \geq L$
⟨*proof*⟩

lemma *bounded-abs*:
fixes $a :: \mathit{real}$
assumes $a \leq x$
and $x \leq b$
shows $|x| \leq \mathit{max}\ |a|\ |b|$
⟨*proof*⟩

lemma *ereal-Sup-lim*:
fixes $a :: 'a$::{*complete-linorder*,*linorder-topology*}
assumes $\bigwedge n.\ b\ n \in s$
and $b \longrightarrow a$
shows $a \leq \mathit{Sup}\ s$
⟨*proof*⟩

lemma *ereal-Inf-lim*:
fixes $a :: 'a$::{*complete-linorder*,*linorder-topology*}
assumes $\bigwedge n.\ b\ n \in s$
and $b \longrightarrow a$
shows $\mathit{Inf}\ s \leq a$
⟨*proof*⟩

lemma *SUP-Lim-ereal*:
fixes $X :: \mathit{nat} \Rightarrow 'a$::{*complete-linorder*,*linorder-topology*}
assumes *inc*: *incseq* X
and l: $X \longrightarrow l$

shows $(SUP\ n.\ X\ n) = l$
⟨*proof*⟩

lemma *INF-Lim-ereal*:
fixes $X :: nat \Rightarrow 'a::\{complete\text{-}linorder,linorder\text{-}topology\}$
assumes *dec*: *decseq X*
and *l*: $X \longrightarrow l$
shows $(INF\ n.\ X\ n) = l$
⟨*proof*⟩

lemma *SUP-eq-LIMSEQ*:
assumes *mono f*
shows $(SUP\ n.\ ereal\ (f\ n)) = ereal\ x \longleftrightarrow f \longrightarrow x$
⟨*proof*⟩

lemma *liminf-ereal-cminus*:
fixes $f :: nat \Rightarrow ereal$
assumes $c \neq -\infty$
shows $liminf\ (\lambda x.\ c - f\ x) = c - limsup\ f$
⟨*proof*⟩

36.4.3 Continuity

lemma *continuous-at-of-ereal*:
$|x0 :: ereal| \neq \infty \Longrightarrow continuous\ (at\ x0)\ real\text{-}of\text{-}ereal$
⟨*proof*⟩

lemma *nhds-ereal*: $nhds\ (ereal\ r) = filtermap\ ereal\ (nhds\ r)$
⟨*proof*⟩

lemma *at-ereal*: $at\ (ereal\ r) = filtermap\ ereal\ (at\ r)$
⟨*proof*⟩

lemma *at-left-ereal*: $at\text{-}left\ (ereal\ r) = filtermap\ ereal\ (at\text{-}left\ r)$
⟨*proof*⟩

lemma *at-right-ereal*: $at\text{-}right\ (ereal\ r) = filtermap\ ereal\ (at\text{-}right\ r)$
⟨*proof*⟩

lemma
shows *at-left-PInf*: $at\text{-}left\ \infty = filtermap\ ereal\ at\text{-}top$
and *at-right-MInf*: $at\text{-}right\ (-\infty) = filtermap\ ereal\ at\text{-}bot$
⟨*proof*⟩

lemma *ereal-tendsto-simps1*:
$((f \circ real\text{-}of\text{-}ereal) \longrightarrow y)\ (at\text{-}left\ (ereal\ x)) \longleftrightarrow (f \longrightarrow y)\ (at\text{-}left\ x)$
$((f \circ real\text{-}of\text{-}ereal) \longrightarrow y)\ (at\text{-}right\ (ereal\ x)) \longleftrightarrow (f \longrightarrow y)\ (at\text{-}right\ x)$
$((f \circ real\text{-}of\text{-}ereal) \longrightarrow y)\ (at\text{-}left\ (\infty::ereal)) \longleftrightarrow (f \longrightarrow y)\ at\text{-}top$
$((f \circ real\text{-}of\text{-}ereal) \longrightarrow y)\ (at\text{-}right\ (-\infty::ereal)) \longleftrightarrow (f \longrightarrow y)\ at\text{-}bot$

⟨*proof*⟩

lemma *ereal-tendsto-simps2*:
((*ereal* ∘ *f*) ⟶ *ereal a*) *F* ⟷ (*f* ⟶ *a*) *F*
((*ereal* ∘ *f*) ⟶ ∞) *F* ⟷ (*LIM x F*. *f x* :> *at-top*)
((*ereal* ∘ *f*) ⟶ −∞) *F* ⟷ (*LIM x F*. *f x* :> *at-bot*)
⟨*proof*⟩

lemma *inverse-infty-ereal-tendsto-0*: *inverse* −∞→ (*0*::*ereal*)
⟨*proof*⟩

lemma *inverse-ereal-tendsto-pos*:
fixes *x* :: *ereal* **assumes** *0* < *x*
shows *inverse* −*x*→ *inverse x*
⟨*proof*⟩

lemma *inverse-ereal-tendsto-at-right-0*: (*inverse* ⟶ ∞) (*at-right* (*0*::*ereal*))
⟨*proof*⟩

lemmas *ereal-tendsto-simps* = *ereal-tendsto-simps1 ereal-tendsto-simps2*

lemma *continuous-at-iff-ereal*:
fixes *f* :: *'a*::*t2-space* ⇒ *real*
shows *continuous* (*at x0 within s*) *f* ⟷ *continuous* (*at x0 within s*) (*ereal* ∘ *f*)
⟨*proof*⟩

lemma *continuous-on-iff-ereal*:
fixes *f* :: *'a*::*t2-space* => *real*
assumes *open A*
shows *continuous-on A f* ⟷ *continuous-on A* (*ereal* ∘ *f*)
⟨*proof*⟩

lemma *continuous-on-real*: *continuous-on* (*UNIV* − {∞, −∞::*ereal*}) *real-of-ereal*
⟨*proof*⟩

lemma *continuous-on-iff-real*:
fixes *f* :: *'a*::*t2-space* ⇒ *ereal*
assumes ∗: ⋀*x*. *x* ∈ *A* ⟹ |*f x*| ≠ ∞
shows *continuous-on A f* ⟷ *continuous-on A* (*real-of-ereal* ∘ *f*)
⟨*proof*⟩

lemma *continuous-uminus-ereal* [*continuous-intros*]: *continuous-on* (*A* :: *ereal set*) *uminus*
⟨*proof*⟩

lemma *ereal-uminus-atMost* [*simp*]: *uminus* ‘ {..(*a*::*ereal*)} = {−*a*..}
⟨*proof*⟩

lemma *continuous-on-inverse-ereal* [*continuous-intros*]:

continuous-on $\{0::ereal\ ..\}$ *inverse*
⟨*proof*⟩

lemma *continuous-inverse-ereal-nonpos*: *continuous-on* $(\{..<0\} :: ereal\ set)$ *inverse*
⟨*proof*⟩

lemma *tendsto-inverse-ereal*:
assumes $(f \longrightarrow (c :: ereal))\ F$
assumes *eventually* $(\lambda x.\ f\ x \geq 0)\ F$
shows $((\lambda x.\ inverse\ (f\ x)) \longrightarrow inverse\ c)\ F$
⟨*proof*⟩

36.4.4 liminf and limsup

lemma *Limsup-ereal-mult-right*:
assumes $F \neq bot\ (c::real) \geq 0$
shows $Limsup\ F\ (\lambda n.\ f\ n * ereal\ c) = Limsup\ F\ f * ereal\ c$
⟨*proof*⟩

lemma *Liminf-ereal-mult-right*:
assumes $F \neq bot\ (c::real) \geq 0$
shows $Liminf\ F\ (\lambda n.\ f\ n * ereal\ c) = Liminf\ F\ f * ereal\ c$
⟨*proof*⟩

lemma *Limsup-ereal-mult-left*:
assumes $F \neq bot\ (c::real) \geq 0$
shows $Limsup\ F\ (\lambda n.\ ereal\ c * f\ n) = ereal\ c * Limsup\ F\ f$
⟨*proof*⟩

lemma *limsup-ereal-mult-right*:
$(c::real) \geq 0 \Longrightarrow limsup\ (\lambda n.\ f\ n * ereal\ c) = limsup\ f * ereal\ c$
⟨*proof*⟩

lemma *limsup-ereal-mult-left*:
$(c::real) \geq 0 \Longrightarrow limsup\ (\lambda n.\ ereal\ c * f\ n) = ereal\ c * limsup\ f$
⟨*proof*⟩

lemma *Limsup-add-ereal-right*:
$F \neq bot \Longrightarrow abs\ c \neq \infty \Longrightarrow Limsup\ F\ (\lambda n.\ g\ n + (c :: ereal)) = Limsup\ F\ g + c$
⟨*proof*⟩

lemma *Limsup-add-ereal-left*:
$F \neq bot \Longrightarrow abs\ c \neq \infty \Longrightarrow Limsup\ F\ (\lambda n.\ (c :: ereal) + g\ n) = c + Limsup\ F\ g$
⟨*proof*⟩

lemma *Liminf-add-ereal-right*:
$F \neq bot \Longrightarrow abs\ c \neq \infty \Longrightarrow Liminf\ F\ (\lambda n.\ g\ n + (c :: ereal)) = Liminf\ F\ g + c$

⟨*proof*⟩

lemma *Liminf-add-ereal-left*:
F ≠ *bot* ⟹ *abs c* ≠ ∞ ⟹ *Liminf F* (λ*n*. (*c* :: *ereal*) + *g n*) = *c* + *Liminf F g*
⟨*proof*⟩

lemma
assumes *F* ≠ *bot*
assumes *nonneg*: *eventually* (λ*x*. *f x* ≥ (*0*::*ereal*)) *F*
shows *Liminf-inverse-ereal*: *Liminf F* (λ*x*. *inverse* (*f x*)) = *inverse* (*Limsup F f*)
and *Limsup-inverse-ereal*: *Limsup F* (λ*x*. *inverse* (*f x*)) = *inverse* (*Liminf F f*)
⟨*proof*⟩

36.4.5 Tests for code generator

value − ∞ :: *ereal*
value |−∞| :: *ereal*
value *4* + *5* / *4* − *ereal 2* :: *ereal*
value *ereal 3* < ∞
value *real-of-ereal* (∞::*ereal*) = *0*

end

37 Radius of Convergence and Summation Tests

theory *Summation*
imports
Complex-Main
~~/*src*/*HOL*/*Library*/*Extended-Real*
~~/*src*/*HOL*/*Library*/*Liminf-Limsup*
begin

The definition of the radius of convergence of a power series, various summability tests, lemmas to compute the radius of convergence etc.

37.1 Rounded dual logarithm

definition *natlog2 n* = (*if n* = *0 then 0 else nat* ⌊*log 2* (*real-of-nat n*)⌋)

lemma *natlog2-0* [*simp*]: *natlog2 0* = *0* ⟨*proof*⟩
lemma *natlog2-1* [*simp*]: *natlog2 1* = *0* ⟨*proof*⟩
lemma *natlog2-eq-0-iff*: *natlog2 n* = *0* ⟷ *n* < *2* ⟨*proof*⟩

lemma *natlog2-power-of-two* [*simp*]: *natlog2* (*2* ^ *n*) = *n*
⟨*proof*⟩

lemma *natlog2-mono*: *m* ≤ *n* ⟹ *natlog2 m* ≤ *natlog2 n*

⟨*proof*⟩

lemma *pow-natlog2-le*: $n > 0 \Longrightarrow 2 \hat{} \text{natlog2}\ n \leq n$
⟨*proof*⟩

lemma *pow-natlog2-gt*: $n > 0 \Longrightarrow 2 * 2 \hat{} \text{natlog2}\ n > n$
and *pow-natlog2-ge*: $n > 0 \Longrightarrow 2 * 2 \hat{} \text{natlog2}\ n \geq n$
⟨*proof*⟩

lemma *natlog2-eqI*:
assumes $n > 0\ 2\hat{}k \leq n\ n < 2 * 2\hat{}k$
shows $\text{natlog2}\ n = k$
⟨*proof*⟩

lemma *natlog2-rec*:
assumes $n \geq 2$
shows $\text{natlog2}\ n = 1 + \text{natlog2}\ (n\ \text{div}\ 2)$
⟨*proof*⟩

fun *natlog2-aux* **where**
$\text{natlog2-aux}\ n\ \text{acc} = (\text{if}\ (n::\text{nat}) < 2\ \text{then acc else natlog2-aux}\ (n\ \text{div}\ 2)\ (\text{acc} + 1))$

lemma *natlog2-aux-correct*:
$\text{natlog2-aux}\ n\ \text{acc} = \text{acc} + \text{natlog2}\ n$
⟨*proof*⟩

lemma *natlog2-code* [*code*]: $\text{natlog2}\ n = \text{natlog2-aux}\ n\ 0$
⟨*proof*⟩

37.2 Convergence tests for infinite sums

37.2.1 Root test

lemma *limsup-root-powser*:
fixes $f :: \text{nat} \Rightarrow 'a :: \{\text{banach}, \text{real-normed-div-algebra}\}$
shows $\text{limsup}\ (\lambda n.\ \text{ereal}\ (\text{root}\ n\ (\text{norm}\ (f\ n * z \hat{}\ n)))) =$
$\text{limsup}\ (\lambda n.\ \text{ereal}\ (\text{root}\ n\ (\text{norm}\ (f\ n)))) * \text{ereal}\ (\text{norm}\ z)$
⟨*proof*⟩

lemma *limsup-root-limit*:
assumes $(\lambda n.\ \text{ereal}\ (\text{root}\ n\ (\text{norm}\ (f\ n)))) \longrightarrow l$ (**is** $?g \longrightarrow$ -)
shows $\text{limsup}\ (\lambda n.\ \text{ereal}\ (\text{root}\ n\ (\text{norm}\ (f\ n)))) = l$
⟨*proof*⟩

lemma *limsup-root-limit'*:
assumes $(\lambda n.\ \text{root}\ n\ (\text{norm}\ (f\ n))) \longrightarrow l$
shows $\text{limsup}\ (\lambda n.\ \text{ereal}\ (\text{root}\ n\ (\text{norm}\ (f\ n)))) = \text{ereal}\ l$
⟨*proof*⟩

lemma *root-test-convergence'*:
fixes $f :: nat \Rightarrow 'a :: banach$
defines $l \equiv limsup\ (\lambda n.\ ereal\ (root\ n\ (norm\ (f\ n))))$
assumes l: $l < 1$
shows *summable f*
⟨*proof*⟩

lemma *root-test-divergence*:
fixes $f :: nat \Rightarrow 'a :: banach$
defines $l \equiv limsup\ (\lambda n.\ ereal\ (root\ n\ (norm\ (f\ n))))$
assumes l: $l > 1$
shows $\neg summable\ f$
⟨*proof*⟩

37.2.2 Cauchy's condensation test

context
fixes $f :: nat \Rightarrow real$
begin

private lemma *condensation-inequality*:
assumes *mono*: $\bigwedge m\ n.\ 0 < m \Longrightarrow m \leq n \Longrightarrow f\ n \leq f\ m$
shows $(\sum k{=}1..{<}n.\ f\ k) \geq (\sum k{=}1..{<}n.\ f\ (2 * 2\ \hat{}\ natlog2\ k))$ (**is** *?thesis1*)
$(\sum k{=}1..{<}n.\ f\ k) \leq (\sum k{=}1..{<}n.\ f\ (2\ \hat{}\ natlog2\ k))$ (**is** *?thesis2*)
⟨*proof*⟩ **lemma** *condensation-condense1*: $(\sum k{=}1..{<}2\hat{}n.\ f\ (2\ \hat{}\ natlog2\ k)) = (\sum k{<}n.\ 2\hat{}k * f\ (2\ \hat{}\ k))$
⟨*proof*⟩ **lemma** *condensation-condense2*: $(\sum k{=}1..{<}2\hat{}n.\ f\ (2 * 2\ \hat{}\ natlog2\ k)) = (\sum k{<}n.\ 2\hat{}k * f\ (2\ \hat{}\ Suc\ k))$
⟨*proof*⟩

lemma *condensation-test*:
assumes *mono*: $\bigwedge m.\ 0 < m \Longrightarrow f\ (Suc\ m) \leq f\ m$
assumes *nonneg*: $\bigwedge n.\ f\ n \geq 0$
shows $summable\ f \longleftrightarrow summable\ (\lambda n.\ 2\hat{}n * f\ (2\hat{}n))$
⟨*proof*⟩

end

37.2.3 Summability of powers

lemma *abs-summable-complex-powr-iff*:
$summable\ (\lambda n.\ norm\ (exp\ (of\text{-}real\ (ln\ (of\text{-}nat\ n)) * s))) \longleftrightarrow Re\ s < -1$
⟨*proof*⟩

lemma *summable-complex-powr-iff*:
assumes $Re\ s < -1$
shows $summable\ (\lambda n.\ exp\ (of\text{-}real\ (ln\ (of\text{-}nat\ n)) * s))$
⟨*proof*⟩

lemma *summable-real-powr-iff*: *summable* (λ*n*. *of-nat n powr s* :: *real*) ⟷ *s* < −*1*
⟨*proof*⟩

lemma *inverse-power-summable*:
assumes *s*: *s* ≥ *2*
shows *summable* (λ*n*. *inverse* (*of-nat n* ^ *s* :: *'a* :: {*real-normed-div-algebra*,*banach*}))
⟨*proof*⟩

lemma *not-summable-harmonic*: ¬*summable* (λ*n*. *inverse* (*of-nat n*) :: *'a* :: *real-normed-field*)
⟨*proof*⟩

37.2.4 Kummer's test

lemma *kummers-test-convergence*:
fixes *f p* :: *nat* ⇒ *real*
assumes *pos-f*: *eventually* (λ*n*. *f n* > *0*) *sequentially*
assumes *nonneg-p*: *eventually* (λ*n*. *p n* ≥ *0*) *sequentially*
defines *l* ≡ *liminf* (λ*n*. *ereal* (*p n* ∗ *f n* / *f* (*Suc n*) − *p* (*Suc n*)))
assumes *l*: *l* > *0*
shows *summable f*
⟨*proof*⟩

lemma *kummers-test-divergence*:
fixes *f p* :: *nat* ⇒ *real*
assumes *pos-f*: *eventually* (λ*n*. *f n* > *0*) *sequentially*
assumes *pos-p*: *eventually* (λ*n*. *p n* > *0*) *sequentially*
assumes *divergent-p*: ¬*summable* (λ*n*. *inverse* (*p n*))
defines *l* ≡ *limsup* (λ*n*. *ereal* (*p n* ∗ *f n* / *f* (*Suc n*) − *p* (*Suc n*)))
assumes *l*: *l* < *0*
shows ¬*summable f*
⟨*proof*⟩

37.2.5 Ratio test

lemma *ratio-test-convergence*:
fixes *f* :: *nat* ⇒ *real*
assumes *pos-f*: *eventually* (λ*n*. *f n* > *0*) *sequentially*
defines *l* ≡ *liminf* (λ*n*. *ereal* (*f n* / *f* (*Suc n*)))
assumes *l*: *l* > *1*
shows *summable f*
⟨*proof*⟩

lemma *ratio-test-divergence*:
fixes *f* :: *nat* ⇒ *real*
assumes *pos-f*: *eventually* (λ*n*. *f n* > *0*) *sequentially*
defines *l* ≡ *limsup* (λ*n*. *ereal* (*f n* / *f* (*Suc n*)))
assumes *l*: *l* < *1*
shows ¬*summable f*

⟨*proof*⟩

37.2.6 Raabe's test

lemma *raabes-test-convergence*:
fixes $f :: nat \Rightarrow real$
  assumes *pos*: *eventually* $(\lambda n.\ f\ n > 0)$ *sequentially*
  defines $l \equiv liminf\ (\lambda n.\ ereal\ (of\text{-}nat\ n * (f\ n\ /\ f\ (Suc\ n) - 1)))$
  assumes *l*: $l > 1$
  shows *summable f*
⟨*proof*⟩

lemma *raabes-test-divergence*:
fixes $f :: nat \Rightarrow real$
  assumes *pos*: *eventually* $(\lambda n.\ f\ n > 0)$ *sequentially*
  defines $l \equiv limsup\ (\lambda n.\ ereal\ (of\text{-}nat\ n * (f\ n\ /\ f\ (Suc\ n) - 1)))$
  assumes *l*: $l < 1$
  shows $\neg summable\ f$
⟨*proof*⟩

37.3 Radius of convergence

The radius of convergence of a power series. This value always exists, ranges from *0* to ∞, and the power series is guaranteed to converge for all inputs with a norm that is smaller than that radius and to diverge for all inputs with a norm that is greater.

definition *conv-radius* :: $(nat \Rightarrow 'a :: banach) \Rightarrow ereal$ **where**
  $conv\text{-}radius\ f = inverse\ (limsup\ (\lambda n.\ ereal\ (root\ n\ (norm\ (f\ n)))))$

lemma *conv-radius-nonneg*: $conv\text{-}radius\ f \geq 0$
⟨*proof*⟩

lemma *conv-radius-zero* [*simp*]: $conv\text{-}radius\ (\lambda\text{-}.\ 0) = \infty$
  ⟨*proof*⟩

lemma *conv-radius-cong*:
  assumes *eventually* $(\lambda x.\ f\ x = g\ x)$ *sequentially*
  shows $conv\text{-}radius\ f = conv\text{-}radius\ g$
⟨*proof*⟩

lemma *conv-radius-altdef*:
  $conv\text{-}radius\ f = liminf\ (\lambda n.\ inverse\ (ereal\ (root\ n\ (norm\ (f\ n)))))$
  ⟨*proof*⟩

lemma *abs-summable-in-conv-radius*:
  fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
  assumes $ereal\ (norm\ z) < conv\text{-}radius\ f$
  shows $summable\ (\lambda n.\ norm\ (f\ n * z\ \hat{}\ n))$
⟨*proof*⟩

lemma *summable-in-conv-radius*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes *ereal* $(norm\ z) < conv\text{-}radius\ f$
shows *summable* $(\lambda n.\ f\ n * z\ \hat{}\ n)$
⟨*proof*⟩

lemma *not-summable-outside-conv-radius*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes *ereal* $(norm\ z) > conv\text{-}radius\ f$
shows $\neg summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
⟨*proof*⟩

lemma *conv-radius-geI*:
assumes *summable* $(\lambda n.\ f\ n * z\ \hat{}\ n :: 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\})$
shows $conv\text{-}radius\ f \geq norm\ z$
⟨*proof*⟩

lemma *conv-radius-leI*:
assumes $\neg summable\ (\lambda n.\ norm\ (f\ n * z\ \hat{}\ n :: 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}))$
shows $conv\text{-}radius\ f \leq norm\ z$
⟨*proof*⟩

lemma *conv-radius-leI'*:
assumes $\neg summable\ (\lambda n.\ f\ n * z\ \hat{}\ n :: 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\})$
shows $conv\text{-}radius\ f \leq norm\ z$
⟨*proof*⟩

lemma *conv-radius-geI-ex*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r < R \Longrightarrow \exists z.\ norm\ z = r \wedge summable\ (\lambda n.\ f\ n * z\hat{}n)$
shows $conv\text{-}radius\ f \geq R$
⟨*proof*⟩

lemma *conv-radius-geI-ex'*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r < R \Longrightarrow summable\ (\lambda n.\ f\ n * of\text{-}real\ r\hat{}n)$
shows $conv\text{-}radius\ f \geq R$
⟨*proof*⟩

lemma *conv-radius-leI-ex*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $R \geq 0$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r > R \Longrightarrow \exists z.\ norm\ z = r \wedge \neg summable\ (\lambda n.\ norm\ (f\ n * z\hat{}n))$
shows $conv\text{-}radius\ f \leq R$
⟨*proof*⟩

lemma *conv-radius-leI-ex'*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $R \geq 0$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r > R \Longrightarrow \neg summable\ (\lambda n.\ f\ n * of\text{-}real\ r\ \hat{}\ n)$
shows *conv-radius* $f \leq R$
⟨*proof*⟩

lemma *conv-radius-eqI*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $R \geq 0$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r < R \Longrightarrow \exists z.\ norm\ z = r \wedge summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r > R \Longrightarrow \exists z.\ norm\ z = r \wedge \neg summable\ (\lambda n.\ norm\ (f\ n * z\ \hat{}\ n))$
shows *conv-radius* $f = R$
⟨*proof*⟩

lemma *conv-radius-eqI'*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $R \geq 0$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r < R \Longrightarrow summable\ (\lambda n.\ f\ n * (of\text{-}real\ r)\ \hat{}\ n)$
assumes $\bigwedge r.\ 0 < r \Longrightarrow ereal\ r > R \Longrightarrow \neg summable\ (\lambda n.\ norm\ (f\ n * (of\text{-}real\ r)\ \hat{}\ n))$
shows *conv-radius* $f = R$
⟨*proof*⟩

lemma *conv-radius-zeroI*:
fixes $f :: nat \Rightarrow 'a :: \{banach,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge z.\ z \neq 0 \Longrightarrow \neg summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
shows *conv-radius* $f = 0$
⟨*proof*⟩

lemma *conv-radius-inftyI'*:
fixes $f :: nat \Rightarrow 'a :: \{banach,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge r.\ r > c \Longrightarrow \exists z.\ norm\ z = r \wedge summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
shows *conv-radius* $f = \infty$
⟨*proof*⟩

lemma *conv-radius-inftyI*:
fixes $f :: nat \Rightarrow 'a :: \{banach,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge r.\ \exists z.\ norm\ z = r \wedge summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
shows *conv-radius* $f = \infty$
⟨*proof*⟩

lemma *conv-radius-inftyI''*:
fixes $f :: nat \Rightarrow 'a :: \{banach,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\bigwedge z.\ summable\ (\lambda n.\ f\ n * z\ \hat{}\ n)$
shows *conv-radius* $f = \infty$

⟨*proof*⟩

lemma *conv-radius-ratio-limit-ereal*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes *nz*: *eventually* $(\lambda n.\ f\ n \neq 0)$ *sequentially*
assumes *lim*: $(\lambda n.\ ereal\ (norm\ (f\ n)\ /\ norm\ (f\ (Suc\ n)))) \longrightarrow c$
shows *conv-radius* $f = c$
⟨*proof*⟩

lemma *conv-radius-ratio-limit-ereal-nonzero*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes *nz*: $c \neq 0$
assumes *lim*: $(\lambda n.\ ereal\ (norm\ (f\ n)\ /\ norm\ (f\ (Suc\ n)))) \longrightarrow c$
shows *conv-radius* $f = c$
⟨*proof*⟩

lemma *conv-radius-ratio-limit*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $c' = ereal\ c$
assumes *nz*: *eventually* $(\lambda n.\ f\ n \neq 0)$ *sequentially*
assumes *lim*: $(\lambda n.\ norm\ (f\ n)\ /\ norm\ (f\ (Suc\ n))) \longrightarrow c$
shows *conv-radius* $f = c'$
⟨*proof*⟩

lemma *conv-radius-ratio-limit-nonzero*:
fixes $f :: nat \Rightarrow 'a :: \{banach, real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $c' = ereal\ c$
assumes *nz*: $c \neq 0$
assumes *lim*: $(\lambda n.\ norm\ (f\ n)\ /\ norm\ (f\ (Suc\ n))) \longrightarrow c$
shows *conv-radius* $f = c'$
⟨*proof*⟩

lemma *conv-radius-mult-power*:
assumes $c \neq (0 :: 'a :: \{real\text{-}normed\text{-}div\text{-}algebra, banach\})$
shows *conv-radius* $(\lambda n.\ c\ \hat{}\ n * f\ n) =$ *conv-radius* $f\ /\ ereal\ (norm\ c)$
⟨*proof*⟩

lemma *conv-radius-mult-power-right*:
assumes $c \neq (0 :: 'a :: \{real\text{-}normed\text{-}div\text{-}algebra, banach\})$
shows *conv-radius* $(\lambda n.\ f\ n * c\ \hat{}\ n) =$ *conv-radius* $f\ /\ ereal\ (norm\ c)$
⟨*proof*⟩

lemma *conv-radius-divide-power*:
assumes $c \neq (0 :: 'a :: \{real\text{-}normed\text{-}div\text{-}algebra, banach\})$
shows *conv-radius* $(\lambda n.\ f\ n\ /\ c\hat{}n) =$ *conv-radius* $f * ereal\ (norm\ c)$
⟨*proof*⟩

lemma *conv-radius-add-ge*:

min (*conv-radius f*) (*conv-radius g*) ≤
conv-radius (λ*x*. *f x* + *g x* :: *'a* :: {*banach*,*real-normed-div-algebra*})
⟨*proof*⟩

lemma *conv-radius-mult-ge*:
fixes *f g* :: *nat* ⇒ (*'a* :: {*banach*,*real-normed-div-algebra*})
shows *conv-radius* (λ*x*. ∑ *i*≤*x*. *f i* ∗ *g* (*x* − *i*)) ≥ *min* (*conv-radius f*) (*conv-radius g*)
⟨*proof*⟩

lemma *le-conv-radius-iff*:
fixes *a* :: *nat* ⇒ *'a*::{*real-normed-div-algebra*,*banach*}
shows *r* ≤ *conv-radius a* ⟷ (∀ *x*. *norm* (*x*−ξ) < *r* ⟶ *summable* (λ*i*. *a i* ∗ (*x* − ξ) ^ *i*))
⟨*proof*⟩

end

38 Uniform Limit and Uniform Convergence

theory *Uniform-Limit*
imports *Topology-Euclidean-Space Summation*
begin

definition *uniformly-on* :: *'a set* ⇒ (*'a* ⇒ *'b*::*metric-space*) ⇒ (*'a* ⇒ *'b*) *filter*
where *uniformly-on S l* = (*INF e*:{*0* <..}. *principal* {*f*. ∀ *x*∈*S*. *dist* (*f x*) (*l x*) < *e*})

abbreviation
uniform-limit S f l ≡ *filterlim f* (*uniformly-on S l*)

definition *uniformly-convergent-on* **where**
uniformly-convergent-on X f ⟷ (∃ *l*. *uniform-limit X f l sequentially*)

definition *uniformly-Cauchy-on* **where**
uniformly-Cauchy-on X f ⟷ (∀ *e*>*0*. ∃ *M*. ∀ *x*∈*X*. ∀ (*m*::*nat*)≥*M*. ∀ *n*≥*M*. *dist* (*f m x*) (*f n x*) < *e*)

lemma *uniform-limit-iff*:
uniform-limit S f l F ⟷ (∀ *e*>*0*. ∀ *F n in F*. ∀ *x*∈*S*. *dist* (*f n x*) (*l x*) < *e*)
⟨*proof*⟩

lemma *uniform-limitD*:
uniform-limit S f l F ⟹ *e* > *0* ⟹ ∀ *F n in F*. ∀ *x*∈*S*. *dist* (*f n x*) (*l x*) < *e*
⟨*proof*⟩

lemma *uniform-limitI*:
(⋀*e*. *e* > *0* ⟹ ∀ *F n in F*. ∀ *x*∈*S*. *dist* (*f n x*) (*l x*) < *e*) ⟹ *uniform-limit S f l F*

⟨*proof*⟩

lemma *uniform-limit-sequentially-iff*:
uniform-limit S f l sequentially ⟷ (∀ *e*>*0*. ∃ *N*. ∀ *n*≥*N*. ∀ *x* ∈ *S*. *dist* (*f n x*) (*l x*) < *e*)
⟨*proof*⟩

lemma *uniform-limit-at-iff*:
uniform-limit S f l (*at x*) ⟷
(∀ *e*>*0*. ∃ *d*>*0*. ∀ *z*. *0* < *dist z x* ∧ *dist z x* < *d* ⟶ (∀ *x*∈*S*. *dist* (*f z x*) (*l x*) < *e*))
⟨*proof*⟩

lemma *uniform-limit-at-le-iff*:
uniform-limit S f l (*at x*) ⟷
(∀ *e*>*0*. ∃ *d*>*0*. ∀ *z*. *0* < *dist z x* ∧ *dist z x* < *d* ⟶ (∀ *x*∈*S*. *dist* (*f z x*) (*l x*) ≤ *e*))
⟨*proof*⟩

lemma *metric-uniform-limit-imp-uniform-limit*:
assumes *f*: *uniform-limit S f a F*
assumes *le*: *eventually* (λ*x*. ∀ *y*∈*S*. *dist* (*g x y*) (*b y*) ≤ *dist* (*f x y*) (*a y*)) *F*
shows *uniform-limit S g b F*
⟨*proof*⟩

lemma *swap-uniform-limit*:
assumes *f*: ∀$_F$ *n in F*. (*f n* ⟶ *g n*) (*at x within S*)
assumes *g*: (*g* ⟶ *l*) *F*
assumes *uc*: *uniform-limit S f h F*
assumes ¬*trivial-limit F*
shows (*h* ⟶ *l*) (*at x within S*)
⟨*proof*⟩

lemma
tendsto-uniform-limitI:
assumes *uniform-limit S f l F*
assumes *x* ∈ *S*
shows ((λ*y*. *f y x*) ⟶ *l x*) *F*
⟨*proof*⟩

lemma *uniform-limit-theorem*:
assumes *c*: ∀$_F$ *n in F*. *continuous-on A* (*f n*)
assumes *ul*: *uniform-limit A f l F*
assumes ¬ *trivial-limit F*
shows *continuous-on A l*
⟨*proof*⟩

lemma *uniformly-Cauchy-onI*:
assumes ⋀*e*. *e* > *0* ⟹ ∃ *M*. ∀ *x*∈*X*. ∀ *m*≥*M*. ∀ *n*≥*M*. *dist* (*f m x*) (*f n x*) < *e*

shows *uniformly-Cauchy-on X f*
⟨*proof*⟩

lemma *uniformly-Cauchy-onI'*:
assumes $\bigwedge e.\ e > 0 \Longrightarrow \exists M.\ \forall x{\in}X.\ \forall m{\geq}M.\ \forall n{>}m.\ dist\ (f\ m\ x)\ (f\ n\ x) < e$
shows *uniformly-Cauchy-on X f*
⟨*proof*⟩

lemma *uniformly-Cauchy-imp-Cauchy*:
uniformly-Cauchy-on X f $\Longrightarrow$ $x \in X \Longrightarrow$ *Cauchy* $(\lambda n.\ f\ n\ x)$
⟨*proof*⟩

lemma *uniform-limit-cong*:
fixes $f\ g :: 'a \Rightarrow 'b \Rightarrow ('c :: metric\text{-}space)$ **and** $h\ i :: 'b \Rightarrow 'c$
assumes *eventually* $(\lambda y.\ \forall x{\in}X.\ f\ y\ x = g\ y\ x)\ F$
assumes $\bigwedge x.\ x \in X \Longrightarrow h\ x = i\ x$
shows *uniform-limit X f h F* $\longleftrightarrow$ *uniform-limit X g i F*
⟨*proof*⟩

lemma *uniform-limit-cong'*:
fixes $f\ g :: 'a \Rightarrow 'b \Rightarrow ('c :: metric\text{-}space)$ **and** $h\ i :: 'b \Rightarrow 'c$
assumes $\bigwedge y\ x.\ x \in X \Longrightarrow f\ y\ x = g\ y\ x$
assumes $\bigwedge x.\ x \in X \Longrightarrow h\ x = i\ x$
shows *uniform-limit X f h F* $\longleftrightarrow$ *uniform-limit X g i F*
⟨*proof*⟩

lemma *uniformly-convergent-uniform-limit-iff*:
uniformly-convergent-on X f $\longleftrightarrow$ *uniform-limit X f* $(\lambda x.\ lim\ (\lambda n.\ f\ n\ x))$ *sequentially*
⟨*proof*⟩

lemma *uniformly-convergentI*: *uniform-limit X f l sequentially* $\Longrightarrow$ *uniformly-convergent-on X f*
⟨*proof*⟩

lemma *uniformly-convergent-on-empty* [*iff*]: *uniformly-convergent-on* $\{\}$ *f*
⟨*proof*⟩

lemma *Cauchy-uniformly-convergent*:
fixes $f :: nat \Rightarrow 'a \Rightarrow 'b :: complete\text{-}space$
assumes *uniformly-Cauchy-on X f*
shows *uniformly-convergent-on X f*
⟨*proof*⟩

lemma *uniformly-convergent-imp-convergent*:
uniformly-convergent-on X f $\Longrightarrow$ $x \in X \Longrightarrow$ *convergent* $(\lambda n.\ f\ n\ x)$
⟨*proof*⟩

lemma *weierstrass-m-test-ev*:
fixes $f :: - \Rightarrow - \Rightarrow - :: banach$

assumes *eventually* $(\lambda n.\ \forall x{\in}A.\ norm\ (f\ n\ x) \leq M\ n)$ *sequentially*
assumes *summable M*
shows *uniform-limit A* $(\lambda n\ x.\ \sum i{<}n.\ f\ i\ x)$ $(\lambda x.\ suminf\ (\lambda i.\ f\ i\ x))$ *sequentially*
⟨*proof*⟩

Alternative version, formulated as in HOL Light

corollary *series-comparison-uniform*:
fixes f :: - ⇒ *nat* ⇒ - :: *banach*
assumes *g*: *summable g* **and** *le*: $\bigwedge n\ x.\ N \leq n \wedge x \in A \Longrightarrow norm(f\ x\ n) \leq g\ n$
shows $\exists l.\ \forall e.\ 0 < e \longrightarrow (\exists N.\ \forall n\ x.\ N \leq n \wedge x \in A \longrightarrow dist(setsum\ (f\ x)\ \{..{<}n\})\ (l\ x) < e)$
⟨*proof*⟩

corollary *weierstrass-m-test*:
fixes f :: - ⇒ - ⇒ - :: *banach*
assumes $\bigwedge n\ x.\ x \in A \Longrightarrow norm\ (f\ n\ x) \leq M\ n$
assumes *summable M*
shows *uniform-limit A* $(\lambda n\ x.\ \sum i{<}n.\ f\ i\ x)$ $(\lambda x.\ suminf\ (\lambda i.\ f\ i\ x))$ *sequentially*
⟨*proof*⟩

corollary *weierstrass-m-test′-ev*:
fixes f :: - ⇒ - ⇒ - :: *banach*
assumes *eventually* $(\lambda n.\ \forall x{\in}A.\ norm\ (f\ n\ x) \leq M\ n)$ *sequentially summable M*

shows *uniformly-convergent-on A* $(\lambda n\ x.\ \sum i{<}n.\ f\ i\ x)$
⟨*proof*⟩

corollary *weierstrass-m-test′*:
fixes f :: - ⇒ - ⇒ - :: *banach*
assumes $\bigwedge n\ x.\ x \in A \Longrightarrow norm\ (f\ n\ x) \leq M\ n$ *summable M*
shows *uniformly-convergent-on A* $(\lambda n\ x.\ \sum i{<}n.\ f\ i\ x)$
⟨*proof*⟩

lemma *uniform-limit-eq-rhs*: *uniform-limit X f l F* $\Longrightarrow l = m \Longrightarrow$ *uniform-limit X f m F*
⟨*proof*⟩

named-theorems *uniform-limit-intros introduction rules for uniform-limit*
⟨*ML*⟩

lemma (**in** *bounded-linear*) *uniform-limit*[*uniform-limit-intros*]:
assumes *uniform-limit X g l F*
shows *uniform-limit X* $(\lambda a\ b.\ f\ (g\ a\ b))$ $(\lambda a.\ f\ (l\ a))$ *F*
⟨*proof*⟩

lemmas *bounded-linear-uniform-limit-intros*[*uniform-limit-intros*] =
bounded-linear.uniform-limit[*OF bounded-linear-Im*]
bounded-linear.uniform-limit[*OF bounded-linear-Re*]
bounded-linear.uniform-limit[*OF bounded-linear-cnj*]

bounded-linear.uniform-limit[*OF bounded-linear-fst*]
bounded-linear.uniform-limit[*OF bounded-linear-snd*]
bounded-linear.uniform-limit[*OF bounded-linear-zero*]
bounded-linear.uniform-limit[*OF bounded-linear-of-real*]
bounded-linear.uniform-limit[*OF bounded-linear-inner-left*]
bounded-linear.uniform-limit[*OF bounded-linear-inner-right*]
bounded-linear.uniform-limit[*OF bounded-linear-divide*]
bounded-linear.uniform-limit[*OF bounded-linear-scaleR-right*]
bounded-linear.uniform-limit[*OF bounded-linear-mult-left*]
bounded-linear.uniform-limit[*OF bounded-linear-mult-right*]
bounded-linear.uniform-limit[*OF bounded-linear-scaleR-left*]

lemmas *uniform-limit-uminus*[*uniform-limit-intros*] =
bounded-linear.uniform-limit[*OF bounded-linear-minus*[*OF bounded-linear-ident*]]

lemma *uniform-limit-const*[*uniform-limit-intros*]: *uniform-limit S* (λ*x y*. *c*) (λ*x*. *c*) *f*
⟨*proof*⟩

lemma *uniform-limit-add*[*uniform-limit-intros*]:
fixes *f g*::'*a* ⇒ '*b* ⇒ '*c*::*real-normed-vector*
assumes *uniform-limit X f l F*
assumes *uniform-limit X g m F*
shows *uniform-limit X* (λ*a b*. *f a b* + *g a b*) (λ*a*. *l a* + *m a*) *F*
⟨*proof*⟩

lemma *uniform-limit-minus*[*uniform-limit-intros*]:
fixes *f g*::'*a* ⇒ '*b* ⇒ '*c*::*real-normed-vector*
assumes *uniform-limit X f l F*
assumes *uniform-limit X g m F*
shows *uniform-limit X* (λ*a b*. *f a b* − *g a b*) (λ*a*. *l a* − *m a*) *F*
⟨*proof*⟩

lemma *uniform-limit-norm*[*uniform-limit-intros*]:
assumes *uniform-limit S g l f*
shows *uniform-limit S* (λ*x y*. *norm* (*g x y*)) (λ*x*. *norm* (*l x*)) *f*
⟨*proof*⟩

lemma (**in** *bounded-bilinear*) *bounded-uniform-limit*[*uniform-limit-intros*]:
assumes *uniform-limit X f l F*
assumes *uniform-limit X g m F*
assumes *bounded* (*m* ‘ *X*)
assumes *bounded* (*l* ‘ *X*)
shows *uniform-limit X* (λ*a b*. *prod* (*f a b*) (*g a b*)) (λ*a*. *prod* (*l a*) (*m a*)) *F*
⟨*proof*⟩

lemmas *bounded-bilinear-bounded-uniform-limit-intros*[*uniform-limit-intros*] =
bounded-bilinear.bounded-uniform-limit[*OF Inner-Product.bounded-bilinear-inner*]
bounded-bilinear.bounded-uniform-limit[*OF Real-Vector-Spaces.bounded-bilinear-mult*]

bounded-bilinear.bounded-uniform-limit[*OF Real-Vector-Spaces.bounded-bilinear-scaleR*]

lemma *uniform-limit-null-comparison*:
assumes $\forall_F\ x\ in\ F.\ \forall\ a{\in}S.\ norm\ (f\ x\ a) \leq g\ x\ a$
assumes *uniform-limit S g* (λ-. *0*) *F*
shows *uniform-limit S f* (λ-. *0*) *F*
⟨*proof*⟩

lemma *uniform-limit-on-union*:
uniform-limit I f g F $\Longrightarrow$ *uniform-limit J f g F* $\Longrightarrow$ *uniform-limit* $(I \cup J)$ *f g F*
⟨*proof*⟩

lemma *uniform-limit-on-empty* [*iff*]:
uniform-limit {} *f g F*
⟨*proof*⟩

lemma *uniform-limit-on-UNION*:
assumes *finite S*
assumes $\bigwedge s.\ s \in S \Longrightarrow$ *uniform-limit* (*h s*) *f g F*
shows *uniform-limit* (*UNION S h*) *f g F*
⟨*proof*⟩

lemma *uniform-limit-on-Union*:
assumes *finite I*
assumes $\bigwedge J.\ J \in I \Longrightarrow$ *uniform-limit J f g F*
shows *uniform-limit* (*Union I*) *f g F*
⟨*proof*⟩

lemma *uniform-limit-on-subset*:
uniform-limit J f g F $\Longrightarrow$ $I \subseteq J \Longrightarrow$ *uniform-limit I f g F*
⟨*proof*⟩

lemma *uniformly-convergent-add*:
uniformly-convergent-on A f $\Longrightarrow$ *uniformly-convergent-on A g*$\Longrightarrow$
uniformly-convergent-on A ($\lambda k\ x.\ f\ k\ x + g\ k\ x :: 'a :: \{$*real-normed-algebra*$\}$)
⟨*proof*⟩

lemma *uniformly-convergent-minus*:
uniformly-convergent-on A f $\Longrightarrow$ *uniformly-convergent-on A g*$\Longrightarrow$
uniformly-convergent-on A ($\lambda k\ x.\ f\ k\ x - g\ k\ x :: 'a :: \{$*real-normed-algebra*$\}$)
⟨*proof*⟩

lemma *uniformly-convergent-mult*:
uniformly-convergent-on A f $\Longrightarrow$
uniformly-convergent-on A ($\lambda k\ x.\ c * f\ k\ x :: 'a :: \{$*real-normed-algebra*$\}$)
⟨*proof*⟩

38.1 Power series and uniform convergence

proposition *powser-uniformly-convergent*:
fixes $a :: nat \Rightarrow 'a::\{real\text{-}normed\text{-}div\text{-}algebra, banach\}$
assumes $r < conv\text{-}radius\ a$
shows *uniformly-convergent-on* $(cball\ \xi\ r)\ (\lambda n\ x.\ \sum i<n.\ a\ i * (x - \xi)\ \hat{}\ i)$
⟨*proof*⟩

lemma *powser-uniform-limit*:
fixes $a :: nat \Rightarrow 'a::\{real\text{-}normed\text{-}div\text{-}algebra, banach\}$
assumes $r < conv\text{-}radius\ a$
shows *uniform-limit* $(cball\ \xi\ r)\ (\lambda n\ x.\ \sum i<n.\ a\ i * (x - \xi)\ \hat{}\ i)\ (\lambda x.\ suminf\ (\lambda i.\ a\ i * (x - \xi)\ \hat{}\ i))\ sequentially$
⟨*proof*⟩

lemma *powser-continuous-suminf*:
fixes $a :: nat \Rightarrow 'a::\{real\text{-}normed\text{-}div\text{-}algebra, banach\}$
assumes $r < conv\text{-}radius\ a$
shows *continuous-on* $(cball\ \xi\ r)\ (\lambda x.\ suminf\ (\lambda i.\ a\ i * (x - \xi)\ \hat{}\ i))$
⟨*proof*⟩

lemma *powser-continuous-sums*:
fixes $a :: nat \Rightarrow 'a::\{real\text{-}normed\text{-}div\text{-}algebra, banach\}$
assumes r: $r < conv\text{-}radius\ a$
and *sm*: $\bigwedge x.\ x \in cball\ \xi\ r \Longrightarrow (\lambda n.\ a\ n * (x - \xi)\ \hat{}\ n)\ sums\ (f\ x)$
shows *continuous-on* $(cball\ \xi\ r)\ f$
⟨*proof*⟩

end

39 Bounded Linear Function

theory *Bounded-Linear-Function*
imports
Topology-Euclidean-Space
Operator-Norm
begin

39.1 Intro rules for *bounded-linear*

named-theorems *bounded-linear-intros*

lemma *onorm-inner-left*:
assumes *bounded-linear r*
shows $onorm\ (\lambda x.\ r\ x \cdot f) \leq onorm\ r * norm\ f$
⟨*proof*⟩

lemma *onorm-inner-right*:
assumes *bounded-linear r*

shows *onorm* $(\lambda x.\ f \cdot r\ x) \leq$ *norm* $f *$ *onorm* r
⟨*proof*⟩

lemmas [*bounded-linear-intros*] =
bounded-linear-zero
bounded-linear-add
bounded-linear-const-mult
bounded-linear-mult-const
bounded-linear-scaleR-const
bounded-linear-const-scaleR
bounded-linear-ident
bounded-linear-setsum
bounded-linear-Pair
bounded-linear-sub
bounded-linear-fst-comp
bounded-linear-snd-comp
bounded-linear-inner-left-comp
bounded-linear-inner-right-comp

39.2 declaration of derivative/continuous/tendsto introduction rules for bounded linear functions

⟨*ML*⟩

39.3 type of bounded linear functions

typedef (**overloaded**) ('*a*, '*b*) *blinfun* ((- $\Rightarrow_L$ /-) [*22*, *21*] *21*) =
{*f*::'*a*::*real-normed-vector*$\Rightarrow$'*b*::*real-normed-vector*. *bounded-linear f*}
morphisms *blinfun-apply Blinfun*
⟨*proof*⟩

declare [[*coercion*
blinfun-apply :: ('*a*::*real-normed-vector* $\Rightarrow_L$ '*b*::*real-normed-vector*) $\Rightarrow$ '*a* $\Rightarrow$ '*b*]]

lemma *bounded-linear-blinfun-apply*[*bounded-linear-intros*]:
bounded-linear g $\Longrightarrow$ *bounded-linear* $(\lambda x.$ *blinfun-apply f* $(g\ x))$
⟨*proof*⟩

setup-lifting *type-definition-blinfun*

lemma *blinfun-eqI*: $(\bigwedge i.$ *blinfun-apply x i* = *blinfun-apply y i*$) \Longrightarrow x = y$
⟨*proof*⟩

lemma *bounded-linear-Blinfun-apply*: *bounded-linear f* $\Longrightarrow$ *blinfun-apply* (*Blinfun f*) = *f*
⟨*proof*⟩

39.4 type class instantiations

instantiation *blinfun* :: (*real-normed-vector*, *real-normed-vector*) *real-normed-vector*

begin

lift-definition *norm-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *real* **is** *onorm* ⟨*proof*⟩

lift-definition *minus-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b*
is $\lambda f\ g\ x.\ f\ x - g\ x$
⟨*proof*⟩

definition *dist-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *real*
where *dist-blinfun a b* = *norm* (*a* − *b*)

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

definition *open-blinfun* :: (*'a* $\Rightarrow_L$ *'b*) *set* $\Rightarrow$ *bool*
where [*code del*]: *open-blinfun S* = (∀ *x*∈*S*. ∀$_F$ (*x'*, *y*) *in uniformity*. *x'* = *x* ⟶ *y* ∈ *S*)

lift-definition *uminus-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* **is** $\lambda f\ x.\ -\ f\ x$
⟨*proof*⟩

lift-definition *zero-blinfun* :: *'a* $\Rightarrow_L$ *'b* **is** $\lambda x.\ 0$
⟨*proof*⟩

lift-definition *plus-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b*
is $\lambda f\ g\ x.\ f\ x + g\ x$
⟨*proof*⟩

lift-definition *scaleR-blinfun*::*real* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b* **is** $\lambda r\ f\ x.\ r *_R f\ x$
⟨*proof*⟩

definition *sgn-blinfun* :: *'a* $\Rightarrow_L$ *'b* $\Rightarrow$ *'a* $\Rightarrow_L$ *'b*
where *sgn-blinfun x* = *scaleR* (*inverse* (*norm x*)) *x*

instance
⟨*proof*⟩

end

declare *uniformity-Abort*[**where** *'a*=(*'a* :: *real-normed-vector*) $\Rightarrow_L$ (*'b* :: *real-normed-vector*), *code*]

lemma *norm-blinfun-eqI*:
assumes *n* ≤ *norm* (*blinfun-apply f x*) / *norm x*
assumes ⋀*x*. *norm* (*blinfun-apply f x*) ≤ *n* ∗ *norm x*
assumes *0* ≤ *n*
shows *norm f* = *n*
⟨*proof*⟩

lemma *norm-blinfun*: *norm* (*blinfun-apply f x*) ≤ *norm f* ∗ *norm x*
⟨*proof*⟩

lemma *norm-blinfun-bound*: *0* ≤ *b* ⟹ (⋀*x*. *norm* (*blinfun-apply f x*) ≤ *b* ∗ *norm x*) ⟹ *norm f* ≤ *b*
⟨*proof*⟩

lemma *bounded-bilinear-blinfun-apply*[*bounded-bilinear*]: *bounded-bilinear blinfun-apply*
⟨*proof*⟩

interpretation *blinfun*: *bounded-bilinear blinfun-apply*
⟨*proof*⟩

lemmas *bounded-linear-apply-blinfun*[*intro*, *simp*] = *blinfun.bounded-linear-left*

context *bounded-bilinear*
begin

named-theorems *bilinear-simps*

lemmas [*bilinear-simps*] =
add-left
add-right
diff-left
diff-right
minus-left
minus-right
scaleR-left
scaleR-right
zero-left
zero-right
setsum-left
setsum-right

end

instance *blinfun* :: (*banach*, *banach*) *banach*
⟨*proof*⟩

39.5 On Euclidean Space

lemma *Zfun-setsum*:
assumes *finite s*
assumes *f*: ⋀*i*. *i* ∈ *s* ⟹ *Zfun* (*f i*) *F*
shows *Zfun* (λ*x*. *setsum* (λ*i*. *f i x*) *s*) *F*
⟨*proof*⟩

lemma *norm-blinfun-euclidean-le*:
fixes a::$'a$::*euclidean-space* $\Rightarrow_L$ $'b$::*real-normed-vector*
shows *norm* $a \leq$ *setsum* $(\lambda x.$ *norm* $(a\ x))$ *Basis*
⟨*proof*⟩

lemma *tendsto-componentwise1*:
fixes a::$'a$::*euclidean-space* $\Rightarrow_L$ $'b$::*real-normed-vector*
and b::$'c \Rightarrow$ $'a \Rightarrow_L$ $'b$
assumes $(\bigwedge j.\ j \in$ *Basis* $\Longrightarrow ((\lambda n.\ b\ n\ j) \longrightarrow a\ j)\ F)$
shows $(b \longrightarrow a)\ F$
⟨*proof*⟩

lift-definition
blinfun-of-matrix::($'b$::*euclidean-space* $\Rightarrow$ $'a$::*euclidean-space* $\Rightarrow$ *real*) $\Rightarrow$ $'a \Rightarrow_L$ $'b$
is $\lambda a\ x.\ \sum i \in$ *Basis*. $\sum j \in$ *Basis*. $((x \cdot j) * a\ i\ j) *_R i$
⟨*proof*⟩

lemma *blinfun-of-matrix-works*:
fixes f::$'a$::*euclidean-space* $\Rightarrow_L$ $'b$::*euclidean-space*
shows *blinfun-of-matrix* $(\lambda i\ j.\ (f\ j) \cdot i) = f$
⟨*proof*⟩

lemma *blinfun-of-matrix-apply*:
blinfun-of-matrix a x $= (\sum i \in$ *Basis*. $\sum j \in$ *Basis*. $((x \cdot j) * a\ i\ j) *_R i)$
⟨*proof*⟩

lemma *blinfun-of-matrix-minus*: *blinfun-of-matrix x* − *blinfun-of-matrix y* = *blinfun-of-matrix* $(x - y)$
⟨*proof*⟩

lemma *norm-blinfun-of-matrix*:
norm (*blinfun-of-matrix a*) $\leq (\sum i \in$ *Basis*. $\sum j \in$ *Basis*. $|a\ i\ j|)$
⟨*proof*⟩

lemma *tendsto-blinfun-of-matrix*:
assumes $\bigwedge i\ j.\ i \in$ *Basis* $\Longrightarrow j \in$ *Basis* $\Longrightarrow ((\lambda n.\ b\ n\ i\ j) \longrightarrow a\ i\ j)\ F$
shows $((\lambda n.$ *blinfun-of-matrix* $(b\ n)) \longrightarrow$ *blinfun-of-matrix a*$)\ F$
⟨*proof*⟩

lemma *tendsto-componentwise*:
fixes a::$'a$::*euclidean-space* $\Rightarrow_L$ $'b$::*euclidean-space*
and b::$'c \Rightarrow$ $'a \Rightarrow_L$ $'b$
shows $(\bigwedge i\ j.\ i \in$ *Basis* $\Longrightarrow j \in$ *Basis* $\Longrightarrow ((\lambda n.\ b\ n\ j \cdot i) \longrightarrow a\ j \cdot i)\ F) \Longrightarrow$
$(b \longrightarrow a)\ F$
⟨*proof*⟩

lemma
continuous-blinfun-componentwiseI:

fixes f:: $'b$::*t2-space* $\Rightarrow$ $'a$::*euclidean-space* $\Rightarrow_L$ $'c$::*euclidean-space*
assumes $\bigwedge i\ j.\ i \in Basis \Longrightarrow j \in Basis \Longrightarrow$ *continuous* $F\ (\lambda x.\ (f\ x)\ j \cdot i)$
shows *continuous* $F\ f$
⟨*proof*⟩

lemma
continuous-blinfun-componentwiseI1:
fixes f:: $'b$::*t2-space* $\Rightarrow$ $'a$::*euclidean-space* $\Rightarrow_L$ $'c$::*real-normed-vector*
assumes $\bigwedge i.\ i \in Basis \Longrightarrow$ *continuous* $F\ (\lambda x.\ f\ x\ i)$
shows *continuous* $F\ f$
⟨*proof*⟩

lemma *bounded-linear-blinfun-matrix*: *bounded-linear* $(\lambda x.\ (x$::-$\Rightarrow_L$ -$)\ j \cdot i)$
⟨*proof*⟩

lemma *continuous-blinfun-matrix*:
fixes f:: $'b$::*t2-space* $\Rightarrow$ $'a$::*real-normed-vector* $\Rightarrow_L$ $'c$::*real-inner*
assumes *continuous* $F\ f$
shows *continuous* $F\ (\lambda x.\ (f\ x)\ j \cdot i)$
⟨*proof*⟩

lemma *continuous-on-blinfun-matrix*:
fixes f::$'a$::*t2-space* $\Rightarrow$ $'b$::*real-normed-vector* $\Rightarrow_L$ $'c$::*real-inner*
assumes *continuous-on* $S\ f$
shows *continuous-on* $S\ (\lambda x.\ (f\ x)\ j \cdot i)$
⟨*proof*⟩

lemma *continuous-on-blinfun-of-matrix*[*continuous-intros*]:
assumes $\bigwedge i\ j.\ i \in Basis \Longrightarrow j \in Basis \Longrightarrow$ *continuous-on* $S\ (\lambda s.\ g\ s\ i\ j)$
shows *continuous-on* $S\ (\lambda s.$ *blinfun-of-matrix* $(g\ s))$
⟨*proof*⟩

lemma *mult-if-delta*:
$(if\ P\ then\ (1$::$'a$::*comm-semiring-1*$)\ else\ 0) * q = (if\ P\ then\ q\ else\ 0)$
⟨*proof*⟩

lemma *compact-blinfun-lemma*:
fixes f :: *nat* $\Rightarrow$ $'a$::*euclidean-space* $\Rightarrow_L$ $'b$::*euclidean-space*
assumes *bounded* (*range* f)
shows $\forall\, d \subseteq Basis.\ \exists\, l$::$'a \Rightarrow_L$ $'b.\ \exists\ r.$
subseq $r \wedge (\forall\, e{>}0.$ *eventually* $(\lambda n.\ \forall\, i{\in}d.$ *dist* $(f\ (r\ n)\ i)\ (l\ i) < e)$ *sequentially*$)$
⟨*proof*⟩

lemma *blinfun-euclidean-eqI*: $(\bigwedge i.\ i \in Basis \Longrightarrow$ *blinfun-apply* $x\ i =$ *blinfun-apply* $y\ i) \Longrightarrow x = y$
⟨*proof*⟩

lemma *Blinfun-eq-matrix*: *bounded-linear* $f \Longrightarrow$ *Blinfun* $f =$ *blinfun-of-matrix* $(\lambda i\ j.\ f\ j \cdot i)$

⟨*proof*⟩

TODO: generalize (via *compact* (*cball ?x ?e*))?

instance *blinfun* :: (*euclidean-space*, *euclidean-space*) *heine-borel*
⟨*proof*⟩

39.6 concrete bounded linear functions

lemma *transfer-bounded-bilinear-bounded-linearI*:
assumes $g = (\lambda i\ x.\ (blinfun\text{-}apply\ (f\ i)\ x))$
shows *bounded-bilinear g* = *bounded-linear f*
⟨*proof*⟩

lemma *transfer-bounded-bilinear-bounded-linear*[*transfer-rule*]:
(*rel-fun* (*rel-fun op* = (*pcr-blinfun op* = *op* =)) *op* =) *bounded-bilinear bounded-linear*
⟨*proof*⟩

context *bounded-bilinear*
begin

lift-definition *prod-left*::$'b \Rightarrow\ 'a \Rightarrow_L\ 'c$ **is** $(\lambda b\ a.\ prod\ a\ b)$
⟨*proof*⟩
declare *prod-left.rep-eq*[*simp*]

lemma *bounded-linear-prod-left*[*bounded-linear*]: *bounded-linear prod-left*
⟨*proof*⟩

lift-definition *prod-right*::$'a \Rightarrow\ 'b \Rightarrow_L\ 'c$ **is** $(\lambda a\ b.\ prod\ a\ b)$
⟨*proof*⟩
declare *prod-right.rep-eq*[*simp*]

lemma *bounded-linear-prod-right*[*bounded-linear*]: *bounded-linear prod-right*
⟨*proof*⟩

end

lift-definition *id-blinfun*::$'a$::*real-normed-vector* $\Rightarrow_L\ 'a$ **is** $\lambda x.\ x$
⟨*proof*⟩

lemmas *blinfun-apply-id-blinfun*[*simp*] = *id-blinfun.rep-eq*

lemma *norm-blinfun-id*[*simp*]:
norm (*id-blinfun*::$'a$::{*real-normed-vector*, *perfect-space*} $\Rightarrow_L\ 'a$) = *1*
⟨*proof*⟩

lemma *norm-blinfun-id-le*:
norm (*id-blinfun*::$'a$::*real-normed-vector* $\Rightarrow_L\ 'a$) $\leq$ *1*
⟨*proof*⟩

lift-definition *fst-blinfun*::('*a*::*real-normed-vector* × '*b*::*real-normed-vector*) ⇒$_L$ '*a*
is *fst*
⟨*proof*⟩

lemma *blinfun-apply-fst-blinfun*[*simp*]: *blinfun-apply fst-blinfun* = *fst*
⟨*proof*⟩

lift-definition *snd-blinfun*::('*a*::*real-normed-vector* × '*b*::*real-normed-vector*) ⇒$_L$
'*b* **is** *snd*
⟨*proof*⟩

lemma *blinfun-apply-snd-blinfun*[*simp*]: *blinfun-apply snd-blinfun* = *snd*
⟨*proof*⟩

lift-definition *blinfun-compose*::
'*a*::*real-normed-vector* ⇒$_L$ '*b*::*real-normed-vector* ⇒
'*c*::*real-normed-vector* ⇒$_L$ '*a* ⇒
'*c* ⇒$_L$ '*b* (**infixl** o_L 55) **is** *op o*
parametric *comp-transfer*
⟨*proof*⟩

lemma *blinfun-apply-blinfun-compose*[*simp*]: (*a* o_L *b*) *c* = *a* (*b c*)
⟨*proof*⟩

lemma *norm-blinfun-compose*:
norm (*f* o_L *g*) ≤ *norm f* ∗ *norm g*
⟨*proof*⟩

lemma *bounded-bilinear-blinfun-compose*[*bounded-bilinear*]: *bounded-bilinear op* o_L
⟨*proof*⟩

lemma *blinfun-compose-zero*[*simp*]:
blinfun-compose 0 = (λ-. *0*)
blinfun-compose x 0 = *0*
⟨*proof*⟩

lift-definition *blinfun-inner-right*::'*a*::*real-inner* ⇒ '*a* ⇒$_L$ *real* **is** *op* ·
⟨*proof*⟩
declare *blinfun-inner-right.rep-eq*[*simp*]

lemma *bounded-linear-blinfun-inner-right*[*bounded-linear*]: *bounded-linear blinfun-inner-right*
⟨*proof*⟩

lift-definition *blinfun-inner-left*::'*a*::*real-inner* ⇒ '*a* ⇒$_L$ *real* **is** λ*x y*. *y* · *x*

⟨*proof*⟩
declare *blinfun-inner-left.rep-eq*[*simp*]

lemma *bounded-linear-blinfun-inner-left*[*bounded-linear*]: *bounded-linear blinfun-inner-left*
⟨*proof*⟩

lift-definition *blinfun-scaleR-right*::*real* ⇒ *'a* ⇒$_L$ *'a*::*real-normed-vector* **is** *op* ∗$_R$
⟨*proof*⟩
declare *blinfun-scaleR-right.rep-eq*[*simp*]

lemma *bounded-linear-blinfun-scaleR-right*[*bounded-linear*]: *bounded-linear blinfun-scaleR-right*
⟨*proof*⟩

lift-definition *blinfun-scaleR-left*::*'a*::*real-normed-vector* ⇒ *real* ⇒$_L$ *'a* **is** λ*x y*. *y* ∗$_R$ *x*
⟨*proof*⟩
lemmas [*simp*] = *blinfun-scaleR-left.rep-eq*

lemma *bounded-linear-blinfun-scaleR-left*[*bounded-linear*]: *bounded-linear blinfun-scaleR-left*
⟨*proof*⟩

lift-definition *blinfun-mult-right*::*'a* ⇒ *'a* ⇒$_L$ *'a*::*real-normed-algebra* **is** *op* ∗
⟨*proof*⟩
declare *blinfun-mult-right.rep-eq*[*simp*]

lemma *bounded-linear-blinfun-mult-right*[*bounded-linear*]: *bounded-linear blinfun-mult-right*
⟨*proof*⟩

lift-definition *blinfun-mult-left*::*'a*::*real-normed-algebra* ⇒ *'a* ⇒$_L$ *'a* **is** λ*x y*. *y* ∗ *x*
⟨*proof*⟩
lemmas [*simp*] = *blinfun-mult-left.rep-eq*

lemma *bounded-linear-blinfun-mult-left*[*bounded-linear*]: *bounded-linear blinfun-mult-left*
⟨*proof*⟩

end

40 Multivariate calculus in Euclidean space

theory *Derivative*
imports *Brouwer-Fixpoint Operator-Norm Uniform-Limit Bounded-Linear-Function*
begin

lemma *onorm-inner-left*:

assumes *bounded-linear r*
shows *onorm* $(\lambda x.\ r\ x \cdot f) \leq$ *onorm r* $*$ *norm f*
⟨*proof*⟩

lemma *onorm-inner-right*:
assumes *bounded-linear r*
shows *onorm* $(\lambda x.\ f \cdot r\ x) \leq$ *norm f* $*$ *onorm r*
⟨*proof*⟩

declare *has-derivative-bounded-linear*[*dest*]

40.1 Derivatives

40.1.1 Combining theorems.

lemmas *has-derivative-id* = *has-derivative-ident*
lemmas *has-derivative-neg* = *has-derivative-minus*
lemmas *has-derivative-sub* = *has-derivative-diff*
lemmas *scaleR-right-has-derivative* = *has-derivative-scaleR-right*
lemmas *scaleR-left-has-derivative* = *has-derivative-scaleR-left*
lemmas *inner-right-has-derivative* = *has-derivative-inner-right*
lemmas *inner-left-has-derivative* = *has-derivative-inner-left*
lemmas *mult-right-has-derivative* = *has-derivative-mult-right*
lemmas *mult-left-has-derivative* = *has-derivative-mult-left*

lemma *has-derivative-add-const*:
(*f has-derivative f'*) *net* $\Longrightarrow$ ($(\lambda x.\ f\ x + c)$ *has-derivative f'*) *net*
⟨*proof*⟩

40.2 Derivative with composed bilinear function.

lemma *has-derivative-bilinear-within*:
assumes (*f has-derivative f'*) (*at x within s*)
and (*g has-derivative g'*) (*at x within s*)
and *bounded-bilinear h*
shows $((\lambda x.\ h\ (f\ x)\ (g\ x))$ *has-derivative* $(\lambda d.\ h\ (f\ x)\ (g'\ d) + h\ (f'\ d)\ (g\ x)))$
(*at x within s*)
⟨*proof*⟩

lemma *has-derivative-bilinear-at*:
assumes (*f has-derivative f'*) (*at x*)
and (*g has-derivative g'*) (*at x*)
and *bounded-bilinear h*
shows $((\lambda x.\ h\ (f\ x)\ (g\ x))$ *has-derivative* $(\lambda d.\ h\ (f\ x)\ (g'\ d) + h\ (f'\ d)\ (g\ x)))$
(*at x*)
⟨*proof*⟩

These are the only cases we'll care about, probably.

lemma *has-derivative-within*: (*f has-derivative f'*) (*at x within s*) $\longleftrightarrow$

bounded-linear $f' \wedge ((\lambda y.\ (1\ /\ norm(y - x)) *_R (f\ y - (f\ x + f'\ (y - x)))) \longrightarrow 0)\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-derivative-at*: $(f\ has\text{-}derivative\ f')\ (at\ x) \longleftrightarrow$
bounded-linear $f' \wedge ((\lambda y.\ (1\ /\ (norm(y - x))) *_R (f\ y - (f\ x + f'\ (y - x)))) \longrightarrow 0)\ (at\ x)$
⟨*proof*⟩

More explicit epsilon-delta forms.

lemma *has-derivative-within'*:
$(f\ has\text{-}derivative\ f')(at\ x\ within\ s) \longleftrightarrow$
bounded-linear $f' \wedge$
$(\forall e{>}0.\ \exists d{>}0.\ \forall x'{\in}s.\ 0 < norm\ (x' - x) \wedge norm\ (x' - x) < d \longrightarrow$
$norm\ (f\ x' - f\ x - f'(x' - x))\ /\ norm\ (x' - x) < e)$
⟨*proof*⟩

lemma *has-derivative-at'*:
$(f\ has\text{-}derivative\ f')\ (at\ x) \longleftrightarrow$ *bounded-linear* $f' \wedge$
$(\forall e{>}0.\ \exists d{>}0.\ \forall x'.\ 0 < norm\ (x' - x) \wedge norm\ (x' - x) < d \longrightarrow$
$norm\ (f\ x' - f\ x - f'(x' - x))\ /\ norm\ (x' - x) < e)$
⟨*proof*⟩

lemma *has-derivative-at-within*:
$(f\ has\text{-}derivative\ f')\ (at\ x) \Longrightarrow (f\ has\text{-}derivative\ f')\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-derivative-within-open*:
$a \in s \Longrightarrow open\ s \Longrightarrow$
$(f\ has\text{-}derivative\ f')\ (at\ a\ within\ s) \longleftrightarrow (f\ has\text{-}derivative\ f')\ (at\ a)$
⟨*proof*⟩

lemma *has-derivative-right*:
fixes $f :: real \Rightarrow real$
and $y :: real$
shows $(f\ has\text{-}derivative\ (op *\ y))\ (at\ x\ within\ (\{x <..\} \cap I)) \longleftrightarrow$
$((\lambda t.\ (f\ x - f\ t)\ /\ (x - t)) \longrightarrow y)\ (at\ x\ within\ (\{x <..\} \cap I))$
⟨*proof*⟩

40.2.1 Caratheodory characterization

lemma *DERIV-within-iff*:
$(f\ has\text{-}field\text{-}derivative\ D)\ (at\ a\ within\ s) \longleftrightarrow ((\lambda z.\ (f\ z - f\ a)\ /\ (z - a)) \longrightarrow D)\ (at\ a\ within\ s)$
⟨*proof*⟩

lemma *DERIV-caratheodory-within*:
$(f\ has\text{-}field\text{-}derivative\ l)\ (at\ x\ within\ s) \longleftrightarrow$
$(\exists g.\ (\forall z.\ f\ z - f\ x = g\ z * (z - x)) \wedge continuous\ (at\ x\ within\ s)\ g \wedge g\ x = l)$

(**is** *?lhs = ?rhs*)
⟨*proof*⟩

40.2.2 Limit transformation for derivatives

lemma *has-derivative-transform-within*:
assumes (*f has-derivative f'*) (*at x within s*)
and *0 < d*
and *x ∈ s*
and ⋀*x'*. ⟦*x' ∈ s*; *dist x' x < d*⟧ ⟹ *f x' = g x'*
shows (*g has-derivative f'*) (*at x within s*)
⟨*proof*⟩

lemma *has-derivative-transform-within-open*:
assumes (*f has-derivative f'*) (*at x*)
and *open s*
and *x ∈ s*
and ⋀*x*. *x∈s* ⟹ *f x = g x*
shows (*g has-derivative f'*) (*at x*)
⟨*proof*⟩

40.3 Differentiability

definition
differentiable-on :: (*'a*::*real-normed-vector* ⇒ *'b*::*real-normed-vector*) ⇒ *'a set* ⇒ *bool*
(**infix** *differentiable'-on 50*)
where *f differentiable-on s* ⟷ (∀ *x*∈*s*. *f differentiable* (*at x within s*))

lemma *differentiableI*: (*f has-derivative f'*) *net* ⟹ *f differentiable net*
⟨*proof*⟩

lemma *differentiable-onD*: ⟦*f differentiable-on S*; *x ∈ S*⟧ ⟹ *f differentiable* (*at x within S*)
⟨*proof*⟩

lemma *differentiable-at-withinI*: *f differentiable* (*at x*) ⟹ *f differentiable* (*at x within s*)
⟨*proof*⟩

lemma *differentiable-at-imp-differentiable-on*:
(⋀*x*. *x ∈ s* ⟹ *f differentiable at x*) ⟹ *f differentiable-on s*
⟨*proof*⟩

corollary *differentiable-iff-scaleR*:
fixes *f* :: *real* ⇒ *'a*::*real-normed-vector*
shows *f differentiable F* ⟷ (∃ *d*. (*f has-derivative* (λ*x*. *x* ∗*R d*)) *F*)
⟨*proof*⟩

lemma *differentiable-within-open*:

assumes $a \in s$
and *open s*
shows *f differentiable* (*at a within s*) $\longleftrightarrow$ *f differentiable* (*at a*)
⟨*proof*⟩

lemma *differentiable-on-eq-differentiable-at*:
open s $\Longrightarrow$ *f differentiable-on s* $\longleftrightarrow$ $(\forall x{\in}s.$ *f differentiable at x*)
⟨*proof*⟩

lemma *differentiable-transform-within*:
assumes *f differentiable* (*at x within s*)
and $0 < d$
and $x \in s$
and $\bigwedge x'.\ [\![x'{\in}s;\ dist\ x'\ x < d]\!] \Longrightarrow f\ x' = g\ x'$
shows *g differentiable* (*at x within s*)
⟨*proof*⟩

40.4 Frechet derivative and Jacobian matrix

definition *frechet-derivative f net* = (*SOME f'*. (*f has-derivative f'*) *net*)

lemma *frechet-derivative-works*:
f differentiable net $\longleftrightarrow$ (*f has-derivative* (*frechet-derivative f net*)) *net*
⟨*proof*⟩

lemma *linear-frechet-derivative*: *f differentiable net* $\Longrightarrow$ *linear* (*frechet-derivative f net*)
⟨*proof*⟩

40.5 Differentiability implies continuity

lemma *differentiable-imp-continuous-within*:
f differentiable (*at x within s*) $\Longrightarrow$ *continuous* (*at x within s*) *f*
⟨*proof*⟩

lemma *differentiable-imp-continuous-on*:
f differentiable-on s $\Longrightarrow$ *continuous-on s f*
⟨*proof*⟩

lemma *differentiable-on-subset*:
f differentiable-on t $\Longrightarrow$ $s \subseteq t$ $\Longrightarrow$ *f differentiable-on s*
⟨*proof*⟩

lemma *differentiable-on-empty*: *f differentiable-on* {}
⟨*proof*⟩

Results about neighborhoods filter.

lemma *eventually-nhds-metric-le*:
eventually P (*nhds a*) = $(\exists\, d{>}0.\ \forall x.\ dist\ x\ a \leq d \longrightarrow P\ x)$

⟨*proof*⟩

lemma *le-nhds*: $F \leq nhds\ a \longleftrightarrow (\forall S.\ open\ S \wedge a \in S \longrightarrow eventually\ (\lambda x.\ x \in S)\ F)$
⟨*proof*⟩

lemma *le-nhds-metric*: $F \leq nhds\ a \longleftrightarrow (\forall e{>}0.\ eventually\ (\lambda x.\ dist\ x\ a < e)\ F)$
⟨*proof*⟩

lemma *le-nhds-metric-le*: $F \leq nhds\ a \longleftrightarrow (\forall e{>}0.\ eventually\ (\lambda x.\ dist\ x\ a \leq e)\ F)$
⟨*proof*⟩

Several results are easier using a "multiplied-out" variant. (I got this idea from Dieudonne's proof of the chain rule).

lemma *has-derivative-within-alt*:
$(f\ has\text{-}derivative\ f')\ (at\ x\ within\ s) \longleftrightarrow bounded\text{-}linear\ f' \wedge$
$(\forall e{>}0.\ \exists d{>}0.\ \forall y{\in}s.\ norm(y - x) < d \longrightarrow norm\ (f\ y - f\ x - f'\ (y - x)) \leq e * norm\ (y - x))$
⟨*proof*⟩

lemma *has-derivative-within-alt2*:
$(f\ has\text{-}derivative\ f')\ (at\ x\ within\ s) \longleftrightarrow bounded\text{-}linear\ f' \wedge$
$(\forall e{>}0.\ eventually\ (\lambda y.\ norm\ (f\ y - f\ x - f'\ (y - x)) \leq e * norm\ (y - x))\ (at\ x\ within\ s))$
⟨*proof*⟩

lemma *has-derivative-at-alt*:
$(f\ has\text{-}derivative\ f')\ (at\ x) \longleftrightarrow$
$bounded\text{-}linear\ f' \wedge$
$(\forall e{>}0.\ \exists d{>}0.\ \forall y.\ norm(y - x) < d \longrightarrow norm\ (f\ y - f\ x - f'(y - x)) \leq e * norm\ (y - x))$
⟨*proof*⟩

40.6 The chain rule

lemma *diff-chain-within*[*derivative-intros*]:
assumes $(f\ has\text{-}derivative\ f')\ (at\ x\ within\ s)$
and $(g\ has\text{-}derivative\ g')\ (at\ (f\ x)\ within\ (f\ `\ s))$
shows $((g \circ f)\ has\text{-}derivative\ (g' \circ f'))(at\ x\ within\ s)$
⟨*proof*⟩

lemma *diff-chain-at*[*derivative-intros*]:
$(f\ has\text{-}derivative\ f')\ (at\ x) \Longrightarrow$
$(g\ has\text{-}derivative\ g')\ (at\ (f\ x)) \Longrightarrow ((g \circ f)\ has\text{-}derivative\ (g' \circ f'))\ (at\ x)$
⟨*proof*⟩

40.7 Composition rules stated just for differentiability

lemma *differentiable-chain-at*:

f differentiable (*at x*) $\Longrightarrow$
g differentiable (*at* (*f x*)) $\Longrightarrow$ (*g* $\circ$ *f*) *differentiable* (*at x*)
⟨*proof*⟩

lemma *differentiable-chain-within*:
f differentiable (*at x within s*) $\Longrightarrow$
g differentiable (*at*(*f x*) *within* (*f* ' *s*)) $\Longrightarrow$ (*g* $\circ$ *f*) *differentiable* (*at x within s*)
⟨*proof*⟩

40.8 Uniqueness of derivative

The general result is a bit messy because we need approachability of the limit point from any direction. But OK for nontrivial intervals etc.

lemma *frechet-derivative-unique-within*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'b*::*real-normed-vector*
assumes (*f has-derivative f'*) (*at x within s*)
and (*f has-derivative f''*) (*at x within s*)
and $\forall\, i{\in}Basis.\ \forall\, e{>}0.\ \exists\, d.\ 0 < |d| \wedge |d| < e \wedge (x + d *_R i) \in s$
shows $f' = f''$
⟨*proof*⟩

lemma *frechet-derivative-unique-at*:
(*f has-derivative f'*) (*at x*) $\Longrightarrow$ (*f has-derivative f''*) (*at x*) $\Longrightarrow$ $f' = f''$
⟨*proof*⟩

lemma *frechet-derivative-unique-within-closed-interval*:
fixes *f*::*'a*::*euclidean-space* $\Rightarrow$ *'b*::*real-normed-vector*
assumes $\forall\, i{\in}Basis.\ a{\cdot}i < b{\cdot}i$
and $x \in cbox\ a\ b$
and (*f has-derivative f'*) (*at x within cbox a b*)
and (*f has-derivative f''*) (*at x within cbox a b*)
shows $f' = f''$
⟨*proof*⟩

lemma *frechet-derivative-unique-within-open-interval*:
fixes *f*::*'a*::*euclidean-space* $\Rightarrow$ *'b*::*real-normed-vector*
assumes $x \in box\ a\ b$
and (*f has-derivative f'*) (*at x within box a b*)
and (*f has-derivative f''*) (*at x within box a b*)
shows $f' = f''$
⟨*proof*⟩

lemma *frechet-derivative-at*:
(*f has-derivative f'*) (*at x*) $\Longrightarrow$ $f' =$ *frechet-derivative f* (*at x*)
⟨*proof*⟩

lemma *frechet-derivative-within-cbox*:
fixes *f* :: *'a*::*euclidean-space* $\Rightarrow$ *'b*::*real-normed-vector*
assumes $\forall\, i{\in}Basis.\ a{\cdot}i < b{\cdot}i$

and $x \in cbox\ a\ b$
and $(f\ has\text{-}derivative\ f')\ (at\ x\ within\ cbox\ a\ b)$
shows $frechet\text{-}derivative\ f\ (at\ x\ within\ cbox\ a\ b) = f'$
⟨*proof*⟩

40.9 The traditional Rolle theorem in one dimension

Derivatives of local minima and maxima are zero.

lemma *has-derivative-local-min*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow real$
assumes *deriv*: $(f\ has\text{-}derivative\ f')\ (at\ x)$
assumes *min*: $eventually\ (\lambda y.\ f\ x \leq f\ y)\ (at\ x)$
shows $f' = (\lambda h.\ 0)$
⟨*proof*⟩

lemma *has-derivative-local-max*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow real$
assumes $(f\ has\text{-}derivative\ f')\ (at\ x)$
assumes $eventually\ (\lambda y.\ f\ y \leq f\ x)\ (at\ x)$
shows $f' = (\lambda h.\ 0)$
⟨*proof*⟩

lemma *differential-zero-maxmin*:
fixes $f::'a::real\text{-}normed\text{-}vector \Rightarrow real$
assumes $x \in s$
and *open s*
and *deriv*: $(f\ has\text{-}derivative\ f')\ (at\ x)$
and *mono*: $(\forall\, y{\in}s.\ f\ y \leq f\ x) \vee (\forall\, y{\in}s.\ f\ x \leq f\ y)$
shows $f' = (\lambda v.\ 0)$
⟨*proof*⟩

lemma *differential-zero-maxmin-component*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::euclidean\text{-}space$
assumes *k*: $k \in Basis$
and *ball*: $0 < e\ (\forall\, y \in ball\ x\ e.\ (f\ y){\cdot}k \leq (f\ x){\cdot}k) \vee (\forall\, y{\in}ball\ x\ e.\ (f\ x){\cdot}k \leq (f\ y){\cdot}k)$
and *diff*: $f\ differentiable\ (at\ x)$
shows $(\sum j{\in}Basis.\ (frechet\text{-}derivative\ f\ (at\ x)\ j \cdot k) *_R j) = (0::'a)$ (**is** *?D k* = *0*)
⟨*proof*⟩

lemma *rolle*:
fixes $f :: real \Rightarrow real$
assumes $a < b$
and $f\ a = f\ b$
and *continuous-on* $\{a\ ..\ b\}\ f$
and $\forall\, x{\in}\{a <..< b\}.\ (f\ has\text{-}derivative\ f'\ x)\ (at\ x)$
shows $\exists\, x{\in}\{a <..< b\}.\ f'\ x = (\lambda v.\ 0)$
⟨*proof*⟩

40.10 One-dimensional mean value theorem

lemma *mvt*:
fixes $f :: real \Rightarrow real$
assumes $a < b$
and *continuous-on* $\{a..b\}$ f
assumes $\forall x{\in}\{a<..<b\}.$ $(f$ *has-derivative* $(f'\ x))$ $(at\ x)$
shows $\exists x{\in}\{a<..<b\}.\ f\ b - f\ a = (f'\ x)\ (b - a)$
⟨*proof*⟩

lemma *mvt-simple*:
fixes $f :: real \Rightarrow real$
assumes $a < b$
and $\forall x{\in}\{a..b\}.$ $(f$ *has-derivative* $f'\ x)$ $(at\ x$ *within* $\{a..b\})$
shows $\exists x{\in}\{a<..<b\}.\ f\ b - f\ a = f'\ x\ (b - a)$
⟨*proof*⟩

lemma *mvt-very-simple*:
fixes $f :: real \Rightarrow real$
assumes $a \leq b$
and $\forall x{\in}\{a\ ..\ b\}.$ $(f$ *has-derivative* $f'\ x)$ $(at\ x$ *within* $\{a\ ..\ b\})$
shows $\exists x{\in}\{a\ ..\ b\}.\ f\ b - f\ a = f'\ x\ (b - a)$
⟨*proof*⟩

A nice generalization (see Havin's proof of 5.19 from Rudin's book).

lemma *mvt-general*:
fixes $f :: real \Rightarrow 'a$::*real-inner*
assumes $a < b$
and *continuous-on* $\{a\ ..\ b\}$ f
and $\forall x{\in}\{a<..<b\}.$ $(f$ *has-derivative* $f'(x))$ $(at\ x)$
shows $\exists x{\in}\{a<..<b\}.$ $norm\ (f\ b - f\ a) \leq norm\ (f'\ x\ (b - a))$
⟨*proof*⟩

40.11 More general bound theorems

lemma *differentiable-bound-general*:
fixes $f :: real \Rightarrow 'a$::*real-normed-vector*
assumes $a < b$
and *f-cont*: *continuous-on* $\{a\ ..\ b\}$ f
and *phi-cont*: *continuous-on* $\{a\ ..\ b\}$ φ
and *f'*: $\bigwedge x.\ a < x \Longrightarrow x < b \Longrightarrow (f$ *has-vector-derivative* $f'\ x)$ $(at\ x)$
and *phi'*: $\bigwedge x.\ a < x \Longrightarrow x < b \Longrightarrow (\varphi$ *has-vector-derivative* $\varphi'\ x)$ $(at\ x)$
and *bnd*: $\bigwedge x.\ a < x \Longrightarrow x < b \Longrightarrow norm\ (f'\ x) \leq \varphi'\ x$
shows $norm\ (f\ b - f\ a) \leq \varphi\ b - \varphi\ a$
⟨*proof*⟩

lemma *differentiable-bound*:
fixes $f :: 'a$::*real-normed-vector* $\Rightarrow 'b$::*real-normed-vector*
assumes *convex* s
and $\forall x{\in}s.$ $(f$ *has-derivative* $f'\ x)$ $(at\ x$ *within* $s)$

and $\forall x{\in}s.\ onorm\ (f'\ x) \leq B$
and x: $x \in s$
and y: $y \in s$
shows $norm\ (f\ x - f\ y) \leq B * norm\ (x - y)$
⟨*proof*⟩

lemma
differentiable-bound-segment:
fixes f::$'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*
assumes $\bigwedge t.\ t \in \{0..1\} \Longrightarrow x0 + t *_R a \in G$
assumes f': $\bigwedge x.\ x \in G \Longrightarrow (f\ has\text{-}derivative\ f'\ x)\ (at\ x\ within\ G)$
assumes B: $\forall x{\in}\{0..1\}.\ onorm\ (f'\ (x0 + x *_R a)) \leq B$
shows $norm\ (f\ (x0 + a) - f\ x0) \leq norm\ a * B$
⟨*proof*⟩

lemma *differentiable-bound-linearization*:
fixes f::$'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*
assumes $\bigwedge t.\ t \in \{0..1\} \Longrightarrow a + t *_R (b - a) \in S$
assumes f'[*derivative-intros*]: $\bigwedge x.\ x \in S \Longrightarrow (f\ has\text{-}derivative\ f'\ x)\ (at\ x\ within\ S)$
assumes B: $\forall x{\in}S.\ onorm\ (f'\ x - f'\ x0) \leq B$
assumes $x0 \in S$
shows $norm\ (f\ b - f\ a - f'\ x0\ (b - a)) \leq norm\ (b - a) * B$
⟨*proof*⟩

In particular.

lemma *has-derivative-zero-constant*:
fixes f :: $'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*
assumes *convex s*
and $\bigwedge x.\ x \in s \Longrightarrow (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ s)$
shows $\exists c.\ \forall x{\in}s.\ f\ x = c$
⟨*proof*⟩

lemma *has-field-derivative-zero-constant*:
assumes *convex s* $\bigwedge x.\ x \in s \Longrightarrow (f\ has\text{-}field\text{-}derivative\ 0)\ (at\ x\ within\ s)$
shows $\exists c.\ \forall x{\in}s.\ f\ (x) = (c :: 'a :: real\text{-}normed\text{-}field)$
⟨*proof*⟩

lemma *has-derivative-zero-unique*:
fixes f :: $'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*
assumes *convex s*
and $\bigwedge x.\ x \in s \Longrightarrow (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ s)$
and $x \in s\ y \in s$
shows $f\ x = f\ y$
⟨*proof*⟩

lemma *has-derivative-zero-unique-connected*:
fixes f :: $'a$::*real-normed-vector* ⇒ $'b$::*real-normed-vector*
assumes *open s connected s*

assumes f: $\bigwedge x.\ x \in s \Longrightarrow (f\ \textit{has-derivative}\ (\lambda x.\ 0))\ (\textit{at}\ x)$
assumes $x \in s\ y \in s$
shows $f\ x = f\ y$
⟨*proof*⟩

40.12 Differentiability of inverse function (most basic form)

lemma *has-derivative-inverse-basic*:
fixes $f :: 'a::\textit{real-normed-vector} \Rightarrow 'b::\textit{real-normed-vector}$
assumes $(f\ \textit{has-derivative}\ f')\ (\textit{at}\ (g\ y))$
and *bounded-linear* g'
and $g' \circ f' = \textit{id}$
and *continuous* $(\textit{at}\ y)\ g$
and *open* t
and $y \in t$
and $\forall z{\in}t.\ f\ (g\ z) = z$
shows $(g\ \textit{has-derivative}\ g')\ (\textit{at}\ y)$
⟨*proof*⟩

Simply rewrite that based on the domain point x.

lemma *has-derivative-inverse-basic-x*:
fixes $f :: 'a::\textit{real-normed-vector} \Rightarrow 'b::\textit{real-normed-vector}$
assumes $(f\ \textit{has-derivative}\ f')\ (\textit{at}\ x)$
and *bounded-linear* g'
and $g' \circ f' = \textit{id}$
and *continuous* $(\textit{at}\ (f\ x))\ g$
and $g\ (f\ x) = x$
and *open* t
and $f\ x \in t$
and $\forall y{\in}t.\ f\ (g\ y) = y$
shows $(g\ \textit{has-derivative}\ g')\ (\textit{at}\ (f\ x))$
⟨*proof*⟩

This is the version in Dieudonne', assuming continuity of f and g.

lemma *has-derivative-inverse-dieudonne*:
fixes $f :: 'a::\textit{real-normed-vector} \Rightarrow 'b::\textit{real-normed-vector}$
assumes *open* s
and *open* $(f\ `\ s)$
and *continuous-on* $s\ f$
and *continuous-on* $(f\ `\ s)\ g$
and $\forall x{\in}s.\ g\ (f\ x) = x$
and $x \in s$
and $(f\ \textit{has-derivative}\ f')\ (\textit{at}\ x)$
and *bounded-linear* g'
and $g' \circ f' = \textit{id}$
shows $(g\ \textit{has-derivative}\ g')\ (\textit{at}\ (f\ x))$
⟨*proof*⟩

Here's the simplest way of not assuming much about g.

lemma *has-derivative-inverse*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes *compact s*
and $x \in s$
and $f\,x \in interior\ (f\ `\ s)$
and *continuous-on s f*
and $\forall\, y{\in}s.\ g\ (f\ y) = y$
and $(f\ has\text{-}derivative\ f')\ (at\ x)$
and *bounded-linear* g'
and $g' \circ f' = id$
shows $(g\ has\text{-}derivative\ g')\ (at\ (f\ x))$
⟨*proof*⟩

40.13 Proving surjectivity via Brouwer fixpoint theorem

lemma *brouwer-surjective*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'n$
assumes *compact t*
and *convex t*
and $t \neq \{\}$
and *continuous-on t f*
and $\forall\, x{\in}s.\ \forall\, y{\in}t.\ x + (y - f\ y) \in t$
and $x \in s$
shows $\exists\, y{\in}t.\ f\ y = x$
⟨*proof*⟩

lemma *brouwer-surjective-cball*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'n$
assumes $e > 0$
and *continuous-on* $(cball\ a\ e)\ f$
and $\forall\, x{\in}s.\ \forall\, y{\in}cball\ a\ e.\ x + (y - f\ y) \in cball\ a\ e$
and $x \in s$
shows $\exists\, y{\in}cball\ a\ e.\ f\ y = x$
⟨*proof*⟩

See Sussmann: "Multidifferential calculus", Theorem 2.1.1

lemma *sussmann-open-mapping*:
fixes $f :: 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::euclidean\text{-}space$
assumes *open s*
and *continuous-on s f*
and $x \in s$
and $(f\ has\text{-}derivative\ f')\ (at\ x)$
and *bounded-linear* $g'\ f' \circ g' = id$
and $t \subseteq s$
and $x \in interior\ t$
shows $f\,x \in interior\ (f\ `\ t)$
⟨*proof*⟩

Hence the following eccentric variant of the inverse function theorem. This has no continuity assumptions, but we do need the inverse function. We

could put $f' \circ g = I$ but this happens to fit with the minimal linear algebra theory I've set up so far.

lemma *right-inverse-linear*:
  fixes f :: *'a*::*euclidean-space* ⇒ *'a*
  assumes *lf*: *linear f*
    and *gf*: $f \circ g = id$
  shows *linear g*
⟨*proof*⟩

lemma *has-derivative-inverse-strong*:
  fixes f :: *'n*::*euclidean-space* ⇒ *'n*
  assumes *open s*
    and $x \in s$
    and *continuous-on s f*
    and $\forall x{\in}s.\ g\ (f\ x) = x$
    and (*f has-derivative f'*) (*at x*)
    and $f' \circ g' = id$
  shows (*g has-derivative g'*) (*at* (*f x*))
⟨*proof*⟩

A rewrite based on the other domain.

lemma *has-derivative-inverse-strong-x*:
  fixes f :: *'a*::*euclidean-space* ⇒ *'a*
  assumes *open s*
    and *g y* ∈ *s*
    and *continuous-on s f*
    and $\forall x{\in}s.\ g\ (f\ x) = x$
    and (*f has-derivative f'*) (*at* (*g y*))
    and $f' \circ g' = id$
    and *f* (*g y*) = *y*
  shows (*g has-derivative g'*) (*at y*)
  ⟨*proof*⟩

On a region.

lemma *has-derivative-inverse-on*:
  fixes f :: *'n*::*euclidean-space* ⇒ *'n*
  assumes *open s*
    and $\forall x{\in}s.$ (*f has-derivative f'*(*x*)) (*at x*)
    and $\forall x{\in}s.\ g\ (f\ x) = x$
    and $f'\ x \circ g'\ x = id$
    and $x \in s$
  shows (*g has-derivative g'*(*x*)) (*at* (*f x*))
  ⟨*proof*⟩

Invertible derivative continous at a point implies local injectivity. It's only for this we need continuity of the derivative, except of course if we want the fact that the inverse derivative is also continuous. So if we know for some other reason that the inverse function exists, it's OK.

proposition *has-derivative-locally-injective*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes $a \in s$
and *open s*
and *bounded-linear* g'
and $g' \circ f' a = id$
and $\bigwedge x.\ x \in s \Longrightarrow (f \text{ has-derivative } f' x)\ (at\ x)$
and $\bigwedge e.\ e > 0 \Longrightarrow \exists d{>}0.\ \forall x.\ dist\ a\ x < d \longrightarrow onorm\ (\lambda v.\ f' x\ v - f' a\ v) < e$
obtains r **where** $r > 0$ *ball a r* $\subseteq s$ *inj-on f* (*ball a r*)
⟨*proof*⟩

40.14 Uniformly convergent sequence of derivatives

lemma *has-derivative-sequence-lipschitz-lemma*:
fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes *convex s*
and $\forall n.\ \forall x{\in}s.\ ((f\ n)\ has\text{-}derivative\ (f'\ n\ x))\ (at\ x\ within\ s)$
and $\forall n{\geq}N.\ \forall x{\in}s.\ \forall h.\ norm\ (f'\ n\ x\ h - g'\ x\ h) \leq e * norm\ h$
and $0 \leq e$
shows $\forall m{\geq}N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall y{\in}s.\ norm\ ((f\ m\ x - f\ n\ x) - (f\ m\ y - f\ n\ y)) \leq 2 * e * norm\ (x - y)$
⟨*proof*⟩

lemma *has-derivative-sequence-lipschitz*:
fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes *convex s*
and $\forall n.\ \forall x{\in}s.\ ((f\ n)\ has\text{-}derivative\ (f'\ n\ x))\ (at\ x\ within\ s)$
and $\forall e{>}0.\ \exists N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall h.\ norm\ (f'\ n\ x\ h - g'\ x\ h) \leq e * norm\ h$
shows $\forall e{>}0.\ \exists N.\ \forall m{\geq}N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall y{\in}s.$
$norm\ ((f\ m\ x - f\ n\ x) - (f\ m\ y - f\ n\ y)) \leq e * norm\ (x - y)$
⟨*proof*⟩

lemma *has-derivative-sequence*:
fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::banach$
assumes *convex s*
and $\forall n.\ \forall x{\in}s.\ ((f\ n)\ has\text{-}derivative\ (f'\ n\ x))\ (at\ x\ within\ s)$
and $\forall e{>}0.\ \exists N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall h.\ norm\ (f'\ n\ x\ h - g'\ x\ h) \leq e * norm\ h$
and $x0 \in s$
and $((\lambda n.\ f\ n\ x0) \longrightarrow l)\ sequentially$
shows $\exists g.\ \forall x{\in}s.\ ((\lambda n.\ f\ n\ x) \longrightarrow g\ x)\ sequentially \wedge (g\ has\text{-}derivative\ g'(x))\ (at\ x\ within\ s)$
⟨*proof*⟩

Can choose to line up antiderivatives if we want.

lemma *has-antiderivative-sequence*:
fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector \Rightarrow 'b::banach$
assumes *convex s*
and $\forall n.\ \forall x{\in}s.\ ((f\ n)\ has\text{-}derivative\ (f'\ n\ x))\ (at\ x\ within\ s)$

and $\forall e{>}0.\ \exists N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall h.\ norm\ (f'\ n\ x\ h - g'\ x\ h) \leq e * norm\ h$
shows $\exists g.\ \forall x{\in}s.\ (g\ has\text{-}derivative\ g'\ x)\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-antiderivative-limit*:
fixes g' :: $'a$::*real-normed-vector* ⇒ $'a$ ⇒ $'b$::*banach*
assumes *convex s*
and $\forall e{>}0.\ \exists f\ f'.\ \forall x{\in}s.$
$(f\ has\text{-}derivative\ (f'\ x))\ (at\ x\ within\ s) \wedge (\forall h.\ norm\ (f'\ x\ h - g'\ x\ h) \leq e * norm\ h)$
shows $\exists g.\ \forall x{\in}s.\ (g\ has\text{-}derivative\ g'\ x)\ (at\ x\ within\ s)$
⟨*proof*⟩

40.15 Differentiation of a series

lemma *has-derivative-series*:
fixes f :: *nat* ⇒ $'a$::*real-normed-vector* ⇒ $'b$::*banach*
assumes *convex s*
and $\bigwedge n\ x.\ x \in s \Longrightarrow ((f\ n)\ has\text{-}derivative\ (f'\ n\ x))\ (at\ x\ within\ s)$
and $\forall e{>}0.\ \exists N.\ \forall n{\geq}N.\ \forall x{\in}s.\ \forall h.\ norm\ (setsum\ (\lambda i.\ f'\ i\ x\ h)\ \{..{<}n\} - g'\ x\ h) \leq e * norm\ h$
and $x \in s$
and $(\lambda n.\ f\ n\ x)\ sums\ l$
shows $\exists g.\ \forall x{\in}s.\ (\lambda n.\ f\ n\ x)\ sums\ (g\ x) \wedge (g\ has\text{-}derivative\ g'\ x)\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-field-derivative-series*:
fixes f :: *nat* ⇒ ($'a$ :: {*real-normed-field*,*banach*}) ⇒ $'a$
assumes *convex s*
assumes $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ has\text{-}field\text{-}derivative\ f'\ n\ x)\ (at\ x\ within\ s)$
assumes *uniform-limit s* $(\lambda n\ x.\ \sum i{<}n.\ f'\ i\ x)\ g'$ *sequentially*
assumes *x0* $\in$ *s summable* $(\lambda n.\ f\ n\ x0)$
shows $\exists g.\ \forall x{\in}s.\ (\lambda n.\ f\ n\ x)\ sums\ g\ x \wedge (g\ has\text{-}field\text{-}derivative\ g'\ x)\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-field-derivative-series'*:
fixes f :: *nat* ⇒ ($'a$ :: {*real-normed-field*,*banach*}) ⇒ $'a$
assumes *convex s*
assumes $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ has\text{-}field\text{-}derivative\ f'\ n\ x)\ (at\ x\ within\ s)$
assumes *uniformly-convergent-on s* $(\lambda n\ x.\ \sum i{<}n.\ f'\ i\ x)$
assumes *x0* $\in$ *s summable* $(\lambda n.\ f\ n\ x0)\ x \in interior\ s$
shows *summable* $(\lambda n.\ f\ n\ x)\ ((\lambda x.\ \sum n.\ f\ n\ x)\ has\text{-}field\text{-}derivative\ (\sum n.\ f'\ n\ x))\ (at\ x)$
⟨*proof*⟩

lemma *differentiable-series*:
fixes f :: *nat* ⇒ ($'a$ :: {*real-normed-field*,*banach*}) ⇒ $'a$

assumes *convex s open s*
assumes $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ \textit{has-field-derivative}\ f'\ n\ x)\ (\textit{at}\ x)$
assumes *uniformly-convergent-on s* $(\lambda n\ x.\ \sum i{<}n.\ f'\ i\ x)$
assumes *x0* $\in$ *s summable* $(\lambda n.\ f\ n\ x0)$ **and** *x*: $x \in s$
shows *summable* $(\lambda n.\ f\ n\ x)$ **and** $(\lambda x.\ \sum n.\ f\ n\ x)$ *differentiable* $(\textit{at}\ x)$
⟨*proof*⟩

lemma *differentiable-series'*:
fixes $f :: \textit{nat} \Rightarrow ('a :: \{\textit{real-normed-field}, \textit{banach}\}) \Rightarrow 'a$
assumes *convex s open s*
assumes $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ \textit{has-field-derivative}\ f'\ n\ x)\ (\textit{at}\ x)$
assumes *uniformly-convergent-on s* $(\lambda n\ x.\ \sum i{<}n.\ f'\ i\ x)$
assumes *x0* $\in$ *s summable* $(\lambda n.\ f\ n\ x0)$
shows $(\lambda x.\ \sum n.\ f\ n\ x)$ *differentiable* $(\textit{at}\ x0)$
⟨*proof*⟩

Considering derivative $\textit{real} \Rightarrow 'b$ as a vector.

definition *vector-derivative f net* = $(\textit{SOME}\ f'.\ (f\ \textit{has-vector-derivative}\ f')\ \textit{net})$

lemma *vector-derivative-unique-within*:
assumes *not-bot*: *at x within s* $\neq$ *bot*
and *f'*: (*f has-vector-derivative f'*) (*at x within s*)
and *f''*: (*f has-vector-derivative f''*) (*at x within s*)
shows $f' = f''$
⟨*proof*⟩

lemma *vector-derivative-unique-at*:
(*f has-vector-derivative f'*) (*at x*) $\Longrightarrow$ (*f has-vector-derivative f''*) (*at x*) $\Longrightarrow$ $f' = f''$
⟨*proof*⟩

lemma *differentiableI-vector*: (*f has-vector-derivative y*) *F* $\Longrightarrow$ *f differentiable F*
⟨*proof*⟩

lemma *vector-derivative-works*:
f differentiable net $\longleftrightarrow$ (*f has-vector-derivative* (*vector-derivative f net*)) *net*
(**is** *?l* = *?r*)
⟨*proof*⟩

lemma *vector-derivative-within*:
assumes *not-bot*: *at x within s* $\neq$ *bot* **and** *y*: (*f has-vector-derivative y*) (*at x within s*)
shows *vector-derivative f* (*at x within s*) = *y*
⟨*proof*⟩

lemma *frechet-derivative-eq-vector-derivative*:
assumes *f differentiable* (*at x*)
shows (*frechet-derivative f* (*at x*)) = $(\lambda r.\ r *_R$ *vector-derivative f* (*at x*))
⟨*proof*⟩

lemma *has-real-derivative*:
fixes $f :: real \Rightarrow real$
assumes $(f\ has\text{-}derivative\ f')\ F$
obtains c **where** $(f\ has\text{-}real\text{-}derivative\ c)\ F$
⟨*proof*⟩

lemma *has-real-derivative-iff*:
fixes $f :: real \Rightarrow real$
shows $(\exists\, c.\ (f\ has\text{-}real\text{-}derivative\ c)\ F) = (\exists\, D.\ (f\ has\text{-}derivative\ D)\ F)$
⟨*proof*⟩

definition *deriv* :: $('a \Rightarrow 'a{::}real\text{-}normed\text{-}field) \Rightarrow 'a \Rightarrow 'a$ **where**
$deriv\ f\ x \equiv SOME\ D.\ DERIV\ f\ x :> D$

lemma *DERIV-imp-deriv*: $DERIV\ f\ x :> f' \Longrightarrow deriv\ f\ x = f'$
⟨*proof*⟩

lemma *DERIV-deriv-iff-has-field-derivative*:
$DERIV\ f\ x :> deriv\ f\ x \longleftrightarrow (\exists f'.\ (f\ has\text{-}field\text{-}derivative\ f')\ (at\ x))$
⟨*proof*⟩

lemma *DERIV-deriv-iff-real-differentiable*:
fixes $x :: real$
shows $DERIV\ f\ x :> deriv\ f\ x \longleftrightarrow f\ differentiable\ at\ x$
⟨*proof*⟩

lemma *real-derivative-chain*:
fixes $x :: real$
shows $f\ differentiable\ at\ x \Longrightarrow g\ differentiable\ at\ (f\ x)$
$\Longrightarrow deriv\ (g\ o\ f)\ x = deriv\ g\ (f\ x) * deriv\ f\ x$
⟨*proof*⟩
lemma *field-derivative-eq-vector-derivative*:
$(deriv\ f\ x) = vector\text{-}derivative\ f\ (at\ x)$
⟨*proof*⟩

lemma *islimpt-closure-open*:
fixes $s :: 'a{::}perfect\text{-}space\ set$
assumes *open s* **and** t: $t = closure\ s\ x \in t$
shows $x\ islimpt\ t$
⟨*proof*⟩

lemma *vector-derivative-unique-within-closed-interval*:
assumes *ab*: $a < b\ x \in cbox\ a\ b$
assumes D: $(f\ has\text{-}vector\text{-}derivative\ f')\ (at\ x\ within\ cbox\ a\ b)\ (f\ has\text{-}vector\text{-}derivative\ f'')\ (at\ x\ within\ cbox\ a\ b)$
shows $f' = f''$
⟨*proof*⟩

lemma *vector-derivative-at*:
(*f has-vector-derivative f'*) (*at x*) ⟹ *vector-derivative f* (*at x*) = *f'*
⟨*proof*⟩

lemma *has-vector-derivative-id-at* [*simp*]: *vector-derivative* (λ*x*. *x*) (*at a*) = *1*
⟨*proof*⟩

lemma *vector-derivative-minus-at* [*simp*]:
f differentiable at a
⟹ *vector-derivative* (λ*x*. − *f x*) (*at a*) = − *vector-derivative f* (*at a*)
⟨*proof*⟩

lemma *vector-derivative-add-at* [*simp*]:
⟦*f differentiable at a*; *g differentiable at a*⟧
⟹ *vector-derivative* (λ*x*. *f x* + *g x*) (*at a*) = *vector-derivative f* (*at a*) + *vector-derivative g* (*at a*)
⟨*proof*⟩

lemma *vector-derivative-diff-at* [*simp*]:
⟦*f differentiable at a*; *g differentiable at a*⟧
⟹ *vector-derivative* (λ*x*. *f x* − *g x*) (*at a*) = *vector-derivative f* (*at a*) − *vector-derivative g* (*at a*)
⟨*proof*⟩

lemma *vector-derivative-mult-at* [*simp*]:
fixes *f g* :: *real* ⇒ *'a* :: *real-normed-algebra*
shows ⟦*f differentiable at a*; *g differentiable at a*⟧
⟹ *vector-derivative* (λ*x*. *f x* ∗ *g x*) (*at a*) = *f a* ∗ *vector-derivative g* (*at a*) + *vector-derivative f* (*at a*) ∗ *g a*
⟨*proof*⟩

lemma *vector-derivative-scaleR-at* [*simp*]:
⟦*f differentiable at a*; *g differentiable at a*⟧
⟹ *vector-derivative* (λ*x*. *f x* ∗$_R$ *g x*) (*at a*) = *f a* ∗$_R$ *vector-derivative g* (*at a*) + *vector-derivative f* (*at a*) ∗$_R$ *g a*
⟨*proof*⟩

lemma *vector-derivative-within-closed-interval*:
assumes *ab*: *a* < *b x* ∈ *cbox a b*
assumes *f*: (*f has-vector-derivative f'*) (*at x within cbox a b*)
shows *vector-derivative f* (*at x within cbox a b*) = *f'*
⟨*proof*⟩

lemma *has-vector-derivative-within-subset*:
(*f has-vector-derivative f'*) (*at x within s*) ⟹ *t* ⊆ *s* ⟹ (*f has-vector-derivative f'*) (*at x within t*)
⟨*proof*⟩

lemma *has-vector-derivative-at-within*:

(*f has-vector-derivative f'*) (*at x*) ⟹ (*f has-vector-derivative f'*) (*at x within s*)
⟨*proof*⟩

lemma *has-vector-derivative-weaken*:
fixes *x D* **and** *f g s t*
assumes *f*: (*f has-vector-derivative D*) (*at x within t*)
and *x* ∈ *s s* ⊆ *t*
and ⋀*x*. *x* ∈ *s* ⟹ *f x* = *g x*
shows (*g has-vector-derivative D*) (*at x within s*)
⟨*proof*⟩

lemma *has-vector-derivative-transform-within*:
assumes (*f has-vector-derivative f'*) (*at x within s*)
and *0* < *d*
and *x* ∈ *s*
and ⋀*x'*. ⟦*x'*∈*s*; *dist x' x* < *d*⟧ ⟹ *f x'* = *g x'*
shows (*g has-vector-derivative f'*) (*at x within s*)
⟨*proof*⟩

lemma *has-vector-derivative-transform-within-open*:
assumes (*f has-vector-derivative f'*) (*at x*)
and *open s*
and *x* ∈ *s*
and ⋀*y*. *y*∈*s* ⟹ *f y* = *g y*
shows (*g has-vector-derivative f'*) (*at x*)
⟨*proof*⟩

lemma *vector-diff-chain-at*:
assumes (*f has-vector-derivative f'*) (*at x*)
and (*g has-vector-derivative g'*) (*at* (*f x*))
shows ((*g* ∘ *f*) *has-vector-derivative* (*f'* *$_R$* *g'*)) (*at x*)
⟨*proof*⟩

lemma *vector-diff-chain-within*:
assumes (*f has-vector-derivative f'*) (*at x within s*)
and (*g has-vector-derivative g'*) (*at* (*f x*) *within f* ‘ *s*)
shows ((*g* ∘ *f*) *has-vector-derivative* (*f'* *$_R$* *g'*)) (*at x within s*)
⟨*proof*⟩

lemma *vector-derivative-const-at* [*simp*]: *vector-derivative* (λ*x*. *c*) (*at a*) = *0*
⟨*proof*⟩

lemma *vector-derivative-at-within-ivl*:
(*f has-vector-derivative f'*) (*at x*) ⟹
a ≤ *x* ⟹ *x* ≤ *b* ⟹ *a*<*b* ⟹ *vector-derivative f* (*at x within* {*a*..*b*}) = *f'*
⟨*proof*⟩

lemma *vector-derivative-chain-at*:
assumes *f differentiable at x* (*g differentiable at* (*f x*))

shows *vector-derivative* $(g \circ f)$ $(at\ x) =$
$vector\text{-}derivative\ f\ (at\ x) *_R vector\text{-}derivative\ g\ (at\ (f\ x))$
⟨*proof*⟩

lemma *field-vector-diff-chain-at*:
assumes *Df*: $(f\ has\text{-}vector\text{-}derivative\ f')\ (at\ x)$
and *Dg*: $(g\ has\text{-}field\text{-}derivative\ g')\ (at\ (f\ x))$
shows $((g \circ f)\ has\text{-}vector\text{-}derivative\ (f' * g'))\ (at\ x)$
⟨*proof*⟩

lemma *vector-derivative-chain-at-general*:
assumes *f differentiable at x* $(\exists g'.\ (g\ has\text{-}field\text{-}derivative\ g')\ (at\ (f\ x)))$
shows *vector-derivative* $(g \circ f)$ $(at\ x) =$
$vector\text{-}derivative\ f\ (at\ x) * deriv\ g\ (f\ x)$
⟨*proof*⟩

lemma *exp-scaleR-has-vector-derivative-right*:
$((\lambda t.\ exp\ (t *_R A))\ has\text{-}vector\text{-}derivative\ exp\ (t *_R A) * A)\ (at\ t\ within\ T)$
⟨*proof*⟩

lemma *exp-times-scaleR-commute*: $exp\ (t *_R A) * A = A * exp\ (t *_R A)$
⟨*proof*⟩

lemma *exp-scaleR-has-vector-derivative-left*: $((\lambda t.\ exp\ (t *_R A))\ has\text{-}vector\text{-}derivative\ A * exp\ (t *_R A))\ (at\ t)$
⟨*proof*⟩

40.16 Relation between convexity and derivative

lemma *convex-on-imp-above-tangent*:
assumes *convex*: *convex-on A f* **and** *connected*: *connected A*
assumes *c*: $c \in interior\ A$ **and** $x : x \in A$
assumes *deriv*: $(f\ has\text{-}field\text{-}derivative\ f')\ (at\ c\ within\ A)$
shows $f\ x - f\ c \geq f' * (x - c)$
⟨*proof*⟩

40.17 Partial derivatives

lemma *eventually-at-Pair-within-TimesI1*:
fixes $x::'a::metric\text{-}space$
assumes $\forall_F\ x'\ in\ at\ x\ within\ X.\ P\ x'$
assumes $P\ x$
shows $\forall_F\ (x', y')\ in\ at\ (x, y)\ within\ X \times Y.\ P\ x'$
⟨*proof*⟩

lemma *eventually-at-Pair-within-TimesI2*:
fixes $x::'a::metric\text{-}space$
assumes $\forall_F\ y'\ in\ at\ y\ within\ Y.\ P\ y'$
assumes $P\ y$
shows $\forall_F\ (x', y')\ in\ at\ (x, y)\ within\ X \times Y.\ P\ y'$

⟨*proof*⟩

lemma *has-derivative-partialsI*:
assumes *fx*: $\bigwedge x\ y.\ x \in X \Longrightarrow y \in Y \Longrightarrow ((\lambda x.\ f\ x\ y)$ *has-derivative blinfun-apply* $(fx\ x\ y))$ $(at\ x\ within\ X)$
assumes *fy*: $\bigwedge x\ y.\ x \in X \Longrightarrow y \in Y \Longrightarrow ((\lambda y.\ f\ x\ y)$ *has-derivative blinfun-apply* $(fy\ x\ y))$ $(at\ y\ within\ Y)$
assumes *fx-cont*: *continuous-on* $(X \times Y)$ $(\lambda(x,\ y).\ fx\ x\ y)$
assumes *fy-cont*: *continuous-on* $(X \times Y)$ $(\lambda(x,\ y).\ fy\ x\ y)$
assumes $x \in X\ y \in Y$
assumes *convex X convex Y*
shows $((\lambda(x,\ y).\ f\ x\ y)$ *has-derivative* $(\lambda(tx,\ ty).\ fx\ x\ y\ tx + fy\ x\ y\ ty))$ $(at\ (x,\ y)\ within\ X \times Y)$
⟨*proof*⟩

end

41 Kurzweil-Henstock Gauge Integration in many dimensions.

theory *Integration*
imports
Derivative
Uniform-Limit
~~/src/HOL/Library/Indicator-Function
begin

lemmas *scaleR-simps* = *scaleR-zero-left scaleR-minus-left scaleR-left-diff-distrib scaleR-zero-right scaleR-minus-right scaleR-right-diff-distrib scaleR-eq-0-iff scaleR-cancel-left scaleR-cancel-right scaleR-add-right scaleR-add-left real-vector-class.scaleR-one*

41.1 Sundries

lemma *conjunctD2*: **assumes** $a \wedge b$ **shows** $a\ b$ ⟨*proof*⟩
lemma *conjunctD3*: **assumes** $a \wedge b \wedge c$ **shows** $a\ b\ c$ ⟨*proof*⟩
lemma *conjunctD4*: **assumes** $a \wedge b \wedge c \wedge d$ **shows** $a\ b\ c\ d$ ⟨*proof*⟩

declare *norm-triangle-ineq4*[*intro*]

lemma *simple-image*: $\{f\ x\ |x\ .\ x \in s\} = f\ `\ s$
⟨*proof*⟩

lemma *linear-simps*:
assumes *bounded-linear f*
shows
$f\ (a + b) = f\ a + f\ b$
$f\ (a - b) = f\ a - f\ b$

$f\ 0 = 0$
$f\ (-\ a) = -\ f\ a$
$f\ (s\ *_R\ v) = s\ *_R\ (f\ v)$
⟨*proof*⟩

lemma *bounded-linearI*:
assumes $\bigwedge x\ y.\ f\ (x + y) = f\ x + f\ y$
and $\bigwedge r\ x.\ f\ (r\ *_R\ x) = r\ *_R\ f\ x$
and $\bigwedge x.\ norm\ (f\ x) \leq norm\ x * K$
shows *bounded-linear f*
⟨*proof*⟩

lemma *bounded-linear-component* [*intro*]: *bounded-linear* (λx::'*a*::*euclidean-space*. $x \cdot k$)
⟨*proof*⟩

lemma *transitive-stepwise-lt-eq*:
assumes ($\bigwedge x\ y\ z$::*nat*. $R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z$)
shows (($\forall m.\ \forall n{>}m.\ R\ m\ n$) $\longleftrightarrow$ ($\forall n.\ R\ n\ (Suc\ n)$))
(**is** *?l* = *?r*)
⟨*proof*⟩

lemma *transitive-stepwise-gt*:
assumes $\bigwedge x\ y\ z.\ R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z\ \bigwedge n.\ R\ n\ (Suc\ n)$
shows $\forall n{>}m.\ R\ m\ n$
⟨*proof*⟩

lemma *transitive-stepwise-le-eq*:
assumes $\bigwedge x.\ R\ x\ x\ \bigwedge x\ y\ z.\ R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z$
shows ($\forall m.\ \forall n{\geq}m.\ R\ m\ n$) $\longleftrightarrow$ ($\forall n.\ R\ n\ (Suc\ n)$)
(**is** *?l* = *?r*)
⟨*proof*⟩

lemma *transitive-stepwise-le*:
assumes $\bigwedge x.\ R\ x\ x\ \bigwedge x\ y\ z.\ R\ x\ y \Longrightarrow R\ y\ z \Longrightarrow R\ x\ z$
and $\bigwedge n.\ R\ n\ (Suc\ n)$
shows $\forall n{\geq}m.\ R\ m\ n$
⟨*proof*⟩

41.2 Some useful lemmas about intervals.

lemma *empty-as-interval*: {} = *cbox One* (*0*::'*a*::*euclidean-space*)
⟨*proof*⟩

lemma *interior-subset-union-intervals*:
assumes $i = cbox\ a\ b$
and $j = cbox\ c\ d$
and $interior\ j \neq \{\}$
and $i \subseteq j \cup s$

and *interior i* ∩ *interior j* = {}
shows *interior i* ⊆ *interior s*
⟨*proof*⟩

lemma *inter-interior-unions-intervals*:
fixes *f*::(*'a*::*euclidean-space*) *set set*
assumes *finite f*
and *open s*
and ∀ *t*∈*f*. ∃ *a b*. *t* = *cbox a b*
and ∀ *t*∈*f*. *s* ∩ (*interior t*) = {}
shows *s* ∩ *interior* (⋃*f*) = {}
⟨*proof*⟩

41.3 Bounds on intervals where they exist.

definition *interval-upperbound* :: (*'a*::*euclidean-space*) *set* ⇒ *'a*
where *interval-upperbound s* = (∑ *i*∈*Basis*. (*SUP x*:*s*. *x*•*i*) *∗*$_R$ *i*)

definition *interval-lowerbound* :: (*'a*::*euclidean-space*) *set* ⇒ *'a*
where *interval-lowerbound s* = (∑ *i*∈*Basis*. (*INF x*:*s*. *x*•*i*) *∗*$_R$ *i*)

lemma *interval-upperbound*[*simp*]:
∀ *i*∈*Basis*. *a*•*i* ≤ *b*•*i* ⟹
interval-upperbound (*cbox a b*) = (*b*::*'a*::*euclidean-space*)
⟨*proof*⟩

lemma *interval-lowerbound*[*simp*]:
∀ *i*∈*Basis*. *a*•*i* ≤ *b*•*i* ⟹
interval-lowerbound (*cbox a b*) = (*a*::*'a*::*euclidean-space*)
⟨*proof*⟩

lemmas *interval-bounds* = *interval-upperbound interval-lowerbound*

lemma
fixes *X*::*real set*
shows *interval-upperbound-real*[*simp*]: *interval-upperbound X* = *Sup X*
and *interval-lowerbound-real*[*simp*]: *interval-lowerbound X* = *Inf X*
⟨*proof*⟩

lemma *interval-bounds'*[*simp*]:
assumes *cbox a b* ≠ {}
shows *interval-upperbound* (*cbox a b*) = *b*
and *interval-lowerbound* (*cbox a b*) = *a*
⟨*proof*⟩

lemma *interval-upperbound-Times*:
assumes *A* ≠ {} **and** *B* ≠ {}
shows *interval-upperbound* (*A* × *B*) = (*interval-upperbound A*, *interval-upperbound*

B)
⟨*proof*⟩

lemma *interval-lowerbound-Times*:
assumes $A \neq \{\}$ **and** $B \neq \{\}$
shows *interval-lowerbound* $(A \times B) =$ (*interval-lowerbound A*, *interval-lowerbound B*)
⟨*proof*⟩

41.4 Content (length, area, volume...) of an interval.

definition *content* $(s::('a::euclidean\text{-}space)\ set) =$
$(if\ s = \{\}\ then\ 0\ else\ (\prod i{\in}Basis.\ (interval\text{-}upperbound\ s){\cdot}i - (interval\text{-}lowerbound\ s){\cdot}i))$

lemma *interval-not-empty*: $\forall\, i{\in}Basis.\ a{\cdot}i \leq b{\cdot}i \Longrightarrow cbox\ a\ b \neq \{\}$
⟨*proof*⟩

lemma *content-cbox*:
fixes $a :: 'a::euclidean\text{-}space$
assumes $\forall\, i{\in}Basis.\ a{\cdot}i \leq b{\cdot}i$
shows $content\ (cbox\ a\ b) = (\prod i{\in}Basis.\ b{\cdot}i - a{\cdot}i)$
⟨*proof*⟩

lemma *content-cbox'*:
fixes $a :: 'a::euclidean\text{-}space$
assumes $cbox\ a\ b \neq \{\}$
shows $content\ (cbox\ a\ b) = (\prod i{\in}Basis.\ b{\cdot}i - a{\cdot}i)$
⟨*proof*⟩

lemma *content-real*: $a \leq b \Longrightarrow content\ \{a..b\} = b - a$
⟨*proof*⟩

lemma *abs-eq-content*: $|y - x| = (if\ x{\leq}y\ then\ content\ \{x\ ..\ y\}\ else\ content\ \{y..x\})$
⟨*proof*⟩

lemma *content-singleton*[*simp*]: $content\ \{a\} = 0$
⟨*proof*⟩

lemma *content-unit*[*iff*]: $content(cbox\ 0\ (One::'a::euclidean\text{-}space)) = 1$
⟨*proof*⟩

lemma *content-pos-le*[*intro*]:
fixes $a::'a::euclidean\text{-}space$
shows $0 \leq content\ (cbox\ a\ b)$
⟨*proof*⟩

corollary *content-nonneg* [*simp*]:
fixes $a::'a::euclidean\text{-}space$

shows ~ *content* (*cbox a b*) < *0*
⟨*proof*⟩

lemma *content-pos-lt*:
fixes *a* :: *'a*::*euclidean-space*
assumes ∀ *i*∈*Basis*. *a*•*i* < *b*•*i*
shows *0* < *content* (*cbox a b*)
⟨*proof*⟩

lemma *content-eq-0*:
content (*cbox a b*) = *0* ⟷ (∃ *i*∈*Basis*. *b*•*i* ≤ *a*•*i*)
⟨*proof*⟩

lemma *cond-cases*: (*P* ⟹ *Q x*) ⟹ (¬ *P* ⟹ *Q y*) ⟹ *Q* (*if P then x else y*)
⟨*proof*⟩

lemma *content-cbox-cases*:
content (*cbox a* (*b*::*'a*::*euclidean-space*)) =
(*if* ∀ *i*∈*Basis*. *a*•*i* ≤ *b*•*i then setprod* (λ*i*. *b*•*i* − *a*•*i*) *Basis else 0*)
⟨*proof*⟩

lemma *content-eq-0-interior*: *content* (*cbox a b*) = *0* ⟷ *interior*(*cbox a b*) = {}
⟨*proof*⟩

lemma *content-pos-lt-eq*:
0 < *content* (*cbox a* (*b*::*'a*::*euclidean-space*)) ⟷ (∀ *i*∈*Basis*. *a*•*i* < *b*•*i*)
⟨*proof*⟩

lemma *content-empty* [*simp*]: *content* {} = *0*
⟨*proof*⟩

lemma *content-real-if* [*simp*]: *content* {*a*..*b*} = (*if a* ≤ *b then b* − *a else 0*)
⟨*proof*⟩

lemma *content-subset*:
assumes *cbox a b* ⊆ *cbox c d*
shows *content* (*cbox a b*) ≤ *content* (*cbox c d*)
⟨*proof*⟩

lemma *content-lt-nz*: *0* < *content* (*cbox a b*) ⟷ *content* (*cbox a b*) ≠ *0*
⟨*proof*⟩

lemma *content-times*[*simp*]: *content* (*A* × *B*) = *content A* ∗ *content B*
⟨*proof*⟩

lemma *content-Pair*: *content* (*cbox* (*a*,*c*) (*b*,*d*)) = *content* (*cbox a b*) ∗ *content* (*cbox c d*)
⟨*proof*⟩

lemma *content-cbox-pair-eq0-D*:
content (cbox (a,c) (b,d)) = 0 ⟹ content (cbox a b) = 0 ∨ content (cbox c d) = 0
⟨*proof*⟩

lemma *content-eq-0-gen*:
fixes *s* :: *'a*::*euclidean-space set*
assumes *bounded s*
shows *content s = 0* ⟷ (∃ *i*∈*Basis*. ∃ *v*. ∀ *x* ∈ *s*. *x* ∙ *i* = *v*) (**is** *- = ?rhs*)
⟨*proof*⟩

lemma *content-0-subset-gen*:
fixes *a* :: *'a*::*euclidean-space*
assumes *content t = 0 s* ⊆ *t bounded t* **shows** *content s = 0*
⟨*proof*⟩

lemma *content-0-subset*: ⟦*content*(*cbox a b*) = *0*; *s* ⊆ *cbox a b*⟧ ⟹ *content s = 0*
⟨*proof*⟩

41.5 The notion of a gauge — simply an open set containing the point.

definition *gauge d* ⟷ (∀ *x*. *x* ∈ *d x* ∧ *open* (*d x*))

lemma *gaugeI*:
assumes ⋀*x*. *x* ∈ *g x*
and ⋀*x*. *open* (*g x*)
shows *gauge g*
⟨*proof*⟩

lemma *gaugeD*[*dest*]:
assumes *gauge d*
shows *x* ∈ *d x*
and *open* (*d x*)
⟨*proof*⟩

lemma *gauge-ball-dependent*: ∀ *x*. *0* < *e x* ⟹ *gauge* (λ*x*. *ball x* (*e x*))
⟨*proof*⟩

lemma *gauge-ball*[*intro*]: *0* < *e* ⟹ *gauge* (λ*x*. *ball x e*)
⟨*proof*⟩

lemma *gauge-trivial*[*intro*!]: *gauge* (λ*x*. *ball x 1*)
⟨*proof*⟩

lemma *gauge-inter*[*intro*]: *gauge d1* ⟹ *gauge d2* ⟹ *gauge* (λ*x*. *d1 x* ∩ *d2 x*)
⟨*proof*⟩

lemma *gauge-inters*:

assumes *finite s*
and $\forall\, d{\in}s.\ gauge\ (f\ d)$
shows *gauge* $(\lambda x.\ \bigcap\{f\ d\ x \mid d.\ d \in s\})$
⟨*proof*⟩

lemma *gauge-existence-lemma*:
$(\forall\, x.\ \exists\, d :: real.\ p\ x \longrightarrow 0 < d \wedge q\ d\ x) \longleftrightarrow (\forall\, x.\ \exists\, d{>}0.\ p\ x \longrightarrow q\ d\ x)$
⟨*proof*⟩

41.6 Divisions.

definition *division-of* (**infixl** *division'-of 40*)
where
s *division-of* $i \longleftrightarrow$
finite $s \wedge$
$(\forall\, k{\in}s.\ k \subseteq i \wedge k \neq \{\} \wedge (\exists\, a\ b.\ k = cbox\ a\ b)) \wedge$
$(\forall\, k1{\in}s.\ \forall\, k2{\in}s.\ k1 \neq k2 \longrightarrow interior(k1) \cap interior(k2) = \{\}) \wedge$
$(\bigcup s = i)$

lemma *division-ofD*[*dest*]:
assumes s *division-of* i
shows *finite* s
and $\bigwedge k.\ k \in s \Longrightarrow k \subseteq i$
and $\bigwedge k.\ k \in s \Longrightarrow k \neq \{\}$
and $\bigwedge k.\ k \in s \Longrightarrow \exists\, a\ b.\ k = cbox\ a\ b$
and $\bigwedge k1\ k2.\ k1 \in s \Longrightarrow k2 \in s \Longrightarrow k1 \neq k2 \Longrightarrow interior(k1) \cap interior(k2) = \{\}$
and $\bigcup s = i$
⟨*proof*⟩

lemma *division-ofI*:
assumes *finite* s
and $\bigwedge k.\ k \in s \Longrightarrow k \subseteq i$
and $\bigwedge k.\ k \in s \Longrightarrow k \neq \{\}$
and $\bigwedge k.\ k \in s \Longrightarrow \exists\, a\ b.\ k = cbox\ a\ b$
and $\bigwedge k1\ k2.\ k1 \in s \Longrightarrow k2 \in s \Longrightarrow k1 \neq k2 \Longrightarrow interior\ k1 \cap interior\ k2 = \{\}$
and $\bigcup s = i$
shows s *division-of* i
⟨*proof*⟩

lemma *division-of-finite*: s *division-of* $i \Longrightarrow$ *finite* s
⟨*proof*⟩

lemma *division-of-self*[*intro*]: $cbox\ a\ b \neq \{\} \Longrightarrow \{cbox\ a\ b\}$ *division-of* $(cbox\ a\ b)$
⟨*proof*⟩

lemma *division-of-trivial*[*simp*]: s *division-of* $\{\} \longleftrightarrow s = \{\}$
⟨*proof*⟩

lemma *division-of-sing*[*simp*]:
  s division-of cbox a (*a*::'*a*::*euclidean-space*) ⟷ *s* = {*cbox a a*}
  (**is** *?l* = *?r*)
⟨*proof*⟩

lemma *elementary-empty*: **obtains** *p* **where** *p division-of* {}
  ⟨*proof*⟩

lemma *elementary-interval*: **obtains** *p* **where** *p division-of* (*cbox a b*)
  ⟨*proof*⟩

lemma *division-contains*: *s division-of i* ⟹ ∀ *x*∈*i*. ∃ *k*∈*s*. *x* ∈ *k*
  ⟨*proof*⟩

lemma *forall-in-division*:
  d division-of i ⟹ (∀ *x*∈*d*. *P x*) ⟷ (∀ *a b*. *cbox a b* ∈ *d* ⟶ *P* (*cbox a b*))
  ⟨*proof*⟩

lemma *division-of-subset*:
  assumes *p division-of* (⋃ *p*)
    and *q* ⊆ *p*
  shows *q division-of* (⋃ *q*)
⟨*proof*⟩

lemma *division-of-union-self*[*intro*]: *p division-of s* ⟹ *p division-of* (⋃ *p*)
  ⟨*proof*⟩

lemma *division-of-content-0*:
  assumes *content* (*cbox a b*) = *0 d division-of* (*cbox a b*)
  shows ∀ *k*∈*d*. *content k* = *0*
  ⟨*proof*⟩

lemma *division-inter*:
  fixes *s1 s2* :: '*a*::*euclidean-space set*
  assumes *p1 division-of s1*
    and *p2 division-of s2*
  shows {*k1* ∩ *k2* | *k1 k2* .*k1* ∈ *p1* ∧ *k2* ∈ *p2* ∧ *k1* ∩ *k2* ≠ {}} *division-of* (*s1* ∩ *s2*)
  (**is** *?A'* *division-of* -)
⟨*proof*⟩

lemma *division-inter-1*:
  assumes *d division-of i*
    and *cbox a* (*b*::'*a*::*euclidean-space*) ⊆ *i*
  shows {*cbox a b* ∩ *k* | *k*. *k* ∈ *d* ∧ *cbox a b* ∩ *k* ≠ {}} *division-of* (*cbox a b*)
⟨*proof*⟩

lemma *elementary-inter*:

fixes $s\ t :: 'a::euclidean\text{-}space\ set$
assumes *p1 division-of s*
and *p2 division-of t*
shows $\exists p.\ p\ division\text{-}of\ (s \cap t)$
⟨*proof*⟩

lemma *elementary-inters*:
assumes *finite f*
and $f \neq \{\}$
and $\forall s{\in}f.\ \exists p.\ p\ division\text{-}of\ (s::('a::euclidean\text{-}space)\ set)$
shows $\exists p.\ p\ division\text{-}of\ (\bigcap f)$
⟨*proof*⟩

lemma *division-disjoint-union*:
assumes *p1 division-of s1*
and *p2 division-of s2*
and *interior s1* $\cap$ *interior s2* $= \{\}$
shows $(p1 \cup p2)\ division\text{-}of\ (s1 \cup s2)$
⟨*proof*⟩

lemma *partial-division-extend-1*:
fixes $a\ b\ c\ d :: 'a::euclidean\text{-}space$
assumes *incl*: $cbox\ c\ d \subseteq cbox\ a\ b$
and *nonempty*: $cbox\ c\ d \neq \{\}$
obtains p **where** $p\ division\text{-}of\ (cbox\ a\ b)\ cbox\ c\ d \in p$
⟨*proof*⟩

lemma *partial-division-extend-interval*:
assumes $p\ division\text{-}of\ (\bigcup p)\ (\bigcup p) \subseteq cbox\ a\ b$
obtains q **where** $p \subseteq q\ q\ division\text{-}of\ cbox\ a\ (b::'a::euclidean\text{-}space)$
⟨*proof*⟩

lemma *elementary-bounded*[*dest*]:
fixes $s :: 'a::euclidean\text{-}space\ set$
shows $p\ division\text{-}of\ s \Longrightarrow bounded\ s$
⟨*proof*⟩

lemma *elementary-subset-cbox*:
$p\ division\text{-}of\ s \Longrightarrow \exists a\ b.\ s \subseteq cbox\ a\ (b::'a::euclidean\text{-}space)$
⟨*proof*⟩

lemma *division-union-intervals-exists*:
fixes $a\ b :: 'a::euclidean\text{-}space$
assumes $cbox\ a\ b \neq \{\}$
obtains p **where** $(insert\ (cbox\ a\ b)\ p)\ division\text{-}of\ (cbox\ a\ b \cup cbox\ c\ d)$
⟨*proof*⟩

lemma *division-of-unions*:
assumes *finite f*

and $\bigwedge p.\ p \in f \Longrightarrow p$ *division-of* $(\bigcup p)$
and $\bigwedge k1\ k2.\ k1 \in \bigcup f \Longrightarrow k2 \in \bigcup f \Longrightarrow k1 \neq k2 \Longrightarrow$ *interior k1* $\cap$ *interior k2* $= \{\}$
shows $\bigcup f$ *division-of* $\bigcup\bigcup f$
⟨*proof*⟩

lemma *elementary-union-interval*:
fixes *a b* :: *'a*::*euclidean-space*
assumes *p division-of* $\bigcup p$
obtains *q* **where** *q division-of* (*cbox a b* $\cup \bigcup p$)
⟨*proof*⟩

lemma *elementary-unions-intervals*:
assumes *fin*: *finite f*
and $\bigwedge s.\ s \in f \Longrightarrow \exists\, a\ b.\ s =$ *cbox a* (*b*::*'a*::*euclidean-space*)
obtains *p* **where** *p division-of* $(\bigcup f)$
⟨*proof*⟩

lemma *elementary-union*:
fixes *s t* :: *'a*::*euclidean-space set*
assumes *ps division-of s pt division-of t*
obtains *p* **where** *p division-of* $(s \cup t)$
⟨*proof*⟩

lemma *partial-division-extend*:
fixes *t* :: *'a*::*euclidean-space set*
assumes *p division-of s*
and *q division-of t*
and $s \subseteq t$
obtains *r* **where** $p \subseteq r$ **and** *r division-of t*
⟨*proof*⟩

41.7 Tagged (partial) divisions.

definition *tagged-partial-division-of* (**infixr** *tagged'-partial'-division'-of 40*)
where *s tagged-partial-division-of i* $\longleftrightarrow$
finite s $\wedge$
$(\forall\, x\ k.\ (x,\ k) \in s \longrightarrow x \in k \wedge k \subseteq i \wedge (\exists\, a\ b.\ k =$ *cbox a b*$)) \wedge$
$(\forall\, x1\ k1\ x2\ k2.\ (x1,\ k1) \in s \wedge (x2,\ k2) \in s \wedge (x1,\ k1) \neq (x2,\ k2) \longrightarrow$
interior k1 $\cap$ *interior k2* $= \{\}$)

lemma *tagged-partial-division-ofD*[*dest*]:
assumes *s tagged-partial-division-of i*
shows *finite s*
and $\bigwedge x\ k.\ (x,k) \in s \Longrightarrow x \in k$
and $\bigwedge x\ k.\ (x,k) \in s \Longrightarrow k \subseteq i$
and $\bigwedge x\ k.\ (x,k) \in s \Longrightarrow \exists\, a\ b.\ k =$ *cbox a b*
and $\bigwedge x1\ k1\ x2\ k2.\ (x1,k1) \in s \Longrightarrow$
$(x2,\ k2) \in s \Longrightarrow (x1,\ k1) \neq (x2,\ k2) \Longrightarrow$ *interior k1* $\cap$ *interior k2* $= \{\}$

⟨*proof*⟩

definition *tagged-division-of* (**infixr** *tagged'-division'-of 40*)
where *s tagged-division-of i* ⟷ *s tagged-partial-division-of i* ∧ (⋃{*k*. ∃ *x*. (*x*,*k*) ∈ *s*} = *i*)

lemma *tagged-division-of-finite*: *s tagged-division-of i* ⟹ *finite s*
⟨*proof*⟩

lemma *tagged-division-of*:
s tagged-division-of i ⟷
finite s ∧
(∀ *x k*. (*x*, *k*) ∈ *s* ⟶ *x* ∈ *k* ∧ *k* ⊆ *i* ∧ (∃ *a b*. *k* = *cbox a b*)) ∧
(∀ *x1 k1 x2 k2*. (*x1*, *k1*) ∈ *s* ∧ (*x2*, *k2*) ∈ *s* ∧ (*x1*, *k1*) ≠ (*x2*, *k2*) ⟶
interior k1 ∩ *interior k2* = {}) ∧
(⋃{*k*. ∃ *x*. (*x*,*k*) ∈ *s*} = *i*)
⟨*proof*⟩

lemma *tagged-division-ofI*:
assumes *finite s*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ *x* ∈ *k*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ *k* ⊆ *i*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ ∃ *a b*. *k* = *cbox a b*
and ⋀*x1 k1 x2 k2*. (*x1*,*k1*) ∈ *s* ⟹ (*x2*, *k2*) ∈ *s* ⟹ (*x1*, *k1*) ≠ (*x2*, *k2*) ⟹
interior k1 ∩ *interior k2* = {}
and (⋃{*k*. ∃ *x*. (*x*,*k*) ∈ *s*} = *i*)
shows *s tagged-division-of i*
⟨*proof*⟩

lemma *tagged-division-ofD*[*dest*]:
assumes *s tagged-division-of i*
shows *finite s*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ *x* ∈ *k*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ *k* ⊆ *i*
and ⋀*x k*. (*x*,*k*) ∈ *s* ⟹ ∃ *a b*. *k* = *cbox a b*
and ⋀*x1 k1 x2 k2*. (*x1*, *k1*) ∈ *s* ⟹ (*x2*, *k2*) ∈ *s* ⟹ (*x1*, *k1*) ≠ (*x2*, *k2*) ⟹
interior k1 ∩ *interior k2* = {}
and (⋃{*k*. ∃ *x*. (*x*,*k*) ∈ *s*} = *i*)
⟨*proof*⟩

lemma *division-of-tagged-division*:
assumes *s tagged-division-of i*
shows (*snd* ‘ *s*) *division-of i*
⟨*proof*⟩

lemma *partial-division-of-tagged-division*:
assumes *s tagged-partial-division-of i*
shows (*snd* ‘ *s*) *division-of* ⋃(*snd* ‘ *s*)
⟨*proof*⟩

lemma *tagged-partial-division-subset*:
assumes *s tagged-partial-division-of i*
and $t \subseteq s$
shows *t tagged-partial-division-of i*
⟨*proof*⟩

lemma *setsum-over-tagged-division-lemma*:
assumes *p tagged-division-of i*
and $\bigwedge u\ v.\ cbox\ u\ v \neq \{\} \Longrightarrow content\ (cbox\ u\ v) = 0 \Longrightarrow d\ (cbox\ u\ v) = 0$
shows $setsum\ (\lambda(x,k).\ d\ k)\ p = setsum\ d\ (snd\ `\ p)$
⟨*proof*⟩

lemma *tag-in-interval*: $p\ tagged\text{-}division\text{-}of\ i \Longrightarrow (x,\ k) \in p \Longrightarrow x \in i$
⟨*proof*⟩

lemma *tagged-division-of-empty*: $\{\}\ tagged\text{-}division\text{-}of\ \{\}$
⟨*proof*⟩

lemma *tagged-partial-division-of-trivial*[*simp*]: $p\ tagged\text{-}partial\text{-}division\text{-}of\ \{\} \longleftrightarrow p = \{\}$
⟨*proof*⟩

lemma *tagged-division-of-trivial*[*simp*]: $p\ tagged\text{-}division\text{-}of\ \{\} \longleftrightarrow p = \{\}$
⟨*proof*⟩

lemma *tagged-division-of-self*: $x \in cbox\ a\ b \Longrightarrow \{(x,cbox\ a\ b)\}\ tagged\text{-}division\text{-}of\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *tagged-division-of-self-real*: $x \in \{a\ ..\ b::real\} \Longrightarrow \{(x,\{a\ ..\ b\})\}\ tagged\text{-}division\text{-}of\ \{a\ ..\ b\}$
⟨*proof*⟩

lemma *tagged-division-union*:
assumes *p1 tagged-division-of s1*
and *p2 tagged-division-of s2*
and $interior\ s1 \cap interior\ s2 = \{\}$
shows $(p1 \cup p2)\ tagged\text{-}division\text{-}of\ (s1 \cup s2)$
⟨*proof*⟩

lemma *tagged-division-unions*:
assumes *finite iset*
and $\forall i{\in}iset.\ pfn\ i\ tagged\text{-}division\text{-}of\ i$
and $\forall i1{\in}iset.\ \forall i2{\in}iset.\ i1 \neq i2 \longrightarrow interior(i1) \cap interior(i2) = \{\}$
shows $\bigcup(pfn\ `\ iset)\ tagged\text{-}division\text{-}of\ (\bigcup iset)$
⟨*proof*⟩

lemma *tagged-partial-division-of-union-self*:

assumes *p tagged-partial-division-of s*
shows *p tagged-division-of* $(\bigcup(snd\ `\ p))$
⟨*proof*⟩

lemma *tagged-division-of-union-self*:
assumes *p tagged-division-of s*
shows *p tagged-division-of* $(\bigcup(snd\ `\ p))$
⟨*proof*⟩

41.8 Fine-ness of a partition w.r.t. a gauge.

definition *fine* (**infixr** *fine 46*)
where $d\ fine\ s \longleftrightarrow (\forall (x,k) \in s.\ k \subseteq d\ x)$

lemma *fineI*:
assumes $\bigwedge x\ k.\ (x,\ k) \in s \Longrightarrow k \subseteq d\ x$
shows *d fine s*
⟨*proof*⟩

lemma *fineD*[*dest*]:
assumes *d fine s*
shows $\bigwedge x\ k.\ (x,k) \in s \Longrightarrow k \subseteq d\ x$
⟨*proof*⟩

lemma *fine-inter*: $(\lambda x.\ d1\ x \cap d2\ x)\ fine\ p \longleftrightarrow d1\ fine\ p \wedge d2\ fine\ p$
⟨*proof*⟩

lemma *fine-inters*:
$(\lambda x.\ \bigcap\{f\ d\ x \mid d.\ \ d \in s\})\ fine\ p \longleftrightarrow (\forall\, d{\in}s.\ (f\ d)\ fine\ p)$
⟨*proof*⟩

lemma *fine-union*: *d fine p1* $\Longrightarrow$ *d fine p2* $\Longrightarrow$ *d fine* $(p1 \cup p2)$
⟨*proof*⟩

lemma *fine-unions*: $(\bigwedge p.\ p \in ps \Longrightarrow d\ fine\ p) \Longrightarrow d\ fine\ (\bigcup ps)$
⟨*proof*⟩

lemma *fine-subset*: $p \subseteq q \Longrightarrow d\ fine\ q \Longrightarrow d\ fine\ p$
⟨*proof*⟩

41.9 Gauge integral. Define on compact intervals first, then use a limit.

definition *has-integral-compact-interval* (**infixr** *has'-integral'-compact'-interval 46*)
where $(f\ has\text{-}integral\text{-}compact\text{-}interval\ y)\ i \longleftrightarrow$
$(\forall\, e{>}0.\ \exists\, d.\ gauge\ d \wedge$
$(\forall\, p.\ p\ tagged\text{-}division\text{-}of\ i \wedge d\ fine\ p \longrightarrow$
$norm\ (setsum\ (\lambda(x,k).\ content\ k *_R f\ x)\ p - y) < e))$

definition *has-integral* ::
  (*'n*::*euclidean-space* ⇒ *'b*::*real-normed-vector*) ⇒ *'b* ⇒ *'n set* ⇒ *bool*
 (**infixr** *has'-integral 46*)
 where (*f has-integral y*) *i* ⟷
  (*if* ∃ *a b*. *i* = *cbox a b*
   then (*f has-integral-compact-interval y*) *i*
   else (∀ *e*>*0*. ∃ *B*>*0*. ∀ *a b*. *ball 0 B* ⊆ *cbox a b* ⟶
    (∃ *z*. ((λ*x*. *if x* ∈ *i then f x else 0*) *has-integral-compact-interval z*) (*cbox a b*)
∧
      norm (*z* − *y*) < *e*)))

lemma *has-integral*:
 (*f has-integral y*) (*cbox a b*) ⟷
  (∀ *e*>*0*. ∃ *d*. *gauge d* ∧
   (∀ *p*. *p tagged-division-of* (*cbox a b*) ∧ *d fine p* ⟶
    norm (*setsum* (λ(*x*,*k*). *content*(*k*) *R* *f x*) *p* − *y*) < *e*))
 ⟨*proof*⟩

lemma *has-integral-real*:
 (*f has-integral y*) {*a* .. *b*::*real*} ⟷
  (∀ *e*>*0*. ∃ *d*. *gauge d* ∧
   (∀ *p*. *p tagged-division-of* {*a* .. *b*} ∧ *d fine p* ⟶
    norm (*setsum* (λ(*x*,*k*). *content*(*k*) *R* *f x*) *p* − *y*) < *e*))
 ⟨*proof*⟩

lemma *has-integralD*[*dest*]:
 assumes (*f has-integral y*) (*cbox a b*)
  and *e* > *0*
 obtains *d* **where** *gauge d*
  and ⋀*p*. *p tagged-division-of* (*cbox a b*) ⟹ *d fine p* ⟹
   norm (*setsum* (λ(*x*,*k*). *content*(*k*) *R* *f*(*x*)) *p* − *y*) < *e*
 ⟨*proof*⟩

lemma *has-integral-alt*:
 (*f has-integral y*) *i* ⟷
  (*if* ∃ *a b*. *i* = *cbox a b*
   then (*f has-integral y*) *i*
   else (∀ *e*>*0*. ∃ *B*>*0*. ∀ *a b*. *ball 0 B* ⊆ *cbox a b* ⟶
    (∃ *z*. ((λ*x*. *if x* ∈ *i then f*(*x*) *else 0*) *has-integral z*) (*cbox a b*) ∧ *norm* (*z* −
y) < *e*)))
 ⟨*proof*⟩

lemma *has-integral-altD*:
 assumes (*f has-integral y*) *i*
  and ¬ (∃ *a b*. *i* = *cbox a b*)
  and *e*>*0*
 obtains *B* **where** *B* > *0*
  and ∀ *a b*. *ball 0 B* ⊆ *cbox a b* ⟶
   (∃ *z*. ((λ*x*. *if x* ∈ *i then f*(*x*) *else 0*) *has-integral z*) (*cbox a b*) ∧ *norm*(*z* − *y*)

$< e$)
⟨*proof*⟩

definition *integrable-on* (**infixr** *integrable'-on 46*)
where *f integrable-on i* ⟷ (∃ *y*. (*f has-integral y*) *i*)

definition *integral i f* = (*SOME y*. (*f has-integral y*) *i* ∨ ~ *f integrable-on i* ∧ *y=0*)

lemma *integrable-integral*[*dest*]: *f integrable-on i* ⟹ (*f has-integral* (*integral i f*)) *i*
⟨*proof*⟩

lemma *not-integrable-integral*: ~ *f integrable-on i* ⟹ *integral i f* = *0*
⟨*proof*⟩

lemma *has-integral-integrable*[*intro*]: (*f has-integral i*) *s* ⟹ *f integrable-on s*
⟨*proof*⟩

lemma *has-integral-integral*: *f integrable-on s* ⟷ (*f has-integral* (*integral s f*)) *s*
⟨*proof*⟩

lemma *setsum-content-null*:
assumes *content* (*cbox a b*) = *0*
and *p tagged-division-of* (*cbox a b*)
shows *setsum* (λ(*x*,*k*). *content k* ∗$_R$ *f x*) *p* = (*0*::'*a*::*real-normed-vector*)
⟨*proof*⟩

41.10 Some basic combining lemmas.

lemma *tagged-division-unions-exists*:
assumes *finite iset*
and ∀ *i*∈*iset*. ∃ *p*. *p tagged-division-of i* ∧ *d fine p*
and ∀ *i1*∈*iset*. ∀ *i2*∈*iset*. *i1* ≠ *i2* ⟶ *interior i1* ∩ *interior i2* = {}
and ⋃ *iset* = *i*
obtains *p* **where** *p tagged-division-of i* **and** *d fine p*
⟨*proof*⟩

41.11 The set we're concerned with must be closed.

lemma *division-of-closed*:
fixes *i* :: '*n*::*euclidean-space set*
shows *s division-of i* ⟹ *closed i*
⟨*proof*⟩

41.12 General bisection principle for intervals; might be useful elsewhere.

lemma *interval-bisection-step*:
fixes *type* :: '*a*::*euclidean-space*

assumes $P\ \{\}$
and $\forall\, s\ t.\ P\ s \wedge P\ t \wedge \mathit{interior}(s) \cap \mathit{interior}(t) = \{\} \longrightarrow P\ (s \cup t)$
and $\neg\ P\ (\mathit{cbox}\ a\ (b::'a))$
obtains $c\ d$ **where** $\neg\ P\ (\mathit{cbox}\ c\ d)$
and $\forall\, i{\in}\mathit{Basis}.\ a{\cdot}i \leq c{\cdot}i \wedge c{\cdot}i \leq d{\cdot}i \wedge d{\cdot}i \leq b{\cdot}i \wedge 2 * (d{\cdot}i - c{\cdot}i) \leq b{\cdot}i - a{\cdot}i$
⟨*proof*⟩

lemma *interval-bisection*:
fixes *type* :: $'a$::*euclidean-space*
assumes $P\ \{\}$
and $(\forall\, s\ t.\ P\ s \wedge P\ t \wedge \mathit{interior}(s) \cap \mathit{interior}(t) = \{\} \longrightarrow P(s \cup t))$
and $\neg\ P\ (\mathit{cbox}\ a\ (b::'a))$
obtains x **where** $x \in \mathit{cbox}\ a\ b$
and $\forall\, e{>}0.\ \exists\, c\ d.\ x \in \mathit{cbox}\ c\ d \wedge \mathit{cbox}\ c\ d \subseteq \mathit{ball}\ x\ e \wedge \mathit{cbox}\ c\ d \subseteq \mathit{cbox}\ a\ b \wedge \neg\ P\ (\mathit{cbox}\ c\ d)$
⟨*proof*⟩

41.13 Cousin's lemma.

lemma *fine-division-exists*:
fixes $a\ b$ :: $'a$::*euclidean-space*
assumes *gauge g*
obtains p **where** p *tagged-division-of* $(\mathit{cbox}\ a\ b)$ *g fine p*
⟨*proof*⟩

lemma *fine-division-exists-real*:
fixes $a\ b$ :: *real*
assumes *gauge g*
obtains p **where** p *tagged-division-of* $\{a\ ..\ b\}$ *g fine p*
⟨*proof*⟩

41.14 Basic theorems about integrals.

lemma *has-integral-unique*:
fixes f :: $'n$::*euclidean-space* ⇒ $'a$::*real-normed-vector*
assumes $(f\ \mathit{has\text{-}integral}\ k1)\ i$
and $(f\ \mathit{has\text{-}integral}\ k2)\ i$
shows $k1 = k2$
⟨*proof*⟩

lemma *integral-unique* [*intro*]: $(f\ \mathit{has\text{-}integral}\ y)\ k \Longrightarrow \mathit{integral}\ k\ f = y$
⟨*proof*⟩

lemma *eq-integralD*: $\mathit{integral}\ k\ f = y \Longrightarrow (f\ \mathit{has\text{-}integral}\ y)\ k \vee {\sim}\ f\ \mathit{integrable\text{-}on}\ k \wedge y{=}0$
⟨*proof*⟩

lemma *has-integral-is-0*:
fixes f :: $'n$::*euclidean-space* ⇒ $'a$::*real-normed-vector*

assumes $\forall x{\in}s.\ f\ x = 0$
shows $(f$ *has-integral* $0)\ s$
⟨*proof*⟩

lemma *has-integral-0*[*simp*]: $((\lambda x{::}'n{::}$*euclidean-space*$.\ 0)$ *has-integral* $0)\ s$
⟨*proof*⟩

lemma *has-integral-0-eq*[*simp*]: $((\lambda x.\ 0)$ *has-integral* $i)\ s \longleftrightarrow i = 0$
⟨*proof*⟩

lemma *has-integral-linear*:
fixes $f :: 'n{::}$*euclidean-space* $\Rightarrow$ $'a{::}$*real-normed-vector*
assumes $(f$ *has-integral* $y)\ s$
and *bounded-linear* h
shows $((h \circ f)$ *has-integral* $((h\ y)))\ s$
⟨*proof*⟩

lemma *has-integral-scaleR-left*:
$(f$ *has-integral* $y)\ s \Longrightarrow ((\lambda x.\ f\ x *_R c)$ *has-integral* $(y *_R c))\ s$
⟨*proof*⟩

lemma *has-integral-mult-left*:
fixes $c :: -$ :: *real-normed-algebra*
shows $(f$ *has-integral* $y)\ s \Longrightarrow ((\lambda x.\ f\ x * c)$ *has-integral* $(y * c))\ s$
⟨*proof*⟩

The case analysis eliminates the condition *f integrable-on s* at the cost of the type class constraint *division-ring*

corollary *integral-mult-left* [*simp*]:
fixes $c{::}\ 'a{::}\{$*real-normed-algebra*,*division-ring*$\}$
shows *integral* $s\ (\lambda x.\ f\ x * c) =$ *integral* $s\ f * c$
⟨*proof*⟩

corollary *integral-mult-right* [*simp*]:
fixes $c{::}\ 'a{::}\{$*real-normed-field*$\}$
shows *integral* $s\ (\lambda x.\ c * f\ x) = c *$ *integral* $s\ f$
⟨*proof*⟩

corollary *integral-divide* [*simp*]:
fixes $z :: 'a{::}$*real-normed-field*
shows *integral* $S\ (\lambda x.\ f\ x\ /\ z) =$ *integral* $S\ (\lambda x.\ f\ x)\ /\ z$
⟨*proof*⟩

lemma *has-integral-mult-right*:
fixes $c :: 'a ::$ *real-normed-algebra*
shows $(f$ *has-integral* $y)\ i \Longrightarrow ((\lambda x.\ c * f\ x)$ *has-integral* $(c * y))\ i$
⟨*proof*⟩

lemma *has-integral-cmul*: $(f$ *has-integral* $k)\ s \Longrightarrow ((\lambda x.\ c *_R f\ x)$ *has-integral* $(c$

$*_R$ k)) s
⟨*proof*⟩

lemma *has-integral-cmult-real*:
fixes c :: *real*
assumes $c \neq 0 \Longrightarrow (f$ *has-integral* $x)$ A
shows $((\lambda x.\ c * f\ x)$ *has-integral* $c * x)$ A
⟨*proof*⟩

lemma *has-integral-neg*: $(f$ *has-integral* $k)$ $s \Longrightarrow ((\lambda x.\ -(f\ x))$ *has-integral* $-k)$ s
⟨*proof*⟩

lemma *has-integral-add*:
fixes f :: $'n$::*euclidean-space* ⇒ $'a$::*real-normed-vector*
assumes $(f$ *has-integral* $k)$ s
and $(g$ *has-integral* $l)$ s
shows $((\lambda x.\ f\ x + g\ x)$ *has-integral* $(k + l))$ s
⟨*proof*⟩

lemma *has-integral-sub*:
$(f$ *has-integral* $k)$ $s \Longrightarrow (g$ *has-integral* $l)$ $s \Longrightarrow$
$((\lambda x.\ f\ x - g\ x)$ *has-integral* $(k - l))$ s
⟨*proof*⟩

lemma *integral-0* [*simp*]:
integral s $(\lambda x$::$'n$::*euclidean-space*. 0::$'m$::*real-normed-vector*$) = 0$
⟨*proof*⟩

lemma *integral-add*: f *integrable-on* $s \Longrightarrow g$ *integrable-on* $s \Longrightarrow$
integral s $(\lambda x.\ f\ x + g\ x) =$ *integral* $s\ f +$ *integral* $s\ g$
⟨*proof*⟩

lemma *integral-cmul* [*simp*]: *integral* s $(\lambda x.\ c *_R f\ x) = c *_R$ *integral* $s\ f$
⟨*proof*⟩

lemma *integral-neg* [*simp*]: *integral* s $(\lambda x.\ -f\ x) = -$ *integral* $s\ f$
⟨*proof*⟩

lemma *integral-diff*: f *integrable-on* $s \Longrightarrow g$ *integrable-on* $s \Longrightarrow$
integral s $(\lambda x.\ f\ x - g\ x) =$ *integral* $s\ f -$ *integral* $s\ g$
⟨*proof*⟩

lemma *integrable-0*: $(\lambda x.\ 0)$ *integrable-on* s
⟨*proof*⟩

lemma *integrable-add*: f *integrable-on* $s \Longrightarrow g$ *integrable-on* $s \Longrightarrow (\lambda x.\ f\ x + g\ x)$
integrable-on s
⟨*proof*⟩

lemma *integrable-cmul*: *f integrable-on s* $\Longrightarrow$ $(\lambda x.\ c *_R f(x))$ *integrable-on s*
⟨*proof*⟩

lemma *integrable-on-cmult-iff*:
fixes *c* :: *real*
assumes $c \neq 0$
shows $(\lambda x.\ c * f\ x)$ *integrable-on s* $\longleftrightarrow$ *f integrable-on s*
⟨*proof*⟩

lemma *integrable-on-cmult-left*:
assumes *f integrable-on s*
shows $(\lambda x.\ of\text{-}real\ c * f\ x)$ *integrable-on s*
⟨*proof*⟩

lemma *integrable-neg*: *f integrable-on s* $\Longrightarrow$ $(\lambda x.\ -f(x))$ *integrable-on s*
⟨*proof*⟩

lemma *integrable-diff*:
f integrable-on s $\Longrightarrow$ *g integrable-on s* $\Longrightarrow$ $(\lambda x.\ f\ x - g\ x)$ *integrable-on s*
⟨*proof*⟩

lemma *integrable-linear*:
f integrable-on s $\Longrightarrow$ *bounded-linear h* $\Longrightarrow$ $(h \circ f)$ *integrable-on s*
⟨*proof*⟩

lemma *integral-linear*:
f integrable-on s $\Longrightarrow$ *bounded-linear h* $\Longrightarrow$ *integral s* $(h \circ f) = h$ (*integral s f*)
⟨*proof*⟩

lemma *integral-component-eq*[*simp*]:
fixes *f* :: *'n*::*euclidean-space* $\Rightarrow$ *'m*::*euclidean-space*
assumes *f integrable-on s*
shows *integral s* $(\lambda x.\ f\ x \cdot k) =$ *integral s f* $\cdot$ *k*
⟨*proof*⟩

lemma *has-integral-setsum*:
assumes *finite t*
and $\forall\, a{\in}t.\ ((f\ a)$ *has-integral* $(i\ a))\ s$
shows $((\lambda x.\ setsum\ (\lambda a.\ f\ a\ x)\ t)$ *has-integral* $(setsum\ i\ t))\ s$
⟨*proof*⟩

lemma *integral-setsum*:
⟦*finite t*; $\forall\, a{\in}t.\ (f\ a)$ *integrable-on s*⟧ $\Longrightarrow$
integral s $(\lambda x.\ setsum\ (\lambda a.\ f\ a\ x)\ t) = setsum\ (\lambda a.\ integral\ s\ (f\ a))\ t$
⟨*proof*⟩

lemma *integrable-setsum*:
⟦*finite t*; $\forall\, a{\in}t.\ (f\ a)$ *integrable-on s*⟧ $\Longrightarrow$ $(\lambda x.\ setsum\ (\lambda a.\ f\ a\ x)\ t)$ *integrable-on s*

⟨*proof*⟩

lemma *has-integral-eq*:
assumes $\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x$
and (*f has-integral k*) *s*
shows (*g has-integral k*) *s*
⟨*proof*⟩

lemma *integrable-eq*: $(\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x) \Longrightarrow f$ *integrable-on* $s \Longrightarrow g$ *integrable-on s*
⟨*proof*⟩

lemma *has-integral-cong*:
assumes $\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x$
shows (*f has-integral i*) *s* = (*g has-integral i*) *s*
⟨*proof*⟩

lemma *integral-cong*:
assumes $\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x$
shows *integral s f* = *integral s g*
⟨*proof*⟩

lemma *integrable-on-cmult-left-iff* [*simp*]:
assumes $c \neq 0$
shows ($\lambda x.$ *of-real c* ∗ *f x*) *integrable-on s* ⟷ *f integrable-on s*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *integrable-on-cmult-right*:
fixes *f* :: - ⇒ *'b* :: {*comm-ring*,*real-algebra-1*,*real-normed-vector*}
assumes *f integrable-on s*
shows ($\lambda x.$ *f x* ∗ *of-real c*) *integrable-on s*
⟨*proof*⟩

lemma *integrable-on-cmult-right-iff* [*simp*]:
fixes *f* :: - ⇒ *'b* :: {*comm-ring*,*real-algebra-1*,*real-normed-vector*}
assumes $c \neq 0$
shows ($\lambda x.$ *f x* ∗ *of-real c*) *integrable-on s* ⟷ *f integrable-on s*
⟨*proof*⟩

lemma *integrable-on-cdivide*:
fixes *f* :: - ⇒ *'b* :: *real-normed-field*
assumes *f integrable-on s*
shows ($\lambda x.$ *f x* / *of-real c*) *integrable-on s*
⟨*proof*⟩

lemma *integrable-on-cdivide-iff* [*simp*]:
fixes *f* :: - ⇒ *'b* :: *real-normed-field*
assumes $c \neq 0$

shows $(\lambda x.\ f\ x\ /\ of\text{-}real\ c)$ *integrable-on* $s \longleftrightarrow f$ *integrable-on* s
⟨*proof*⟩

lemma *has-integral-null* [*intro*]:
assumes $content(cbox\ a\ b) = 0$
shows $(f\ has\text{-}integral\ 0)\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *has-integral-null-real* [*intro*]:
assumes $content\ \{a\ ..\ b{::}real\} = 0$
shows $(f\ has\text{-}integral\ 0)\ \{a\ ..\ b\}$
⟨*proof*⟩

lemma *has-integral-null-eq*[*simp*]: $content\ (cbox\ a\ b) = 0 \Longrightarrow (f\ has\text{-}integral\ i)\ (cbox\ a\ b) \longleftrightarrow i = 0$
⟨*proof*⟩

lemma *integral-null* [*simp*]: $content\ (cbox\ a\ b) = 0 \Longrightarrow integral\ (cbox\ a\ b)\ f = 0$
⟨*proof*⟩

lemma *integrable-on-null* [*intro*]: $content\ (cbox\ a\ b) = 0 \Longrightarrow f$ *integrable-on* $(cbox\ a\ b)$
⟨*proof*⟩

lemma *has-integral-empty*[*intro*]: $(f\ has\text{-}integral\ 0)\ \{\}$
⟨*proof*⟩

lemma *has-integral-empty-eq*[*simp*]: $(f\ has\text{-}integral\ i)\ \{\} \longleftrightarrow i = 0$
⟨*proof*⟩

lemma *integrable-on-empty*[*intro*]: f *integrable-on* $\{\}$
⟨*proof*⟩

lemma *integral-empty*[*simp*]: $integral\ \{\}\ f = 0$
⟨*proof*⟩

lemma *has-integral-refl*[*intro*]:
fixes $a :: 'a{::}euclidean\text{-}space$
shows $(f\ has\text{-}integral\ 0)\ (cbox\ a\ a)$
and $(f\ has\text{-}integral\ 0)\ \{a\}$
⟨*proof*⟩

lemma *integrable-on-refl*[*intro*]: f *integrable-on* $cbox\ a\ a$
⟨*proof*⟩

lemma *integral-refl* [*simp*]: $integral\ (cbox\ a\ a)\ f = 0$
⟨*proof*⟩

lemma *integral-singleton* [*simp*]: $integral\ \{a\}\ f = 0$

⟨*proof*⟩

lemma *integral-blinfun-apply*:
assumes *f integrable-on s*
shows *integral s* (λ*x*. *blinfun-apply h* (*f x*)) = *blinfun-apply h* (*integral s f*)
⟨*proof*⟩

lemma *blinfun-apply-integral*:
assumes *f integrable-on s*
shows *blinfun-apply* (*integral s f*) *x* = *integral s* (λ*y*. *blinfun-apply* (*f y*) *x*)
⟨*proof*⟩

41.15 Cauchy-type criterion for integrability.

lemma *integrable-cauchy*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'a*::{*real-normed-vector*,*complete-space*}
shows *f integrable-on cbox a b* ⟷
(∀ *e*>*0*.∃ *d*. *gauge d* ∧
(∀ *p1 p2*. *p1 tagged-division-of* (*cbox a b*) ∧ *d fine p1* ∧
p2 tagged-division-of (*cbox a b*) ∧ *d fine p2* ⟶
norm (*setsum* (λ(*x*,*k*). *content k* ∗*R f x*) *p1* −
setsum (λ(*x*,*k*). *content k* ∗*R f x*) *p2*) < *e*))
(**is** *?l* = (∀ *e*>*0*. ∃ *d*. *?P e d*))
⟨*proof*⟩

41.16 Additivity of integral on abutting intervals.

lemma *interval-split*:
fixes *a* :: *'a*::*euclidean-space*
assumes *k* ∈ *Basis*
shows
cbox a b ∩ {*x*. *x*•*k* ≤ *c*} = *cbox a* (∑ *i*∈*Basis*. (*if i* = *k then min* (*b*•*k*) *c else b*•*i*) ∗*R i*)
cbox a b ∩ {*x*. *x*•*k* ≥ *c*} = *cbox* (∑ *i*∈*Basis*. (*if i* = *k then max* (*a*•*k*) *c else a*•*i*) ∗*R i*) *b*
⟨*proof*⟩

lemma *content-split*:
fixes *a* :: *'a*::*euclidean-space*
assumes *k* ∈ *Basis*
shows *content* (*cbox a b*) = *content*(*cbox a b* ∩ {*x*. *x*•*k* ≤ *c*}) + *content*(*cbox a b* ∩ {*x*. *x*•*k* ≥ *c*})
⟨*proof*⟩

lemma *division-split-left-inj*:
fixes *type* :: *'a*::*euclidean-space*
assumes *d division-of i*
and *k1* ∈ *d*
and *k2* ∈ *d*
and *k1* ≠ *k2*

and *k1* ∩ {*x*::'*a*. *x*•*k* ≤ *c*} = *k2* ∩ {*x*. *x*•*k* ≤ *c*}
and *k*: *k*∈*Basis*
shows *content*(*k1* ∩ {*x*. *x*•*k* ≤ *c*}) = *0*
⟨*proof*⟩

lemma *division-split-right-inj*:
fixes *type* :: '*a*::*euclidean-space*
assumes *d division-of i*
and *k1* ∈ *d*
and *k2* ∈ *d*
and *k1* ≠ *k2*
and *k1* ∩ {*x*::'*a*. *x*•*k* ≥ *c*} = *k2* ∩ {*x*. *x*•*k* ≥ *c*}
and *k*: *k* ∈ *Basis*
shows *content* (*k1* ∩ {*x*. *x*•*k* ≥ *c*}) = *0*
⟨*proof*⟩

lemma *tagged-division-split-left-inj*:
fixes *x1* :: '*a*::*euclidean-space*
assumes *d*: *d tagged-division-of i*
and *k12*: (*x1*, *k1*) ∈ *d*
(*x2*, *k2*) ∈ *d*
k1 ≠ *k2*
k1 ∩ {*x*. *x*•*k* ≤ *c*} = *k2* ∩ {*x*. *x*•*k* ≤ *c*}
k ∈ *Basis*
shows *content* (*k1* ∩ {*x*. *x*•*k* ≤ *c*}) = *0*
⟨*proof*⟩

lemma *tagged-division-split-right-inj*:
fixes *x1* :: '*a*::*euclidean-space*
assumes *d*: *d tagged-division-of i*
and *k12*: (*x1*, *k1*) ∈ *d*
(*x2*, *k2*) ∈ *d*
k1 ≠ *k2*
k1 ∩ {*x*. *x*•*k* ≥ *c*} = *k2* ∩ {*x*. *x*•*k* ≥ *c*}
k ∈ *Basis*
shows *content* (*k1* ∩ {*x*. *x*•*k* ≥ *c*}) = *0*
⟨*proof*⟩

lemma *division-split*:
fixes *a* :: '*a*::*euclidean-space*
assumes *p division-of* (*cbox a b*)
and *k*: *k*∈*Basis*
shows {*l* ∩ {*x*. *x*•*k* ≤ *c*} | *l*. *l* ∈ *p* ∧ *l* ∩ {*x*. *x*•*k* ≤ *c*} ≠ {}} *division-of*(*cbox a b* ∩ {*x*. *x*•*k* ≤ *c*})
(**is** *?p1 division-of ?I1*)
and {*l* ∩ {*x*. *x*•*k* ≥ *c*} | *l*. *l* ∈ *p* ∧ *l* ∩ {*x*. *x*•*k* ≥ *c*} ≠ {}} *division-of* (*cbox a b* ∩ {*x*. *x*•*k* ≥ *c*})
(**is** *?p2 division-of ?I2*)
⟨*proof*⟩

lemma *has-integral-split*:
fixes f :: *'a::euclidean-space* ⇒ *'b::real-normed-vector*
assumes *fi*: (f *has-integral* i) (*cbox* a b ∩ {x. x•k ≤ c})
and *fj*: (f *has-integral* j) (*cbox* a b ∩ {x. x•k ≥ c})
and k: k ∈ *Basis*
shows (f *has-integral* (i + j)) (*cbox* a b)
⟨*proof*⟩

41.17 A sort of converse, integrability on subintervals.

lemma *tagged-division-union-interval*:
fixes a :: *'a::euclidean-space*
assumes *p1 tagged-division-of* (*cbox* a b ∩ {x. x•k ≤ (c::*real*)})
and *p2 tagged-division-of* (*cbox* a b ∩ {x. x•k ≥ c})
and k: k ∈ *Basis*
shows (*p1* ∪ *p2*) *tagged-division-of* (*cbox* a b)
⟨*proof*⟩

lemma *tagged-division-union-interval-real*:
fixes a :: *real*
assumes *p1 tagged-division-of* ({a .. b} ∩ {x. x•k ≤ (c::*real*)})
and *p2 tagged-division-of* ({a .. b} ∩ {x. x•k ≥ c})
and k: k ∈ *Basis*
shows (*p1* ∪ *p2*) *tagged-division-of* {a .. b}
⟨*proof*⟩

lemma *has-integral-separate-sides*:
fixes f :: *'a::euclidean-space* ⇒ *'b::real-normed-vector*
assumes (f *has-integral* i) (*cbox* a b)
and e > *0*
and k: k ∈ *Basis*
obtains d **where** *gauge* d
∀ *p1 p2*. *p1 tagged-division-of* (*cbox* a b ∩ {x. x•k ≤ c}) ∧ d *fine p1* ∧
p2 tagged-division-of (*cbox* a b ∩ {x. x•k ≥ c}) ∧ d *fine p2* ⟶
norm ((*setsum* (λ(x,k). *content* k *∗R* f x) *p1* + *setsum* (λ(x,k). *content* k *∗R* f x) *p2*) − i) < e
⟨*proof*⟩

lemma *integrable-split*[*intro*]:
fixes f :: *'a::euclidean-space* ⇒ *'b*::{*real-normed-vector*,*complete-space*}
assumes f *integrable-on cbox a b*
and k: k ∈ *Basis*
shows f *integrable-on* (*cbox* a b ∩ {x. x•k ≤ c}) (**is** *?t1*)
and f *integrable-on* (*cbox* a b ∩ {x. x•k ≥ c}) (**is** *?t2*)
⟨*proof*⟩

41.18 Generalized notion of additivity.

definition *neutral opp* = (*SOME* x. ∀ y. *opp* x y = y ∧ *opp* y x = y)

definition *operative* :: $('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow (('b{::}euclidean\text{-}space)\ set \Rightarrow 'a) \Rightarrow bool$
where *operative opp f* $\longleftrightarrow$
$(\forall\, a\ b.\ content\ (cbox\ a\ b) = 0 \longrightarrow f\ (cbox\ a\ b) = neutral\ opp) \wedge$
$(\forall\, a\ b\ c.\ \forall k{\in}Basis.\ f\ (cbox\ a\ b) = opp\ (f(cbox\ a\ b \cap \{x.\ x{\cdot}k \leq c\}))\ (f\ (cbox\ a\ b \cap \{x.\ x{\cdot}k \geq c\})))$

lemma *operativeD*[*dest*]:
fixes *type* :: $'a{::}euclidean\text{-}space$
assumes *operative opp f*
shows $\bigwedge a\ b{::}'a.\ content\ (cbox\ a\ b) = 0 \Longrightarrow f\ (cbox\ a\ b) = neutral\ opp$
and $\bigwedge a\ b\ c\ k.\ k \in Basis \Longrightarrow f\ (cbox\ a\ b) =$
$opp\ (f\ (cbox\ a\ b \cap \{x.\ x{\cdot}k \leq c\}))\ (f\ (cbox\ a\ b \cap \{x.\ x{\cdot}k \geq c\}))$
⟨*proof*⟩

lemma *property-empty-interval*: $\forall\, a\ b.\ content\ (cbox\ a\ b) = 0 \longrightarrow P\ (cbox\ a\ b) \Longrightarrow P\ \{\}$
⟨*proof*⟩

lemma *operative-empty*: *operative opp f* $\Longrightarrow f\ \{\} = neutral\ opp$
⟨*proof*⟩

41.19 Using additivity of lifted function to encode definedness.

fun *lifted* **where**
lifted (*opp* :: $'a \Rightarrow 'a \Rightarrow 'b$) (*Some x*) (*Some y*) = *Some* (*opp x y*)
| *lifted opp None* - = (*None*::$'b$ *option*)
| *lifted opp* - *None* = *None*

lemma *lifted-simp-1*[*simp*]: *lifted opp v None* = *None*
⟨*proof*⟩

definition *monoidal opp* $\longleftrightarrow$
$(\forall\, x\ y.\ opp\ x\ y = opp\ y\ x) \wedge$
$(\forall\, x\ y\ z.\ opp\ x\ (opp\ y\ z) = opp\ (opp\ x\ y)\ z) \wedge$
$(\forall\, x.\ opp\ (neutral\ opp)\ x = x)$

lemma *monoidalI*:
assumes $\bigwedge x\ y.\ opp\ x\ y = opp\ y\ x$
and $\bigwedge x\ y\ z.\ opp\ x\ (opp\ y\ z) = opp\ (opp\ x\ y)\ z$
and $\bigwedge x.\ opp\ (neutral\ opp)\ x = x$
shows *monoidal opp*
⟨*proof*⟩

lemma *monoidal-ac*:
assumes *monoidal opp*
shows [*simp*]: *opp* (*neutral opp*) *a* = *a*
and [*simp*]: *opp a* (*neutral opp*) = *a*

and *opp a b = opp b a*
and *opp (opp a b) c = opp a (opp b c)*
and *opp a (opp b c) = opp b (opp a c)*
⟨*proof*⟩

lemma *neutral-lifted* [*cong*]:
assumes *monoidal opp*
shows *neutral* (*lifted opp*) = *Some* (*neutral opp*)
⟨*proof*⟩

lemma *monoidal-lifted*[*intro*]:
assumes *monoidal opp*
shows *monoidal* (*lifted opp*)
⟨*proof*⟩

definition *support opp f s* = {*x*. *x*∈*s* ∧ *f x* ≠ *neutral opp*}
definition *fold′ opp e s* = (*if finite s then Finite-Set.fold opp e s else e*)
definition *iterate opp s f* = *fold′* (λ*x a*. *opp* (*f x*) *a*) (*neutral opp*) (*support opp f s*)

lemma *support-subset*[*intro*]: *support opp f s* ⊆ *s*
⟨*proof*⟩

lemma *support-empty*[*simp*]: *support opp f* {} = {}
⟨*proof*⟩

lemma *comp-fun-commute-monoidal*[*intro*]:
assumes *monoidal opp*
shows *comp-fun-commute opp*
⟨*proof*⟩

lemma *support-clauses*:
⋀*f g s*. *support opp f* {} = {}
⋀*f g s*. *support opp f* (*insert x s*) =
(*if f*(*x*) = *neutral opp then support opp f s else insert x* (*support opp f s*))
⋀*f g s*. *support opp f* (*s* − {*x*}) = (*support opp f s*) − {*x*}
⋀*f g s*. *support opp f* (*s* ∪ *t*) = (*support opp f s*) ∪ (*support opp f t*)
⋀*f g s*. *support opp f* (*s* ∩ *t*) = (*support opp f s*) ∩ (*support opp f t*)
⋀*f g s*. *support opp f* (*s* − *t*) = (*support opp f s*) − (*support opp f t*)
⋀*f g s*. *support opp g* (*f ‘ s*) = *f ‘* (*support opp* (*g* ∘ *f*) *s*)
⟨*proof*⟩

lemma *finite-support*[*intro*]: *finite s* ⟹ *finite* (*support opp f s*)
⟨*proof*⟩

lemma *iterate-empty*[*simp*]: *iterate opp* {} *f* = *neutral opp*
⟨*proof*⟩

lemma *iterate-insert*[*simp*]:

assumes *monoidal opp*
and *finite s*
shows *iterate opp* (*insert x s*) *f* =
(*if x* ∈ *s then iterate opp s f else opp* (*f x*) (*iterate opp s f*))
⟨*proof*⟩

lemma *iterate-some*:
⟦*monoidal opp*; *finite s*⟧ ⟹ *iterate* (*lifted opp*) *s* (λ*x*. *Some*(*f x*)) = *Some* (*iterate opp s f*)
⟨*proof*⟩

41.20 Two key instances of additivity.

lemma *neutral-add*[*simp*]: *neutral op* + = (*0*::'*a*::*comm-monoid-add*)
⟨*proof*⟩

lemma *operative-content*[*intro*]: *operative* (*op* +) *content*
⟨*proof*⟩

lemma *monoidal-monoid*[*intro*]: *monoidal* ((*op* +)::('*a*::*comm-monoid-add*) ⇒ '*a* ⇒ '*a*)
⟨*proof*⟩

lemma *operative-integral*:
fixes *f* :: '*a*::*euclidean-space* ⇒ '*b*::*banach*
shows *operative* (*lifted*(*op* +)) (λ*i*. *if f integrable-on i then Some*(*integral i f*) *else None*)
⟨*proof*⟩

41.21 Points of division of a partition.

definition *division-points* (*k*::('*a*::*euclidean-space*) *set*) *d* =
{(*j*,*x*). *j* ∈ *Basis* ∧ (*interval-lowerbound k*)·*j* < *x* ∧ *x* < (*interval-upperbound k*)·*j* ∧
(∃ *i*∈*d*. (*interval-lowerbound i*)·*j* = *x* ∨ (*interval-upperbound i*)·*j* = *x*)}

lemma *division-points-finite*:
fixes *i* :: '*a*::*euclidean-space set*
assumes *d division-of i*
shows *finite* (*division-points i d*)
⟨*proof*⟩

lemma *division-points-subset*:
fixes *a* :: '*a*::*euclidean-space*
assumes *d division-of* (*cbox a b*)
and ∀ *i*∈*Basis*. *a*·*i* < *b*·*i* *a*·*k* < *c* *c* < *b*·*k*
and *k*: *k* ∈ *Basis*
shows *division-points* (*cbox a b* ∩ {*x*. *x*·*k* ≤ *c*}) {*l* ∩ {*x*. *x*·*k* ≤ *c*} | *l* . *l* ∈ *d* ∧ *l* ∩ {*x*. *x*·*k* ≤ *c*} ≠ {}} ⊆
division-points (*cbox a b*) *d* (**is** *?t1*)

and *division-points* (*cbox a b* ∩ {*x*. *x*•*k* ≥ *c*}) {*l* ∩ {*x*. *x*•*k* ≥ *c*} | *l* . *l* ∈ *d* ∧ ~(*l* ∩ {*x*. *x*•*k* ≥ *c*} = {})} ⊆
division-points (*cbox a b*) *d* (**is** *?t2*)
⟨*proof*⟩

lemma *division-points-psubset*:
fixes *a* :: *'a*::*euclidean-space*
assumes *d division-of* (*cbox a b*)
and ∀ *i*∈*Basis*. *a*•*i* < *b*•*i* *a*•*k* < *c* *c* < *b*•*k*
and *l* ∈ *d*
and *interval-lowerbound l*•*k* = *c* ∨ *interval-upperbound l*•*k* = *c*
and *k*: *k* ∈ *Basis*
shows *division-points* (*cbox a b* ∩ {*x*. *x*•*k* ≤ *c*}) {*l* ∩ {*x*. *x*•*k* ≤ *c*} | *l*. *l*∈*d* ∧ *l* ∩ {*x*. *x*•*k* ≤ *c*} ≠ {}} ⊂
division-points (*cbox a b*) *d* (**is** *?D1* ⊂ *?D*)
and *division-points* (*cbox a b* ∩ {*x*. *x*•*k* ≥ *c*}) {*l* ∩ {*x*. *x*•*k* ≥ *c*} | *l*. *l*∈*d* ∧ *l* ∩ {*x*. *x*•*k* ≥ *c*} ≠ {}} ⊂
division-points (*cbox a b*) *d* (**is** *?D2* ⊂ *?D*)
⟨*proof*⟩

41.22 Preservation by divisions and tagged divisions.

lemma *support-support*[*simp*]:*support opp f* (*support opp f s*) = *support opp f s*
⟨*proof*⟩

lemma *iterate-support*[*simp*]: *iterate opp* (*support opp f s*) *f* = *iterate opp s f*
⟨*proof*⟩

lemma *iterate-expand-cases*:
iterate opp s f = (*if finite*(*support opp f s*) *then iterate opp* (*support opp f s*) *f else neutral opp*)
⟨*proof*⟩

lemma *iterate-image*:
assumes *monoidal opp*
and *inj-on f s*
shows *iterate opp* (*f* ' *s*) *g* = *iterate opp s* (*g* ∘ *f*)
⟨*proof*⟩

lemma *iterate-nonzero-image-lemma*:
assumes *monoidal opp*
and *finite s g*(*a*) = *neutral opp*
and ∀ *x*∈*s*. ∀ *y*∈*s*. *f x* = *f y* ∧ *x* ≠ *y* ⟶ *g*(*f x*) = *neutral opp*
shows *iterate opp* {*f x* | *x*. *x* ∈ *s* ∧ *f x* ≠ *a*} *g* = *iterate opp s* (*g* ∘ *f*)
⟨*proof*⟩

lemma *iterate-eq-neutral*:

assumes *monoidal opp*
and $\bigwedge x.\ x \in s \Longrightarrow f\ x = neutral\ opp$
shows *iterate opp s f* = *neutral opp*
⟨*proof*⟩

lemma *iterate-op*:
$[\![monoidal\ opp; finite\ s]\!]$
$\Longrightarrow iterate\ opp\ s\ (\lambda x.\ opp\ (f\ x)\ (g\ x)) = opp\ (iterate\ opp\ s\ f)\ (iterate\ opp\ s\ g)$
⟨*proof*⟩

lemma *iterate-eq*:
assumes *monoidal opp*
and $\bigwedge x.\ x \in s \Longrightarrow f\ x = g\ x$
shows *iterate opp s f* = *iterate opp s g*
⟨*proof*⟩

lemma *nonempty-witness*:
assumes $s \neq \{\}$
obtains x **where** $x \in s$
⟨*proof*⟩

lemma *operative-division*:
fixes $f :: 'a::euclidean\text{-}space\ set \Rightarrow 'b$
assumes *monoidal opp*
and *operative opp f*
and *d division-of* (*cbox a b*)
shows *iterate opp d f* = *f* (*cbox a b*)
⟨*proof*⟩

lemma *iterate-image-nonzero*:
assumes *monoidal opp*
and *finite s*
and $\bigwedge x\ y.\ \forall x{\in}s.\ \forall y{\in}s.\ x \neq y \wedge f\ x = f\ y \longrightarrow g\ (f\ x) = neutral\ opp$
shows *iterate opp* (*f ' s*) *g* = *iterate opp s* (*g* ∘ *f*)
⟨*proof*⟩

lemma *operative-tagged-division*:
assumes *monoidal opp*
and *operative opp f*
and *d tagged-division-of* (*cbox a b*)
shows *iterate opp d* (λ(*x*,*l*). *f l*) = *f* (*cbox a b*)
⟨*proof*⟩

41.23 Additivity of content.

lemma *setsum-iterate*:
assumes *finite s*
shows *setsum f s* = *iterate op* + *s f*
⟨*proof*⟩

lemma *additive-content-division*:
d division-of (*cbox a b*) ⟹ *setsum content d* = *content* (*cbox a b*)
⟨*proof*⟩

lemma *additive-content-tagged-division*:
d tagged-division-of (*cbox a b*) ⟹ *setsum* (λ(*x*,*l*). *content l*) *d* = *content* (*cbox a b*)
⟨*proof*⟩

41.24 Finally, the integral of a constant

lemma *has-integral-const* [*intro*]:
fixes *a b* :: *'a*::*euclidean-space*
shows ((λ*x*. *c*) *has-integral* (*content* (*cbox a b*) ∗$_R$ *c*)) (*cbox a b*)
⟨*proof*⟩

lemma *has-integral-const-real* [*intro*]:
fixes *a b* :: *real*
shows ((λ*x*. *c*) *has-integral* (*content* {*a* .. *b*} ∗$_R$ *c*)) {*a* .. *b*}
⟨*proof*⟩

lemma *integral-const* [*simp*]:
fixes *a b* :: *'a*::*euclidean-space*
shows *integral* (*cbox a b*) (λ*x*. *c*) = *content* (*cbox a b*) ∗$_R$ *c*
⟨*proof*⟩

lemma *integral-const-real* [*simp*]:
fixes *a b* :: *real*
shows *integral* {*a* .. *b*} (λ*x*. *c*) = *content* {*a* .. *b*} ∗$_R$ *c*
⟨*proof*⟩

41.25 Bounds on the norm of Riemann sums and the integral itself.

lemma *dsum-bound*:
assumes *p division-of* (*cbox a b*)
and *norm c* ≤ *e*
shows *norm* (*setsum* (λ*l*. *content l* ∗$_R$ *c*) *p*) ≤ *e* ∗ *content*(*cbox a b*)
⟨*proof*⟩

lemma *rsum-bound*:
assumes *p*: *p tagged-division-of* (*cbox a b*)
and ∀ *x*∈*cbox a b*. *norm* (*f x*) ≤ *e*
shows *norm* (*setsum* (λ(*x*,*k*). *content k* ∗$_R$ *f x*) *p*) ≤ *e* ∗ *content* (*cbox a b*)
⟨*proof*⟩

lemma *rsum-diff-bound*:
assumes *p tagged-division-of* (*cbox a b*)

and $\forall x{\in}cbox\ a\ b.\ norm\ (f\ x - g\ x) \leq e$
shows $norm\ (setsum\ (\lambda(x,k).\ content\ k *_R f\ x)\ p - setsum\ (\lambda(x,k).\ content\ k *_R g\ x)\ p) \leq$
$e * content\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *has-integral-bound*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes $0 \leq B$
and $(f\ has\text{-}integral\ i)\ (cbox\ a\ b)$
and $\forall x{\in}cbox\ a\ b.\ norm\ (f\ x) \leq B$
shows $norm\ i \leq B * content\ (cbox\ a\ b)$
⟨*proof*⟩

corollary *has-integral-bound-real*:
fixes $f :: real \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes $0 \leq B$
and $(f\ has\text{-}integral\ i)\ \{a\ ..\ b\}$
and $\forall x{\in}\{a\ ..\ b\}.\ norm\ (f\ x) \leq B$
shows $norm\ i \leq B * content\ \{a\ ..\ b\}$
⟨*proof*⟩

corollary *integrable-bound*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}real\text{-}normed\text{-}vector$
assumes $0 \leq B$
and $f\ integrable\text{-}on\ (cbox\ a\ b)$
and $\bigwedge x.\ x{\in}cbox\ a\ b \Longrightarrow norm\ (f\ x) \leq B$
shows $norm\ (integral\ (cbox\ a\ b)\ f) \leq B * content\ (cbox\ a\ b)$
⟨*proof*⟩

41.26 Similar theorems about relationship among components.

lemma *rsum-component-le*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}euclidean\text{-}space$
assumes $p\ tagged\text{-}division\text{-}of\ (cbox\ a\ b)$
and $\forall x{\in}cbox\ a\ b.\ (f\ x)\cdot i \leq (g\ x)\cdot i$
shows $(setsum\ (\lambda(x,k).\ content\ k *_R f\ x)\ p)\cdot i \leq (setsum\ (\lambda(x,k).\ content\ k *_R g\ x)\ p)\cdot i$
⟨*proof*⟩

lemma *has-integral-component-le*:
fixes $f\ g :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}euclidean\text{-}space$
assumes k: $k \in Basis$
assumes $(f\ has\text{-}integral\ i)\ s\ (g\ has\text{-}integral\ j)\ s$
and $\forall x{\in}s.\ (f\ x)\cdot k \leq (g\ x)\cdot k$
shows $i\cdot k \leq j\cdot k$
⟨*proof*⟩

lemma *integral-component-le*:
fixes *g f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *k* ∈ *Basis*
and *f integrable-on s g integrable-on s*
and ∀ *x*∈*s*. (*f x*)·*k* ≤ (*g x*)·*k*
shows (*integral s f*)·*k* ≤ (*integral s g*)·*k*
⟨*proof*⟩

lemma *has-integral-component-nonneg*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *k* ∈ *Basis*
and (*f has-integral i*) *s*
and ∀ *x*∈*s*. *0* ≤ (*f x*)·*k*
shows *0* ≤ *i*·*k*
⟨*proof*⟩

lemma *integral-component-nonneg*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *k* ∈ *Basis*
and ∀ *x*∈*s*. *0* ≤ (*f x*)·*k*
shows *0* ≤ (*integral s f*)·*k*
⟨*proof*⟩

lemma *has-integral-component-neg*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *k* ∈ *Basis*
and (*f has-integral i*) *s*
and ∀ *x*∈*s*. (*f x*)·*k* ≤ *0*
shows *i*·*k* ≤ *0*
⟨*proof*⟩

lemma *has-integral-component-lbound*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes (*f has-integral i*) (*cbox a b*)
and ∀ *x*∈*cbox a b*. *B* ≤ *f*(*x*)·*k*
and *k* ∈ *Basis*
shows *B* ∗ *content* (*cbox a b*) ≤ *i*·*k*
⟨*proof*⟩

lemma *has-integral-component-ubound*:
fixes *f*::*'a*::*euclidean-space* => *'b*::*euclidean-space*
assumes (*f has-integral i*) (*cbox a b*)
and ∀ *x*∈*cbox a b*. *f x*·*k* ≤ *B*
and *k* ∈ *Basis*
shows *i*·*k* ≤ *B* ∗ *content* (*cbox a b*)
⟨*proof*⟩

lemma *integral-component-lbound*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*

assumes *f integrable-on cbox a b*
and $\forall x{\in}cbox\ a\ b.\ B \leq f(x){\cdot}k$
and $k \in Basis$
shows $B * content\ (cbox\ a\ b) \leq (integral(cbox\ a\ b)\ f){\cdot}k$
⟨*proof*⟩

lemma *integral-component-lbound-real*:
assumes *f integrable-on* $\{a :: real\ ..\ b\}$
and $\forall x{\in}\{a\ ..\ b\}.\ B \leq f(x){\cdot}k$
and $k \in Basis$
shows $B * content\ \{a\ ..\ b\} \leq (integral\ \{a\ ..\ b\}\ f){\cdot}k$
⟨*proof*⟩

lemma *integral-component-ubound*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}euclidean\text{-}space$
assumes *f integrable-on cbox a b*
and $\forall x{\in}cbox\ a\ b.\ f\ x{\cdot}k \leq B$
and $k \in Basis$
shows $(integral\ (cbox\ a\ b)\ f){\cdot}k \leq B * content\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *integral-component-ubound-real*:
fixes $f :: real \Rightarrow 'a{::}euclidean\text{-}space$
assumes *f integrable-on* $\{a\ ..\ b\}$
and $\forall x{\in}\{a\ ..\ b\}.\ f\ x{\cdot}k \leq B$
and $k \in Basis$
shows $(integral\ \{a\ ..\ b\}\ f){\cdot}k \leq B * content\ \{a\ ..\ b\}$
⟨*proof*⟩

41.27 Uniform limit of integrable functions is integrable.

lemma *real-arch-invD*:
$0 < (e{::}real) \Longrightarrow (\exists n{::}nat.\ n \neq 0 \wedge 0 < inverse\ (real\ n) \wedge inverse\ (real\ n) < e)$
⟨*proof*⟩

lemma *integrable-uniform-limit*:
fixes $f :: 'a{::}euclidean\text{-}space \Rightarrow 'b{::}banach$
assumes $\forall e{>}0.\ \exists g.\ (\forall x{\in}cbox\ a\ b.\ norm\ (f\ x - g\ x) \leq e) \wedge g\ integrable\text{-}on\ cbox\ a\ b$
shows *f integrable-on cbox a b*
⟨*proof*⟩

lemmas *integrable-uniform-limit-real* = *integrable-uniform-limit* [**where** $'a{=}real$, *simplified*]

41.28 Negligible sets.

definition *negligible* $(s{::}\ 'a{::}euclidean\text{-}space\ set) \longleftrightarrow$
$(\forall a\ b.\ ((indicator\ s :: 'a{\Rightarrow}real)\ has\text{-}integral\ 0)\ (cbox\ a\ b))$

41.29 Negligibility of hyperplane.

lemma *setsum-nonzero-image-lemma*:
assumes *finite s*
and $g\ a = 0$
and $\forall x{\in}s.\ \forall y{\in}s.\ f\ x = f\ y \wedge x \neq y \longrightarrow g\ (f\ x) = 0$
shows *setsum g* $\{f\ x\ |x.\ x \in s \wedge f\ x \neq a\}$ = *setsum* $(g \circ f)\ s$
⟨*proof*⟩

lemma *interval-doublesplit*:
fixes a :: $'a$::*euclidean-space*
assumes $k \in Basis$
shows *cbox a b* $\cap$ $\{x\ .\ |x{\cdot}k - c| \leq (e{::}real)\}$ =
cbox $(\sum i{\in}Basis.\ (if\ i = k\ then\ max\ (a{\cdot}k)\ (c - e)\ else\ a{\cdot}i) *_R i)$
$(\sum i{\in}Basis.\ (if\ i = k\ then\ min\ (b{\cdot}k)\ (c + e)\ else\ b{\cdot}i) *_R i)$
⟨*proof*⟩

lemma *division-doublesplit*:
fixes a :: $'a$::*euclidean-space*
assumes *p division-of* (*cbox a b*)
and k: $k \in Basis$
shows $\{l \cap \{x.\ |x{\cdot}k - c| \leq e\}\ |l.\ l \in p \wedge l \cap \{x.\ |x{\cdot}k - c| \leq e\} \neq \{\}\}$
division-of (*cbox a b* $\cap$ $\{x.\ |x{\cdot}k - c| \leq e\}$)
⟨*proof*⟩

lemma *content-doublesplit*:
fixes a :: $'a$::*euclidean-space*
assumes $0 < e$
and k: $k \in Basis$
obtains d **where** $0 < d$ **and** *content* (*cbox a b* $\cap$ $\{x.\ |x{\cdot}k - c| \leq d\}$) $< e$
⟨*proof*⟩

lemma *negligible-standard-hyperplane*[*intro*]:
fixes k :: $'a$::*euclidean-space*
assumes k: $k \in Basis$
shows *negligible* $\{x.\ x{\cdot}k = c\}$
⟨*proof*⟩

41.30 A technical lemma about "refinement" of division.

lemma *tagged-division-finer*:
fixes p :: ($'a$::*euclidean-space* $\times$ ($'a$::*euclidean-space set*)) *set*
assumes *p tagged-division-of* (*cbox a b*)
and *gauge d*
obtains q **where** *q tagged-division-of* (*cbox a b*)
and *d fine q*
and $\forall (x,k) \in p.\ k \subseteq d(x) \longrightarrow (x,k) \in q$
⟨*proof*⟩

41.31 Hence the main theorem about negligible sets.

lemma *finite-product-dependent*:
assumes *finite s*
and $\bigwedge x.\ x \in s \Longrightarrow$ *finite* $(t\ x)$
shows *finite* $\{(i,\ j)\ |i\ j.\ i \in s \wedge j \in t\ i\}$
⟨*proof*⟩

lemma *sum-sum-product*:
assumes *finite s*
and $\forall\, i{\in}s.$ *finite* $(t\ i)$
shows *setsum* $(\lambda i.$ *setsum* $(x\ i)\ (t\ i)$::*real*$)\ s =$
setsum $(\lambda(i,j).\ x\ i\ j)\ \{(i,j)\ |\ i\ j.\ i \in s \wedge j \in t\ i\}$
⟨*proof*⟩

lemma *has-integral-negligible*:
fixes f :: *'b*::*euclidean-space* ⇒ *'a*::*real-normed-vector*
assumes *negligible s*
and $\forall\, x{\in}(t - s).\ f\ x = 0$
shows $(f$ *has-integral 0*$)\ t$
⟨*proof*⟩

lemma *has-integral-spike*:
fixes f :: *'b*::*euclidean-space* ⇒ *'a*::*real-normed-vector*
assumes *negligible s*
and $(\forall\, x{\in}(t - s).\ g\ x = f\ x)$
and $(f$ *has-integral* $y)\ t$
shows $(g$ *has-integral* $y)\ t$
⟨*proof*⟩

lemma *has-integral-spike-eq*:
assumes *negligible s*
and $\forall\, x{\in}(t - s).\ g\ x = f\ x$
shows $((f$ *has-integral* $y)\ t \longleftrightarrow (g$ *has-integral* $y)\ t)$
⟨*proof*⟩

lemma *integrable-spike*:
assumes *negligible s*
and $\forall\, x{\in}(t - s).\ g\ x = f\ x$
and f *integrable-on t*
shows g *integrable-on* t
⟨*proof*⟩

lemma *integral-spike*:
assumes *negligible s*
and $\forall\, x{\in}(t - s).\ g\ x = f\ x$
shows *integral t f* = *integral t g*
⟨*proof*⟩

41.32 Some other trivialities about negligible sets.

lemma *negligible-subset*[*intro*]:
assumes *negligible s*
and $t \subseteq s$
shows *negligible t*
⟨*proof*⟩

lemma *negligible-diff*[*intro?*]:
assumes *negligible s*
shows *negligible* $(s - t)$
⟨*proof*⟩

lemma *negligible-inter*:
assumes *negligible s* ∨ *negligible t*
shows *negligible* $(s \cap t)$
⟨*proof*⟩

lemma *negligible-union*:
assumes *negligible s*
and *negligible t*
shows *negligible* $(s \cup t)$
⟨*proof*⟩

lemma *negligible-union-eq*[*simp*]: *negligible* $(s \cup t) \longleftrightarrow$ *negligible s* ∧ *negligible t*
⟨*proof*⟩

lemma *negligible-sing*[*intro*]: *negligible* {*a*::′*a*::*euclidean-space*}
⟨*proof*⟩

lemma *negligible-insert*[*simp*]: *negligible* (*insert a s*) ⟷ *negligible s*
⟨*proof*⟩

lemma *negligible-empty*[*iff*]: *negligible* {}
⟨*proof*⟩

lemma *negligible-finite*[*intro*]:
assumes *finite s*
shows *negligible s*
⟨*proof*⟩

lemma *negligible-unions*[*intro*]:
assumes *finite s*
and ∀ *t*∈*s*. *negligible t*
shows *negligible*(⋃ *s*)
⟨*proof*⟩

lemma *negligible*:
negligible s ⟷ (∀ *t*::(′*a*::*euclidean-space*) *set*. ((*indicator s*::′*a*⇒*real*) *has-integral* *0*) *t*)

⟨*proof*⟩

41.33 Finite case of the spike theorem is quite commonly needed.

lemma *has-integral-spike-finite*:
assumes *finite s*
and ∀ *x*∈*t*−*s*. *g x* = *f x*
and (*f has-integral y*) *t*
shows (*g has-integral y*) *t*
⟨*proof*⟩

lemma *has-integral-spike-finite-eq*:
assumes *finite s*
and ∀ *x*∈*t*−*s*. *g x* = *f x*
shows ((*f has-integral y*) *t* ⟷ (*g has-integral y*) *t*)
⟨*proof*⟩

lemma *integrable-spike-finite*:
assumes *finite s*
and ∀ *x*∈*t*−*s*. *g x* = *f x*
and *f integrable-on t*
shows *g integrable-on t*
⟨*proof*⟩

41.34 In particular, the boundary of an interval is negligible.

lemma *negligible-frontier-interval*: *negligible*(*cbox* (*a*::′*a*::*euclidean-space*) *b* − *box a b*)
⟨*proof*⟩

lemma *has-integral-spike-interior*:
assumes ∀ *x*∈*box a b*. *g x* = *f x*
and (*f has-integral y*) (*cbox a b*)
shows (*g has-integral y*) (*cbox a b*)
⟨*proof*⟩

lemma *has-integral-spike-interior-eq*:
assumes ∀ *x*∈*box a b*. *g x* = *f x*
shows (*f has-integral y*) (*cbox a b*) ⟷ (*g has-integral y*) (*cbox a b*)
⟨*proof*⟩

lemma *integrable-spike-interior*:
assumes ∀ *x*∈*box a b*. *g x* = *f x*
and *f integrable-on cbox a b*
shows *g integrable-on cbox a b*
⟨*proof*⟩

41.35 Integrability of continuous functions.

lemma *neutral-and*[*simp*]: *neutral op* $\wedge$ = *True*
⟨*proof*⟩

lemma *monoidal-and*[*intro*]: *monoidal op* $\wedge$
⟨*proof*⟩

lemma *iterate-and*[*simp*]:
assumes *finite s*
shows (*iterate op* $\wedge$) *s p* $\longleftrightarrow$ ($\forall$ *x*∈*s*. *p x*)
⟨*proof*⟩

lemma *operative-division-and*:
assumes *operative op* $\wedge$ *P*
and *d division-of* (*cbox a b*)
shows ($\forall$ *i*∈*d*. *P i*) $\longleftrightarrow$ *P* (*cbox a b*)
⟨*proof*⟩

lemma *operative-approximable*:
fixes *f* :: *'b*::*euclidean-space* ⇒ *'a*::*banach*
assumes *0* ≤ *e*
shows *operative op* $\wedge$ (λ*i*. $\exists$ *g*. ($\forall$ *x*∈*i*. *norm* (*f x* − *g* (*x*::*'b*)) ≤ *e*) $\wedge$ *g integrable-on i*)
⟨*proof*⟩

lemma *approximable-on-division*:
fixes *f* :: *'b*::*euclidean-space* ⇒ *'a*::*banach*
assumes *0* ≤ *e*
and *d division-of* (*cbox a b*)
and $\forall$ *i*∈*d*. $\exists$ *g*. ($\forall$ *x*∈*i*. *norm* (*f x* − *g x*) ≤ *e*) $\wedge$ *g integrable-on i*
obtains *g* **where** $\forall$ *x*∈*cbox a b*. *norm* (*f x* − *g x*) ≤ *e g integrable-on cbox a b*
⟨*proof*⟩

lemma *integrable-continuous*:
fixes *f* :: *'b*::*euclidean-space* ⇒ *'a*::*banach*
assumes *continuous-on* (*cbox a b*) *f*
shows *f integrable-on cbox a b*
⟨*proof*⟩

lemma *integrable-continuous-real*:
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *continuous-on* {*a* .. *b*} *f*
shows *f integrable-on* {*a* .. *b*}
⟨*proof*⟩

41.36 Specialization of additivity to one dimension.

lemma
shows *real-inner-1-left*: *inner 1 x* = *x*

and *real-inner-1-right*: *inner x 1 = x*
⟨*proof*⟩

lemma *content-real-eq-0*: *content* {*a .. b*::*real*} = *0* ⟷ *a* ≥ *b*
⟨*proof*⟩

lemma *interval-real-split*:
{*a .. b*::*real*} ∩ {*x. x* ≤ *c*} = {*a .. min b c*}
{*a .. b*} ∩ {*x. c* ≤ *x*} = {*max a c .. b*}
⟨*proof*⟩

lemma *operative-1-lt*:
assumes *monoidal opp*
shows *operative opp f* ⟷ ((∀ *a b. b* ≤ *a* ⟶ *f* {*a .. b*::*real*} = *neutral opp*) ∧
(∀ *a b c. a* < *c* ∧ *c* < *b* ⟶ *opp* (*f* {*a .. c*}) (*f* {*c .. b*}) = *f* {*a .. b*}))
⟨*proof*⟩

lemma *operative-1-le*:
assumes *monoidal opp*
shows *operative opp f* ⟷ ((∀ *a b. b* ≤ *a* ⟶ *f* {*a .. b*::*real*} = *neutral opp*) ∧
(∀ *a b c. a* ≤ *c* ∧ *c* ≤ *b* ⟶ *opp* (*f* {*a .. c*}) (*f* {*c .. b*}) = *f* {*a .. b*}))
⟨*proof*⟩

41.37 Special case of additivity we need for the FCT.

lemma *additive-tagged-division-1*:
fixes *f* :: *real* ⇒ *'a*::*real-normed-vector*
assumes *a* ≤ *b*
and *p tagged-division-of* {*a..b*}
shows *setsum* (λ(*x*,*k*). *f*(*Sup k*) − *f*(*Inf k*)) *p* = *f b* − *f a*
⟨*proof*⟩

41.38 A useful lemma allowing us to factor out the content size.

lemma *has-integral-factor-content*:
(*f has-integral i*) (*cbox a b*) ⟷
(∀ *e*>*0*. ∃ *d*. *gauge d* ∧ (∀ *p*. *p tagged-division-of* (*cbox a b*) ∧ *d fine p* ⟶
norm (*setsum* (λ(*x*,*k*). *content k* ∗*R f x*) *p* − *i*) ≤ *e* ∗ *content* (*cbox a b*)))
⟨*proof*⟩

lemma *has-integral-factor-content-real*:
(*f has-integral i*) {*a .. b*::*real*} ⟷
(∀ *e*>*0*. ∃ *d*. *gauge d* ∧ (∀ *p*. *p tagged-division-of* {*a .. b*} ∧ *d fine p* ⟶
norm (*setsum* (λ(*x*,*k*). *content k* ∗*R f x*) *p* − *i*) ≤ *e* ∗ *content* {*a .. b*}))
⟨*proof*⟩

41.39 Fundamental theorem of calculus.

lemma *interval-bounds-real*:

fixes q b :: *real*
assumes $a \leq b$
shows *Sup* $\{a..b\} = b$
and *Inf* $\{a..b\} = a$
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus*:
fixes f :: *real* ⇒ $'a$::*banach*
assumes $a \leq b$
and $\forall x \in \{a \,..\, b\}$. ($f$ *has-vector-derivative* f' x) (*at* x *within* $\{a \,..\, b\}$)
shows (f' *has-integral* (f b − f a)) $\{a \,..\, b\}$
⟨*proof*⟩

lemma *ident-has-integral*:
fixes a::*real*
assumes $a \leq b$
shows ((λx. x) *has-integral* (b^2 − a^2) / *2*) $\{a..b\}$
⟨*proof*⟩

lemma *integral-ident* [*simp*]:
fixes a::*real*
assumes $a \leq b$
shows *integral* $\{a..b\}$ (λx. x) = (*if* $a \leq b$ *then* (b^2 − a^2) / *2 else 0*)
⟨*proof*⟩

lemma *ident-integrable-on*:
fixes a::*real*
shows (λx. x) *integrable-on* $\{a..b\}$
⟨*proof*⟩

41.40 Taylor series expansion

lemma (**in** *bounded-bilinear*) *setsum-prod-derivatives-has-vector-derivative*:
assumes $p>0$
and *f0*: *Df* 0 = f
and *Df*: ⋀m t. $m < p \Longrightarrow a \leq t \Longrightarrow t \leq b \Longrightarrow$
(*Df* m *has-vector-derivative* *Df* (*Suc* m) t) (*at* t *within* $\{a \,..\, b\}$)
and *g0*: *Dg* 0 = g
and *Dg*: ⋀m t. $m < p \Longrightarrow a \leq t \Longrightarrow t \leq b \Longrightarrow$
(*Dg* m *has-vector-derivative* *Dg* (*Suc* m) t) (*at* t *within* $\{a \,..\, b\}$)
and *ivl*: $a \leq t$ $t \leq b$
shows ((λt. $\sum i<p$. (−1) ^ i $*_R$ *prod* (*Df* i t) (*Dg* (p − *Suc* i) t))
has-vector-derivative
prod (f t) (*Dg* p t) − (−1) ^ p $*_R$ *prod* (*Df* p t) (g t))
(*at* t *within* $\{a \,..\, b\}$)
⟨*proof*⟩

lemma
fixes f::*real*⇒$'a$::*banach*

assumes $p>0$
and *f0*: $Df\ 0 = f$
and *Df*: $\bigwedge m\ t.\ m < p \Longrightarrow a \leq t \Longrightarrow t \leq b \Longrightarrow$
($Df\ m$ *has-vector-derivative* $Df\ (Suc\ m)\ t)\ (at\ t\ within\ \{a\ ..\ b\})$
and *ivl*: $a \leq b$
defines $i \equiv \lambda x.\ ((b - x)\ \hat{}\ (p - 1)\ /\ fact\ (p - 1)) *_R Df\ p\ x$
shows *taylor-has-integral*:
(i *has-integral* $f\ b - (\sum i{<}p.\ ((b - a)\ \hat{}\ i\ /\ fact\ i) *_R Df\ i\ a))\ \{a..b\}$
and *taylor-integral*:
$f\ b = (\sum i{<}p.\ ((b - a)\ \hat{}\ i\ /\ fact\ i) *_R Df\ i\ a) + integral\ \{a..b\}\ i$
and *taylor-integrable*:
i *integrable-on* $\{a\ ..\ b\}$
⟨*proof*⟩

41.41 Attempt a systematic general set of "offset" results for components.

lemma *gauge-modify*:
assumes $(\forall s.\ open\ s \longrightarrow open\ \{x.\ f(x) \in s\})\ gauge\ d$
shows $gauge\ (\lambda x.\ \{y.\ f\ y \in d\ (f\ x)\})$
⟨*proof*⟩

41.42 Only need trivial subintervals if the interval itself is trivial.

lemma *division-of-nontrivial*:
fixes $s :: 'a{::}euclidean\text{-}space\ set\ set$
assumes $s\ division\text{-}of\ (cbox\ a\ b)$
and $content\ (cbox\ a\ b) \neq 0$
shows $\{k.\ k \in s \wedge content\ k \neq 0\}\ division\text{-}of\ (cbox\ a\ b)$
⟨*proof*⟩

41.43 Integrability on subintervals.

lemma *operative-integrable*:
fixes $f :: 'b{::}euclidean\text{-}space \Rightarrow 'a{::}banach$
shows $operative\ op \wedge (\lambda i.\ f\ integrable\text{-}on\ i)$
⟨*proof*⟩

lemma *integrable-subinterval*:
fixes $f :: 'b{::}euclidean\text{-}space \Rightarrow 'a{::}banach$
assumes $f\ integrable\text{-}on\ cbox\ a\ b$
and $cbox\ c\ d \subseteq cbox\ a\ b$
shows $f\ integrable\text{-}on\ cbox\ c\ d$
⟨*proof*⟩

lemma *integrable-subinterval-real*:
fixes $f :: real \Rightarrow 'a{::}banach$
assumes $f\ integrable\text{-}on\ \{a\ ..\ b\}$

and $\{c\ ..\ d\} \subseteq \{a\ ..\ b\}$
shows *f integrable-on* $\{c\ ..\ d\}$
⟨*proof*⟩

41.44 Combining adjacent intervals in 1 dimension.

lemma *has-integral-combine*:
fixes $a\ b\ c$ :: *real*
assumes $a \leq c$
and $c \leq b$
and (*f has-integral i*) $\{a\ ..\ c\}$
and (*f has-integral* (j::'*a*::*banach*)) $\{c\ ..\ b\}$
shows (*f has-integral* $(i + j)$) $\{a\ ..\ b\}$
⟨*proof*⟩

lemma *integral-combine*:
fixes *f* :: *real* ⇒ '*a*::*banach*
assumes $a \leq c$
and $c \leq b$
and *f integrable-on* $\{a\ ..\ b\}$
shows *integral* $\{a\ ..\ c\}$ *f* + *integral* $\{c\ ..\ b\}$ *f* = *integral* $\{a\ ..\ b\}$ *f*
⟨*proof*⟩

lemma *integrable-combine*:
fixes *f* :: *real* ⇒ '*a*::*banach*
assumes $a \leq c$
and $c \leq b$
and *f integrable-on* $\{a\ ..\ c\}$
and *f integrable-on* $\{c\ ..\ b\}$
shows *f integrable-on* $\{a\ ..\ b\}$
⟨*proof*⟩

41.45 Reduce integrability to "local" integrability.

lemma *integrable-on-little-subintervals*:
fixes *f* :: '*b*::*euclidean-space* ⇒ '*a*::*banach*
assumes $\forall\, x{\in}cbox\ a\ b.\ \exists\, d{>}0.\ \forall\, u\ v.\ x \in cbox\ u\ v \wedge cbox\ u\ v \subseteq ball\ x\ d \wedge cbox\ u\ v \subseteq cbox\ a\ b \longrightarrow$
f integrable-on cbox u v
shows *f integrable-on cbox a b*
⟨*proof*⟩

41.46 Second FCT or existence of antiderivative.

lemma *integrable-const*[*intro*]: $(\lambda x.\ c)$ *integrable-on cbox a b*
⟨*proof*⟩

lemma *integral-has-vector-derivative-continuous-at*:
fixes *f* :: *real* ⇒ '*a*::*banach*
assumes *f*: *f integrable-on* $\{a..b\}$

and x: $x \in \{a..b\}$
and fx: *continuous* (*at x within* $\{a..b\}$) f
shows $((\lambda u.\ \textit{integral}\ \{a..u\}\ f)\ \textit{has-vector-derivative}\ f\ x)\ (\textit{at}\ x\ \textit{within}\ \{a..b\})$
⟨*proof*⟩

lemma *integral-has-vector-derivative*:
fixes $f :: \textit{real} \Rightarrow 'a::\textit{banach}$
assumes *continuous-on* $\{a\ ..\ b\}\ f$
and $x \in \{a\ ..\ b\}$
shows $((\lambda u.\ \textit{integral}\ \{a\ ..\ u\}\ f)\ \textit{has-vector-derivative}\ f(x))\ (\textit{at}\ x\ \textit{within}\ \{a\ ..\ b\})$
⟨*proof*⟩

lemma *antiderivative-continuous*:
fixes $q\ b :: \textit{real}$
assumes *continuous-on* $\{a\ ..\ b\}\ f$
obtains g **where** $\forall x{\in}\{a\ ..\ b\}.\ (g\ \textit{has-vector-derivative}\ (f\ x::-::\textit{banach}))\ (\textit{at}\ x\ \textit{within}\ \{a\ ..\ b\})$
⟨*proof*⟩

41.47 Combined fundamental theorem of calculus.

lemma *antiderivative-integral-continuous*:
fixes $f :: \textit{real} \Rightarrow 'a::\textit{banach}$
assumes *continuous-on* $\{a\ ..\ b\}\ f$
obtains g **where** $\forall u{\in}\{a\ ..\ b\}.\ \forall v \in \{a\ ..\ b\}.\ u \le v \longrightarrow (f\ \textit{has-integral}\ (g\ v - g\ u))\ \{u\ ..\ v\}$
⟨*proof*⟩

41.48 General "twiddling" for interval-to-interval function image.

lemma *has-integral-twiddle*:
assumes $0 < r$
and $\forall x.\ h(g\ x) = x$
and $\forall x.\ g(h\ x) = x$
and $\forall x.\ \textit{continuous}\ (\textit{at}\ x)\ g$
and $\forall u\ v.\ \exists w\ z.\ g\ `\ \textit{cbox}\ u\ v = \textit{cbox}\ w\ z$
and $\forall u\ v.\ \exists w\ z.\ h\ `\ \textit{cbox}\ u\ v = \textit{cbox}\ w\ z$
and $\forall u\ v.\ \textit{content}(g\ `\ \textit{cbox}\ u\ v) = r * \textit{content}\ (\textit{cbox}\ u\ v)$
and $(f\ \textit{has-integral}\ i)\ (\textit{cbox}\ a\ b)$
shows $((\lambda x.\ f(g\ x))\ \textit{has-integral}\ (1\ /\ r) *_R i)\ (h\ `\ \textit{cbox}\ a\ b)$
⟨*proof*⟩

41.49 Special case of a basic affine transformation.

lemma *interval-image-affinity-interval*:
$\exists u\ v.\ (\lambda x.\ m *_R (x::'a::\textit{euclidean-space}) + c)\ `\ \textit{cbox}\ a\ b = \textit{cbox}\ u\ v$
⟨*proof*⟩

lemma *content-image-affinity-cbox*:

content((λx::'a::*euclidean-space*. *m* *$_R$ *x* + *c*) ‘ *cbox a b*) =
|*m*| ^ *DIM*('*a*) * *content* (*cbox a b*) (**is** *?l* = *?r*)
⟨*proof*⟩

lemma *has-integral-affinity*:
fixes *a* :: '*a*::*euclidean-space*
assumes (*f has-integral i*) (*cbox a b*)
and *m* ≠ *0*
shows ((λx. *f*(*m* *$_R$ *x* + *c*)) *has-integral* ((*1* / (|*m*| ^ *DIM*('*a*))) *$_R$ *i*)) ((λx. (*1* / *m*) *$_R$ *x* + −((*1* / *m*) *$_R$ *c*)) ‘ *cbox a b*)
⟨*proof*⟩

lemma *integrable-affinity*:
assumes *f integrable-on cbox a b*
and *m* ≠ *0*
shows (λx. *f*(*m* *$_R$ *x* + *c*)) *integrable-on* ((λx. (*1* / *m*) *$_R$ *x* + −((*1*/*m*) *$_R$ *c*)) ‘ *cbox a b*)
⟨*proof*⟩

lemmas *has-integral-affinity01* = *has-integral-affinity* [*of* - - *0 1*::*real*, *simplified*]

41.50 Special case of stretching coordinate axes separately.

lemma *image-stretch-interval*:
(λx. ∑ *k*∈*Basis*. (*m k* * (*x*•*k*)) *$_R$ *k*) ‘ *cbox a* (*b*::'*a*::*euclidean-space*) =
(*if* (*cbox a b*) = {} *then* {} *else*
cbox (∑ *k*∈*Basis*. (*min* (*m k* * (*a*•*k*)) (*m k* * (*b*•*k*))) *$_R$ *k*::'*a*)
(∑ *k*∈*Basis*. (*max* (*m k* * (*a*•*k*)) (*m k* * (*b*•*k*))) *$_R$ *k*))
⟨*proof*⟩

lemma *interval-image-stretch-interval*:
∃ *u v*. (λx. ∑ *k*∈*Basis*. (*m k* * (*x*•*k*))*$_R$ *k*) ‘ *cbox a* (*b*::'*a*::*euclidean-space*) = *cbox u* (*v*::'*a*::*euclidean-space*)
⟨*proof*⟩

lemma *content-image-stretch-interval*:
content ((λx::'*a*::*euclidean-space*. (∑ *k*∈*Basis*. (*m k* * (*x*•*k*))*$_R$ *k*)::'*a*) ‘ *cbox a b*) =
|*setprod m Basis*| * *content* (*cbox a b*)
⟨*proof*⟩

lemma *has-integral-stretch*:
fixes *f* :: '*a*::*euclidean-space* ⇒ '*b*::*real-normed-vector*
assumes (*f has-integral i*) (*cbox a b*)
and ∀ *k*∈*Basis*. *m k* ≠ *0*
shows ((λx. *f* (∑ *k*∈*Basis*. (*m k* * (*x*•*k*))*$_R$ *k*)) *has-integral*
((*1*/ |*setprod m Basis*|) *$_R$ *i*)) ((λx. (∑ *k*∈*Basis*. (*1* / *m k* * (*x*•*k*))*$_R$ *k*)) ‘ *cbox a b*)
⟨*proof*⟩

lemma *integrable-stretch*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::real\text{-}normed\text{-}vector$
assumes f *integrable-on cbox a b*
and $\forall k{\in}Basis.\ m\ k \neq 0$
shows $(\lambda x::'a.\ f\ (\sum k{\in}Basis.\ (m\ k * (x{\cdot}k)){*}_R\ k))$ *integrable-on*
$((\lambda x.\ \sum k{\in}Basis.\ (1\ /\ m\ k * (x{\cdot}k)){*}_R\ k)\ `\ cbox\ a\ b)$
⟨*proof*⟩

41.51 even more special cases.

lemma *uminus-interval-vector*[*simp*]:
fixes $a\ b :: 'a::euclidean\text{-}space$
shows $uminus\ `\ cbox\ a\ b = cbox\ (-b)\ (-a)$
⟨*proof*⟩

lemma *has-integral-reflect-lemma*[*intro*]:
assumes $(f\ has\text{-}integral\ i)\ (cbox\ a\ b)$
shows $((\lambda x.\ f(-x))\ has\text{-}integral\ i)\ (cbox\ (-b)\ (-a))$
⟨*proof*⟩

lemma *has-integral-reflect-lemma-real*[*intro*]:
assumes $(f\ has\text{-}integral\ i)\ \{a\ ..\ b::real\}$
shows $((\lambda x.\ f(-x))\ has\text{-}integral\ i)\ \{-b\ ..\ -a\}$
⟨*proof*⟩

lemma *has-integral-reflect*[*simp*]:
$((\lambda x.\ f\ (-x))\ has\text{-}integral\ i)\ (cbox\ (-b)\ (-a)) \longleftrightarrow (f\ has\text{-}integral\ i)\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *integrable-reflect*[*simp*]: $(\lambda x.\ f(-x))\ integrable\text{-}on\ cbox\ (-b)\ (-a) \longleftrightarrow f$ *integrable-on cbox a b*
⟨*proof*⟩

lemma *integrable-reflect-real*[*simp*]: $(\lambda x.\ f(-x))\ integrable\text{-}on\ \{-b\ ..\ -a\} \longleftrightarrow f$ *integrable-on* $\{a\ ..\ b::real\}$
⟨*proof*⟩

lemma *integral-reflect*[*simp*]: $integral\ (cbox\ (-b)\ (-a))\ (\lambda x.\ f\ (-x)) = integral\ (cbox\ a\ b)\ f$
⟨*proof*⟩

lemma *integral-reflect-real*[*simp*]: $integral\ \{-b\ ..\ -a\}\ (\lambda x.\ f\ (-x)) = integral\ \{a\ ..\ b::real\}\ f$
⟨*proof*⟩

41.52 Stronger form of FCT; quite a tedious proof.

lemma *bgauge-existence-lemma*: $(\forall x{\in}s.\ \exists d::real.\ 0 < d \wedge q\ d\ x) \longleftrightarrow (\forall x.\ \exists d{>}0.\ x{\in}s \longrightarrow q\ d\ x)$

⟨*proof*⟩

lemma *additive-tagged-division-1'*:
fixes f :: *real* ⇒ 'a::*real-normed-vector*
assumes $a \leq b$
and p *tagged-division-of* $\{a..b\}$
shows *setsum* $(\lambda(x,k).\ f\ (Sup\ k) - f(Inf\ k))\ p = f\ b - f\ a$
⟨*proof*⟩

lemma *split-minus*[*simp*]: $(\lambda(x,\ k).\ f\ x\ k)\ x - (\lambda(x,\ k).\ g\ x\ k)\ x = (\lambda(x,\ k).\ f\ x\ k - g\ x\ k)\ x$
⟨*proof*⟩

lemma *norm-triangle-le-sub*: *norm* x + *norm* $y \leq e \Longrightarrow$ *norm* $(x - y) \leq e$
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-interior*:
fixes f :: *real* ⇒ 'a::*real-normed-vector*
assumes $a \leq b$
and *continuous-on* $\{a\ ..\ b\}\ f$
and $\forall x{\in}\{a <..< b\}.\ (f\ has\text{-}vector\text{-}derivative\ f'(x))\ (at\ x)$
shows $(f'\ has\text{-}integral\ (f\ b - f\ a))\ \{a\ ..\ b\}$
⟨*proof*⟩

41.53 Stronger form with finite number of exceptional points.

lemma *fundamental-theorem-of-calculus-interior-strong*:
fixes f :: *real* ⇒ 'a::*banach*
assumes *finite* s
and $a \leq b$
and *continuous-on* $\{a\ ..\ b\}\ f$
and $\forall x{\in}\{a <..< b\} - s.\ (f\ has\text{-}vector\text{-}derivative\ f'(x))\ (at\ x)$
shows $(f'\ has\text{-}integral\ (f\ b - f\ a))\ \{a\ ..\ b\}$
⟨*proof*⟩

lemma *fundamental-theorem-of-calculus-strong*:
fixes f :: *real* ⇒ 'a::*banach*
assumes *finite* s
and $a \leq b$
and *continuous-on* $\{a\ ..\ b\}\ f$
and $\forall x{\in}\{a\ ..\ b\} - s.\ (f\ has\text{-}vector\text{-}derivative\ f'(x))\ (at\ x)$
shows $(f'\ has\text{-}integral\ (f\ b - f\ a))\ \{a\ ..\ b\}$
⟨*proof*⟩

lemma *indefinite-integral-continuous-left*:
fixes f:: *real* ⇒ 'a::*banach*
assumes f *integrable-on* $\{a\ ..\ b\}$
and $a < c$
and $c \leq b$

and $e > 0$
obtains d **where** $d > 0$
and $\forall t.\ c - d < t \wedge t \leq c \longrightarrow norm\ (integral\ \{a\ ..\ c\}\ f - integral\ \{a\ ..\ t\}\ f) < e$
⟨*proof*⟩

lemma *indefinite-integral-continuous-right*:
fixes $f :: real \Rightarrow 'a::banach$
assumes f *integrable-on* $\{a\ ..\ b\}$
and $a \leq c$
and $c < b$
and $e > 0$
obtains d **where** $0 < d$
and $\forall t.\ c \leq t \wedge t < c + d \longrightarrow norm\ (integral\ \{a\ ..\ c\}\ f - integral\ \{a\ ..\ t\}\ f) < e$
⟨*proof*⟩

lemma *indefinite-integral-continuous*:
fixes $f :: real \Rightarrow 'a::banach$
assumes f *integrable-on* $\{a\ ..\ b\}$
shows *continuous-on* $\{a\ ..\ b\}$ $(\lambda x.\ integral\ \{a\ ..\ x\}\ f)$
⟨*proof*⟩

41.54 This doesn't directly involve integration, but that gives an easy proof.

lemma *has-derivative-zero-unique-strong-interval*:
fixes $f :: real \Rightarrow 'a::banach$
assumes *finite* k
and *continuous-on* $\{a\ ..\ b\}$ f
and $f\ a = y$
and $\forall x \in (\{a\ ..\ b\} - k).\ (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ \{a\ ..\ b\})\ x \in \{a\ ..\ b\}$
shows $f\ x = y$
⟨*proof*⟩

41.55 Generalize a bit to any convex set.

lemma *has-derivative-zero-unique-strong-convex*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow 'b::banach$
assumes *convex* s
and *finite* k
and *continuous-on* s f
and $c \in s$
and $f\ c = y$
and $\forall x \in (s - k).\ (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ s)$
and $x \in s$
shows $f\ x = y$
⟨*proof*⟩

Also to any open connected set with finite set of exceptions. Could generalize to locally convex set with limpt-free set of exceptions.

lemma *has-derivative-zero-unique-strong-connected*:
fixes f :: *'a::euclidean-space* ⇒ *'b::banach*
assumes *connected s*
and *open s*
and *finite k*
and *continuous-on s f*
and $c \in s$
and $f\ c = y$
and $\forall x{\in}(s - k).\ (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ s)$
and $x{\in}s$
shows $f\ x = y$
⟨*proof*⟩

lemma *has-derivative-zero-connected-constant*:
fixes f :: *'a::euclidean-space* ⇒ *'b::banach*
assumes *connected s*
and *open s*
and *finite k*
and *continuous-on s f*
and $\forall x{\in}(s - k).\ (f\ has\text{-}derivative\ (\lambda h.\ 0))\ (at\ x\ within\ s)$
obtains *c* **where** $\bigwedge x.\ x \in s \Longrightarrow f(x) = c$
⟨*proof*⟩

41.56 Integrating characteristic function of an interval

lemma *has-integral-restrict-open-subinterval*:
fixes f :: *'a::euclidean-space* ⇒ *'b::banach*
assumes $(f\ has\text{-}integral\ i)\ (cbox\ c\ d)$
and $cbox\ c\ d \subseteq cbox\ a\ b$
shows $((\lambda x.\ if\ x \in box\ c\ d\ then\ f\ x\ else\ 0)\ has\text{-}integral\ i)\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *has-integral-restrict-closed-subinterval*:
fixes f :: *'a::euclidean-space* ⇒ *'b::banach*
assumes $(f\ has\text{-}integral\ i)\ (cbox\ c\ d)$
and $cbox\ c\ d \subseteq cbox\ a\ b$
shows $((\lambda x.\ if\ x \in cbox\ c\ d\ then\ f\ x\ else\ 0)\ has\text{-}integral\ i)\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *has-integral-restrict-closed-subintervals-eq*:
fixes f :: *'a::euclidean-space* ⇒ *'b::banach*
assumes $cbox\ c\ d \subseteq cbox\ a\ b$
shows $((\lambda x.\ if\ x \in cbox\ c\ d\ then\ f\ x\ else\ 0)\ has\text{-}integral\ i)\ (cbox\ a\ b) \longleftrightarrow (f\ has\text{-}integral\ i)\ (cbox\ c\ d)$
(**is** *?l* = *?r*)
⟨*proof*⟩

Hence we can apply the limit process uniformly to all integrals.

lemma *has-integral'*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
shows $(f\ has\text{-}integral\ i)\ s \longleftrightarrow$
$(\forall e>0.\ \exists B>0.\ \forall a\ b.\ ball\ 0\ B \subseteq cbox\ a\ b \longrightarrow$
$(\exists z.\ ((\lambda x.\ if\ x \in s\ then\ f(x)\ else\ 0)\ has\text{-}integral\ z)\ (cbox\ a\ b) \wedge norm(z - i) < e))$
$(\mathbf{is}\ ?l \longleftrightarrow (\forall e>0.\ ?r\ e))$
⟨*proof*⟩

lemma *has-integral-le*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $(f\ has\text{-}integral\ i)\ s$
and $(g\ has\text{-}integral\ j)\ s$
and $\forall x \in s.\ f\ x \leq g\ x$
shows $i \leq j$
⟨*proof*⟩

lemma *integral-le*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $f\ integrable\text{-}on\ s$
and $g\ integrable\text{-}on\ s$
and $\forall x \in s.\ f\ x \leq g\ x$
shows $integral\ s\ f \leq integral\ s\ g$
⟨*proof*⟩

lemma *has-integral-nonneg*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $(f\ has\text{-}integral\ i)\ s$
and $\forall x \in s.\ 0 \leq f\ x$
shows $0 \leq i$
⟨*proof*⟩

lemma *integral-nonneg*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $f\ integrable\text{-}on\ s$
and $\forall x \in s.\ 0 \leq f\ x$
shows $0 \leq integral\ s\ f$
⟨*proof*⟩

Hence a general restriction property.

lemma *has-integral-restrict*[*simp*]:
assumes $s \subseteq t$
shows $((\lambda x.\ if\ x \in s\ then\ f\ x\ else\ (0::'a::banach))\ has\text{-}integral\ i)\ t \longleftrightarrow (f\ has\text{-}integral\ i)\ s$
⟨*proof*⟩

lemma *has-integral-restrict-univ*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$

shows $((\lambda x.\ \textit{if}\ x \in s\ \textit{then}\ f\ x\ \textit{else}\ 0)\ \textit{has-integral}\ i)\ \textit{UNIV} \longleftrightarrow (f\ \textit{has-integral}\ i)\ s$
⟨*proof*⟩

lemma *has-integral-on-superset*:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
assumes $\forall x.\ x \notin s \longrightarrow f\ x = 0$
and $s \subseteq t$
and $(f\ \textit{has-integral}\ i)\ s$
shows $(f\ \textit{has-integral}\ i)\ t$
⟨*proof*⟩

lemma *integrable-on-superset*:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
assumes $\forall x.\ x \notin s \longrightarrow f\ x = 0$
and $s \subseteq t$
and $f\ \textit{integrable-on}\ s$
shows $f\ \textit{integrable-on}\ t$
⟨*proof*⟩

lemma *integral-restrict-univ*[*intro*]:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
shows $f\ \textit{integrable-on}\ s \Longrightarrow \textit{integral}\ \textit{UNIV}\ (\lambda x.\ \textit{if}\ x \in s\ \textit{then}\ f\ x\ \textit{else}\ 0) = \textit{integral}\ s\ f$
⟨*proof*⟩

lemma *integrable-restrict-univ*:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
shows $(\lambda x.\ \textit{if}\ x \in s\ \textit{then}\ f\ x\ \textit{else}\ 0)\ \textit{integrable-on}\ \textit{UNIV} \longleftrightarrow f\ \textit{integrable-on}\ s$
⟨*proof*⟩

lemma *negligible-on-intervals*: $\textit{negligible}\ s \longleftrightarrow (\forall a\ b.\ \textit{negligible}(s \cap \textit{cbox}\ a\ b))$ (**is** $?l \longleftrightarrow\ ?r$)
⟨*proof*⟩

lemma *has-integral-spike-set-eq*:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
assumes $\textit{negligible}\ ((s - t) \cup (t - s))$
shows $(f\ \textit{has-integral}\ y)\ s \longleftrightarrow (f\ \textit{has-integral}\ y)\ t$
⟨*proof*⟩

lemma *has-integral-spike-set*[*dest*]:
fixes $f :: 'n::\textit{euclidean-space} \Rightarrow 'a::\textit{banach}$
assumes $\textit{negligible}\ ((s - t) \cup (t - s))$
and $(f\ \textit{has-integral}\ y)\ s$
shows $(f\ \textit{has-integral}\ y)\ t$
⟨*proof*⟩

lemma *integrable-spike-set*[*dest*]:

fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'a$::*banach*
assumes *negligible* $((s - t) \cup (t - s))$
and f *integrable-on* s
shows f *integrable-on* t
⟨*proof*⟩

lemma *integrable-spike-set-eq*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'a$::*banach*
assumes *negligible* $((s - t) \cup (t - s))$
shows f *integrable-on* $s \longleftrightarrow f$ *integrable-on* t
⟨*proof*⟩

41.57 More lemmas that are useful later

lemma *has-integral-subset-component-le*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'m$::*euclidean-space*
assumes k: $k \in Basis$
and *as*: $s \subseteq t$ $(f$ *has-integral* $i)$ s $(f$ *has-integral* $j)$ t $\forall x{\in}t.\ 0 \leq f(x){\cdot}k$
shows $i{\cdot}k \leq j{\cdot}k$
⟨*proof*⟩

lemma *has-integral-subset-le*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ *real*
assumes $s \subseteq t$
and $(f$ *has-integral* $i)$ s
and $(f$ *has-integral* $j)$ t
and $\forall x{\in}t.\ 0 \leq f\ x$
shows $i \leq j$
⟨*proof*⟩

lemma *integral-subset-component-le*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'m$::*euclidean-space*
assumes $k \in Basis$
and $s \subseteq t$
and f *integrable-on* s
and f *integrable-on* t
and $\forall x \in t.\ 0 \leq f\ x \cdot k$
shows $(integral\ s\ f){\cdot}k \leq (integral\ t\ f){\cdot}k$
⟨*proof*⟩

lemma *integral-subset-le*:
fixes f :: $'n$::*euclidean-space* $\Rightarrow$ *real*
assumes $s \subseteq t$
and f *integrable-on* s
and f *integrable-on* t
and $\forall x \in t.\ 0 \leq f\ x$
shows $integral\ s\ f \leq integral\ t\ f$
⟨*proof*⟩

lemma *has-integral-alt'*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
shows $(f \text{ has-integral } i)\ s \longleftrightarrow (\forall a\ b.\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) \text{ integrable-on } cbox\ a\ b) \wedge$
$(\forall e{>}0.\ \exists B{>}0.\ \forall a\ b.\ ball\ 0\ B \subseteq cbox\ a\ b \longrightarrow$
$norm\ (integral\ (cbox\ a\ b)\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) - i) < e)$
(is *?l = ?r*)
⟨*proof*⟩

41.58 Continuity of the integral (for a 1-dimensional interval).

lemma *integrable-alt*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
shows $f \text{ integrable-on } s \longleftrightarrow$
$(\forall a\ b.\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) \text{ integrable-on } cbox\ a\ b) \wedge$
$(\forall e{>}0.\ \exists B{>}0.\ \forall a\ b\ c\ d.\ ball\ 0\ B \subseteq cbox\ a\ b \wedge ball\ 0\ B \subseteq cbox\ c\ d \longrightarrow$
$norm\ (integral\ (cbox\ a\ b)\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) -$
$integral\ (cbox\ c\ d)\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0)) < e)$
(is *?l = ?r*)
⟨*proof*⟩

lemma *integrable-altD*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $f \text{ integrable-on } s$
shows $\bigwedge a\ b.\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) \text{ integrable-on } cbox\ a\ b$
and $\bigwedge e.\ e > 0 \Longrightarrow \exists B{>}0.\ \forall a\ b\ c\ d.\ ball\ 0\ B \subseteq cbox\ a\ b \wedge ball\ 0\ B \subseteq cbox\ c\ d \longrightarrow$
$norm\ (integral\ (cbox\ a\ b)\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0) - integral\ (cbox\ c\ d)\ (\lambda x.\ \text{if } x \in s \text{ then } f\ x \text{ else } 0)) < e$
⟨*proof*⟩

lemma *integrable-on-subcbox*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $f \text{ integrable-on } s$
and $cbox\ a\ b \subseteq s$
shows $f \text{ integrable-on } cbox\ a\ b$
⟨*proof*⟩

41.59 A straddling criterion for integrability

lemma *integrable-straddle-interval*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $\forall e{>}0.\ \exists g\ \ h\ i\ j.\ (g \text{ has-integral } i)\ (cbox\ a\ b) \wedge (h \text{ has-integral } j)\ (cbox\ a\ b) \wedge$
$norm\ (i - j) < e \wedge (\forall x{\in}cbox\ a\ b.\ (g\ x) \le f\ x \wedge f\ x \le h\ x)$
shows $f \text{ integrable-on } cbox\ a\ b$
⟨*proof*⟩

lemma *integrable-straddle*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow real$
assumes $\forall e{>}0.\ \exists g\ h\ i\ j.\ (g\ has\text{-}integral\ i)\ s \wedge (h\ has\text{-}integral\ j)\ s \wedge$
$norm\ (i - j) < e \wedge (\forall x{\in}s.\ g\ x \leq f\ x \wedge f\ x \leq h\ x)$
shows $f\ integrable\text{-}on\ s$
⟨*proof*⟩

41.60 Adding integrals over several sets

lemma *has-integral-union*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $(f\ has\text{-}integral\ i)\ s$
and $(f\ has\text{-}integral\ j)\ t$
and $negligible\ (s \cap t)$
shows $(f\ has\text{-}integral\ (i + j))\ (s \cup t)$
⟨*proof*⟩

lemma *has-integral-unions*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes *finite* t
and $\forall s{\in}t.\ (f\ has\text{-}integral\ (i\ s))\ s$
and $\forall s{\in}t.\ \forall s'{\in}t.\ s \neq s' \longrightarrow negligible\ (s \cap s')$
shows $(f\ has\text{-}integral\ (setsum\ i\ t))\ (\bigcup t)$
⟨*proof*⟩

In particular adding integrals over a division, maybe not of an interval.

lemma *has-integral-combine-division*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $d\ division\text{-}of\ s$
and $\forall k{\in}d.\ (f\ has\text{-}integral\ (i\ k))\ k$
shows $(f\ has\text{-}integral\ (setsum\ i\ d))\ s$
⟨*proof*⟩

lemma *integral-combine-division-bottomup*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $d\ division\text{-}of\ s$
and $\forall k{\in}d.\ f\ integrable\text{-}on\ k$
shows $integral\ s\ f = setsum\ (\lambda i.\ integral\ i\ f)\ d$
⟨*proof*⟩

lemma *has-integral-combine-division-topdown*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes $f\ integrable\text{-}on\ s$
and $d\ division\text{-}of\ k$
and $k \subseteq s$
shows $(f\ has\text{-}integral\ (setsum\ (\lambda i.\ integral\ i\ f)\ d))\ k$
⟨*proof*⟩

lemma *integral-combine-division-topdown*:

fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes f *integrable-on* s
and d *division-of* s
shows *integral* s f = *setsum* $(\lambda i.$ *integral* i $f)$ d
⟨*proof*⟩

lemma *integrable-combine-division*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes d *division-of* s
and $\forall i \in d.$ f *integrable-on* i
shows f *integrable-on* s
⟨*proof*⟩

lemma *integrable-on-subdivision*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes d *division-of* i
and f *integrable-on* s
and $i \subseteq s$
shows f *integrable-on* i
⟨*proof*⟩

41.61 Also tagged divisions

lemma *has-integral-combine-tagged-division*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes p *tagged-division-of* s
and $\forall (x,k) \in p.$ $(f$ *has-integral* $(i\ k))\ k$
shows $(f$ *has-integral* $(setsum\ (\lambda(x,k).\ i\ k)\ p))\ s$
⟨*proof*⟩

lemma *integral-combine-tagged-division-bottomup*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes p *tagged-division-of* $(cbox\ a\ b)$
and $\forall (x,k) \in p.$ f *integrable-on* k
shows *integral* $(cbox\ a\ b)$ f = *setsum* $(\lambda(x,k).$ *integral* k $f)$ p
⟨*proof*⟩

lemma *has-integral-combine-tagged-division-topdown*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes f *integrable-on* $cbox\ a\ b$
and p *tagged-division-of* $(cbox\ a\ b)$
shows $(f$ *has-integral* $(setsum\ (\lambda(x,k).\ integral\ k\ f)\ p))\ (cbox\ a\ b)$
⟨*proof*⟩

lemma *integral-combine-tagged-division-topdown*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes f *integrable-on* $cbox\ a\ b$
and p *tagged-division-of* $(cbox\ a\ b)$
shows *integral* $(cbox\ a\ b)$ f = *setsum* $(\lambda(x,k).$ *integral* k $f)$ p

⟨*proof*⟩

41.62 Henstock's lemma

lemma *henstock-lemma-part1*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'a::banach$
assumes *f integrable-on cbox a b*
and $e > 0$
and *gauge d*
and $(\forall p.\ p\ tagged\text{-}division\text{-}of\ (cbox\ a\ b) \wedge d\ fine\ p \longrightarrow$
$norm\ (setsum\ (\lambda(x,k).\ content\ k *_R f\ x)\ p - integral(cbox\ a\ b)\ f) < e)$
and *p*: *p tagged-partial-division-of* (*cbox a b*) *d fine p*
shows $norm\ (setsum\ (\lambda(x,k).\ content\ k *_R f\ x - integral\ k\ f)\ p) \leq e$
(**is** $?x \leq e$)
⟨*proof*⟩

lemma *henstock-lemma-part2*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::euclidean\text{-}space$
assumes *f integrable-on cbox a b*
and $e > 0$
and *gauge d*
and $\forall p.\ p\ tagged\text{-}division\text{-}of\ (cbox\ a\ b) \wedge d\ fine\ p \longrightarrow$
$norm\ (setsum\ (\lambda(x,k).\ content\ k *_R f\ x)\ p - integral\ (cbox\ a\ b)\ f) < e$
and *p tagged-partial-division-of* (*cbox a b*)
and *d fine p*
shows $setsum\ (\lambda(x,k).\ norm\ (content\ k *_R f\ x - integral\ k\ f))\ p \leq 2 * real\ (DIM('n)) * e$
⟨*proof*⟩

lemma *henstock-lemma*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::euclidean\text{-}space$
assumes *f integrable-on cbox a b*
and $e > 0$
obtains *d* **where** *gauge d*
and $\forall p.\ p\ tagged\text{-}partial\text{-}division\text{-}of\ (cbox\ a\ b) \wedge d\ fine\ p \longrightarrow$
$setsum\ (\lambda(x,k).\ norm(content\ k *_R f\ x - integral\ k\ f))\ p < e$
⟨*proof*⟩

41.63 Geometric progression

FIXME: Should one or more of these theorems be moved to `~~/src/HOL/Set_Interval.thy`, alongside *geometric-sum*?

lemma *sum-gp-basic*:
fixes $x :: 'a::ring\text{-}1$
shows $(1 - x) * setsum\ (\lambda i.\ x\hat{}i)\ \{0 .. n\} = (1 - x\hat{}(Suc\ n))$
⟨*proof*⟩

lemma *sum-gp-multiplied*:
assumes *mn*: $m \leq n$

shows $((1::'a::\{field\}) - x) * setsum\ (op\ \hat{}\ x)\ \{m..n\} = x\hat{}m - x\hat{}\ Suc\ n$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *sum-gp*:
$setsum\ (op\ \hat{}\ (x::'a::\{field\}))\ \{m\ ..\ n\} =$
$(if\ n < m\ then\ 0$
$\ else\ if\ x = 1\ then\ of\text{-}nat\ ((n + 1) - m)$
$\ else\ (x\hat{}\ m - x\hat{}\ (Suc\ n)) / (1 - x))$
⟨*proof*⟩

lemma *sum-gp-offset*:
$setsum\ (op\ \hat{}\ (x::'a::\{field\}))\ \{m\ ..\ m+n\} =$
$(if\ x = 1\ then\ of\text{-}nat\ n + 1\ else\ x\hat{}m * (1 - x\hat{}Suc\ n) / (1 - x))$
⟨*proof*⟩

41.64 Monotone convergence (bounded interval first)

lemma *monotone-convergence-interval*:
fixes $f :: nat \Rightarrow 'n::euclidean\text{-}space \Rightarrow real$
assumes $\forall k.\ (f\ k)\ integrable\text{-}on\ cbox\ a\ b$
and $\forall k.\ \forall x{\in}cbox\ a\ b.(f\ k\ x) \leq f\ (Suc\ k)\ x$
and $\forall x{\in}cbox\ a\ b.\ ((\lambda k.\ f\ k\ x) \longrightarrow g\ x)\ sequentially$
and $bounded\ \{integral\ (cbox\ a\ b)\ (f\ k) \mid k\ .\ k \in UNIV\}$
shows $g\ integrable\text{-}on\ cbox\ a\ b \wedge ((\lambda k.\ integral\ (cbox\ a\ b)\ (f\ k)) \longrightarrow integral\ (cbox\ a\ b)\ g)\ sequentially$
⟨*proof*⟩

lemma *monotone-convergence-increasing*:
fixes $f :: nat \Rightarrow 'n::euclidean\text{-}space \Rightarrow real$
assumes $\forall k.\ (f\ k)\ integrable\text{-}on\ s$
and $\forall k.\ \forall x{\in}s.\ (f\ k\ x) \leq (f\ (Suc\ k)\ x)$
and $\forall x{\in}s.\ ((\lambda k.\ f\ k\ x) \longrightarrow g\ x)\ sequentially$
and $bounded\ \{integral\ s\ (f\ k) |\ k.\ True\}$
shows $g\ integrable\text{-}on\ s \wedge ((\lambda k.\ integral\ s\ (f\ k)) \longrightarrow integral\ s\ g)\ sequentially$
⟨*proof*⟩

lemma *has-integral-monotone-convergence-increasing*:
fixes $f :: nat \Rightarrow 'a::euclidean\text{-}space \Rightarrow real$
assumes f: $\bigwedge k.\ (f\ k\ has\text{-}integral\ x\ k)\ s$
assumes $\bigwedge k\ x.\ x \in s \Longrightarrow f\ k\ x \leq f\ (Suc\ k)\ x$
assumes $\bigwedge x.\ x \in s \Longrightarrow (\lambda k.\ f\ k\ x) \longrightarrow g\ x$
assumes $x \longrightarrow x'$
shows $(g\ has\text{-}integral\ x')\ s$
⟨*proof*⟩

lemma *monotone-convergence-decreasing*:
fixes $f :: nat \Rightarrow 'n::euclidean\text{-}space \Rightarrow real$
assumes $\forall k.\ (f\ k)\ integrable\text{-}on\ s$

and $\forall k.\ \forall x{\in}s.\ f\ (Suc\ k)\ x \leq f\ k\ x$
and $\forall x{\in}s.\ ((\lambda k.\ f\ k\ x) \longrightarrow g\ x)$ *sequentially*
and *bounded* $\{integral\ s\ (f\ k)|\ k.\ True\}$
shows g *integrable-on* $s \wedge ((\lambda k.\ integral\ s\ (f\ k)) \longrightarrow integral\ s\ g)$ *sequentially*
⟨*proof*⟩

41.65 Absolute integrability (this is the same as Lebesgue integrability)

definition *absolutely-integrable-on* (**infixr** *absolutely′-integrable′-on 46*)
where f *absolutely-integrable-on* $s \longleftrightarrow f$ *integrable-on* $s \wedge (\lambda x.\ (norm(f\ x)))$ *integrable-on* s

lemma *absolutely-integrable-onI*[*intro?*]:
f *integrable-on* $s \Longrightarrow$
$(\lambda x.\ (norm(f\ x)))$ *integrable-on* $s \Longrightarrow f$ *absolutely-integrable-on* s
⟨*proof*⟩

lemma *absolutely-integrable-onD*[*dest*]:
assumes f *absolutely-integrable-on* s
shows f *integrable-on* s
and $(\lambda x.\ norm\ (f\ x))$ *integrable-on* s
⟨*proof*⟩

lemma *integral-norm-bound-integral*:
fixes $f :: 'n{::}euclidean\text{-}space \Rightarrow 'a{::}banach$
assumes f *integrable-on* s
and g *integrable-on* s
and $\forall x{\in}s.\ norm\ (f\ x) \leq g\ x$
shows $norm\ (integral\ s\ f) \leq integral\ s\ g$
⟨*proof*⟩

lemma *integral-norm-bound-integral-component*:
fixes $f :: 'n{::}euclidean\text{-}space \Rightarrow 'a{::}banach$
fixes $g :: 'n \Rightarrow 'b{::}euclidean\text{-}space$
assumes f *integrable-on* s
and g *integrable-on* s
and $\forall x{\in}s.\ norm(f\ x) \leq (g\ x){\cdot}k$
shows $norm\ (integral\ s\ f) \leq (integral\ s\ g){\cdot}k$
⟨*proof*⟩

lemma *has-integral-norm-bound-integral-component*:
fixes $f :: 'n{::}euclidean\text{-}space \Rightarrow 'a{::}banach$
fixes $g :: 'n \Rightarrow 'b{::}euclidean\text{-}space$
assumes $(f\ has\text{-}integral\ i)\ s$
and $(g\ has\text{-}integral\ j)\ s$
and $\forall x{\in}s.\ norm\ (f\ x) \leq (g\ x){\cdot}k$

shows *norm i* ≤ *j*•*k*
⟨*proof*⟩

lemma *absolutely-integrable-le*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'a*::*banach*
assumes *f absolutely-integrable-on s*
shows *norm* (*integral s f*) ≤ *integral s* (λ*x*. *norm* (*f x*))
⟨*proof*⟩

lemma *absolutely-integrable-0*[*intro*]:
(λ*x*. *0*) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-cmul*[*intro*]:
f absolutely-integrable-on s ⟹
(λ*x*. *c* *$_R$ *f x*) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-neg*[*intro*]:
f absolutely-integrable-on s ⟹
(λ*x*. −*f*(*x*)) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-norm*[*intro*]:
f absolutely-integrable-on s ⟹
(λ*x*. *norm* (*f x*)) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-abs*[*intro*]:
f absolutely-integrable-on s ⟹
(λ*x*. |*f x*::*real*|) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-on-subinterval*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'a*::*banach*
shows *f absolutely-integrable-on s* ⟹
cbox a b ⊆ *s* ⟹ *f absolutely-integrable-on cbox a b*
⟨*proof*⟩

lemma *absolutely-integrable-bounded-variation*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'a*::*banach*
assumes *f absolutely-integrable-on UNIV*
obtains *B* **where** ∀ *d*. *d division-of* (⋃ *d*) ⟶ *setsum* (λ*k*. *norm*(*integral k f*))
d ≤ *B*
⟨*proof*⟩

lemma *helplemma*:
assumes *setsum* (λ*x*. *norm* (*f x* − *g x*)) *s* < *e*
and *finite s*

shows $|setsum\ (\lambda x.\ norm(f\ x))\ s - setsum\ (\lambda x.\ norm(g\ x))\ s| < e$
⟨*proof*⟩

lemma *bounded-variation-absolutely-integrable-interval*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *f*: *f integrable-on cbox a b*
and ∗: $\forall d.\ d\ division\text{-}of\ (cbox\ a\ b) \longrightarrow setsum\ (\lambda k.\ norm(integral\ k\ f))\ d \leq B$
shows *f absolutely-integrable-on cbox a b*
⟨*proof*⟩

lemma *bounded-variation-absolutely-integrable*:
fixes $f :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *f integrable-on UNIV*
and $\forall d.\ d\ division\text{-}of\ (\bigcup d) \longrightarrow setsum\ (\lambda k.\ norm\ (integral\ k\ f))\ d \leq B$
shows *f absolutely-integrable-on UNIV*
⟨*proof*⟩

lemma *absolutely-integrable-restrict-univ*:
$(\lambda x.\ if\ x \in s\ then\ f\ x\ else\ (0::'a::banach))$ *absolutely-integrable-on UNIV* ⟷
f absolutely-integrable-on s
⟨*proof*⟩

lemma *absolutely-integrable-add*[*intro*]:
fixes $f\ g :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *f absolutely-integrable-on s*
and *g absolutely-integrable-on s*
shows $(\lambda x.\ f\ x + g\ x)$ *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-sub*[*intro*]:
fixes $f\ g :: 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *f absolutely-integrable-on s*
and *g absolutely-integrable-on s*
shows $(\lambda x.\ f\ x - g\ x)$ *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-linear*:
fixes $f :: 'm::euclidean\text{-}space \Rightarrow 'n::euclidean\text{-}space$
and $h :: 'n::euclidean\text{-}space \Rightarrow 'p::euclidean\text{-}space$
assumes *f absolutely-integrable-on s*
and *bounded-linear h*
shows $(h \circ f)$ *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-setsum*:
fixes $f :: 'a \Rightarrow 'n::euclidean\text{-}space \Rightarrow 'm::euclidean\text{-}space$
assumes *finite t*
and $\bigwedge a.\ a \in t \Longrightarrow (f\ a)$ *absolutely-integrable-on s*
shows $(\lambda x.\ setsum\ (\lambda a.\ f\ a\ x)\ t)$ *absolutely-integrable-on s*

⟨*proof*⟩

lemma *absolutely-integrable-vector-abs*:
fixes *f* :: *'a*::*euclidean-space* => *'b*::*euclidean-space*
and *T* :: *'c*::*euclidean-space* ⇒ *'b*
assumes *f*: *f absolutely-integrable-on s*
shows (λ*x*. (∑ *i*∈*Basis*. |*f x*•*T i*| ∗*R i*)) *absolutely-integrable-on s*
(**is** *?Tf absolutely-integrable-on s*)
⟨*proof*⟩

lemma *absolutely-integrable-max*:
fixes *f g* :: *'m*::*euclidean-space* ⇒ *'n*::*euclidean-space*
assumes *f absolutely-integrable-on s*
and *g absolutely-integrable-on s*
shows (λ*x*. (∑ *i*∈*Basis*. *max* (*f*(*x*)•*i*) (*g*(*x*)•*i*) ∗*R i*)::*'n*) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-min*:
fixes *f g*::*'m*::*euclidean-space* ⇒ *'n*::*euclidean-space*
assumes *f absolutely-integrable-on s*
and *g absolutely-integrable-on s*
shows (λ*x*. (∑ *i*∈*Basis*. *min* (*f*(*x*)•*i*) (*g*(*x*)•*i*) ∗*R i*)::*'n*) *absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-abs-eq*:
fixes *f*::*'n*::*euclidean-space* ⇒ *'m*::*euclidean-space*
shows *f absolutely-integrable-on s* ⟷ *f integrable-on s* ∧
(λ*x*. (∑ *i*∈*Basis*. |*f x*•*i*| ∗*R i*)::*'m*) *integrable-on s*
(**is** *?l* = *?r*)
⟨*proof*⟩

lemma *nonnegative-absolutely-integrable*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'m*::*euclidean-space*
assumes ∀ *x*∈*s*. ∀ *i*∈*Basis*. *0* ≤ *f x* • *i*
and *f integrable-on s*
shows *f absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-integrable-bound*:
fixes *f* :: *'n*::*euclidean-space* ⇒ *'m*::*euclidean-space*
assumes ∀ *x*∈*s*. *norm* (*f x*) ≤ *g x*
and *f integrable-on s*
and *g integrable-on s*
shows *f absolutely-integrable-on s*
⟨*proof*⟩

lemma *absolutely-integrable-absolutely-integrable-bound*:

fixes f :: $'n$::*euclidean-space* $\Rightarrow$ $'m$::*euclidean-space*
and g :: $'n$::*euclidean-space* $\Rightarrow$ $'p$::*euclidean-space*
assumes $\forall x{\in}s.$ *norm* $(f\ x) \leq$ *norm* $(g\ x)$
and f *integrable-on* s
and g *absolutely-integrable-on* s
shows f *absolutely-integrable-on* s
⟨*proof*⟩

lemma *absolutely-integrable-inf-real*:
assumes *finite* k
and $k \neq \{\}$
and $\forall i{\in}k.$ $(\lambda x.\ (fs\ x\ i)$::*real*$)$ *absolutely-integrable-on* s
shows $(\lambda x.\ (Inf\ ((fs\ x)\ `\ k)))$ *absolutely-integrable-on* s
⟨*proof*⟩

lemma *absolutely-integrable-sup-real*:
assumes *finite* k
and $k \neq \{\}$
and $\forall i{\in}k.$ $(\lambda x.\ (fs\ x\ i)$::*real*$)$ *absolutely-integrable-on* s
shows $(\lambda x.\ (Sup\ ((fs\ x)\ `\ k)))$ *absolutely-integrable-on* s
⟨*proof*⟩

41.66 differentiation under the integral sign

lemma *tube-lemma*:
assumes *compact* K
assumes *open* W
assumes $\{x0\} \times K \subseteq W$
shows $\exists X0.\ x0 \in X0 \wedge$ *open* $X0 \wedge X0 \times K \subseteq W$
⟨*proof*⟩

lemma *continuous-on-prod-compactE*:
fixes *fx*::$'a$::*topological-space* $\times$ $'b$::*topological-space* $\Rightarrow$ $'c$::*metric-space*
and *e*::*real*
assumes *cont-fx*: *continuous-on* $(U \times C)$ *fx*
assumes *compact* C
assumes [*intro*]: $x0 \in U$
notes [*continuous-intros*] = *continuous-on-compose2*[*OF cont-fx*]
assumes $e > 0$
obtains $X0$ **where** $x0 \in X0$ *open* $X0$
$\forall x{\in}X0 \cap U.\ \forall t \in C.$ *dist* $(fx\ (x,\ t))\ (fx\ (x0,\ t)) \leq e$
⟨*proof*⟩

lemma *integral-continuous-on-param*:
fixes f::$'a$::*topological-space* $\Rightarrow$ $'b$::*euclidean-space* $\Rightarrow$ $'c$::*banach*
assumes *cont-fx*: *continuous-on* $(U \times cbox\ a\ b)\ (\lambda(x,\ t).\ f\ x\ t)$
shows *continuous-on* U $(\lambda x.\ integral\ (cbox\ a\ b)\ (f\ x))$
⟨*proof*⟩

lemma *eventually-closed-segment*:
fixes *x0*::*'a*::*real-normed-vector*
assumes *open X0 x0* ∈ *X0*
shows ∀$_F$ *x in at x0 within U*. *closed-segment x0 x* ⊆ *X0*
⟨*proof*⟩

lemma *leibniz-rule*:
fixes *f*::*'a*::*banach* ⇒ *'b*::*euclidean-space* ⇒ *'c*::*banach*
assumes *fx*: ⋀*x t*. *x* ∈ *U* ⟹ *t* ∈ *cbox a b* ⟹
((λ*x*. *f x t*) *has-derivative blinfun-apply* (*fx x t*)) (*at x within U*)
assumes *integrable-f2*: ⋀*x*. *x* ∈ *U* ⟹ *f x integrable-on cbox a b*
assumes *cont-fx*: *continuous-on* (*U* × (*cbox a b*)) (λ(*x*, *t*). *fx x t*)
assumes [*intro*]: *x0* ∈ *U*
assumes *convex U*
shows
((λ*x*. *integral* (*cbox a b*) (*f x*)) *has-derivative integral* (*cbox a b*) (*fx x0*)) (*at x0 within U*)
(**is** (*?F has-derivative ?dF*) -)
⟨*proof*⟩

lemma
has-vector-derivative-eq-has-derivative-blinfun:
(*f has-vector-derivative f'*) (*at x within U*) ⟷
(*f has-derivative blinfun-scaleR-left f'*) (*at x within U*)
⟨*proof*⟩

lemma *leibniz-rule-vector-derivative*:
fixes *f*::*real* ⇒ *'b*::*euclidean-space* ⇒ *'c*::*banach*
assumes *fx*: ⋀*x t*. *x* ∈ *U* ⟹ *t* ∈ *cbox a b* ⟹
((λ*x*. *f x t*) *has-vector-derivative* (*fx x t*)) (*at x within U*)
assumes *integrable-f2*: ⋀*x*. *x* ∈ *U* ⟹ (*f x*) *integrable-on cbox a b*
assumes *cont-fx*: *continuous-on* (*U* × *cbox a b*) (λ(*x*, *t*). *fx x t*)
assumes *U*: *x0* ∈ *U convex U*
shows ((λ*x*. *integral* (*cbox a b*) (*f x*)) *has-vector-derivative integral* (*cbox a b*) (*fx x0*))
(*at x0 within U*)
⟨*proof*⟩

lemma
has-field-derivative-eq-has-derivative-blinfun:
(*f has-field-derivative f'*) (*at x within U*) ⟷ (*f has-derivative blinfun-mult-right f'*) (*at x within U*)
⟨*proof*⟩

lemma *leibniz-rule-field-derivative*:
fixes *f*::*'a*::{*real-normed-field*, *banach*} ⇒ *'b*::*euclidean-space* ⇒ *'a*
assumes *fx*: ⋀*x t*. *x* ∈ *U* ⟹ *t* ∈ *cbox a b* ⟹ ((λ*x*. *f x t*) *has-field-derivative fx x t*) (*at x within U*)
assumes *integrable-f2*: ⋀*x*. *x* ∈ *U* ⟹ (*f x*) *integrable-on cbox a b*

assumes *cont-fx*: *continuous-on* $(U \times (cbox\ a\ b))\ (\lambda(x,\ t).\ fx\ x\ t)$
assumes U: $x0 \in U$ *convex* U
shows $((\lambda x.\ integral\ (cbox\ a\ b)\ (f\ x))$ *has-field-derivative integral* $(cbox\ a\ b)\ (fx\ x0))\ (at\ x0\ within\ U)$
⟨*proof*⟩

41.67 Exchange uniform limit and integral

lemma
uniform-limit-integral:
fixes $f::'a \Rightarrow 'b::euclidean\text{-}space \Rightarrow 'c::banach$
assumes u: *uniform-limit* $(cbox\ a\ b)\ f\ g\ F$
assumes c: $\bigwedge n.$ *continuous-on* $(cbox\ a\ b)\ (f\ n)$
assumes [*simp*]: $F \neq bot$
obtains $I\ J$ **where**
$\bigwedge n.\ (f\ n\ has\text{-}integral\ I\ n)\ (cbox\ a\ b)$
$(g\ has\text{-}integral\ J)\ (cbox\ a\ b)$
$(I \longrightarrow J)\ F$
⟨*proof*⟩

41.68 Dominated convergence

lemma *dominated-convergence*:
fixes $f :: nat \Rightarrow 'n::euclidean\text{-}space \Rightarrow real$
assumes $\bigwedge k.\ (f\ k)$ *integrable-on* s h *integrable-on* s
and $\bigwedge k.\ \forall x \in s.\ norm\ (f\ k\ x) \leq h\ x$
and $\forall x \in s.\ ((\lambda k.\ f\ k\ x) \longrightarrow g\ x)$ *sequentially*
shows g *integrable-on* s
and $((\lambda k.\ integral\ s\ (f\ k)) \longrightarrow integral\ s\ g)$ *sequentially*
⟨*proof*⟩

lemma *has-integral-dominated-convergence*:
fixes $f :: nat \Rightarrow 'n::euclidean\text{-}space \Rightarrow real$
assumes $\bigwedge k.\ (f\ k\ has\text{-}integral\ y\ k)\ s$ h *integrable-on* s
$\bigwedge k.\ \forall x{\in}s.\ norm\ (f\ k\ x) \leq h\ x\ \forall x{\in}s.\ (\lambda k.\ f\ k\ x) \longrightarrow g\ x$
and x: $y \longrightarrow x$
shows $(g\ has\text{-}integral\ x)\ s$
⟨*proof*⟩

41.69 Compute a double integral using iterated integrals and switching the order of integration

lemma *setcomp-dot1*: $\{z.\ P\ (z \cdot (i,0))\} = \{(x,y).\ P(x \cdot i)\}$
⟨*proof*⟩

lemma *setcomp-dot2*: $\{z.\ P\ (z \cdot (0,i))\} = \{(x,y).\ P(y \cdot i)\}$
⟨*proof*⟩

lemma *Sigma-Int-Paircomp1*: $(Sigma\ A\ B) \cap \{(x,\ y).\ P\ x\} = Sigma\ (A \cap \{x.\ P\ x\})\ B$

⟨*proof*⟩

lemma *Sigma-Int-Paircomp2*: (*Sigma A B*) ∩ {(*x*, *y*). *P y*} = *Sigma A* (λ*z*. *B z* ∩ {*y*. *P y*})
⟨*proof*⟩

lemma *continuous-on-imp-integrable-on-Pair1*:
fixes *f* :: - ⇒ *'b*::*banach*
assumes *con*: *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) *f* **and** *x*: *x* ∈ *cbox a b*
shows (λ*y*. *f* (*x*, *y*)) *integrable-on* (*cbox c d*)
⟨*proof*⟩

lemma *integral-integrable-2dim*:
fixes *f* :: (*'a*::*euclidean-space* ∗ *'b*::*euclidean-space*) ⇒ *'c*::*banach*
assumes *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) *f*
shows (λ*x*. *integral* (*cbox c d*) (λ*y*. *f* (*x*,*y*))) *integrable-on cbox a b*
⟨*proof*⟩

lemma *norm-diff2*: ⟦*y* = *y1* + *y2*; *x* = *x1* + *x2*; *e* = *e1* + *e2*; *norm*(*y1* − *x1*) ≤ *e1*; *norm*(*y2* − *x2*) ≤ *e2*⟧
⟹ *norm*(*y* − *x*) ≤ *e*
⟨*proof*⟩

lemma *integral-split*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::{*real-normed-vector*,*complete-space*}
assumes *f*: *f integrable-on* (*cbox a b*)
and *k*: *k* ∈ *Basis*
shows *integral* (*cbox a b*) *f* =
integral (*cbox a b* ∩ {*x*. *x*∙*k* ≤ *c*}) *f* +
integral (*cbox a b* ∩ {*x*. *x*∙*k* ≥ *c*}) *f*
⟨*proof*⟩

lemma *integral-swap-operative*:
fixes *f* :: (*'a*::*euclidean-space* ∗ *'b*::*euclidean-space*) ⇒ *'c*::*banach*
assumes *f*: *continuous-on s f* **and** *e*: *0* < *e*
shows *operative*(*op* ∧)
(λ*k*. ∀ *a b c d*.
cbox (*a*,*c*) (*b*,*d*) ⊆ *k* ∧ *cbox* (*a*,*c*) (*b*,*d*) ⊆ *s*
⟶ *norm*(*integral* (*cbox* (*a*,*c*) (*b*,*d*)) *f* −
integral (*cbox a b*) (λ*x*. *integral* (*cbox c d*) (λ*y*. *f*((*x*,*y*)))))
≤ *e* ∗ *content* (*cbox* (*a*,*c*) (*b*,*d*)))
⟨*proof*⟩

lemma *integral-Pair-const*:
integral (*cbox* (*a*,*c*) (*b*,*d*)) (λ*x*. *k*) =
integral (*cbox a b*) (λ*x*. *integral* (*cbox c d*) (λ*y*. *k*))
⟨*proof*⟩

lemma *norm-minus2*: *norm* (*x1*−*x2*, *y1*−*y2*) = *norm* (*x2*−*x1*, *y2*−*y1*)

⟨*proof*⟩

lemma *integral-prod-continuous*:
fixes *f* :: ('*a*::*euclidean-space* ∗ '*b*::*euclidean-space*) ⇒ '*c*::*banach*
assumes *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) *f* (**is** *continuous-on ?CBOX f*)
shows *integral* (*cbox* (*a*,*c*) (*b*,*d*)) *f* = *integral* (*cbox a b*) (λ*x*. *integral* (*cbox c d*) (λ*y*. *f*(*x*,*y*)))
⟨*proof*⟩

lemma *swap-continuous*:
assumes *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) (λ(*x*,*y*). *f x y*)
shows *continuous-on* (*cbox* (*c*,*a*) (*d*,*b*)) (λ(*x*, *y*). *f y x*)
⟨*proof*⟩

lemma *integral-swap-2dim*:
fixes *f* :: ['*a*::*euclidean-space*, '*b*::*euclidean-space*] ⇒ '*c*::*banach*
assumes *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) (λ(*x*,*y*). *f x y*)
shows *integral* (*cbox* (*a*, *c*) (*b*, *d*)) (λ(*x*, *y*). *f x y*) = *integral* (*cbox* (*c*, *a*) (*d*, *b*)) (λ(*x*, *y*). *f y x*)
⟨*proof*⟩

theorem *integral-swap-continuous*:
fixes *f* :: ['*a*::*euclidean-space*, '*b*::*euclidean-space*] ⇒ '*c*::*banach*
assumes *continuous-on* (*cbox* (*a*,*c*) (*b*,*d*)) (λ(*x*,*y*). *f x y*)
shows *integral* (*cbox a b*) (λ*x*. *integral* (*cbox c d*) (*f x*)) =
integral (*cbox c d*) (λ*y*. *integral* (*cbox a b*) (λ*x*. *f x y*))
⟨*proof*⟩

end

42 Instantiates the finite Cartesian product of Euclidean spaces as a Euclidean space.

theory *Cartesian-Euclidean-Space*
imports *Finite-Cartesian-Product Integration*
begin

lemma *delta-mult-idempotent*:
(*if k*=*a then 1 else* (*0*::'*a*::*semiring-1*)) ∗ (*if k*=*a then 1 else 0*) = (*if k*=*a then 1 else 0*)
⟨*proof*⟩

lemma *setsum-UNIV-sum*:
fixes *g* :: '*a*::*finite* + '*b*::*finite* ⇒ -
shows (∑ *x*∈*UNIV*. *g x*) = (∑ *x*∈*UNIV*. *g* (*Inl x*)) + (∑ *x*∈*UNIV*. *g* (*Inr x*))
⟨*proof*⟩

lemma *setsum-mult-product*:

*setsum h {..<A * B :: nat} = (∑ i∈{..<A}. ∑ j∈{..<B}. h (j + i * B))*
⟨*proof*⟩

42.1 Basic componentwise operations on vectors.

instantiation *vec* :: (*times*, *finite*) *times*
begin

definition *op* * ≡ (λ *x y*. (χ *i*. (*x*$*i*) * (*y*$*i*)))
instance ⟨*proof*⟩

end

instantiation *vec* :: (*one*, *finite*) *one*
begin

definition *1* ≡ (χ *i*. *1*)
instance ⟨*proof*⟩

end

instantiation *vec* :: (*ord*, *finite*) *ord*
begin

definition *x* ≤ *y* ⟷ (∀ *i*. *x*$*i* ≤ *y*$*i*)
definition *x* < (*y*::'*a*^'*b*) ⟷ *x* ≤ *y* ∧ ¬ *y* ≤ *x*
instance ⟨*proof*⟩

end

The ordering on one-dimensional vectors is linear.

class *cart-one* =
assumes *UNIV-one*: *card* (*UNIV* :: '*a set*) = *Suc 0*
begin

subclass *finite*
⟨*proof*⟩

end

instance *vec*:: (*order*, *finite*) *order*
⟨*proof*⟩

instance *vec* :: (*linorder*, *cart-one*) *linorder*
⟨*proof*⟩

Constant Vectors

definition *vec x* = (χ *i*. *x*)

lemma *interval-cbox-cart*: $\{a::real\hat{}'n..b\} = cbox\ a\ b$
⟨*proof*⟩

Also the scalar-vector multiplication.

definition *vector-scalar-mult*:: $'a::times \Rightarrow 'a\ \hat{}\ 'n \Rightarrow 'a\ \hat{}\ 'n$ (**infixl** $*s$ *70*)
where $c *s\ x = (\chi\ i.\ c * (x\$i))$

42.2 A naive proof procedure to lift really trivial arithmetic stuff from the basis of the vector space.

lemma *setsum-cong-aux*:
$(\bigwedge x.\ x \in A \Longrightarrow f\ x = g\ x) \Longrightarrow setsum\ f\ A = setsum\ g\ A$
⟨*proof*⟩

hide-fact (**open**) *setsum-cong-aux*

⟨*ML*⟩

lemma *vec-0*[*simp*]: $vec\ 0 = 0$ ⟨*proof*⟩
lemma *vec-1*[*simp*]: $vec\ 1 = 1$ ⟨*proof*⟩

lemma *vec-inj*[*simp*]: $vec\ x = vec\ y \longleftrightarrow x = y$ ⟨*proof*⟩

lemma *vec-in-image-vec*: $vec\ x \in (vec\ `\ S) \longleftrightarrow x \in S$ ⟨*proof*⟩

lemma *vec-add*: $vec(x + y) = vec\ x + vec\ y$ ⟨*proof*⟩
lemma *vec-sub*: $vec(x - y) = vec\ x - vec\ y$ ⟨*proof*⟩
lemma *vec-cmul*: $vec(c * x) = c *s\ vec\ x$ ⟨*proof*⟩
lemma *vec-neg*: $vec(- x) = - vec\ x$ ⟨*proof*⟩

lemma *vec-setsum*:
assumes *finite S*
shows $vec(setsum\ f\ S) = setsum\ (vec \circ f)\ S$
⟨*proof*⟩

Obvious "component-pushing".

lemma *vec-component* [*simp*]: $vec\ x\ \$\ i = x$
⟨*proof*⟩

lemma *vector-mult-component* [*simp*]: $(x * y)\$i = x\$i * y\$i$
⟨*proof*⟩

lemma *vector-smult-component* [*simp*]: $(c *s\ y)\$i = c * (y\$i)$
⟨*proof*⟩

lemma *cond-component*: $(if\ b\ then\ x\ else\ y)\$i = (if\ b\ then\ x\$i\ else\ y\$i)$ ⟨*proof*⟩

lemmas *vector-component* =
vec-component vector-add-component vector-mult-component

vector-smult-component vector-minus-component vector-uminus-component vector-scaleR-component cond-component

42.3 Some frequently useful arithmetic lemmas over vectors.

instance *vec* :: (*semigroup-mult*, *finite*) *semigroup-mult*
⟨*proof*⟩

instance *vec* :: (*monoid-mult*, *finite*) *monoid-mult*
⟨*proof*⟩

instance *vec* :: (*ab-semigroup-mult*, *finite*) *ab-semigroup-mult*
⟨*proof*⟩

instance *vec* :: (*comm-monoid-mult*, *finite*) *comm-monoid-mult*
⟨*proof*⟩

instance *vec* :: (*semiring*, *finite*) *semiring*
⟨*proof*⟩

instance *vec* :: (*semiring-0*, *finite*) *semiring-0*
⟨*proof*⟩
instance *vec* :: (*semiring-1*, *finite*) *semiring-1*
⟨*proof*⟩
instance *vec* :: (*comm-semiring*, *finite*) *comm-semiring*
⟨*proof*⟩

instance *vec* :: (*comm-semiring-0*, *finite*) *comm-semiring-0* ⟨*proof*⟩
instance *vec* :: (*cancel-comm-monoid-add*, *finite*) *cancel-comm-monoid-add* ⟨*proof*⟩
instance *vec* :: (*semiring-0-cancel*, *finite*) *semiring-0-cancel* ⟨*proof*⟩
instance *vec* :: (*comm-semiring-0-cancel*, *finite*) *comm-semiring-0-cancel* ⟨*proof*⟩
instance *vec* :: (*ring*, *finite*) *ring* ⟨*proof*⟩
instance *vec* :: (*semiring-1-cancel*, *finite*) *semiring-1-cancel* ⟨*proof*⟩
instance *vec* :: (*comm-semiring-1*, *finite*) *comm-semiring-1* ⟨*proof*⟩

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

instance *vec* :: (*real-algebra*, *finite*) *real-algebra*
⟨*proof*⟩

instance *vec* :: (*real-algebra-1*, *finite*) *real-algebra-1* ⟨*proof*⟩

lemma *of-nat-index*: (*of-nat n* :: *'a*::*semiring-1* ^ *'n*)$*i* = *of-nat n*
⟨*proof*⟩

lemma *one-index* [*simp*]: (*1* :: *'a* :: *one* ^ *'n*) $ *i* = *1*
⟨*proof*⟩

lemma *neg-one-index* [*simp*]: (*− 1* :: *'a* :: {*one*, *uminus*} ^ *'n*) $ *i* = *− 1*

⟨*proof*⟩

instance *vec* :: (*semiring-char-0*, *finite*) *semiring-char-0*
⟨*proof*⟩

instance *vec* :: (*numeral*, *finite*) *numeral* ⟨*proof*⟩
instance *vec* :: (*semiring-numeral*, *finite*) *semiring-numeral* ⟨*proof*⟩

lemma *numeral-index* [*simp*]: *numeral w $ i = numeral w*
⟨*proof*⟩

lemma *neg-numeral-index* [*simp*]: *− numeral w $ i = − numeral w*
⟨*proof*⟩

instance *vec* :: (*comm-ring-1*, *finite*) *comm-ring-1* ⟨*proof*⟩
instance *vec* :: (*ring-char-0*, *finite*) *ring-char-0* ⟨*proof*⟩

lemma *vector-smult-assoc*: *a ∗s (b ∗s x) = ((a::'a::semigroup-mult) ∗ b) ∗s x*
⟨*proof*⟩
lemma *vector-sadd-rdistrib*: *((a::'a::semiring) + b) ∗s x = a ∗s x + b ∗s x*
⟨*proof*⟩
lemma *vector-add-ldistrib*: *(c::'a::semiring) ∗s (x + y) = c ∗s x + c ∗s y*
⟨*proof*⟩
lemma *vector-smult-lzero*[*simp*]: *(0::'a::mult-zero) ∗s x = 0* ⟨*proof*⟩
lemma *vector-smult-lid*[*simp*]: *(1::'a::monoid-mult) ∗s x = x* ⟨*proof*⟩
lemma *vector-ssub-ldistrib*: *(c::'a::ring) ∗s (x − y) = c ∗s x − c ∗s y*
⟨*proof*⟩
lemma *vector-smult-rneg*: *(c::'a::ring) ∗s −x = −(c ∗s x)* ⟨*proof*⟩
lemma *vector-smult-lneg*: *− (c::'a::ring) ∗s x = −(c ∗s x)* ⟨*proof*⟩
lemma *vector-sneg-minus1*: *−x = (−1::'a::ring-1) ∗s x* ⟨*proof*⟩
lemma *vector-smult-rzero*[*simp*]: *c ∗s 0 = (0::'a::mult-zero ^ 'n)* ⟨*proof*⟩
lemma *vector-sub-rdistrib*: *((a::'a::ring) − b) ∗s x = a ∗s x − b ∗s x*
⟨*proof*⟩

lemma *vec-eq*[*simp*]: *(vec m = vec n) ⟷ (m = n)*
⟨*proof*⟩

lemma *norm-eq-0-imp*: *norm x = 0 ==> x = (0::real ^'n)* ⟨*proof*⟩
lemma *vector-mul-eq-0*[*simp*]: *(a ∗s x = 0) ⟷ a = (0::'a::idom) ∨ x = 0*
⟨*proof*⟩
lemma *vector-mul-lcancel*[*simp*]: *a ∗s x = a ∗s y ⟷ a = (0::real) ∨ x = y*
⟨*proof*⟩
lemma *vector-mul-rcancel*[*simp*]: *a ∗s x = b ∗s x ⟷ (a::real) = b ∨ x = 0*
⟨*proof*⟩
lemma *vector-mul-lcancel-imp*: *a ≠ (0::real) ==> a ∗s x = a ∗s y ==> (x = y)*
⟨*proof*⟩
lemma *vector-mul-rcancel-imp*: *x ≠ 0 ⟹ (a::real) ∗s x = b ∗s x ==> a = b*
⟨*proof*⟩

lemma *component-le-norm-cart*: $|x\$i| <= norm\ x$
⟨*proof*⟩

lemma *norm-bound-component-le-cart*: $norm\ x <= e ==> |x\$i| <= e$
⟨*proof*⟩

lemma *norm-bound-component-lt-cart*: $norm\ x < e ==> |x\$i| < e$
⟨*proof*⟩

lemma *norm-le-l1-cart*: $norm\ x <= setsum(\lambda i.\ |x\$i|)\ UNIV$
⟨*proof*⟩

lemma *scalar-mult-eq-scaleR*: $c *s x = c *_R x$
⟨*proof*⟩

lemma *dist-mul*[*simp*]: $dist\ (c *s x)\ (c *s y) = |c| * dist\ x\ y$
⟨*proof*⟩

lemma *setsum-component* [*simp*]:
fixes f:: $'a \Rightarrow ('b::comm\text{-}monoid\text{-}add)\ \hat{}\ 'n$
shows $(setsum\ f\ S)\$i = setsum\ (\lambda x.\ (f\ x)\$i)\ S$
⟨*proof*⟩

lemma *setsum-eq*: $setsum\ f\ S = (\chi\ i.\ setsum\ (\lambda x.\ (f\ x)\$i\ )\ S)$
⟨*proof*⟩

lemma *setsum-cmul*:
fixes f:: $'c \Rightarrow ('a::semiring\text{-}1)\ \hat{}\ 'n$
shows $setsum\ (\lambda x.\ c *s f\ x)\ S = c *s setsum\ f\ S$
⟨*proof*⟩

lemma *setsum-norm-allsubsets-bound-cart*:
fixes f:: $'a \Rightarrow real\ \hat{}\ 'n$
assumes *fP*: *finite* P **and** *fPs*: $\bigwedge Q.\ Q \subseteq P \Longrightarrow norm\ (setsum\ f\ Q) \leq e$
shows $setsum\ (\lambda x.\ norm\ (f\ x))\ P \leq 2 * real\ CARD('n) *\ e$
⟨*proof*⟩

42.4 Closures and interiors of halfspaces

lemma *interior-halfspace-le* [*simp*]:
assumes $a \neq 0$
shows $interior\ \{x.\ a \cdot x \leq b\} = \{x.\ a \cdot x < b\}$
⟨*proof*⟩

lemma *interior-halfspace-ge* [*simp*]:
$a \neq 0 \Longrightarrow interior\ \{x.\ a \cdot x \geq b\} = \{x.\ a \cdot x > b\}$
⟨*proof*⟩

lemma *interior-halfspace-component-le* [*simp*]:
  interior $\{x.\ x\$k \leq a\} = \{x :: (real,'n::finite)\ vec.\ x\$k < a\}$ (**is** *?LE*)
 and *interior-halfspace-component-ge* [*simp*]:
  interior $\{x.\ x\$k \geq a\} = \{x :: (real,'n::finite)\ vec.\ x\$k > a\}$ (**is** *?GE*)
⟨*proof*⟩

lemma *closure-halfspace-lt* [*simp*]:
 assumes $a \neq 0$
  shows *closure* $\{x.\ a \cdot x < b\} = \{x.\ a \cdot x \leq b\}$
⟨*proof*⟩

lemma *closure-halfspace-gt* [*simp*]:
 $a \neq 0 \Longrightarrow$ *closure* $\{x.\ a \cdot x > b\} = \{x.\ a \cdot x \geq b\}$
⟨*proof*⟩

lemma *closure-halfspace-component-lt* [*simp*]:
  closure $\{x.\ x\$k < a\} = \{x :: (real,'n::finite)\ vec.\ x\$k \leq a\}$ (**is** *?LE*)
 and *closure-halfspace-component-gt* [*simp*]:
  closure $\{x.\ x\$k > a\} = \{x :: (real,'n::finite)\ vec.\ x\$k \geq a\}$ (**is** *?GE*)
⟨*proof*⟩

lemma *interior-hyperplane* [*simp*]:
 assumes $a \neq 0$
  shows *interior* $\{x.\ a \cdot x = b\} = \{\}$
⟨*proof*⟩

lemma *frontier-halfspace-le*:
 assumes $a \neq 0 \vee b \neq 0$
  shows *frontier* $\{x.\ a \cdot x \leq b\} = \{x.\ a \cdot x = b\}$
⟨*proof*⟩

lemma *frontier-halfspace-ge*:
 assumes $a \neq 0 \vee b \neq 0$
  shows *frontier* $\{x.\ a \cdot x \geq b\} = \{x.\ a \cdot x = b\}$
⟨*proof*⟩

lemma *frontier-halfspace-lt*:
 assumes $a \neq 0 \vee b \neq 0$
  shows *frontier* $\{x.\ a \cdot x < b\} = \{x.\ a \cdot x = b\}$
⟨*proof*⟩

lemma *frontier-halfspace-gt*:
 assumes $a \neq 0 \vee b \neq 0$
  shows *frontier* $\{x.\ a \cdot x > b\} = \{x.\ a \cdot x = b\}$
⟨*proof*⟩

lemma *interior-standard-hyperplane*:
 interior $\{x :: (real,'n::finite)\ vec.\ x\$k = a\} = \{\}$
⟨*proof*⟩

42.5 Matrix operations

Matrix notation. NB: an MxN matrix is of type $(('a, 'n)\ vec, 'm)\ vec$, not $(('a, 'm)\ vec, 'n)\ vec$

definition *matrix-matrix-mult* :: $('a::semiring\text{-}1)\ \hat{}\ 'n\hat{}\ 'm \Rightarrow 'a\ \hat{}\ 'p\hat{}\ 'n \Rightarrow 'a\ \hat{}\ 'p\ \hat{}\ 'm$
  (**infixl** $**$ *70*)
  where $m ** m' == (\chi\ i\ j.\ setsum\ (\lambda k.\ ((m\$i)\$k) * ((m'\$k)\$j))\ (UNIV :: 'n\ set)) :: 'a\ \hat{}\ 'p\ \hat{}\ 'm$

definition *matrix-vector-mult* :: $('a::semiring\text{-}1)\ \hat{}\ 'n\hat{}\ 'm \Rightarrow 'a\ \hat{}\ 'n \Rightarrow 'a\ \hat{}\ 'm$
  (**infixl** $*v$ *70*)
  where $m *v x \equiv (\chi\ i.\ setsum\ (\lambda j.\ ((m\$i)\$j) * (x\$j))\ (UNIV :: 'n\ set)) :: 'a\hat{}\ 'm$

definition *vector-matrix-mult* :: $'a\ \hat{}\ 'm \Rightarrow ('a::semiring\text{-}1)\ \hat{}\ 'n\hat{}\ 'm \Rightarrow 'a\ \hat{}\ 'n$
  (**infixl** $v*$ *70*)
  where $v\ v* m == (\chi\ j.\ setsum\ (\lambda i.\ ((m\$i)\$j) * (v\$i))\ (UNIV :: 'm\ set)) :: 'a\hat{}\ 'n$

definition $(mat::'a::zero => 'a\ \hat{}\ 'n\hat{}\ 'n)\ k = (\chi\ i\ j.\ if\ i = j\ then\ k\ else\ 0)$
definition *transpose* **where**
  $(transpose::'a\hat{}\ 'n\hat{}\ 'm \Rightarrow 'a\hat{}\ 'm\hat{}\ 'n)\ A = (\chi\ i\ j.\ ((A\$j)\$i))$
definition $(row::'m => 'a\ \hat{}\ 'n\hat{}\ 'm \Rightarrow 'a\ \hat{}\ 'n)\ i\ A = (\chi\ j.\ ((A\$i)\$j))$
definition $(column::'n =>'a\hat{}\ 'n\hat{}\ 'm =>'a\hat{}\ 'm)\ j\ A = (\chi\ i.\ ((A\$i)\$j))$
definition $rows(A::'a\hat{}\ 'n\hat{}\ 'm) = \{\ row\ i\ A \mid i.\ i \in (UNIV :: 'm\ set)\}$
definition $columns(A::'a\hat{}\ 'n\hat{}\ 'm) = \{\ column\ i\ A \mid i.\ i \in (UNIV :: 'n\ set)\}$

lemma *mat-0*[*simp*]: *mat 0 = 0* ⟨*proof*⟩
lemma *matrix-add-ldistrib*: $(A ** (B + C)) = (A ** B) + (A ** C)$
  ⟨*proof*⟩

lemma *matrix-mul-lid*:
  fixes $A :: 'a::semiring\text{-}1\ \hat{}\ 'm\ \hat{}\ 'n$
  shows *mat 1* $**$ *A = A*
  ⟨*proof*⟩

lemma *matrix-mul-rid*:
  fixes $A :: 'a::semiring\text{-}1\ \hat{}\ 'm\ \hat{}\ 'n$
  shows *A* $**$ *mat 1 = A*
  ⟨*proof*⟩

lemma *matrix-mul-assoc*: $A ** (B ** C) = (A ** B) ** C$
  ⟨*proof*⟩

lemma *matrix-vector-mul-assoc*: $A *v (B *v x) = (A ** B) *v x$
  ⟨*proof*⟩

lemma *matrix-vector-mul-lid*: *mat 1* $*v$ *x* = $(x::'a::semiring\text{-}1\ \hat{}\ 'n)$

⟨*proof*⟩

lemma *matrix-transpose-mul*:
transpose(*A* ** *B*) = *transpose B* ** *transpose* (*A*::*'a*::*comm-semiring-1* ^ - ^ -)
⟨*proof*⟩

lemma *matrix-eq*:
fixes *A B* :: *'a*::*semiring-1* ^ *'n* ^ *'m*
shows *A* = *B* ⟷ (∀ *x*. *A* *v *x* = *B* *v *x*) (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *matrix-vector-mul-component*: ((*A*::*real* ^ - ^ -) *v *x*)$*k* = (*A*$*k*) · *x*
⟨*proof*⟩

lemma *dot-lmul-matrix*: ((*x*::*real* ^ -) *v** *A*) · *y* = *x* · (*A* *v *y*)
⟨*proof*⟩

lemma *transpose-mat*: *transpose* (*mat n*) = *mat n*
⟨*proof*⟩

lemma *transpose-transpose*: *transpose*(*transpose A*) = *A*
⟨*proof*⟩

lemma *row-transpose*:
fixes *A*:: *'a*::*semiring-1* ^ - ^ -
shows *row i* (*transpose A*) = *column i A*
⟨*proof*⟩

lemma *column-transpose*:
fixes *A*:: *'a*::*semiring-1* ^ - ^ -
shows *column i* (*transpose A*) = *row i A*
⟨*proof*⟩

lemma *rows-transpose*: *rows*(*transpose* (*A*::*'a*::*semiring-1* ^ - ^ -)) = *columns A*
⟨*proof*⟩

lemma *columns-transpose*: *columns*(*transpose* (*A*::*'a*::*semiring-1* ^ - ^ -)) = *rows A*
⟨*proof*⟩

Two sometimes fruitful ways of looking at matrix-vector multiplication.

lemma *matrix-mult-dot*: *A* *v *x* = (χ *i*. *A*$*i* · *x*)
⟨*proof*⟩

lemma *matrix-mult-vsum*:
(*A*::*'a*::*comm-semiring-1* ^ *'n* ^ *'m*) *v *x* = *setsum* (λ*i*. (*x*$*i*) *s *column i A*) (*UNIV*::
'n set)
⟨*proof*⟩

lemma *vector-componentwise*:

$(x::'a::ring\text{-}1\hat{}'n) = (\chi\ j.\ \sum i{\in}UNIV.\ (x\$i) * (axis\ i\ 1 :: 'a\hat{}'n)\ \$\ j)$
⟨*proof*⟩

lemma *basis-expansion*: $setsum\ (\lambda i.\ (x\$i) *s\ axis\ i\ 1)\ UNIV = (x::('a::ring\text{-}1)\hat{}'n)$
⟨*proof*⟩

lemma *linear-componentwise*:
fixes f:: $real\ \hat{}'m \Rightarrow real\ \hat{}\ -$
assumes *lf*: *linear f*
shows $(f\ x)\$j = setsum\ (\lambda i.\ (x\$i) * (f\ (axis\ i\ 1)\$j))\ (UNIV :: 'm\ set)$ (**is** *?lhs = ?rhs*)
⟨*proof*⟩

Inverse matrices (not necessarily square)

definition
$invertible(A::'a::semiring\text{-}1\hat{}'n\hat{}'m) \longleftrightarrow (\exists A'::'a\hat{}'m\hat{}'n.\ A ** A' = mat\ 1 \wedge A' ** A = mat\ 1)$

definition
$matrix\text{-}inv(A::\ 'a::semiring\text{-}1\hat{}'n\hat{}'m) =$
$(SOME\ A'::'a\hat{}'m\hat{}'n.\ A ** A' = mat\ 1 \wedge A' ** A = mat\ 1)$

Correspondence between matrices and linear operators.

definition $matrix :: ('a::\{plus,times,\ one,\ zero\}\hat{}'m \Rightarrow 'a\ \hat{}\ 'n) \Rightarrow 'a\hat{}'m\hat{}'n$
where $matrix\ f = (\chi\ i\ j.\ (f(axis\ j\ 1))\$i)$

lemma *matrix-vector-mul-linear*: $linear(\lambda x.\ A *v\ (x::real\ \hat{}\ -))$
⟨*proof*⟩

lemma *matrix-works*:
assumes *lf*: *linear f*
shows $matrix\ f *v\ x = f\ (x::real\ \hat{}\ 'n)$
⟨*proof*⟩

lemma *matrix-vector-mul*: $linear\ f ==> f = (\lambda x.\ matrix\ f *v\ (x::real\ \hat{}\ 'n))$
⟨*proof*⟩

lemma *matrix-of-matrix-vector-mul*: $matrix(\lambda x.\ A *v\ (x :: real\ \hat{}\ 'n)) = A$
⟨*proof*⟩

lemma *matrix-compose*:
assumes *lf*: $linear\ (f::real\hat{}'n \Rightarrow real\hat{}'m)$
and *lg*: $linear\ (g::real\hat{}'m \Rightarrow real\hat{}-)$
shows $matrix\ (g \circ f) = matrix\ g ** matrix\ f$
⟨*proof*⟩

lemma *matrix-vector-column*:
$(A::'a::comm\text{-}semiring\text{-}1\hat{}'n\hat{}-) *v\ x = setsum\ (\lambda i.\ (x\$i) *s\ ((transpose\ A)\$i))$

(*UNIV*:: *'n set*)
⟨*proof*⟩

lemma *adjoint-matrix*: *adjoint*(λ*x*. (*A*::*real^'n^'m*) ∗*v* *x*) = (λ*x*. *transpose A* ∗*v* *x*)
⟨*proof*⟩

lemma *matrix-adjoint*: **assumes** *lf*: *linear* (*f* :: *real^'n* ⇒ *real ^'m*)
shows *matrix*(*adjoint f*) = *transpose*(*matrix f*)
⟨*proof*⟩

42.6 lambda skolemization on cartesian products

lemma *lambda-skolem*: (∀ *i*. ∃ *x*. *P i x*) ⟷
(∃ *x*::*'a ^ 'n*. ∀ *i*. *P i* (*x* $ *i*)) (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

lemma *vector-sub-project-orthogonal-cart*: (*b*::*real^'n*) · (*x* − ((*b* · *x*) / (*b* · *b*)) ∗*s* *b*) = *0*
⟨*proof*⟩

lemma *left-invertible-transpose*:
(∃(*B*). *B* ∗∗ *transpose* (*A*) = *mat* (*1*::*'a*::*comm-semiring-1*)) ⟷ (∃(*B*). *A* ∗∗ *B* = *mat 1*)
⟨*proof*⟩

lemma *right-invertible-transpose*:
(∃(*B*). *transpose* (*A*) ∗∗ *B* = *mat* (*1*::*'a*::*comm-semiring-1*)) ⟷ (∃(*B*). *B* ∗∗ *A* = *mat 1*)
⟨*proof*⟩

lemma *matrix-left-invertible-injective*:
(∃ *B*. (*B*::*real^'m^'n*) ∗∗ (*A*::*real^'n^'m*) = *mat 1*) ⟷ (∀ *x y*. *A* ∗*v* *x* = *A* ∗*v* *y* ⟶ *x* = *y*)
⟨*proof*⟩

lemma *matrix-left-invertible-ker*:
(∃ *B*. (*B*::*real ^'m^'n*) ∗∗ (*A*::*real^'n^'m*) = *mat 1*) ⟷ (∀ *x*. *A* ∗*v* *x* = *0* ⟶ *x* = *0*)
⟨*proof*⟩

lemma *matrix-right-invertible-surjective*:
(∃ *B*. (*A*::*real^'n^'m*) ∗∗ (*B*::*real^'m^'n*) = *mat 1*) ⟷ *surj* (λ*x*. *A* ∗*v* *x*)
⟨*proof*⟩

lemma *matrix-left-invertible-independent-columns*:
fixes *A* :: *real^'n^'m*
shows (∃(*B*::*real ^'m^'n*). *B* ∗∗ *A* = *mat 1*) ⟷
(∀ *c*. *setsum* (λ*i*. *c i* ∗*s column i A*) (*UNIV* :: *'n set*) = *0* ⟶ (∀ *i*. *c i* = *0*))

(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *matrix-right-invertible-independent-rows*:
fixes *A* :: *real^'n^'m*
shows (∃(*B*::*real^'m^'n*). *A* ** *B* = *mat 1*) $\longleftrightarrow$
(∀ *c*. *setsum* (λ*i*. *c i* *s *row i A*) (*UNIV* :: *'m set*) = *0* $\longrightarrow$ (∀ *i*. *c i* = *0*))
⟨*proof*⟩

lemma *matrix-right-invertible-span-columns*:
(∃(*B*::*real ^'n^'m*). (*A*::*real ^'m^'n*) ** *B* = *mat 1*) $\longleftrightarrow$
span (*columns A*) = *UNIV* (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *matrix-left-invertible-span-rows*:
(∃(*B*::*real^'m^'n*). *B* ** (*A*::*real^'n^'m*) = *mat 1*) $\longleftrightarrow$ *span* (*rows A*) = *UNIV*
⟨*proof*⟩

The same result in terms of square matrices.

lemma *matrix-left-right-inverse*:
fixes *A A'* :: *real ^'n^'n*
shows *A* ** *A'* = *mat 1* $\longleftrightarrow$ *A'* ** *A* = *mat 1*
⟨*proof*⟩

Considering an n-element vector as an n-by-1 or 1-by-n matrix.

definition *rowvector v* = (χ *i j*. (*v*$*j*))

definition *columnvector v* = (χ *i j*. (*v*$*i*))

lemma *transpose-columnvector*: *transpose*(*columnvector v*) = *rowvector v*
⟨*proof*⟩

lemma *transpose-rowvector*: *transpose*(*rowvector v*) = *columnvector v*
⟨*proof*⟩

lemma *dot-rowvector-columnvector*: *columnvector* (*A* *v *v*) = *A* ** *columnvector v*
⟨*proof*⟩

lemma *dot-matrix-product*:
(*x*::*real^'n*) · *y* = (((*rowvector x* ::*real^'n^1*) ** (*columnvector y* :: *real^1^'n*))$*1*)$*1*
⟨*proof*⟩

lemma *dot-matrix-vector-mul*:
fixes *A B* :: *real ^'n ^'n* **and** *x y* :: *real ^'n*
shows (*A* *v *x*) · (*B* *v *y*) =
(((*rowvector x* :: *real^'n^1*) ** ((*transpose A* ** *B*) ** (*columnvector y* :: *real ^1^'n*)))$*1*)$*1*
⟨*proof*⟩

lemma *infnorm-cart:infnorm* (*x::real^'n*) = *Sup* {|*x$i*| |*i*. *i∈UNIV*}
⟨*proof*⟩

lemma *component-le-infnorm-cart*: |*x$i*| ≤ *infnorm* (*x::real^'n*)
⟨*proof*⟩

lemma *continuous-component*: *continuous F f* ⟹ *continuous F* (λ*x*. *f x* $ *i*)
⟨*proof*⟩

lemma *continuous-on-component*: *continuous-on s f* ⟹ *continuous-on s* (λ*x*. *f x* $ *i*)
⟨*proof*⟩

lemma *closed-positive-orthant*: *closed* {*x::real^'n*. ∀ *i*. *0* ≤*x$i*}
⟨*proof*⟩

lemma *bounded-component-cart*: *bounded s* ⟹ *bounded* ((λ*x*. *x* $ *i*) ' *s*)
⟨*proof*⟩

lemma *compact-lemma-cart*:
fixes *f* :: *nat* ⇒ *'a::heine-borel* ^ *'n*
assumes *f*: *bounded* (*range f*)
shows ∃ *l r*. *subseq r* ∧
(∀ *e*>*0*. *eventually* (λ*n*. ∀ *i*∈*d*. *dist* (*f* (*r n*) $ *i*) (*l* $ *i*) < *e*) *sequentially*)
(**is** *?th d*)
⟨*proof*⟩

instance *vec* :: (*heine-borel*, *finite*) *heine-borel*
⟨*proof*⟩

lemma *interval-cart*:
fixes *a* :: *real^'n*
shows *box a b* = {*x::real^'n*. ∀ *i*. *a$i* < *x$i* ∧ *x$i* < *b$i*}
and *cbox a b* = {*x::real^'n*. ∀ *i*. *a$i* ≤ *x$i* ∧ *x$i* ≤ *b$i*}
⟨*proof*⟩

lemma *mem-interval-cart*:
fixes *a* :: *real^'n*
shows *x* ∈ *box a b* ⟷ (∀ *i*. *a$i* < *x$i* ∧ *x$i* < *b$i*)
and *x* ∈ *cbox a b* ⟷ (∀ *i*. *a$i* ≤ *x$i* ∧ *x$i* ≤ *b$i*)
⟨*proof*⟩

lemma *interval-eq-empty-cart*:
fixes *a* :: *real^'n*
shows (*box a b* = {} ⟷ (∃ *i*. *b$i* ≤ *a$i*)) (**is** *?th1*)
and (*cbox a b* = {} ⟷ (∃ *i*. *b$i* < *a$i*)) (**is** *?th2*)
⟨*proof*⟩

lemma *interval-ne-empty-cart*:
fixes $a :: real\hat{}'n$
shows $cbox\ a\ b \neq \{\} \longleftrightarrow (\forall i.\ a\$i \leq b\$i)$
and $box\ a\ b \neq \{\} \longleftrightarrow (\forall i.\ a\$i < b\$i)$
⟨*proof*⟩

lemma *subset-interval-imp-cart*:
fixes $a :: real\hat{}'n$
shows $(\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i) \Longrightarrow cbox\ c\ d \subseteq cbox\ a\ b$
and $(\forall i.\ a\$i < c\$i \wedge d\$i < b\$i) \Longrightarrow cbox\ c\ d \subseteq box\ a\ b$
and $(\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i) \Longrightarrow box\ c\ d \subseteq cbox\ a\ b$
and $(\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i) \Longrightarrow box\ c\ d \subseteq box\ a\ b$
⟨*proof*⟩

lemma *interval-sing*:
fixes $a :: 'a::linorder\hat{}'n$
shows $\{a\ ..\ a\} = \{a\} \wedge \{a<..<a\} = \{\}$
⟨*proof*⟩

lemma *subset-interval-cart*:
fixes $a :: real\hat{}'n$
shows $cbox\ c\ d \subseteq cbox\ a\ b \longleftrightarrow (\forall i.\ c\$i \leq d\$i) \longrightarrow (\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i)$ (**is** *?th1*)
and $cbox\ c\ d \subseteq box\ a\ b \longleftrightarrow (\forall i.\ c\$i \leq d\$i) \longrightarrow (\forall i.\ a\$i < c\$i \wedge d\$i < b\$i)$ (**is** *?th2*)
and $box\ c\ d \subseteq cbox\ a\ b \longleftrightarrow (\forall i.\ c\$i < d\$i) \longrightarrow (\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i)$ (**is** *?th3*)
and $box\ c\ d \subseteq box\ a\ b \longleftrightarrow (\forall i.\ c\$i < d\$i) \longrightarrow (\forall i.\ a\$i \leq c\$i \wedge d\$i \leq b\$i)$ (**is** *?th4*)
⟨*proof*⟩

lemma *disjoint-interval-cart*:
fixes $a::real\hat{}'n$
shows $cbox\ a\ b \cap cbox\ c\ d = \{\} \longleftrightarrow (\exists i.\ (b\$i < a\$i \vee d\$i < c\$i \vee b\$i < c\$i \vee d\$i < a\$i))$ (**is** *?th1*)
and $cbox\ a\ b \cap box\ c\ d = \{\} \longleftrightarrow (\exists i.\ (b\$i < a\$i \vee d\$i \leq c\$i \vee b\$i \leq c\$i \vee d\$i \leq a\$i))$ (**is** *?th2*)
and $box\ a\ b \cap cbox\ c\ d = \{\} \longleftrightarrow (\exists i.\ (b\$i \leq a\$i \vee d\$i < c\$i \vee b\$i \leq c\$i \vee d\$i \leq a\$i))$ (**is** *?th3*)
and $box\ a\ b \cap box\ c\ d = \{\} \longleftrightarrow (\exists i.\ (b\$i \leq a\$i \vee d\$i \leq c\$i \vee b\$i \leq c\$i \vee d\$i \leq a\$i))$ (**is** *?th4*)
⟨*proof*⟩

lemma *inter-interval-cart*:
fixes $a :: real\hat{}'n$
shows $cbox\ a\ b \cap cbox\ c\ d = \{(\chi\ i.\ max\ (a\$i)\ (c\$i))\ ..\ (\chi\ i.\ min\ (b\$i)\ (d\$i))\}$
⟨*proof*⟩

lemma *closed-interval-left-cart*:
fixes $b :: real\hat{}'n$
shows *closed* $\{x::real\hat{}'n.\ \forall\, i.\ x\$i \leq b\$i\}$
⟨*proof*⟩

lemma *closed-interval-right-cart*:
fixes $a::real\hat{}'n$
shows *closed* $\{x::real\hat{}'n.\ \forall\, i.\ a\$i \leq x\$i\}$
⟨*proof*⟩

lemma *is-interval-cart*:
is-interval $(s::(real\hat{}'n)\ set) \longleftrightarrow$
$(\forall\, a{\in}s.\ \forall\, b{\in}s.\ \forall\, x.\ (\forall\, i.\ ((a\$i \leq x\$i \wedge x\$i \leq b\$i) \vee (b\$i \leq x\$i \wedge x\$i \leq a\$i))) \longrightarrow x \in s)$
⟨*proof*⟩

lemma *closed-halfspace-component-le-cart*: *closed* $\{x::real\hat{}'n.\ x\$i \leq a\}$
⟨*proof*⟩

lemma *closed-halfspace-component-ge-cart*: *closed* $\{x::real\hat{}'n.\ x\$i \geq a\}$
⟨*proof*⟩

lemma *open-halfspace-component-lt-cart*: *open* $\{x::real\hat{}'n.\ x\$i < a\}$
⟨*proof*⟩

lemma *open-halfspace-component-gt-cart*: *open* $\{x::real\hat{}'n.\ x\$i > a\}$
⟨*proof*⟩

lemma *Lim-component-le-cart*:
fixes $f :: 'a \Rightarrow real\hat{}'n$
assumes $(f \longrightarrow l)\ net\ \neg\ (trivial\text{-}limit\ net)\ \ eventually\ (\lambda x.\ f\ x\ \$i \leq b)\ net$
shows $l\$i \leq b$
⟨*proof*⟩

lemma *Lim-component-ge-cart*:
fixes $f :: 'a \Rightarrow real\hat{}'n$
assumes $(f \longrightarrow l)\ net\ \neg\ (trivial\text{-}limit\ net)\ \ eventually\ (\lambda x.\ b \leq (f\ x)\$i)\ net$
shows $b \leq l\$i$
⟨*proof*⟩

lemma *Lim-component-eq-cart*:
fixes $f :: 'a \Rightarrow real\hat{}'n$
assumes *net*: $(f \longrightarrow l)\ net\ \sim(trivial\text{-}limit\ net)$ **and** *ev*:*eventually* $(\lambda x.\ f(x)\$i = b)\ net$
shows $l\$i = b$
⟨*proof*⟩

lemma *connected-ivt-component-cart*:

fixes $x :: real\hat{}'n$
shows *connected* $s \Longrightarrow x \in s \Longrightarrow y \in s \Longrightarrow x\$k \leq a \Longrightarrow a \leq y\$k \Longrightarrow (\exists z{\in}s.\ z\$k = a)$
⟨*proof*⟩

lemma *subspace-substandard-cart*: *subspace* $\{x::real\hat{}\text{-}.\ (\forall i.\ P\ i \longrightarrow x\$i = 0)\}$
⟨*proof*⟩

lemma *closed-substandard-cart*:
closed $\{x::'a::real\text{-}normed\text{-}vector\ \hat{}\ 'n.\ \forall i.\ P\ i \longrightarrow x\$i = 0\}$
⟨*proof*⟩

lemma *dim-substandard-cart*: $dim\ \{x::real\hat{}'n.\ \forall i.\ i \notin d \longrightarrow x\$i = 0\} = card\ d$
(**is** *dim ?A* = -)
⟨*proof*⟩

lemma *affinity-inverses*:
assumes *m0*: $m \neq (0::'a::field)$
shows $(\lambda x.\ m *s\ x + c) \circ (\lambda x.\ inverse(m) *s\ x + (-(inverse(m) *s\ c))) = id$
$(\lambda x.\ inverse(m) *s\ x + (-(inverse(m) *s\ c))) \circ (\lambda x.\ m *s\ x + c) = id$
⟨*proof*⟩

lemma *vector-affinity-eq*:
assumes *m0*: $(m::'a::field) \neq 0$
shows $m *s\ x + c = y \longleftrightarrow x = inverse\ m *s\ y + -(inverse\ m *s\ c)$
⟨*proof*⟩

lemma *vector-eq-affinity*:
$(m::'a::field) \neq 0 ==> (y = m *s\ x + c \longleftrightarrow inverse(m) *s\ y + -(inverse(m) *s\ c) = x)$
⟨*proof*⟩

lemma *vector-cart*:
fixes $f :: real\hat{}'n \Rightarrow real$
shows $(\chi\ i.\ f\ (axis\ i\ 1)) = (\sum i{\in}Basis.\ f\ i *_R\ i)$
⟨*proof*⟩

lemma *const-vector-cart*:$((\chi\ i.\ d)::real\hat{}'n) = (\sum i{\in}Basis.\ d *_R\ i)$
⟨*proof*⟩

42.7 Convex Euclidean Space

lemma *Cart-1*:$(1::real\hat{}'n) = \sum Basis$
⟨*proof*⟩

declare *vector-add-ldistrib*[*simp*] *vector-ssub-ldistrib*[*simp*] *vector-smult-assoc*[*simp*] *vector-smult-rneg*[*simp*]
declare *vector-sadd-rdistrib*[*simp*] *vector-sub-rdistrib*[*simp*]

lemmas *vector-component-simps* = *vector-minus-component vector-smult-component vector-add-component less-eq-vec-def vec-lambda-beta vector-uminus-component*

lemma *convex-box-cart*:
assumes $\bigwedge i.\ convex\ \{x.\ P\ i\ x\}$
shows $convex\ \{x.\ \forall i.\ P\ i\ (x\$i)\}$
⟨*proof*⟩

lemma *convex-positive-orthant-cart*: $convex\ \{x::real\hat{}'n.\ (\forall i.\ 0 \le x\$i)\}$
⟨*proof*⟩

lemma *unit-interval-convex-hull-cart*:
$cbox\ (0::real\hat{}'n)\ 1 = convex\ hull\ \{x.\ \forall i.\ (x\$i = 0) \vee (x\$i = 1)\}$
⟨*proof*⟩

lemma *cube-convex-hull-cart*:
assumes $0 < d$
obtains $s::(real\hat{}'n)\ set$
where $finite\ s\ cbox\ (x - (\chi\ i.\ d))\ (x + (\chi\ i.\ d)) = convex\ hull\ s$
⟨*proof*⟩

42.8 Derivative

definition $jacobian\ f\ net = matrix(frechet\text{-}derivative\ f\ net)$

lemma *jacobian-works*:
$(f::(real\hat{}'a) \Rightarrow (real\hat{}'b))\ differentiable\ net \longleftrightarrow$
$(f\ has\text{-}derivative\ (\lambda h.\ (jacobian\ f\ net) *v h))\ net$
⟨*proof*⟩

42.9 Component of the differential must be zero if it exists at a local maximum or minimum for that corresponding component.

lemma *differential-zero-maxmin-cart*:
fixes $f::real\hat{}'a \Rightarrow real\hat{}'b$
assumes $0 < e\ ((\forall y \in ball\ x\ e.\ (f\ y)\$k \le (f\ x)\$k) \vee (\forall y \in ball\ x\ e.\ (f\ x)\$k \le (f\ y)\$k))$
$f\ differentiable\ (at\ x)$
shows $jacobian\ f\ (at\ x)\ \$\ k = 0$
⟨*proof*⟩

42.10 Lemmas for working on $(real,\ 1)\ vec$

lemma *forall-1*[*simp*]: $(\forall i::1.\ P\ i) \longleftrightarrow P\ 1$
⟨*proof*⟩

lemma *ex-1*[*simp*]: $(\exists x::1.\ P\ x) \longleftrightarrow P\ 1$
⟨*proof*⟩

lemma *exhaust-2*:
fixes $x :: 2$
shows $x = 1 \vee x = 2$
⟨*proof*⟩

lemma *forall-2*: $(\forall i{::}2.\ P\ i) \longleftrightarrow P\ 1 \wedge P\ 2$
⟨*proof*⟩

lemma *exhaust-3*:
fixes $x :: 3$
shows $x = 1 \vee x = 2 \vee x = 3$
⟨*proof*⟩

lemma *forall-3*: $(\forall i{::}3.\ P\ i) \longleftrightarrow P\ 1 \wedge P\ 2 \wedge P\ 3$
⟨*proof*⟩

lemma *UNIV-1* [*simp*]: $UNIV = \{1{::}1\}$
⟨*proof*⟩

lemma *UNIV-2*: $UNIV = \{1{::}2,\ 2{::}2\}$
⟨*proof*⟩

lemma *UNIV-3*: $UNIV = \{1{::}3,\ 2{::}3,\ 3{::}3\}$
⟨*proof*⟩

lemma *setsum-1*: $setsum\ f\ (UNIV{::}1\ set) = f\ 1$
⟨*proof*⟩

lemma *setsum-2*: $setsum\ f\ (UNIV{::}2\ set) = f\ 1 + f\ 2$
⟨*proof*⟩

lemma *setsum-3*: $setsum\ f\ (UNIV{::}3\ set) = f\ 1 + f\ 2 + f\ 3$
⟨*proof*⟩

instantiation *num1* :: *cart-one*
begin

instance
⟨*proof*⟩

end

42.11 The collapse of the general concepts to dimension one.

lemma *vector-one*: $(x{::}'a\ \hat{}\ 1) = (\chi\ i.\ (x\$1))$
⟨*proof*⟩

lemma *forall-one*: $(\forall (x{::}'a\ \hat{}\ 1).\ P\ x) \longleftrightarrow (\forall x.\ P(\chi\ i.\ x))$

⟨*proof*⟩

lemma *norm-vector-1*: *norm* (*x* :: -^*1*) = *norm* (*x*$*1*)
⟨*proof*⟩

lemma *norm-real*: *norm*(*x*::*real* ^ *1*) = |*x*$*1*|
⟨*proof*⟩

lemma *dist-real*: *dist*(*x*::*real* ^ *1*) *y* = |(*x*$*1*) − (*y*$*1*)|
⟨*proof*⟩

42.12 Explicit vector construction from lists.

definition *vector l* = (χ *i*. *foldr* (λ*x f n*. *fun-upd* (*f* (*n*+*1*)) *n x*) *l* (λ*n x*. *0*) *1 i*)

lemma *vector-1*: (*vector*[*x*]) $*1* = *x*
⟨*proof*⟩

lemma *vector-2*:
(*vector*[*x*,*y*]) $*1* = *x*
(*vector*[*x*,*y*] :: *'a*^*2*)$*2* = (*y*::*'a*::*zero*)
⟨*proof*⟩

lemma *vector-3*:
(*vector* [*x*,*y*,*z*] ::(*'a*::*zero*)^*3*)$*1* = *x*
(*vector* [*x*,*y*,*z*] ::(*'a*::*zero*)^*3*)$*2* = *y*
(*vector* [*x*,*y*,*z*] ::(*'a*::*zero*)^*3*)$*3* = *z*
⟨*proof*⟩

lemma *forall-vector-1*: (∀ *v*::*'a*::*zero*^*1*. *P v*) ⟷ (∀ *x*. *P*(*vector*[*x*]))
⟨*proof*⟩

lemma *forall-vector-2*: (∀ *v*::*'a*::*zero*^*2*. *P v*) ⟷ (∀ *x y*. *P*(*vector*[*x*, *y*]))
⟨*proof*⟩

lemma *forall-vector-3*: (∀ *v*::*'a*::*zero*^*3*. *P v*) ⟷ (∀ *x y z*. *P*(*vector*[*x*, *y*, *z*]))
⟨*proof*⟩

lemma *bounded-linear-component-cart*[*intro*]: *bounded-linear* (λ*x*::*real*^*'n*. *x* $ *k*)
⟨*proof*⟩

lemma *integral-component-eq-cart*[*simp*]:
fixes *f* :: *'n*::*euclidean-space* ⇒ *real*^*'m*
assumes *f integrable-on s*
shows *integral s* (λ*x*. *f x* $ *k*) = *integral s f* $ *k*
⟨*proof*⟩

lemma *interval-split-cart*:
{*a*..*b*::*real*^*'n*} ∩ {*x*. *x*$*k* ≤ *c*} = {*a* .. (χ *i*. *if i* = *k then min* (*b*$*k*) *c else b*$*i*)}

cbox a b ∩ {*x*. *x*$*k* ≥ *c*} = {(χ *i*. *if i* = *k then max* (*a*$*k*) *c else a*$*i*) .. *b*}
⟨*proof*⟩

end

43 Fashoda meet theorem

theory *Fashoda*
imports *Brouwer-Fixpoint Path-Connected Cartesian-Euclidean-Space*
begin

43.1 Bijections between intervals.

definition *interval-bij* :: ′*a* × ′*a* ⇒ ′*a* × ′*a* ⇒ ′*a* ⇒ ′*a*::*euclidean-space*
where *interval-bij* =
(λ(*a*, *b*) (*u*, *v*) *x*. (∑ *i*∈*Basis*. (*u*•*i* + (*x*•*i* − *a*•*i*) / (*b*•*i* − *a*•*i*) ∗ (*v*•*i* − *u*•*i*)) ∗$_R$ *i*))

lemma *interval-bij-affine*:
interval-bij (*a*,*b*) (*u*,*v*) = (λ*x*. (∑ *i*∈*Basis*. ((*v*•*i* − *u*•*i*) / (*b*•*i* − *a*•*i*) ∗ (*x*•*i*)) ∗$_R$ *i*) +
(∑ *i*∈*Basis*. (*u*•*i* − (*v*•*i* − *u*•*i*) / (*b*•*i* − *a*•*i*) ∗ (*a*•*i*)) ∗$_R$ *i*))
⟨*proof*⟩

lemma *continuous-interval-bij*:
fixes *a b* :: ′*a*::*euclidean-space*
shows *continuous* (*at x*) (*interval-bij* (*a*, *b*) (*u*, *v*))
⟨*proof*⟩

lemma *continuous-on-interval-bij*: *continuous-on s* (*interval-bij* (*a*, *b*) (*u*, *v*))
⟨*proof*⟩

lemma *in-interval-interval-bij*:
fixes *a b u v x* :: ′*a*::*euclidean-space*
assumes *x* ∈ *cbox a b*
and *cbox u v* ≠ {}
shows *interval-bij* (*a*, *b*) (*u*, *v*) *x* ∈ *cbox u v*
⟨*proof*⟩

lemma *interval-bij-bij*:
∀(*i*::′*a*::*euclidean-space*)∈*Basis*. *a*•*i* < *b*•*i* ∧ *u*•*i* < *v*•*i* ⟹
interval-bij (*a*, *b*) (*u*, *v*) (*interval-bij* (*u*, *v*) (*a*, *b*) *x*) = *x*
⟨*proof*⟩

lemma *interval-bij-bij-cart*: **fixes** *x*::*real*^′*n* **assumes** ∀ *i*. *a*$*i* < *b*$*i* ∧ *u*$*i* < *v*$*i*
shows *interval-bij* (*a*,*b*) (*u*,*v*) (*interval-bij* (*u*,*v*) (*a*,*b*) *x*) = *x*
⟨*proof*⟩

43.2 Fashoda meet theorem

lemma *infnorm-2*:
fixes x :: *real^2*
shows *infnorm* x = *max* $|x\$1|$ $|x\$2|$
⟨*proof*⟩

lemma *infnorm-eq-1-2*:
fixes x :: *real^2*
shows *infnorm* $x = 1 \longleftrightarrow$
$|x\$1| \leq 1 \wedge |x\$2| \leq 1 \wedge (x\$1 = -1 \vee x\$1 = 1 \vee x\$2 = -1 \vee x\$2 = 1)$
⟨*proof*⟩

lemma *infnorm-eq-1-imp*:
fixes x :: *real^2*
assumes *infnorm* $x = 1$
shows $|x\$1| \leq 1$ **and** $|x\$2| \leq 1$
⟨*proof*⟩

lemma *fashoda-unit*:
fixes f g :: *real* ⇒ *real^2*
assumes f ' $\{-1 .. 1\} \subseteq$ *cbox* (-1) 1
and g ' $\{-1 .. 1\} \subseteq$ *cbox* (-1) 1
and *continuous-on* $\{-1 .. 1\}$ f
and *continuous-on* $\{-1 .. 1\}$ g
and f $(-1)\$1 = -1$
and f $1\$1 = 1$ g (-1) $\$2 = -1$
and g 1 $\$2 = 1$
shows $\exists s \in \{-1 .. 1\}.\ \exists t \in \{-1 .. 1\}.\ f\ s = g\ t$
⟨*proof*⟩

lemma *fashoda-unit-path*:
fixes f g :: *real* ⇒ *real^2*
assumes *path* f
and *path* g
and *path-image* $f \subseteq$ *cbox* (-1) 1
and *path-image* $g \subseteq$ *cbox* (-1) 1
and (*pathstart* f)$\$1 = -1$
and (*pathfinish* f)$\$1 = 1$
and (*pathstart* g)$\$2 = -1$
and (*pathfinish* g)$\$2 = 1$
obtains z **where** $z \in$ *path-image* f **and** $z \in$ *path-image* g
⟨*proof*⟩

lemma *fashoda*:
fixes b :: *real^2*
assumes *path* f
and *path* g
and *path-image* $f \subseteq$ *cbox* a b
and *path-image* $g \subseteq$ *cbox* a b

and (*pathstart f*)$*1* = *a*$*1*
and (*pathfinish f*)$*1* = *b*$*1*
and (*pathstart g*)$*2* = *a*$*2*
and (*pathfinish g*)$*2* = *b*$*2*
obtains *z* **where** *z* ∈ *path-image f* **and** *z* ∈ *path-image g*
⟨*proof*⟩

43.3 Some slightly ad hoc lemmas I use below

lemma *segment-vertical*:
fixes *a* :: *real^2*
assumes *a*$*1* = *b*$*1*
shows *x* ∈ *closed-segment a b* ⟷
x$*1* = *a*$*1* ∧ *x*$*1* = *b*$*1* ∧ (*a*$*2* ≤ *x*$*2* ∧ *x*$*2* ≤ *b*$*2* ∨ *b*$*2* ≤ *x*$*2* ∧ *x*$*2* ≤ *a*$*2*)
(**is** - = *?R*)
⟨*proof*⟩

lemma *segment-horizontal*:
fixes *a* :: *real^2*
assumes *a*$*2* = *b*$*2*
shows *x* ∈ *closed-segment a b* ⟷
x$*2* = *a*$*2* ∧ *x*$*2* = *b*$*2* ∧ (*a*$*1* ≤ *x*$*1* ∧ *x*$*1* ≤ *b*$*1* ∨ *b*$*1* ≤ *x*$*1* ∧ *x*$*1* ≤ *a*$*1*)
(**is** - = *?R*)
⟨*proof*⟩

43.4 Useful Fashoda corollary pointed out to me by Tom Hales

lemma *fashoda-interlace*:
fixes *a* :: *real^2*
assumes *path f*
and *path g*
and *path-image f* ⊆ *cbox a b*
and *path-image g* ⊆ *cbox a b*
and (*pathstart f*)$*2* = *a*$*2*
and (*pathfinish f*)$*2* = *a*$*2*
and (*pathstart g*)$*2* = *a*$*2*
and (*pathfinish g*)$*2* = *a*$*2*
and (*pathstart f*)$*1* < (*pathstart g*)$*1*
and (*pathstart g*)$*1* < (*pathfinish f*)$*1*
and (*pathfinish f*)$*1* < (*pathfinish g*)$*1*
obtains *z* **where** *z* ∈ *path-image f* **and** *z* ∈ *path-image g*
⟨*proof*⟩

end

43.5 The type of non-negative extended real numbers

theory *Extended-Nonnegative-Real*
imports *Extended-Real Indicator-Function*
begin

lemma *ereal-ineq-diff-add*:
assumes $b \neq (-\infty::ereal)$ $a \geq b$
shows $a = b + (a-b)$
⟨*proof*⟩

lemma *Limsup-const-add*:
fixes $c :: 'a::\{complete\text{-}linorder, linorder\text{-}topology, topological\text{-}monoid\text{-}add, ordered\text{-}ab\text{-}semigroup\text{-}add\}$
shows $F \neq bot \Longrightarrow Limsup\ F\ (\lambda x.\ c + f\ x) = c + Limsup\ F\ f$
⟨*proof*⟩

lemma *Liminf-const-add*:
fixes $c :: 'a::\{complete\text{-}linorder, linorder\text{-}topology, topological\text{-}monoid\text{-}add, ordered\text{-}ab\text{-}semigroup\text{-}add\}$
shows $F \neq bot \Longrightarrow Liminf\ F\ (\lambda x.\ c + f\ x) = c + Liminf\ F\ f$
⟨*proof*⟩

lemma *Liminf-add-const*:
fixes $c :: 'a::\{complete\text{-}linorder, linorder\text{-}topology, topological\text{-}monoid\text{-}add, ordered\text{-}ab\text{-}semigroup\text{-}add\}$
shows $F \neq bot \Longrightarrow Liminf\ F\ (\lambda x.\ f\ x + c) = Liminf\ F\ f + c$
⟨*proof*⟩

lemma *sums-offset*:
fixes $f\ g :: nat \Rightarrow 'a :: \{t2\text{-}space, topological\text{-}comm\text{-}monoid\text{-}add\}$
assumes $(\lambda n.\ f\ (n + i))\ sums\ l$ **shows** $f\ sums\ (l + (\sum j<i.\ f\ j))$
⟨*proof*⟩

lemma *suminf-offset*:
fixes $f\ g :: nat \Rightarrow 'a :: \{t2\text{-}space, topological\text{-}comm\text{-}monoid\text{-}add\}$
shows $summable\ (\lambda j.\ f\ (j + i)) \Longrightarrow suminf\ f = (\sum j.\ f\ (j + i)) + (\sum j<i.\ f\ j)$
⟨*proof*⟩

lemma *eventually-at-left-1*: $(\bigwedge z::real.\ 0 < z \Longrightarrow z < 1 \Longrightarrow P\ z) \Longrightarrow eventually\ P\ (at\text{-}left\ 1)$
⟨*proof*⟩

lemma *mult-eq-1*:
fixes $a\ b :: 'a :: \{ordered\text{-}semiring, comm\text{-}monoid\text{-}mult\}$
shows $0 \leq a \Longrightarrow a \leq 1 \Longrightarrow b \leq 1 \Longrightarrow a * b = 1 \longleftrightarrow (a = 1 \wedge b = 1)$
⟨*proof*⟩

lemma *ereal-add-diff-cancel*:
fixes $a\ b :: ereal$
shows $|b| \neq \infty \Longrightarrow (a + b) - b = a$
⟨*proof*⟩

lemma *add-top*:
fixes $x :: 'a::\{order\text{-}top, ordered\text{-}comm\text{-}monoid\text{-}add\}$
shows $0 \leq x \Longrightarrow x + top = top$
⟨*proof*⟩

lemma *top-add*:
fixes $x :: 'a::\{order\text{-}top, ordered\text{-}comm\text{-}monoid\text{-}add\}$
shows $0 \leq x \Longrightarrow top + x = top$
⟨*proof*⟩

lemma *le-lfp*: $mono\ f \Longrightarrow x \leq lfp\ f \Longrightarrow f\ x \leq lfp\ f$
⟨*proof*⟩

lemma *lfp-transfer*:
assumes α: *sup-continuous* α **and** f: *sup-continuous* f **and** *mg*: *mono* g
assumes *bot*: $\alpha\ bot \leq lfp\ g$ **and** *eq*: $\bigwedge x.\ x \leq lfp\ f \Longrightarrow \alpha\ (f\ x) = g\ (\alpha\ x)$
shows $\alpha\ (lfp\ f) = lfp\ g$
⟨*proof*⟩

lemma *sup-continuous-applyD*: *sup-continuous* $f \Longrightarrow$ *sup-continuous* $(\lambda x.\ f\ x\ h)$
⟨*proof*⟩

lemma *sup-continuous-SUP*[*order-continuous-intros*]:
fixes $M :: - \Rightarrow - \Rightarrow 'a::complete\text{-}lattice$
assumes M: $\bigwedge i.\ i \in I \Longrightarrow$ *sup-continuous* $(M\ i)$
shows *sup-continuous* $(SUP\ i{:}I.\ M\ i)$
⟨*proof*⟩

lemma *sup-continuous-apply-SUP*[*order-continuous-intros*]:
fixes $M :: - \Rightarrow - \Rightarrow 'a::complete\text{-}lattice$
shows $(\bigwedge i.\ i \in I \Longrightarrow$ *sup-continuous* $(M\ i)) \Longrightarrow$ *sup-continuous* $(\lambda x.\ SUP\ i{:}I.\ M\ i\ x)$
⟨*proof*⟩

lemma *sup-continuous-lfp'*[*order-continuous-intros*]:
assumes *1*: *sup-continuous* f
assumes *2*: $\bigwedge g.$ *sup-continuous* $g \Longrightarrow$ *sup-continuous* $(f\ g)$
shows *sup-continuous* $(lfp\ f)$
⟨*proof*⟩

lemma *sup-continuous-lfp''*[*order-continuous-intros*]:
assumes *1*: $\bigwedge s.$ *sup-continuous* $(f\ s)$
assumes *2*: $\bigwedge g.$ *sup-continuous* $g \Longrightarrow$ *sup-continuous* $(\lambda s.\ f\ s\ (g\ s))$
shows *sup-continuous* $(\lambda x.\ lfp\ (f\ x))$
⟨*proof*⟩

lemma *mono-INF-fun*:
$(\bigwedge x\ y.\ mono\ (F\ x\ y)) \Longrightarrow mono\ (\lambda z\ x.\ INF\ y : X\ x.\ F\ x\ y\ z :: 'a :: complete\text{-}lattice)$

⟨*proof*⟩

lemma *continuous-on-max*:
fixes *f g* :: *'a*::*topological-space* ⇒ *'b*::*linorder-topology*
shows *continuous-on A f* ⟹ *continuous-on A g* ⟹ *continuous-on A* (λ*x*. *max* (*f x*) (*g x*))
⟨*proof*⟩

lemma *continuous-on-cmult-ereal*:
|*c*::*ereal*| ≠ ∞ ⟹ *continuous-on A f* ⟹ *continuous-on A* (λ*x*. *c* ∗ *f x*)
⟨*proof*⟩

context *linordered-nonzero-semiring*
begin

lemma *of-nat-nonneg* [*simp*]: *0* ≤ *of-nat n*
⟨*proof*⟩

lemma *of-nat-mono*[*simp*]: *i* ≤ *j* ⟹ *of-nat i* ≤ *of-nat j*
⟨*proof*⟩

end

lemma *real-of-nat-Sup*:
assumes *A* ≠ {} *bdd-above A*
shows *of-nat* (*Sup A*) = (*SUP a*:*A*. *of-nat a* :: *real*)
⟨*proof*⟩

lemma *of-nat-less*[*simp*]:
i < *j* ⟹ *of-nat i* < (*of-nat j*::*'a*::{*linordered-nonzero-semiring*, *semiring-char-0*})
⟨*proof*⟩

lemma *of-nat-le-iff*[*simp*]:
of-nat i ≤ (*of-nat j*::*'a*::{*linordered-nonzero-semiring*, *semiring-char-0*}) ⟷ *i* ≤ *j*
⟨*proof*⟩

lemma (**in** *complete-lattice*) *SUP-sup-const1*:
I ≠ {} ⟹ (*SUP i*:*I*. *sup c* (*f i*)) = *sup c* (*SUP i*:*I*. *f i*)
⟨*proof*⟩

lemma (**in** *complete-lattice*) *SUP-sup-const2*:
I ≠ {} ⟹ (*SUP i*:*I*. *sup* (*f i*) *c*) = *sup* (*SUP i*:*I*. *f i*) *c*
⟨*proof*⟩

lemma *one-less-of-natD*:
(*1*::*'a*::*linordered-semidom*) < *of-nat n* ⟹ *1* < *n*
⟨*proof*⟩

lemma *setsum-le-suminf*:
fixes $f :: nat \Rightarrow 'a::\{ordered\text{-}comm\text{-}monoid\text{-}add, linorder\text{-}topology\}$
shows $summable\ f \Longrightarrow finite\ I \Longrightarrow \forall\, m{\in}-\ I.\ 0 \le f\ m \Longrightarrow setsum\ f\ I \le suminf\ f$
⟨*proof*⟩

43.6 Defining the extended non-negative reals

Basic definitions and type class setup

typedef $ennreal = \{x :: ereal.\ 0 \le x\}$
morphisms *enn2ereal e2ennreal'*
⟨*proof*⟩

definition $e2ennreal\ x = e2ennreal'\ (max\ 0\ x)$

lemma *enn2ereal-range*: $e2ennreal\ `\ \{0..\} = UNIV$
⟨*proof*⟩

lemma *type-definition-ennreal'*: *type-definition enn2ereal e2ennreal* $\{x.\ 0 \le x\}$
⟨*proof*⟩

setup-lifting *type-definition-ennreal'*

declare [[*coercion e2ennreal*]]

instantiation *ennreal* :: *complete-linorder*
begin

lift-definition *top-ennreal* :: *ennreal* **is** *top* ⟨*proof*⟩
lift-definition *bot-ennreal* :: *ennreal* **is** *0* ⟨*proof*⟩
lift-definition *sup-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal* **is** *sup* ⟨*proof*⟩
lift-definition *inf-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal* **is** *inf* ⟨*proof*⟩

lift-definition *Inf-ennreal* :: *ennreal set* ⇒ *ennreal* **is** *Inf*
⟨*proof*⟩

lift-definition *Sup-ennreal* :: *ennreal set* ⇒ *ennreal* **is** *sup 0* ∘ *Sup*
⟨*proof*⟩

lift-definition *less-eq-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *bool* **is** *op* ≤ ⟨*proof*⟩
lift-definition *less-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *bool* **is** *op* < ⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *pcr-ennreal-enn2ereal*[*simp*]: *pcr-ennreal* (*enn2ereal x*) *x*
⟨*proof*⟩

lemma *rel-fun-eq-pcr-ennreal*: *rel-fun op* = *pcr-ennreal f g* ⟷ *f* = *enn2ereal* ∘ *g*
⟨*proof*⟩

instantiation *ennreal* :: *infinity*
begin

definition *infinity-ennreal* :: *ennreal*
where
[*simp*]: ∞ = (*top*::*ennreal*)

instance ⟨*proof*⟩

end

instantiation *ennreal* :: {*semiring-1-no-zero-divisors*, *comm-semiring-1*}
begin

lift-definition *one-ennreal* :: *ennreal* **is** *1* ⟨*proof*⟩
lift-definition *zero-ennreal* :: *ennreal* **is** *0* ⟨*proof*⟩
lift-definition *plus-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal* **is** *op* + ⟨*proof*⟩
lift-definition *times-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal* **is** *op* ∗ ⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation *ennreal* :: *minus*
begin

lift-definition *minus-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal* **is** λ*a b*. *max 0* (*a* − *b*)
⟨*proof*⟩

instance ⟨*proof*⟩

end

instance *ennreal* :: *numeral* ⟨*proof*⟩

instantiation *ennreal* :: *inverse*
begin

lift-definition *inverse-ennreal* :: *ennreal* ⇒ *ennreal* **is** *inverse*
⟨*proof*⟩

definition *divide-ennreal* :: *ennreal* ⇒ *ennreal* ⇒ *ennreal*
where *x div y* = *x* ∗ *inverse* (*y* :: *ennreal*)

instance ⟨*proof*⟩

end

lemma *ennreal-zero-less-one*: *0* < (*1*::*ennreal*) — TODO: remove
⟨*proof*⟩

instance *ennreal* :: *dioid*
⟨*proof*⟩

instance *ennreal* :: *ordered-comm-semiring*
⟨*proof*⟩

instance *ennreal* :: *linordered-nonzero-semiring*
⟨*proof*⟩

instance *ennreal* :: *strict-ordered-ab-semigroup-add*
⟨*proof*⟩

declare [[*coercion of-nat* :: *nat* ⇒ *ennreal*]]

lemma *e2ennreal-neg*: *x* ≤ *0* ⟹ *e2ennreal x* = *0*
⟨*proof*⟩

lemma *e2ennreal-mono*: *x* ≤ *y* ⟹ *e2ennreal x* ≤ *e2ennreal y*
⟨*proof*⟩

lemma *enn2ereal-nonneg*[*simp*]: *0* ≤ *enn2ereal x*
⟨*proof*⟩

lemma *ereal-ennreal-cases*:
obtains *b* **where** *0* ≤ *a a* = *enn2ereal b* | *a* < *0*
⟨*proof*⟩

lemma *rel-fun-liminf*[*transfer-rule*]: *rel-fun* (*rel-fun op* = *pcr-ennreal*) *pcr-ennreal liminf liminf*
⟨*proof*⟩

lemma *rel-fun-limsup*[*transfer-rule*]: *rel-fun* (*rel-fun op* = *pcr-ennreal*) *pcr-ennreal limsup limsup*
⟨*proof*⟩

lemma *setsum-enn2ereal*[*simp*]: (⋀*i*. *i* ∈ *I* ⟹ *0* ≤ *f i*) ⟹ (∑ *i*∈*I*. *enn2ereal* (*f i*)) = *enn2ereal* (*setsum f I*)
⟨*proof*⟩

lemma *transfer-e2ennreal-setsum* [*transfer-rule*]:
rel-fun (*rel-fun op* = *pcr-ennreal*) (*rel-fun op* = *pcr-ennreal*) *setsum setsum*

⟨*proof*⟩

lemma *enn2ereal-of-nat*[*simp*]: *enn2ereal* (*of-nat n*) = *ereal n*
⟨*proof*⟩

lemma *enn2ereal-numeral*[*simp*]: *enn2ereal* (*numeral a*) = *numeral a*
⟨*proof*⟩

lemma *transfer-numeral*[*transfer-rule*]: *pcr-ennreal* (*numeral a*) (*numeral a*)
⟨*proof*⟩

43.7 Cancellation simprocs

lemma *ennreal-add-left-cancel*: $a + b = a + c \longleftrightarrow a = (\infty::ennreal) \vee b = c$
⟨*proof*⟩

lemma *ennreal-add-left-cancel-le*: $a + b \leq a + c \longleftrightarrow a = (\infty::ennreal) \vee b \leq c$
⟨*proof*⟩

lemma *ereal-add-left-cancel-less*:
fixes *a b c* :: *ereal*
shows $0 \leq a \Longrightarrow 0 \leq b \Longrightarrow a + b < a + c \longleftrightarrow a \neq \infty \wedge b < c$
⟨*proof*⟩

lemma *ennreal-add-left-cancel-less*: $a + b < a + c \longleftrightarrow a \neq (\infty::ennreal) \wedge b < c$
⟨*proof*⟩

⟨*ML*⟩

43.8 Order with top

lemma *ennreal-zero-less-top*[*simp*]: $0 < (top::ennreal)$
⟨*proof*⟩

lemma *ennreal-one-less-top*[*simp*]: $1 < (top::ennreal)$
⟨*proof*⟩

lemma *ennreal-zero-neq-top*[*simp*]: $0 \neq (top::ennreal)$
⟨*proof*⟩

lemma *ennreal-top-neq-zero*[*simp*]: $(top::ennreal) \neq 0$
⟨*proof*⟩

lemma *ennreal-top-neq-one*[*simp*]: $top \neq (1::ennreal)$
⟨*proof*⟩

lemma *ennreal-one-neq-top*[*simp*]: $1 \neq (top::ennreal)$
⟨*proof*⟩

lemma *ennreal-add-less-top*[*simp*]:
fixes $a\ b :: ennreal$
shows $a + b < top \longleftrightarrow a < top \wedge b < top$
⟨*proof*⟩

lemma *ennreal-add-eq-top*[*simp*]:
fixes $a\ b :: ennreal$
shows $a + b = top \longleftrightarrow a = top \vee b = top$
⟨*proof*⟩

lemma *ennreal-setsum-less-top*[*simp*]:
fixes $f :: 'a \Rightarrow ennreal$
shows $finite\ I \Longrightarrow (\sum i{\in}I.\ f\ i) < top \longleftrightarrow (\forall i{\in}I.\ f\ i < top)$
⟨*proof*⟩

lemma *ennreal-setsum-eq-top*[*simp*]:
fixes $f :: 'a \Rightarrow ennreal$
shows $finite\ I \Longrightarrow (\sum i{\in}I.\ f\ i) = top \longleftrightarrow (\exists i{\in}I.\ f\ i = top)$
⟨*proof*⟩

lemma *ennreal-mult-eq-top-iff*:
fixes $a\ b :: ennreal$
shows $a * b = top \longleftrightarrow (a = top \wedge b \neq 0) \vee (b = top \wedge a \neq 0)$
⟨*proof*⟩

lemma *ennreal-top-eq-mult-iff*:
fixes $a\ b :: ennreal$
shows $top = a * b \longleftrightarrow (a = top \wedge b \neq 0) \vee (b = top \wedge a \neq 0)$
⟨*proof*⟩

lemma *ennreal-mult-less-top*:
fixes $a\ b :: ennreal$
shows $a * b < top \longleftrightarrow (a = 0 \vee b = 0 \vee (a < top \wedge b < top))$
⟨*proof*⟩

lemma *top-power-ennreal*: $top\ \hat{}\ n = (if\ n = 0\ then\ 1\ else\ top :: ennreal)$
⟨*proof*⟩

lemma *ennreal-setprod-eq-0*[*simp*]:
fixes $f :: 'a \Rightarrow ennreal$
shows $(setprod\ f\ A = 0) = (finite\ A \wedge (\exists i{\in}A.\ f\ i = 0))$
⟨*proof*⟩

lemma *ennreal-setprod-eq-top*:
fixes $f :: 'a \Rightarrow ennreal$
shows $(\prod i{\in}I.\ f\ i) = top \longleftrightarrow (finite\ I \wedge ((\forall i{\in}I.\ f\ i \neq 0) \wedge (\exists i{\in}I.\ f\ i = top)))$
⟨*proof*⟩

lemma *ennreal-top-mult*: $top * a = (if\ a = 0\ then\ 0\ else\ top :: ennreal)$

⟨*proof*⟩

lemma *ennreal-mult-top*: $a * top = (if\ a = 0\ then\ 0\ else\ top :: ennreal)$
⟨*proof*⟩

lemma *enn2ereal-eq-top-iff*[*simp*]: $enn2ereal\ x = \infty \longleftrightarrow x = top$
⟨*proof*⟩

lemma *enn2ereal-top*: $enn2ereal\ top = \infty$
⟨*proof*⟩

lemma *e2ennreal-infty*: $e2ennreal\ \infty = top$
⟨*proof*⟩

lemma *ennreal-top-minus*[*simp*]: $top - x = (top::ennreal)$
⟨*proof*⟩

lemma *minus-top-ennreal*: $x - top = (if\ x = top\ then\ top\ else\ 0:: ennreal)$
⟨*proof*⟩

lemma *bot-ennreal*: $bot = (0::ennreal)$
⟨*proof*⟩

lemma *ennreal-of-nat-neq-top*[*simp*]: $of\text{-}nat\ i \neq (top::ennreal)$
⟨*proof*⟩

lemma *numeral-eq-of-nat*: $(numeral\ a::ennreal) = of\text{-}nat\ (numeral\ a)$
⟨*proof*⟩

lemma *of-nat-less-top*: $of\text{-}nat\ i < (top::ennreal)$
⟨*proof*⟩

lemma *top-neq-numeral*[*simp*]: $top \neq (numeral\ i::ennreal)$
⟨*proof*⟩

lemma *ennreal-numeral-less-top*[*simp*]: $numeral\ i < (top::ennreal)$
⟨*proof*⟩

lemma *ennreal-add-bot*[*simp*]: $bot + x = (x::ennreal)$
⟨*proof*⟩

instance *ennreal* :: *semiring-char-0*
⟨*proof*⟩

43.9 Arithmetic

lemma *ennreal-minus-zero*[*simp*]: $a - (0::ennreal) = a$
⟨*proof*⟩

lemma *ennreal-add-diff-cancel-right*[*simp*]:
fixes $x\ y\ z :: ennreal$ **shows** $y \neq top \Longrightarrow (x + y) - y = x$
⟨*proof*⟩

lemma *ennreal-add-diff-cancel-left*[*simp*]:
fixes $x\ y\ z :: ennreal$ **shows** $y \neq top \Longrightarrow (y + x) - y = x$
⟨*proof*⟩

lemma
fixes $a\ b :: ennreal$
shows $a - b = 0 \Longrightarrow a \leq b$
⟨*proof*⟩

lemma *ennreal-minus-cancel*:
fixes $a\ b\ c :: ennreal$
shows $c \neq top \Longrightarrow a \leq c \Longrightarrow b \leq c \Longrightarrow c - a = c - b \Longrightarrow a = b$
⟨*proof*⟩

lemma *sup-const-add-ennreal*:
fixes $a\ b\ c :: ennreal$
shows $sup\ (c + a)\ (c + b) = c + sup\ a\ b$
⟨*proof*⟩

lemma *ennreal-diff-add-assoc*:
fixes $a\ b\ c :: ennreal$
shows $a \leq b \Longrightarrow c + b - a = c + (b - a)$
⟨*proof*⟩

lemma *mult-divide-eq-ennreal*:
fixes $a\ b :: ennreal$
shows $b \neq 0 \Longrightarrow b \neq top \Longrightarrow (a * b) / b = a$
⟨*proof*⟩

lemma *divide-mult-eq*: $a \neq 0 \Longrightarrow a \neq \infty \Longrightarrow x * a / (b * a) = x / (b{::}ennreal)$
⟨*proof*⟩

lemma *ennreal-mult-divide-eq*:
fixes $a\ b :: ennreal$
shows $b \neq 0 \Longrightarrow b \neq top \Longrightarrow (a * b) / b = a$
⟨*proof*⟩

lemma *ennreal-add-diff-cancel*:
fixes $a\ b :: ennreal$
shows $b \neq \infty \Longrightarrow (a + b) - b = a$
⟨*proof*⟩

lemma *ennreal-minus-eq-0*:
$a - b = 0 \Longrightarrow a \leq (b{::}ennreal)$
⟨*proof*⟩

lemma *ennreal-mono-minus-cancel*:
fixes *a b c* :: *ennreal*
shows $a - b \leq a - c \Longrightarrow a < top \Longrightarrow b \leq a \Longrightarrow c \leq a \Longrightarrow c \leq b$
⟨*proof*⟩

lemma *ennreal-mono-minus*:
fixes *a b c* :: *ennreal*
shows $c \leq b \Longrightarrow a - b \leq a - c$
⟨*proof*⟩

lemma *ennreal-minus-pos-iff*:
fixes *a b* :: *ennreal*
shows $a < top \vee b < top \Longrightarrow 0 < a - b \Longrightarrow b < a$
⟨*proof*⟩

lemma *ennreal-inverse-top*[*simp*]: *inverse top* = (*0*::*ennreal*)
⟨*proof*⟩

lemma *ennreal-inverse-zero*[*simp*]: *inverse 0* = (*top*::*ennreal*)
⟨*proof*⟩

lemma *ennreal-top-divide*: *top* / (*x*::*ennreal*) = (*if x* = *top then 0 else top*)
⟨*proof*⟩

lemma *ennreal-zero-divide*[*simp*]: *0* / (*x*::*ennreal*) = *0*
⟨*proof*⟩

lemma *ennreal-divide-zero*[*simp*]: *x* / (*0*::*ennreal*) = (*if x* = *0 then 0 else top*)
⟨*proof*⟩

lemma *ennreal-divide-top*[*simp*]: *x* / (*top*::*ennreal*) = *0*
⟨*proof*⟩

lemma *ennreal-times-divide*: *a* ∗ (*b* / *c*) = *a* ∗ *b* / (*c*::*ennreal*)
⟨*proof*⟩

lemma *ennreal-zero-less-divide*: *0* < *a* / *b* ⟷ (*0* < *a* ∧ *b* < (*top*::*ennreal*))
⟨*proof*⟩

lemma *divide-right-mono-ennreal*:
fixes *a b c* :: *ennreal*
shows $a \leq b \Longrightarrow a$ / $c \leq b$ / c
⟨*proof*⟩

lemma *ennreal-mult-strict-right-mono*: (*a*::*ennreal*) < *c* ⟹ *0* < *b* ⟹ *b* < *top*
⟹ *a* ∗ *b* < *c* ∗ *b*
⟨*proof*⟩

lemma *ennreal-indicator-less*[*simp*]:
indicator A x ≤ (*indicator B x*::*ennreal*) ⟷ (*x* ∈ *A* ⟶ *x* ∈ *B*)
⟨*proof*⟩

lemma *ennreal-inverse-positive*: *0* < *inverse x* ⟷ (*x*::*ennreal*) ≠ *top*
⟨*proof*⟩

lemma *ennreal-inverse-mult'*: ((*0* < *b* ∨ *a* < *top*) ∧ (*0* < *a* ∨ *b* < *top*)) ⟹ *inverse* (*a* * *b*::*ennreal*) = *inverse a* * *inverse b*
⟨*proof*⟩

lemma *ennreal-inverse-mult*: *a* < *top* ⟹ *b* < *top* ⟹ *inverse* (*a* * *b*::*ennreal*) = *inverse a* * *inverse b*
⟨*proof*⟩

lemma *ennreal-inverse-1*[*simp*]: *inverse* (*1*::*ennreal*) = *1*
⟨*proof*⟩

lemma *ennreal-inverse-eq-0-iff*[*simp*]: *inverse* (*a*::*ennreal*) = *0* ⟷ *a* = *top*
⟨*proof*⟩

lemma *ennreal-inverse-eq-top-iff*[*simp*]: *inverse* (*a*::*ennreal*) = *top* ⟷ *a* = *0*
⟨*proof*⟩

lemma *ennreal-divide-eq-0-iff*[*simp*]: (*a*::*ennreal*) / *b* = *0* ⟷ (*a* = *0* ∨ *b* = *top*)
⟨*proof*⟩

lemma *ennreal-divide-eq-top-iff*: (*a*::*ennreal*) / *b* = *top* ⟷ ((*a* ≠ *0* ∧ *b* = *0*) ∨ (*a* = *top* ∧ *b* ≠ *top*))
⟨*proof*⟩

lemma *one-divide-one-divide-ennreal*[*simp*]: *1* / (*1* / *c*) = (*c*::*ennreal*)
including *ennreal.lifting*
⟨*proof*⟩

lemma *ennreal-mult-left-cong*:
((*a*::*ennreal*) ≠ *0* ⟹ *b* = *c*) ⟹ *a* * *b* = *a* * *c*
⟨*proof*⟩

lemma *ennreal-mult-right-cong*:
((*a*::*ennreal*) ≠ *0* ⟹ *b* = *c*) ⟹ *b* * *a* = *c* * *a*
⟨*proof*⟩

lemma *ennreal-zero-less-mult-iff*: *0* < *a* * *b* ⟷ *0* < *a* ∧ *0* < (*b*::*ennreal*)
⟨*proof*⟩

lemma *less-diff-eq-ennreal*:
fixes *a b c* :: *ennreal*
shows *b* < *top* ∨ *c* < *top* ⟹ *a* < *b* − *c* ⟷ *a* + *c* < *b*

⟨*proof*⟩

lemma *diff-add-cancel-ennreal*:
fixes *a b* :: *ennreal* **shows** $a \leq b \Longrightarrow b - a + a = b$
⟨*proof*⟩

lemma *ennreal-diff-self*[*simp*]: $a \neq top \Longrightarrow a - a = (0::ennreal)$
⟨*proof*⟩

lemma *ennreal-minus-mono*:
fixes *a b c* :: *ennreal*
shows $a \leq c \Longrightarrow d \leq b \Longrightarrow a - b \leq c - d$
⟨*proof*⟩

lemma *ennreal-minus-eq-top*[*simp*]: $a - (b::ennreal) = top \longleftrightarrow a = top$
⟨*proof*⟩

lemma *ennreal-divide-self*[*simp*]: $a \neq 0 \Longrightarrow a < top \Longrightarrow a \,/\, a = (1::ennreal)$
⟨*proof*⟩

43.10 Coercion from *real* to *ennreal*

lift-definition *ennreal* :: *real* ⇒ *ennreal* **is** *sup 0* ∘ *ereal*
⟨*proof*⟩

declare [[*coercion ennreal*]]

lemma *ennreal-cases*[*cases type*: *ennreal*]:
fixes *x* :: *ennreal*
obtains (*real*) *r* :: *real* **where** $0 \leq r$ $x = ennreal\ r$ | (*top*) $x = top$
⟨*proof*⟩

lemmas *ennreal2-cases* = *ennreal-cases*[*case-product ennreal-cases*]
lemmas *ennreal3-cases* = *ennreal-cases*[*case-product ennreal2-cases*]

lemma *ennreal-neq-top*[*simp*]: $ennreal\ r \neq top$
⟨*proof*⟩

lemma *top-neq-ennreal*[*simp*]: $top \neq ennreal\ r$
⟨*proof*⟩

lemma *ennreal-less-top*[*simp*]: $ennreal\ x < top$
⟨*proof*⟩

lemma *ennreal-neg*: $x \leq 0 \Longrightarrow ennreal\ x = 0$
⟨*proof*⟩

lemma *ennreal-inj*[*simp*]:
$0 \leq a \Longrightarrow 0 \leq b \Longrightarrow ennreal\ a = ennreal\ b \longleftrightarrow a = b$

⟨*proof*⟩

lemma *ennreal-le-iff*[*simp*]: $0 \leq y \Longrightarrow ennreal\ x \leq ennreal\ y \longleftrightarrow x \leq y$
⟨*proof*⟩

lemma *le-ennreal-iff*: $0 \leq r \Longrightarrow x \leq ennreal\ r \longleftrightarrow (\exists\, q{\geq}0.\ x = ennreal\ q \wedge q \leq r)$
⟨*proof*⟩

lemma *ennreal-less-iff*: $0 \leq r \Longrightarrow ennreal\ r < ennreal\ q \longleftrightarrow r < q$
⟨*proof*⟩

lemma *ennreal-eq-zero-iff*[*simp*]: $0 \leq x \Longrightarrow ennreal\ x = 0 \longleftrightarrow x = 0$
⟨*proof*⟩

lemma *ennreal-less-zero-iff*[*simp*]: $0 < ennreal\ x \longleftrightarrow 0 < x$
⟨*proof*⟩

lemma *ennreal-lessI*: $0 < q \Longrightarrow r < q \Longrightarrow ennreal\ r < ennreal\ q$
⟨*proof*⟩

lemma *ennreal-leI*: $x \leq y \Longrightarrow ennreal\ x \leq ennreal\ y$
⟨*proof*⟩

lemma *enn2ereal-ennreal*[*simp*]: $0 \leq x \Longrightarrow enn2ereal\ (ennreal\ x) = x$
⟨*proof*⟩

lemma *e2ennreal-enn2ereal*[*simp*]: $e2ennreal\ (enn2ereal\ x) = x$
⟨*proof*⟩

lemma *ennreal-0*[*simp*]: $ennreal\ 0 = 0$
⟨*proof*⟩

lemma *ennreal-1*[*simp*]: $ennreal\ 1 = 1$
⟨*proof*⟩

lemma *ennreal-eq-0-iff*: $ennreal\ x = 0 \longleftrightarrow x \leq 0$
⟨*proof*⟩

lemma *ennreal-le-iff2*: $ennreal\ x \leq ennreal\ y \longleftrightarrow ((0 \leq y \wedge x \leq y) \vee (x \leq 0 \wedge y \leq 0))$
⟨*proof*⟩

lemma *ennreal-eq-1*[*simp*]: $ennreal\ x = 1 \longleftrightarrow x = 1$
⟨*proof*⟩

lemma *ennreal-le-1*[*simp*]: $ennreal\ x \leq 1 \longleftrightarrow x \leq 1$
⟨*proof*⟩

lemma *ennreal-ge-1*[*simp*]: *ennreal x* ≥ *1* ⟷ *x* ≥ *1*
⟨*proof*⟩

lemma *ennreal-plus*[*simp*]:
0 ≤ *a* ⟹ *0* ≤ *b* ⟹ *ennreal* (*a* + *b*) = *ennreal a* + *ennreal b*
⟨*proof*⟩

lemma *setsum-ennreal*[*simp*]: (⋀*i*. *i* ∈ *I* ⟹ *0* ≤ *f i*) ⟹ (∑ *i*∈*I*. *ennreal* (*f i*)) = *ennreal* (*setsum f I*)
⟨*proof*⟩

lemma *ennreal-of-nat-eq-real-of-nat*: *of-nat i* = *ennreal* (*of-nat i*)
⟨*proof*⟩

lemma *of-nat-le-ennreal-iff*[*simp*]: *0* ≤ *r* ⟹ *of-nat i* ≤ *ennreal r* ⟷ *of-nat i* ≤ *r*
⟨*proof*⟩

lemma *ennreal-le-of-nat-iff*[*simp*]: *ennreal r* ≤ *of-nat i* ⟷ *r* ≤ *of-nat i*
⟨*proof*⟩

lemma *ennreal-indicator*: *ennreal* (*indicator A x*) = *indicator A x*
⟨*proof*⟩

lemma *ennreal-numeral*[*simp*]: *ennreal* (*numeral n*) = *numeral n*
⟨*proof*⟩

lemma *min-ennreal*: *0* ≤ *x* ⟹ *0* ≤ *y* ⟹ *min* (*ennreal x*) (*ennreal y*) = *ennreal* (*min x y*)
⟨*proof*⟩

lemma *ennreal-half*[*simp*]: *ennreal* (*1*/*2*) = *inverse 2*
⟨*proof*⟩

lemma *ennreal-minus*: *0* ≤ *q* ⟹ *ennreal r* − *ennreal q* = *ennreal* (*r* − *q*)
⟨*proof*⟩

lemma *ennreal-minus-top*[*simp*]: *ennreal a* − *top* = *0*
⟨*proof*⟩

lemma *ennreal-mult*: *0* ≤ *a* ⟹ *0* ≤ *b* ⟹ *ennreal* (*a* ∗ *b*) = *ennreal a* ∗ *ennreal b*
⟨*proof*⟩

lemma *ennreal-mult'*: *0* ≤ *a* ⟹ *ennreal* (*a* ∗ *b*) = *ennreal a* ∗ *ennreal b*
⟨*proof*⟩

lemma *indicator-mult-ennreal*: *indicator A x* ∗ *ennreal r* = *ennreal* (*indicator A x* ∗ *r*)

⟨*proof*⟩

lemma *ennreal-mult''*: $0 \leq b \Longrightarrow ennreal\ (a * b) = ennreal\ a * ennreal\ b$
⟨*proof*⟩

lemma *numeral-mult-ennreal*: $0 \leq x \Longrightarrow numeral\ b * ennreal\ x = ennreal\ (numeral\ b * x)$
⟨*proof*⟩

lemma *ennreal-power*: $0 \leq r \Longrightarrow ennreal\ r\ \hat{}\ n = ennreal\ (r\ \hat{}\ n)$
⟨*proof*⟩

lemma *power-eq-top-ennreal*: $x\ \hat{}\ n = top \longleftrightarrow (n \neq 0 \wedge (x{::}ennreal) = top)$
⟨*proof*⟩

lemma *inverse-ennreal*: $0 < r \Longrightarrow inverse\ (ennreal\ r) = ennreal\ (inverse\ r)$
⟨*proof*⟩

lemma *divide-ennreal*: $0 \leq r \Longrightarrow 0 < q \Longrightarrow ennreal\ r\ /\ ennreal\ q = ennreal\ (r\ /\ q)$
⟨*proof*⟩

lemma *ennreal-inverse-power*: $inverse\ (x\ \hat{}\ n :: ennreal) = inverse\ x\ \hat{}\ n$
⟨*proof*⟩

lemma *ennreal-divide-numeral*: $0 \leq x \Longrightarrow ennreal\ x\ /\ numeral\ b = ennreal\ (x\ /\ numeral\ b)$
⟨*proof*⟩

lemma *setprod-ennreal*: $(\bigwedge i.\ i \in A \Longrightarrow 0 \leq f\ i) \Longrightarrow (\prod i{\in}A.\ ennreal\ (f\ i)) = ennreal\ (setprod\ f\ A)$
⟨*proof*⟩

lemma *mult-right-ennreal-cancel*: $a * ennreal\ c = b * ennreal\ c \longleftrightarrow (a = b \vee c \leq 0)$
⟨*proof*⟩

lemma *ennreal-le-epsilon*:
$(\bigwedge e{::}real.\ y < top \Longrightarrow 0 < e \Longrightarrow x \leq y + ennreal\ e) \Longrightarrow x \leq y$
⟨*proof*⟩

lemma *ennreal-rat-dense*:
fixes $x\ y :: ennreal$
shows $x < y \Longrightarrow \exists\, r{::}rat.\ x < real\text{-}of\text{-}rat\ r \wedge real\text{-}of\text{-}rat\ r < y$
⟨*proof*⟩

lemma *ennreal-Ex-less-of-nat*: $(x{::}ennreal) < top \Longrightarrow \exists\, n.\ x < of\text{-}nat\ n$
⟨*proof*⟩

43.11 Coercion from *ennreal* to *real*

definition *enn2real x = real-of-ereal (enn2ereal x)*

lemma *enn2real-nonneg*[*simp*]: *0 ≤ enn2real x*
⟨*proof*⟩

lemma *enn2real-mono*: *a ≤ b ⟹ b < top ⟹ enn2real a ≤ enn2real b*
⟨*proof*⟩

lemma *enn2real-of-nat*[*simp*]: *enn2real (of-nat n) = n*
⟨*proof*⟩

lemma *enn2real-ennreal*[*simp*]: *0 ≤ r ⟹ enn2real (ennreal r) = r*
⟨*proof*⟩

lemma *ennreal-enn2real*[*simp*]: *r < top ⟹ ennreal (enn2real r) = r*
⟨*proof*⟩

lemma *real-of-ereal-enn2ereal*[*simp*]: *real-of-ereal (enn2ereal x) = enn2real x*
⟨*proof*⟩

lemma *enn2real-top*[*simp*]: *enn2real top = 0*
⟨*proof*⟩

lemma *enn2real-0*[*simp*]: *enn2real 0 = 0*
⟨*proof*⟩

lemma *enn2real-1*[*simp*]: *enn2real 1 = 1*
⟨*proof*⟩

lemma *enn2real-numeral*[*simp*]: *enn2real (numeral n) = (numeral n)*
⟨*proof*⟩

lemma *enn2real-mult*: *enn2real (a * b) = enn2real a * enn2real b*
⟨*proof*⟩

lemma *enn2real-leI*: *0 ≤ B ⟹ x ≤ ennreal B ⟹ enn2real x ≤ B*
⟨*proof*⟩

lemma *enn2real-positive-iff*: *0 < enn2real x ⟷ (0 < x ∧ x < top)*
⟨*proof*⟩

43.12 Coercion from *enat* to *ennreal*

definition *ennreal-of-enat* :: *enat ⇒ ennreal*
where
ennreal-of-enat n = (case n of ∞ ⇒ top | enat n ⇒ of-nat n)

declare [[*coercion ennreal-of-enat*]]

declare [[*coercion of-nat* :: *nat* ⇒ *ennreal*]]

lemma *ennreal-of-enat-infty*[*simp*]: *ennreal-of-enat* ∞ = ∞
⟨*proof*⟩

lemma *ennreal-of-enat-enat*[*simp*]: *ennreal-of-enat* (*enat n*) = *of-nat n*
⟨*proof*⟩

lemma *ennreal-of-enat-0*[*simp*]: *ennreal-of-enat 0* = *0*
⟨*proof*⟩

lemma *ennreal-of-enat-1*[*simp*]: *ennreal-of-enat 1* = *1*
⟨*proof*⟩

lemma *ennreal-top-neq-of-nat*[*simp*]: (*top*::*ennreal*) ≠ *of-nat i*
⟨*proof*⟩

lemma *ennreal-of-enat-inj*[*simp*]: *ennreal-of-enat i* = *ennreal-of-enat j* ⟷ *i* = *j*
⟨*proof*⟩

lemma *ennreal-of-enat-le-iff*[*simp*]: *ennreal-of-enat m* ≤ *ennreal-of-enat n* ⟷ *m* ≤ *n*
⟨*proof*⟩

lemma *of-nat-less-ennreal-of-nat*[*simp*]: *of-nat n* ≤ *ennreal-of-enat x* ⟷ *of-nat n* ≤ *x*
⟨*proof*⟩

lemma *ennreal-of-enat-Sup*: *ennreal-of-enat* (*Sup X*) = (*SUP x*:*X*. *ennreal-of-enat x*)
⟨*proof*⟩

lemma *ennreal-of-enat-eSuc*[*simp*]: *ennreal-of-enat* (*eSuc x*) = *1* + *ennreal-of-enat x*
⟨*proof*⟩

43.13 Topology on *ennreal*

lemma *enn2ereal-Iio*: *enn2ereal* −‘ {..<*a*} = (*if 0* ≤ *a then* {..< *e2ennreal a*} *else* {})
⟨*proof*⟩

lemma *enn2ereal-Ioi*: *enn2ereal* −‘ {*a* <..} = (*if 0* ≤ *a then* {*e2ennreal a* <..} *else UNIV*)
⟨*proof*⟩

instantiation *ennreal* :: *linear-continuum-topology*
begin

definition *open-ennreal* :: *ennreal set* ⇒ *bool*
where (*open* :: *ennreal set* ⇒ *bool*) = *generate-topology* (*range lessThan* ∪ *range greaterThan*)

instance
⟨*proof*⟩

end

lemma *continuous-on-e2ennreal*: *continuous-on A e2ennreal*
⟨*proof*⟩

lemma *continuous-at-e2ennreal*: *continuous* (*at x within A*) *e2ennreal*
⟨*proof*⟩

lemma *continuous-on-enn2ereal*: *continuous-on UNIV enn2ereal*
⟨*proof*⟩

lemma *continuous-at-enn2ereal*: *continuous* (*at x within A*) *enn2ereal*
⟨*proof*⟩

lemma *sup-continuous-e2ennreal*[*order-continuous-intros*]:
assumes *f*: *sup-continuous f* **shows** *sup-continuous* (λ*x*. *e2ennreal* (*f x*))
⟨*proof*⟩

lemma *sup-continuous-enn2ereal*[*order-continuous-intros*]:
assumes *f*: *sup-continuous f* **shows** *sup-continuous* (λ*x*. *enn2ereal* (*f x*))
⟨*proof*⟩

lemma *sup-continuous-mult-left-ennreal'*:
fixes *c* :: *ennreal*
shows *sup-continuous* (λ*x*. *c* ∗ *x*)
⟨*proof*⟩

lemma *sup-continuous-mult-left-ennreal*[*order-continuous-intros*]:
sup-continuous f ⟹ *sup-continuous* (λ*x*. *c* ∗ *f x* :: *ennreal*)
⟨*proof*⟩

lemma *sup-continuous-mult-right-ennreal*[*order-continuous-intros*]:
sup-continuous f ⟹ *sup-continuous* (λ*x*. *f x* ∗ *c* :: *ennreal*)
⟨*proof*⟩

lemma *sup-continuous-divide-ennreal*[*order-continuous-intros*]:
fixes *f g* :: *'a*::*complete-lattice* ⇒ *ennreal*
shows *sup-continuous f* ⟹ *sup-continuous* (λ*x*. *f x* / *c*)
⟨*proof*⟩

lemma *transfer-enn2ereal-continuous-on* [*transfer-rule*]:
rel-fun (*op* =) (*rel-fun* (*rel-fun op* = *pcr-ennreal*) *op* =) *continuous-on continuous-on*

⟨*proof*⟩

lemma *transfer-sup-continuous*[*transfer-rule*]:
(*rel-fun* (*rel-fun* (*op* =) *pcr-ennreal*) *op* =) *sup-continuous sup-continuous*
⟨*proof*⟩

lemma *continuous-on-ennreal*[*tendsto-intros*]:
continuous-on A f ⟹ *continuous-on A* (λ*x*. *ennreal* (*f x*))
⟨*proof*⟩

lemma *tendsto-ennrealD*:
assumes *lim*: ((λ*x*. *ennreal* (*f x*)) ⟶ *ennreal x*) *F*
assumes ∗: ∀ *F x in F*. *0* ≤ *f x* **and** *x*: *0* ≤ *x*
shows (*f* ⟶ *x*) *F*
⟨*proof*⟩

lemma *tendsto-ennreal-iff*[*simp*]:
∀ *F x in F*. *0* ≤ *f x* ⟹ *0* ≤ *x* ⟹ ((λ*x*. *ennreal* (*f x*)) ⟶ *ennreal x*) *F* ⟷ (*f* ⟶ *x*) *F*
⟨*proof*⟩

lemma *tendsto-enn2ereal-iff*[*simp*]: ((λ*i*. *enn2ereal* (*f i*)) ⟶ *enn2ereal x*) *F* ⟷ (*f* ⟶ *x*) *F*
⟨*proof*⟩

lemma *continuous-on-add-ennreal*:
fixes *f g* :: *'a*::*topological-space* ⇒ *ennreal*
shows *continuous-on A f* ⟹ *continuous-on A g* ⟹ *continuous-on A* (λ*x*. *f x* + *g x*)
⟨*proof*⟩

lemma *continuous-on-inverse-ennreal*[*continuous-intros*]:
fixes *f* :: *'a*::*topological-space* ⇒ *ennreal*
shows *continuous-on A f* ⟹ *continuous-on A* (λ*x*. *inverse* (*f x*))
⟨*proof*⟩

instance *ennreal* :: *topological-comm-monoid-add*
⟨*proof*⟩

lemma *sup-continuous-add-ennreal*[*order-continuous-intros*]:
fixes *f g* :: *'a*::*complete-lattice* ⇒ *ennreal*
shows *sup-continuous f* ⟹ *sup-continuous g* ⟹ *sup-continuous* (λ*x*. *f x* + *g x*)
⟨*proof*⟩

lemma *ennreal-suminf-lessD*: (∑ *i*. *f i* :: *ennreal*) < *x* ⟹ *f i* < *x*
⟨*proof*⟩

lemma *sums-ennreal*[*simp*]: (⋀*i*. *0* ≤ *f i*) ⟹ *0* ≤ *x* ⟹ (λ*i*. *ennreal* (*f i*)) *sums*

ennreal x $\longleftrightarrow$ *f sums x*
⟨*proof*⟩

lemma *summable-suminf-not-top*: $(\bigwedge i.\ 0 \le f\ i) \Longrightarrow (\sum i.\ ennreal\ (f\ i)) \neq top \Longrightarrow summable\ f$
⟨*proof*⟩

lemma *suminf-ennreal*[*simp*]:
$(\bigwedge i.\ 0 \le f\ i) \Longrightarrow (\sum i.\ ennreal\ (f\ i)) \neq top \Longrightarrow (\sum i.\ ennreal\ (f\ i)) = ennreal\ (\sum i.\ f\ i)$
⟨*proof*⟩

lemma *sums-enn2ereal*[*simp*]: $(\lambda i.\ enn2ereal\ (f\ i))\ sums\ enn2ereal\ x \longleftrightarrow f\ sums\ x$
⟨*proof*⟩

lemma *suminf-enn2ereal*[*simp*]: $(\sum i.\ enn2ereal\ (f\ i)) = enn2ereal\ (suminf\ f)$
⟨*proof*⟩

lemma *transfer-e2ennreal-suminf* [*transfer-rule*]: *rel-fun* (*rel-fun op* = *pcr-ennreal*) *pcr-ennreal suminf suminf*
⟨*proof*⟩

lemma *ennreal-suminf-cmult*[*simp*]: $(\sum i.\ r * f\ i) = r * (\sum i.\ f\ i::ennreal)$
⟨*proof*⟩

lemma *ennreal-suminf-multc*[*simp*]: $(\sum i.\ f\ i * r) = (\sum i.\ f\ i::ennreal) * r$
⟨*proof*⟩

lemma *ennreal-suminf-divide*[*simp*]: $(\sum i.\ f\ i\ /\ r) = (\sum i.\ f\ i::ennreal)\ /\ r$
⟨*proof*⟩

lemma *ennreal-suminf-neq-top*: $summable\ f \Longrightarrow (\bigwedge i.\ 0 \le f\ i) \Longrightarrow (\sum i.\ ennreal\ (f\ i)) \neq top$
⟨*proof*⟩

lemma *suminf-ennreal-eq*:
$(\bigwedge i.\ 0 \le f\ i) \Longrightarrow f\ sums\ x \Longrightarrow (\sum i.\ ennreal\ (f\ i)) = ennreal\ x$
⟨*proof*⟩

lemma *ennreal-suminf-bound-add*:
fixes $f :: nat \Rightarrow ennreal$
shows $(\bigwedge N.\ (\sum n{<}N.\ f\ n) + y \le x) \Longrightarrow suminf\ f + y \le x$
⟨*proof*⟩

lemma *ennreal-suminf-SUP-eq-directed*:
fixes $f :: 'a \Rightarrow nat \Rightarrow ennreal$
assumes $*$: $\bigwedge N\ i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow finite\ N \Longrightarrow \exists k{\in}I.\ \forall n{\in}N.\ f\ i\ n \le f\ k\ n \wedge f\ j\ n \le f\ k\ n$

shows $(\sum n.\ SUP\ i{:}I.\ f\ i\ n) = (SUP\ i{:}I.\ \sum n.\ f\ i\ n)$
⟨*proof*⟩

lemma *INF-ennreal-add-const*:
fixes $f\ g :: nat \Rightarrow ennreal$
shows $(INF\ i.\ f\ i + c) = (INF\ i.\ f\ i) + c$
⟨*proof*⟩

lemma *INF-ennreal-const-add*:
fixes $f\ g :: nat \Rightarrow ennreal$
shows $(INF\ i.\ c + f\ i) = c + (INF\ i.\ f\ i)$
⟨*proof*⟩

lemma *SUP-mult-left-ennreal*: $c * (SUP\ i{:}I.\ f\ i) = (SUP\ i{:}I.\ c * f\ i :: ennreal)$
⟨*proof*⟩

lemma *SUP-mult-right-ennreal*: $(SUP\ i{:}I.\ f\ i) * c = (SUP\ i{:}I.\ f\ i * c :: ennreal)$
⟨*proof*⟩

lemma *SUP-divide-ennreal*: $(SUP\ i{:}I.\ f\ i) / c = (SUP\ i{:}I.\ f\ i / c :: ennreal)$
⟨*proof*⟩

lemma *ennreal-SUP-of-nat-eq-top*: $(SUP\ x.\ of\text{-}nat\ x :: ennreal) = top$
⟨*proof*⟩

lemma *ennreal-SUP-eq-top*:
fixes $f :: 'a \Rightarrow ennreal$
assumes $\bigwedge n.\ \exists i{\in}I.\ of\text{-}nat\ n \leq f\ i$
shows $(SUP\ i : I.\ f\ i) = top$
⟨*proof*⟩

lemma *ennreal-INF-const-minus*:
fixes $f :: 'a \Rightarrow ennreal$
shows $I \neq \{\} \Longrightarrow (SUP\ x{:}I.\ c - f\ x) = c - (INF\ x{:}I.\ f\ x)$
⟨*proof*⟩

lemma *of-nat-Sup-ennreal*:
assumes $A \neq \{\}$ *bdd-above* A
shows *of-nat* $(Sup\ A) = (SUP\ a{:}A.\ of\text{-}nat\ a :: ennreal)$
⟨*proof*⟩

lemma *ennreal-tendsto-const-minus*:
fixes $g :: 'a \Rightarrow ennreal$
assumes *ae*: $\forall_F\ x\ in\ F.\ g\ x \leq c$
assumes *g*: $((\lambda x.\ c - g\ x) \longrightarrow 0)\ F$
shows $(g \longrightarrow c)\ F$
⟨*proof*⟩

lemma *ennreal-SUP-add*:

fixes $f\ g :: nat \Rightarrow ennreal$
shows $incseq\ f \Longrightarrow incseq\ g \Longrightarrow (SUP\ i.\ f\ i + g\ i) = SUPREMUM\ UNIV\ f + SUPREMUM\ UNIV\ g$
⟨*proof*⟩

lemma *ennreal-SUP-setsum*:
fixes $f :: 'a \Rightarrow nat \Rightarrow ennreal$
shows $(\bigwedge i.\ i \in I \Longrightarrow incseq\ (f\ i)) \Longrightarrow (SUP\ n.\ \sum i{\in}I.\ f\ i\ n) = (\sum i{\in}I.\ SUP\ n.\ f\ i\ n)$
⟨*proof*⟩

lemma *ennreal-liminf-minus*:
fixes $f :: nat \Rightarrow ennreal$
shows $(\bigwedge n.\ f\ n \leq c) \Longrightarrow liminf\ (\lambda n.\ c - f\ n) = c - limsup\ f$
⟨*proof*⟩

lemma *ennreal-continuous-on-cmult*:
$(c{::}ennreal) < top \Longrightarrow continuous\text{-}on\ A\ f \Longrightarrow continuous\text{-}on\ A\ (\lambda x.\ c * f\ x)$
⟨*proof*⟩

lemma *ennreal-tendsto-cmult*:
$(c{::}ennreal) < top \Longrightarrow (f \longrightarrow x)\ F \Longrightarrow ((\lambda x.\ c * f\ x) \longrightarrow c * x)\ F$
⟨*proof*⟩

lemma *tendsto-ennrealI*[*intro*, *simp*]:
$(f \longrightarrow x)\ F \Longrightarrow ((\lambda x.\ ennreal\ (f\ x)) \longrightarrow ennreal\ x)\ F$
⟨*proof*⟩

lemma *ennreal-suminf-minus*:
fixes $f\ g :: nat \Rightarrow ennreal$
shows $(\bigwedge i.\ g\ i \leq f\ i) \Longrightarrow suminf\ f \neq top \Longrightarrow suminf\ g \neq top \Longrightarrow (\sum i.\ f\ i - g\ i) = suminf\ f - suminf\ g$
⟨*proof*⟩

lemma *ennreal-Sup-countable-SUP*:
$A \neq \{\} \Longrightarrow \exists f{::}nat \Rightarrow ennreal.\ incseq\ f \wedge range\ f \subseteq A \wedge Sup\ A = (SUP\ i.\ f\ i)$
⟨*proof*⟩

lemma *ennreal-SUP-countable-SUP*:
$A \neq \{\} \Longrightarrow \exists f{::}nat \Rightarrow ennreal.\ range\ f \subseteq g`A \wedge SUPREMUM\ A\ g = SUPREMUM\ UNIV\ f$
⟨*proof*⟩

lemma *of-nat-tendsto-top-ennreal*: $(\lambda n{::}nat.\ of\text{-}nat\ n :: ennreal) \longrightarrow top$
⟨*proof*⟩

lemma *SUP-sup-continuous-ennreal*:
fixes $f :: ennreal \Rightarrow 'a{::}complete\text{-}lattice$
assumes f: $sup\text{-}continuous\ f$ **and** $I \neq \{\}$

shows $(SUP\ i{:}I.\ f\ (g\ i)) = f\ (SUP\ i{:}I.\ g\ i)$
⟨*proof*⟩

lemma *ennreal-suminf-SUP-eq*:
fixes $f :: nat \Rightarrow nat \Rightarrow ennreal$
shows $(\bigwedge i.\ incseq\ (\lambda n.\ f\ n\ i)) \Longrightarrow (\sum i.\ SUP\ n.\ f\ n\ i) = (SUP\ n.\ \sum i.\ f\ n\ i)$
⟨*proof*⟩

lemma *ennreal-SUP-add-left*:
fixes $c :: ennreal$
shows $I \neq \{\} \Longrightarrow (SUP\ i{:}I.\ f\ i + c) = (SUP\ i{:}I.\ f\ i) + c$
⟨*proof*⟩

lemma *ennreal-SUP-const-minus*:
fixes $f :: 'a \Rightarrow ennreal$
shows $I \neq \{\} \Longrightarrow c < top \Longrightarrow (INF\ x{:}I.\ c - f\ x) = c - (SUP\ x{:}I.\ f\ x)$
⟨*proof*⟩

43.14 Approximation lemmas

lemma *INF-approx-ennreal*:
fixes x::*ennreal* **and** e::*real*
assumes $e > 0$
assumes *INF*: $x = (INF\ i : A.\ f\ i)$
assumes $x \neq \infty$
shows $\exists i \in A.\ f\ i < x + e$
⟨*proof*⟩

lemma *SUP-approx-ennreal*:
fixes x::*ennreal* **and** e::*real*
assumes $e > 0\ A \neq \{\}$
assumes *SUP*: $x = (SUP\ i : A.\ f\ i)$
assumes $x \neq \infty$
shows $\exists i \in A.\ x < f\ i + e$
⟨*proof*⟩

lemma *ennreal-approx-SUP*:
fixes x::*ennreal*
assumes *f-bound*: $\bigwedge i.\ i \in A \Longrightarrow f\ i \leq x$
assumes *approx*: $\bigwedge e.\ (e{::}real) > 0 \Longrightarrow \exists i \in A.\ x \leq f\ i + e$
shows $x = (SUP\ i : A.\ f\ i)$
⟨*proof*⟩

lemma *ennreal-approx-INF*:
fixes x::*ennreal*
assumes *f-bound*: $\bigwedge i.\ i \in A \Longrightarrow x \leq f\ i$
assumes *approx*: $\bigwedge e.\ (e{::}real) > 0 \Longrightarrow \exists i \in A.\ f\ i \leq x + e$
shows $x = (INF\ i : A.\ f\ i)$
⟨*proof*⟩

lemma *ennreal-approx-unit*:
$(\bigwedge a{::}ennreal.\ 0 < a \Longrightarrow a < 1 \Longrightarrow a * z \leq y) \Longrightarrow z \leq y$
⟨*proof*⟩

lemma *suminf-ennreal2*:
$(\bigwedge i.\ 0 \leq f\ i) \Longrightarrow summable\ f \Longrightarrow (\sum i.\ ennreal\ (f\ i)) = ennreal\ (\sum i.\ f\ i)$
⟨*proof*⟩

lemma *less-top-ennreal*: $x < top \longleftrightarrow (\exists r{\geq}0.\ x = ennreal\ r)$
⟨*proof*⟩

lemma *tendsto-top-iff-ennreal*:
fixes $f :: {'a} \Rightarrow ennreal$
shows $(f \longrightarrow top)\ F \longleftrightarrow (\forall l{\geq}0.\ eventually\ (\lambda x.\ ennreal\ l < f\ x)\ F)$
⟨*proof*⟩

lemma *ennreal-tendsto-top-eq-at-top*:
$((\lambda z.\ ennreal\ (f\ z)) \longrightarrow top)\ F \longleftrightarrow (LIM\ z\ F.\ f\ z :> at\text{-}top)$
⟨*proof*⟩

lemma *tendsto-0-if-Limsup-eq-0-ennreal*:
fixes $f :: - \Rightarrow ennreal$
shows $Limsup\ F\ f = 0 \Longrightarrow (f \longrightarrow 0)\ F$
⟨*proof*⟩

lemma *diff-le-self-ennreal*[*simp*]: $a - b \leq (a{::}ennreal)$
⟨*proof*⟩

lemma *ennreal-ineq-diff-add*: $b \leq a \Longrightarrow a = b + (a - b{::}ennreal)$
⟨*proof*⟩

lemma *ennreal-mult-strict-left-mono*: $(a{::}ennreal) < c \Longrightarrow 0 < b \Longrightarrow b < top \Longrightarrow b * a < b * c$
⟨*proof*⟩

lemma *ennreal-between*: $0 < e \Longrightarrow 0 < x \Longrightarrow x < top \Longrightarrow x - e < (x{::}ennreal)$
⟨*proof*⟩

lemma *minus-less-iff-ennreal*: $b < top \Longrightarrow b \leq a \Longrightarrow a - b < c \longleftrightarrow a < c + (b{::}ennreal)$
⟨*proof*⟩

lemma *tendsto-zero-ennreal*:
assumes *ev*: $\bigwedge r.\ 0 < r \Longrightarrow \forall_F\ x\ in\ F.\ f\ x < ennreal\ r$
shows $(f \longrightarrow 0)\ F$
⟨*proof*⟩

lifting-update *ennreal.lifting*

lifting-forget *ennreal.lifting*

end

44 Limits on the Extended real number line

theory *Extended-Real-Limits*
imports
Topology-Euclidean-Space
~~/*src*/*HOL*/*Library*/*Extended-Real*
~~/*src*/*HOL*/*Library*/*Extended-Nonnegative-Real*
~~/*src*/*HOL*/*Library*/*Indicator-Function*
begin

lemma *compact-UNIV*:
compact (*UNIV* :: *'a*::{*complete-linorder*,*linorder-topology*,*second-countable-topology*} *set*)
⟨*proof*⟩

lemma *compact-eq-closed*:
fixes *S* :: *'a*::{*complete-linorder*,*linorder-topology*,*second-countable-topology*} *set*
shows *compact S* ⟷ *closed S*
⟨*proof*⟩

lemma *closed-contains-Sup-cl*:
fixes *S* :: *'a*::{*complete-linorder*,*linorder-topology*,*second-countable-topology*} *set*
assumes *closed S*
and *S* ≠ {}
shows *Sup S* ∈ *S*
⟨*proof*⟩

lemma *closed-contains-Inf-cl*:
fixes *S* :: *'a*::{*complete-linorder*,*linorder-topology*,*second-countable-topology*} *set*
assumes *closed S*
and *S* ≠ {}
shows *Inf S* ∈ *S*
⟨*proof*⟩

instance *ereal* :: *second-countable-topology*
⟨*proof*⟩

This is a copy from *ereal* :: *second-countable-topology*. Maybe find a common super class of topological spaces where the rational numbers are densely embedded ?

instance *ennreal* :: *second-countable-topology*
⟨*proof*⟩

lemma *ereal-open-closed-aux*:

fixes S :: *ereal set*
assumes *open* S
and *closed* S
and S: $(-\infty) \notin S$
shows $S = \{\}$
⟨*proof*⟩

lemma *ereal-open-closed*:
fixes S :: *ereal set*
shows *open* $S \wedge$ *closed* $S \longleftrightarrow S = \{\} \vee S =$ *UNIV*
⟨*proof*⟩

lemma *ereal-open-atLeast*:
fixes x :: *ereal*
shows *open* $\{x..\} \longleftrightarrow x = -\infty$
⟨*proof*⟩

lemma *mono-closed-real*:
fixes S :: *real set*
assumes *mono*: $\forall\, y\ z.\ y \in S \wedge y \leq z \longrightarrow z \in S$
and *closed* S
shows $S = \{\} \vee S =$ *UNIV* $\vee\ (\exists\, a.\ S = \{a..\})$
⟨*proof*⟩

lemma *mono-closed-ereal*:
fixes S :: *real set*
assumes *mono*: $\forall\, y\ z.\ y \in S \wedge y \leq z \longrightarrow z \in S$
and *closed* S
shows $\exists\, a.\ S = \{x.\ a \leq$ *ereal* $x\}$
⟨*proof*⟩

lemma *Liminf-within*:
fixes f :: *'a*::*metric-space* ⇒ *'b*::*complete-lattice*
shows *Liminf* (*at* x *within* S) $f =$ (*SUP* e:{*0*<..}. *INF* y:($S \cap$ *ball* $x\ e - \{x\}$). $f\ y$)
⟨*proof*⟩

lemma *Limsup-within*:
fixes f :: *'a*::*metric-space* ⇒ *'b*::*complete-lattice*
shows *Limsup* (*at* x *within* S) $f =$ (*INF* e:{*0*<..}. *SUP* y:($S \cap$ *ball* $x\ e - \{x\}$). $f\ y$)
⟨*proof*⟩

lemma *Liminf-at*:
fixes f :: *'a*::*metric-space* ⇒ *'b*::*complete-lattice*
shows *Liminf* (*at* x) $f =$ (*SUP* e:{*0*<..}. *INF* y:(*ball* $x\ e - \{x\}$). $f\ y$)
⟨*proof*⟩

lemma *Limsup-at*:

fixes $f :: 'a$::*metric-space* $\Rightarrow$ $'b$::*complete-lattice*
shows *Limsup* (*at x*) *f* = (*INF e*:{*0*<..}. *SUP y*:(*ball x e* − {*x*}). *f y*)
⟨*proof*⟩

lemma *min-Liminf-at*:
fixes $f :: 'a$::*metric-space* $\Rightarrow$ $'b$::*complete-linorder*
shows *min* (*f x*) (*Liminf* (*at x*) *f*) = (*SUP e*:{*0*<..}. *INF y*:*ball x e*. *f y*)
⟨*proof*⟩

44.1 monoset

definition (**in** *order*) *mono-set*:
mono-set S ⟷ (∀ *x y*. *x* ≤ *y* ⟶ *x* ∈ *S* ⟶ *y* ∈ *S*)

lemma (**in** *order*) *mono-greaterThan* [*intro*, *simp*]: *mono-set* {*B*<..} ⟨*proof*⟩
lemma (**in** *order*) *mono-atLeast* [*intro*, *simp*]: *mono-set* {*B*..} ⟨*proof*⟩
lemma (**in** *order*) *mono-UNIV* [*intro*, *simp*]: *mono-set UNIV* ⟨*proof*⟩
lemma (**in** *order*) *mono-empty* [*intro*, *simp*]: *mono-set* {} ⟨*proof*⟩

lemma (**in** *complete-linorder*) *mono-set-iff*:
fixes *S* :: *'a set*
defines *a* ≡ *Inf S*
shows *mono-set S* ⟷ *S* = {*a* <..} ∨ *S* = {*a*..} (**is** - = *?c*)
⟨*proof*⟩

lemma *ereal-open-mono-set*:
fixes *S* :: *ereal set*
shows *open S* ∧ *mono-set S* ⟷ *S* = *UNIV* ∨ *S* = {*Inf S* <..}
⟨*proof*⟩

lemma *ereal-closed-mono-set*:
fixes *S* :: *ereal set*
shows *closed S* ∧ *mono-set S* ⟷ *S* = {} ∨ *S* = {*Inf S* ..}
⟨*proof*⟩

lemma *ereal-Liminf-Sup-monoset*:
fixes *f* :: *'a* ⇒ *ereal*
shows *Liminf net f* =
Sup {*l*. ∀ *S*. *open S* ⟶ *mono-set S* ⟶ *l* ∈ *S* ⟶ *eventually* (λ*x*. *f x* ∈ *S*) *net*}
(**is** - = *Sup ?A*)
⟨*proof*⟩

lemma *ereal-Limsup-Inf-monoset*:
fixes *f* :: *'a* ⇒ *ereal*
shows *Limsup net f* =
Inf {*l*. ∀ *S*. *open S* ⟶ *mono-set* (*uminus* ‘ *S*) ⟶ *l* ∈ *S* ⟶ *eventually* (λ*x*. *f x* ∈ *S*) *net*}
(**is** - = *Inf ?A*)

⟨*proof*⟩

lemma *liminf-bounded-open*:
fixes $x :: nat \Rightarrow ereal$
shows $x0 \leq liminf\ x \longleftrightarrow (\forall S.\ open\ S \longrightarrow mono\text{-}set\ S \longrightarrow x0 \in S \longrightarrow (\exists N.\ \forall n{\geq}N.\ x\ n \in S))$
(**is** - $\longleftrightarrow$ *?P x0*)
⟨*proof*⟩

44.2 Relate extended reals and the indicator function

lemma *ereal-indicator-le-0*: $(indicator\ S\ x{::}ereal) \leq 0 \longleftrightarrow x \notin S$
⟨*proof*⟩

lemma *ereal-indicator*: $ereal\ (indicator\ A\ x) = indicator\ A\ x$
⟨*proof*⟩

lemma *ereal-mult-indicator*: $ereal\ (x * indicator\ A\ y) = ereal\ x * indicator\ A\ y$
⟨*proof*⟩

lemma *ereal-indicator-mult*: $ereal\ (indicator\ A\ y * x) = indicator\ A\ y * ereal\ x$
⟨*proof*⟩

lemma *ereal-indicator-nonneg*[*simp*, *intro*]: $0 \leq (indicator\ A\ x :: ereal)$
⟨*proof*⟩

lemma *indicator-inter-arith-ereal*: $indicator\ A\ x * indicator\ B\ x = (indicator\ (A \cap B)\ x :: ereal)$
⟨*proof*⟩

end

45 Permutations, both general and specifically on finite sets.

theory *Permutations*
imports *Binomial*
begin

45.1 Transpositions

lemma *swap-id-idempotent* [*simp*]:
$Fun.swap\ a\ b\ id \circ Fun.swap\ a\ b\ id = id$
⟨*proof*⟩

lemma *inv-swap-id*:
$inv\ (Fun.swap\ a\ b\ id) = Fun.swap\ a\ b\ id$
⟨*proof*⟩

lemma *swap-id-eq*:
Fun.swap a b id x = (*if x* = *a then b else if x* = *b then a else x*)
⟨*proof*⟩

45.2 Basic consequences of the definition

definition *permutes* (**infixr** *permutes 41*)
where (*p permutes S*) ⟷ (∀ *x*. *x* ∉ *S* ⟶ *p x* = *x*) ∧ (∀ *y*. ∃!*x*. *p x* = *y*)

lemma *permutes-in-image*: *p permutes S* ⟹ *p x* ∈ *S* ⟷ *x* ∈ *S*
⟨*proof*⟩

lemma *permutes-image*: *p permutes S* ⟹ *p* ' *S* = *S*
⟨*proof*⟩

lemma *permutes-inj*: *p permutes S* ⟹ *inj p*
⟨*proof*⟩

lemma *permutes-surj*: *p permutes s* ⟹ *surj p*
⟨*proof*⟩

lemma *permutes-bij*: *p permutes s* ⟹ *bij p*
⟨*proof*⟩

lemma *permutes-imp-bij*: *p permutes S* ⟹ *bij-betw p S S*
⟨*proof*⟩

lemma *bij-imp-permutes*: *bij-betw p S S* ⟹ (⋀*x*. *x* ∉ *S* ⟹ *p x* = *x*) ⟹ *p permutes S*
⟨*proof*⟩

lemma *permutes-inv-o*:
assumes *pS*: *p permutes S*
shows *p* ∘ *inv p* = *id*
and *inv p* ∘ *p* = *id*
⟨*proof*⟩

lemma *permutes-inverses*:
fixes *p* :: *'a* ⇒ *'a*
assumes *pS*: *p permutes S*
shows *p* (*inv p x*) = *x*
and *inv p* (*p x*) = *x*
⟨*proof*⟩

lemma *permutes-subset*: *p permutes S* ⟹ *S* ⊆ *T* ⟹ *p permutes T*
⟨*proof*⟩

lemma *permutes-empty*[*simp*]: *p permutes* {} ⟷ *p* = *id*
⟨*proof*⟩

lemma *permutes-sing*[*simp*]: *p permutes* $\{a\} \longleftrightarrow p = id$
⟨*proof*⟩

lemma *permutes-univ*: *p permutes UNIV* $\longleftrightarrow (\forall y.\ \exists!x.\ p\ x = y)$
⟨*proof*⟩

lemma *permutes-inv-eq*: *p permutes S* $\Longrightarrow$ *inv p y* $= x \longleftrightarrow p\ x = y$
⟨*proof*⟩

lemma *permutes-swap-id*: $a \in S \Longrightarrow b \in S \Longrightarrow$ *Fun.swap a b id permutes S*
⟨*proof*⟩

lemma *permutes-superset*: *p permutes S* $\Longrightarrow (\forall x \in S - T.\ p\ x = x) \Longrightarrow$ *p permutes T*
⟨*proof*⟩

45.3 Group properties

lemma *permutes-id*: *id permutes S*
⟨*proof*⟩

lemma *permutes-compose*: *p permutes S* $\Longrightarrow$ *q permutes S* $\Longrightarrow$ $q \circ p$ *permutes S*
⟨*proof*⟩

lemma *permutes-inv*:
assumes *pS*: *p permutes S*
shows *inv p permutes S*
⟨*proof*⟩

lemma *permutes-inv-inv*:
assumes *pS*: *p permutes S*
shows *inv* (*inv p*) $= p$
⟨*proof*⟩

45.4 The number of permutations on a finite set

lemma *permutes-insert-lemma*:
assumes *pS*: *p permutes* (*insert a S*)
shows *Fun.swap a* (*p a*) *id* $\circ$ *p permutes S*
⟨*proof*⟩

lemma *permutes-insert*: $\{p.\ p$ *permutes* (*insert a S*)$\} =$
$(\lambda(b,p).$ *Fun.swap a b id* $\circ\ p)$ ‘ $\{(b,p).\ b \in$ *insert a S* $\wedge\ p \in \{p.\ p$ *permutes S*$\}\}$
⟨*proof*⟩

lemma *card-permutations*:
assumes *Sn*: *card S* $= n$
and *fS*: *finite S*
shows *card* $\{p.\ p$ *permutes S*$\}$ $=$ *fact n*

⟨*proof*⟩

lemma *finite-permutations*:
assumes *fS*: *finite S*
shows *finite* {*p*. *p permutes S*}
⟨*proof*⟩

45.5 Permutations of index set for iterated operations

lemma (**in** *comm-monoid-set*) *permute*:
assumes *p permutes S*
shows *F g S* = *F* (*g* ∘ *p*) *S*
⟨*proof*⟩

45.6 Various combinations of transpositions with 2, 1 and 0 common elements

lemma *swap-id-common*: *a* ≠ *c* ⟹ *b* ≠ *c* ⟹
Fun.swap a b id ∘ *Fun.swap a c id* = *Fun.swap b c id* ∘ *Fun.swap a b id*
⟨*proof*⟩

lemma *swap-id-common'*: *a* ≠ *b* ⟹ *a* ≠ *c* ⟹
Fun.swap a c id ∘ *Fun.swap b c id* = *Fun.swap b c id* ∘ *Fun.swap a b id*
⟨*proof*⟩

lemma *swap-id-independent*: *a* ≠ *c* ⟹ *a* ≠ *d* ⟹ *b* ≠ *c* ⟹ *b* ≠ *d* ⟹
Fun.swap a b id ∘ *Fun.swap c d id* = *Fun.swap c d id* ∘ *Fun.swap a b id*
⟨*proof*⟩

45.7 Permutations as transposition sequences

inductive *swapidseq* :: *nat* ⇒ ('*a* ⇒ '*a*) ⇒ *bool*
where
id[*simp*]: *swapidseq 0 id*
| *comp-Suc*: *swapidseq n p* ⟹ *a* ≠ *b* ⟹ *swapidseq* (*Suc n*) (*Fun.swap a b id* ∘ *p*)

declare *id*[*unfolded id-def*, *simp*]

definition *permutation p* ⟷ (∃ *n*. *swapidseq n p*)

45.8 Some closure properties of the set of permutations, with lengths

lemma *permutation-id*[*simp*]: *permutation id*
⟨*proof*⟩

declare *permutation-id*[*unfolded id-def*, *simp*]

lemma *swapidseq-swap*: *swapidseq* (*if a* = *b then 0 else 1*) (*Fun.swap a b id*)

⟨*proof*⟩

lemma *permutation-swap-id*: *permutation* (*Fun.swap a b id*)
⟨*proof*⟩

lemma *swapidseq-comp-add*: *swapidseq n p* ⟹ *swapidseq m q* ⟹ *swapidseq* (*n* + *m*) (*p* ∘ *q*)
⟨*proof*⟩

lemma *permutation-compose*: *permutation p* ⟹ *permutation q* ⟹ *permutation* (*p* ∘ *q*)
⟨*proof*⟩

lemma *swapidseq-endswap*: *swapidseq n p* ⟹ *a* ≠ *b* ⟹ *swapidseq* (*Suc n*) (*p* ∘ *Fun.swap a b id*)
⟨*proof*⟩

lemma *swapidseq-inverse-exists*: *swapidseq n p* ⟹ ∃ *q*. *swapidseq n q* ∧ *p* ∘ *q* = *id* ∧ *q* ∘ *p* = *id*
⟨*proof*⟩

lemma *swapidseq-inverse*:
assumes *H*: *swapidseq n p*
shows *swapidseq n* (*inv p*)
⟨*proof*⟩

lemma *permutation-inverse*: *permutation p* ⟹ *permutation* (*inv p*)
⟨*proof*⟩

45.9 The identity map only has even transposition sequences

lemma *symmetry-lemma*:
assumes ⋀*a b c d*. *P a b c d* ⟹ *P a b d c*
and ⋀*a b c d*. *a* ≠ *b* ⟹ *c* ≠ *d* ⟹
a = *c* ∧ *b* = *d* ∨ *a* = *c* ∧ *b* ≠ *d* ∨ *a* ≠ *c* ∧ *b* = *d* ∨ *a* ≠ *c* ∧ *a* ≠ *d* ∧ *b* ≠ *c* ∧ *b* ≠ *d* ⟹
P a b c d
shows ⋀*a b c d*. *a* ≠ *b* ⟶ *c* ≠ *d* ⟶ *P a b c d*
⟨*proof*⟩

lemma *swap-general*: *a* ≠ *b* ⟹ *c* ≠ *d* ⟹
Fun.swap a b id ∘ *Fun.swap c d id* = *id* ∨
(∃ *x y z*. *x* ≠ *a* ∧ *y* ≠ *a* ∧ *z* ≠ *a* ∧ *x* ≠ *y* ∧
Fun.swap a b id ∘ *Fun.swap c d id* = *Fun.swap x y id* ∘ *Fun.swap a z id*)
⟨*proof*⟩

lemma *swapidseq-id-iff* [*simp*]: *swapidseq 0 p* ⟷ *p* = *id*
⟨*proof*⟩

lemma *swapidseq-cases*: *swapidseq n p* $\longleftrightarrow$
n = 0 ∧ p = id ∨ (∃ a b q m. n = Suc m ∧ p = Fun.swap a b id ∘ q ∧ swapidseq m q ∧ a ≠ b)
⟨*proof*⟩

lemma *fixing-swapidseq-decrease*:
assumes *spn*: *swapidseq n p*
and *ab*: *a ≠ b*
and *pa*: *(Fun.swap a b id ∘ p) a = a*
shows *n ≠ 0 ∧ swapidseq (n − 1) (Fun.swap a b id ∘ p)*
⟨*proof*⟩

lemma *swapidseq-identity-even*:
assumes *swapidseq n (id :: 'a ⇒ 'a)*
shows *even n*
⟨*proof*⟩

45.10 Therefore we have a welldefined notion of parity

definition *evenperm p = even (SOME n. swapidseq n p)*

lemma *swapidseq-even-even*:
assumes *m*: *swapidseq m p*
and *n*: *swapidseq n p*
shows *even m* $\longleftrightarrow$ *even n*
⟨*proof*⟩

lemma *evenperm-unique*:
assumes *p*: *swapidseq n p*
and *n*:*even n = b*
shows *evenperm p = b*
⟨*proof*⟩

45.11 And it has the expected composition properties

lemma *evenperm-id*[*simp*]: *evenperm id = True*
⟨*proof*⟩

lemma *evenperm-swap*: *evenperm (Fun.swap a b id) = (a = b)*
⟨*proof*⟩

lemma *evenperm-comp*:
assumes *p*: *permutation p*
and *q*:*permutation q*
shows *evenperm (p ∘ q) = (evenperm p = evenperm q)*
⟨*proof*⟩

lemma *evenperm-inv*:
assumes *p*: *permutation p*
shows *evenperm (inv p) = evenperm p*

⟨*proof*⟩

45.12 A more abstract characterization of permutations

lemma *bij-iff*: *bij f* ⟷ (∀ *x*. ∃!*y*. *f y* = *x*)
⟨*proof*⟩

lemma *permutation-bijective*:
assumes *p*: *permutation p*
shows *bij p*
⟨*proof*⟩

lemma *permutation-finite-support*:
assumes *p*: *permutation p*
shows *finite* {*x*. *p x* ≠ *x*}
⟨*proof*⟩

lemma *bij-inv-eq-iff*: *bij p* ⟹ *x* = *inv p y* ⟷ *p x* = *y*
⟨*proof*⟩

lemma *bij-swap-comp*:
assumes *bp*: *bij p*
shows *Fun.swap a b id* ∘ *p* = *Fun.swap* (*inv p a*) (*inv p b*) *p*
⟨*proof*⟩

lemma *bij-swap-ompose-bij*: *bij p* ⟹ *bij* (*Fun.swap a b id* ∘ *p*)
⟨*proof*⟩

lemma *permutation-lemma*:
assumes *fS*: *finite S*
and *p*: *bij p*
and *pS*: ∀ *x*. *x*∉ *S* ⟶ *p x* = *x*
shows *permutation p*
⟨*proof*⟩

lemma *permutation*: *permutation p* ⟷ *bij p* ∧ *finite* {*x*. *p x* ≠ *x*}
(**is** *?lhs* ⟷ *?b* ∧ *?f*)
⟨*proof*⟩

lemma *permutation-inverse-works*:
assumes *p*: *permutation p*
shows *inv p* ∘ *p* = *id*
and *p* ∘ *inv p* = *id*
⟨*proof*⟩

lemma *permutation-inverse-compose*:
assumes *p*: *permutation p*
and *q*: *permutation q*
shows *inv* (*p* ∘ *q*) = *inv q* ∘ *inv p*

⟨*proof*⟩

45.13 Relation to "permutes"

lemma *permutation-permutes*: *permutation p* ⟷ (∃ *S*. *finite S* ∧ *p permutes S*)
⟨*proof*⟩

45.14 Hence a sort of induction principle composing by swaps

lemma *permutes-induct*: *finite S* ⟹ *P id* ⟹
(⋀ *a b p*. *a* ∈ *S* ⟹ *b* ∈ *S* ⟹ *P p* ⟹ *P p* ⟹ *permutation p* ⟹ *P* (*Fun.swap a b id* ∘ *p*)) ⟹
(⋀*p*. *p permutes S* ⟹ *P p*)
⟨*proof*⟩

45.15 Sign of a permutation as a real number

definition *sign p* = (*if evenperm p then* (*1*::*int*) *else* −*1*)

lemma *sign-nz*: *sign p* ≠ *0*
⟨*proof*⟩

lemma *sign-id*: *sign id* = *1*
⟨*proof*⟩

lemma *sign-inverse*: *permutation p* ⟹ *sign* (*inv p*) = *sign p*
⟨*proof*⟩

lemma *sign-compose*: *permutation p* ⟹ *permutation q* ⟹ *sign* (*p* ∘ *q*) = *sign p* ∗ *sign q*
⟨*proof*⟩

lemma *sign-swap-id*: *sign* (*Fun.swap a b id*) = (*if a* = *b then 1 else* −*1*)
⟨*proof*⟩

lemma *sign-idempotent*: *sign p* ∗ *sign p* = *1*
⟨*proof*⟩

45.16 More lemmas about permutations

lemma *permutes-natset-le*:
fixes *S* :: *'a*::*wellorder set*
assumes *p*: *p permutes S*
and *le*: ∀ *i* ∈ *S*. *p i* ≤ *i*
shows *p* = *id*
⟨*proof*⟩

lemma *permutes-natset-ge*:
fixes *S* :: *'a*::*wellorder set*
assumes *p*: *p permutes S*

and *le*: $\forall\, i \in S.\ p\ i \geq i$
shows $p = id$
⟨*proof*⟩

lemma *image-inverse-permutations*: $\{inv\ p\ |p.\ p\ permutes\ S\} = \{p.\ p\ permutes\ S\}$
⟨*proof*⟩

lemma *image-compose-permutations-left*:
assumes *q*: *q permutes S*
shows $\{q \circ p \mid p.\ p\ permutes\ S\} = \{p\ .\ p\ permutes\ S\}$
⟨*proof*⟩

lemma *image-compose-permutations-right*:
assumes *q*: *q permutes S*
shows $\{p \circ q \mid p.\ p\ permutes\ S\} = \{p\ .\ p\ permutes\ S\}$
⟨*proof*⟩

lemma *permutes-in-seg*: *p permutes* $\{1\ ..n\} \Longrightarrow i \in \{1..n\} \Longrightarrow 1 \leq p\ i \wedge p\ i \leq n$
⟨*proof*⟩

lemma *setsum-permutations-inverse*:
$setsum\ f\ \{p.\ p\ permutes\ S\} = setsum\ (\lambda p.\ f(inv\ p))\ \{p.\ p\ permutes\ S\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *setum-permutations-compose-left*:
assumes *q*: *q permutes S*
shows $setsum\ f\ \{p.\ p\ permutes\ S\} = setsum\ (\lambda p.\ f(q \circ p))\ \{p.\ p\ permutes\ S\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *sum-permutations-compose-right*:
assumes *q*: *q permutes S*
shows $setsum\ f\ \{p.\ p\ permutes\ S\} = setsum\ (\lambda p.\ f(p \circ q))\ \{p.\ p\ permutes\ S\}$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

45.17 Sum over a set of permutations (could generalize to iteration)

lemma *setsum-over-permutations-insert*:
assumes *fS*: *finite S*
and *aS*: $a \notin S$
shows $setsum\ f\ \{p.\ p\ permutes\ (insert\ a\ S)\} =$
$setsum\ (\lambda b.\ setsum\ (\lambda q.\ f\ (Fun.swap\ a\ b\ id \circ q))\ \{p.\ p\ permutes\ S\})\ (insert\ a\ S)$
⟨*proof*⟩

end

46 Traces, Determinant of square matrices and some properties

theory *Determinants*
imports
Cartesian-Euclidean-Space
~~/src/HOL/Library/Permutations
begin

46.1 Trace

definition *trace* :: *'a::semiring-1^'n^'n* ⇒ *'a*
where *trace A* = *setsum* (λ*i*. ((*A$i*)*$i*)) (*UNIV*::*'n set*)

lemma *trace-0*: *trace* (*mat 0*) = *0*
⟨*proof*⟩

lemma *trace-I*: *trace* (*mat 1* :: *'a::semiring-1^'n^'n*) = *of-nat*(*CARD*(*'n*))
⟨*proof*⟩

lemma *trace-add*: *trace* ((*A*::*'a::comm-semiring-1^'n^'n*) + *B*) = *trace A* + *trace B*
⟨*proof*⟩

lemma *trace-sub*: *trace* ((*A*::*'a::comm-ring-1^'n^'n*) − *B*) = *trace A* − *trace B*
⟨*proof*⟩

lemma *trace-mul-sym*: *trace* ((*A*::*'a::comm-semiring-1^'n^'m*) ** *B*) = *trace* (*B***A*)
⟨*proof*⟩

Definition of determinant.

definition *det*:: *'a::comm-ring-1^'n^'n* ⇒ *'a* **where**
det A =
setsum (λ*p*. *of-int* (*sign p*) * *setprod* (λ*i*. *Aip i*) (*UNIV* :: *'n set*))
{*p*. *p permutes* (*UNIV* :: *'n set*)}

A few general lemmas we need below.

lemma *setprod-permute*:
assumes *p*: *p permutes S*
shows *setprod f S* = *setprod* (*f* ∘ *p*) *S*
⟨*proof*⟩

lemma *setproduct-permute-nat-interval*:
fixes *m n* :: *nat*
shows *p permutes* {*m..n*} ⟹ *setprod f* {*m..n*} = *setprod* (*f* ∘ *p*) {*m..n*}

⟨*proof*⟩

Basic determinant properties.

lemma *det-transpose*: *det* (*transpose A*) = *det* (*A*::*'a*::*comm-ring-1* ^*'n*^*'n*)
⟨*proof*⟩

lemma *det-lowerdiagonal*:
fixes *A* :: *'a*::*comm-ring-1*^(*'n*::{*finite*,*wellorder*})^(*'n*::{*finite*,*wellorder*})
assumes *ld*: ⋀*i j*. *i* < *j* ⟹ *A*$*i*$*j* = *0*
shows *det A* = *setprod* (λ*i*. *A*$*i*$*i*) (*UNIV*:: *'n set*)
⟨*proof*⟩

lemma *det-upperdiagonal*:
fixes *A* :: *'a*::*comm-ring-1*^*'n*::{*finite*,*wellorder*}^*'n*::{*finite*,*wellorder*}
assumes *ld*: ⋀*i j*. *i* > *j* ⟹ *A*$*i*$*j* = *0*
shows *det A* = *setprod* (λ*i*. *A*$*i*$*i*) (*UNIV*:: *'n set*)
⟨*proof*⟩

lemma *det-diagonal*:
fixes *A* :: *'a*::*comm-ring-1*^*'n*^*'n*
assumes *ld*: ⋀*i j*. *i* ≠ *j* ⟹ *A*$*i*$*j* = *0*
shows *det A* = *setprod* (λ*i*. *A*$*i*$*i*) (*UNIV*::*'n set*)
⟨*proof*⟩

lemma *det-I*: *det* (*mat 1* :: *'a*::*comm-ring-1*^*'n*^*'n*) = *1*
⟨*proof*⟩

lemma *det-0*: *det* (*mat 0* :: *'a*::*comm-ring-1*^*'n*^*'n*) = *0*
⟨*proof*⟩

lemma *det-permute-rows*:
fixes *A* :: *'a*::*comm-ring-1*^*'n*^*'n*
assumes *p*: *p permutes* (*UNIV* :: *'n*::*finite set*)
shows *det* (χ *i*. *A*$*p i* :: *'a*^*'n*^*'n*) = *of-int* (*sign p*) ∗ *det A*
⟨*proof*⟩

lemma *det-permute-columns*:
fixes *A* :: *'a*::*comm-ring-1*^*'n*^*'n*
assumes *p*: *p permutes* (*UNIV* :: *'n set*)
shows *det*(χ *i j*. *A*$*i*$ *p j* :: *'a*^*'n*^*'n*) = *of-int* (*sign p*) ∗ *det A*
⟨*proof*⟩

lemma *det-identical-rows*:
fixes *A* :: *'a*::*linordered-idom*^*'n*^*'n*
assumes *ij*: *i* ≠ *j*
and *r*: *row i A* = *row j A*
shows *det A* = *0*
⟨*proof*⟩

lemma *det-identical-columns*:
fixes $A :: 'a::linordered\text{-}idom\hat{}'n\hat{}'n$
assumes *ij*: $i \neq j$
and *r*: $column\ i\ A = column\ j\ A$
shows $det\ A = 0$
⟨*proof*⟩

lemma *det-zero-row*:
fixes $A :: 'a::\{idom, ring\text{-}char\text{-}0\}\hat{}'n\hat{}'n$
assumes *r*: $row\ i\ A = 0$
shows $det\ A = 0$
⟨*proof*⟩

lemma *det-zero-column*:
fixes $A :: 'a::\{idom,ring\text{-}char\text{-}0\}\hat{}'n\hat{}'n$
assumes *r*: $column\ i\ A = 0$
shows $det\ A = 0$
⟨*proof*⟩

lemma *det-row-add*:
fixes $a\ b\ c :: 'n::finite \Rightarrow - \hat{}\ 'n$
shows $det((\chi\ i.\ if\ i = k\ then\ a\ i + b\ i\ else\ c\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n) =$
$det((\chi\ i.\ if\ i = k\ then\ a\ i\ else\ c\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n) +$
$det((\chi\ i.\ if\ i = k\ then\ b\ i\ else\ c\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n)$
⟨*proof*⟩

lemma *det-row-mul*:
fixes $a\ b :: 'n::finite \Rightarrow - \hat{}\ 'n$
shows $det((\chi\ i.\ if\ i = k\ then\ c *s a\ i\ else\ b\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n) =$
$c * det((\chi\ i.\ if\ i = k\ then\ a\ i\ else\ b\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n)$
⟨*proof*⟩

lemma *det-row-0*:
fixes $b :: 'n::finite \Rightarrow - \hat{}\ 'n$
shows $det((\chi\ i.\ if\ i = k\ then\ 0\ else\ b\ i)::'a::comm\text{-}ring\text{-}1\hat{}'n\hat{}'n) = 0$
⟨*proof*⟩

lemma *det-row-operation*:
fixes $A :: 'a::linordered\text{-}idom\hat{}'n\hat{}'n$
assumes *ij*: $i \neq j$
shows $det\ (\chi\ k.\ if\ k = i\ then\ row\ i\ A + c *s row\ j\ A\ else\ row\ k\ A) = det\ A$
⟨*proof*⟩

lemma *det-row-span*:
fixes $A :: real\hat{}'n\hat{}'n$
assumes *x*: $x \in span\ \{row\ j\ A\ |j.\ j \neq i\}$
shows $det\ (\chi\ k.\ if\ k = i\ then\ row\ i\ A + x\ else\ row\ k\ A) = det\ A$
⟨*proof*⟩

May as well do this, though it's a bit unsatisfactory since it ignores exact

duplicates by considering the rows/columns as a set.

lemma *det-dependent-rows*:
fixes *A*:: *real^'n^'n*
assumes *d*: *dependent* (*rows A*)
shows *det A = 0*
⟨*proof*⟩

lemma *det-dependent-columns*:
assumes *d*: *dependent* (*columns* (*A*::*real^'n^'n*))
shows *det A = 0*
⟨*proof*⟩

Multilinearity and the multiplication formula.

lemma *Cart-lambda-cong*: (⋀*x*. *f x = g x*) ⟹ (*vec-lambda f*::*'a^'n*) = (*vec-lambda g* :: *'a^'n*)
⟨*proof*⟩

lemma *det-linear-row-setsum*:
assumes *fS*: *finite S*
shows *det* ((χ *i*. *if i = k then setsum* (*a i*) *S else c i*)::*'a*::*comm-ring-1^'n^'n*)
=
setsum (λ*j*. *det* ((χ *i*. *if i = k then a i j else c i*)::*'a^'n^'n*)) *S*
⟨*proof*⟩

lemma *finite-bounded-functions*:
assumes *fS*: *finite S*
shows *finite* {*f*. (∀ *i* ∈ {*1*.. (*k*::*nat*)}. *f i* ∈ *S*) ∧ (∀ *i*. *i* ∉ {*1* .. *k*} ⟶ *f i = i*)}
⟨*proof*⟩

lemma *det-linear-rows-setsum-lemma*:
assumes *fS*: *finite S*
and *fT*: *finite T*
shows *det* ((χ *i*. *if i* ∈ *T then setsum* (*a i*) *S else c i*):: *'a*::*comm-ring-1^'n^'n*)
=
setsum (λ*f*. *det*((χ *i*. *if i* ∈ *T then a i* (*f i*) *else c i*)::*'a^'n^'n*))
{*f*. (∀ *i* ∈ *T*. *f i* ∈ *S*) ∧ (∀ *i*. *i* ∉ *T* ⟶ *f i = i*)}
⟨*proof*⟩

lemma *det-linear-rows-setsum*:
fixes *S* :: *'n*::*finite set*
assumes *fS*: *finite S*
shows *det* (χ *i*. *setsum* (*a i*) *S*) =
setsum (λ*f*. *det* (χ *i*. *a i* (*f i*) :: *'a*::*comm-ring-1 ^ 'n^'n*)) {*f*. ∀ *i*. *f i* ∈ *S*}
⟨*proof*⟩

lemma *matrix-mul-setsum-alt*:
fixes *A B* :: *'a*::*comm-ring-1^'n^'n*
shows *A ** B* = (χ *i*. *setsum* (λ*k*. *Aik ∗s B $ k*) (*UNIV* :: *'n set*))

⟨*proof*⟩

lemma *det-rows-mul*:
det((χ *i*. *c i* ∗*s a i*)::*'a*::*comm-ring-1*^*'n*^*'n*) =
setprod (λ*i*. *c i*) (*UNIV*:: *'n set*) ∗ *det*((χ *i*. *a i*)::*'a*^*'n*^*'n*)
⟨*proof*⟩

lemma *det-mul*:
fixes *A B* :: *'a*::*linordered-idom*^*'n*^*'n*
shows *det* (*A* ∗∗ *B*) = *det A* ∗ *det B*
⟨*proof*⟩

Relation to invertibility.

lemma *invertible-left-inverse*:
fixes *A* :: *real*^*'n*^*'n*
shows *invertible A* ⟷ (∃(*B*::*real*^*'n*^*'n*). *B*∗∗ *A* = *mat 1*)
⟨*proof*⟩

lemma *invertible-righ-inverse*:
fixes *A* :: *real*^*'n*^*'n*
shows *invertible A* ⟷ (∃(*B*::*real*^*'n*^*'n*). *A*∗∗ *B* = *mat 1*)
⟨*proof*⟩

lemma *invertible-det-nz*:
fixes *A*::*real* ^*'n*^*'n*
shows *invertible A* ⟷ *det A* ≠ *0*
⟨*proof*⟩

Cramer's rule.

lemma *cramer-lemma-transpose*:
fixes *A*:: *real*^*'n*^*'n*
and *x* :: *real*^*'n*
shows *det* ((χ *i*. *if i* = *k then setsum* (λ*i*. *x*$*i* ∗*s row i A*) (*UNIV*::*'n set*)
else row i A)::*real*^*'n*^*'n*) = *x*$*k* ∗ *det A*
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *cramer-lemma*:
fixes *A* :: *real*^*'n*^*'n*
shows *det*((χ *i j*. *if j* = *k then* (*A* ∗*v x*)$*i else A*$*i*$*j*):: *real*^*'n*^*'n*) = *x*$*k* ∗ *det A*
⟨*proof*⟩

lemma *cramer*:
fixes *A* ::*real*^*'n*^*'n*
assumes *d0*: *det A* ≠ *0*
shows *A* ∗*v x* = *b* ⟷ *x* = (χ *k*. *det*(χ *i j*. *if j*=*k then b*$*i else A*$*i*$*j*) / *det A*)
⟨*proof*⟩

Orthogonality of a transformation and matrix.

definition *orthogonal-transformation f* $\longleftrightarrow$ *linear f* $\wedge$ $(\forall\, v\ w.\ f\ v \cdot f\ w = v \cdot w)$

lemma *orthogonal-transformation*:
orthogonal-transformation f $\longleftrightarrow$ *linear f* $\wedge$ $(\forall\, (v{::}real\ \hat{}\text{-}).\ norm\ (f\ v) = norm\ v)$
⟨*proof*⟩

definition *orthogonal-matrix* $(Q{::}'a{::}semiring\text{-}1\hat{}'n\hat{}'n) \longleftrightarrow$
transpose Q ** *Q* = *mat 1* $\wedge$ *Q* ** *transpose Q* = *mat 1*

lemma *orthogonal-matrix*: *orthogonal-matrix* $(Q{::}\ real\ \hat{}'n\hat{}'n) \longleftrightarrow$ *transpose Q* ** *Q* = *mat 1*
⟨*proof*⟩

lemma *orthogonal-matrix-id*: *orthogonal-matrix* $(mat\ 1 :: \text{-}\hat{}'n\hat{}'n)$
⟨*proof*⟩

lemma *orthogonal-matrix-mul*:
fixes *A* :: *real* $\hat{}'n\hat{}'n$
assumes *oA* : *orthogonal-matrix A*
and *oB*: *orthogonal-matrix B*
shows *orthogonal-matrix*(*A* ** *B*)
⟨*proof*⟩

lemma *orthogonal-transformation-matrix*:
fixes *f*:: *real*$\hat{}'n \Rightarrow$ *real*$\hat{}'n$
shows *orthogonal-transformation f* $\longleftrightarrow$ *linear f* $\wedge$ *orthogonal-matrix*(*matrix f*)
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *det-orthogonal-matrix*:
fixes *Q*:: *'a*::*linordered-idom*$\hat{}'n\hat{}'n$
assumes *oQ*: *orthogonal-matrix Q*
shows *det Q* = *1* $\vee$ *det Q* = − *1*
⟨*proof*⟩

Linearity of scaling, and hence isometry, that preserves origin.

lemma *scaling-linear*:
fixes *f* :: *real* $\hat{}'n \Rightarrow$ *real* $\hat{}'n$
assumes *f0*: *f 0* = *0*
and *fd*: $\forall\, x\ y.$ *dist* (*f x*) (*f y*) = *c* ∗ *dist x y*
shows *linear f*
⟨*proof*⟩

lemma *isometry-linear*:
f (*0*:: *real*$\hat{}'n$) = (*0*:: *real*$\hat{}'n$) $\Longrightarrow \forall\, x\ y.$ *dist*(*f x*) (*f y*) = *dist x y* $\Longrightarrow$ *linear f*
⟨*proof*⟩

Hence another formulation of orthogonal transformation.

lemma *orthogonal-transformation-isometry*:

orthogonal-transformation f $\longleftrightarrow$ *f*(*0*::*real*^'*n*) = (*0*::*real*^'*n*) $\wedge$ ($\forall$ *x y*. *dist*(*f x*) (*f y*) = *dist x y*)
⟨*proof*⟩

Can extend an isometry from unit sphere.

lemma *isometry-sphere-extend*:
fixes *f*:: *real* ^'*n* $\Rightarrow$ *real* ^'*n*
assumes *f1*: $\forall$ *x*. *norm x* = *1* $\longrightarrow$ *norm* (*f x*) = *1*
and *fd1*: $\forall$ *x y*. *norm x* = *1* $\longrightarrow$ *norm y* = *1* $\longrightarrow$ *dist* (*f x*) (*f y*) = *dist x y*
shows $\exists$ *g*. *orthogonal-transformation g* $\wedge$ ($\forall$ *x*. *norm x* = *1* $\longrightarrow$ *g x* = *f x*)
⟨*proof*⟩

Rotation, reflection, rotoinversion.

definition *rotation-matrix Q* $\longleftrightarrow$ *orthogonal-matrix Q* $\wedge$ *det Q* = *1*
definition *rotoinversion-matrix Q* $\longleftrightarrow$ *orthogonal-matrix Q* $\wedge$ *det Q* = $-$ *1*

lemma *orthogonal-rotation-or-rotoinversion*:
fixes *Q* :: *'a*::*linordered-idom*^'*n*^'*n*
shows *orthogonal-matrix Q* $\longleftrightarrow$ *rotation-matrix Q* $\vee$ *rotoinversion-matrix Q*
⟨*proof*⟩

Explicit formulas for low dimensions.

lemma *setprod-neutral-const*: *setprod f* {(*1*::*nat*)..*1*} = *f 1*
⟨*proof*⟩

lemma *setprod-2*: *setprod f* {(*1*::*nat*)..*2*} = *f 1* * *f 2*
⟨*proof*⟩

lemma *setprod-3*: *setprod f* {(*1*::*nat*)..*3*} = *f 1* * *f 2* * *f 3*
⟨*proof*⟩

lemma *det-1*: *det* (*A*::*'a*::*comm-ring-1*^*1*^*1*) = *A$1$1*
⟨*proof*⟩

lemma *det-2*: *det* (*A*::*'a*::*comm-ring-1*^*2*^*2*) = *A$1$1* * *A$2$2* $-$ *A$1$2* * *A$2$1*
⟨*proof*⟩

lemma *det-3*:
det (*A*::*'a*::*comm-ring-1*^*3*^*3*) =
A$1$1 * *A$2$2* * *A$3$3* +
A$1$2 * *A$2$3* * *A$3$1* +
A$1$3 * *A$2$1* * *A$3$2* $-$
A$1$1 * *A$2$3* * *A$3$2* $-$
A$1$2 * *A$2$1* * *A$3$3* $-$
A$1$3 * *A$2$2* * *A$3$1*
⟨*proof*⟩

end

47 Pointwise order on product types

theory *Product-Order*
imports *Product-plus Conditionally-Complete-Lattices*
begin

47.1 Pointwise ordering

instantiation *prod* :: (*ord*, *ord*) *ord*
begin

definition
$x \leq y \longleftrightarrow fst\ x \leq fst\ y \wedge snd\ x \leq snd\ y$

definition
$(x::'a \times 'b) < y \longleftrightarrow x \leq y \wedge \neg\ y \leq x$

instance ⟨*proof*⟩

end

lemma *fst-mono*: $x \leq y \Longrightarrow fst\ x \leq fst\ y$
⟨*proof*⟩

lemma *snd-mono*: $x \leq y \Longrightarrow snd\ x \leq snd\ y$
⟨*proof*⟩

lemma *Pair-mono*: $x \leq x' \Longrightarrow y \leq y' \Longrightarrow (x, y) \leq (x', y')$
⟨*proof*⟩

lemma *Pair-le* [*simp*]: $(a, b) \leq (c, d) \longleftrightarrow a \leq c \wedge b \leq d$
⟨*proof*⟩

instance *prod* :: (*preorder*, *preorder*) *preorder*
⟨*proof*⟩

instance *prod* :: (*order*, *order*) *order*
⟨*proof*⟩

47.2 Binary infimum and supremum

instantiation *prod* :: (*inf*, *inf*) *inf*
begin

definition $inf\ x\ y = (inf\ (fst\ x)\ (fst\ y), inf\ (snd\ x)\ (snd\ y))$

lemma *inf-Pair-Pair* [*simp*]: $inf\ (a, b)\ (c, d) = (inf\ a\ c, inf\ b\ d)$
⟨*proof*⟩

lemma *fst-inf* [*simp*]: $fst\ (inf\ x\ y) = inf\ (fst\ x)\ (fst\ y)$

⟨*proof*⟩

lemma *snd-inf* [*simp*]: *snd* (*inf x y*) = *inf* (*snd x*) (*snd y*)
⟨*proof*⟩

instance ⟨*proof*⟩

end

instance *prod* :: (*semilattice-inf*, *semilattice-inf*) *semilattice-inf*
⟨*proof*⟩

instantiation *prod* :: (*sup*, *sup*) *sup*
begin

definition
sup x y = (*sup* (*fst x*) (*fst y*), *sup* (*snd x*) (*snd y*))

lemma *sup-Pair-Pair* [*simp*]: *sup* (*a*, *b*) (*c*, *d*) = (*sup a c*, *sup b d*)
⟨*proof*⟩

lemma *fst-sup* [*simp*]: *fst* (*sup x y*) = *sup* (*fst x*) (*fst y*)
⟨*proof*⟩

lemma *snd-sup* [*simp*]: *snd* (*sup x y*) = *sup* (*snd x*) (*snd y*)
⟨*proof*⟩

instance ⟨*proof*⟩

end

instance *prod* :: (*semilattice-sup*, *semilattice-sup*) *semilattice-sup*
⟨*proof*⟩

instance *prod* :: (*lattice*, *lattice*) *lattice* ⟨*proof*⟩

instance *prod* :: (*distrib-lattice*, *distrib-lattice*) *distrib-lattice*
⟨*proof*⟩

47.3 Top and bottom elements

instantiation *prod* :: (*top*, *top*) *top*
begin

definition
top = (*top*, *top*)

instance ⟨*proof*⟩

end

lemma *fst-top* [*simp*]: *fst top* = *top*
⟨*proof*⟩

lemma *snd-top* [*simp*]: *snd top* = *top*
⟨*proof*⟩

lemma *Pair-top-top*: (*top*, *top*) = *top*
⟨*proof*⟩

instance *prod* :: (*order-top*, *order-top*) *order-top*
⟨*proof*⟩

instantiation *prod* :: (*bot*, *bot*) *bot*
begin

definition
bot = (*bot*, *bot*)

instance ⟨*proof*⟩

end

lemma *fst-bot* [*simp*]: *fst bot* = *bot*
⟨*proof*⟩

lemma *snd-bot* [*simp*]: *snd bot* = *bot*
⟨*proof*⟩

lemma *Pair-bot-bot*: (*bot*, *bot*) = *bot*
⟨*proof*⟩

instance *prod* :: (*order-bot*, *order-bot*) *order-bot*
⟨*proof*⟩

instance *prod* :: (*bounded-lattice*, *bounded-lattice*) *bounded-lattice* ⟨*proof*⟩

instance *prod* :: (*boolean-algebra*, *boolean-algebra*) *boolean-algebra*
⟨*proof*⟩

47.4 Complete lattice operations

instantiation *prod* :: (*Inf*, *Inf*) *Inf*
begin

definition *Inf A* = (*INF x*:*A*. *fst x*, *INF x*:*A*. *snd x*)

instance ⟨*proof*⟩

end

instantiation *prod* :: (*Sup*, *Sup*) *Sup*
begin

definition *Sup A* = (*SUP x*:*A*. *fst x*, *SUP x*:*A*. *snd x*)

instance ⟨*proof*⟩

end

instance *prod* :: (*conditionally-complete-lattice*, *conditionally-complete-lattice*)
conditionally-complete-lattice
⟨*proof*⟩

instance *prod* :: (*complete-lattice*, *complete-lattice*) *complete-lattice*
⟨*proof*⟩

lemma *fst-Sup*: *fst* (*Sup A*) = (*SUP x*:*A*. *fst x*)
⟨*proof*⟩

lemma *snd-Sup*: *snd* (*Sup A*) = (*SUP x*:*A*. *snd x*)
⟨*proof*⟩

lemma *fst-Inf*: *fst* (*Inf A*) = (*INF x*:*A*. *fst x*)
⟨*proof*⟩

lemma *snd-Inf*: *snd* (*Inf A*) = (*INF x*:*A*. *snd x*)
⟨*proof*⟩

lemma *fst-SUP*: *fst* (*SUP x*:*A*. *f x*) = (*SUP x*:*A*. *fst* (*f x*))
⟨*proof*⟩

lemma *snd-SUP*: *snd* (*SUP x*:*A*. *f x*) = (*SUP x*:*A*. *snd* (*f x*))
⟨*proof*⟩

lemma *fst-INF*: *fst* (*INF x*:*A*. *f x*) = (*INF x*:*A*. *fst* (*f x*))
⟨*proof*⟩

lemma *snd-INF*: *snd* (*INF x*:*A*. *f x*) = (*INF x*:*A*. *snd* (*f x*))
⟨*proof*⟩

lemma *SUP-Pair*: (*SUP x*:*A*. (*f x*, *g x*)) = (*SUP x*:*A*. *f x*, *SUP x*:*A*. *g x*)
⟨*proof*⟩

lemma *INF-Pair*: (*INF x*:*A*. (*f x*, *g x*)) = (*INF x*:*A*. *f x*, *INF x*:*A*. *g x*)
⟨*proof*⟩

Alternative formulations for set infima and suprema over the product of two complete lattices:

lemma *INF-prod-alt-def*:
INFIMUM A f = (*INFIMUM A* (*fst o f*), *INFIMUM A* (*snd o f*))
⟨*proof*⟩

lemma *SUP-prod-alt-def*:
SUPREMUM A f = (*SUPREMUM A* (*fst o f*), *SUPREMUM A* (*snd o f*))
⟨*proof*⟩

47.5 Complete distributive lattices

instance *prod* :: (*complete-distrib-lattice*, *complete-distrib-lattice*) *complete-distrib-lattice*
⟨*proof*⟩

end
theory *Ordered-Euclidean-Space*
imports
Cartesian-Euclidean-Space
~~/*src*/*HOL*/*Library*/*Product-Order*
begin

47.6 An ordering on euclidean spaces that will allow us to talk about intervals

class *ordered-euclidean-space* = *ord* + *inf* + *sup* + *abs* + *Inf* + *Sup* + *euclidean-space* +
assumes *eucl-le*: $x \leq y \longleftrightarrow (\forall i{\in}Basis.\ x \cdot i \leq y \cdot i)$
assumes *eucl-less-le-not-le*: $x < y \longleftrightarrow x \leq y \wedge \neg\ y \leq x$
assumes *eucl-inf*: *inf x y* = $(\sum i{\in}Basis.\ inf\ (x \cdot i)\ (y \cdot i) *_R i)$
assumes *eucl-sup*: *sup x y* = $(\sum i{\in}Basis.\ sup\ (x \cdot i)\ (y \cdot i) *_R i)$
assumes *eucl-Inf*: *Inf X* = $(\sum i{\in}Basis.\ (INF\ x{:}X.\ x \cdot i) *_R i)$
assumes *eucl-Sup*: *Sup X* = $(\sum i{\in}Basis.\ (SUP\ x{:}X.\ x \cdot i) *_R i)$
assumes *eucl-abs*: $|x| = (\sum i{\in}Basis.\ |x \cdot i| *_R i)$
begin

subclass *order*
⟨*proof*⟩

subclass *ordered-ab-group-add-abs*
⟨*proof*⟩

subclass *ordered-real-vector*
⟨*proof*⟩

subclass *lattice*
⟨*proof*⟩

subclass *distrib-lattice*

⟨*proof*⟩

subclass *conditionally-complete-lattice*
⟨*proof*⟩

lemma *inner-Basis-inf-left*: $i \in Basis \Longrightarrow inf\ x\ y \cdot i = inf\ (x \cdot i)\ (y \cdot i)$
and *inner-Basis-sup-left*: $i \in Basis \Longrightarrow sup\ x\ y \cdot i = sup\ (x \cdot i)\ (y \cdot i)$
⟨*proof*⟩

lemma *inner-Basis-INF-left*: $i \in Basis \Longrightarrow (INF\ x{:}X.\ f\ x) \cdot i = (INF\ x{:}X.\ f\ x \cdot i)$
and *inner-Basis-SUP-left*: $i \in Basis \Longrightarrow (SUP\ x{:}X.\ f\ x) \cdot i = (SUP\ x{:}X.\ f\ x \cdot i)$
⟨*proof*⟩

lemma *abs-inner*: $i \in Basis \Longrightarrow |x| \cdot i = |x \cdot i|$
⟨*proof*⟩

lemma
abs-scaleR: $|a *_R b| = |a| *_R |b|$
⟨*proof*⟩

lemma *interval-inner-leI*:
assumes $x \in \{a\ ..\ b\}\ 0 \le i$
shows $a \cdot i \le x \cdot i\ x \cdot i \le b \cdot i$
⟨*proof*⟩

lemma *inner-nonneg-nonneg*:
shows $0 \le a \Longrightarrow 0 \le b \Longrightarrow 0 \le a \cdot b$
⟨*proof*⟩

lemma *inner-Basis-mono*:
shows $a \le b \Longrightarrow c \in Basis \Longrightarrow a \cdot c \le b \cdot c$
⟨*proof*⟩

lemma *Basis-nonneg*[*intro*, *simp*]: $i \in Basis \Longrightarrow 0 \le i$
⟨*proof*⟩

lemma *Sup-eq-maximum-componentwise*:
fixes $s{::}'a\ set$
assumes *i*: $\bigwedge b.\ b \in Basis \Longrightarrow X \cdot b = i\ b \cdot b$
assumes *sup*: $\bigwedge b\ x.\ b \in Basis \Longrightarrow x \in s \Longrightarrow x \cdot b \le X \cdot b$
assumes *i-s*: $\bigwedge b.\ b \in Basis \Longrightarrow (i\ b \cdot b) \in (\lambda x.\ x \cdot b)\ `\ s$
shows $Sup\ s = X$
⟨*proof*⟩

lemma *Inf-eq-minimum-componentwise*:
assumes *i*: $\bigwedge b.\ b \in Basis \Longrightarrow X \cdot b = i\ b \cdot b$
assumes *sup*: $\bigwedge b\ x.\ b \in Basis \Longrightarrow x \in s \Longrightarrow X \cdot b \le x \cdot b$

assumes *i-s*: $\bigwedge b.\ b \in Basis \Longrightarrow (i\ b \cdot b) \in (\lambda x.\ x \cdot b)\ `\ s$
shows $Inf\ s = X$
⟨*proof*⟩

end

lemma
compact-attains-Inf-componentwise:
fixes $b::'a::ordered\text{-}euclidean\text{-}space$
assumes $b \in Basis$ **assumes** $X \neq \{\}$ *compact* X
obtains x **where** $x \in X\ x \cdot b = Inf\ X \cdot b\ \bigwedge y.\ y \in X \Longrightarrow x \cdot b \leq y \cdot b$
⟨*proof*⟩

lemma
compact-attains-Sup-componentwise:
fixes $b::'a::ordered\text{-}euclidean\text{-}space$
assumes $b \in Basis$ **assumes** $X \neq \{\}$ *compact* X
obtains x **where** $x \in X\ x \cdot b = Sup\ X \cdot b\ \bigwedge y.\ y \in X \Longrightarrow y \cdot b \leq x \cdot b$
⟨*proof*⟩

lemma (**in** *order*) *atLeastatMost-empty'*[*simp*]:
$(\sim\ a <= b) \Longrightarrow \{a..b\} = \{\}$
⟨*proof*⟩

instance *real* :: *ordered-euclidean-space*
⟨*proof*⟩

lemma *in-Basis-prod-iff*:
fixes $i::'a::euclidean\text{-}space*'b::euclidean\text{-}space$
shows $i \in Basis \longleftrightarrow fst\ i = 0 \wedge snd\ i \in Basis \vee snd\ i = 0 \wedge fst\ i \in Basis$
⟨*proof*⟩

instantiation *prod* :: (*abs*, *abs*) *abs*
begin

definition $|x| = (|fst\ x|, |snd\ x|)$

instance ⟨*proof*⟩

end

instance *prod* :: (*ordered-euclidean-space*, *ordered-euclidean-space*) *ordered-euclidean-space*
⟨*proof*⟩

Instantiation for intervals on *ordered-euclidean-space*

lemma
fixes $a :: 'a::ordered\text{-}euclidean\text{-}space$
shows *cbox-interval*: $cbox\ a\ b = \{a..b\}$
and *interval-cbox*: $\{a..b\} = cbox\ a\ b$

and *eucl-le-atMost*: $\{x.\ \forall i{\in}Basis.\ x \cdot i <= a \cdot i\} = \{..a\}$
and *eucl-le-atLeast*: $\{x.\ \forall i{\in}Basis.\ a \cdot i <= x \cdot i\} = \{a..\}$
⟨*proof*⟩

lemma *closed-eucl-atLeastAtMost*[*simp*, *intro*]:
fixes a :: $'a$::*ordered-euclidean-space*
shows *closed* $\{a..b\}$
⟨*proof*⟩

lemma *closed-eucl-atMost*[*simp*, *intro*]:
fixes a :: $'a$::*ordered-euclidean-space*
shows *closed* $\{..a\}$
⟨*proof*⟩

lemma *closed-eucl-atLeast*[*simp*, *intro*]:
fixes a :: $'a$::*ordered-euclidean-space*
shows *closed* $\{a..\}$
⟨*proof*⟩

lemma *bounded-closed-interval*:
fixes a :: $'a$::*ordered-euclidean-space*
shows *bounded* $\{a\ ..\ b\}$
⟨*proof*⟩

lemma *convex-closed-interval*:
fixes a :: $'a$::*ordered-euclidean-space*
shows *convex* $\{a\ ..\ b\}$
⟨*proof*⟩

lemma *image-smult-interval*:$(\lambda x.\ m *_R (x::-::ordered\text{-}euclidean\text{-}space))\ `\ \{a\ ..\ b\} =$
$(if\ \{a\ ..\ b\} = \{\}\ then\ \{\}\ else\ if\ 0 \leq m\ then\ \{m *_R a\ ..\ m *_R b\}\ else\ \{m *_R b\ ..\ m *_R a\})$
⟨*proof*⟩

lemma *is-interval-closed-interval*:
is-interval $\{a\ ..\ (b::'a::ordered\text{-}euclidean\text{-}space)\}$
⟨*proof*⟩

lemma *compact-interval*:
fixes $a\ b::'a$::*ordered-euclidean-space*
shows *compact* $\{a\ ..\ b\}$
⟨*proof*⟩

lemma *homeomorphic-closed-intervals*:
fixes a :: $'a$::*euclidean-space* **and** b **and** c :: $'a$::*euclidean-space* **and** d
assumes *box* $a\ b \neq \{\}$ **and** *box* $c\ d \neq \{\}$
shows (*cbox* $a\ b$) *homeomorphic* (*cbox* $c\ d$)
⟨*proof*⟩

lemma *homeomorphic-closed-intervals-real*:
fixes *a*::*real* **and** *b* **and** *c*::*real* **and** *d*
assumes *a*<*b* **and** *c*<*d*
shows {*a*..*b*} *homeomorphic* {*c*..*d*}
⟨*proof*⟩

no-notation
eucl-less (**infix** <*e* 50)

lemma *One-nonneg*: 0 ≤ (∑ *Basis*::'*a*::*ordered-euclidean-space*)
⟨*proof*⟩

lemma *content-closed-interval*:
fixes *a* :: '*a*::*ordered-euclidean-space*
assumes *a* ≤ *b*
shows *content* {*a* .. *b*} = (∏ *i*∈*Basis*. *b*•*i* − *a*•*i*)
⟨*proof*⟩

lemma *integrable-const-ivl*[*intro*]:
fixes *a*::'*a*::*ordered-euclidean-space*
shows (λ*x*. *c*) *integrable-on* {*a* .. *b*}
⟨*proof*⟩

lemma *integrable-on-subinterval*:
fixes *f* :: '*n*::*ordered-euclidean-space* ⇒ '*a*::*banach*
assumes *f integrable-on s*
and {*a* .. *b*} ⊆ *s*
shows *f integrable-on* {*a* .. *b*}
⟨*proof*⟩

lemma
fixes *a b*::'*a*::*ordered-euclidean-space*
shows *bdd-above-cbox*[*intro*, *simp*]: *bdd-above* (*cbox a b*)
and *bdd-below-cbox*[*intro*, *simp*]: *bdd-below* (*cbox a b*)
and *bdd-above-box*[*intro*, *simp*]: *bdd-above* (*box a b*)
and *bdd-below-box*[*intro*, *simp*]: *bdd-below* (*box a b*)
⟨*proof*⟩

instantiation *vec* :: (*ordered-euclidean-space*, *finite*) *ordered-euclidean-space*
begin

definition *inf x y* = (χ *i*. *inf* (*x* $ *i*) (*y* $ *i*))
definition *sup x y* = (χ *i*. *sup* (*x* $ *i*) (*y* $ *i*))
definition *Inf X* = (χ *i*. (*INF x*:*X*. *x* $ *i*))
definition *Sup X* = (χ *i*. (*SUP x*:*X*. *x* $ *i*))
definition |*x*| = (χ *i*. |*x* $ *i*|)

instance

⟨*proof*⟩

end

end

48 Bounded Continuous Functions

theory *Bounded-Continuous-Function*
imports *Integration*
begin

48.1 Definition

definition *bcontfun* :: ('*a*::*topological-space* ⇒ '*b*::*metric-space*) *set*
  where *bcontfun* = {*f*. *continuous-on UNIV f* ∧ *bounded* (*range f*)}

typedef (**overloaded**) ('*a*, '*b*) *bcontfun* =
    bcontfun :: ('*a*::*topological-space* ⇒ '*b*::*metric-space*) *set*
  ⟨*proof*⟩

lemma *bcontfunE*:
  assumes *f* ∈ *bcontfun*
  obtains *y* **where** *continuous-on UNIV f* ⋀*x*. *dist* (*f x*) *u* ≤ *y*
  ⟨*proof*⟩

lemma *bcontfunE'*:
  assumes *f* ∈ *bcontfun*
  obtains *y* **where** *continuous-on UNIV f* ⋀*x*. *dist* (*f x*) *undefined* ≤ *y*
  ⟨*proof*⟩

lemma *bcontfunI*: *continuous-on UNIV f* ⟹ (⋀*x*. *dist* (*f x*) *u* ≤ *b*) ⟹ *f* ∈ *bcontfun*
  ⟨*proof*⟩

lemma *bcontfunI'*: *continuous-on UNIV f* ⟹ (⋀*x*. *dist* (*f x*) *undefined* ≤ *b*) ⟹ *f* ∈ *bcontfun*
  ⟨*proof*⟩

lemma *continuous-on-Rep-bcontfun*[*intro*, *simp*]: *continuous-on T* (*Rep-bcontfun x*)
  ⟨*proof*⟩

instantiation *bcontfun* :: (*topological-space*, *metric-space*) *metric-space*
begin

definition *dist-bcontfun* :: ('*a*, '*b*) *bcontfun* ⇒ ('*a*, '*b*) *bcontfun* ⇒ *real*
  where *dist-bcontfun f g* = (*SUP x*. *dist* (*Rep-bcontfun f x*) (*Rep-bcontfun g x*))

definition *uniformity-bcontfun* :: $(('a, 'b)\ bcontfun \times ('a, 'b)\ bcontfun)\ filter$
where $uniformity\text{-}bcontfun = (INF\ e{:}\{0 <..\}.\ principal\ \{(x, y).\ dist\ x\ y < e\})$

definition *open-bcontfun* :: $('a, 'b)\ bcontfun\ set \Rightarrow bool$
where $open\text{-}bcontfun\ S = (\forall\, x{\in}S.\ \forall_F\ (x', y)\ in\ uniformity.\ x' = x \longrightarrow y \in S)$

lemma *dist-bounded*:
fixes $f :: ('a, 'b)\ bcontfun$
shows $dist\ (Rep\text{-}bcontfun\ f\ x)\ (Rep\text{-}bcontfun\ g\ x) \leq dist\ f\ g$
⟨*proof*⟩

lemma *dist-bound*:
fixes $f :: ('a, 'b)\ bcontfun$
assumes $\bigwedge x.\ dist\ (Rep\text{-}bcontfun\ f\ x)\ (Rep\text{-}bcontfun\ g\ x) \leq b$
shows $dist\ f\ g \leq b$
⟨*proof*⟩

lemma *dist-bounded-Abs*:
fixes $f\ g :: 'a \Rightarrow 'b$
assumes $f \in bcontfun\ g \in bcontfun$
shows $dist\ (f\ x)\ (g\ x) \leq dist\ (Abs\text{-}bcontfun\ f)\ (Abs\text{-}bcontfun\ g)$
⟨*proof*⟩

lemma *const-bcontfun*: $(\lambda x{::}'a.\ b{::}'b) \in bcontfun$
⟨*proof*⟩

lemma *dist-fun-lt-imp-dist-val-lt*:
assumes $dist\ f\ g < e$
shows $dist\ (Rep\text{-}bcontfun\ f\ x)\ (Rep\text{-}bcontfun\ g\ x) < e$
⟨*proof*⟩

lemma *dist-val-lt-imp-dist-fun-le*:
assumes $\forall\, x.\ dist\ (Rep\text{-}bcontfun\ f\ x)\ (Rep\text{-}bcontfun\ g\ x) < e$
shows $dist\ f\ g \leq e$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *closed-Pi-bcontfun*:
fixes $I :: 'a{::}metric\text{-}space\ set$
and $X :: 'a \Rightarrow 'b{::}complete\text{-}space\ set$
assumes $\bigwedge i.\ i \in I \Longrightarrow closed\ (X\ i)$
shows $closed\ (Abs\text{-}bcontfun\ `\ (Pi\ I\ X \cap bcontfun))$
⟨*proof*⟩

48.2 Complete Space

instance *bcontfun* :: (*metric-space*, *complete-space*) *complete-space*
⟨*proof*⟩

48.3 Supremum norm for a normed vector space

instantiation *bcontfun* :: (*topological-space*, *real-normed-vector*) *real-vector*
begin

definition $-f = Abs\text{-}bcontfun\ (\lambda x.\ -(Rep\text{-}bcontfun\ f\ x))$

definition $f + g = Abs\text{-}bcontfun\ (\lambda x.\ Rep\text{-}bcontfun\ f\ x + Rep\text{-}bcontfun\ g\ x)$

definition $f - g = Abs\text{-}bcontfun\ (\lambda x.\ Rep\text{-}bcontfun\ f\ x - Rep\text{-}bcontfun\ g\ x)$

definition $0 = Abs\text{-}bcontfun\ (\lambda x.\ 0)$

definition $scaleR\ r\ f = Abs\text{-}bcontfun\ (\lambda x.\ r *_R Rep\text{-}bcontfun\ f\ x)$

lemma *plus-cont*:
fixes $f\ g :: 'a \Rightarrow 'b$
assumes f: $f \in bcontfun$
and g: $g \in bcontfun$
shows $(\lambda x.\ f\ x + g\ x) \in bcontfun$
⟨*proof*⟩

lemma *Rep-bcontfun-plus*[*simp*]: $Rep\text{-}bcontfun\ (f + g)\ x = Rep\text{-}bcontfun\ f\ x + Rep\text{-}bcontfun\ g\ x$
⟨*proof*⟩

lemma *uminus-cont*:
fixes $f :: 'a \Rightarrow 'b$
assumes $f \in bcontfun$
shows $(\lambda x.\ -f\ x) \in bcontfun$
⟨*proof*⟩

lemma *Rep-bcontfun-uminus*[*simp*]: $Rep\text{-}bcontfun\ (-f)\ x = -\ Rep\text{-}bcontfun\ f\ x$
⟨*proof*⟩

lemma *minus-cont*:
fixes $f\ g :: 'a \Rightarrow 'b$
assumes f: $f \in bcontfun$
and g: $g \in bcontfun$
shows $(\lambda x.\ f\ x - g\ x) \in bcontfun$
⟨*proof*⟩

lemma *Rep-bcontfun-minus*[*simp*]: $Rep\text{-}bcontfun\ (f - g)\ x = Rep\text{-}bcontfun\ f\ x - Rep\text{-}bcontfun\ g\ x$
⟨*proof*⟩

lemma *scaleR-cont*:
fixes a :: *real*
and f :: $'a \Rightarrow 'b$
assumes $f \in$ *bcontfun*
shows $(\lambda x.\ a *_R f\ x) \in$ *bcontfun*
⟨*proof*⟩

lemma *Rep-bcontfun-scaleR*[*simp*]: *Rep-bcontfun* $(a *_R g)\ x = a *_R$ *Rep-bcontfun* $g\ x$
⟨*proof*⟩

instance
⟨*proof*⟩

end

instantiation *bcontfun* :: (*topological-space*, *real-normed-vector*) *real-normed-vector*
begin

definition *norm-bcontfun* :: $('a, 'b)$ *bcontfun* $\Rightarrow$ *real*
where *norm-bcontfun f* = *dist f 0*

definition *sgn* $(f::('a,'b)$ *bcontfun*$) = f\ /_R$ *norm f*

instance
⟨*proof*⟩

end

lemma *bcontfun-normI*: *continuous-on UNIV f* $\Longrightarrow (\bigwedge x.$ *norm* $(f\ x) \leq b) \Longrightarrow f \in$ *bcontfun*
⟨*proof*⟩

lemma *norm-bounded*:
fixes f :: $('a::$*topological-space*, $'b::$*real-normed-vector*$)$ *bcontfun*
shows *norm* (*Rep-bcontfun f x*) $\leq$ *norm f*
⟨*proof*⟩

lemma *norm-bound*:
fixes f :: $('a::$*topological-space*, $'b::$*real-normed-vector*$)$ *bcontfun*
assumes $\bigwedge x.$ *norm* (*Rep-bcontfun f x*) $\leq b$
shows *norm f* $\leq b$
⟨*proof*⟩

48.4 Continuously Extended Functions

definition *clamp* :: $'a::$*euclidean-space* $\Rightarrow\ 'a \Rightarrow\ 'a \Rightarrow\ 'a$ **where**
clamp a b x = $(\sum i \in$ *Basis*. (*if* $x \cdot i < a \cdot i$ *then* $a \cdot i$ *else if* $x \cdot i \leq b \cdot i$ *then* $x \cdot i$ *else*

$b \cdot i) *_R i)$

definition *ext-cont* :: ('*a*::*euclidean-space* ⇒ '*b*::*real-normed-vector*) ⇒ '*a* ⇒ '*a* ⇒ ('*a*, '*b*) *bcontfun*
  where *ext-cont f a b* = *Abs-bcontfun* ((λ*x*. *f* (*clamp a b x*)))

lemma *ext-cont-def'*:
  ext-cont f a b = *Abs-bcontfun* (λ*x*.
    f (∑ *i*∈*Basis*. (*if x*·*i* < *a*·*i then a*·*i else if x*·*i* ≤ *b*·*i then x*·*i else b*·*i*) *$*_R$ *i*))
  ⟨*proof*⟩

lemma *clamp-in-interval*:
  assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* · *i* ≤ *b* · *i*
  shows *clamp a b x* ∈ *cbox a b*
  ⟨*proof*⟩

lemma *dist-clamps-le-dist-args*:
  fixes *x* :: '*a*::*euclidean-space*
  assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* · *i* ≤ *b* · *i*
  shows *dist* (*clamp a b y*) (*clamp a b x*) ≤ *dist y x*
⟨*proof*⟩

lemma *clamp-continuous-at*:
  fixes *f* :: '*a*::*euclidean-space* ⇒ '*b*::*metric-space*
    and *x* :: '*a*
  assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* · *i* ≤ *b* · *i*
    and *f-cont*: *continuous-on* (*cbox a b*) *f*
  shows *continuous* (*at x*) (λ*x*. *f* (*clamp a b x*))
  ⟨*proof*⟩

lemma *clamp-continuous-on*:
  fixes *f* :: '*a*::*euclidean-space* ⇒ '*b*::*metric-space*
  assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* · *i* ≤ *b* · *i*
    and *f-cont*: *continuous-on* (*cbox a b*) *f*
  shows *continuous-on UNIV* (λ*x*. *f* (*clamp a b x*))
  ⟨*proof*⟩

lemma *clamp-bcontfun*:
  fixes *f* :: '*a*::*euclidean-space* ⇒ '*b*::*real-normed-vector*
  assumes ⋀*i*. *i* ∈ *Basis* ⟹ *a* · *i* ≤ *b* · *i*
    and *continuous*: *continuous-on* (*cbox a b*) *f*
  shows (λ*x*. *f* (*clamp a b x*)) ∈ *bcontfun*
⟨*proof*⟩

lemma *integral-clamp*:
  integral {*t0*::*real*..*clamp t0 t1 x*} *f* =
    (*if x* < *t0 then* 0 *else if x* ≤ *t1 then integral* {*t0*..*x*} *f else integral* {*t0*..*t1*} *f*)
  ⟨*proof*⟩

declare [[*coercion Rep-bcontfun*]]

lemma *ext-cont-cancel*[*simp*]:
fixes *x a b* :: *'a*::*euclidean-space*
assumes *x*: *x* ∈ *cbox a b*
and *continuous-on* (*cbox a b*) *f*
shows *ext-cont f a b x* = *f x*
⟨*proof*⟩

lemma *ext-cont-cong*:
assumes *t0* = *s0*
and *t1* = *s1*
and ⋀*t*. *t* ∈ (*cbox t0 t1*) ⟹ *f t* = *g t*
and *continuous-on* (*cbox t0 t1*) *f*
and *continuous-on* (*cbox s0 s1*) *g*
and *ord*: ⋀*i*. *i* ∈ *Basis* ⟹ *t0* ∙ *i* ≤ *t1* ∙ *i*
shows *ext-cont f t0 t1* = *ext-cont g s0 s1*
⟨*proof*⟩

end

49 The Bernstein-Weierstrass and Stone-Weierstrass Theorems

By L C Paulson (2015)

theory *Weierstrass*
imports *Uniform-Limit Path-Connected*
begin

49.1 Bernstein polynomials

definition *Bernstein* :: [*nat*,*nat*,*real*] ⇒ *real* **where**
Bernstein n k x ≡ *of-nat* (*n choose k*) ∗ *x* ^ *k* ∗ (*1* − *x*) ^ (*n* − *k*)

lemma *Bernstein-nonneg*: ⟦*0* ≤ *x*; *x* ≤ *1*⟧ ⟹ *0* ≤ *Bernstein n k x*
⟨*proof*⟩

lemma *Bernstein-pos*: ⟦*0* < *x*; *x* < *1*; *k* ≤ *n*⟧ ⟹ *0* < *Bernstein n k x*
⟨*proof*⟩

lemma *sum-Bernstein* [*simp*]: (∑ *k* = *0*..*n*. *Bernstein n k x*) = *1*
⟨*proof*⟩

lemma *binomial-deriv1*:
(∑*k*=*0*..*n*. (*of-nat k* ∗ *of-nat* (*n choose k*)) ∗ *a*^(*k*−*1*) ∗ *b*^(*n*−*k*)) = *real-of-nat n* ∗ (*a*+*b*) ^ (*n*−*1*)
⟨*proof*⟩

lemma *binomial-deriv2*:
$(\sum k{=}0..n.\ (\textit{of-nat}\ k * \textit{of-nat}\ (k{-}1) * \textit{of-nat}\ (n\ \textit{choose}\ k)) * a\hat{}(k{-}2) * b\hat{}(n{-}k)) =$
$\textit{of-nat}\ n * \textit{of-nat}\ (n{-}1) * (a{+}b{::}\textit{real})\ \hat{}\ (n{-}2)$
⟨*proof*⟩

lemma *sum-k-Bernstein* [*simp*]: $(\sum k = 0..n.\ \textit{real}\ k * \textit{Bernstein}\ n\ k\ x) = \textit{of-nat}\ n * x$
⟨*proof*⟩

lemma *sum-kk-Bernstein* [*simp*]: $(\sum\ k = 0..n.\ \textit{real}\ k * (\textit{real}\ k - 1) * \textit{Bernstein}\ n\ k\ x) = \textit{real}\ n * (\textit{real}\ n - 1) * x^2$
⟨*proof*⟩

49.2 Explicit Bernstein version of the 1D Weierstrass approximation theorem

lemma *Bernstein-Weierstrass*:
fixes $f :: \textit{real} \Rightarrow \textit{real}$
assumes *contf*: *continuous-on* $\{0..1\}\ f$ **and** *e*: $0 < e$
shows $\exists N.\ \forall n\ x.\ N \leq n \wedge x \in \{0..1\} \longrightarrow |f\ x - (\sum k = 0..n.\ f(k/n) * \textit{Bernstein}\ n\ k\ x)| < e$
⟨*proof*⟩

49.3 General Stone-Weierstrass theorem

Source: Bruno Brosowski and Frank Deutsch. An Elementary Proof of the Stone-Weierstrass Theorem. Proceedings of the American Mathematical Society Volume 81, Number 1, January 1981. DOI: 10.2307/2043993 http://www.jstor.org/stable/2043993

locale *function-ring-on* =
fixes $R :: ('a{::}\textit{t2-space} \Rightarrow \textit{real})\ \textit{set}$ **and** $s :: 'a\ \textit{set}$
assumes *compact*: *compact s*
assumes *continuous*: $f \in R \Longrightarrow$ *continuous-on s f*
assumes *add*: $f \in R \Longrightarrow g \in R \Longrightarrow (\lambda x.\ f\ x + g\ x) \in R$
assumes *mult*: $f \in R \Longrightarrow g \in R \Longrightarrow (\lambda x.\ f\ x * g\ x) \in R$
assumes *const*: $(\lambda\text{-}.\ c) \in R$
assumes *separable*: $x \in s \Longrightarrow y \in s \Longrightarrow x \neq y \Longrightarrow \exists f{\in}R.\ f\ x \neq f\ y$

begin
lemma *minus*: $f \in R \Longrightarrow (\lambda x.\ -\ f\ x) \in R$
⟨*proof*⟩

lemma *diff*: $f \in R \Longrightarrow g \in R \Longrightarrow (\lambda x.\ f\ x - g\ x) \in R$
⟨*proof*⟩

lemma *power*: $f \in R \Longrightarrow (\lambda x.\ f\ x\ \hat{}\ n) \in R$

⟨*proof*⟩

lemma *setsum*: ⟦*finite I*; ⋀*i*. *i* ∈ *I* ⟹ *f i* ∈ *R*⟧ ⟹ (λ*x*. ∑ *i* ∈ *I*. *f i x*) ∈ *R*
⟨*proof*⟩

lemma *setprod*: ⟦*finite I*; ⋀*i*. *i* ∈ *I* ⟹ *f i* ∈ *R*⟧ ⟹ (λ*x*. ∏ *i* ∈ *I*. *f i x*) ∈ *R*
⟨*proof*⟩

definition *normf* :: (*'a*::*t2-space* ⇒ *real*) ⇒ *real*
where *normf f* ≡ *SUP x*:*s*. |*f x*|

lemma *normf-upper*: ⟦*continuous-on s f*; *x* ∈ *s*⟧ ⟹ |*f x*| ≤ *normf f*
⟨*proof*⟩

lemma *normf-least*: *s* ≠ {} ⟹ (⋀*x*. *x* ∈ *s* ⟹ |*f x*| ≤ *M*) ⟹ *normf f* ≤ *M*
⟨*proof*⟩

end

lemma (**in** *function-ring-on*) *one*:
assumes *U*: *open U* **and** *t0*: *t0* ∈ *s t0* ∈ *U* **and** *t1*: *t1* ∈ *s*−*U*
shows ∃ *V*. *open V* ∧ *t0* ∈ *V* ∧ *s* ∩ *V* ⊆ *U* ∧
(∀ *e*>*0*. ∃*f* ∈ *R*. *f* ' *s* ⊆ {*0*..*1*} ∧ (∀ *t* ∈ *s* ∩ *V*. *f t* < *e*) ∧ (∀ *t* ∈ *s* −
U. *f t* > *1* − *e*))
⟨*proof*⟩

Non-trivial case, with A and B both non-empty

lemma (**in** *function-ring-on*) *two-special*:
assumes *A*: *closed A A* ⊆ *s a* ∈ *A*
and *B*: *closed B B* ⊆ *s b* ∈ *B*
and *disj*: *A* ∩ *B* = {}
and *e*: *0* < *e e* < *1*
shows ∃*f* ∈ *R*. *f* ' *s* ⊆ {*0*..*1*} ∧ (∀ *x* ∈ *A*. *f x* < *e*) ∧ (∀ *x* ∈ *B*. *f x* > *1* − *e*)
⟨*proof*⟩

lemma (**in** *function-ring-on*) *two*:
assumes *A*: *closed A A* ⊆ *s*
and *B*: *closed B B* ⊆ *s*
and *disj*: *A* ∩ *B* = {}
and *e*: *0* < *e e* < *1*
shows ∃*f* ∈ *R*. *f* ' *s* ⊆ {*0*..*1*} ∧ (∀ *x* ∈ *A*. *f x* < *e*) ∧ (∀ *x* ∈ *B*. *f x* > *1* − *e*)
⟨*proof*⟩

The special case where f is non-negative and $e < (1::'a) \,/\, (3::'a)$

lemma (**in** *function-ring-on*) *Stone-Weierstrass-special*:
assumes *f*: *continuous-on s f* **and** *fpos*: ⋀*x*. *x* ∈ *s* ⟹ *f x* ≥ *0*
and *e*: *0* < *e e* < *1*/*3*
shows ∃ *g* ∈ *R*. ∀ *x*∈*s*. |*f x* − *g x*| < *2*∗*e*
⟨*proof*⟩

The "unpretentious" formulation

lemma (**in** *function-ring-on*) *Stone-Weierstrass-basic*:
assumes f: *continuous-on s f* **and** e: $e > 0$
shows $\exists g \in R.\ \forall x{\in}s.\ |f\ x - g\ x| < e$
⟨*proof*⟩

theorem (**in** *function-ring-on*) *Stone-Weierstrass*:
assumes f: *continuous-on s f*
shows $\exists F{\in}UNIV \to R.$ *LIM n sequentially. F n :> uniformly-on s f*
⟨*proof*⟩

A HOL Light formulation

corollary *Stone-Weierstrass-HOL*:
fixes R :: (*'a*::*t2-space* ⇒ *real*) *set* **and** s :: *'a set*
assumes *compact s* $\bigwedge c.\ P(\lambda x.\ c{::}real)$
$\bigwedge f.\ P\ f \Longrightarrow$ *continuous-on s f*
$\bigwedge f\ g.\ P(f) \wedge P(g) \Longrightarrow P(\lambda x.\ f\ x + g\ x)$ $\bigwedge f\ g.\ P(f) \wedge P(g) \Longrightarrow P(\lambda x.\ f\ x * g\ x)$
$\bigwedge x\ y.\ x \in s \wedge y \in s \wedge {\sim}(x = y) \Longrightarrow \exists f.\ P(f) \wedge {\sim}(f\ x = f\ y)$
continuous-on s f
$0 < e$
shows $\exists g.\ P(g) \wedge (\forall x \in s.\ |f\ x - g\ x| < e)$
⟨*proof*⟩

49.4 Polynomial functions

inductive *real-polynomial-function* :: (*'a*::*real-normed-vector* ⇒ *real*) ⇒ *bool* **where**
linear: *bounded-linear f* ⟹ *real-polynomial-function f*
| *const*: *real-polynomial-function* (λ*x*. *c*)
| *add*: ⟦*real-polynomial-function f*; *real-polynomial-function g*⟧ ⟹ *real-polynomial-function* (λ*x*. *f x* + *g x*)
| *mult*: ⟦*real-polynomial-function f*; *real-polynomial-function g*⟧ ⟹ *real-polynomial-function* (λ*x*. *f x* ∗ *g x*)

declare *real-polynomial-function.intros* [*intro*]

definition *polynomial-function* :: (*'a*::*real-normed-vector* ⇒ *'b*::*real-normed-vector*) ⇒ *bool*
where
polynomial-function p ≡ (∀*f*. *bounded-linear f* ⟶ *real-polynomial-function* (*f o p*))

lemma *real-polynomial-function-eq*: *real-polynomial-function p* = *polynomial-function p*
⟨*proof*⟩

lemma *polynomial-function-const* [*iff*]: *polynomial-function* (λ*x*. *c*)
⟨*proof*⟩

lemma *polynomial-function-bounded-linear*:
bounded-linear f $\Longrightarrow$ *polynomial-function f*
⟨*proof*⟩

lemma *polynomial-function-id* [*iff*]: *polynomial-function*($\lambda x.\ x$)
⟨*proof*⟩

lemma *polynomial-function-add* [*intro*]:
⟦*polynomial-function f*; *polynomial-function g*⟧ $\Longrightarrow$ *polynomial-function* ($\lambda x.\ f\ x + g\ x$)
⟨*proof*⟩

lemma *polynomial-function-mult* [*intro*]:
assumes *f*: *polynomial-function f* **and** *g*: *polynomial-function g*
shows *polynomial-function* ($\lambda x.\ f\ x *_R g\ x$)
⟨*proof*⟩

lemma *polynomial-function-cmul* [*intro*]:
assumes *f*: *polynomial-function f*
shows *polynomial-function* ($\lambda x.\ c *_R f\ x$)
⟨*proof*⟩

lemma *polynomial-function-minus* [*intro*]:
assumes *f*: *polynomial-function f*
shows *polynomial-function* ($\lambda x.\ -\ f\ x$)
⟨*proof*⟩

lemma *polynomial-function-diff* [*intro*]:
⟦*polynomial-function f*; *polynomial-function g*⟧ $\Longrightarrow$ *polynomial-function* ($\lambda x.\ f\ x - g\ x$)
⟨*proof*⟩

lemma *polynomial-function-setsum* [*intro*]:
⟦*finite I*; $\bigwedge i.\ i \in I \Longrightarrow$ *polynomial-function* ($\lambda x.\ f\ x\ i$)⟧ $\Longrightarrow$ *polynomial-function* ($\lambda x.\ setsum\ (f\ x)\ I$)
⟨*proof*⟩

lemma *real-polynomial-function-minus* [*intro*]:
real-polynomial-function f $\Longrightarrow$ *real-polynomial-function* ($\lambda x.\ -\ f\ x$)
⟨*proof*⟩

lemma *real-polynomial-function-diff* [*intro*]:
⟦*real-polynomial-function f*; *real-polynomial-function g*⟧ $\Longrightarrow$ *real-polynomial-function* ($\lambda x.\ f\ x - g\ x$)
⟨*proof*⟩

lemma *real-polynomial-function-setsum* [*intro*]:
⟦*finite I*; $\bigwedge i.\ i \in I \Longrightarrow$ *real-polynomial-function* ($\lambda x.\ f\ x\ i$)⟧ $\Longrightarrow$ *real-polynomial-function*

$(\lambda x.\ setsum\ (f\ x)\ I)$
⟨*proof*⟩

lemma *real-polynomial-function-power* [*intro*]:
$real\text{-}polynomial\text{-}function\ f \Longrightarrow real\text{-}polynomial\text{-}function\ (\lambda x.\ f\ x\ \hat{}\ n)$
⟨*proof*⟩

lemma *real-polynomial-function-compose* [*intro*]:
assumes f: *polynomial-function f* **and** g: *real-polynomial-function g*
shows *real-polynomial-function* $(g\ o\ f)$
⟨*proof*⟩

lemma *polynomial-function-compose* [*intro*]:
assumes f: *polynomial-function f* **and** g: *polynomial-function g*
shows *polynomial-function* $(g\ o\ f)$
⟨*proof*⟩

lemma *setsum-max-0*:
fixes x::*real*
shows $(\sum i = 0..max\ m\ n.\ x\hat{}i * (if\ i \leq m\ then\ a\ i\ else\ 0)) = (\sum i = 0..m.\ x\hat{}i * a\ i)$
⟨*proof*⟩

lemma *real-polynomial-function-imp-setsum*:
assumes *real-polynomial-function f*
shows $\exists\ a\ n{::}nat.\ f = (\lambda x.\ \sum i{=}0..n.\ a\ i * x\ \hat{}\ i)$
⟨*proof*⟩

lemma *real-polynomial-function-iff-setsum*:
$real\text{-}polynomial\text{-}function\ f \longleftrightarrow (\exists\ a\ n{::}nat.\ f = (\lambda x.\ \sum i{=}0..n.\ a\ i * x\ \hat{}\ i))$
⟨*proof*⟩

lemma *polynomial-function-iff-Basis-inner*:
fixes $f :: 'a{::}real\text{-}normed\text{-}vector \Rightarrow 'b{::}euclidean\text{-}space$
shows $polynomial\text{-}function\ f \longleftrightarrow (\forall\ b{\in}Basis.\ real\text{-}polynomial\text{-}function\ (\lambda x.\ inner\ (f\ x)\ b))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

49.5 Stone-Weierstrass theorem for polynomial functions

First, we need to show that they are continous, differentiable and separable.

lemma *continuous-real-polymonial-function*:
assumes *real-polynomial-function f*
shows *continuous* $(at\ x)\ f$
⟨*proof*⟩

lemma *continuous-polymonial-function*:
fixes $f :: 'a{::}real\text{-}normed\text{-}vector \Rightarrow 'b{::}euclidean\text{-}space$

assumes *polynomial-function f*
shows *continuous* (*at x*) *f*
⟨*proof*⟩

lemma *continuous-on-polymonial-function*:
fixes *f* :: *'a*::*real-normed-vector* ⇒ *'b*::*euclidean-space*
assumes *polynomial-function f*
shows *continuous-on s f*
⟨*proof*⟩

lemma *has-real-derivative-polynomial-function*:
assumes *real-polynomial-function p*
shows ∃ *p'*. *real-polynomial-function p'* ∧
(∀ *x*. (*p has-real-derivative* (*p' x*)) (*at x*))
⟨*proof*⟩

lemma *has-vector-derivative-polynomial-function*:
fixes *p* :: *real* ⇒ *'a*::*euclidean-space*
assumes *polynomial-function p*
shows ∃ *p'*. *polynomial-function p'* ∧
(∀ *x*. (*p has-vector-derivative* (*p' x*)) (*at x*))
⟨*proof*⟩

lemma *real-polynomial-function-separable*:
fixes *x* :: *'a*::*euclidean-space*
assumes *x* ≠ *y* **shows** ∃ *f*. *real-polynomial-function f* ∧ *f x* ≠ *f y*
⟨*proof*⟩

lemma *Stone-Weierstrass-real-polynomial-function*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *real*
assumes *compact s continuous-on s f* 0 < *e*
shows ∃ *g*. *real-polynomial-function g* ∧ (∀ *x* ∈ *s*. |*f x* − *g x*| < *e*)
⟨*proof*⟩

lemma *Stone-Weierstrass-polynomial-function*:
fixes *f* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *s*: *compact s*
and *f*: *continuous-on s f*
and *e*: 0 < *e*
shows ∃ *g*. *polynomial-function g* ∧ (∀ *x* ∈ *s*. *norm*(*f x* − *g x*) < *e*)
⟨*proof*⟩

49.6 Polynomial functions as paths

One application is to pick a smooth approximation to a path, or just pick a smooth path anyway in an open connected set

lemma *path-polynomial-function*:
fixes *g* :: *real* ⇒ *'b*::*euclidean-space*
shows *polynomial-function g* ⟹ *path g*

⟨*proof*⟩

lemma *path-approx-polynomial-function*:
fixes $g :: real \Rightarrow 'b::euclidean\text{-}space$
assumes *path g 0* $< e$
shows $\exists p.$ *polynomial-function p* $\wedge$
pathstart p = *pathstart g* $\wedge$
pathfinish p = *pathfinish g* $\wedge$
$(\forall t \in \{0..1\}.$ *norm*$(p\ t - g\ t) < e)$
⟨*proof*⟩

lemma *connected-open-polynomial-connected*:
fixes $s :: 'a::euclidean\text{-}space\ set$
assumes *s*: *open s connected s*
and $x \in s\ y \in s$
shows $\exists g.$ *polynomial-function g* $\wedge$ *path-image g* $\subseteq s \wedge$
pathstart g = $x \wedge$ *pathfinish g* = y
⟨*proof*⟩

hide-fact *linear add mult const*

end

50 Non-negative, non-positive integers and reals

theory *Nonpos-Ints*
imports *Complex-Main*
begin

50.1 Non-positive integers

The set of non-positive integers on a ring. (in analogy to the set of non-negative integers $\mathbb{N}$) This is useful e.g. for the Gamma function.

definition *nonpos-Ints* ($\mathbb{Z}_{\leq 0}$) **where** $\mathbb{Z}_{\leq 0} = \{$*of-int n* $|n.\ n \leq 0\}$

lemma *zero-in-nonpos-Ints* [*simp*,*intro*]: $0 \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *neg-one-in-nonpos-Ints* [*simp*,*intro*]: $-1 \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *neg-numeral-in-nonpos-Ints* [*simp*,*intro*]: $-numeral\ n \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *one-notin-nonpos-Ints* [*simp*]: $(1 :: 'a :: ring\text{-}char\text{-}0) \notin \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *numeral-notin-nonpos-Ints* [*simp*]: $(numeral\ n :: 'a :: ring\text{-}char\text{-}0) \notin \mathbb{Z}_{\leq 0}$

⟨*proof*⟩

lemma *minus-of-nat-in-nonpos-Ints* [*simp*, *intro*]: $-$ *of-nat* $n \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *of-nat-in-nonpos-Ints-iff*: (*of-nat* n :: $'a$ :: {*ring-1*,*ring-char-0*}) $\in \mathbb{Z}_{\leq 0}$ $\longleftrightarrow$ $n = 0$
⟨*proof*⟩

lemma *nonpos-Ints-of-int*: $n \leq 0 \Longrightarrow$ *of-int* $n \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-IntsI*:
$x \in \mathbb{Z} \Longrightarrow x \leq 0 \Longrightarrow (x :: 'a :: linordered\text{-}idom) \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Ints-subset-Ints*: $\mathbb{Z}_{\leq 0} \subseteq \mathbb{Z}$
⟨*proof*⟩

lemma *nonpos-Ints-nonpos* [*dest*]: $x \in \mathbb{Z}_{\leq 0} \Longrightarrow x \leq (0 :: 'a :: linordered\text{-}idom)$
⟨*proof*⟩

lemma *nonpos-Ints-Int* [*dest*]: $x \in \mathbb{Z}_{\leq 0} \Longrightarrow x \in \mathbb{Z}$
⟨*proof*⟩

lemma *nonpos-Ints-cases*:
assumes $x \in \mathbb{Z}_{\leq 0}$
obtains n **where** $x =$ *of-int* n $n \leq 0$
⟨*proof*⟩

lemma *nonpos-Ints-cases'*:
assumes $x \in \mathbb{Z}_{\leq 0}$
obtains n **where** $x = -$*of-nat* n
⟨*proof*⟩

lemma *of-real-in-nonpos-Ints-iff*: (*of-real* x :: $'a$ :: *real-algebra-1*) $\in \mathbb{Z}_{\leq 0} \longleftrightarrow x \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Ints-altdef*: $\mathbb{Z}_{\leq 0} = \{n \in \mathbb{Z}.\ (n :: 'a :: linordered\text{-}idom) \leq 0\}$
⟨*proof*⟩

lemma *uminus-in-Nats-iff*: $-x \in \mathbb{N} \longleftrightarrow x \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *uminus-in-nonpos-Ints-iff*: $-x \in \mathbb{Z}_{\leq 0} \longleftrightarrow x \in \mathbb{N}$
⟨*proof*⟩

lemma *nonpos-Ints-mult*: $x \in \mathbb{Z}_{\leq 0} \Longrightarrow y \in \mathbb{Z}_{\leq 0} \Longrightarrow x * y \in \mathbb{N}$

⟨*proof*⟩

lemma *Nats-mult-nonpos-Ints*: $x \in \mathbb{N} \Longrightarrow y \in \mathbb{Z}_{\leq 0} \Longrightarrow x * y \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Ints-mult-Nats*:
$x \in \mathbb{Z}_{\leq 0} \Longrightarrow y \in \mathbb{N} \Longrightarrow x * y \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Ints-add*:
$x \in \mathbb{Z}_{\leq 0} \Longrightarrow y \in \mathbb{Z}_{\leq 0} \Longrightarrow x + y \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Ints-diff-Nats*:
$x \in \mathbb{Z}_{\leq 0} \Longrightarrow y \in \mathbb{N} \Longrightarrow x - y \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *Nats-diff-nonpos-Ints*:
$x \in \mathbb{N} \Longrightarrow y \in \mathbb{Z}_{\leq 0} \Longrightarrow x - y \in \mathbb{N}$
⟨*proof*⟩

lemma *plus-of-nat-eq-0-imp*: $z + \textit{of-nat}\ n = 0 \Longrightarrow z \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

50.2 Non-negative reals

definition *nonneg-Reals* :: *'a*::*real-algebra-1 set* $(\mathbb{R}_{\geq 0})$
where $\mathbb{R}_{\geq 0} = \{\textit{of-real}\ r \mid r.\ r \geq 0\}$

lemma *nonneg-Reals-of-real-iff* [*simp*]: $\textit{of-real}\ r \in \mathbb{R}_{\geq 0} \longleftrightarrow r \geq 0$
⟨*proof*⟩

lemma *nonneg-Reals-subset-Reals*: $\mathbb{R}_{\geq 0} \subseteq \mathbb{R}$
⟨*proof*⟩

lemma *nonneg-Reals-Real* [*dest*]: $x \in \mathbb{R}_{\geq 0} \Longrightarrow x \in \mathbb{R}$
⟨*proof*⟩

lemma *nonneg-Reals-of-nat-I* [*simp*]: $\textit{of-nat}\ n \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-cases*:
assumes $x \in \mathbb{R}_{\geq 0}$
obtains r **where** $x = \textit{of-real}\ r\ r \geq 0$
⟨*proof*⟩

lemma *nonneg-Reals-zero-I* [*simp*]: $0 \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-one-I* [*simp*]: $1 \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-minus-one-I* [*simp*]: $-1 \notin \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-numeral-I* [*simp*]: $numeral\ w \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-minus-numeral-I* [*simp*]: $-\ numeral\ w \notin \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-add-I* [*simp*]: $[\![a \in \mathbb{R}_{\geq 0};\ b \in \mathbb{R}_{\geq 0}]\!] \Longrightarrow a + b \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-mult-I* [*simp*]: $[\![a \in \mathbb{R}_{\geq 0};\ b \in \mathbb{R}_{\geq 0}]\!] \Longrightarrow a * b \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-inverse-I* [*simp*]:
fixes a :: *'a*::*real-div-algebra*
shows $a \in \mathbb{R}_{\geq 0} \Longrightarrow inverse\ a \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-divide-I* [*simp*]:
fixes a :: *'a*::*real-div-algebra*
shows $[\![a \in \mathbb{R}_{\geq 0};\ b \in \mathbb{R}_{\geq 0}]\!] \Longrightarrow a\ /\ b \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonneg-Reals-pow-I* [*simp*]: $a \in \mathbb{R}_{\geq 0} \Longrightarrow a\hat{}n \in \mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *complex-nonneg-Reals-iff*: $z \in \mathbb{R}_{\geq 0} \longleftrightarrow Re\ z \geq 0 \wedge Im\ z = 0$
⟨*proof*⟩

lemma *ii-not-nonneg-Reals* [*iff*]: $\mathrm{i} \notin \mathbb{R}_{\geq 0}$
⟨*proof*⟩

50.3 Non-positive reals

definition *nonpos-Reals* :: *'a*::*real-algebra-1 set* $(\mathbb{R}_{\leq 0})$
where $\mathbb{R}_{\leq 0} = \{of\text{-}real\ r \mid r.\ r \leq 0\}$

lemma *nonpos-Reals-of-real-iff* [*simp*]: $of\text{-}real\ r \in \mathbb{R}_{\leq 0} \longleftrightarrow r \leq 0$
⟨*proof*⟩

lemma *nonpos-Reals-subset-Reals*: $\mathbb{R}_{\leq 0} \subseteq \mathbb{R}$
⟨*proof*⟩

lemma *nonpos-Ints-subset-nonpos-Reals*: $\mathbb{Z}_{\leq 0} \subseteq \mathbb{R}_{\leq 0}$

⟨*proof*⟩

lemma *nonpos-Reals-of-nat-iff* [*simp*]: *of-nat n* ∈ $\mathbb{R}_{\leq 0}$ ⟷ *n=0*
⟨*proof*⟩

lemma *nonpos-Reals-Real* [*dest*]: *x* ∈ $\mathbb{R}_{\leq 0}$ ⟹ *x* ∈ $\mathbb{R}$
⟨*proof*⟩

lemma *nonpos-Reals-cases*:
assumes *x* ∈ $\mathbb{R}_{\leq 0}$
obtains *r* **where** *x* = *of-real r r* ≤ *0*
⟨*proof*⟩

lemma *uminus-nonneg-Reals-iff* [*simp*]: −*x* ∈ $\mathbb{R}_{\geq 0}$ ⟷ *x* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *uminus-nonpos-Reals-iff* [*simp*]: −*x* ∈ $\mathbb{R}_{\leq 0}$ ⟷ *x* ∈ $\mathbb{R}_{\geq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-zero-I* [*simp*]: *0* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-one-I* [*simp*]: *1* ∉ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-numeral-I* [*simp*]: *numeral w* ∉ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-add-I* [*simp*]: ⟦*a* ∈ $\mathbb{R}_{\leq 0}$; *b* ∈ $\mathbb{R}_{\leq 0}$⟧ ⟹ *a* + *b* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-mult-I1*: ⟦*a* ∈ $\mathbb{R}_{\geq 0}$; *b* ∈ $\mathbb{R}_{\leq 0}$⟧ ⟹ *a* ∗ *b* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-mult-I2*: ⟦*a* ∈ $\mathbb{R}_{\leq 0}$; *b* ∈ $\mathbb{R}_{\geq 0}$⟧ ⟹ *a* ∗ *b* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-mult-of-nat-iff*:
fixes *a*:: *'a* :: *real-div-algebra* **shows** *a* ∗ *of-nat n* ∈ $\mathbb{R}_{\leq 0}$ ⟷ *a* ∈ $\mathbb{R}_{\leq 0}$ ∨ *n=0*
⟨*proof*⟩

lemma *nonpos-Reals-inverse-I*:
fixes *a* :: *'a*::*real-div-algebra*
shows *a* ∈ $\mathbb{R}_{\leq 0}$ ⟹ *inverse a* ∈ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-divide-I1*:
fixes *a* :: *'a*::*real-div-algebra*
shows ⟦*a* ∈ $\mathbb{R}_{\geq 0}$; *b* ∈ $\mathbb{R}_{\leq 0}$⟧ ⟹ *a* / *b* ∈ $\mathbb{R}_{\leq 0}$

⟨*proof*⟩

lemma *nonpos-Reals-divide-I2*:
fixes a :: $'a$::*real-div-algebra*
shows $[\![a \in \mathbb{R}_{\leq 0};\ b \in \mathbb{R}_{\geq 0}]\!] \Longrightarrow a\ /\ b \in \mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *nonpos-Reals-divide-of-nat-iff*:
fixes a:: $'a$ :: *real-div-algebra* **shows** $a\ /\ \textit{of-nat}\ n \in \mathbb{R}_{\leq 0} \longleftrightarrow a \in \mathbb{R}_{\leq 0} \vee n{=}0$
⟨*proof*⟩

lemma *nonpos-Reals-pow-I*: $[\![a \in \mathbb{R}_{\leq 0};\ \textit{odd}\ n]\!] \Longrightarrow a\hat{}n \in \mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *complex-nonpos-Reals-iff*: $z \in \mathbb{R}_{\leq 0} \longleftrightarrow \textit{Re}\ z \leq 0 \wedge \textit{Im}\ z = 0$
⟨*proof*⟩

lemma *ii-not-nonpos-Reals* [*iff*]: $\mathrm{i} \notin \mathbb{R}_{\leq 0}$
⟨*proof*⟩

end

51 Complex Analysis Basics

theory *Complex-Analysis-Basics*
imports *Cartesian-Euclidean-Space* ~~/*src*/*HOL*/*Library*/*Nonpos-Ints*
begin

51.1 General lemmas

lemma *nonneg-Reals-cmod-eq-Re*: $z \in \mathbb{R}_{\geq 0} \Longrightarrow \textit{norm}\ z = \textit{Re}\ z$
⟨*proof*⟩

lemma *has-derivative-mult-right*:
fixes c:: $'a$ :: *real-normed-algebra*
shows $((\textit{op} * c)\ \textit{has-derivative}\ (\textit{op} * c))\ F$
⟨*proof*⟩

lemma *has-derivative-of-real*[*derivative-intros*, *simp*]:
$(f\ \textit{has-derivative}\ f')\ F \Longrightarrow ((\lambda x.\ \textit{of-real}\ (f\ x))\ \textit{has-derivative}\ (\lambda x.\ \textit{of-real}\ (f'\ x)))\ F$
⟨*proof*⟩

lemma *has-vector-derivative-real-complex*:
$\textit{DERIV}\ f\ (\textit{of-real}\ a) :> f' \Longrightarrow ((\lambda x.\ f\ (\textit{of-real}\ x))\ \textit{has-vector-derivative}\ f')\ (\textit{at}\ a\ \textit{within}\ s)$
⟨*proof*⟩

lemma *fact-cancel*:

fixes $c :: 'a$::*real-field*
shows *of-nat* (*Suc n*) $*$ c / (*fact* (*Suc n*)) = c / (*fact n*)
⟨*proof*⟩

lemma *bilinear-times*:
fixes c::$'a$::*real-algebra* **shows** *bilinear* ($\lambda x\ y$::$'a$. $x*y$)
⟨*proof*⟩

lemma *linear-cnj*: *linear cnj*
⟨*proof*⟩

lemma *tendsto-Re-upper*:
assumes ~ (*trivial-limit F*)
(*f* ⟶ *l*) *F*
eventually (λx. *Re*(*f x*) ≤ *b*) *F*
shows *Re*(*l*) ≤ *b*
⟨*proof*⟩

lemma *tendsto-Re-lower*:
assumes ~ (*trivial-limit F*)
(*f* ⟶ *l*) *F*
eventually (λx. *b* ≤ *Re*(*f x*)) *F*
shows *b* ≤ *Re*(*l*)
⟨*proof*⟩

lemma *tendsto-Im-upper*:
assumes ~ (*trivial-limit F*)
(*f* ⟶ *l*) *F*
eventually (λx. *Im*(*f x*) ≤ *b*) *F*
shows *Im*(*l*) ≤ *b*
⟨*proof*⟩

lemma *tendsto-Im-lower*:
assumes ~ (*trivial-limit F*)
(*f* ⟶ *l*) *F*
eventually (λx. *b* ≤ *Im*(*f x*)) *F*
shows *b* ≤ *Im*(*l*)
⟨*proof*⟩

lemma *lambda-zero*: (λh::$'a$::*mult-zero*. *0*) = *op* $*$ *0*
⟨*proof*⟩

lemma *lambda-one*: (λx::$'a$::*monoid-mult*. x) = *op* $*$ *1*
⟨*proof*⟩

lemma *continuous-mult-left*:
fixes c::$'a$::*real-normed-algebra*
shows *continuous F f* ⟹ *continuous F* (λx. c $*$ *f x*)
⟨*proof*⟩

lemma *continuous-mult-right*:
fixes *c*::*'a*::*real-normed-algebra*
shows *continuous F f* $\Longrightarrow$ *continuous F* $(\lambda x.\ f\ x * c)$
⟨*proof*⟩

lemma *continuous-on-mult-left*:
fixes *c*::*'a*::*real-normed-algebra*
shows *continuous-on s f* $\Longrightarrow$ *continuous-on s* $(\lambda x.\ c * f\ x)$
⟨*proof*⟩

lemma *continuous-on-mult-right*:
fixes *c*::*'a*::*real-normed-algebra*
shows *continuous-on s f* $\Longrightarrow$ *continuous-on s* $(\lambda x.\ f\ x * c)$
⟨*proof*⟩

lemma *uniformly-continuous-on-cmul-right* [*continuous-intros*]:
fixes *f* :: *'a*::*real-normed-vector* $\Rightarrow$ *'b*::*real-normed-algebra*
shows *uniformly-continuous-on s f* $\Longrightarrow$ *uniformly-continuous-on s* $(\lambda x.\ f\ x * c)$
⟨*proof*⟩

lemma *uniformly-continuous-on-cmul-left*[*continuous-intros*]:
fixes *f* :: *'a*::*real-normed-vector* $\Rightarrow$ *'b*::*real-normed-algebra*
assumes *uniformly-continuous-on s f*
shows *uniformly-continuous-on s* $(\lambda x.\ c * f\ x)$
⟨*proof*⟩

lemma *continuous-within-norm-id* [*continuous-intros*]: *continuous* (*at x within S*) *norm*
⟨*proof*⟩

lemma *continuous-on-norm-id* [*continuous-intros*]: *continuous-on S norm*
⟨*proof*⟩

51.2 DERIV stuff

lemma *DERIV-zero-connected-constant*:
fixes *f* :: *'a*::{*real-normed-field*,*euclidean-space*} $\Rightarrow$ *'a*
assumes *connected s*
and *open s*
and *finite k*
and *continuous-on s f*
and $\forall x{\in}(s - k).$ *DERIV f x* :> *0*
obtains *c* **where** $\bigwedge x.\ x \in s \Longrightarrow f(x) = c$
⟨*proof*⟩

lemma *DERIV-zero-constant*:
fixes *f* :: *'a*::{*real-normed-field*, *real-inner*} $\Rightarrow$ *'a*
shows ⟦*convex s*;

$\bigwedge x.\ x{\in}s \Longrightarrow (f \textit{ has-field-derivative } 0)\ (\textit{at } x \textit{ within } s)]\!]$
$\Longrightarrow \exists\, c.\ \forall\, x \in s.\ f(x) = c$
⟨*proof*⟩

lemma *DERIV-zero-unique*:
fixes f :: $'a$::{*real-normed-field*, *real-inner*} ⇒ $'a$
assumes *convex s*
and *d0*: $\bigwedge x.\ x{\in}s \Longrightarrow (f \textit{ has-field-derivative } 0)\ (\textit{at } x \textit{ within } s)$
and $a \in s$
and $x \in s$
shows $f\ x = f\ a$
⟨*proof*⟩

lemma *DERIV-zero-connected-unique*:
fixes f :: $'a$::{*real-normed-field*, *real-inner*} ⇒ $'a$
assumes *connected s*
and *open s*
and *d0*: $\bigwedge x.\ x{\in}s \Longrightarrow \textit{DERIV } f\ x :> 0$
and $a \in s$
and $x \in s$
shows $f\ x = f\ a$
⟨*proof*⟩

lemma *DERIV-transform-within*:
assumes $(f \textit{ has-field-derivative } f')\ (\textit{at } a \textit{ within } s)$
and $0 < d\ a \in s$
and $\bigwedge x.\ x{\in}s \Longrightarrow \textit{dist } x\ a < d \Longrightarrow f\ x = g\ x$
shows $(g \textit{ has-field-derivative } f')\ (\textit{at } a \textit{ within } s)$
⟨*proof*⟩

lemma *DERIV-transform-within-open*:
assumes $\textit{DERIV } f\ a :> f'$
and *open s* $a \in s$
and $\bigwedge x.\ x{\in}s \Longrightarrow f\ x = g\ x$
shows $\textit{DERIV } g\ a :> f'$
⟨*proof*⟩

lemma *DERIV-transform-at*:
assumes $\textit{DERIV } f\ a :> f'$
and $0 < d$
and $\bigwedge x.\ \textit{dist } x\ a < d \Longrightarrow f\ x = g\ x$
shows $\textit{DERIV } g\ a :> f'$
⟨*proof*⟩

lemma *DERIV-zero-UNIV-unique*:
fixes f :: $'a$::{*real-normed-field*, *real-inner*} ⇒ $'a$
shows $(\bigwedge x.\ \textit{DERIV } f\ x :> 0) \Longrightarrow f\ x = f\ a$
⟨*proof*⟩

51.3 Some limit theorems about real part of real series etc.

lemma *sums-vec-nth* :
assumes f *sums* a
shows $(\lambda x.\ f\ x\ \$\ i)$ *sums* $a\ \$\ i$
⟨*proof*⟩

lemma *summable-vec-nth* :
assumes *summable* f
shows *summable* $(\lambda x.\ f\ x\ \$\ i)$
⟨*proof*⟩

51.4 Complex number lemmas

lemma
shows *open-halfspace-Re-lt*: *open* $\{z.\ Re(z) < b\}$
and *open-halfspace-Re-gt*: *open* $\{z.\ Re(z) > b\}$
and *closed-halfspace-Re-ge*: *closed* $\{z.\ Re(z) \geq b\}$
and *closed-halfspace-Re-le*: *closed* $\{z.\ Re(z) \leq b\}$
and *closed-halfspace-Re-eq*: *closed* $\{z.\ Re(z) = b\}$
and *open-halfspace-Im-lt*: *open* $\{z.\ Im(z) < b\}$
and *open-halfspace-Im-gt*: *open* $\{z.\ Im(z) > b\}$
and *closed-halfspace-Im-ge*: *closed* $\{z.\ Im(z) \geq b\}$
and *closed-halfspace-Im-le*: *closed* $\{z.\ Im(z) \leq b\}$
and *closed-halfspace-Im-eq*: *closed* $\{z.\ Im(z) = b\}$
⟨*proof*⟩

lemma *closed-complex-Reals*: *closed* $(\mathbb{R} :: \textit{complex set})$
⟨*proof*⟩

lemma *closed-Real-halfspace-Re-le*: *closed* $(\mathbb{R} \cap \{w.\ Re\ w \leq x\})$
⟨*proof*⟩

corollary *closed-nonpos-Reals-complex* [*simp*]: *closed* $(\mathbb{R}_{\leq 0} :: \textit{complex set})$
⟨*proof*⟩

lemma *closed-Real-halfspace-Re-ge*: *closed* $(\mathbb{R} \cap \{w.\ x \leq Re(w)\})$
⟨*proof*⟩

corollary *closed-nonneg-Reals-complex* [*simp*]: *closed* $(\mathbb{R}_{\geq 0} :: \textit{complex set})$
⟨*proof*⟩

lemma *closed-real-abs-le*: *closed* $\{w \in \mathbb{R}.\ |Re\ w| \leq r\}$
⟨*proof*⟩

lemma *real-lim*:
fixes l::*complex*
assumes $(f \longrightarrow l)\ F$ **and** ~(*trivial-limit* F) **and** *eventually* $P\ F$ **and** $\bigwedge a.\ P\ a \Longrightarrow f\ a \in \mathbb{R}$
shows $l \in \mathbb{R}$

⟨*proof*⟩

lemma *real-lim-sequentially*:
fixes *l*::*complex*
shows (*f* ⟶ *l*) *sequentially* ⟹ (∃ *N*. ∀ *n*≥*N*. *f n* ∈ ℝ) ⟹ *l* ∈ ℝ
⟨*proof*⟩

lemma *real-series*:
fixes *l*::*complex*
shows *f sums l* ⟹ (⋀*n*. *f n* ∈ ℝ) ⟹ *l* ∈ ℝ
⟨*proof*⟩

lemma *Lim-null-comparison-Re*:
assumes *eventually* (λ*x*. *norm*(*f x*) ≤ *Re*(*g x*)) *F* (*g* ⟶ 0) *F* **shows** (*f* ⟶ 0) *F*
⟨*proof*⟩

51.5 Holomorphic functions

definition *field-differentiable* :: ['*a* ⇒ '*a*::*real-normed-field*, '*a filter*] ⇒ *bool*
(**infixr** (*field'-differentiable*) 50)
where *f field-differentiable F* ≡ ∃ *f'*. (*f has-field-derivative f'*) *F*

lemma *field-differentiable-derivI*:
f field-differentiable (*at x*) ⟹ (*f has-field-derivative deriv f x*) (*at x*)
⟨*proof*⟩

lemma *field-differentiable-imp-continuous-at*:
f field-differentiable (*at x within s*) ⟹ *continuous* (*at x within s*) *f*
⟨*proof*⟩

lemma *field-differentiable-within-subset*:
⟦*f field-differentiable* (*at x within s*); *t* ⊆ *s*⟧
⟹ *f field-differentiable* (*at x within t*)
⟨*proof*⟩

lemma *field-differentiable-at-within*:
⟦*f field-differentiable* (*at x*)⟧
⟹ *f field-differentiable* (*at x within s*)
⟨*proof*⟩

lemma *field-differentiable-linear* [*simp*,*derivative-intros*]: (*op* ∗ *c*) *field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-const* [*simp*,*derivative-intros*]: (λ*z*. *c*) *field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-ident* [*simp*,*derivative-intros*]: $(\lambda z.\ z)$ *field-differentiable* *F*
⟨*proof*⟩

lemma *field-differentiable-id* [*simp*,*derivative-intros*]: *id field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-minus* [*derivative-intros*]:
f field-differentiable F $\Longrightarrow$ $(\lambda z.\ -\ (f\ z))$ *field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-add* [*derivative-intros*]:
assumes *f field-differentiable F g field-differentiable F*
shows $(\lambda z.\ f\ z\ +\ g\ z)$ *field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-add-const* [*simp*,*derivative-intros*]:
op + *c field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-setsum* [*derivative-intros*]:
$(\bigwedge i.\ i \in I \Longrightarrow (f\ i)$ *field-differentiable F*$) \Longrightarrow (\lambda z.\ \sum i{\in}I.\ f\ i\ z)$ *field-differentiable* *F*
⟨*proof*⟩

lemma *field-differentiable-diff* [*derivative-intros*]:
assumes *f field-differentiable F g field-differentiable F*
shows $(\lambda z.\ f\ z\ -\ g\ z)$ *field-differentiable F*
⟨*proof*⟩

lemma *field-differentiable-inverse* [*derivative-intros*]:
assumes *f field-differentiable* (*at a within s*) $f\ a \neq 0$
shows $(\lambda z.$ *inverse* $(f\ z))$ *field-differentiable* (*at a within s*)
⟨*proof*⟩

lemma *field-differentiable-mult* [*derivative-intros*]:
assumes *f field-differentiable* (*at a within s*)
g field-differentiable (*at a within s*)
shows $(\lambda z.\ f\ z\ *\ g\ z)$ *field-differentiable* (*at a within s*)
⟨*proof*⟩

lemma *field-differentiable-divide* [*derivative-intros*]:
assumes *f field-differentiable* (*at a within s*)
g field-differentiable (*at a within s*)
$g\ a \neq 0$
shows $(\lambda z.\ f\ z\ /\ g\ z)$ *field-differentiable* (*at a within s*)
⟨*proof*⟩

lemma *field-differentiable-power* [*derivative-intros*]:

assumes *f field-differentiable* (*at a within s*)
shows (λ*z*. *f z* ^ *n*) *field-differentiable* (*at a within s*)
⟨*proof*⟩

lemma *field-differentiable-transform-within*:
0 < *d* ⟹
x ∈ *s* ⟹
(⋀*x'*. *x'* ∈ *s* ⟹ *dist x' x* < *d* ⟹ *f x'* = *g x'*) ⟹
f field-differentiable (*at x within s*)
⟹ *g field-differentiable* (*at x within s*)
⟨*proof*⟩

lemma *field-differentiable-compose-within*:
assumes *f field-differentiable* (*at a within s*)
g field-differentiable (*at* (*f a*) *within f*'*s*)
shows (*g o f*) *field-differentiable* (*at a within s*)
⟨*proof*⟩

lemma *field-differentiable-compose*:
f field-differentiable at z ⟹ *g field-differentiable at* (*f z*)
⟹ (*g o f*) *field-differentiable at z*
⟨*proof*⟩

lemma *field-differentiable-within-open*:
⟦*a* ∈ *s*; *open s*⟧ ⟹ *f field-differentiable at a within s* ⟷
f field-differentiable at a
⟨*proof*⟩

51.6 Caratheodory characterization

lemma *field-differentiable-caratheodory-at*:
f field-differentiable (*at z*) ⟷
(∃ *g*. (∀ *w*. *f*(*w*) − *f*(*z*) = *g*(*w*) ∗ (*w* − *z*)) ∧ *continuous* (*at z*) *g*)
⟨*proof*⟩

lemma *field-differentiable-caratheodory-within*:
f field-differentiable (*at z within s*) ⟷
(∃ *g*. (∀ *w*. *f*(*w*) − *f*(*z*) = *g*(*w*) ∗ (*w* − *z*)) ∧ *continuous* (*at z within s*) *g*)
⟨*proof*⟩

51.7 Holomorphic

definition *holomorphic-on* :: [*complex* ⇒ *complex*, *complex set*] ⇒ *bool*
(**infixl** (*holomorphic'-on*) *50*)
where *f holomorphic-on s* ≡ ∀ *x*∈*s*. *f field-differentiable* (*at x within s*)

named-theorems *holomorphic-intros structural introduction rules for holomorphic-on*

lemma *holomorphic-onI* [*intro?*]: (⋀*x*. *x* ∈ *s* ⟹ *f field-differentiable* (*at x within s*)) ⟹ *f holomorphic-on s*

⟨*proof*⟩

lemma *holomorphic-onD* [*dest?*]: ⟦*f holomorphic-on s*; *x* ∈ *s*⟧ ⟹ *f field-differentiable* (*at x within s*)
⟨*proof*⟩

lemma *holomorphic-on-imp-differentiable-at*:
⟦*f holomorphic-on s*; *open s*; *x* ∈ *s*⟧ ⟹ *f field-differentiable* (*at x*)
⟨*proof*⟩

lemma *holomorphic-on-empty* [*holomorphic-intros*]: *f holomorphic-on* {}
⟨*proof*⟩

lemma *holomorphic-on-open*:
open s ⟹ *f holomorphic-on s* ⟷ (∀ *x* ∈ *s*. ∃ *f'*. *DERIV f x* :> *f'*)
⟨*proof*⟩

lemma *holomorphic-on-imp-continuous-on*:
f holomorphic-on s ⟹ *continuous-on s f*
⟨*proof*⟩

lemma *holomorphic-on-subset* [*elim*]:
f holomorphic-on s ⟹ *t* ⊆ *s* ⟹ *f holomorphic-on t*
⟨*proof*⟩

lemma *holomorphic-transform*: ⟦*f holomorphic-on s*; ⋀*x*. *x* ∈ *s* ⟹ *f x* = *g x*⟧
⟹ *g holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-cong*: *s* = *t* ==> (⋀*x*. *x* ∈ *s* ⟹ *f x* = *g x*) ⟹ *f holomorphic-on s* ⟷ *g holomorphic-on t*
⟨*proof*⟩

lemma *holomorphic-on-linear* [*simp*, *holomorphic-intros*]: (*op* ∗ *c*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-const* [*simp*, *holomorphic-intros*]: (λ*z*. *c*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-ident* [*simp*, *holomorphic-intros*]: (λ*x*. *x*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-id* [*simp*, *holomorphic-intros*]: *id holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-compose*:

f holomorphic-on s $\Longrightarrow$ *g holomorphic-on* (*f* ' *s*) $\Longrightarrow$ (*g o f*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-compose-gen*:
f holomorphic-on s $\Longrightarrow$ *g holomorphic-on t* $\Longrightarrow$ *f* ' *s* $\subseteq$ *t* $\Longrightarrow$ (*g o f*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-minus* [*holomorphic-intros*]: *f holomorphic-on s* $\Longrightarrow$ ($\lambda z.$ $-$(*f z*)) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-add* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; *g holomorphic-on s*⟧ $\Longrightarrow$ ($\lambda z.$ *f z* + *g z*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-diff* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; *g holomorphic-on s*⟧ $\Longrightarrow$ ($\lambda z.$ *f z* $-$ *g z*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-mult* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; *g holomorphic-on s*⟧ $\Longrightarrow$ ($\lambda z.$ *f z* $*$ *g z*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-inverse* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; $\bigwedge z.$ *z* $\in$ *s* $\Longrightarrow$ *f z* $\neq$ *0*⟧ $\Longrightarrow$ ($\lambda z.$ *inverse* (*f z*)) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-divide* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; *g holomorphic-on s*; $\bigwedge z.$ *z* $\in$ *s* $\Longrightarrow$ *g z* $\neq$ *0*⟧ $\Longrightarrow$ ($\lambda z.$ *f z* / *g z*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-power* [*holomorphic-intros*]:
f holomorphic-on s $\Longrightarrow$ ($\lambda z.$ (*f z*) ^ *n*) *holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-setsum* [*holomorphic-intros*]:
($\bigwedge i.$ *i* $\in$ *I* $\Longrightarrow$ (*f i*) *holomorphic-on s*) $\Longrightarrow$ ($\lambda x.$ *setsum* ($\lambda i.$ *f i x*) *I*) *holomorphic-on s*
⟨*proof*⟩

lemma *DERIV-deriv-iff-field-differentiable*:
DERIV f x :> *deriv f x* $\longleftrightarrow$ *f field-differentiable at x*
⟨*proof*⟩

lemma *holomorphic-derivI*:
⟦*f holomorphic-on S*; *open S*; *x* $\in$ *S*⟧

$\Longrightarrow$ (*f has-field-derivative deriv f x*) (*at x within T*)
⟨*proof*⟩

lemma *complex-derivative-chain*:
f field-differentiable at x $\Longrightarrow$ *g field-differentiable at* (*f x*)
$\Longrightarrow$ *deriv* (*g o f*) *x* = *deriv g* (*f x*) ∗ *deriv f x*
⟨*proof*⟩

lemma *deriv-linear* [*simp*]: *deriv* (λ*w*. *c* ∗ *w*) = (λ*z*. *c*)
⟨*proof*⟩

lemma *deriv-ident* [*simp*]: *deriv* (λ*w*. *w*) = (λ*z*. *1*)
⟨*proof*⟩

lemma *deriv-id* [*simp*]: *deriv id* = (λ*z*. *1*)
⟨*proof*⟩

lemma *deriv-const* [*simp*]: *deriv* (λ*w*. *c*) = (λ*z*. *0*)
⟨*proof*⟩

lemma *deriv-add* [*simp*]:
⟦*f field-differentiable at z*; *g field-differentiable at z*⟧
$\Longrightarrow$ *deriv* (λ*w*. *f w* + *g w*) *z* = *deriv f z* + *deriv g z*
⟨*proof*⟩

lemma *deriv-diff* [*simp*]:
⟦*f field-differentiable at z*; *g field-differentiable at z*⟧
$\Longrightarrow$ *deriv* (λ*w*. *f w* − *g w*) *z* = *deriv f z* − *deriv g z*
⟨*proof*⟩

lemma *deriv-mult* [*simp*]:
⟦*f field-differentiable at z*; *g field-differentiable at z*⟧
$\Longrightarrow$ *deriv* (λ*w*. *f w* ∗ *g w*) *z* = *f z* ∗ *deriv g z* + *deriv f z* ∗ *g z*
⟨*proof*⟩

lemma *deriv-cmult* [*simp*]:
f field-differentiable at z $\Longrightarrow$ *deriv* (λ*w*. *c* ∗ *f w*) *z* = *c* ∗ *deriv f z*
⟨*proof*⟩

lemma *deriv-cmult-right* [*simp*]:
f field-differentiable at z $\Longrightarrow$ *deriv* (λ*w*. *f w* ∗ *c*) *z* = *deriv f z* ∗ *c*
⟨*proof*⟩

lemma *deriv-cdivide-right* [*simp*]:
f field-differentiable at z $\Longrightarrow$ *deriv* (λ*w*. *f w* / *c*) *z* = *deriv f z* / *c*
⟨*proof*⟩

lemma *complex-derivative-transform-within-open*:
⟦*f holomorphic-on s*; *g holomorphic-on s*; *open s*; *z* ∈ *s*; ⋀*w*. *w* ∈ *s* $\Longrightarrow$ *f w* = *g*

$w\rrbracket$
$\Longrightarrow$ *deriv f z = deriv g z*
⟨*proof*⟩

lemma *deriv-compose-linear*:
*f field-differentiable at (c * z)* $\Longrightarrow$ *deriv (λw. f (c * w)) z = c * deriv f (c * z)*
⟨*proof*⟩

lemma *nonzero-deriv-nonconstant*:
assumes *df*: *DERIV f ξ :> df* **and** *S*: *open S ξ ∈ S* **and** *df ≠ 0*
shows ¬ *f constant-on S*
⟨*proof*⟩

lemma *holomorphic-nonconstant*:
assumes *holf*: *f holomorphic-on S* **and** *open S ξ ∈ S deriv f ξ ≠ 0*
shows ¬ *f constant-on S*
⟨*proof*⟩

51.8 Analyticity on a set

definition *analytic-on* (**infixl** (*analytic'-on*) *50*)
where
f analytic-on s ≡ ∀ *x* ∈ *s*. ∃ *e*. *0 < e* ∧ *f holomorphic-on* (*ball x e*)

lemma *analytic-imp-holomorphic*: *f analytic-on s* $\Longrightarrow$ *f holomorphic-on s*
⟨*proof*⟩

lemma *analytic-on-open*: *open s* $\Longrightarrow$ *f analytic-on s* $\longleftrightarrow$ *f holomorphic-on s*
⟨*proof*⟩

lemma *analytic-on-imp-differentiable-at*:
f analytic-on s $\Longrightarrow$ *x* ∈ *s* $\Longrightarrow$ *f field-differentiable* (*at x*)
⟨*proof*⟩

lemma *analytic-on-subset*: *f analytic-on s* $\Longrightarrow$ *t* ⊆ *s* $\Longrightarrow$ *f analytic-on t*
⟨*proof*⟩

lemma *analytic-on-Un*: *f analytic-on* (*s* ∪ *t*) $\longleftrightarrow$ *f analytic-on s* ∧ *f analytic-on t*
⟨*proof*⟩

lemma *analytic-on-Union*: *f analytic-on* (⋃ *s*) $\longleftrightarrow$ (∀ *t* ∈ *s*. *f analytic-on t*)
⟨*proof*⟩

lemma *analytic-on-UN*: *f analytic-on* (⋃ *i*∈*I*. *s i*) $\longleftrightarrow$ (∀ *i*∈*I*. *f analytic-on* (*s i*))
⟨*proof*⟩

lemma *analytic-on-holomorphic*:
f analytic-on s $\longleftrightarrow$ (∃ *t*. *open t* ∧ *s* ⊆ *t* ∧ *f holomorphic-on t*)
(**is** *?lhs = ?rhs*)

⟨*proof*⟩

lemma *analytic-on-linear*: $(op * c)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-const*: $(\lambda z.\ c)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-ident*: $(\lambda x.\ x)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-id*: *id analytic-on* s
⟨*proof*⟩

lemma *analytic-on-compose*:
assumes f: f *analytic-on* s
and g: g *analytic-on* $(f\ `\ s)$
shows $(g\ o\ f)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-compose-gen*:
f *analytic-on* $s \Longrightarrow g$ *analytic-on* $t \Longrightarrow (\bigwedge z.\ z \in s \Longrightarrow f\ z \in t)$
$\Longrightarrow g\ o\ f$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-neg*:
f *analytic-on* $s \Longrightarrow (\lambda z.\ -(f\ z))$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-add*:
assumes f: f *analytic-on* s
and g: g *analytic-on* s
shows $(\lambda z.\ f\ z + g\ z)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-diff*:
assumes f: f *analytic-on* s
and g: g *analytic-on* s
shows $(\lambda z.\ f\ z - g\ z)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-mult*:
assumes f: f *analytic-on* s
and g: g *analytic-on* s
shows $(\lambda z.\ f\ z * g\ z)$ *analytic-on* s
⟨*proof*⟩

lemma *analytic-on-inverse*:
assumes f: f *analytic-on* s

and *nz*: $(\bigwedge z.\ z \in s \Longrightarrow f\ z \neq 0)$
shows $(\lambda z.\ inverse\ (f\ z))\ analytic\text{-}on\ s$
⟨*proof*⟩

lemma *analytic-on-divide*:
assumes *f*: $f\ analytic\text{-}on\ s$
and *g*: $g\ analytic\text{-}on\ s$
and *nz*: $(\bigwedge z.\ z \in s \Longrightarrow g\ z \neq 0)$
shows $(\lambda z.\ f\ z\ /\ g\ z)\ analytic\text{-}on\ s$
⟨*proof*⟩

lemma *analytic-on-power*:
$f\ analytic\text{-}on\ s \Longrightarrow (\lambda z.\ (f\ z)\ \hat{}\ n)\ analytic\text{-}on\ s$
⟨*proof*⟩

lemma *analytic-on-setsum*:
$(\bigwedge i.\ i \in I \Longrightarrow (f\ i)\ analytic\text{-}on\ s) \Longrightarrow (\lambda x.\ setsum\ (\lambda i.\ f\ i\ x)\ I)\ analytic\text{-}on\ s$
⟨*proof*⟩

lemma *deriv-left-inverse*:
assumes $f\ holomorphic\text{-}on\ S$ **and** $g\ holomorphic\text{-}on\ T$
and $open\ S$ **and** $open\ T$
and $f\ `\ S \subseteq T$
and [*simp*]: $\bigwedge z.\ z \in S \Longrightarrow g\ (f\ z) = z$
and $w \in S$
shows $deriv\ f\ w * deriv\ g\ (f\ w) = 1$
⟨*proof*⟩

51.9 analyticity at a point

lemma *analytic-at-ball*:
$f\ analytic\text{-}on\ \{z\} \longleftrightarrow (\exists e.\ 0<e \wedge f\ holomorphic\text{-}on\ ball\ z\ e)$
⟨*proof*⟩

lemma *analytic-at*:
$f\ analytic\text{-}on\ \{z\} \longleftrightarrow (\exists s.\ open\ s \wedge z \in s \wedge f\ holomorphic\text{-}on\ s)$
⟨*proof*⟩

lemma *analytic-on-analytic-at*:
$f\ analytic\text{-}on\ s \longleftrightarrow (\forall z \in s.\ f\ analytic\text{-}on\ \{z\})$
⟨*proof*⟩

lemma *analytic-at-two*:
$f\ analytic\text{-}on\ \{z\} \wedge g\ analytic\text{-}on\ \{z\} \longleftrightarrow$
$(\exists s.\ open\ s \wedge z \in s \wedge f\ holomorphic\text{-}on\ s \wedge g\ holomorphic\text{-}on\ s)$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

51.10 Combining theorems for derivative with "analytic at" hypotheses

lemma
assumes *f analytic-on* $\{z\}$ *g analytic-on* $\{z\}$
shows *complex-derivative-add-at*: *deriv* $(\lambda w.\ f\ w + g\ w)\ z = deriv\ f\ z + deriv\ g\ z$
and *complex-derivative-diff-at*: *deriv* $(\lambda w.\ f\ w - g\ w)\ z = deriv\ f\ z - deriv\ g\ z$
and *complex-derivative-mult-at*: *deriv* $(\lambda w.\ f\ w * g\ w)\ z =$
$f\ z * deriv\ g\ z + deriv\ f\ z * g\ z$
⟨*proof*⟩

lemma *deriv-cmult-at*:
f analytic-on $\{z\} \Longrightarrow$ *deriv* $(\lambda w.\ c * f\ w)\ z = c * deriv\ f\ z$
⟨*proof*⟩

lemma *deriv-cmult-right-at*:
f analytic-on $\{z\} \Longrightarrow$ *deriv* $(\lambda w.\ f\ w * c)\ z = deriv\ f\ z * c$
⟨*proof*⟩

51.11 Complex differentiation of sequences and series

lemma *has-complex-derivative-sequence*:
fixes s :: *complex set*
assumes *cvs*: *convex s*
and *df*: $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ has\text{-}field\text{-}derivative\ f'\ n\ x)\ (at\ x\ within\ s)$
and *conv*: $\bigwedge e.\ 0 < e \Longrightarrow \exists N.\ \forall n\ x.\ n \geq N \longrightarrow x \in s \longrightarrow norm\ (f'\ n\ x - g'\ x) \leq e$
and $\exists x\ l.\ x \in s \wedge ((\lambda n.\ f\ n\ x) \longrightarrow l)\ sequentially$
shows $\exists g.\ \forall x \in s.\ ((\lambda n.\ f\ n\ x) \longrightarrow g\ x)\ sequentially \wedge$
$(g\ has\text{-}field\text{-}derivative\ (g'\ x))\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *has-complex-derivative-series*:
fixes s :: *complex set*
assumes *cvs*: *convex s*
and *df*: $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ has\text{-}field\text{-}derivative\ f'\ n\ x)\ (at\ x\ within\ s)$
and *conv*: $\bigwedge e.\ 0 < e \Longrightarrow \exists N.\ \forall n\ x.\ n \geq N \longrightarrow x \in s \longrightarrow cmod\ ((\sum i{<}n.\ f'\ i\ x) - g'\ x) \leq e$
and $\exists x\ l.\ x \in s \wedge ((\lambda n.\ f\ n\ x)\ sums\ l)$
shows $\exists g.\ \forall x \in s.\ ((\lambda n.\ f\ n\ x)\ sums\ g\ x) \wedge ((g\ has\text{-}field\text{-}derivative\ g'\ x)\ (at\ x\ within\ s))$
⟨*proof*⟩

lemma *field-differentiable-series*:
fixes f :: *nat* ⇒ *complex* ⇒ *complex*
assumes *convex s open s*
assumes $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ has\text{-}field\text{-}derivative\ f'\ n\ x)\ (at\ x)$
assumes *uniformly-convergent-on s* $(\lambda n\ x.\ \sum i{<}n.\ f'\ i\ x)$

assumes *x0* ∈ *s summable* (λ*n*. *f n x0*) **and** *x*: *x* ∈ *s*
shows *summable* (λ*n*. *f n x*) **and** (λ*x*. ∑ *n*. *f n x*) *field-differentiable* (*at x*)
⟨*proof*⟩

lemma *field-differentiable-series'*:
fixes *f* :: *nat* ⇒ *complex* ⇒ *complex*
assumes *convex s open s*
assumes ⋀*n x*. *x* ∈ *s* ⟹ (*f n has-field-derivative f' n x*) (*at x*)
assumes *uniformly-convergent-on s* (λ*n x*. ∑ *i*<*n*. *f' i x*)
assumes *x0* ∈ *s summable* (λ*n*. *f n x0*)
shows (λ*x*. ∑ *n*. *f n x*) *field-differentiable* (*at x0*)
⟨*proof*⟩

51.12 Bound theorem

lemma *field-differentiable-bound*:
fixes *s* :: *complex set*
assumes *cvs*: *convex s*
and *df*: ⋀*z*. *z* ∈ *s* ⟹ (*f has-field-derivative f' z*) (*at z within s*)
and *dn*: ⋀*z*. *z* ∈ *s* ⟹ *norm* (*f' z*) ≤ *B*
and *x* ∈ *s* *y* ∈ *s*
shows *norm*(*f x* − *f y*) ≤ *B* ∗ *norm*(*x* − *y*)
⟨*proof*⟩

51.13 Inverse function theorem for complex derivatives

lemma *has-complex-derivative-inverse-basic*:
fixes *f* :: *complex* ⇒ *complex*
shows *DERIV f* (*g y*) :> *f'* ⟹
f' ≠ *0* ⟹
continuous (*at y*) *g* ⟹
open t ⟹
y ∈ *t* ⟹
(⋀*z*. *z* ∈ *t* ⟹ *f* (*g z*) = *z*)
⟹ *DERIV g y* :> *inverse* (*f'*)
⟨*proof*⟩

lemma *has-complex-derivative-inverse-strong*:
fixes *f* :: *complex* ⇒ *complex*
shows *DERIV f x* :> *f'* ⟹
f' ≠ *0* ⟹
open s ⟹
x ∈ *s* ⟹
continuous-on s f ⟹
(⋀*z*. *z* ∈ *s* ⟹ *g* (*f z*) = *z*)
⟹ *DERIV g* (*f x*) :> *inverse* (*f'*)
⟨*proof*⟩

lemma *has-complex-derivative-inverse-strong-x*:

fixes $f :: complex \Rightarrow complex$
shows $DERIV\ f\ (g\ y) :> f' \Longrightarrow$
$f' \neq 0 \Longrightarrow$
$open\ s \Longrightarrow$
$continuous\text{-}on\ s\ f \Longrightarrow$
$g\ y \in s \Longrightarrow f(g\ y) = y \Longrightarrow$
$(\bigwedge z.\ z \in s \Longrightarrow g\ (f\ z) = z)$
$\Longrightarrow DERIV\ g\ y :> inverse\ (f')$
⟨*proof*⟩

51.14 Taylor on Complex Numbers

lemma *setsum-Suc-reindex*:
fixes $f :: nat \Rightarrow 'a::ab\text{-}group\text{-}add$
shows $setsum\ f\ \{0..n\} = f\ 0 - f\ (Suc\ n) + setsum\ (\lambda i.\ f\ (Suc\ i))\ \{0..n\}$
⟨*proof*⟩

lemma *complex-taylor*:
assumes *s*: *convex s*
and *f*: $\bigwedge i\ x.\ x \in s \Longrightarrow i \leq n \Longrightarrow (f\ i\ has\text{-}field\text{-}derivative\ f\ (Suc\ i)\ x)\ (at\ x\ within\ s)$
and *B*: $\bigwedge x.\ x \in s \Longrightarrow cmod\ (f\ (Suc\ n)\ x) \leq B$
and *w*: $w \in s$
and *z*: $z \in s$
shows $cmod(f\ 0\ z - (\sum i{\leq}n.\ f\ i\ w * (z{-}w)\ \hat{}\ i\ /\ (fact\ i)))$
$\leq B * cmod(z - w)\ \hat{}(Suc\ n)\ /\ fact\ n$
⟨*proof*⟩

Something more like the traditional MVT for real components

lemma *complex-mvt-line*:
assumes $\bigwedge u.\ u \in closed\text{-}segment\ w\ z \Longrightarrow (f\ has\text{-}field\text{-}derivative\ f'(u))\ (at\ u)$
shows $\exists u.\ u \in closed\text{-}segment\ w\ z \wedge Re(f\ z) - Re(f\ w) = Re(f'(u) * (z - w))$
⟨*proof*⟩

lemma *complex-taylor-mvt*:
assumes $\bigwedge i\ x.\ [\![x \in closed\text{-}segment\ w\ z;\ i \leq n]\!] \Longrightarrow ((f\ i)\ has\text{-}field\text{-}derivative\ f\ (Suc\ i)\ x)\ (at\ x)$
shows $\exists u.\ u \in closed\text{-}segment\ w\ z \wedge$
$Re\ (f\ 0\ z) =$
$Re\ ((\sum i = 0..n.\ f\ i\ w * (z - w)\ \hat{}\ i\ /\ (fact\ i)) +$
$(f\ (Suc\ n)\ u * (z{-}u)\ \hat{}n\ /\ (fact\ n)) * (z - w))$
⟨*proof*⟩

51.15 Polynomal function extremal theorem, from HOL Light

lemma *polyfun-extremal-lemma*:
fixes $c :: nat \Rightarrow 'a::real\text{-}normed\text{-}div\text{-}algebra$
assumes $0 < e$
shows $\exists M.\ \forall z.\ M \leq norm(z) \longrightarrow norm\ (\sum i{\leq}n.\ c(i) * z\ \hat{}\ i) \leq e * norm(z)\ \hat{}\ (Suc\ n)$

⟨*proof*⟩

lemma *polyfun-extremal*:
fixes $c :: nat \Rightarrow 'a$::*real-normed-div-algebra*
assumes k: $c\ k \neq 0\ 1 \leq k$ **and** kn: $k \leq n$
shows *eventually* $(\lambda z.\ norm\ (\sum i \leq n.\ c(i) * z\hat{}i) \geq B)$ *at-infinity*
⟨*proof*⟩

end

52 Complex Transcendental Functions

By John Harrison et al. Ported from HOL Light by L C Paulson (2015)

theory *Complex-Transcendental*
imports
Complex-Analysis-Basics
Summation
begin

definition *moebius a b c d* $= (\lambda z.\ (a*z+b)\ /\ (c*z+d :: 'a :: field))$

lemma *moebius-inverse*:
assumes $a * d \neq b * c\ c * z + d \neq 0$
shows *moebius d* $(-b)$ $(-c)$ *a* (*moebius a b c d z*) $= z$
⟨*proof*⟩

lemma *moebius-inverse'*:
assumes $a * d \neq b * c\ c * z - a \neq 0$
shows *moebius a b c d* (*moebius d* $(-b)$ $(-c)$ *a z*) $= z$
⟨*proof*⟩

lemma *cmod-add-real-less*:
assumes $Im\ z \neq 0\ r \neq 0$
shows *cmod* $(z + r) <$ *cmod z* $+ |r|$
⟨*proof*⟩

lemma *cmod-diff-real-less*: $Im\ z \neq 0 \Longrightarrow x \neq 0 \Longrightarrow$ *cmod* $(z - x) <$ *cmod z* $+ |x|$
⟨*proof*⟩

lemma *cmod-square-less-1-plus*:
assumes $Im\ z = 0 \Longrightarrow |Re\ z| < 1$
shows $(cmod\ z)^2 < 1 +$ *cmod* $(1 - z^2)$
⟨*proof*⟩

52.1 The Exponential Function is Differentiable and Continuous

lemma *field-differentiable-within-exp*: *exp field-differentiable* (*at z within s*)
⟨*proof*⟩

lemma *continuous-within-exp*:
fixes *z*::'*a*::{*real-normed-field*,*banach*}
shows *continuous* (*at z within s*) *exp*
⟨*proof*⟩

lemma *holomorphic-on-exp* [*holomorphic-intros*]: *exp holomorphic-on s*
⟨*proof*⟩

52.2 Euler and de Moivre formulas.

The sine series times *i*

lemma *sin-ii-eq*: (λ*n*. (*ii* ∗ *sin-coeff n*) ∗ *z*^*n*) *sums* (*ii* ∗ *sin z*)
⟨*proof*⟩

theorem *exp-Euler*: *exp*(*ii* ∗ *z*) = *cos*(*z*) + *ii* ∗ *sin*(*z*)
⟨*proof*⟩

corollary *exp-minus-Euler*: *exp*(−(*ii* ∗ *z*)) = *cos*(*z*) − *ii* ∗ *sin*(*z*)
⟨*proof*⟩

lemma *sin-exp-eq*: *sin z* = (*exp*(*ii* ∗ *z*) − *exp*(−(*ii* ∗ *z*))) / (*2*∗*ii*)
⟨*proof*⟩

lemma *sin-exp-eq'*: *sin z* = *ii* ∗ (*exp*(−(*ii* ∗ *z*)) − *exp*(*ii* ∗ *z*)) / *2*
⟨*proof*⟩

lemma *cos-exp-eq*: *cos z* = (*exp*(*ii* ∗ *z*) + *exp*(−(*ii* ∗ *z*))) / *2*
⟨*proof*⟩

52.3 Relationships between real and complex trig functions

lemma *real-sin-eq* [*simp*]:
fixes *x*::*real*
shows *Re*(*sin*(*of-real x*)) = *sin x*
⟨*proof*⟩

lemma *real-cos-eq* [*simp*]:
fixes *x*::*real*
shows *Re*(*cos*(*of-real x*)) = *cos x*
⟨*proof*⟩

lemma *DeMoivre*: (*cos z* + *ii* ∗ *sin z*) ^ *n* = *cos*(*n* ∗ *z*) + *ii* ∗ *sin*(*n* ∗ *z*)
⟨*proof*⟩

lemma *exp-cnj*:
fixes *z*::*complex*
shows *cnj* (*exp z*) = *exp* (*cnj z*)
⟨*proof*⟩

lemma *cnj-sin*: *cnj*(*sin z*) = *sin*(*cnj z*)
⟨*proof*⟩

lemma *cnj-cos*: *cnj*(*cos z*) = *cos*(*cnj z*)
⟨*proof*⟩

lemma *field-differentiable-at-sin*: *sin field-differentiable at z*
⟨*proof*⟩

lemma *field-differentiable-within-sin*: *sin field-differentiable* (*at z within s*)
⟨*proof*⟩

lemma *field-differentiable-at-cos*: *cos field-differentiable at z*
⟨*proof*⟩

lemma *field-differentiable-within-cos*: *cos field-differentiable* (*at z within s*)
⟨*proof*⟩

lemma *holomorphic-on-sin*: *sin holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-on-cos*: *cos holomorphic-on s*
⟨*proof*⟩

52.4 Get a nice real/imaginary separation in Euler's formula.

lemma *Euler*: *exp*(*z*) = *of-real*(*exp*(*Re z*)) ∗
(*of-real*(*cos*(*Im z*)) + *ii* ∗ *of-real*(*sin*(*Im z*)))
⟨*proof*⟩

lemma *Re-sin*: *Re*(*sin z*) = *sin*(*Re z*) ∗ (*exp*(*Im z*) + *exp*(−(*Im z*))) / *2*
⟨*proof*⟩

lemma *Im-sin*: *Im*(*sin z*) = *cos*(*Re z*) ∗ (*exp*(*Im z*) − *exp*(−(*Im z*))) / *2*
⟨*proof*⟩

lemma *Re-cos*: *Re*(*cos z*) = *cos*(*Re z*) ∗ (*exp*(*Im z*) + *exp*(−(*Im z*))) / *2*
⟨*proof*⟩

lemma *Im-cos*: *Im*(*cos z*) = *sin*(*Re z*) ∗ (*exp*(−(*Im z*)) − *exp*(*Im z*)) / *2*
⟨*proof*⟩

lemma *Re-sin-pos*: *0* < *Re z* ⟹ *Re z* < *pi* ⟹ *Re* (*sin z*) > *0*

⟨*proof*⟩

lemma *Im-sin-nonneg*: $Re\ z = 0 \Longrightarrow 0 \leq Im\ z \Longrightarrow 0 \leq Im\ (sin\ z)$
⟨*proof*⟩

lemma *Im-sin-nonneg2*: $Re\ z = pi \Longrightarrow Im\ z \leq 0 \Longrightarrow 0 \leq Im\ (sin\ z)$
⟨*proof*⟩

52.5 More on the Polar Representation of Complex Numbers

lemma *exp-Complex*: $exp(Complex\ r\ t) = of\text{-}real(exp\ r) * Complex\ (cos\ t)\ (sin\ t)$
⟨*proof*⟩

lemma *exp-eq-1*: $exp\ z = 1 \longleftrightarrow Re(z) = 0 \wedge (\exists\, n{::}int.\ Im(z) = of\text{-}int\ (2 * n) * pi)$
⟨*proof*⟩

lemma *exp-eq*: $exp\ w = exp\ z \longleftrightarrow (\exists\, n{::}int.\ w = z + (of\text{-}int\ (2 * n) * pi) * ii)$
(**is** *?lhs = ?rhs*)
⟨*proof*⟩

lemma *exp-complex-eqI*: $|Im\ w - Im\ z| < 2{*}pi \Longrightarrow exp\ w = exp\ z \Longrightarrow w = z$
⟨*proof*⟩

lemma *exp-integer-2pi*:
assumes $n \in \mathbb{Z}$
shows $exp((2 * n * pi) * ii) = 1$
⟨*proof*⟩

lemma *sin-cos-eq-iff*: $sin\ y = sin\ x \wedge cos\ y = cos\ x \longleftrightarrow (\exists\, n{::}int.\ y = x + 2 * n * pi)$
⟨*proof*⟩

lemma *exp-i-ne-1*:
assumes $0 < x\ x < 2{*}pi$
shows $exp(\mathrm{i} * of\text{-}real\ x) \neq 1$
⟨*proof*⟩

lemma *sin-eq-0*:
fixes *z*::*complex*
shows $sin\ z = 0 \longleftrightarrow (\exists\, n{::}int.\ z = of\text{-}real(n * pi))$
⟨*proof*⟩

lemma *cos-eq-0*:
fixes *z*::*complex*
shows $cos\ z = 0 \longleftrightarrow (\exists\, n{::}int.\ z = of\text{-}real(n * pi) + of\text{-}real\ pi/2)$
⟨*proof*⟩

lemma *cos-eq-1*:
fixes *z*::*complex*
shows *cos z = 1 ⟷ (∃ n::int. z = of-real(2 * n * pi))*
⟨*proof*⟩

lemma *csin-eq-1*:
fixes *z*::*complex*
shows *sin z = 1 ⟷ (∃ n::int. z = of-real(2 * n * pi) + of-real pi/2)*
⟨*proof*⟩

lemma *csin-eq-minus1*:
fixes *z*::*complex*
shows *sin z = −1 ⟷ (∃ n::int. z = of-real(2 * n * pi) + 3/2*pi)*
(**is** *- = ?rhs*)
⟨*proof*⟩

lemma *ccos-eq-minus1*:
fixes *z*::*complex*
shows *cos z = −1 ⟷ (∃ n::int. z = of-real(2 * n * pi) + pi)*
⟨*proof*⟩

lemma *sin-eq-1*: *sin x = 1 ⟷ (∃ n::int. x = (2 * n + 1 / 2) * pi)*
(**is** *- = ?rhs*)
⟨*proof*⟩

lemma *sin-eq-minus1*: *sin x = −1 ⟷ (∃ n::int. x = (2*n + 3/2) * pi)* (**is** *- = ?rhs*)
⟨*proof*⟩

lemma *cos-eq-minus1*: *cos x = −1 ⟷ (∃ n::int. x = (2*n + 1) * pi)*
(**is** *- = ?rhs*)
⟨*proof*⟩

lemma *dist-exp-ii-1*: *norm(exp(ii * of-real t) − 1) = 2 * |sin(t / 2)|*
⟨*proof*⟩

lemma *sinh-complex*:
fixes *z* :: *complex*
shows *(exp z − inverse (exp z)) / 2 = −ii * sin(ii * z)*
⟨*proof*⟩

lemma *sin-ii-times*:
fixes *z* :: *complex*
shows *sin(ii * z) = ii * ((exp z − inverse (exp z)) / 2)*
⟨*proof*⟩

lemma *sinh-real*:
fixes *x* :: *real*
shows *of-real((exp x − inverse (exp x)) / 2) = −ii * sin(ii * of-real x)*

⟨*proof*⟩

lemma *cosh-complex*:
fixes *z* :: *complex*
shows (*exp z* + *inverse* (*exp z*)) / *2* = *cos*(*ii* ∗ *z*)
⟨*proof*⟩

lemma *cosh-real*:
fixes *x* :: *real*
shows *of-real*((*exp x* + *inverse* (*exp x*)) / *2*) = *cos*(*ii* ∗ *of-real x*)
⟨*proof*⟩

lemmas *cos-ii-times* = *cosh-complex* [*symmetric*]

lemma *norm-cos-squared*:
norm(*cos z*) ^ *2* = *cos*(*Re z*) ^ *2* + (*exp*(*Im z*) − *inverse*(*exp*(*Im z*))) ^ *2* / *4*
⟨*proof*⟩

lemma *norm-sin-squared*:
norm(*sin z*) ^ *2* = (*exp*(*2* ∗ *Im z*) + *inverse*(*exp*(*2* ∗ *Im z*)) − *2* ∗ *cos*(*2* ∗ *Re z*)) / *4*
⟨*proof*⟩

lemma *exp-uminus-Im*: *exp* (− *Im z*) ≤ *exp* (*cmod z*)
⟨*proof*⟩

lemma *norm-cos-le*:
fixes *z*::*complex*
shows *norm*(*cos z*) ≤ *exp*(*norm z*)
⟨*proof*⟩

lemma *norm-cos-plus1-le*:
fixes *z*::*complex*
shows *norm*(*1* + *cos z*) ≤ *2* ∗ *exp*(*norm z*)
⟨*proof*⟩

52.6 Taylor series for complex exponential, sine and cosine.

declare *power-Suc* [*simp del*]

lemma *Taylor-exp*:
norm(*exp z* − (∑ *k*≤*n*. *z* ^ *k* / (*fact k*))) ≤ *exp*|*Re z*| ∗ (*norm z*) ^ (*Suc n*) / (*fact n*)
⟨*proof*⟩

lemma
assumes *0* ≤ *u u* ≤ *1*
shows *cmod-sin-le-exp*: *cmod* (*sin* (*u* ∗$_R$ *z*)) ≤ *exp* |*Im z*|
and *cmod-cos-le-exp*: *cmod* (*cos* (*u* ∗$_R$ *z*)) ≤ *exp* |*Im z*|

⟨*proof*⟩

lemma *Taylor-sin*:
norm(*sin z* − (∑ *k*≤*n*. *complex-of-real* (*sin-coeff k*) ∗ *z* ^ *k*))
≤ *exp*|*Im z*| ∗ (*norm z*) ^ (*Suc n*) / (*fact n*)
⟨*proof*⟩

lemma *Taylor-cos*:
norm(*cos z* − (∑ *k*≤*n*. *complex-of-real* (*cos-coeff k*) ∗ *z* ^ *k*))
≤ *exp*|*Im z*| ∗ (*norm z*) ^ *Suc n* / (*fact n*)
⟨*proof*⟩

declare *power-Suc* [*simp*]

32-bit Approximation to e

lemma *e-approx-32*: |*exp*(*1*) − *5837465777* / *2147483648*| ≤ (*inverse*(*2* ^ *32*)::*real*)
⟨*proof*⟩

lemma *e-less-3*: *exp 1* < (*3*::*real*)
⟨*proof*⟩

lemma *ln3-gt-1*: *ln 3* > (*1*::*real*)
⟨*proof*⟩

52.7 The argument of a complex number

definition *Arg* :: *complex* ⇒ *real* **where**
Arg z ≡ *if z* = *0 then 0*
else THE t. *0* ≤ *t* ∧ *t* < *2*∗*pi* ∧
z = *of-real*(*norm z*) ∗ *exp*(*ii* ∗ *of-real t*)

lemma *Arg-0* [*simp*]: *Arg*(*0*) = *0*
⟨*proof*⟩

lemma *Arg-unique-lemma*:
assumes *z*: *z* = *of-real*(*norm z*) ∗ *exp*(*ii* ∗ *of-real t*)
and *z'*: *z* = *of-real*(*norm z*) ∗ *exp*(*ii* ∗ *of-real t'*)
and *t*: *0* ≤ *t t* < *2*∗*pi*
and *t'*: *0* ≤ *t' t'* < *2*∗*pi*
and *nz*: *z* ≠ *0*
shows *t'* = *t*
⟨*proof*⟩

lemma *Arg*: *0* ≤ *Arg z* & *Arg z* < *2*∗*pi* & *z* = *of-real*(*norm z*) ∗ *exp*(*ii* ∗ *of-real*(*Arg z*))
⟨*proof*⟩

corollary
shows *Arg-ge-0*: *0* ≤ *Arg z*

and *Arg-lt-2pi*: $Arg\ z < 2*pi$
and *Arg-eq*: $z = of\text{-}real(norm\ z) * exp(ii * of\text{-}real(Arg\ z))$
⟨*proof*⟩

lemma *complex-norm-eq-1-exp*: $norm\ z = 1 \longleftrightarrow (\exists\, t.\ z = exp(ii * of\text{-}real\ t))$
⟨*proof*⟩

lemma *Arg-unique*: $[\![of\text{-}real\ r * exp(ii * of\text{-}real\ a) = z;\ 0 < r;\ 0 \le a;\ a < 2*pi]\!] \Longrightarrow Arg\ z = a$
⟨*proof*⟩

lemma *Arg-minus*: $z \neq 0 \Longrightarrow Arg\ (-z) = (if\ Arg\ z < pi\ then\ Arg\ z + pi\ else\ Arg\ z - pi)$
⟨*proof*⟩

lemma *Arg-times-of-real* [*simp*]: $0 < r \Longrightarrow Arg\ (of\text{-}real\ r * z) = Arg\ z$
⟨*proof*⟩

lemma *Arg-times-of-real2* [*simp*]: $0 < r \Longrightarrow Arg\ (z * of\text{-}real\ r) = Arg\ z$
⟨*proof*⟩

lemma *Arg-divide-of-real* [*simp*]: $0 < r \Longrightarrow Arg\ (z\ /\ of\text{-}real\ r) = Arg\ z$
⟨*proof*⟩

lemma *Arg-le-pi*: $Arg\ z \le pi \longleftrightarrow 0 \le Im\ z$
⟨*proof*⟩

lemma *Arg-lt-pi*: $0 < Arg\ z \wedge Arg\ z < pi \longleftrightarrow 0 < Im\ z$
⟨*proof*⟩

lemma *Arg-eq-0*: $Arg\ z = 0 \longleftrightarrow z \in \mathbb{R} \wedge 0 \le Re\ z$
⟨*proof*⟩

corollary *Arg-gt-0*:
assumes $z \in \mathbb{R} \Longrightarrow Re\ z < 0$
shows $Arg\ z > 0$
⟨*proof*⟩

lemma *Arg-of-real*: $Arg(of\text{-}real\ x) = 0 \longleftrightarrow 0 \le x$
⟨*proof*⟩

lemma *Arg-eq-pi*: $Arg\ z = pi \longleftrightarrow z \in \mathbb{R} \wedge Re\ z < 0$
⟨*proof*⟩

lemma *Arg-eq-0-pi*: $Arg\ z = 0 \vee Arg\ z = pi \longleftrightarrow z \in \mathbb{R}$
⟨*proof*⟩

lemma *Arg-inverse*: $Arg(inverse\ z) = (if\ z \in \mathbb{R} \wedge 0 \le Re\ z\ then\ Arg\ z\ else\ 2*pi - Arg\ z)$

⟨*proof*⟩

lemma *Arg-eq-iff*:
assumes $w \neq 0$ $z \neq 0$
shows $Arg\ w = Arg\ z \longleftrightarrow (\exists x.\ 0 < x\ \&\ w = of\text{-}real\ x * z)$
⟨*proof*⟩

lemma *Arg-inverse-eq-0*: $Arg(inverse\ z) = 0 \longleftrightarrow Arg\ z = 0$
⟨*proof*⟩

lemma *Arg-divide*:
assumes $w \neq 0$ $z \neq 0$ $Arg\ w \leq Arg\ z$
shows $Arg(z\ /\ w) = Arg\ z - Arg\ w$
⟨*proof*⟩

lemma *Arg-le-div-sum*:
assumes $w \neq 0$ $z \neq 0$ $Arg\ w \leq Arg\ z$
shows $Arg\ z = Arg\ w + Arg(z\ /\ w)$
⟨*proof*⟩

lemma *Arg-le-div-sum-eq*:
assumes $w \neq 0$ $z \neq 0$
shows $Arg\ w \leq Arg\ z \longleftrightarrow Arg\ z = Arg\ w + Arg(z\ /\ w)$
⟨*proof*⟩

lemma *Arg-diff*:
assumes $w \neq 0$ $z \neq 0$
shows $Arg\ w - Arg\ z = (if\ Arg\ z \leq Arg\ w\ then\ Arg(w\ /\ z)\ else\ Arg(w/z) - 2*pi)$
⟨*proof*⟩

lemma *Arg-add*:
assumes $w \neq 0$ $z \neq 0$
shows $Arg\ w + Arg\ z = (if\ Arg\ w + Arg\ z < 2*pi\ then\ Arg(w * z)\ else\ Arg(w * z) + 2*pi)$
⟨*proof*⟩

lemma *Arg-times*:
assumes $w \neq 0$ $z \neq 0$
shows $Arg\ (w * z) = (if\ Arg\ w + Arg\ z < 2*pi\ then\ Arg\ w + Arg\ z\ else\ (Arg\ w + Arg\ z) - 2*pi)$
⟨*proof*⟩

lemma *Arg-cnj*: $Arg(cnj\ z) = (if\ z \in \mathbb{R} \wedge 0 \leq Re\ z\ then\ Arg\ z\ else\ 2*pi - Arg\ z)$
⟨*proof*⟩

lemma *Arg-real*: $z \in \mathbb{R} \Longrightarrow Arg\ z = (if\ 0 \leq Re\ z\ then\ 0\ else\ pi)$
⟨*proof*⟩

lemma *Arg-exp*: $0 \leq Im\ z \Longrightarrow Im\ z < 2{*}pi \Longrightarrow Arg(exp\ z) = Im\ z$
⟨*proof*⟩

lemma *complex-split-polar*:
obtains $r\ a$::*real* **where** $z = complex\text{-}of\text{-}real\ r * (cos\ a + \mathrm{i} * sin\ a)\ 0 \leq r\ 0 \leq a\ a < 2{*}pi$
⟨*proof*⟩

lemma *Re-Im-le-cmod*: $Im\ w * sin\ \varphi + Re\ w * cos\ \varphi \leq cmod\ w$
⟨*proof*⟩

52.8 Analytic properties of tangent function

lemma *cnj-tan*: $cnj(tan\ z) = tan(cnj\ z)$
⟨*proof*⟩

lemma *field-differentiable-at-tan*: $\sim(cos\ z = 0) \Longrightarrow tan\ field\text{-}differentiable\ at\ z$
⟨*proof*⟩

lemma *field-differentiable-within-tan*: $\sim(cos\ z = 0)$
$\Longrightarrow tan\ field\text{-}differentiable\ (at\ z\ within\ s)$
⟨*proof*⟩

lemma *continuous-within-tan*: $\sim(cos\ z = 0) \Longrightarrow continuous\ (at\ z\ within\ s)\ tan$
⟨*proof*⟩

lemma *continuous-on-tan* [*continuous-intros*]: $(\bigwedge z.\ z \in s \Longrightarrow \sim(cos\ z = 0)) \Longrightarrow continuous\text{-}on\ s\ tan$
⟨*proof*⟩

lemma *holomorphic-on-tan*: $(\bigwedge z.\ z \in s \Longrightarrow \sim(cos\ z = 0)) \Longrightarrow tan\ holomorphic\text{-}on\ s$
⟨*proof*⟩

52.9 Complex logarithms (the conventional principal value)

instantiation *complex* :: *ln*
begin

definition *ln-complex* :: *complex* ⇒ *complex*
where $ln\text{-}complex \equiv \lambda z.\ THE\ w.\ exp\ w = z\ \&\ -pi < Im(w)\ \&\ Im(w) \leq pi$

lemma
assumes $z \neq 0$
shows *exp-Ln* [*simp*]: $exp(ln\ z) = z$
and *mpi-less-Im-Ln*: $-pi < Im(ln\ z)$
and *Im-Ln-le-pi*: $Im(ln\ z) \leq pi$
⟨*proof*⟩

lemma *Ln-exp* [*simp*]:
assumes $-pi < Im(z)$ $Im(z) \leq pi$
shows $ln(exp\ z) = z$
⟨*proof*⟩

52.10 Relation to Real Logarithm

lemma *Ln-of-real*:
assumes $0 < z$
shows $ln(of\text{-}real\ z{::}complex) = of\text{-}real(ln\ z)$
⟨*proof*⟩

corollary *Ln-in-Reals* [*simp*]: $z \in \mathbb{R} \Longrightarrow Re\ z > 0 \Longrightarrow ln\ z \in \mathbb{R}$
⟨*proof*⟩

corollary *Im-Ln-of-real* [*simp*]: $r > 0 \Longrightarrow Im\ (ln\ (of\text{-}real\ r)) = 0$
⟨*proof*⟩

lemma *cmod-Ln-Reals* [*simp*]: $z \in \mathbb{R} \Longrightarrow 0 < Re\ z \Longrightarrow cmod\ (ln\ z) = norm\ (ln\ (Re\ z))$
⟨*proof*⟩

lemma *Ln-1*: $ln\ 1 = (0{::}complex)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

abbreviation *Ln* :: *complex* ⇒ *complex*
where $Ln \equiv ln$

lemma *Ln-eq-iff*: $w \neq 0 \Longrightarrow z \neq 0 \Longrightarrow (Ln\ w = Ln\ z \longleftrightarrow w = z)$
⟨*proof*⟩

lemma *Ln-unique*: $exp(z) = w \Longrightarrow -pi < Im(z) \Longrightarrow Im(z) \leq pi \Longrightarrow Ln\ w = z$
⟨*proof*⟩

lemma *Re-Ln* [*simp*]: $z \neq 0 \Longrightarrow Re(Ln\ z) = ln(norm\ z)$
⟨*proof*⟩

corollary *ln-cmod-le*:
assumes *z*: $z \neq 0$
shows $ln\ (cmod\ z) \leq cmod\ (Ln\ z)$
⟨*proof*⟩

proposition *exists-complex-root*:
fixes *z* :: *complex*

assumes $n \neq 0$ **obtains** w **where** $z = w \hat{} n$
⟨*proof*⟩

corollary *exists-complex-root-nonzero*:
fixes z::*complex*
assumes $z \neq 0$ $n \neq 0$
obtains w **where** $w \neq 0$ $z = w \hat{} n$
⟨*proof*⟩

52.11 The Unwinding Number and the Ln-product Formula

Note that in this special case the unwinding number is -1, 0 or 1.

definition *unwinding* :: *complex* ⇒ *complex* **where**
$unwinding(z) = (z - Ln(exp\ z))\ /\ (of\text{-}real(2{*}pi) * ii)$

lemma *unwinding-2pi*: $(2{*}pi) * ii * unwinding(z) = z - Ln(exp\ z)$
⟨*proof*⟩

lemma *Ln-times-unwinding*:
$w \neq 0 \Longrightarrow z \neq 0 \Longrightarrow Ln(w * z) = Ln(w) + Ln(z) - (2{*}pi) * ii * unwinding(Ln\ w + Ln\ z)$
⟨*proof*⟩

52.12 Derivative of Ln away from the branch cut

lemma
assumes $z \notin \mathbb{R}_{\leq 0}$
shows *has-field-derivative-Ln*: $(Ln\ has\text{-}field\text{-}derivative\ inverse(z))\ (at\ z)$
and *Im-Ln-less-pi*: $Im\ (Ln\ z) < pi$
⟨*proof*⟩

declare *has-field-derivative-Ln* [*derivative-intros*]
declare *has-field-derivative-Ln* [*THEN DERIV-chain2*, *derivative-intros*]

lemma *field-differentiable-at-Ln*: $z \notin \mathbb{R}_{\leq 0} \Longrightarrow Ln\ field\text{-}differentiable\ at\ z$
⟨*proof*⟩

lemma *field-differentiable-within-Ln*: $z \notin \mathbb{R}_{\leq 0}$
$\Longrightarrow Ln\ field\text{-}differentiable\ (at\ z\ within\ s)$
⟨*proof*⟩

lemma *continuous-at-Ln*: $z \notin \mathbb{R}_{\leq 0} \Longrightarrow continuous\ (at\ z)\ Ln$
⟨*proof*⟩

lemma *isCont-Ln'* [*simp*]:
$[\![isCont\ f\ z;\ f\ z \notin \mathbb{R}_{\leq 0}]\!] \Longrightarrow isCont\ (\lambda x.\ Ln\ (f\ x))\ z$
⟨*proof*⟩

lemma *continuous-within-Ln*: $z \notin \mathbb{R}_{\leq 0} \Longrightarrow continuous\ (at\ z\ within\ s)\ Ln$

⟨*proof*⟩

lemma *continuous-on-Ln* [*continuous-intros*]: $(\bigwedge z.\ z \in s \Longrightarrow z \notin \mathbb{R}_{\leq 0}) \Longrightarrow$ *continuous-on s Ln*
⟨*proof*⟩

lemma *holomorphic-on-Ln*: $(\bigwedge z.\ z \in s \Longrightarrow z \notin \mathbb{R}_{\leq 0}) \Longrightarrow$ *Ln holomorphic-on s*
⟨*proof*⟩

52.13 Quadrant-type results for Ln

lemma *cos-lt-zero-pi*: $pi/2 < x \Longrightarrow x < 3*pi/2 \Longrightarrow cos\ x < 0$
⟨*proof*⟩

lemma *Re-Ln-pos-lt*:
assumes $z \neq 0$
shows $|Im(Ln\ z)| < pi/2 \longleftrightarrow 0 < Re(z)$
⟨*proof*⟩

lemma *Re-Ln-pos-le*:
assumes $z \neq 0$
shows $|Im(Ln\ z)| \leq pi/2 \longleftrightarrow 0 \leq Re(z)$
⟨*proof*⟩

lemma *Im-Ln-pos-lt*:
assumes $z \neq 0$
shows $0 < Im(Ln\ z) \wedge Im(Ln\ z) < pi \longleftrightarrow 0 < Im(z)$
⟨*proof*⟩

lemma *Im-Ln-pos-le*:
assumes $z \neq 0$
shows $0 \leq Im(Ln\ z) \wedge Im(Ln\ z) \leq pi \longleftrightarrow 0 \leq Im(z)$
⟨*proof*⟩

lemma *Re-Ln-pos-lt-imp*: $0 < Re(z) \Longrightarrow |Im(Ln\ z)| < pi/2$
⟨*proof*⟩

lemma *Im-Ln-pos-lt-imp*: $0 < Im(z) \Longrightarrow 0 < Im(Ln\ z) \wedge Im(Ln\ z) < pi$
⟨*proof*⟩

A reference to the set of positive real numbers

lemma *Im-Ln-eq-0*: $z \neq 0 \Longrightarrow (Im(Ln\ z) = 0 \longleftrightarrow 0 < Re(z) \wedge Im(z) = 0)$
⟨*proof*⟩

lemma *Im-Ln-eq-pi*: $z \neq 0 \Longrightarrow (Im(Ln\ z) = pi \longleftrightarrow Re(z) < 0 \wedge Im(z) = 0)$
⟨*proof*⟩

52.14 More Properties of Ln

lemma *cnj-Ln*: $z \notin \mathbb{R}_{\leq 0} \Longrightarrow cnj(Ln\ z) = Ln(cnj\ z)$

⟨*proof*⟩

lemma *Ln-inverse*: $z \notin \mathbb{R}_{\leq 0} \Longrightarrow Ln(inverse\ z) = -(Ln\ z)$
⟨*proof*⟩

lemma *Ln-minus1* [*simp*]: $Ln(-1) = ii * pi$
⟨*proof*⟩

lemma *Ln-ii* [*simp*]: $Ln\ ii = ii * of\text{-}real\ pi/2$
⟨*proof*⟩

lemma *Ln-minus-ii* [*simp*]: $Ln(-ii) = -\ (ii * pi/2)$
⟨*proof*⟩

lemma *Ln-times*:
assumes $w \neq 0\ z \neq 0$
shows $Ln(w * z) =$
$(if\ Im(Ln\ w + Ln\ z) \leq -pi\ then$
$(Ln(w) + Ln(z)) + ii * of\text{-}real(2*pi)$
$else\ if\ Im(Ln\ w + Ln\ z) > pi\ then$
$(Ln(w) + Ln(z)) - ii * of\text{-}real(2*pi)$
$else\ Ln(w) + Ln(z))$
⟨*proof*⟩

corollary *Ln-times-simple*:
$[\![w \neq 0; z \neq 0; -pi < Im(Ln\ w) + Im(Ln\ z); Im(Ln\ w) + Im(Ln\ z) \leq pi]\!]$
$\Longrightarrow Ln(w * z) = Ln(w) + Ln(z)$
⟨*proof*⟩

corollary *Ln-times-of-real*:
$[\![r > 0; z \neq 0]\!] \Longrightarrow Ln(of\text{-}real\ r * z) = ln\ r + Ln(z)$
⟨*proof*⟩

corollary *Ln-divide-of-real*:
$[\![r > 0; z \neq 0]\!] \Longrightarrow Ln(z\ /\ of\text{-}real\ r) = Ln(z) - ln\ r$
⟨*proof*⟩

lemma *Ln-minus*:
assumes $z \neq 0$
shows $Ln(-z) = (if\ Im(z) \leq 0 \wedge \sim(Re(z) < 0 \wedge Im(z) = 0)$
$then\ Ln(z) + ii * pi$
$else\ Ln(z) - ii * pi)$ (**is** - = *?rhs*)
⟨*proof*⟩

lemma *Ln-inverse-if*:
assumes $z \neq 0$
shows $Ln\ (inverse\ z) = (if\ z \in \mathbb{R}_{\leq 0}\ then\ -(Ln\ z) + \mathrm{i} * 2 * complex\text{-}of\text{-}real\ pi\ else\ -(Ln\ z))$
⟨*proof*⟩

lemma *Ln-times-ii*:
assumes $z \neq 0$
shows $Ln(ii * z) = (\textit{if } 0 \leq Re(z) \mid Im(z) < 0$
$\textit{then } Ln(z) + ii * \textit{of-real } pi/2$
$\textit{else } Ln(z) - ii * \textit{of-real}(3 * pi/2))$
⟨*proof*⟩

lemma *Ln-of-nat*: $0 < n \Longrightarrow Ln\ (\textit{of-nat } n) = \textit{of-real}\ (ln\ (\textit{of-nat } n))$
⟨*proof*⟩

lemma *Ln-of-nat-over-of-nat*:
assumes $m > 0\ n > 0$
shows $Ln\ (\textit{of-nat } m\ /\ \textit{of-nat } n) = \textit{of-real}\ (ln\ (\textit{of-nat } m) - ln\ (\textit{of-nat } n))$
⟨*proof*⟩

52.15 Relation between Ln and Arg, and hence continuity of Arg

lemma *Arg-Ln*:
assumes $0 < Arg\ z$ **shows** $Arg\ z = Im(Ln(-z)) + pi$
⟨*proof*⟩

lemma *continuous-at-Arg*:
assumes $z \notin \mathbb{R}_{\geq 0}$
shows *continuous* (*at z*) *Arg*
⟨*proof*⟩

lemma *Ln-series*:
fixes z :: *complex*
assumes *norm z* < *1*
shows (λ*n*. (−*1*) ^*Suc n* / *of-nat n* ∗ *z*^*n*) *sums ln* (*1* + *z*) (**is** (λ*n*. *?f n* ∗ *z*^*n*) *sums* -)
⟨*proof*⟩

lemma *Ln-approx-linear*:
fixes z :: *complex*
assumes *norm z* < *1*
shows *norm* (*ln* (*1* + *z*) − *z*) ≤ *norm z*^*2* / (*1* − *norm z*)
⟨*proof*⟩

Relation between Arg and arctangent in upper halfplane

lemma *Arg-arctan-upperhalf*:
assumes *0* < *Im z*
shows *Arg z* = *pi*/*2* − *arctan*(*Re z* / *Im z*)
⟨*proof*⟩

lemma *Arg-eq-Im-Ln*:
assumes *0* ≤ *Im z 0* < *Re z*

shows *Arg z = Im (Ln z)*
⟨*proof*⟩

lemma *continuous-within-upperhalf-Arg*:
assumes *z ≠ 0*
shows *continuous (at z within {z. 0 ≤ Im z}) Arg*
⟨*proof*⟩

lemma *continuous-on-upperhalf-Arg*: *continuous-on ({z. 0 ≤ Im z} − {0}) Arg*
⟨*proof*⟩

lemma *open-Arg-less-Int*:
assumes *0 ≤ s t ≤ 2∗pi*
shows *open ({y. s < Arg y} ∩ {y. Arg y < t})*
⟨*proof*⟩

lemma *open-Arg-gt*: *open {z. t < Arg z}*
⟨*proof*⟩

lemma *closed-Arg-le*: *closed {z. Arg z ≤ t}*
⟨*proof*⟩

52.16 Complex Powers

lemma *powr-to-1* [*simp*]: *z powr 1 = (z::complex)*
⟨*proof*⟩

lemma *powr-nat*:
fixes *n::nat* **and** *z::complex* **shows** *z powr n = (if z = 0 then 0 else z^n)*
⟨*proof*⟩

lemma *powr-add-complex*:
fixes *w::complex* **shows** *w powr (z1 + z2) = w powr z1 ∗ w powr z2*
⟨*proof*⟩

lemma *powr-minus-complex*:
fixes *w::complex* **shows** *w powr (−z) = inverse(w powr z)*
⟨*proof*⟩

lemma *powr-diff-complex*:
fixes *w::complex* **shows** *w powr (z1 − z2) = w powr z1 / w powr z2*
⟨*proof*⟩

lemma *norm-powr-real*: *w ∈ ℝ ⟹ 0 < Re w ⟹ norm(w powr z) = exp(Re z ∗ ln(Re w))*
⟨*proof*⟩

lemma *cnj-powr*:
assumes *Im a = 0 ⟹ Re a ≥ 0*

shows *cnj (a powr b) = cnj a powr cnj b*
⟨*proof*⟩

lemma *powr-real-real*:
⟦*w ∈ ℝ*; *z ∈ ℝ*; *0 < Re w*⟧ ⟹ *w powr z = exp(Re z * ln(Re w))*
⟨*proof*⟩

lemma *powr-of-real*:
fixes *x*::*real* **and** *y*::*real*
shows *0 < x ⟹ of-real x powr (of-real y*::*complex) = of-real (x powr y)*
⟨*proof*⟩

lemma *norm-powr-real-mono*:
⟦*w ∈ ℝ*; *1 < Re w*⟧
⟹ *cmod(w powr z1) ≤ cmod(w powr z2) ⟷ Re z1 ≤ Re z2*
⟨*proof*⟩

lemma *powr-times-real*:
⟦*x ∈ ℝ*; *y ∈ ℝ*; *0 ≤ Re x*; *0 ≤ Re y*⟧
⟹ *(x * y) powr z = x powr z * y powr z*
⟨*proof*⟩

lemma *powr-neg-real-complex*:
shows *(− of-real x) powr a = (−1) powr (of-real (sgn x) * a) * of-real x powr (a :: complex)*
⟨*proof*⟩

lemma *has-field-derivative-powr*:
fixes *z :: complex*
shows *z ∉ ℝ≤0 ⟹ ((λz. z powr s) has-field-derivative (s * z powr (s − 1))) (at z)*
⟨*proof*⟩

declare *has-field-derivative-powr*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *has-field-derivative-powr-right*:
*w ≠ 0 ⟹ ((λz. w powr z) has-field-derivative Ln w * w powr z) (at z)*
⟨*proof*⟩

lemma *field-differentiable-powr-right*:
fixes *w*::*complex*
shows
w ≠ 0 ⟹ (λz. w powr z) field-differentiable (at z)
⟨*proof*⟩

lemma *holomorphic-on-powr-right*:
f holomorphic-on s ⟹ w ≠ 0 ⟹ (λz. w powr (f z)) holomorphic-on s
⟨*proof*⟩

lemma *norm-powr-real-powr*:
$w \in \mathbb{R} \Longrightarrow 0 < Re\ w \Longrightarrow norm(w\ powr\ z) = Re\ w\ powr\ Re\ z$
⟨*proof*⟩

52.17 Some Limits involving Logarithms

lemma *lim-Ln-over-power*:
fixes *s*::*complex*
assumes $0 < Re\ s$
shows $((\lambda n.\ Ln\ n\ /\ (n\ powr\ s)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-Ln-over-n*: $((\lambda n.\ Ln(of\text{-}nat\ n)\ /\ of\text{-}nat\ n) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *Ln-Reals-eq*: $x \in \mathbb{R} \Longrightarrow Re\ x > 0 \Longrightarrow Ln\ x = of\text{-}real\ (ln\ (Re\ x))$
⟨*proof*⟩

lemma *powr-Reals-eq*: $x \in \mathbb{R} \Longrightarrow Re\ x > 0 \Longrightarrow x\ powr\ complex\text{-}of\text{-}real\ y = of\text{-}real\ (x\ powr\ y)$
⟨*proof*⟩

lemma *lim-ln-over-power*:
fixes *s* :: *real*
assumes $0 < s$
shows $((\lambda n.\ ln\ n\ /\ (n\ powr\ s)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-ln-over-n*: $((\lambda n.\ ln(real\text{-}of\text{-}nat\ n)\ /\ of\text{-}nat\ n) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-1-over-complex-power*:
assumes $0 < Re\ s$
shows $((\lambda n.\ 1\ /\ (of\text{-}nat\ n\ powr\ s)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-1-over-real-power*:
fixes *s* :: *real*
assumes $0 < s$
shows $((\lambda n.\ 1\ /\ (of\text{-}nat\ n\ powr\ s)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-1-over-Ln*: $((\lambda n.\ 1\ /\ Ln(of\text{-}nat\ n)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

lemma *lim-1-over-ln*: $((\lambda n.\ 1\ /\ ln(real\text{-}of\text{-}nat\ n)) \longrightarrow 0)\ sequentially$
⟨*proof*⟩

52.18 Relation between Square Root and exp/ln, hence its derivative

lemma *csqrt-exp-Ln*:
assumes $z \neq 0$
shows $csqrt\ z = exp(Ln(z)\ /\ 2)$
⟨*proof*⟩

lemma *csqrt-inverse*:
assumes $z \notin \mathbb{R}_{\leq 0}$
shows $csqrt\ (inverse\ z) = inverse\ (csqrt\ z)$
⟨*proof*⟩

lemma *cnj-csqrt*:
assumes $z \notin \mathbb{R}_{\leq 0}$
shows $cnj(csqrt\ z) = csqrt(cnj\ z)$
⟨*proof*⟩

lemma *has-field-derivative-csqrt*:
assumes $z \notin \mathbb{R}_{\leq 0}$
shows $(csqrt\ has\text{-}field\text{-}derivative\ inverse(2 * csqrt\ z))\ (at\ z)$
⟨*proof*⟩

lemma *field-differentiable-at-csqrt*:
$z \notin \mathbb{R}_{\leq 0} \Longrightarrow csqrt\ field\text{-}differentiable\ at\ z$
⟨*proof*⟩

lemma *field-differentiable-within-csqrt*:
$z \notin \mathbb{R}_{\leq 0} \Longrightarrow csqrt\ field\text{-}differentiable\ (at\ z\ within\ s)$
⟨*proof*⟩

lemma *continuous-at-csqrt*:
$z \notin \mathbb{R}_{\leq 0} \Longrightarrow continuous\ (at\ z)\ csqrt$
⟨*proof*⟩

corollary *isCont-csqrt'* [*simp*]:
$[\![isCont\ f\ z;\ f\ z \notin \mathbb{R}_{\leq 0}]\!] \Longrightarrow isCont\ (\lambda x.\ csqrt\ (f\ x))\ z$
⟨*proof*⟩

lemma *continuous-within-csqrt*:
$z \notin \mathbb{R}_{\leq 0} \Longrightarrow continuous\ (at\ z\ within\ s)\ csqrt$
⟨*proof*⟩

lemma *continuous-on-csqrt* [*continuous-intros*]:
$(\bigwedge z.\ z \in s \Longrightarrow z \notin \mathbb{R}_{\leq 0}) \Longrightarrow continuous\text{-}on\ s\ csqrt$
⟨*proof*⟩

lemma *holomorphic-on-csqrt*:
$(\bigwedge z.\ z \in s \Longrightarrow z \notin \mathbb{R}_{\leq 0}) \Longrightarrow csqrt\ holomorphic\text{-}on\ s$
⟨*proof*⟩

lemma *continuous-within-closed-nontrivial*:
closed s $\Longrightarrow$ *a* $\notin$ *s* ==> *continuous* (*at a within s*) *f*
⟨*proof*⟩

lemma *continuous-within-csqrt-posreal*:
continuous (*at z within* ($\mathbb{R} \cap \{w.\ 0 \leq Re(w)\}$)) *csqrt*
⟨*proof*⟩

52.19 Complex arctangent

The branch cut gives standard bounds in the real case.

definition *Arctan* :: *complex* ⇒ *complex* **where**
Arctan ≡ λ*z*. (i/*2*) ∗ *Ln*((*1* − i∗*z*) / (*1* + i∗*z*))

lemma *Arctan-def-moebius*: *Arctan z* = i/*2* ∗ *Ln* (*moebius* (−i) *1* i *1 z*)
⟨*proof*⟩

lemma *Ln-conv-Arctan*:
assumes *z* ≠ −*1*
shows *Ln z* = −*2*∗i ∗ *Arctan* (*moebius 1* (− *1*) (− i) (− i) *z*)
⟨*proof*⟩

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

lemma *Im-complex-div-lemma*: *Im*((*1* − i∗*z*) / (*1* + i∗*z*)) = *0* ⟷ *Re z* = *0*
⟨*proof*⟩

lemma *Re-complex-div-lemma*: *0* < *Re*((*1* − i∗*z*) / (*1* + i∗*z*)) ⟷ *norm z* < *1*
⟨*proof*⟩

lemma *tan-Arctan*:
assumes $z^2 \neq -1$
shows [*simp*]:*tan*(*Arctan z*) = *z*
⟨*proof*⟩

lemma *Arctan-tan* [*simp*]:
assumes |*Re z*| < *pi*/*2*
shows *Arctan*(*tan z*) = *z*
⟨*proof*⟩

lemma
assumes *Re z* = *0* $\Longrightarrow$ |*Im z*| < *1*
shows *Re-Arctan-bounds*: |*Re*(*Arctan z*)| < *pi*/*2*
and *has-field-derivative-Arctan*: (*Arctan has-field-derivative inverse*($1 + z^2$))
(*at z*)
⟨*proof*⟩

lemma *field-differentiable-at-Arctan*: $(Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow Arctan$ *field-differentiable at* z
⟨*proof*⟩

lemma *field-differentiable-within-Arctan*:
$(Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow Arctan$ *field-differentiable* $(at\ z\ within\ s)$
⟨*proof*⟩

declare *has-field-derivative-Arctan* [*derivative-intros*]
declare *has-field-derivative-Arctan* [*THEN DERIV-chain2*, *derivative-intros*]

lemma *continuous-at-Arctan*:
$(Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow continuous\ (at\ z)\ Arctan$
⟨*proof*⟩

lemma *continuous-within-Arctan*:
$(Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow continuous\ (at\ z\ within\ s)\ Arctan$
⟨*proof*⟩

lemma *continuous-on-Arctan* [*continuous-intros*]:
$(\bigwedge z.\ z \in s \Longrightarrow Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow continuous\text{-}on\ s\ Arctan$
⟨*proof*⟩

lemma *holomorphic-on-Arctan*:
$(\bigwedge z.\ z \in s \Longrightarrow Re\ z = 0 \Longrightarrow |Im\ z| < 1) \Longrightarrow Arctan\ holomorphic\text{-}on\ s$
⟨*proof*⟩

lemma *Arctan-series*:
assumes z: $norm\ (z :: complex) < 1$
defines $g \equiv \lambda n.\ if\ odd\ n\ then\ -\mathrm{i}*\mathrm{i}\ \hat{}\ n\ /\ n\ else\ 0$
defines $h \equiv \lambda z\ n.\ (-1)\ \hat{}\ n\ /\ of\text{-}nat\ (2*n+1) * (z::complex)\ \hat{}\ (2*n+1)$
shows $(\lambda n.\ g\ n * z\ \hat{}\ n)\ sums\ Arctan\ z$
and $h\ z\ sums\ Arctan\ z$
⟨*proof*⟩

A quickly-converging series for the logarithm, based on the arctangent.

lemma *ln-series-quadratic*:
assumes x: $x > (0::real)$
shows $(\lambda n.\ (2*((x - 1)\ /\ (x + 1))\ \hat{}\ (2*n+1)\ /\ of\text{-}nat\ (2*n+1)))\ sums\ ln\ x$
⟨*proof*⟩

52.20 Real arctangent

lemma *norm-exp-ii-times* [*simp*]: $norm\ (exp(\mathrm{i} * of\text{-}real\ y)) = 1$
⟨*proof*⟩

lemma *norm-exp-imaginary*: $norm(exp\ z) = 1 \Longrightarrow Re\ z = 0$
⟨*proof*⟩

lemma *Im-Arctan-of-real* [*simp*]: *Im* (*Arctan* (*of-real x*)) = *0*
⟨*proof*⟩

lemma *arctan-eq-Re-Arctan*: *arctan x* = *Re* (*Arctan* (*of-real x*))
⟨*proof*⟩

lemma *Arctan-of-real*: *Arctan* (*of-real x*) = *of-real* (*arctan x*)
⟨*proof*⟩

lemma *Arctan-in-Reals* [*simp*]: $z \in \mathbb{R} \Longrightarrow$ *Arctan z* $\in \mathbb{R}$
⟨*proof*⟩

declare *arctan-one* [*simp*]

lemma *arctan-less-pi4-pos*: *x* < *1* ⟹ *arctan x* < *pi*/*4*
⟨*proof*⟩

lemma *arctan-less-pi4-neg*: −*1* < *x* ⟹ −(*pi*/*4*) < *arctan x*
⟨*proof*⟩

lemma *arctan-less-pi4*: |*x*| < *1* ⟹ |*arctan x*| < *pi*/*4*
⟨*proof*⟩

lemma *arctan-le-pi4*: |*x*| ≤ *1* ⟹ |*arctan x*| ≤ *pi*/*4*
⟨*proof*⟩

lemma *abs-arctan*: |*arctan x*| = *arctan* |*x*|
⟨*proof*⟩

lemma *arctan-add-raw*:
assumes |*arctan x* + *arctan y*| < *pi*/*2*
shows *arctan x* + *arctan y* = *arctan*((*x* + *y*) / (*1* − *x* ∗ *y*))
⟨*proof*⟩

lemma *arctan-inverse*:
assumes *0* < *x*
shows *arctan*(*inverse x*) = *pi*/*2* − *arctan x*
⟨*proof*⟩

lemma *arctan-add-small*:
assumes |*x* ∗ *y*| < *1*
shows (*arctan x* + *arctan y* = *arctan*((*x* + *y*) / (*1* − *x* ∗ *y*)))
⟨*proof*⟩

lemma *abs-arctan-le*:
fixes *x*::*real* **shows** |*arctan x*| ≤ |*x*|
⟨*proof*⟩

lemma *arctan-le-self*: *0* ≤ *x* ⟹ *arctan x* ≤ *x*

⟨*proof*⟩

lemma *abs-tan-ge*: $|x| < pi/2 \Longrightarrow |x| \leq |tan\ x|$
⟨*proof*⟩

52.21 Inverse Sine

definition *Arcsin* :: *complex* ⇒ *complex* **where**
Arcsin ≡ λ*z*. −i ∗ *Ln*(i ∗ *z* + *csqrt*(1 − z^2))

lemma *Arcsin-body-lemma*: i ∗ *z* + *csqrt*(1 − z^2) ≠ 0
⟨*proof*⟩

lemma *Arcsin-range-lemma*: |*Re z*| < 1 ⟹ 0 < *Re*(i ∗ *z* + *csqrt*(1 − z^2))
⟨*proof*⟩

lemma *Re-Arcsin*: *Re*(*Arcsin z*) = *Im* (*Ln* (i ∗ *z* + *csqrt*(1 − z^2)))
⟨*proof*⟩

lemma *Im-Arcsin*: *Im*(*Arcsin z*) = − *ln* (*cmod* (i ∗ *z* + *csqrt* (1 − z^2)))
⟨*proof*⟩

lemma *one-minus-z2-notin-nonpos-Reals*:
assumes (*Im z* = 0 ⟹ |*Re z*| < 1)
shows 1 − z^2 ∉ $\mathbb{R}_{\leq 0}$
⟨*proof*⟩

lemma *isCont-Arcsin-lemma*:
assumes *le0*: *Re* (i ∗ *z* + *csqrt* (1 − z^2)) ≤ 0 **and** (*Im z* = 0 ⟹ |*Re z*| < 1)
shows *False*
⟨*proof*⟩

lemma *isCont-Arcsin*:
assumes (*Im z* = 0 ⟹ |*Re z*| < 1)
shows *isCont Arcsin z*
⟨*proof*⟩

lemma *isCont-Arcsin'* [*simp*]:
shows *isCont f z* ⟹ (*Im* (*f z*) = 0 ⟹ |*Re* (*f z*)| < 1) ⟹ *isCont* (λ*x*. *Arcsin* (*f x*)) *z*
⟨*proof*⟩

lemma *sin-Arcsin* [*simp*]: *sin*(*Arcsin z*) = *z*
⟨*proof*⟩

lemma *Re-eq-pihalf-lemma*:
|*Re z*| = *pi*/2 ⟹ *Im z* = 0 ⟹
Re ((*exp* (i∗*z*) + *inverse* (*exp* (i∗*z*))) / 2) = 0 ∧ 0 ≤ *Im* ((*exp* (i∗*z*) + *inverse* (*exp* (i∗*z*))) / 2)

⟨*proof*⟩

lemma *Re-less-pihalf-lemma*:
assumes $|Re\ z| < pi\ /\ 2$
shows $0 < Re\ ((exp\ (\mathrm{i}*z) + inverse\ (exp\ (\mathrm{i}*z)))\ /\ 2)$
⟨*proof*⟩

lemma *Arcsin-sin*:
assumes $|Re\ z| < pi/2 \vee (|Re\ z| = pi/2 \wedge Im\ z = 0)$
shows $Arcsin(sin\ z) = z$
⟨*proof*⟩

lemma *Arcsin-unique*:
$[\![sin\ z = w; |Re\ z| < pi/2 \vee (|Re\ z| = pi/2 \wedge Im\ z = 0)]\!] \Longrightarrow Arcsin\ w = z$
⟨*proof*⟩

lemma *Arcsin-0* [*simp*]: $Arcsin\ 0 = 0$
⟨*proof*⟩

lemma *Arcsin-1* [*simp*]: $Arcsin\ 1 = pi/2$
⟨*proof*⟩

lemma *Arcsin-minus-1* [*simp*]: $Arcsin(-1) = -\ (pi/2)$
⟨*proof*⟩

lemma *has-field-derivative-Arcsin*:
assumes $(Im\ z = 0 \Longrightarrow |Re\ z| < 1)$
shows $(Arcsin\ has\text{-}field\text{-}derivative\ inverse(cos(Arcsin\ z)))\ (at\ z)$
⟨*proof*⟩

declare *has-field-derivative-Arcsin* [*derivative-intros*]
declare *has-field-derivative-Arcsin* [*THEN DERIV-chain2*, *derivative-intros*]

lemma *field-differentiable-at-Arcsin*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow Arcsin\ field\text{-}differentiable\ at\ z$
⟨*proof*⟩

lemma *field-differentiable-within-Arcsin*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow Arcsin\ field\text{-}differentiable\ (at\ z\ within\ s)$
⟨*proof*⟩

lemma *continuous-within-Arcsin*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow continuous\ (at\ z\ within\ s)\ Arcsin$
⟨*proof*⟩

lemma *continuous-on-Arcsin* [*continuous-intros*]:
$(\bigwedge z.\ z \in s \Longrightarrow Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow continuous\text{-}on\ s\ Arcsin$
⟨*proof*⟩

lemma *holomorphic-on-Arcsin*: $(\bigwedge z.\ z \in s \Longrightarrow Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow$ *Arcsin holomorphic-on s*
⟨*proof*⟩

52.22 Inverse Cosine

definition *Arccos* :: *complex* ⇒ *complex* **where**
Arccos ≡ $\lambda z.\ -\mathrm{i} * Ln(z + \mathrm{i} * csqrt(1 - z^2))$

lemma *Arccos-range-lemma*: $|Re\ z| < 1 \Longrightarrow 0 < Im(z + \mathrm{i} * csqrt(1 - z^2))$
⟨*proof*⟩

lemma *Arccos-body-lemma*: $z + \mathrm{i} * csqrt(1 - z^2) \neq 0$
⟨*proof*⟩

lemma *Re-Arccos*: $Re(Arccos\ z) = Im\ (Ln\ (z + \mathrm{i} * csqrt(1 - z^2)))$
⟨*proof*⟩

lemma *Im-Arccos*: $Im(Arccos\ z) = -\ ln\ (cmod\ (z + \mathrm{i} * csqrt\ (1 - z^2)))$
⟨*proof*⟩

A very tricky argument to find!

lemma *isCont-Arccos-lemma*:
assumes *eq0*: $Im\ (z + \mathrm{i} * csqrt\ (1 - z^2)) = 0$ **and** $(Im\ z = 0 \Longrightarrow |Re\ z| < 1)$
shows *False*
⟨*proof*⟩

lemma *isCont-Arccos*:
assumes $(Im\ z = 0 \Longrightarrow |Re\ z| < 1)$
shows *isCont Arccos z*
⟨*proof*⟩

lemma *isCont-Arccos'* [*simp*]:
shows $isCont\ f\ z \Longrightarrow (Im\ (f\ z) = 0 \Longrightarrow |Re\ (f\ z)| < 1) \Longrightarrow isCont\ (\lambda x.\ Arccos\ (f\ x))\ z$
⟨*proof*⟩

lemma *cos-Arccos* [*simp*]: $cos(Arccos\ z) = z$
⟨*proof*⟩

lemma *Arccos-cos*:
assumes $0 < Re\ z\ \&\ Re\ z < pi\ \vee$
$Re\ z = 0\ \&\ 0 \leq Im\ z\ \vee$
$Re\ z = pi\ \&\ Im\ z \leq 0$
shows $Arccos(cos\ z) = z$
⟨*proof*⟩

lemma *Arccos-unique*:
⟦$cos\ z = w$;

$0 < Re\ z \wedge Re\ z < pi \vee$
$Re\ z = 0 \wedge 0 \leq Im\ z \vee$
$Re\ z = pi \wedge Im\ z \leq 0 \rrbracket \Longrightarrow Arccos\ w = z$
⟨*proof*⟩

lemma *Arccos-0* [*simp*]: $Arccos\ 0 = pi/2$
⟨*proof*⟩

lemma *Arccos-1* [*simp*]: $Arccos\ 1 = 0$
⟨*proof*⟩

lemma *Arccos-minus1*: $Arccos(-1) = pi$
⟨*proof*⟩

lemma *has-field-derivative-Arccos*:
assumes $(Im\ z = 0 \Longrightarrow |Re\ z| < 1)$
shows $(Arccos\ has\text{-}field\text{-}derivative - inverse(sin(Arccos\ z)))\ (at\ z)$
⟨*proof*⟩

declare *has-field-derivative-Arcsin* [*derivative-intros*]
declare *has-field-derivative-Arcsin* [*THEN DERIV-chain2*, *derivative-intros*]

lemma *field-differentiable-at-Arccos*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow Arccos\ field\text{-}differentiable\ at\ z$
⟨*proof*⟩

lemma *field-differentiable-within-Arccos*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow Arccos\ field\text{-}differentiable\ (at\ z\ within\ s)$
⟨*proof*⟩

lemma *continuous-within-Arccos*:
$(Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow continuous\ (at\ z\ within\ s)\ Arccos$
⟨*proof*⟩

lemma *continuous-on-Arccos* [*continuous-intros*]:
$(\bigwedge z.\ z \in s \Longrightarrow Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow continuous\text{-}on\ s\ Arccos$
⟨*proof*⟩

lemma *holomorphic-on-Arccos*: $(\bigwedge z.\ z \in s \Longrightarrow Im\ z = 0 \Longrightarrow |Re\ z| < 1) \Longrightarrow$
$Arccos\ holomorphic\text{-}on\ s$
⟨*proof*⟩

52.23 Upper and Lower Bounds for Inverse Sine and Cosine

lemma *Arcsin-bounds*: $|Re\ z| < 1 \Longrightarrow |Re(Arcsin\ z)| < pi/2$
⟨*proof*⟩

lemma *Arccos-bounds*: $|Re\ z| < 1 \Longrightarrow 0 < Re(Arccos\ z) \wedge Re(Arccos\ z) < pi$
⟨*proof*⟩

lemma *Re-Arccos-bounds*: $-pi < Re(Arccos\ z) \wedge Re(Arccos\ z) \leq pi$
⟨*proof*⟩

lemma *Re-Arccos-bound*: $|Re(Arccos\ z)| \leq pi$
⟨*proof*⟩

lemma *Re-Arcsin-bounds*: $-pi < Re(Arcsin\ z)\ \&\ Re(Arcsin\ z) \leq pi$
⟨*proof*⟩

lemma *Re-Arcsin-bound*: $|Re(Arcsin\ z)| \leq pi$
⟨*proof*⟩

52.24 Interrelations between Arcsin and Arccos

lemma *cos-Arcsin-nonzero*:
assumes $z^2 \neq 1$ **shows** $cos(Arcsin\ z) \neq 0$
⟨*proof*⟩

lemma *sin-Arccos-nonzero*:
assumes $z^2 \neq 1$ **shows** $sin(Arccos\ z) \neq 0$
⟨*proof*⟩

lemma *cos-sin-csqrt*:
assumes $0 < cos(Re\ z) \ \vee \ cos(Re\ z) = 0 \wedge Im\ z * sin(Re\ z) \leq 0$
shows $cos\ z = csqrt(1 - (sin\ z)^2)$
⟨*proof*⟩

lemma *sin-cos-csqrt*:
assumes $0 < sin(Re\ z) \ \vee \ sin(Re\ z) = 0 \wedge 0 \leq Im\ z * cos(Re\ z)$
shows $sin\ z = csqrt(1 - (cos\ z)^2)$
⟨*proof*⟩

lemma *Arcsin-Arccos-csqrt-pos*:
$(0 < Re\ z \mid Re\ z = 0\ \&\ 0 \leq Im\ z) \Longrightarrow Arcsin\ z = Arccos(csqrt(1 - z^2))$
⟨*proof*⟩

lemma *Arccos-Arcsin-csqrt-pos*:
$(0 < Re\ z \mid Re\ z = 0\ \&\ 0 \leq Im\ z) \Longrightarrow Arccos\ z = Arcsin(csqrt(1 - z^2))$
⟨*proof*⟩

lemma *sin-Arccos*:
$0 < Re\ z \mid Re\ z = 0\ \&\ 0 \leq Im\ z \Longrightarrow sin(Arccos\ z) = csqrt(1 - z^2)$
⟨*proof*⟩

lemma *cos-Arcsin*:
$0 < Re\ z \mid Re\ z = 0\ \&\ 0 \leq Im\ z \Longrightarrow cos(Arcsin\ z) = csqrt(1 - z^2)$
⟨*proof*⟩

52.25 Relationship with Arcsin on the Real Numbers

lemma *Im-Arcsin-of-real*:
assumes $|x| \leq 1$
shows $Im\ (Arcsin\ (of\text{-}real\ x)) = 0$
⟨*proof*⟩

corollary *Arcsin-in-Reals* [*simp*]: $z \in \mathbb{R} \Longrightarrow |Re\ z| \leq 1 \Longrightarrow Arcsin\ z \in \mathbb{R}$
⟨*proof*⟩

lemma *arcsin-eq-Re-Arcsin*:
assumes $|x| \leq 1$
shows $arcsin\ x = Re\ (Arcsin\ (of\text{-}real\ x))$
⟨*proof*⟩

lemma *of-real-arcsin*: $|x| \leq 1 \Longrightarrow of\text{-}real(arcsin\ x) = Arcsin(of\text{-}real\ x)$
⟨*proof*⟩

52.26 Relationship with Arccos on the Real Numbers

lemma *Im-Arccos-of-real*:
assumes $|x| \leq 1$
shows $Im\ (Arccos\ (of\text{-}real\ x)) = 0$
⟨*proof*⟩

corollary *Arccos-in-Reals* [*simp*]: $z \in \mathbb{R} \Longrightarrow |Re\ z| \leq 1 \Longrightarrow Arccos\ z \in \mathbb{R}$
⟨*proof*⟩

lemma *arccos-eq-Re-Arccos*:
assumes $|x| \leq 1$
shows $arccos\ x = Re\ (Arccos\ (of\text{-}real\ x))$
⟨*proof*⟩

lemma *of-real-arccos*: $|x| \leq 1 \Longrightarrow of\text{-}real(arccos\ x) = Arccos(of\text{-}real\ x)$
⟨*proof*⟩

52.27 Some interrelationships among the real inverse trig functions.

lemma *arccos-arctan*:
assumes $-1 < x\ x < 1$
shows $arccos\ x = pi/2 - arctan(x\ /\ sqrt(1 - x^2))$
⟨*proof*⟩

lemma *arcsin-plus-arccos*:
assumes $-1 \leq x\ x \leq 1$
shows $arcsin\ x + arccos\ x = pi/2$
⟨*proof*⟩

lemma *arcsin-arccos-eq*: $-1 \leq x \Longrightarrow x \leq 1 \Longrightarrow arcsin\ x = pi/2 - arccos\ x$

⟨*proof*⟩

lemma *arccos-arcsin-eq*: $-1 \leq x \Longrightarrow x \leq 1 \Longrightarrow \arccos x = pi/2 - \arcsin x$
⟨*proof*⟩

lemma *arcsin-arctan*: $-1 < x \Longrightarrow x < 1 \Longrightarrow \arcsin x = \arctan(x \,/\, sqrt(1 - x^2))$
⟨*proof*⟩

lemma *csqrt-1-diff-eq*: $csqrt\ (1 - (of\text{-}real\ x)^2) = (if\ x\hat{}2 \leq 1\ then\ sqrt\ (1 - x\hat{}2)\ else\ \mathrm{i} * sqrt\ (x\hat{}2 - 1))$
⟨*proof*⟩

lemma *arcsin-arccos-sqrt-pos*: $0 \leq x \Longrightarrow x \leq 1 \Longrightarrow \arcsin x = \arccos(sqrt(1 - x^2))$
⟨*proof*⟩

lemma *arcsin-arccos-sqrt-neg*: $-1 \leq x \Longrightarrow x \leq 0 \Longrightarrow \arcsin x = -\arccos(sqrt(1 - x^2))$
⟨*proof*⟩

lemma *arccos-arcsin-sqrt-pos*: $0 \leq x \Longrightarrow x \leq 1 \Longrightarrow \arccos x = \arcsin(sqrt(1 - x^2))$
⟨*proof*⟩

lemma *arccos-arcsin-sqrt-neg*: $-1 \leq x \Longrightarrow x \leq 0 \Longrightarrow \arccos x = pi - \arcsin(sqrt(1 - x^2))$
⟨*proof*⟩

52.28 continuity results for arcsin and arccos.

lemma *continuous-on-Arcsin-real* [*continuous-intros*]:
continuous-on $\{w \in \mathbb{R}.\ |Re\ w| \leq 1\}$ *Arcsin*
⟨*proof*⟩

lemma *continuous-within-Arcsin-real*:
continuous (*at z within* $\{w \in \mathbb{R}.\ |Re\ w| \leq 1\}$) *Arcsin*
⟨*proof*⟩

lemma *continuous-on-Arccos-real*:
continuous-on $\{w \in \mathbb{R}.\ |Re\ w| \leq 1\}$ *Arccos*
⟨*proof*⟩

lemma *continuous-within-Arccos-real*:
continuous (*at z within* $\{w \in \mathbb{R}.\ |Re\ w| \leq 1\}$) *Arccos*
⟨*proof*⟩

52.29 Roots of unity

lemma *complex-root-unity*:

fixes j::*nat*
assumes $n \neq 0$
shows $exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n)\ \hat{}\ n = 1$
⟨*proof*⟩

lemma *complex-root-unity-eq*:
fixes j::*nat* **and** k::*nat*
assumes $1 \leq n$
shows $(exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) = exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ k\ /\ of\text{-}nat\ n)$
$\longleftrightarrow j\ mod\ n = k\ mod\ n)$
⟨*proof*⟩

corollary *bij-betw-roots-unity*:
$bij\text{-}betw\ (\lambda j.\ exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n))$
$\{..<n\}\ \{exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) \mid j.\ j < n\}$
⟨*proof*⟩

lemma *complex-root-unity-eq-1*:
fixes j::*nat* **and** k::*nat*
assumes $1 \leq n$
shows $exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) = 1 \longleftrightarrow n\ dvd\ j$
⟨*proof*⟩

lemma *finite-complex-roots-unity-explicit*:
$finite\ \{exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) \mid j{::}nat.\ j < n\}$
⟨*proof*⟩

lemma *card-complex-roots-unity-explicit*:
$card\ \{exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) \mid j{::}nat.\ j < n\} = n$
⟨*proof*⟩

lemma *complex-roots-unity*:
assumes $1 \leq n$
shows $\{z{::}complex.\ z\ \hat{}\ n = 1\} = \{exp(2 * of\text{-}real\ pi * \mathrm{i} * of\text{-}nat\ j\ /\ of\text{-}nat\ n) \mid j{::}nat.\ j < n\}$
⟨*proof*⟩

lemma *card-complex-roots-unity*: $1 \leq n \Longrightarrow card\ \{z{::}complex.\ z\ \hat{}\ n = 1\} = n$
⟨*proof*⟩

lemma *complex-not-root-unity*:
$1 \leq n \Longrightarrow \exists u{::}complex.\ norm\ u = 1 \wedge u\ \hat{}\ n \neq 1$
⟨*proof*⟩

end

53 Complex path integrals and Cauchy's integral theorem

By John Harrison et al. Ported from HOL Light by L C Paulson (2015)

theory *Cauchy-Integral-Thm*
imports *Complex-Transcendental Weierstrass Ordered-Euclidean-Space*
begin

53.1 Homeomorphisms of arc images

lemma *homeomorphism-arc*:
fixes $g :: real \Rightarrow 'a::t2\text{-}space$
assumes *arc g*
obtains h **where** *homeomorphism* $\{0..1\}$ (*path-image g*) *g h*
⟨*proof*⟩

lemma *homeomorphic-arc-image-interval*:
fixes $g :: real \Rightarrow 'a::t2\text{-}space$ **and** $a::real$
assumes *arc g* $a < b$
shows (*path-image g*) *homeomorphic* $\{a..b\}$
⟨*proof*⟩

lemma *homeomorphic-arc-images*:
fixes $g :: real \Rightarrow 'a::t2\text{-}space$ **and** $h :: real \Rightarrow 'b::t2\text{-}space$
assumes *arc g arc h*
shows (*path-image g*) *homeomorphic* (*path-image h*)
⟨*proof*⟩

53.2 Piecewise differentiable functions

definition *piecewise-differentiable-on*
(**infixr** *piecewise'-differentiable'-on 50*)
where *f piecewise-differentiable-on i* ≡
continuous-on i f ∧
(∃ *s*. *finite s* ∧ (∀ *x* ∈ *i* − *s*. *f differentiable* (*at x within i*)))

lemma *piecewise-differentiable-on-imp-continuous-on*:
f piecewise-differentiable-on s ⟹ *continuous-on s f*
⟨*proof*⟩

lemma *piecewise-differentiable-on-subset*:
f piecewise-differentiable-on s ⟹ $t \le s$ ⟹ *f piecewise-differentiable-on t*
⟨*proof*⟩

lemma *differentiable-on-imp-piecewise-differentiable*:
fixes $a:: 'a::\{linorder\text{-}topology, real\text{-}normed\text{-}vector\}$
shows *f differentiable-on* $\{a..b\}$ ⟹ *f piecewise-differentiable-on* $\{a..b\}$
⟨*proof*⟩

lemma *differentiable-imp-piecewise-differentiable*:
$(\bigwedge x.\ x \in s \Longrightarrow f$ *differentiable* $(at\ x\ within\ s))$
$\Longrightarrow f$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *piecewise-differentiable-const* [*iff*]: $(\lambda x.\ z)$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *piecewise-differentiable-compose*:
$[\![f$ *piecewise-differentiable-on* s; g *piecewise-differentiable-on* $(f\ `\ s)$;
$\bigwedge x.$ *finite* $(s \cap f-`\{x\})]\!]$
$\Longrightarrow (g\ o\ f)$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *piecewise-differentiable-affine*:
fixes m::*real*
assumes f *piecewise-differentiable-on* $((\lambda x.\ m *_R x + c)\ `\ s)$
shows $(f\ o\ (\lambda x.\ m *_R x + c))$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *piecewise-differentiable-cases*:
fixes c::*real*
assumes f *piecewise-differentiable-on* $\{a..c\}$
g *piecewise-differentiable-on* $\{c..b\}$
$a \le c\ c \le b\ f\ c = g\ c$
shows $(\lambda x.$ *if* $x \le c$ *then* $f\ x$ *else* $g\ x)$ *piecewise-differentiable-on* $\{a..b\}$
⟨*proof*⟩

lemma *piecewise-differentiable-neg*:
f *piecewise-differentiable-on* $s \Longrightarrow (\lambda x.\ -(f\ x))$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *piecewise-differentiable-add*:
assumes f *piecewise-differentiable-on* i
g *piecewise-differentiable-on* i
shows $(\lambda x.\ f\ x + g\ x)$ *piecewise-differentiable-on* i
⟨*proof*⟩

lemma *piecewise-differentiable-diff*:
$[\![f$ *piecewise-differentiable-on* s; g *piecewise-differentiable-on* $s]\!]$
$\Longrightarrow (\lambda x.\ f\ x - g\ x)$ *piecewise-differentiable-on* s
⟨*proof*⟩

lemma *continuous-on-joinpaths-D1*:
continuous-on $\{0..1\}$ $(g1\ +\!\!+\!\!+\ g2) \Longrightarrow$ *continuous-on* $\{0..1\}$ $g1$
⟨*proof*⟩

lemma *continuous-on-joinpaths-D2*:
$[\![$*continuous-on* $\{0..1\}$ $(g1\ +\!\!+\!\!+\ g2)$; *pathfinish* $g1$ = *pathstart* $g2]\!] \Longrightarrow$ *continuous-on*

$\{0..1\}$ *g2*
⟨*proof*⟩

lemma *piecewise-differentiable-D1*:
(*g1* +++ *g2*) *piecewise-differentiable-on* $\{0..1\} \Longrightarrow$ *g1 piecewise-differentiable-on* $\{0..1\}$
⟨*proof*⟩

lemma *piecewise-differentiable-D2*:
⟦(*g1* +++ *g2*) *piecewise-differentiable-on* $\{0..1\}$; *pathfinish g1* = *pathstart g2*⟧
$\Longrightarrow$ *g2 piecewise-differentiable-on* $\{0..1\}$
⟨*proof*⟩

53.2.1 The concept of continuously differentiable

John Harrison writes as follows:

"The usual assumption in complex analysis texts is that a path γ should be piecewise continuously differentiable, which ensures that the path integral exists at least for any continuous f, since all piecewise continuous functions are integrable. However, our notion of validity is weaker, just piecewise differentiability... [namely] continuity plus differentiability except on a nite set ... [Our] underlying theory of integration is the Kurzweil-Henstock theory. In contrast to the Riemann or Lebesgue theory (but in common with a simple notion based on antiderivatives), this can integrate all derivatives."

"Formalizing basic complex analysis." From Insight to Proof: Festschrift in Honour of Andrzej Trybulec. Studies in Logic, Grammar and Rhetoric 10.23 (2007): 151-165.

And indeed he does not assume that his derivatives are continuous, but the penalty is unreasonably difficult proofs concerning winding numbers. We need a self-contained and straightforward theorem asserting that all derivatives can be integrated before we can adopt Harrison's choice.

definition *C1-differentiable-on* :: (*real* ⇒ *'a*::*real-normed-vector*) ⇒ *real set* ⇒ *bool*
(**infix** *C1'-differentiable'-on 50*)
where
f C1-differentiable-on s ⟷
(∃ *D*. (∀ *x* ∈ *s*. (*f has-vector-derivative* (*D x*)) (*at x*)) ∧ *continuous-on s D*)

lemma *C1-differentiable-on-eq*:
f C1-differentiable-on s ⟷
(∀ *x* ∈ *s*. *f differentiable at x*) ∧ *continuous-on s* (λ*x*. *vector-derivative f* (*at x*))
⟨*proof*⟩

lemma *C1-differentiable-on-subset*:
f C1-differentiable-on t $\Longrightarrow$ *s* ⊆ *t* $\Longrightarrow$ *f C1-differentiable-on s*

⟨*proof*⟩

lemma *C1-differentiable-compose*:
⟦*f C1-differentiable-on s*; *g C1-differentiable-on* (*f ' s*);
⋀*x*. *finite* (*s* ∩ *f* −'{*x*})⟧
⟹ (*g o f*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-diff-imp-diff*: *f C1-differentiable-on s* ⟹ *f differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-ident* [*simp*, *derivative-intros*]: (λ*x*. *x*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-const* [*simp*, *derivative-intros*]: (λ*z*. *a*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-add* [*simp*, *derivative-intros*]:
f C1-differentiable-on s ⟹ *g C1-differentiable-on s* ⟹ (λ*x*. *f x* + *g x*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-minus* [*simp*, *derivative-intros*]:
f C1-differentiable-on s ⟹ (λ*x*. − *f x*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-diff* [*simp*, *derivative-intros*]:
f C1-differentiable-on s ⟹ *g C1-differentiable-on s* ⟹ (λ*x*. *f x* − *g x*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-mult* [*simp*, *derivative-intros*]:
fixes *f g* :: *real* ⇒ *'a* :: *real-normed-algebra*
shows *f C1-differentiable-on s* ⟹ *g C1-differentiable-on s* ⟹ (λ*x*. *f x* ∗ *g x*) *C1-differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-on-scaleR* [*simp*, *derivative-intros*]:
f C1-differentiable-on s ⟹ *g C1-differentiable-on s* ⟹ (λ*x*. *f x* ∗$_R$ *g x*) *C1-differentiable-on s*
⟨*proof*⟩

definition *piecewise-C1-differentiable-on*
(**infixr** *piecewise'-C1'-differentiable'-on* 50)
where *f piecewise-C1-differentiable-on i* ≡
continuous-on i f ∧

$(\exists\, s.\ \mathit{finite}\ s \wedge (f\ \mathit{C1\text{-}differentiable\text{-}on}\ (i - s)))$

lemma *C1-differentiable-imp-piecewise*:
$f\ \mathit{C1\text{-}differentiable\text{-}on}\ s \Longrightarrow f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s$
⟨*proof*⟩

lemma *piecewise-C1-imp-differentiable*:
$f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ i \Longrightarrow f\ \mathit{piecewise\text{-}differentiable\text{-}on}\ i$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-compose*:
$[\![f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s;\ g\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ (f\ `\ s);$
$\bigwedge x.\ \mathit{finite}\ (s \cap f - `\{x\})]\!]$
$\Longrightarrow (g\ o\ f)\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-on-subset*:
$f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s \Longrightarrow t \le s \Longrightarrow f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ t$
⟨*proof*⟩

lemma *C1-differentiable-imp-continuous-on*:
$f\ \mathit{C1\text{-}differentiable\text{-}on}\ s \Longrightarrow \mathit{continuous\text{-}on}\ s\ f$
⟨*proof*⟩

lemma *C1-differentiable-on-empty* [*iff*]: $f\ \mathit{C1\text{-}differentiable\text{-}on}\ \{\}$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-affine*:
fixes m::*real*
assumes $f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ ((\lambda x.\ m * x + c)\ `\ s)$
shows $(f\ o\ (\lambda x.\ m *_R x + c))\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-cases*:
fixes c::*real*
assumes $f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ \{a..c\}$
$g\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ \{c..b\}$
$a \le c\ c \le b\ f\ c = g\ c$
shows $(\lambda x.\ \mathit{if}\ x \le c\ \mathit{then}\ f\ x\ \mathit{else}\ g\ x)\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ \{a..b\}$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-neg*:
$f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s \Longrightarrow (\lambda x.\ -(f\ x))\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ s$
⟨*proof*⟩

lemma *piecewise-C1-differentiable-add*:
assumes $f\ \mathit{piecewise\text{-}C1\text{-}differentiable\text{-}on}\ i$

g piecewise-C1-differentiable-on i
shows $(\lambda x.\ f\ x + g\ x)$ *piecewise-C1-differentiable-on i*
⟨*proof*⟩

lemma *piecewise-C1-differentiable-diff*:
$[\![f$ *piecewise-C1-differentiable-on s*; *g piecewise-C1-differentiable-on s*$]\!]$
$\Longrightarrow (\lambda x.\ f\ x - g\ x)$ *piecewise-C1-differentiable-on s*
⟨*proof*⟩

lemma *piecewise-C1-differentiable-D1*:
fixes *g1* :: *real* ⇒ *'a*::*real-normed-field*
assumes *(g1 +++ g2) piecewise-C1-differentiable-on {0..1}*
shows *g1 piecewise-C1-differentiable-on {0..1}*
⟨*proof*⟩

lemma *piecewise-C1-differentiable-D2*:
fixes *g2* :: *real* ⇒ *'a*::*real-normed-field*
assumes *(g1 +++ g2) piecewise-C1-differentiable-on {0..1} pathfinish g1 = pathstart g2*
shows *g2 piecewise-C1-differentiable-on {0..1}*
⟨*proof*⟩

53.3 Valid paths, and their start and finish

lemma *Diff-Un-eq*: $A - (B \cup C) = A - B - C$
⟨*proof*⟩

definition *valid-path* :: *(real* ⇒ *'a* :: *real-normed-vector)* ⇒ *bool*
where *valid-path f* ≡ *f piecewise-C1-differentiable-on {0..1*::*real}*

definition *closed-path* :: *(real* ⇒ *'a* :: *real-normed-vector)* ⇒ *bool*
where *closed-path g* ≡ *g 0 = g 1*

53.3.1 In particular, all results for paths apply

lemma *valid-path-imp-path*: *valid-path g* $\Longrightarrow$ *path g*
⟨*proof*⟩

lemma *connected-valid-path-image*: *valid-path g* $\Longrightarrow$ *connected*(*path-image g*)
⟨*proof*⟩

lemma *compact-valid-path-image*: *valid-path g* $\Longrightarrow$ *compact*(*path-image g*)
⟨*proof*⟩

lemma *bounded-valid-path-image*: *valid-path g* $\Longrightarrow$ *bounded*(*path-image g*)
⟨*proof*⟩

lemma *closed-valid-path-image*: *valid-path g* $\Longrightarrow$ *closed*(*path-image g*)
⟨*proof*⟩

proposition *valid-path-compose*:
assumes *valid-path g*
and *der*: $\bigwedge x.\ x \in$ *path-image g* $\Longrightarrow \exists f'.$ (*f has-field-derivative f'*) (*at x*)
and *con*: *continuous-on* (*path-image g*) (*deriv f*)
shows *valid-path* (*f o g*)
⟨*proof*⟩

53.4 Contour Integrals along a path

This definition is for complex numbers only, and does not generalise to line integrals in a vector field

piecewise differentiable function on [0,1]

definition *has-contour-integral* :: (*complex* ⇒ *complex*) ⇒ *complex* ⇒ (*real* ⇒ *complex*) ⇒ *bool*
(**infixr** *has'-contour'-integral 50*)
where (*f has-contour-integral i*) *g* ≡
((λ*x*. *f*(*g x*) ∗ *vector-derivative g* (*at x within* {*0..1*}))
has-integral i) {*0..1*}

definition *contour-integrable-on*
(**infixr** *contour'-integrable'-on 50*)
where *f contour-integrable-on g* ≡ ∃ *i*. (*f has-contour-integral i*) *g*

definition *contour-integral*
where *contour-integral g f* ≡ @*i*. (*f has-contour-integral i*) *g* ∨ ~ *f contour-integrable-on g* ∧ *i=0*

lemma *not-integrable-contour-integral*: ~ *f contour-integrable-on g* ⟹ *contour-integral g f* = *0*
⟨*proof*⟩

lemma *contour-integral-unique*: (*f has-contour-integral i*) *g* ⟹ *contour-integral g f* = *i*
⟨*proof*⟩

corollary *has-contour-integral-eqpath*:
⟦(*f has-contour-integral y*) *p*; *f contour-integrable-on* γ;
contour-integral p f = *contour-integral* γ *f*⟧
⟹ (*f has-contour-integral y*) γ
⟨*proof*⟩

lemma *has-contour-integral-integral*:
f contour-integrable-on i ⟹ (*f has-contour-integral* (*contour-integral i f*)) *i*
⟨*proof*⟩

lemma *has-contour-integral-unique*:
(*f has-contour-integral i*) *g* ⟹ (*f has-contour-integral j*) *g* ⟹ *i* = *j*
⟨*proof*⟩

lemma *has-contour-integral-integrable*: $(f \text{ has-contour-integral } i)\ g \Longrightarrow f \text{ contour-integrable-on } g$
⟨*proof*⟩

lemma *vector-derivative-within-interior*:
$[\![x \in \text{interior } s;\ \text{NO-MATCH UNIV } s]\!] \Longrightarrow \text{vector-derivative } f\ (\text{at } x \text{ within } s) = \text{vector-derivative } f\ (\text{at } x)$
⟨*proof*⟩

lemma *has-integral-localized-vector-derivative*:
$((\lambda x.\ f\ (g\ x) * \text{vector-derivative } g\ (\text{at } x \text{ within } \{a..b\}))\ \text{has-integral } i)\ \{a..b\} \longleftrightarrow ((\lambda x.\ f\ (g\ x) * \text{vector-derivative } g\ (\text{at } x))\ \text{has-integral } i)\ \{a..b\}$
⟨*proof*⟩

lemma *integrable-on-localized-vector-derivative*:
$(\lambda x.\ f\ (g\ x) * \text{vector-derivative } g\ (\text{at } x \text{ within } \{a..b\}))\ \text{integrable-on } \{a..b\} \longleftrightarrow (\lambda x.\ f\ (g\ x) * \text{vector-derivative } g\ (\text{at } x))\ \text{integrable-on } \{a..b\}$
⟨*proof*⟩

lemma *has-contour-integral*:
$(f \text{ has-contour-integral } i)\ g \longleftrightarrow ((\lambda x.\ f\ (g\ x) * \text{vector-derivative } g\ (\text{at } x))\ \text{has-integral } i)\ \{0..1\}$
⟨*proof*⟩

lemma *contour-integrable-on*:
$f \text{ contour-integrable-on } g \longleftrightarrow (\lambda t.\ f(g\ t) * \text{vector-derivative } g\ (\text{at } t))\ \text{integrable-on } \{0..1\}$
⟨*proof*⟩

53.5 Reversing a path

lemma *valid-path-imp-reverse*:
assumes *valid-path g*
shows *valid-path*(*reversepath g*)
⟨*proof*⟩

lemma *valid-path-reversepath* [*simp*]: *valid-path*(*reversepath g*) ⟷ *valid-path g*
⟨*proof*⟩

lemma *has-contour-integral-reversepath*:
assumes *valid-path g* $(f \text{ has-contour-integral } i)\ g$
shows $(f \text{ has-contour-integral } (-i))$ (*reversepath g*)
⟨*proof*⟩

lemma *contour-integrable-reversepath*:

valid-path g $\Longrightarrow$ *f contour-integrable-on g* $\Longrightarrow$ *f contour-integrable-on* (*reversepath g*)
⟨*proof*⟩

lemma *contour-integrable-reversepath-eq*:
valid-path g $\Longrightarrow$ (*f contour-integrable-on* (*reversepath g*) $\longleftrightarrow$ *f contour-integrable-on g*)
⟨*proof*⟩

lemma *contour-integral-reversepath*:
assumes *valid-path g*
shows *contour-integral* (*reversepath g*) *f* = − (*contour-integral g f*)
⟨*proof*⟩

53.6 Joining two paths together

lemma *valid-path-join*:
assumes *valid-path g1 valid-path g2 pathfinish g1* = *pathstart g2*
shows *valid-path*(*g1* +++ *g2*)
⟨*proof*⟩

lemma *valid-path-join-D1*:
fixes *g1* :: *real* ⇒ *'a*::*real-normed-field*
shows *valid-path* (*g1* +++ *g2*) $\Longrightarrow$ *valid-path g1*
⟨*proof*⟩

lemma *valid-path-join-D2*:
fixes *g2* :: *real* ⇒ *'a*::*real-normed-field*
shows ⟦*valid-path* (*g1* +++ *g2*); *pathfinish g1* = *pathstart g2*⟧ $\Longrightarrow$ *valid-path g2*
⟨*proof*⟩

lemma *valid-path-join-eq* [*simp*]:
fixes *g2* :: *real* ⇒ *'a*::*real-normed-field*
shows *pathfinish g1* = *pathstart g2* $\Longrightarrow$ (*valid-path*(*g1* +++ *g2*) $\longleftrightarrow$ *valid-path g1* ∧ *valid-path g2*)
⟨*proof*⟩

lemma *has-contour-integral-join*:
assumes (*f has-contour-integral i1*) *g1* (*f has-contour-integral i2*) *g2*
valid-path g1 valid-path g2
shows (*f has-contour-integral* (*i1* + *i2*)) (*g1* +++ *g2*)
⟨*proof*⟩

lemma *contour-integrable-joinI*:
assumes *f contour-integrable-on g1 f contour-integrable-on g2*
valid-path g1 valid-path g2
shows *f contour-integrable-on* (*g1* +++ *g2*)
⟨*proof*⟩

lemma *contour-integrable-joinD1*:
assumes *f contour-integrable-on* (*g1 +++ g2*) *valid-path g1*
shows *f contour-integrable-on g1*
⟨*proof*⟩

lemma *contour-integrable-joinD2*:
assumes *f contour-integrable-on* (*g1 +++ g2*) *valid-path g2*
shows *f contour-integrable-on g2*
⟨*proof*⟩

lemma *contour-integrable-join* [*simp*]:
shows
⟦*valid-path g1*; *valid-path g2*⟧
⟹ *f contour-integrable-on* (*g1 +++ g2*) ⟷ *f contour-integrable-on g1* ∧ *f contour-integrable-on g2*
⟨*proof*⟩

lemma *contour-integral-join* [*simp*]:
shows
⟦*f contour-integrable-on g1*; *f contour-integrable-on g2*; *valid-path g1*; *valid-path g2*⟧
⟹ *contour-integral* (*g1 +++ g2*) *f* = *contour-integral g1 f* + *contour-integral g2 f*
⟨*proof*⟩

53.7 Shifting the starting point of a (closed) path

lemma *shiftpath-alt-def*: *shiftpath a f* = (λ*x*. *if x* ≤ *1*−*a then f* (*a* + *x*) *else f* (*a* + *x* − *1*))
⟨*proof*⟩

lemma *valid-path-shiftpath* [*intro*]:
assumes *valid-path g pathfinish g* = *pathstart g a* ∈ {*0..1*}
shows *valid-path*(*shiftpath a g*)
⟨*proof*⟩

lemma *has-contour-integral-shiftpath*:
assumes *f*: (*f has-contour-integral i*) *g valid-path g*
and *a*: *a* ∈ {*0..1*}
shows (*f has-contour-integral i*) (*shiftpath a g*)
⟨*proof*⟩

lemma *has-contour-integral-shiftpath-D*:
assumes (*f has-contour-integral i*) (*shiftpath a g*)
valid-path g pathfinish g = *pathstart g a* ∈ {*0..1*}
shows (*f has-contour-integral i*) *g*
⟨*proof*⟩

lemma *has-contour-integral-shiftpath-eq*:

assumes *valid-path g pathfinish g = pathstart g a* $\in \{0..1\}$
shows (*f has-contour-integral i*) (*shiftpath a g*) $\longleftrightarrow$ (*f has-contour-integral i*) *g*
⟨*proof*⟩

lemma *contour-integrable-on-shiftpath-eq*:
assumes *valid-path g pathfinish g = pathstart g a* $\in \{0..1\}$
shows *f contour-integrable-on* (*shiftpath a g*) $\longleftrightarrow$ *f contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integral-shiftpath*:
assumes *valid-path g pathfinish g = pathstart g a* $\in \{0..1\}$
shows *contour-integral* (*shiftpath a g*) *f* = *contour-integral g f*
⟨*proof*⟩

53.8 More about straight-line paths

lemma *has-vector-derivative-linepath-within*:
(*linepath a b has-vector-derivative* (*b* − *a*)) (*at x within s*)
⟨*proof*⟩

lemma *vector-derivative-linepath-within*:
x $\in \{0..1\} \Longrightarrow$ *vector-derivative* (*linepath a b*) (*at x within* $\{0..1\}$) = *b* − *a*
⟨*proof*⟩

lemma *vector-derivative-linepath-at* [*simp*]: *vector-derivative* (*linepath a b*) (*at x*)
= *b* − *a*
⟨*proof*⟩

lemma *valid-path-linepath* [*iff*]: *valid-path* (*linepath a b*)
⟨*proof*⟩

lemma *has-contour-integral-linepath*:
shows (*f has-contour-integral i*) (*linepath a b*) $\longleftrightarrow$
((λ*x*. *f*(*linepath a b x*) ∗ (*b* − *a*)) *has-integral i*) $\{0..1\}$
⟨*proof*⟩

lemma *linepath-in-path*:
shows *x* $\in \{0..1\} \Longrightarrow$ *linepath a b x* $\in$ *closed-segment a b*
⟨*proof*⟩

lemma *linepath-image-01*: *linepath a b* ‘ $\{0..1\}$ = *closed-segment a b*
⟨*proof*⟩

lemma *linepath-in-convex-hull*:
fixes *x*::*real*
assumes *a*: *a* $\in$ *convex hull s*
and *b*: *b* $\in$ *convex hull s*
and *x*: 0≤*x x*≤1
shows *linepath a b x* $\in$ *convex hull s*

⟨*proof*⟩

lemma *Re-linepath*: *Re*(*linepath* (*of-real a*) (*of-real b*) *x*) = (*1* − *x*)∗*a* + *x*∗*b*
⟨*proof*⟩

lemma *Im-linepath*: *Im*(*linepath* (*of-real a*) (*of-real b*) *x*) = *0*
⟨*proof*⟩

lemma *has-contour-integral-trivial* [*iff*]: (*f has-contour-integral 0*) (*linepath a a*)
⟨*proof*⟩

lemma *contour-integral-trivial* [*simp*]: *contour-integral* (*linepath a a*) *f* = *0*
⟨*proof*⟩

53.9 Relation to subpath construction

lemma *valid-path-subpath*:
fixes *g* :: *real* ⇒ *'a* :: *real-normed-vector*
assumes *valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*}
shows *valid-path*(*subpath u v g*)
⟨*proof*⟩

lemma *has-contour-integral-subpath-refl* [*iff*]: (*f has-contour-integral 0*) (*subpath u u g*)
⟨*proof*⟩

lemma *contour-integrable-subpath-refl* [*iff*]: *f contour-integrable-on* (*subpath u u g*)
⟨*proof*⟩

lemma *contour-integral-subpath-refl* [*simp*]: *contour-integral* (*subpath u u g*) *f* = *0*
⟨*proof*⟩

lemma *has-contour-integral-subpath*:
assumes *f*: *f contour-integrable-on g* **and** *g*: *valid-path g*
and *uv*: *u* ∈ {*0..1*} *v* ∈ {*0..1*} *u* ≤ *v*
shows (*f has-contour-integral integral* {*u..v*} (λ*x*. *f*(*g x*) ∗ *vector-derivative g* (*at x*)))
(*subpath u v g*)
⟨*proof*⟩

lemma *contour-integrable-subpath*:
assumes *f contour-integrable-on g valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*}
shows *f contour-integrable-on* (*subpath u v g*)
⟨*proof*⟩

lemma *has-integral-integrable-integral*: (*f has-integral i*) *s* ⟷ *f integrable-on s* ∧ *integral s f* = *i*

⟨*proof*⟩

lemma *has-integral-contour-integral-subpath*:
assumes *f contour-integrable-on g valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*} *u* ≤ *v*
shows (((λ*x*. *f*(*g x*) ∗ *vector-derivative g* (*at x*)))
has-integral contour-integral (*subpath u v g*) *f*) {*u..v*}
⟨*proof*⟩

lemma *contour-integral-subcontour-integral*:
assumes *f contour-integrable-on g valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*} *u* ≤ *v*
shows *contour-integral* (*subpath u v g*) *f* =
integral {*u..v*} (λ*x*. *f*(*g x*) ∗ *vector-derivative g* (*at x*))
⟨*proof*⟩

lemma *contour-integral-subpath-combine-less*:
assumes *f contour-integrable-on g valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*} *w* ∈ {*0..1*}
u<*v v*<*w*
shows *contour-integral* (*subpath u v g*) *f* + *contour-integral* (*subpath v w g*) *f*
=
contour-integral (*subpath u w g*) *f*
⟨*proof*⟩

lemma *contour-integral-subpath-combine*:
assumes *f contour-integrable-on g valid-path g u* ∈ {*0..1*} *v* ∈ {*0..1*} *w* ∈ {*0..1*}
shows *contour-integral* (*subpath u v g*) *f* + *contour-integral* (*subpath v w g*) *f*
=
contour-integral (*subpath u w g*) *f*
⟨*proof*⟩

lemma *contour-integral-integral*:
contour-integral g f = *integral* {*0..1*} (λ*x*. *f* (*g x*) ∗ *vector-derivative g* (*at x*))
⟨*proof*⟩

Cauchy's theorem where there's a primitive

lemma *contour-integral-primitive-lemma*:
fixes *f* :: *complex* ⇒ *complex* **and** *g* :: *real* ⇒ *complex*
assumes *a* ≤ *b*
and ⋀*x*. *x* ∈ *s* ⟹ (*f has-field-derivative f*′ *x*) (*at x within s*)
and *g piecewise-differentiable-on* {*a..b*} ⋀*x*. *x* ∈ {*a..b*} ⟹ *g x* ∈ *s*
shows ((λ*x*. *f*′(*g x*) ∗ *vector-derivative g* (*at x within* {*a..b*}))
has-integral (*f*(*g b*) − *f*(*g a*))) {*a..b*}
⟨*proof*⟩

lemma *contour-integral-primitive*:
assumes ⋀*x*. *x* ∈ *s* ⟹ (*f has-field-derivative f*′ *x*) (*at x within s*)
and *valid-path g path-image g* ⊆ *s*
shows (*f*′ *has-contour-integral* (*f*(*pathfinish g*) − *f*(*pathstart g*))) *g*
⟨*proof*⟩

corollary *Cauchy-theorem-primitive*:
assumes $\bigwedge x.\ x \in s \Longrightarrow (f \text{ has-field-derivative } f'\ x)\ (\text{at } x \text{ within } s)$
and *valid-path g path-image g* $\subseteq s$ *pathfinish g = pathstart g*
shows $(f'$ *has-contour-integral 0*$)\ g$
⟨*proof*⟩

Existence of path integral for continuous function

lemma *contour-integrable-continuous-linepath*:
assumes *continuous-on* (*closed-segment a b*) *f*
shows *f contour-integrable-on* (*linepath a b*)
⟨*proof*⟩

lemma *has-field-der-id*: $((\lambda x.\ x^2\ /\ 2)$ *has-field-derivative* $x)\ (\text{at } x)$
⟨*proof*⟩

lemma *contour-integral-id* [*simp*]: *contour-integral* (*linepath a b*) $(\lambda y.\ y) = (b\hat{}2 - a\hat{}2)/2$
⟨*proof*⟩

lemma *contour-integrable-on-const* [*iff*]: $(\lambda x.\ c)$ *contour-integrable-on* (*linepath a b*)
⟨*proof*⟩

lemma *contour-integrable-on-id* [*iff*]: $(\lambda x.\ x)$ *contour-integrable-on* (*linepath a b*)
⟨*proof*⟩

53.10 Arithmetical combining theorems

lemma *has-contour-integral-neg*:
$(f$ *has-contour-integral i*$)\ g \Longrightarrow ((\lambda x.\ -(f\ x))$ *has-contour-integral* $(-i))\ g$
⟨*proof*⟩

lemma *has-contour-integral-add*:
$[\![(f1$ *has-contour-integral i1*$)\ g;\ (f2$ *has-contour-integral i2*$)\ g]\!]$
$\Longrightarrow ((\lambda x.\ f1\ x + f2\ x)$ *has-contour-integral* $(i1 + i2))\ g$
⟨*proof*⟩

lemma *has-contour-integral-diff*:
$[\![(f1$ *has-contour-integral i1*$)\ g;\ (f2$ *has-contour-integral i2*$)\ g]\!]$
$\Longrightarrow ((\lambda x.\ f1\ x - f2\ x)$ *has-contour-integral* $(i1 - i2))\ g$
⟨*proof*⟩

lemma *has-contour-integral-lmul*:
$(f$ *has-contour-integral i*$)\ g \Longrightarrow ((\lambda x.\ c * (f\ x))$ *has-contour-integral* $(c*i))\ g$
⟨*proof*⟩

lemma *has-contour-integral-rmul*:
$(f$ *has-contour-integral i*$)\ g \Longrightarrow ((\lambda x.\ (f\ x) * c)$ *has-contour-integral* $(i*c))\ g$
⟨*proof*⟩

lemma *has-contour-integral-div*:
(*f has-contour-integral i*) *g* $\Longrightarrow$ ((λ*x*. *f x*/*c*) *has-contour-integral* (*i*/*c*)) *g*
⟨*proof*⟩

lemma *has-contour-integral-eq*:
⟦(*f has-contour-integral y*) *p*; ⋀*x*. *x* ∈ *path-image p* $\Longrightarrow$ *f x* = *g x*⟧ $\Longrightarrow$ (*g has-contour-integral y*) *p*
⟨*proof*⟩

lemma *has-contour-integral-bound-linepath*:
assumes (*f has-contour-integral i*) (*linepath a b*)
0 ≤ *B* ⋀*x*. *x* ∈ *closed-segment a b* $\Longrightarrow$ *norm*(*f x*) ≤ *B*
shows *norm i* ≤ *B* ∗ *norm*(*b* − *a*)
⟨*proof*⟩

lemma *has-contour-integral-const-linepath*: ((λ*x*. *c*) *has-contour-integral c*∗(*b* − *a*))(*linepath a b*)
⟨*proof*⟩

lemma *has-contour-integral-0*: ((λ*x*. 0) *has-contour-integral* 0) *g*
⟨*proof*⟩

lemma *has-contour-integral-is-0*:
(⋀*z*. *z* ∈ *path-image g* $\Longrightarrow$ *f z* = 0) $\Longrightarrow$ (*f has-contour-integral* 0) *g*
⟨*proof*⟩

lemma *has-contour-integral-setsum*:
⟦*finite s*; ⋀*a*. *a* ∈ *s* $\Longrightarrow$ (*f a has-contour-integral i a*) *p*⟧
$\Longrightarrow$ ((λ*x*. *setsum* (λ*a*. *f a x*) *s*) *has-contour-integral setsum i s*) *p*
⟨*proof*⟩

53.11 Operations on path integrals

lemma *contour-integral-const-linepath* [*simp*]: *contour-integral* (*linepath a b*) (λ*x*. *c*) = *c*∗(*b* − *a*)
⟨*proof*⟩

lemma *contour-integral-neg*:
f contour-integrable-on g $\Longrightarrow$ *contour-integral g* (λ*x*. −(*f x*)) = −(*contour-integral g f*)
⟨*proof*⟩

lemma *contour-integral-add*:
f1 contour-integrable-on g $\Longrightarrow$ *f2 contour-integrable-on g* $\Longrightarrow$ *contour-integral g* (λ*x*. *f1 x* + *f2 x*) =
contour-integral g f1 + *contour-integral g f2*

⟨*proof*⟩

lemma *contour-integral-diff*:
f1 contour-integrable-on g ⟹ *f2 contour-integrable-on g* ⟹ *contour-integral g* (λ*x*. *f1 x* − *f2 x*) =
contour-integral g f1 − *contour-integral g f2*
⟨*proof*⟩

lemma *contour-integral-lmul*:
shows *f contour-integrable-on g*
⟹ *contour-integral g* (λ*x*. *c* ∗ *f x*) = *c*∗*contour-integral g f*
⟨*proof*⟩

lemma *contour-integral-rmul*:
shows *f contour-integrable-on g*
⟹ *contour-integral g* (λ*x*. *f x* ∗ *c*) = *contour-integral g f* ∗ *c*
⟨*proof*⟩

lemma *contour-integral-div*:
shows *f contour-integrable-on g*
⟹ *contour-integral g* (λ*x*. *f x* / *c*) = *contour-integral g f* / *c*
⟨*proof*⟩

lemma *contour-integral-eq*:
(⋀*x*. *x* ∈ *path-image p* ⟹ *f x* = *g x*) ⟹ *contour-integral p f* = *contour-integral p g*
⟨*proof*⟩

lemma *contour-integral-eq-0*:
(⋀*z*. *z* ∈ *path-image g* ⟹ *f z* = *0*) ⟹ *contour-integral g f* = *0*
⟨*proof*⟩

lemma *contour-integral-bound-linepath*:
shows
⟦*f contour-integrable-on* (*linepath a b*);
0 ≤ *B*; ⋀*x*. *x* ∈ *closed-segment a b* ⟹ *norm*(*f x*) ≤ *B*⟧
⟹ *norm*(*contour-integral* (*linepath a b*) *f*) ≤ *B*∗*norm*(*b* − *a*)
⟨*proof*⟩

lemma *contour-integral-0* [*simp*]: *contour-integral g* (λ*x*. *0*) = *0*
⟨*proof*⟩

lemma *contour-integral-setsum*:
⟦*finite s*; ⋀*a*. *a* ∈ *s* ⟹ (*f a*) *contour-integrable-on p*⟧
⟹ *contour-integral p* (λ*x*. *setsum* (λ*a*. *f a x*) *s*) = *setsum* (λ*a*. *contour-integral p* (*f a*)) *s*
⟨*proof*⟩

lemma *contour-integrable-eq*:

⟦*f contour-integrable-on p*; ⋀*x*. *x* ∈ *path-image p* ⟹ *f x* = *g x*⟧ ⟹ *g contour-integrable-on p*
⟨*proof*⟩

53.12 Arithmetic theorems for path integrability

lemma *contour-integrable-neg*:
f contour-integrable-on g ⟹ (λ*x*. −(*f x*)) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-add*:
⟦*f1 contour-integrable-on g*; *f2 contour-integrable-on g*⟧ ⟹ (λ*x*. *f1 x* + *f2 x*) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-diff*:
⟦*f1 contour-integrable-on g*; *f2 contour-integrable-on g*⟧ ⟹ (λ*x*. *f1 x* − *f2 x*) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-lmul*:
f contour-integrable-on g ⟹ (λ*x*. *c* ∗ *f x*) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-rmul*:
f contour-integrable-on g ⟹ (λ*x*. *f x* ∗ *c*) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-div*:
f contour-integrable-on g ⟹ (λ*x*. *f x* / *c*) *contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-setsum*:
⟦*finite s*; ⋀*a*. *a* ∈ *s* ⟹ (*f a*) *contour-integrable-on p*⟧
⟹ (λ*x*. *setsum* (λ*a*. *f a x*) *s*) *contour-integrable-on p*
⟨*proof*⟩

53.13 Reversing a path integral

lemma *has-contour-integral-reverse-linepath*:
(*f has-contour-integral i*) (*linepath a b*)
⟹ (*f has-contour-integral* (−*i*)) (*linepath b a*)
⟨*proof*⟩

lemma *contour-integral-reverse-linepath*:
continuous-on (*closed-segment a b*) *f*
⟹ *contour-integral* (*linepath a b*) *f* = − (*contour-integral*(*linepath b a*) *f*)
⟨*proof*⟩

lemma *has-contour-integral-split*:
assumes f: (f *has-contour-integral* i) (*linepath* a c) (f *has-contour-integral* j) (*linepath* c b)
and k: $0 \leq k$ $k \leq 1$
and c: $c - a = k *_R (b - a)$
shows (f *has-contour-integral* $(i + j)$) (*linepath* a b)
⟨*proof*⟩

lemma *continuous-on-closed-segment-transform*:
assumes f: *continuous-on* (*closed-segment* a b) f
and k: $0 \leq k$ $k \leq 1$
and c: $c - a = k *_R (b - a)$
shows *continuous-on* (*closed-segment* a c) f
⟨*proof*⟩

lemma *contour-integral-split*:
assumes f: *continuous-on* (*closed-segment* a b) f
and k: $0 \leq k$ $k \leq 1$
and c: $c - a = k *_R (b - a)$
shows *contour-integral*(*linepath* a b) f = *contour-integral*(*linepath* a c) f + *contour-integral*(*linepath* c b) f
⟨*proof*⟩

lemma *contour-integral-split-linepath*:
assumes f: *continuous-on* (*closed-segment* a b) f
and c: $c \in$ *closed-segment* a b
shows *contour-integral*(*linepath* a b) f = *contour-integral*(*linepath* a c) f + *contour-integral*(*linepath* c b) f
⟨*proof*⟩

lemma *has-contour-integral-midpoint*:
assumes (f *has-contour-integral* i) (*linepath* a (*midpoint* a b))
(f *has-contour-integral* j) (*linepath* (*midpoint* a b) b)
shows (f *has-contour-integral* $(i + j)$) (*linepath* a b)
⟨*proof*⟩

lemma *contour-integral-midpoint*:
continuous-on (*closed-segment* a b) f
$\Longrightarrow$ *contour-integral* (*linepath* a b) f =
contour-integral (*linepath* a (*midpoint* a b)) f + *contour-integral* (*linepath* (*midpoint* a b) b) f
⟨*proof*⟩

A couple of special case lemmas that are useful below

lemma *triangle-linear-has-chain-integral*:

((λx. m*x + d) *has-contour-integral* 0) (*linepath a b +++ linepath b c +++ linepath c a*)
⟨*proof*⟩

lemma *has-chain-integral-chain-integral3*:
(*f has-contour-integral i*) (*linepath a b +++ linepath b c +++ linepath c d*)
⟹ *contour-integral* (*linepath a b*) *f* + *contour-integral* (*linepath b c*) *f* + *contour-integral* (*linepath c d*) *f* = *i*
⟨*proof*⟩

lemma *has-chain-integral-chain-integral4*:
(*f has-contour-integral i*) (*linepath a b +++ linepath b c +++ linepath c d +++ linepath d e*)
⟹ *contour-integral* (*linepath a b*) *f* + *contour-integral* (*linepath b c*) *f* + *contour-integral* (*linepath c d*) *f* + *contour-integral* (*linepath d e*) *f* = *i*
⟨*proof*⟩

53.14 Reversing the order in a double path integral

The condition is stronger than needed but it's often true in typical situations

lemma *fst-im-cbox* [*simp*]: *cbox c d* ≠ {} ⟹ (*fst* ' *cbox* (*a*,*c*) (*b*,*d*)) = *cbox a b*
⟨*proof*⟩

lemma *snd-im-cbox* [*simp*]: *cbox a b* ≠ {} ⟹ (*snd* ' *cbox* (*a*,*c*) (*b*,*d*)) = *cbox c d*
⟨*proof*⟩

lemma *contour-integral-swap*:
assumes *fcon*: *continuous-on* (*path-image g* × *path-image h*) (λ(*y1*,*y2*). *f y1 y2*)
and *vp*: *valid-path g valid-path h*
and *gvcon*: *continuous-on* {*0..1*} (λt. *vector-derivative g* (*at t*))
and *hvcon*: *continuous-on* {*0..1*} (λt. *vector-derivative h* (*at t*))
shows *contour-integral g* (λw. *contour-integral h* (*f w*)) =
contour-integral h (λz. *contour-integral g* (λw. *f w z*))
⟨*proof*⟩

53.15 The key quadrisection step

lemma *norm-sum-half*:
assumes *norm*(*a* + *b*) >= *e*
shows *norm a* >= *e*/*2* ∨ *norm b* >= *e*/*2*
⟨*proof*⟩

lemma *norm-sum-lemma*:
assumes *e* ≤ *norm* (*a* + *b* + *c* + *d*)
shows *e* / *4* ≤ *norm a* ∨ *e* / *4* ≤ *norm b* ∨ *e* / *4* ≤ *norm c* ∨ *e* / *4* ≤ *norm d*
⟨*proof*⟩

lemma *Cauchy-theorem-quadrisection*:
assumes *f*: *continuous-on* (*convex hull* {*a*,*b*,*c*}) *f*
and *dist*: *dist a b* ≤ *K dist b c* ≤ *K dist c a* ≤ *K*
and *e*: *e* ∗ *K*^*2* ≤
norm (*contour-integral*(*linepath a b*) *f* + *contour-integral*(*linepath b c*) *f* + *contour-integral*(*linepath c a*) *f*)
shows ∃ *a*′ *b*′ *c*′.
a′ ∈ *convex hull* {*a*,*b*,*c*} ∧ *b*′ ∈ *convex hull* {*a*,*b*,*c*} ∧ *c*′ ∈ *convex hull* {*a*,*b*,*c*} ∧
dist a′ *b*′ ≤ *K*/*2* ∧ *dist b*′ *c*′ ≤ *K*/*2* ∧ *dist c*′ *a*′ ≤ *K*/*2* ∧
e ∗ (*K*/*2*)^*2* ≤ *norm*(*contour-integral*(*linepath a*′ *b*′) *f* + *contour-integral*(*linepath b*′ *c*′) *f* + *contour-integral*(*linepath c*′ *a*′) *f*)
⟨*proof*⟩

53.16 Cauchy's theorem for triangles

lemma *triangle-points-closer*:
fixes *a*::*complex*
shows ⟦*x* ∈ *convex hull* {*a*,*b*,*c*}; *y* ∈ *convex hull* {*a*,*b*,*c*}⟧
⟹ *norm*(*x* − *y*) ≤ *norm*(*a* − *b*) ∨
norm(*x* − *y*) ≤ *norm*(*b* − *c*) ∨
norm(*x* − *y*) ≤ *norm*(*c* − *a*)
⟨*proof*⟩

lemma *holomorphic-point-small-triangle*:
assumes *x*: *x* ∈ *s*
and *f*: *continuous-on s f*
and *cd*: *f field-differentiable* (*at x within s*)
and *e*: *0* < *e*
shows ∃ *k*>*0*. ∀ *a b c*. *dist a b* ≤ *k* ∧ *dist b c* ≤ *k* ∧ *dist c a* ≤ *k* ∧
x ∈ *convex hull* {*a*,*b*,*c*} ∧ *convex hull* {*a*,*b*,*c*} ⊆ *s*
⟶ *norm*(*contour-integral*(*linepath a b*) *f* + *contour-integral*(*linepath b c*) *f* +
contour-integral(*linepath c a*) *f*)
≤ *e*∗(*dist a b* + *dist b c* + *dist c a*)^*2*
(**is** ∃ *k*>*0*. ∀ *a b c*. - ⟶ *?normle a b c*)
⟨*proof*⟩

locale *Chain* =
fixes *x0 At Follows*
assumes *At0*: *At x0 0*
and *AtSuc*: ⋀*x n*. *At x n* ⟹ ∃ *x*′. *At x*′ (*Suc n*) ∧ *Follows x*′ *x*
begin
primrec *f* **where**
f 0 = *x0*
| *f* (*Suc n*) = (*SOME x*. *At x* (*Suc n*) ∧ *Follows x* (*f n*))

lemma *At*: *At* (*f n*) *n*
⟨*proof*⟩

lemma *Follows*: *Follows* (*f*(*Suc n*)) (*f n*)
⟨*proof*⟩

declare *f.simps*(*2*) [*simp del*]
end

lemma *Chain3*:
assumes *At0*: *At x0 y0 z0 0*
and *AtSuc*: ⋀*x y z n*. *At x y z n* ⟹ ∃ *x' y' z'*. *At x' y' z'* (*Suc n*) ∧ *Follows x' y' z' x y z*
obtains *f g h* **where**
f 0 = *x0 g 0* = *y0 h 0* = *z0*
⋀*n*. *At* (*f n*) (*g n*) (*h n*) *n*
⋀*n*. *Follows* (*f*(*Suc n*)) (*g*(*Suc n*)) (*h*(*Suc n*)) (*f n*) (*g n*) (*h n*)
⟨*proof*⟩

lemma *Cauchy-theorem-triangle*:
assumes *f holomorphic-on* (*convex hull* {*a*,*b*,*c*})
shows (*f has-contour-integral 0*) (*linepath a b* +++ *linepath b c* +++ *linepath c a*)
⟨*proof*⟩

53.17 Version needing function holomorphic in interior only

lemma *Cauchy-theorem-flat-lemma*:
assumes *f*: *continuous-on* (*convex hull* {*a*,*b*,*c*}) *f*
and *c*: *c* − *a* = *k* ∗*R* (*b* − *a*)
and *k*: *0* ≤ *k*
shows *contour-integral* (*linepath a b*) *f* + *contour-integral* (*linepath b c*) *f* + *contour-integral* (*linepath c a*) *f* = *0*
⟨*proof*⟩

lemma *Cauchy-theorem-flat*:
assumes *f*: *continuous-on* (*convex hull* {*a*,*b*,*c*}) *f*
and *c*: *c* − *a* = *k* ∗*R* (*b* − *a*)
shows *contour-integral* (*linepath a b*) *f* +
contour-integral (*linepath b c*) *f* +
contour-integral (*linepath c a*) *f* = *0*
⟨*proof*⟩

lemma *Cauchy-theorem-triangle-interior*:
assumes *contf*: *continuous-on* (*convex hull* {*a*,*b*,*c*}) *f*
and *holf*: *f holomorphic-on interior* (*convex hull* {*a*,*b*,*c*})
shows (*f has-contour-integral 0*) (*linepath a b* +++ *linepath b c* +++ *linepath c a*)

⟨*proof*⟩

53.18 Version allowing finite number of exceptional points

lemma *Cauchy-theorem-triangle-cofinite*:
assumes *continuous-on* (*convex hull* {*a*,*b*,*c*}) *f*
and *finite s*
and (⋀*x*. *x* ∈ *interior*(*convex hull* {*a*,*b*,*c*}) − *s* ⟹ *f field-differentiable* (*at x*))
shows (*f has-contour-integral 0*) (*linepath a b* +++ *linepath b c* +++ *linepath c a*)
⟨*proof*⟩

53.19 Cauchy's theorem for an open starlike set

lemma *starlike-convex-subset*:
assumes *s*: *a* ∈ *s closed-segment b c* ⊆ *s* **and** *subs*: ⋀*x*. *x* ∈ *s* ⟹ *closed-segment a x* ⊆ *s*
shows *convex hull* {*a*,*b*,*c*} ⊆ *s*
⟨*proof*⟩

lemma *triangle-contour-integrals-starlike-primitive*:
assumes *contf*: *continuous-on s f*
and *s*: *a* ∈ *s open s*
and *x*: *x* ∈ *s*
and *subs*: ⋀*y*. *y* ∈ *s* ⟹ *closed-segment a y* ⊆ *s*
and *zer*: ⋀*b c*. *closed-segment b c* ⊆ *s*
⟹ *contour-integral* (*linepath a b*) *f* + *contour-integral* (*linepath b c*) *f* +
contour-integral (*linepath c a*) *f* = *0*
shows ((λ*x*. *contour-integral*(*linepath a x*) *f*) *has-field-derivative f x*) (*at x*)
⟨*proof*⟩

lemma *holomorphic-starlike-primitive*:
fixes *f* :: *complex* ⇒ *complex*
assumes *contf*: *continuous-on s f*
and *s*: *starlike s* **and** *os*: *open s*
and *k*: *finite k*
and *fcd*: ⋀*x*. *x* ∈ *s* − *k* ⟹ *f field-differentiable at x*
shows ∃ *g*. ∀ *x* ∈ *s*. (*g has-field-derivative f x*) (*at x*)
⟨*proof*⟩

lemma *Cauchy-theorem-starlike*:
⟦*open s*; *starlike s*; *finite k*; *continuous-on s f*;
⋀*x*. *x* ∈ *s* − *k* ⟹ *f field-differentiable at x*;
valid-path g; *path-image g* ⊆ *s*; *pathfinish g* = *pathstart g*⟧
⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

lemma *Cauchy-theorem-starlike-simple*:
⟦*open s*; *starlike s*; *f holomorphic-on s*; *valid-path g*; *path-image g* ⊆ *s*; *pathfinish g* = *pathstart g*⟧
⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

53.20 Cauchy's theorem for a convex set

For a convex set we can avoid assuming openness and boundary analyticity

lemma *triangle-contour-integrals-convex-primitive*:
assumes *contf*: *continuous-on s f*
and *s*: *a* ∈ *s convex s*
and *x*: *x* ∈ *s*
and *zer*: ⋀*b c*. ⟦*b* ∈ *s*; *c* ∈ *s*⟧
⟹ *contour-integral* (*linepath a b*) *f* + *contour-integral* (*linepath b c*) *f* +
contour-integral (*linepath c a*) *f* = *0*
shows ((λ*x*. *contour-integral*(*linepath a x*) *f*) *has-field-derivative f x*) (*at x within s*)
⟨*proof*⟩

lemma *contour-integral-convex-primitive*:
⟦*convex s*; *continuous-on s f*;
⋀*a b c*. ⟦*a* ∈ *s*; *b* ∈ *s*; *c* ∈ *s*⟧ ⟹ (*f has-contour-integral 0*) (*linepath a b* +++ *linepath b c* +++ *linepath c a*)⟧
⟹ ∃ *g*. ∀ *x* ∈ *s*. (*g has-field-derivative f x*) (*at x within s*)
⟨*proof*⟩

lemma *holomorphic-convex-primitive*:
fixes *f* :: *complex* ⇒ *complex*
shows
⟦*convex s*; *finite k*; *continuous-on s f*;
⋀*x*. *x* ∈ *interior s* − *k* ⟹ *f field-differentiable at x*⟧
⟹ ∃ *g*. ∀ *x* ∈ *s*. (*g has-field-derivative f x*) (*at x within s*)
⟨*proof*⟩

lemma *Cauchy-theorem-convex*:
⟦*continuous-on s f*; *convex s*; *finite k*;
⋀*x*. *x* ∈ *interior s* − *k* ⟹ *f field-differentiable at x*;
valid-path g; *path-image g* ⊆ *s*;
pathfinish g = *pathstart g*⟧ ⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

lemma *Cauchy-theorem-convex-simple*:
⟦*f holomorphic-on s*; *convex s*;
valid-path g; *path-image g* ⊆ *s*;
pathfinish g = *pathstart g*⟧ ⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

In particular for a disc

lemma *Cauchy-theorem-disc*:
⟦*finite k*; *continuous-on* (*cball a e*) *f*;
⋀*x*. *x* ∈ *ball a e* − *k* ⟹ *f field-differentiable at x*;
valid-path g; *path-image g* ⊆ *cball a e*;
pathfinish g = *pathstart g*⟧ ⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

lemma *Cauchy-theorem-disc-simple*:
⟦*f holomorphic-on* (*ball a e*); *valid-path g*; *path-image g* ⊆ *ball a e*;
pathfinish g = *pathstart g*⟧ ⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

53.21 Generalize integrability to local primitives

lemma *contour-integral-local-primitive-lemma*:
fixes *f* :: *complex*⇒*complex*
shows
⟦*g piecewise-differentiable-on* {*a*..*b*};
⋀*x*. *x* ∈ *s* ⟹ (*f has-field-derivative f* ′ *x*) (*at x within s*);
⋀*x*. *x* ∈ {*a*..*b*} ⟹ *g x* ∈ *s*⟧
⟹ (λ*x*. *f* ′ (*g x*) ∗ *vector-derivative g* (*at x within* {*a*..*b*}))
integrable-on {*a*..*b*}
⟨*proof*⟩

lemma *contour-integral-local-primitive-any*:
fixes *f* :: *complex* ⇒ *complex*
assumes *gpd*: *g piecewise-differentiable-on* {*a*..*b*}
and *dh*: ⋀*x*. *x* ∈ *s*
⟹ ∃ *d h*. 0 < *d* ∧
(∀ *y*. *norm*(*y* − *x*) < *d* ⟶ (*h has-field-derivative f y*) (*at y within s*))
and *gs*: ⋀*x*. *x* ∈ {*a*..*b*} ⟹ *g x* ∈ *s*
shows (λ*x*. *f*(*g x*) ∗ *vector-derivative g* (*at x*)) *integrable-on* {*a*..*b*}
⟨*proof*⟩

lemma *contour-integral-local-primitive*:
fixes *f* :: *complex* ⇒ *complex*
assumes *g*: *valid-path g path-image g* ⊆ *s*
and *dh*: ⋀*x*. *x* ∈ *s*
⟹ ∃ *d h*. 0 < *d* ∧
(∀ *y*. *norm*(*y* − *x*) < *d* ⟶ (*h has-field-derivative f y*) (*at y within s*))
shows *f contour-integrable-on g*
⟨*proof*⟩

In particular if a function is holomorphic

lemma *contour-integrable-holomorphic*:
assumes *contf*: *continuous-on s f*

and *os*: *open s*
and *k*: *finite k*
and *g*: *valid-path g path-image g* $\subseteq$ *s*
and *fcd*: $\bigwedge x.\ x \in s - k \Longrightarrow f$ *field-differentiable at x*
shows *f contour-integrable-on g*
⟨*proof*⟩

lemma *contour-integrable-holomorphic-simple*:
assumes *fh*: *f holomorphic-on s*
and *os*: *open s*
and *g*: *valid-path g path-image g* $\subseteq$ *s*
shows *f contour-integrable-on g*
⟨*proof*⟩

lemma *continuous-on-inversediff*:
fixes *z*:: *'a*::*real-normed-field* **shows** *z* $\notin$ *s* $\Longrightarrow$ *continuous-on s* ($\lambda w.\ 1\ /\ (w - z)$)
⟨*proof*⟩

corollary *contour-integrable-inversediff*:
⟦*valid-path g*; *z* $\notin$ *path-image g*⟧ $\Longrightarrow$ ($\lambda w.\ 1\ /\ (w-z)$) *contour-integrable-on g*
⟨*proof*⟩

Key fact that path integral is the same for a "nearby" path. This is the main lemma for the homotopy form of Cauchy's theorem and is also useful if we want "without loss of generality" to assume some nice properties of a path (e.g. smoothness). It can also be used to define the integrals of analytic functions over arbitrary continuous paths. This is just done for winding numbers now.

A technical definition to avoid duplication of similar proofs, for paths joined at the ends versus looping paths

definition *linked-paths* :: *bool* $\Rightarrow$ (*real* $\Rightarrow$ *'a*) $\Rightarrow$ (*real* $\Rightarrow$ *'a*::*topological-space*) $\Rightarrow$ *bool*
where *linked-paths atends g h* ==
(*if atends then pathstart h* = *pathstart g* $\wedge$ *pathfinish h* = *pathfinish g*
else pathfinish g = *pathstart g* $\wedge$ *pathfinish h* = *pathstart h*)

This formulation covers two cases: g and h share their start and end points; g and h both loop upon themselves.

lemma *contour-integral-nearby*:
assumes *os*: *open s* **and** *p*: *path p path-image p* $\subseteq$ *s*
shows
$\exists\, d.\ 0 < d \wedge$
($\forall\, g\ h.$ *valid-path g* $\wedge$ *valid-path h* $\wedge$
($\forall\, t \in \{0..1\}.$ *norm*(*g t* − *p t*) < *d* $\wedge$ *norm*(*h t* − *p t*) < *d*) $\wedge$
linked-paths atends g h
$\longrightarrow$ *path-image g* $\subseteq$ *s* $\wedge$ *path-image h* $\subseteq$ *s* $\wedge$

$(\forall f.\ f\ holomorphic\text{-}on\ s \longrightarrow contour\text{-}integral\ h\ f = contour\text{-}integral\ g\ f))$
⟨*proof*⟩

lemma
assumes *open s path p path-image p* ⊆ *s*
shows *contour-integral-nearby-ends*:
$\exists d.\ 0 < d \wedge$
$(\forall g\ h.\ valid\text{-}path\ g \wedge valid\text{-}path\ h \wedge$
$(\forall t \in \{0..1\}.\ norm(g\ t - p\ t) < d \wedge norm(h\ t - p\ t) < d) \wedge$
$pathstart\ h = pathstart\ g \wedge pathfinish\ h = pathfinish\ g$
$\longrightarrow path\text{-}image\ g \subseteq s \wedge$
$path\text{-}image\ h \subseteq s \wedge$
$(\forall f.\ f\ holomorphic\text{-}on\ s$
$\longrightarrow contour\text{-}integral\ h\ f = contour\text{-}integral\ g\ f))$
and *contour-integral-nearby-loops*:
$\exists d.\ 0 < d \wedge$
$(\forall g\ h.\ valid\text{-}path\ g \wedge valid\text{-}path\ h \wedge$
$(\forall t \in \{0..1\}.\ norm(g\ t - p\ t) < d \wedge norm(h\ t - p\ t) < d) \wedge$
$pathfinish\ g = pathstart\ g \wedge pathfinish\ h = pathstart\ h$
$\longrightarrow path\text{-}image\ g \subseteq s \wedge$
$path\text{-}image\ h \subseteq s \wedge$
$(\forall f.\ f\ holomorphic\text{-}on\ s$
$\longrightarrow contour\text{-}integral\ h\ f = contour\text{-}integral\ g\ f))$
⟨*proof*⟩

corollary *differentiable-polynomial-function*:
fixes *p* :: *real* ⇒ *'a*::*euclidean-space*
shows *polynomial-function p* ⟹ *p differentiable-on s*
⟨*proof*⟩

lemma *C1-differentiable-polynomial-function*:
fixes *p* :: *real* ⇒ *'a*::*euclidean-space*
shows *polynomial-function p* ⟹ *p C1-differentiable-on s*
⟨*proof*⟩

lemma *valid-path-polynomial-function*:
fixes *p* :: *real* ⇒ *'a*::*euclidean-space*
shows *polynomial-function p* ⟹ *valid-path p*
⟨*proof*⟩

lemma *valid-path-subpath-trivial* [*simp*]:
fixes *g* :: *real* ⇒ *'a*::*euclidean-space*
shows *z* ≠ *g x* ⟹ *valid-path* (*subpath x x g*)
⟨*proof*⟩

lemma *contour-integral-bound-exists*:
assumes *s*: *open s*

and *g*: *valid-path g*
and *pag*: *path-image g* $\subseteq$ *s*
shows $\exists L.\ 0 < L \wedge$
$(\forall f\ B.\ f$ *holomorphic-on s* $\wedge\ (\forall z \in s.\ norm(f\ z) \leq B)$
$\longrightarrow$ *norm*(*contour-integral g f*) $\leq L*B$)
⟨*proof*⟩

53.22 Constancy of a function from a connected set into a finite, disconnected or discrete set

Still missing: versions for a set that is smaller than R, or countable.

lemma *continuous-disconnected-range-constant*:
assumes *s*: *connected s*
and *conf*: *continuous-on s f*
and *fim*: *f* ' *s* $\subseteq$ *t*
and *cct*: $\bigwedge y.\ y \in t \Longrightarrow$ *connected-component-set t y* = $\{y\}$
shows $\exists a.\ \forall x \in s.\ f\ x = a$
⟨*proof*⟩

lemma *discrete-subset-disconnected*:
fixes *s* :: *'a*::*topological-space set*
fixes *t* :: *'b*::*real-normed-vector set*
assumes *conf*: *continuous-on s f*
and *no*: $\bigwedge x.\ x \in s \Longrightarrow \exists e{>}0.\ \forall y.\ y \in s \wedge f\ y \neq f\ x \longrightarrow e \leq norm\ (f\ y - f\ x)$
shows *f* ' *s* $\subseteq$ $\{y.$ *connected-component-set* (*f* ' *s*) *y* = $\{y\}\}$
⟨*proof*⟩

lemma *finite-implies-discrete*:
fixes *s* :: *'a*::*topological-space set*
assumes *finite* (*f* ' *s*)
shows $(\forall x \in s.\ \exists e{>}0.\ \forall y.\ y \in s \wedge f\ y \neq f\ x \longrightarrow e \leq norm\ (f\ y - f\ x))$
⟨*proof*⟩

This proof requires the existence of two separate values of the range type.

lemma *finite-range-constant-imp-connected*:
assumes $\bigwedge f$::*'a*::*topological-space* $\Rightarrow$ *'b*::*real-normed-algebra-1*.
$[\![$*continuous-on s f*; *finite*(*f* ' *s*)$]\!] \Longrightarrow \exists a.\ \forall x \in s.\ f\ x = a$
shows *connected s*
⟨*proof*⟩

lemma *continuous-disconnected-range-constant-eq*:
(*connected s* $\longleftrightarrow$
$(\forall f$::*'a*::*topological-space* $\Rightarrow$ *'b*::*real-normed-algebra-1*.
$\forall t.$ *continuous-on s f* $\wedge$ *f* ' *s* $\subseteq$ *t* $\wedge\ (\forall y \in t.$ *connected-component-set t y* = $\{y\}$)
$\longrightarrow\ (\exists a$::*'b*$.\ \forall x \in s.\ f\ x = a)))$ (**is** *?thesis1*)
and *continuous-discrete-range-constant-eq*:

(*connected s* ⟷
(∀ *f*::*'a*::*topological-space* ⇒ *'b*::*real-normed-algebra-1*.
continuous-on s f ∧
(∀ *x* ∈ *s*. ∃ *e*. *0* < *e* ∧ (∀ *y*. *y* ∈ *s* ∧ (*f y* ≠ *f x*) ⟶ *e* ≤ *norm*(*f y* − *f x*)))
⟶ (∃ *a*::*'b*. ∀ *x* ∈ *s*. *f x* = *a*))) (**is** *?thesis2*)
and *continuous-finite-range-constant-eq*:
(*connected s* ⟷
(∀ *f*::*'a*::*topological-space* ⇒ *'b*::*real-normed-algebra-1*.
continuous-on s f ∧ *finite* (*f* ‘ *s*)
⟶ (∃ *a*::*'b*. ∀ *x* ∈ *s*. *f x* = *a*))) (**is** *?thesis3*)
⟨*proof*⟩

lemma *continuous-discrete-range-constant*:
fixes *f* :: *'a*::*topological-space* ⇒ *'b*::*real-normed-algebra-1*
assumes *s*: *connected s*
and *continuous-on s f*
and ⋀*x*. *x* ∈ *s* ⟹ ∃ *e*>*0*. ∀ *y*. *y* ∈ *s* ∧ *f y* ≠ *f x* ⟶ *e* ≤ *norm* (*f y* − *f x*)
shows ∃ *a*. ∀ *x* ∈ *s*. *f x* = *a*
⟨*proof*⟩

lemma *continuous-finite-range-constant*:
fixes *f* :: *'a*::*topological-space* ⇒ *'b*::*real-normed-algebra-1*
assumes *connected s*
and *continuous-on s f*
and *finite* (*f* ‘ *s*)
shows ∃ *a*. ∀ *x* ∈ *s*. *f x* = *a*
⟨*proof*⟩

We can treat even non-rectifiable paths as having a "length" for bounds on analytic functions in open sets.

53.23 Winding Numbers

definition *winding-number*:: [*real* ⇒ *complex*, *complex*] ⇒ *complex* **where**
winding-number γ *z* ≡
@*n*. ∀ *e* > *0*. ∃ *p*. *valid-path p* ∧ *z* ∉ *path-image p* ∧
pathstart p = *pathstart* γ ∧
pathfinish p = *pathfinish* γ ∧
(∀ *t* ∈ {*0*..*1*}. *norm*(γ *t* − *p t*) < *e*) ∧
contour-integral p (λ*w*. *1*/(*w* − *z*)) = *2* ∗ *pi* ∗ *ii* ∗ *n*

lemma *winding-number*:
assumes *path* γ *z* ∉ *path-image* γ *0* < *e*
shows ∃ *p*. *valid-path p* ∧ *z* ∉ *path-image p* ∧
pathstart p = *pathstart* γ ∧
pathfinish p = *pathfinish* γ ∧
(∀ *t* ∈ {*0*..*1*}. *norm* (γ *t* − *p t*) < *e*) ∧
contour-integral p (λ*w*. *1*/(*w* − *z*)) = *2* ∗ *pi* ∗ *ii* ∗ *winding-number* γ
z

⟨*proof*⟩

lemma *winding-number-unique*:
assumes γ: *path* γ *z* $\notin$ *path-image* γ
and *pi*:
$\bigwedge e.\ e>0 \Longrightarrow \exists p.\ \textit{valid-path}\ p \wedge z \notin \textit{path-image}\ p \wedge$
pathstart p = pathstart $\gamma \wedge$ *pathfinish p = pathfinish* $\gamma \wedge$
$(\forall t \in \{0..1\}.\ \textit{norm}\ (\gamma\ t - p\ t) < e) \wedge$
contour-integral p $(\lambda w.\ 1/(w - z)) = 2 * pi * ii * n$
shows *winding-number* γ *z* = *n*
⟨*proof*⟩

lemma *winding-number-unique-loop*:
assumes γ: *path* γ *z* $\notin$ *path-image* γ
and *loop*: *pathfinish* γ = *pathstart* γ
and *pi*:
$\bigwedge e.\ e>0 \Longrightarrow \exists p.\ \textit{valid-path}\ p \wedge z \notin \textit{path-image}\ p \wedge$
pathfinish p = pathstart p $\wedge$
$(\forall t \in \{0..1\}.\ \textit{norm}\ (\gamma\ t - p\ t) < e) \wedge$
contour-integral p $(\lambda w.\ 1/(w - z)) = 2 * pi * ii * n$
shows *winding-number* γ *z* = *n*
⟨*proof*⟩

lemma *winding-number-valid-path*:
assumes *valid-path* γ *z* $\notin$ *path-image* γ
shows *winding-number* γ *z* = *1/(2*pi*ii)* * *contour-integral* γ $(\lambda w.\ 1/(w - z))$
⟨*proof*⟩

lemma *has-contour-integral-winding-number*:
assumes γ: *valid-path* γ *z* $\notin$ *path-image* γ
shows $((\lambda w.\ 1/(w - z))$ *has-contour-integral* *(2*pi*ii*winding-number* γ *z))* γ
⟨*proof*⟩

lemma *winding-number-trivial* [*simp*]: *z* $\neq$ *a* $\Longrightarrow$ *winding-number*(*linepath a a*) *z* = *0*
⟨*proof*⟩

lemma *winding-number-subpath-trivial* [*simp*]: *z* $\neq$ *g x* $\Longrightarrow$ *winding-number* (*subpath x x g*) *z* = *0*
⟨*proof*⟩

lemma *winding-number-join*:
assumes *g1*: *path g1 z* $\notin$ *path-image g1*
and *g2*: *path g2 z* $\notin$ *path-image g2*
and *pathfinish g1 = pathstart g2*
shows *winding-number*(*g1 +++ g2*) *z* = *winding-number g1 z* + *winding-number g2 z*

⟨*proof*⟩

lemma *winding-number-reversepath*:
assumes *path* γ *z* $\notin$ *path-image* γ
shows *winding-number*(*reversepath* γ) *z* = − (*winding-number* γ *z*)
⟨*proof*⟩

lemma *winding-number-shiftpath*:
assumes γ: *path* γ *z* $\notin$ *path-image* γ
and *pathfinish* γ = *pathstart* γ *a* ∈ {*0..1*}
shows *winding-number*(*shiftpath a* γ) *z* = *winding-number* γ *z*
⟨*proof*⟩

lemma *winding-number-split-linepath*:
assumes *c* ∈ *closed-segment a b z* $\notin$ *closed-segment a b*
shows *winding-number*(*linepath a b*) *z* = *winding-number*(*linepath a c*) *z* + *winding-number*(*linepath c b*) *z*
⟨*proof*⟩

lemma *winding-number-cong*:
(⋀*t*. ⟦*0* ≤ *t*; *t* ≤ *1*⟧ ⟹ *p t* = *q t*) ⟹ *winding-number p z* = *winding-number q z*
⟨*proof*⟩

lemma *winding-number-offset*: *winding-number p z* = *winding-number* (λ*w*. *p w* − *z*) *0*
⟨*proof*⟩

lemma *winding-number-join-pos-combined*:
⟦*valid-path* γ*1*; *z* $\notin$ *path-image* γ*1*; *0* < *Re*(*winding-number* γ*1 z*);
valid-path γ*2*; *z* $\notin$ *path-image* γ*2*; *0* < *Re*(*winding-number* γ*2 z*); *pathfinish* γ*1* = *pathstart* γ*2*⟧
⟹ *valid-path*(γ*1* +++ γ*2*) ∧ *z* $\notin$ *path-image*(γ*1* +++ γ*2*) ∧ *0* < *Re*(*winding-number*(γ*1* +++ γ*2*) *z*)
⟨*proof*⟩

lemma *Re-winding-number*:
⟦*valid-path* γ; *z* $\notin$ *path-image* γ⟧
⟹ *Re*(*winding-number* γ *z*) = *Im*(*contour-integral* γ (λ*w*. *1*/(*w* − *z*))) / (*2*∗*pi*)
⟨*proof*⟩

lemma *winding-number-pos-le*:
assumes γ: *valid-path* γ *z* $\notin$ *path-image* γ
and *ge*: ⋀*x*. ⟦*0* < *x*; *x* < *1*⟧ ⟹ *0* ≤ *Im* (*vector-derivative* γ (*at x*) ∗ *cnj*(γ

$x - z))$
shows $0 \leq Re(winding\text{-}number\ \gamma\ z)$
⟨*proof*⟩

lemma *winding-number-pos-lt-lemma*:
assumes γ: *valid-path* γ $z \notin$ *path-image* γ
and e: $0 < e$
and ge: $\bigwedge x.\ [\![0 < x; x < 1]\!] \Longrightarrow e \leq Im\ (vector\text{-}derivative\ \gamma\ (at\ x)\ /\ (\gamma\ x - z))$
shows $0 < Re(winding\text{-}number\ \gamma\ z)$
⟨*proof*⟩

lemma *winding-number-pos-lt*:
assumes γ: *valid-path* γ $z \notin$ *path-image* γ
and e: $0 < e$
and ge: $\bigwedge x.\ [\![0 < x; x < 1]\!] \Longrightarrow e \leq Im\ (vector\text{-}derivative\ \gamma\ (at\ x) * cnj(\gamma\ x - z))$
shows $0 < Re\ (winding\text{-}number\ \gamma\ z)$
⟨*proof*⟩

53.24 The winding number is an integer

Proof from the book Complex Analysis by Lars V. Ahlfors, Chapter 4, section 2.1, Also on page 134 of Serge Lang's book with the name title, etc.

lemma *exp-fg*:
fixes z::*complex*
assumes g: $(g\ has\text{-}vector\text{-}derivative\ g')\ (at\ x\ within\ s)$
and f: $(f\ has\text{-}vector\text{-}derivative\ (g'\ /\ (g\ x - z)))\ (at\ x\ within\ s)$
and z: $g\ x \neq z$
shows $((\lambda x.\ exp(-f\ x) * (g\ x - z))\ has\text{-}vector\text{-}derivative\ 0)\ (at\ x\ within\ s)$
⟨*proof*⟩

lemma *winding-number-exp-integral*:
fixes z::*complex*
assumes γ: γ *piecewise-C1-differentiable-on* $\{a..b\}$
and ab: $a \leq b$
and z: $z \notin \gamma$ ‘ $\{a..b\}$
shows $(\lambda x.\ vector\text{-}derivative\ \gamma\ (at\ x)\ /\ (\gamma\ x - z))\ integrable\text{-}on\ \{a..b\}$
(**is** *?thesis1*)
$exp\ (-\ (integral\ \{a..b\}\ (\lambda x.\ vector\text{-}derivative\ \gamma\ (at\ x)\ /\ (\gamma\ x - z)))) * (\gamma\ b - z) = \gamma\ a - z$
(**is** *?thesis2*)
⟨*proof*⟩

corollary *winding-number-exp-2pi*:
$[\![path\ p; z \notin path\text{-}image\ p]\!]$
$\Longrightarrow pathfinish\ p - z = exp\ (2 * pi * ii * winding\text{-}number\ p\ z) * (pathstart\ p - z)$
⟨*proof*⟩

53.25 The version with complex integers and equality

lemma *integer-winding-number-eq*:
assumes γ: *path* γ **and** z: $z \notin$ *path-image* γ
shows *winding-number* γ $z \in \mathbb{Z} \longleftrightarrow$ *pathfinish* γ = *pathstart* γ
⟨*proof*⟩

theorem *integer-winding-number*:
⟦*path* γ; *pathfinish* γ = *pathstart* γ; $z \notin$ *path-image* γ⟧ ⟹ *winding-number* γ z $\in \mathbb{Z}$
⟨*proof*⟩

If the winding number's magnitude is at least one, then the path must contain points in every direction.*) We can thus bound the winding number of a path that doesn't intersect a given ray.

lemma *winding-number-pos-meets*:
fixes z::*complex*
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** *1*: *Re* (*winding-number* γ z) $\geq$ *1*
and w: $w \neq z$
shows $\exists$ a::*real*. *0* < $a \wedge z + a*(w - z) \in$ *path-image* γ
⟨*proof*⟩

lemma *winding-number-big-meets*:
fixes z::*complex*
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** |*Re* (*winding-number* γ z)| $\geq$ *1*
and w: $w \neq z$
shows $\exists$ a::*real*. *0* < $a \wedge z + a*(w - z) \in$ *path-image* γ
⟨*proof*⟩

lemma *winding-number-less-1*:
fixes z::*complex*
shows
⟦*valid-path* γ; $z \notin$ *path-image* γ; $w \neq z$;
⋀a::*real*. *0* < a ⟹ $z + a*(w - z) \notin$ *path-image* γ⟧
⟹ |*Re*(*winding-number* γ z)| < *1*
⟨*proof*⟩

One way of proving that WN=1 for a loop.

lemma *winding-number-eq-1*:
fixes z::*complex*
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** *loop*: *pathfinish* γ = *pathstart* γ
and *0*: *0* < *Re*(*winding-number* γ z) **and** *2*: *Re*(*winding-number* γ z) < *2*
shows *winding-number* γ z = *1*
⟨*proof*⟩

53.26 Continuity of winding number and invariance on connected sets.

lemma *continuous-at-winding-number*:
fixes *z*::*complex*
assumes γ: *path* γ **and** *z*: $z \notin$ *path-image* γ
shows *continuous* (*at z*) (*winding-number* γ)
⟨*proof*⟩

corollary *continuous-on-winding-number*:
path $\gamma \Longrightarrow$ *continuous-on* $(-$ *path-image* $\gamma)$ $(\lambda w.$ *winding-number* γ $w)$
⟨*proof*⟩

53.27 The winding number is constant on a connected region

lemma *winding-number-constant*:
assumes γ: *path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ **and** *cs*: *connected s* **and** *sg*: $s \cap$ *path-image* $\gamma = \{\}$
shows $\exists k.\ \forall z \in s.$ *winding-number* γ $z = k$
⟨*proof*⟩

lemma *winding-number-eq*:
⟦*path* γ; *pathfinish* γ = *pathstart* γ; $w \in s$; $z \in s$; *connected s*; $s \cap$ *path-image* $\gamma = \{\}$⟧
$\Longrightarrow$ *winding-number* γ w = *winding-number* γ z
⟨*proof*⟩

lemma *open-winding-number-levelsets*:
assumes γ: *path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ
shows *open* $\{z.\ z \notin$ *path-image* $\gamma \wedge$ *winding-number* γ $z = k\}$
⟨*proof*⟩

53.28 Winding number is zero "outside" a curve, in various senses

lemma *winding-number-zero-in-outside*:
assumes γ: *path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ **and** *z*: $z \in$ *outside* (*path-image* γ)
shows *winding-number* γ $z = 0$
⟨*proof*⟩

lemma *winding-number-zero-outside*:
⟦*path* γ; *convex s*; *pathfinish* γ = *pathstart* γ; $z \notin s$; *path-image* $\gamma \subseteq s$⟧ $\Longrightarrow$ *winding-number* γ $z = 0$
⟨*proof*⟩

lemma *winding-number-zero-at-infinity*:
assumes γ: *path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ
shows $\exists B.\ \forall z.\ B \leq$ *norm* $z \longrightarrow$ *winding-number* γ $z = 0$
⟨*proof*⟩

lemma *winding-number-zero-point*:
⟦*path* γ; *convex* s; *pathfinish* γ = *pathstart* γ; *open* s; *path-image* $\gamma \subseteq s$⟧
$\Longrightarrow \exists z.\ z \in s \wedge$ *winding-number* $\gamma\ z = 0$
⟨*proof*⟩

If a path winds round a set, it winds rounds its inside.

lemma *winding-number-around-inside*:
assumes γ: *path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ
and *cls*: *closed* s **and** *cos*: *connected* s **and** *s-disj*: $s \cap$ *path-image* $\gamma = \{\}$
and z: $z \in s$ **and** *wn-nz*: *winding-number* $\gamma\ z \neq 0$ **and** w: $w \in s \cup$ *inside* s
shows *winding-number* $\gamma\ w$ = *winding-number* $\gamma\ z$
⟨*proof*⟩

Bounding a WN by 1/2 for a path and point in opposite halfspaces.

lemma *winding-number-subpath-continuous*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ
shows *continuous-on* $\{0..1\}$ $(\lambda x.$ *winding-number*(*subpath* 0 x γ) $z)$
⟨*proof*⟩

lemma *winding-number-ivt-pos*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** $0 \leq w$ $w \leq$ *Re*(*winding-number* γ z)
shows $\exists t \in \{0..1\}$. *Re*(*winding-number*(*subpath* 0 t γ) z) = w
⟨*proof*⟩

lemma *winding-number-ivt-neg*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** *Re*(*winding-number* γ z) $\leq w$ $w \leq 0$
shows $\exists t \in \{0..1\}$. *Re*(*winding-number*(*subpath* 0 t γ) z) = w
⟨*proof*⟩

lemma *winding-number-ivt-abs*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** $0 \leq w$ $w \leq |$*Re*(*winding-number* γ z)|
shows $\exists t \in \{0..1\}$. $|$*Re* (*winding-number* (*subpath* 0 t γ) z)| = w
⟨*proof*⟩

lemma *winding-number-lt-half-lemma*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ **and** *az*: $a \cdot z \leq b$ **and** *pag*: *path-image* $\gamma \subseteq \{w.\ a \cdot w > b\}$
shows *Re*(*winding-number* γ z) < 1/2
⟨*proof*⟩

lemma *winding-number-lt-half*:
assumes *valid-path* γ $a \cdot z \leq b$ *path-image* $\gamma \subseteq \{w.\ a \cdot w > b\}$
shows $|$*Re* (*winding-number* γ z)| < 1/2
⟨*proof*⟩

lemma *winding-number-le-half*:
assumes γ: *valid-path* γ **and** z: $z \notin$ *path-image* γ
and *anz*: $a \neq 0$ **and** *azb*: $a \cdot z \leq b$ **and** *pag*: *path-image* $\gamma \subseteq \{w.\ a \cdot w \geq b\}$
shows $|Re\ (winding\text{-}number\ \gamma\ z)| \leq 1/2$
⟨*proof*⟩

lemma *winding-number-lt-half-linepath*: $z \notin$ *closed-segment a b* $\Longrightarrow |Re\ (winding\text{-}number\ (linepath\ a\ b)\ z)| < 1/2$
⟨*proof*⟩

Positivity of WN for a linepath.

lemma *winding-number-linepath-pos-lt*:
assumes $0 < Im\ ((b - a) * cnj\ (b - z))$
shows $0 < Re(winding\text{-}number(linepath\ a\ b)\ z)$
⟨*proof*⟩

53.29 Cauchy's integral formula, again for a convex enclosing set.

lemma *Cauchy-integral-formula-weak*:
assumes s: *convex* s **and** *finite* k **and** *conf*: *continuous-on* s f
and *fcd*: $(\bigwedge x.\ x \in interior\ s - k \Longrightarrow f$ *field-differentiable at* $x)$
and z: $z \in interior\ s - k$ **and** *vpg*: *valid-path* γ
and *pasz*: *path-image* $\gamma \subseteq s - \{z\}$ **and** *loop*: *pathfinish* γ = *pathstart* γ
shows $((\lambda w.\ f\ w\ /\ (w - z))$ *has-contour-integral* $(2{*}pi * ii * winding\text{-}number\ \gamma\ z * f\ z))\ \gamma$
⟨*proof*⟩

theorem *Cauchy-integral-formula-convex-simple*:
⟦*convex* s; f *holomorphic-on* s; $z \in interior\ s$; *valid-path* γ; *path-image* $\gamma \subseteq s - \{z\}$;
pathfinish γ = *pathstart* γ⟧
$\Longrightarrow ((\lambda w.\ f\ w\ /\ (w - z))$ *has-contour-integral* $(2{*}pi * ii * winding\text{-}number\ \gamma\ z * f\ z))\ \gamma$
⟨*proof*⟩

53.30 Homotopy forms of Cauchy's theorem

proposition *Cauchy-theorem-homotopic*:
assumes *hom*: *if atends then homotopic-paths s g h else homotopic-loops s g h*
and *open s* **and** f: f *holomorphic-on s*
and *vpg*: *valid-path g* **and** *vph*: *valid-path h*
shows *contour-integral g f* = *contour-integral h f*
⟨*proof*⟩

proposition *Cauchy-theorem-homotopic-paths*:
assumes *hom*: *homotopic-paths s g h*
and *open s* **and** f: f *holomorphic-on s*
and *vpg*: *valid-path g* **and** *vph*: *valid-path h*

shows *contour-integral g f = contour-integral h f*
⟨*proof*⟩

proposition *Cauchy-theorem-homotopic-loops*:
assumes *hom*: *homotopic-loops s g h*
and *open s* **and** *f*: *f holomorphic-on s*
and *vpg*: *valid-path g* **and** *vph*: *valid-path h*
shows *contour-integral g f = contour-integral h f*
⟨*proof*⟩

lemma *has-contour-integral-newpath*:
⟦(*f has-contour-integral y*) *h*; *f contour-integrable-on g*; *contour-integral g f = contour-integral h f*⟧
⟹ (*f has-contour-integral y*) *g*
⟨*proof*⟩

lemma *Cauchy-theorem-null-homotopic*:
⟦*f holomorphic-on s*; *open s*; *valid-path g*; *homotopic-loops s g* (*linepath a a*)⟧
⟹ (*f has-contour-integral 0*) *g*
⟨*proof*⟩

53.31 More winding number properties

including the fact that it's +-1 inside a simple closed curve.

lemma *winding-number-homotopic-paths*:
assumes *homotopic-paths* (−{*z*}) *g h*
shows *winding-number g z = winding-number h z*
⟨*proof*⟩

lemma *winding-number-homotopic-loops*:
assumes *homotopic-loops* (−{*z*}) *g h*
shows *winding-number g z = winding-number h z*
⟨*proof*⟩

lemma *winding-number-paths-linear-eq*:
⟦*path g*; *path h*; *pathstart h = pathstart g*; *pathfinish h = pathfinish g*;
⋀*t*. *t* ∈ {*0..1*} ⟹ *z* ∉ *closed-segment* (*g t*) (*h t*)⟧
⟹ *winding-number h z = winding-number g z*
⟨*proof*⟩

lemma *winding-number-loops-linear-eq*:
⟦*path g*; *path h*; *pathfinish g = pathstart g*; *pathfinish h = pathstart h*;
⋀*t*. *t* ∈ {*0..1*} ⟹ *z* ∉ *closed-segment* (*g t*) (*h t*)⟧
⟹ *winding-number h z = winding-number g z*
⟨*proof*⟩

lemma *winding-number-nearby-paths-eq*:
⟦*path g*; *path h*;
pathstart h = pathstart g; *pathfinish h = pathfinish g*;

$\bigwedge t.\ t \in \{0..1\} \Longrightarrow norm(h\ t - g\ t) < norm(g\ t - z)]\!]$
$\Longrightarrow$ *winding-number h z = winding-number g z*
⟨*proof*⟩

lemma *winding-number-nearby-loops-eq*:
$[\![$*path g*; *path h*;
pathfinish g = pathstart g;
pathfinish h = pathstart h;
$\bigwedge t.\ t \in \{0..1\} \Longrightarrow norm(h\ t - g\ t) < norm(g\ t - z)]\!]$
$\Longrightarrow$ *winding-number h z = winding-number g z*
⟨*proof*⟩

proposition *winding-number-subpath-combine*:
$[\![$*path g*; $z \notin$ *path-image g*;
$u \in \{0..1\}$; $v \in \{0..1\}$; $w \in \{0..1\}]\!]$
$\Longrightarrow$ *winding-number (subpath u v g) z + winding-number (subpath v w g) z =*
winding-number (subpath u w g) z
⟨*proof*⟩

53.32 Partial circle path

definition *part-circlepath* :: [*complex*, *real*, *real*, *real*, *real*] ⇒ *complex*
where *part-circlepath z r s t* ≡ λ*x*. *z + of-real r * exp (ii * of-real (linepath s t x))*

lemma *pathstart-part-circlepath* [*simp*]:
pathstart(*part-circlepath z r s t*) = *z + r*exp*(*ii * s*)
⟨*proof*⟩

lemma *pathfinish-part-circlepath* [*simp*]:
pathfinish(*part-circlepath z r s t*) = *z + r*exp*(*ii*t*)
⟨*proof*⟩

proposition *has-vector-derivative-part-circlepath* [*derivative-intros*]:
((*part-circlepath z r s t*) *has-vector-derivative*
(*ii * r * (of-real t − of-real s*) * *exp*(*ii * linepath s t x*)))
(*at x within X*)
⟨*proof*⟩

corollary *vector-derivative-part-circlepath*:
vector-derivative (*part-circlepath z r s t*) (*at x*) =
*ii * r * (of-real t − of-real s*) * *exp*(*ii * linepath s t x*)
⟨*proof*⟩

corollary *vector-derivative-part-circlepath01*:
$[\![0 \le x; x \le 1]\!]$
$\Longrightarrow$ *vector-derivative* (*part-circlepath z r s t*) (*at x within* $\{0..1\}$) =
*ii * r * (of-real t − of-real s*) * *exp*(*ii * linepath s t x*)

⟨*proof*⟩

lemma *valid-path-part-circlepath* [*simp*]: *valid-path* (*part-circlepath z r s t*)
⟨*proof*⟩

lemma *path-part-circlepath* [*simp*]: *path* (*part-circlepath z r s t*)
⟨*proof*⟩

proposition *path-image-part-circlepath*:
assumes $s \leq t$
shows *path-image* (*part-circlepath z r s t*) = $\{z + r * exp(ii * of\text{-}real\ x) \mid x.\ s \leq x \wedge x \leq t\}$
⟨*proof*⟩

corollary *path-image-part-circlepath-subset*:
$[\![s \leq t;\ 0 \leq r]\!] \Longrightarrow path\text{-}image(part\text{-}circlepath\ z\ r\ s\ t) \subseteq sphere\ z\ r$
⟨*proof*⟩

proposition *in-path-image-part-circlepath*:
assumes $w \in path\text{-}image(part\text{-}circlepath\ z\ r\ s\ t)$ $s \leq t$ $0 \leq r$
shows $norm(w - z) = r$
⟨*proof*⟩

proposition *finite-bounded-log*: *finite* $\{z{::}complex.\ norm\ z \leq b \wedge exp\ z = w\}$
⟨*proof*⟩

lemma *finite-bounded-log2*:
fixes *a*::*complex*
assumes $a \neq 0$
shows *finite* $\{z.\ norm\ z \leq b \wedge exp(a*z) = w\}$
⟨*proof*⟩

proposition *has-contour-integral-bound-part-circlepath-strong*:
assumes *fi*: (*f has-contour-integral i*) (*part-circlepath z r s t*)
and *finite k* **and** *le*: $0 \leq B$ $0 < r$ $s \leq t$
and *B*: $\bigwedge x.\ x \in path\text{-}image(part\text{-}circlepath\ z\ r\ s\ t) - k \Longrightarrow norm(f\ x) \leq B$
shows $cmod\ i \leq B * r * (t - s)$
⟨*proof*⟩

corollary *has-contour-integral-bound-part-circlepath*:
$[\![(f\ has\text{-}contour\text{-}integral\ i)\ (part\text{-}circlepath\ z\ r\ s\ t);$
$0 \leq B;\ 0 < r;\ s \leq t;$
$\bigwedge x.\ x \in path\text{-}image(part\text{-}circlepath\ z\ r\ s\ t) \Longrightarrow norm(f\ x) \leq B]\!]$
$\Longrightarrow norm\ i \leq B*r*(t - s)$
⟨*proof*⟩

proposition *contour-integrable-continuous-part-circlepath*:
continuous-on (*path-image* (*part-circlepath z r s t*)) *f*
$\Longrightarrow$ *f contour-integrable-on* (*part-circlepath z r s t*)

⟨*proof*⟩

proposition *winding-number-part-circlepath-pos-less*:
assumes $s < t$ **and** *no*: $norm(w - z) < r$
shows $0 < Re\ (winding\text{-}number(part\text{-}circlepath\ z\ r\ s\ t)\ w)$
⟨*proof*⟩

proposition *simple-path-part-circlepath*:
$simple\text{-}path(part\text{-}circlepath\ z\ r\ s\ t) \longleftrightarrow (r \neq 0 \wedge s \neq t \wedge |s - t| \leq 2{*}pi)$
⟨*proof*⟩

proposition *arc-part-circlepath*:
assumes $r \neq 0\ s \neq t\ |s - t| < 2{*}pi$
shows $arc\ (part\text{-}circlepath\ z\ r\ s\ t)$
⟨*proof*⟩

53.33 Special case of one complete circle

definition *circlepath* :: $[complex, real, real] \Rightarrow complex$
where $circlepath\ z\ r \equiv part\text{-}circlepath\ z\ r\ 0\ (2{*}pi)$

lemma *circlepath*: $circlepath\ z\ r = (\lambda x.\ z + r * exp(2 * of\text{-}real\ pi * ii * of\text{-}real\ x))$
⟨*proof*⟩

lemma *pathstart-circlepath* [*simp*]: $pathstart\ (circlepath\ z\ r) = z + r$
⟨*proof*⟩

lemma *pathfinish-circlepath* [*simp*]: $pathfinish\ (circlepath\ z\ r) = z + r$
⟨*proof*⟩

lemma *circlepath-minus*: $circlepath\ z\ (-r)\ x = circlepath\ z\ r\ (x + 1/2)$
⟨*proof*⟩

lemma *circlepath-add1*: $circlepath\ z\ r\ (x+1) = circlepath\ z\ r\ x$
⟨*proof*⟩

lemma *circlepath-add-half*: $circlepath\ z\ r\ (x + 1/2) = circlepath\ z\ r\ (x - 1/2)$
⟨*proof*⟩

lemma *path-image-circlepath-minus-subset*:
$path\text{-}image\ (circlepath\ z\ (-r)) \subseteq path\text{-}image\ (circlepath\ z\ r)$
⟨*proof*⟩

lemma *path-image-circlepath-minus*: $path\text{-}image\ (circlepath\ z\ (-r)) = path\text{-}image\ (circlepath\ z\ r)$
⟨*proof*⟩

proposition *has-vector-derivative-circlepath* [*derivative-intros*]:

((*circlepath z r*) *has-vector-derivative* (*2 * pi * ii * r * exp* (*2 * of-real pi * ii * of-real x*)))
(*at x within X*)
⟨*proof*⟩

corollary *vector-derivative-circlepath*:
vector-derivative (*circlepath z r*) (*at x*) =
*2 * pi * ii * r * exp*(*2 * of-real pi * ii * x*)
⟨*proof*⟩

corollary *vector-derivative-circlepath01*:
⟦*0 ≤ x*; *x ≤ 1*⟧
⟹ *vector-derivative* (*circlepath z r*) (*at x within* {*0..1*}) =
*2 * pi * ii * r * exp*(*2 * of-real pi * ii * x*)
⟨*proof*⟩

lemma *valid-path-circlepath* [*simp*]: *valid-path* (*circlepath z r*)
⟨*proof*⟩

lemma *path-circlepath* [*simp*]: *path* (*circlepath z r*)
⟨*proof*⟩

lemma *path-image-circlepath-nonneg*:
assumes *0 ≤ r* **shows** *path-image* (*circlepath z r*) = *sphere z r*
⟨*proof*⟩

proposition *path-image-circlepath* [*simp*]:
path-image (*circlepath z r*) = *sphere z* |*r*|
⟨*proof*⟩

lemma *has-contour-integral-bound-circlepath-strong*:
⟦(*f has-contour-integral i*) (*circlepath z r*);
finite k; *0 ≤ B*; *0 < r*;
⋀*x*. ⟦*norm*(*x − z*) = *r*; *x ∉ k*⟧ ⟹ *norm*(*f x*) ≤ *B*⟧
⟹ *norm i* ≤ *B*∗(*2*∗*pi*∗*r*)
⟨*proof*⟩

corollary *has-contour-integral-bound-circlepath*:
⟦(*f has-contour-integral i*) (*circlepath z r*);
0 ≤ B; *0 < r*; ⋀*x*. *norm*(*x − z*) = *r* ⟹ *norm*(*f x*) ≤ *B*⟧
⟹ *norm i* ≤ *B*∗(*2*∗*pi*∗*r*)
⟨*proof*⟩

proposition *contour-integrable-continuous-circlepath*:
continuous-on (*path-image* (*circlepath z r*)) *f*
⟹ *f contour-integrable-on* (*circlepath z r*)
⟨*proof*⟩

lemma *simple-path-circlepath*: *simple-path*(*circlepath z r*) ⟷ (*r ≠ 0*)

⟨*proof*⟩

lemma *notin-path-image-circlepath* [*simp*]: *cmod* $(w - z) < r \Longrightarrow w \notin$ *path-image* (*circlepath z r*)
⟨*proof*⟩

proposition *contour-integral-circlepath*:
$0 < r \Longrightarrow$ *contour-integral* (*circlepath z r*) ($\lambda w.$ *1 / (w − z)*) = *2 * complex-of-real pi ** i
⟨*proof*⟩

lemma *winding-number-circlepath-centre*: $0 < r \Longrightarrow$ *winding-number* (*circlepath z r*) *z = 1*
⟨*proof*⟩

proposition *winding-number-circlepath*:
assumes *norm*$(w - z) < r$ **shows** *winding-number*(*circlepath z r*) *w = 1*
⟨*proof*⟩

Hence the Cauchy formula for points inside a circle.

theorem *Cauchy-integral-circlepath*:
assumes *continuous-on* (*cball z r*) *f f holomorphic-on* (*ball z r*) *norm*$(w - z) <$ *r*
shows (($\lambda u.$ *f u*/(*u − w*)) *has-contour-integral* (*2 * of-real pi * ii * f w*))
(*circlepath z r*)
⟨*proof*⟩

corollary *Cauchy-integral-circlepath-simple*:
assumes *f holomorphic-on cball z r norm*$(w - z) < r$
shows (($\lambda u.$ *f u*/(*u − w*)) *has-contour-integral* (*2 * of-real pi * ii * f w*))
(*circlepath z r*)
⟨*proof*⟩

lemma *no-bounded-connected-component-imp-winding-number-zero*:
assumes *g*: *path g path-image g* $\subseteq$ *s pathfinish g = pathstart g z* $\notin$ *s*
and *nb*: $\bigwedge z.$ *bounded* (*connected-component-set* (*− s*) *z*) $\longrightarrow z \in s$
shows *winding-number g z = 0*
⟨*proof*⟩

lemma *no-bounded-path-component-imp-winding-number-zero*:
assumes *g*: *path g path-image g* $\subseteq$ *s pathfinish g = pathstart g z* $\notin$ *s*
and *nb*: $\bigwedge z.$ *bounded* (*path-component-set* (*− s*) *z*) $\longrightarrow z \in s$
shows *winding-number g z = 0*
⟨*proof*⟩

53.34 Uniform convergence of path integral

Uniform convergence when the derivative of the path is bounded, and in particular for the special case of a circle.

proposition *contour-integral-uniform-limit*:
assumes *ev-fint*: *eventually* $(\lambda n::'a.\ (f\ n)\ \textit{contour-integrable-on}\ \gamma)\ F$
and *ev-no*: $\bigwedge e.\ 0 < e \Longrightarrow \textit{eventually}\ (\lambda n.\ \forall x \in \textit{path-image}\ \gamma.\ \textit{norm}(f\ n\ x - l\ x) < e)\ F$
and *noleB*: $\bigwedge t.\ t \in \{0..1\} \Longrightarrow \textit{norm}\ (\textit{vector-derivative}\ \gamma\ (\textit{at}\ t)) \leq B$
and γ: *valid-path* γ
and [*simp*]: $\sim$ (*trivial-limit* F)
shows l *contour-integrable-on* γ $((\lambda n.\ \textit{contour-integral}\ \gamma\ (f\ n)) \longrightarrow \textit{contour-integral}\ \gamma\ l)\ F$
⟨*proof*⟩

proposition *contour-integral-uniform-limit-circlepath*:
assumes *ev-fint*: *eventually* $(\lambda n::'a.\ (f\ n)\ \textit{contour-integrable-on}\ (\textit{circlepath}\ z\ r))\ F$
and *ev-no*: $\bigwedge e.\ 0 < e \Longrightarrow \textit{eventually}\ (\lambda n.\ \forall x \in \textit{path-image}\ (\textit{circlepath}\ z\ r).\ \textit{norm}(f\ n\ x - l\ x) < e)\ F$
and [*simp*]: $\sim$ (*trivial-limit* F) $0 < r$
shows l *contour-integrable-on* $(\textit{circlepath}\ z\ r)$ $((\lambda n.\ \textit{contour-integral}\ (\textit{circlepath}\ z\ r)\ (f\ n)) \longrightarrow \textit{contour-integral}\ (\textit{circlepath}\ z\ r)\ l)\ F$
⟨*proof*⟩

53.35 General stepping result for derivative formulas.

lemma *sum-sqs-eq*:
fixes $x::'a::\textit{idom}$ **shows** $x * x + y * y = x * (y * 2) \Longrightarrow y = x$
⟨*proof*⟩

proposition *Cauchy-next-derivative*:
assumes *continuous-on* $(\textit{path-image}\ \gamma)\ f'$
and *leB*: $\bigwedge t.\ t \in \{0..1\} \Longrightarrow \textit{norm}\ (\textit{vector-derivative}\ \gamma\ (\textit{at}\ t)) \leq B$
and *int*: $\bigwedge w.\ w \in s - \textit{path-image}\ \gamma \Longrightarrow ((\lambda u.\ f'\ u\ /\ (u - w)\ \hat{}\ k)\ \textit{has-contour-integral}\ f\ w)\ \gamma$
and k: $k \neq 0$
and *open* s
and γ: *valid-path* γ
and w: $w \in s - \textit{path-image}\ \gamma$
shows $(\lambda u.\ f'\ u\ /\ (u - w)\ \hat{}\ (\textit{Suc}\ k))\ \textit{contour-integrable-on}\ \gamma$
and $(f\ \textit{has-field-derivative}\ (k * \textit{contour-integral}\ \gamma\ (\lambda u.\ f'\ u/(u - w)\ \hat{}\ (\textit{Suc}\ k))))$
$(\textit{at}\ w)$ (**is** *?thes2*)
⟨*proof*⟩

corollary *Cauchy-next-derivative-circlepath*:
assumes *contf*: *continuous-on* $(\textit{path-image}\ (\textit{circlepath}\ z\ r))\ f$
and *int*: $\bigwedge w.\ w \in \textit{ball}\ z\ r \Longrightarrow ((\lambda u.\ f\ u\ /\ (u - w)\ \hat{}\ k)\ \textit{has-contour-integral}\ g$

w) (*circlepath z r*)
and *k*: $k \neq 0$
and *w*: $w \in ball\ z\ r$
shows ($\lambda u.\ f\ u\ /\ (u - w)\ \hat{}\ (Suc\ k)$) *contour-integrable-on* (*circlepath z r*)
(**is** *?thes1*)
and (*g has-field-derivative* ($k * contour\text{-}integral$ (*circlepath z r*) ($\lambda u.\ f\ u/(u - w)\ \hat{}\ (Suc\ k)$))) (*at w*)
(**is** *?thes2*)
⟨*proof*⟩

In particular, the first derivative formula.

proposition *Cauchy-derivative-integral-circlepath*:
assumes *contf*: *continuous-on* (*cball z r*) *f*
and *holf*: *f holomorphic-on ball z r*
and *w*: $w \in ball\ z\ r$
shows ($\lambda u.\ f\ u/(u - w)\ \hat{}\ 2$) *contour-integrable-on* (*circlepath z r*)
(**is** *?thes1*)
and (*f has-field-derivative* ($1\ /\ (2 * of\text{-}real\ pi * ii) * contour\text{-}integral$(*circlepath z r*) ($\lambda u.\ f\ u\ /\ (u - w)\ \hat{}\ 2$))) (*at w*)
(**is** *?thes2*)
⟨*proof*⟩

53.36 Existence of all higher derivatives.

proposition *derivative-is-holomorphic*:
assumes *open s*
and *fder*: $\bigwedge z.\ z \in s \Longrightarrow$ (*f has-field-derivative f' z*) (*at z*)
shows *f' holomorphic-on s*
⟨*proof*⟩

lemma *holomorphic-deriv* [*holomorphic-intros*]:
⟦*f holomorphic-on s*; *open s*⟧ $\Longrightarrow$ (*deriv f*) *holomorphic-on s*
⟨*proof*⟩

lemma *analytic-deriv*: *f analytic-on s* $\Longrightarrow$ (*deriv f*) *analytic-on s*
⟨*proof*⟩

lemma *holomorphic-higher-deriv* [*holomorphic-intros*]: ⟦*f holomorphic-on s*; *open s*⟧ $\Longrightarrow$ (*deriv* ^^ *n*) *f holomorphic-on s*
⟨*proof*⟩

lemma *analytic-higher-deriv*: *f analytic-on s* $\Longrightarrow$ (*deriv* ^^ *n*) *f analytic-on s*
⟨*proof*⟩

lemma *has-field-derivative-higher-deriv*:
⟦*f holomorphic-on s*; *open s*; $x \in s$⟧
$\Longrightarrow$ ((*deriv* ^^ *n*) *f has-field-derivative* (*deriv* ^^ (*Suc n*)) *f x*) (*at x*)
⟨*proof*⟩

lemma *valid-path-compose-holomorphic*:
assumes *valid-path g* **and** *holo:f holomorphic-on s* **and** *open s path-image g* $\subseteq$ *s*
shows *valid-path* (*f o g*)
⟨*proof*⟩

53.37 Morera's theorem.

lemma *Morera-local-triangle-ball*:
assumes $\bigwedge z.\ z \in s$
$\Longrightarrow \exists e\ a.\ 0 < e \wedge z \in ball\ a\ e \wedge continuous\text{-}on\ (ball\ a\ e)\ f \wedge$
$(\forall b\ c.\ closed\text{-}segment\ b\ c \subseteq ball\ a\ e$
$\longrightarrow contour\text{-}integral\ (linepath\ a\ b)\ f +$
$contour\text{-}integral\ (linepath\ b\ c)\ f +$
$contour\text{-}integral\ (linepath\ c\ a)\ f = 0)$
shows *f analytic-on s*
⟨*proof*⟩

lemma *Morera-local-triangle*:
assumes $\bigwedge z.\ z \in s$
$\Longrightarrow \exists t.\ open\ t \wedge z \in t \wedge continuous\text{-}on\ t\ f \wedge$
$(\forall a\ b\ c.\ convex\ hull\ \{a,b,c\} \subseteq t$
$\longrightarrow contour\text{-}integral\ (linepath\ a\ b)\ f +$
$contour\text{-}integral\ (linepath\ b\ c)\ f +$
$contour\text{-}integral\ (linepath\ c\ a)\ f = 0)$
shows *f analytic-on s*
⟨*proof*⟩

proposition *Morera-triangle*:
$[\![continuous\text{-}on\ s\ f;\ open\ s;$
$\bigwedge a\ b\ c.\ convex\ hull\ \{a,b,c\} \subseteq s$
$\longrightarrow contour\text{-}integral\ (linepath\ a\ b)\ f +$
$contour\text{-}integral\ (linepath\ b\ c)\ f +$
$contour\text{-}integral\ (linepath\ c\ a)\ f = 0]\!]$
$\Longrightarrow f\ analytic\text{-}on\ s$
⟨*proof*⟩

53.38 Combining theorems for higher derivatives including Leibniz rule.

lemma *higher-deriv-linear* [*simp*]:
$(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ c*w) = (\lambda z.\ if\ n = 0\ then\ c*z\ else\ if\ n = 1\ then\ c\ else\ 0)$
⟨*proof*⟩

lemma *higher-deriv-const* [*simp*]: $(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ c) = (\lambda w.\ if\ n{=}0\ then\ c\ else\ 0)$
⟨*proof*⟩

lemma *higher-deriv-ident* [*simp*]:
$(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ w)\ z = (if\ n = 0\ then\ z\ else\ if\ n = 1\ then\ 1\ else\ 0)$

⟨*proof*⟩

corollary *higher-deriv-id* [*simp*]:
(*deriv* ^^ *n*) *id z* = (*if n* = *0 then z else if n* = *1 then 1 else 0*)
⟨*proof*⟩

lemma *has-complex-derivative-funpow-1*:
⟦(*f has-field-derivative 1*) (*at z*); *f z* = *z*⟧ ⟹ (*f*^^*n has-field-derivative 1*) (*at z*)
⟨*proof*⟩

proposition *higher-deriv-uminus*:
assumes *f holomorphic-on s open s* **and** *z*: *z* ∈ *s*
shows (*deriv* ^^ *n*) (λ*w*. −(*f w*)) *z* = − ((*deriv* ^^ *n*) *f z*)
⟨*proof*⟩

proposition *higher-deriv-add*:
fixes *z*::*complex*
assumes *f holomorphic-on s g holomorphic-on s open s* **and** *z*: *z* ∈ *s*
shows (*deriv* ^^ *n*) (λ*w*. *f w* + *g w*) *z* = (*deriv* ^^ *n*) *f z* + (*deriv* ^^ *n*) *g z*
⟨*proof*⟩

corollary *higher-deriv-diff*:
fixes *z*::*complex*
assumes *f holomorphic-on s g holomorphic-on s open s* **and** *z*: *z* ∈ *s*
shows (*deriv* ^^ *n*) (λ*w*. *f w* − *g w*) *z* = (*deriv* ^^ *n*) *f z* − (*deriv* ^^ *n*) *g z*
⟨*proof*⟩

lemma *bb*: *Suc n choose k* = (*n choose k*) + (*if k* = *0 then 0 else* (*n choose* (*k* − *1*)))
⟨*proof*⟩

proposition *higher-deriv-mult*:
fixes *z*::*complex*
assumes *f holomorphic-on s g holomorphic-on s open s* **and** *z*: *z* ∈ *s*
shows (*deriv* ^^ *n*) (λ*w*. *f w* ∗ *g w*) *z* =
(∑ *i* = *0*..*n*. *of-nat* (*n choose i*) ∗ (*deriv* ^^ *i*) *f z* ∗ (*deriv* ^^ (*n* − *i*)) *g z*)
⟨*proof*⟩

proposition *higher-deriv-transform-within-open*:
fixes *z*::*complex*
assumes *f holomorphic-on s g holomorphic-on s open s* **and** *z*: *z* ∈ *s*
and *fg*: ⋀*w*. *w* ∈ *s* ⟹ *f w* = *g w*
shows (*deriv* ^^ *i*) *f z* = (*deriv* ^^ *i*) *g z*
⟨*proof*⟩

proposition *higher-deriv-compose-linear*:
fixes *z*::*complex*
assumes *f*: *f holomorphic-on t* **and** *s*: *open s* **and** *t*: *open t* **and** *z*: $z \in s$
and *fg*: $\bigwedge w.\ w \in s \Longrightarrow u * w \in t$
shows $(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ f\ (u * w))\ z = u\hat{}n * (deriv\ \hat{}\hat{}\ n)\ f\ (u * z)$
⟨*proof*⟩

lemma *higher-deriv-add-at*:
assumes *f analytic-on* $\{z\}$ *g analytic-on* $\{z\}$
shows $(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ f\ w + g\ w)\ z = (deriv\ \hat{}\hat{}\ n)\ f\ z + (deriv\ \hat{}\hat{}\ n)\ g\ z$
⟨*proof*⟩

lemma *higher-deriv-diff-at*:
assumes *f analytic-on* $\{z\}$ *g analytic-on* $\{z\}$
shows $(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ f\ w - g\ w)\ z = (deriv\ \hat{}\hat{}\ n)\ f\ z - (deriv\ \hat{}\hat{}\ n)\ g\ z$
⟨*proof*⟩

lemma *higher-deriv-uminus-at*:
f analytic-on $\{z\} \Longrightarrow (deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ -(f\ w))\ z = -\ ((deriv\ \hat{}\hat{}\ n)\ f\ z)$
⟨*proof*⟩

lemma *higher-deriv-mult-at*:
assumes *f analytic-on* $\{z\}$ *g analytic-on* $\{z\}$
shows $(deriv\ \hat{}\hat{}\ n)\ (\lambda w.\ f\ w * g\ w)\ z =$
$(\sum i = 0..n.\ of\text{-}nat\ (n\ choose\ i) * (deriv\ \hat{}\hat{}\ i)\ f\ z * (deriv\ \hat{}\hat{}\ (n - i))\ g\ z)$
⟨*proof*⟩

Nonexistence of isolated singularities and a stronger integral formula.

proposition *no-isolated-singularity*:
fixes *z*::*complex*
assumes *f*: *continuous-on s f* **and** *holf*: *f holomorphic-on* $(s - k)$ **and** *s*: *open s* **and** *k*: *finite k*
shows *f holomorphic-on s*
⟨*proof*⟩

proposition *Cauchy-integral-formula-convex*:
assumes *s*: *convex s* **and** *k*: *finite k* **and** *contf*: *continuous-on s f*
and *fcd*: $(\bigwedge x.\ x \in interior\ s - k \Longrightarrow f\ field\text{-}differentiable\ at\ x)$
and *z*: $z \in interior\ s$ **and** *vpg*: *valid-path* γ
and *pasz*: *path-image* $\gamma \subseteq s - \{z\}$ **and** *loop*: *pathfinish* γ = *pathstart* γ
shows $((\lambda w.\ f\ w\ /\ (w - z))\ has\text{-}contour\text{-}integral\ (2*pi * ii * winding\text{-}number\ \gamma\ z * f\ z))\ \gamma$
⟨*proof*⟩

Formula for higher derivatives.

proposition *Cauchy-has-contour-integral-higher-derivative-circlepath*:
assumes *contf*: *continuous-on* $(cball\ z\ r)$ *f*
and *holf*: *f holomorphic-on ball z r*

and w: $w \in ball\ z\ r$
shows $((\lambda u.\ f\ u\ /\ (u - w)\ \hat{}\ (Suc\ k))\ has\text{-}contour\text{-}integral\ ((2 * pi * ii)\ /\ (fact\ k) * (deriv\ \hat{}\hat{}\ k)\ f\ w))$
$(circlepath\ z\ r)$
⟨*proof*⟩

proposition *Cauchy-higher-derivative-integral-circlepath*:
assumes *contf*: *continuous-on* $(cball\ z\ r)\ f$
and *holf*: $f\ holomorphic\text{-}on\ ball\ z\ r$
and w: $w \in ball\ z\ r$
shows $(\lambda u.\ f\ u\ /\ (u - w)\hat{}(Suc\ k))\ contour\text{-}integrable\text{-}on\ (circlepath\ z\ r)$
(**is** *?thes1*)
and $(deriv\ \hat{}\hat{}\ k)\ f\ w = (fact\ k)\ /\ (2 * pi * ii) * contour\text{-}integral(circlepath\ z\ r)\ (\lambda u.\ f\ u/(u - w)\hat{}(Suc\ k))$
(**is** *?thes2*)
⟨*proof*⟩

corollary *Cauchy-contour-integral-circlepath*:
assumes *continuous-on* $(cball\ z\ r)\ f\ f\ holomorphic\text{-}on\ ball\ z\ r\ w \in ball\ z\ r$
shows $contour\text{-}integral(circlepath\ z\ r)\ (\lambda u.\ f\ u/(u - w)\hat{}(Suc\ k)) = (2 * pi * ii) * (deriv\ \hat{}\hat{}\ k)\ f\ w\ /\ (fact\ k)$
⟨*proof*⟩

corollary *Cauchy-contour-integral-circlepath-2*:
assumes *continuous-on* $(cball\ z\ r)\ f\ f\ holomorphic\text{-}on\ ball\ z\ r\ w \in ball\ z\ r$
shows $contour\text{-}integral(circlepath\ z\ r)\ (\lambda u.\ f\ u/(u - w)\hat{}2) = (2 * pi * ii) * deriv\ f\ w$
⟨*proof*⟩

53.39 A holomorphic function is analytic, i.e. has local power series.

theorem *holomorphic-power-series*:
assumes *holf*: $f\ holomorphic\text{-}on\ ball\ z\ r$
and w: $w \in ball\ z\ r$
shows $((\lambda n.\ (deriv\ \hat{}\hat{}\ n)\ f\ z\ /\ (fact\ n) * (w - z)\hat{}n)\ sums\ f\ w)$
⟨*proof*⟩

53.40 The Liouville theorem and the Fundamental Theorem of Algebra.

These weak Liouville versions don't even need the derivative formula.

lemma *Liouville-weak-0*:
assumes *holf*: $f\ holomorphic\text{-}on\ UNIV$ **and** *inf*: $(f \longrightarrow 0)\ at\text{-}infinity$
shows $f\ z = 0$
⟨*proof*⟩

proposition *Liouville-weak*:
assumes $f\ holomorphic\text{-}on\ UNIV$ **and** $(f \longrightarrow l)\ at\text{-}infinity$

shows $f\ z = l$
⟨*proof*⟩

proposition *Liouville-weak-inverse*:
assumes f *holomorphic-on UNIV* **and** *unbounded*: $\bigwedge B.$ *eventually* $(\lambda x.$ *norm* $(f\ x) \geq B)$ *at-infinity*
obtains z **where** $f\ z = 0$
⟨*proof*⟩

In particular we get the Fundamental Theorem of Algebra.

theorem *fundamental-theorem-of-algebra*:
fixes $a :: nat \Rightarrow complex$
assumes $a\ 0 = 0 \vee (\exists\, i \in \{1..n\}.\ a\ i \neq 0)$
obtains z **where** $(\sum i{\leq}n.\ a\ i * z\hat{}i) = 0$
⟨*proof*⟩

53.41 Weierstrass convergence theorem.

proposition *holomorphic-uniform-limit*:
assumes *cont*: *eventually* $(\lambda n.$ *continuous-on* $(cball\ z\ r)\ (f\ n) \wedge (f\ n)$ *holomorphic-on* $ball\ z\ r)\ F$
and *lim*: $\bigwedge e.\ 0 < e \Longrightarrow$ *eventually* $(\lambda n.\ \forall x \in cball\ z\ r.\ norm(f\ n\ x - g\ x) < e)\ F$
and F: $\sim$ *trivial-limit* F
obtains *continuous-on* $(cball\ z\ r)\ g\ g$ *holomorphic-on* $ball\ z\ r$
⟨*proof*⟩

Version showing that the limit is the limit of the derivatives.

proposition *has-complex-derivative-uniform-limit*:
fixes z::*complex*
assumes *cont*: *eventually* $(\lambda n.$ *continuous-on* $(cball\ z\ r)\ (f\ n) \wedge$
$(\forall w \in ball\ z\ r.\ ((f\ n)$ *has-field-derivative* $(f'\ n\ w))\ (at\ w)))\ F$
and *lim*: $\bigwedge e.\ 0 < e \Longrightarrow$ *eventually* $(\lambda n.\ \forall x \in cball\ z\ r.\ norm(f\ n\ x - g\ x) < e)\ F$
and F: $\sim$ *trivial-limit* F **and** $0 < r$
obtains g' **where**
continuous-on $(cball\ z\ r)\ g$
$\bigwedge w.\ w \in ball\ z\ r \Longrightarrow (g$ *has-field-derivative* $(g'\ w))\ (at\ w) \wedge ((\lambda n.\ f'\ n\ w) \longrightarrow g'\ w)\ F$
⟨*proof*⟩

53.42 Some more simple/convenient versions for applications.

lemma *holomorphic-uniform-sequence*:
assumes s: *open* s
and *hol-fn*: $\bigwedge n.\ (f\ n)$ *holomorphic-on* s
and *to-g*: $\bigwedge x.\ x \in s$

$\Longrightarrow \exists\, d.\ 0 < d \wedge \mathit{cball}\ x\ d \subseteq s \wedge$
$(\forall\, e.\ 0 < e \longrightarrow \mathit{eventually}\ (\lambda n.\ \forall\, y \in \mathit{cball}\ x\ d.\ \mathit{norm}(f\ n\ y - g\ y) < e)\ \mathit{sequentially})$
shows *g holomorphic-on s*
⟨*proof*⟩

lemma *has-complex-derivative-uniform-sequence*:
fixes *s* :: *complex set*
assumes *s*: *open s*
and *hfd*: $\bigwedge n\ x.\ x \in s \Longrightarrow ((f\ n)\ \mathit{has\text{-}field\text{-}derivative}\ f'\ n\ x)\ (\mathit{at}\ x)$
and *to-g*: $\bigwedge x.\ x \in s$
$\Longrightarrow \exists\, d.\ 0 < d \wedge \mathit{cball}\ x\ d \subseteq s \wedge$
$(\forall\, e.\ 0 < e \longrightarrow \mathit{eventually}\ (\lambda n.\ \forall\, y \in \mathit{cball}\ x\ d.\ \mathit{norm}(f\ n\ y - g\ y) < e)\ \mathit{sequentially})$
shows $\exists\, g'.\ \forall\, x \in s.\ (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ x)\ (\mathit{at}\ x) \wedge ((\lambda n.\ f'\ n\ x) \longrightarrow g'\ x)\ \mathit{sequentially}$
⟨*proof*⟩

53.43 On analytic functions defined by a series.

lemma *series-and-derivative-comparison*:
fixes *s* :: *complex set*
assumes *s*: *open s*
and *h*: *summable h*
and *hfd*: $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ \mathit{has\text{-}field\text{-}derivative}\ f'\ n\ x)\ (\mathit{at}\ x)$
and *to-g*: $\bigwedge n\ x.\ [\![N \le n;\ x \in s]\!] \Longrightarrow \mathit{norm}(f\ n\ x) \le h\ n$
obtains *g g'* **where** $\forall\, x \in s.\ ((\lambda n.\ f\ n\ x)\ \mathit{sums}\ g\ x) \wedge ((\lambda n.\ f'\ n\ x)\ \mathit{sums}\ g'\ x) \wedge (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ x)\ (\mathit{at}\ x)$
⟨*proof*⟩

A version where we only have local uniform/comparative convergence.

lemma *series-and-derivative-comparison-local*:
fixes *s* :: *complex set*
assumes *s*: *open s*
and *hfd*: $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ \mathit{has\text{-}field\text{-}derivative}\ f'\ n\ x)\ (\mathit{at}\ x)$
and *to-g*: $\bigwedge x.\ x \in s \Longrightarrow$
$\exists\, d\ h\ N.\ 0 < d \wedge \mathit{summable}\ h \wedge (\forall\, n\ y.\ N \le n \wedge y \in \mathit{ball}\ x\ d \longrightarrow \mathit{norm}(f\ n\ y) \le h\ n)$
shows $\exists\, g\ g'.\ \forall\, x \in s.\ ((\lambda n.\ f\ n\ x)\ \mathit{sums}\ g\ x) \wedge ((\lambda n.\ f'\ n\ x)\ \mathit{sums}\ g'\ x) \wedge (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ x)\ (\mathit{at}\ x)$
⟨*proof*⟩

Sometimes convenient to compare with a complex series of positive reals. (?)

lemma *series-and-derivative-comparison-complex*:
fixes *s* :: *complex set*
assumes *s*: *open s*
and *hfd*: $\bigwedge n\ x.\ x \in s \Longrightarrow (f\ n\ \mathit{has\text{-}field\text{-}derivative}\ f'\ n\ x)\ (\mathit{at}\ x)$
and *to-g*: $\bigwedge x.\ x \in s \Longrightarrow$

$\exists\, d\ h\ N.\ 0 < d \wedge \mathit{summable}\ h \wedge \mathit{range}\ h \subseteq \mathit{nonneg\text{-}Reals} \wedge (\forall\, n\ y.\ N \leq n \wedge y \in \mathit{ball}\ x\ d \longrightarrow \mathit{cmod}(f\ n\ y) \leq \mathit{cmod}\ (h\ n))$
shows $\exists\, g\ g'.\ \forall\, x \in s.\ ((\lambda n.\ f\ n\ x)\ \mathit{sums}\ g\ x) \wedge ((\lambda n.\ f'\ n\ x)\ \mathit{sums}\ g'\ x) \wedge (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ x)\ (\mathit{at}\ x)$
⟨*proof*⟩

In particular, a power series is analytic inside circle of convergence.

lemma *power-series-and-derivative-0*:
fixes $a :: \mathit{nat} \Rightarrow \mathit{complex}$ **and** r::*real*
assumes $\mathit{summable}\ (\lambda n.\ a\ n * r\hat{}n)$
shows $\exists\, g\ g'.\ \forall\, z.\ \mathit{cmod}\ z < r \longrightarrow ((\lambda n.\ a\ n * z\hat{}n)\ \mathit{sums}\ g\ z) \wedge ((\lambda n.\ \mathit{of\text{-}nat}\ n * a\ n * z\hat{}(n - 1))\ \mathit{sums}\ g'\ z) \wedge (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ z)\ (\mathit{at}\ z)$
⟨*proof*⟩

proposition *power-series-and-derivative*:
fixes $a :: \mathit{nat} \Rightarrow \mathit{complex}$ **and** r::*real*
assumes $\mathit{summable}\ (\lambda n.\ a\ n * r\hat{}n)$
obtains $g\ g'$ **where** $\forall\, z \in \mathit{ball}\ w\ r.\ ((\lambda n.\ a\ n * (z - w)\ \hat{}\ n)\ \mathit{sums}\ g\ z) \wedge ((\lambda n.\ \mathit{of\text{-}nat}\ n * a\ n * (z - w)\ \hat{}\ (n - 1))\ \mathit{sums}\ g'\ z) \wedge (g\ \mathit{has\text{-}field\text{-}derivative}\ g'\ z)\ (\mathit{at}\ z)$
⟨*proof*⟩

proposition *power-series-holomorphic*:
assumes $\bigwedge w.\ w \in \mathit{ball}\ z\ r \Longrightarrow ((\lambda n.\ a\ n*(w - z)\hat{}n)\ \mathit{sums}\ f\ w)$
shows $f\ \mathit{holomorphic\text{-}on}\ \mathit{ball}\ z\ r$
⟨*proof*⟩

corollary *holomorphic-iff-power-series*:
$f\ \mathit{holomorphic\text{-}on}\ \mathit{ball}\ z\ r \longleftrightarrow (\forall\, w \in \mathit{ball}\ z\ r.\ (\lambda n.\ (\mathit{deriv}\ \hat{}\hat{}\ n)\ f\ z\ /\ (\mathit{fact}\ n) * (w - z)\hat{}n)\ \mathit{sums}\ f\ w)$
⟨*proof*⟩

corollary *power-series-analytic*:
$(\bigwedge w.\ w \in \mathit{ball}\ z\ r \Longrightarrow (\lambda n.\ a\ n*(w - z)\hat{}n)\ \mathit{sums}\ f\ w) \Longrightarrow f\ \mathit{analytic\text{-}on}\ \mathit{ball}\ z\ r$
⟨*proof*⟩

corollary *analytic-iff-power-series*:
$f\ \mathit{analytic\text{-}on}\ \mathit{ball}\ z\ r \longleftrightarrow (\forall\, w \in \mathit{ball}\ z\ r.\ (\lambda n.\ (\mathit{deriv}\ \hat{}\hat{}\ n)\ f\ z\ /\ (\mathit{fact}\ n) * (w - z)\hat{}n)\ \mathit{sums}\ f\ w)$
⟨*proof*⟩

53.44 Equality between holomorphic functions, on open ball then connected set.

lemma *holomorphic-fun-eq-on-ball*:
⟦$f\ \mathit{holomorphic\text{-}on}\ \mathit{ball}\ z\ r$; $g\ \mathit{holomorphic\text{-}on}\ \mathit{ball}\ z\ r$;

$w \in ball\ z\ r$;
$\bigwedge n.\ (deriv\ \hat{}\hat{}\ n)\ f\ z = (deriv\ \hat{}\hat{}\ n)\ g\ z\rrbracket$
$\Longrightarrow f\ w = g\ w$
⟨*proof*⟩

lemma *holomorphic-fun-eq-0-on-ball*:
$\llbracket f\ holomorphic\text{-}on\ ball\ z\ r;\ \ w \in ball\ z\ r;$
$\bigwedge n.\ (deriv\ \hat{}\hat{}\ n)\ f\ z = 0\rrbracket$
$\Longrightarrow f\ w = 0$
⟨*proof*⟩

lemma *holomorphic-fun-eq-0-on-connected*:
assumes *holf*: $f\ holomorphic\text{-}on\ s$ **and** *open s*
and *cons*: *connected s*
and *der*: $\bigwedge n.\ (deriv\ \hat{}\hat{}\ n)\ f\ z = 0$
and $z \in s\ w \in s$
shows $f\ w = 0$
⟨*proof*⟩

lemma *holomorphic-fun-eq-on-connected*:
assumes $f\ holomorphic\text{-}on\ s\ g\ holomorphic\text{-}on\ s$ **and** *open s connected s*
and $\bigwedge n.\ (deriv\ \hat{}\hat{}\ n)\ f\ z = (deriv\ \hat{}\hat{}\ n)\ g\ z$
and $z \in s\ w \in s$
shows $f\ w = g\ w$
⟨*proof*⟩

lemma *holomorphic-fun-eq-const-on-connected*:
assumes *holf*: $f\ holomorphic\text{-}on\ s$ **and** *open s*
and *cons*: *connected s*
and *der*: $\bigwedge n.\ 0 < n \Longrightarrow (deriv\ \hat{}\hat{}\ n)\ f\ z = 0$
and $z \in s\ w \in s$
shows $f\ w = f\ z$
⟨*proof*⟩

53.45 Some basic lemmas about poles/singularities.

lemma *pole-lemma*:
assumes *holf*: $f\ holomorphic\text{-}on\ s$ **and** a: $a \in interior\ s$
shows $(\lambda z.\ if\ z = a\ then\ deriv\ f\ a$
$else\ (f\ z - f\ a)\ /\ (z - a))\ holomorphic\text{-}on\ s$ (**is** *?F holomorphic-on s*)
⟨*proof*⟩

proposition *pole-theorem*:
assumes *holg*: $g\ holomorphic\text{-}on\ s$ **and** a: $a \in interior\ s$
and *eq*: $\bigwedge z.\ z \in s - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
shows $(\lambda z.\ if\ z = a\ then\ deriv\ g\ a$
$else\ f\ z - g\ a/(z - a))\ holomorphic\text{-}on\ s$
⟨*proof*⟩

lemma *pole-lemma-open*:
assumes *f holomorphic-on s open s*
shows $(\lambda z.\ \mathit{if}\ z = a\ \mathit{then}\ \mathit{deriv}\ f\ a\ \mathit{else}\ (f\ z - f\ a)/(z - a))\ \mathit{holomorphic\text{-}on}\ s$
⟨*proof*⟩

proposition *pole-theorem-open*:
assumes *holg*: *g holomorphic-on s* **and** *s*: *open s*
and *eq*: $\bigwedge z.\ z \in s - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
shows $(\lambda z.\ \mathit{if}\ z = a\ \mathit{then}\ \mathit{deriv}\ g\ a\ \mathit{else}\ f\ z - g\ a/(z - a))\ \mathit{holomorphic\text{-}on}\ s$
⟨*proof*⟩

proposition *pole-theorem-0*:
assumes *holg*: *g holomorphic-on s* **and** *a*: $a \in \mathit{interior}\ s$
and *eq*: $\bigwedge z.\ z \in s - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
and [*simp*]: $f\ a = \mathit{deriv}\ g\ a\ g\ a = 0$
shows *f holomorphic-on s*
⟨*proof*⟩

proposition *pole-theorem-open-0*:
assumes *holg*: *g holomorphic-on s* **and** *s*: *open s*
and *eq*: $\bigwedge z.\ z \in s - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
and [*simp*]: $f\ a = \mathit{deriv}\ g\ a\ g\ a = 0$
shows *f holomorphic-on s*
⟨*proof*⟩

lemma *pole-theorem-analytic*:
assumes *g*: *g analytic-on s*
and *eq*: $\bigwedge z.\ z \in s \Longrightarrow \exists\, d.\ 0 < d \wedge (\forall\, w \in \mathit{ball}\ z\ d - \{a\}.\ g\ w = (w - a) * f\ w)$
shows $(\lambda z.\ \mathit{if}\ z = a\ \mathit{then}\ \mathit{deriv}\ g\ a\ \mathit{else}\ f\ z - g\ a/(z - a))\ \mathit{analytic\text{-}on}\ s$
⟨*proof*⟩

lemma *pole-theorem-analytic-0*:
assumes *g*: *g analytic-on s*
and *eq*: $\bigwedge z.\ z \in s \Longrightarrow \exists\, d.\ 0 < d \wedge (\forall\, w \in \mathit{ball}\ z\ d - \{a\}.\ g\ w = (w - a) * f\ w)$
and [*simp*]: $f\ a = \mathit{deriv}\ g\ a\ g\ a = 0$
shows *f analytic-on s*
⟨*proof*⟩

lemma *pole-theorem-analytic-open-superset*:
assumes *g*: *g analytic-on s* **and** $s \subseteq t$ *open t*
and *eq*: $\bigwedge z.\ z \in t - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
shows $(\lambda z.\ \mathit{if}\ z = a\ \mathit{then}\ \mathit{deriv}\ g\ a\ \mathit{else}\ f\ z - g\ a/(z - a))\ \mathit{analytic\text{-}on}\ s$
⟨*proof*⟩

lemma *pole-theorem-analytic-open-superset-0*:
assumes *g*: *g analytic-on s* $s \subseteq t$ *open t* $\bigwedge z.\ z \in t - \{a\} \Longrightarrow g\ z = (z - a) * f\ z$
and [*simp*]: $f\ a = deriv\ g\ a\ g\ a = 0$
shows *f analytic-on s*
⟨*proof*⟩

53.46 General, homology form of Cauchy's theorem.

Proof is based on Dixon's, as presented in Lang's "Complex Analysis" book (page 147).

lemma *contour-integral-continuous-on-linepath-2D*:
assumes *open u* **and** *cont-dw*: $\bigwedge w.\ w \in u \Longrightarrow F\ w$ *contour-integrable-on* (*linepath a b*)
and *cond-uu*: *continuous-on* $(u \times u)$ $(\lambda(x,y).\ F\ x\ y)$
and *abu*: *closed-segment a b* $\subseteq u$
shows *continuous-on u* $(\lambda w.$ *contour-integral* (*linepath a b*) $(F\ w))$
⟨*proof*⟩

This version has *polynomial-function* γ as an additional assumption.

lemma *Cauchy-integral-formula-global-weak*:
assumes *u*: *open u* **and** *holf*: *f holomorphic-on u*
and *z*: $z \in u$ **and** γ: *polynomial-function* γ
and *pasz*: *path-image* $\gamma \subseteq u - \{z\}$ **and** *loop*: *pathfinish* γ = *pathstart* γ
and *zero*: $\bigwedge w.\ w \notin u \Longrightarrow$ *winding-number* $\gamma\ w = 0$
shows $((\lambda w.\ f\ w\ /\ (w - z))$ *has-contour-integral* $(2{*}pi * ii *$ *winding-number* $\gamma\ z * f\ z))\ \gamma$
⟨*proof*⟩

theorem *Cauchy-integral-formula-global*:
assumes *s*: *open s* **and** *holf*: *f holomorphic-on s*
and *z*: $z \in s$ **and** *vpg*: *valid-path* γ
and *pasz*: *path-image* $\gamma \subseteq s - \{z\}$ **and** *loop*: *pathfinish* γ = *pathstart* γ
and *zero*: $\bigwedge w.\ w \notin s \Longrightarrow$ *winding-number* $\gamma\ w = 0$
shows $((\lambda w.\ f\ w\ /\ (w - z))$ *has-contour-integral* $(2{*}pi * ii *$ *winding-number* $\gamma\ z * f\ z))\ \gamma$
⟨*proof*⟩

theorem *Cauchy-theorem-global*:
assumes *s*: *open s* **and** *holf*: *f holomorphic-on s*
and *vpg*: *valid-path* γ **and** *loop*: *pathfinish* γ = *pathstart* γ
and *pas*: *path-image* $\gamma \subseteq s$
and *zero*: $\bigwedge w.\ w \notin s \Longrightarrow$ *winding-number* $\gamma\ w = 0$
shows (*f has-contour-integral 0*) γ
⟨*proof*⟩

corollary *Cauchy-theorem-global-outside*:
assumes *open s f holomorphic-on s valid-path* γ *pathfinish* γ = *pathstart* γ *path-image* $\gamma \subseteq s$
$\bigwedge w.\ w \notin s \Longrightarrow w \in outside(path\text{-}image\ \gamma)$
shows (*f has-contour-integral 0*) γ
⟨*proof*⟩

end

54 Conformal Mappings. Consequences of Cauchy's integral theorem.

By John Harrison et al. Ported from HOL Light by L C Paulson (2016)

Also Cauchy's residue theorem by Wenda Li (2016)

theory *Conformal-Mappings*
imports ~~/*src*/*HOL*/*Multivariate-Analysis*/*Cauchy-Integral-Thm*

begin

54.1 Cauchy's inequality and more versions of Liouville

lemma *Cauchy-higher-deriv-bound*:
assumes *holf*: *f holomorphic-on* (*ball z r*)
and *contf*: *continuous-on* (*cball z r*) *f*
and $0 < r$ **and** $0 < n$
and *fin* : $\bigwedge w.\ w \in ball\ z\ r \Longrightarrow f\ w \in ball\ y\ B0$
shows *norm* ((*deriv* ^^ *n*) *f z*) ≤ (*fact n*) * *B0* / *r*^*n*
⟨*proof*⟩

proposition *Cauchy-inequality*:
assumes *holf*: *f holomorphic-on* (*ball* ξ *r*)
and *contf*: *continuous-on* (*cball* ξ *r*) *f*
and $0 < r$
and *nof*: $\bigwedge x.\ norm(\xi - x) = r \Longrightarrow norm(f\ x) \leq B$
shows *norm* ((*deriv* ^^ *n*) *f* ξ) ≤ (*fact n*) * *B* / *r*^*n*
⟨*proof*⟩

proposition *Liouville-polynomial*:
assumes *holf*: *f holomorphic-on UNIV*
and *nof*: $\bigwedge z.\ A \leq norm\ z \Longrightarrow norm(f\ z) \leq B * norm\ z\ \hat{}\ n$
shows *f* ξ = ($\sum k \leq n.$ (*deriv*^^*k*) *f 0* / *fact k* * ξ ^ *k*)
⟨*proof*⟩

Every bounded entire function is a constant function.

theorem *Liouville-theorem*:
assumes *holf*: *f holomorphic-on UNIV*

and *bf*: *bounded* (*range f*)
obtains *c* **where** $\bigwedge z.\ f\ z = c$
⟨*proof*⟩

A holomorphic function f has only isolated zeros unless f is 0.

proposition *powser-0-nonzero*:
fixes *a* :: *nat* ⇒ *'a*::{*real-normed-field*,*banach*}
assumes *r*: $0 < r$
and *sm*: $\bigwedge x.\ norm\ (x - \xi) < r \Longrightarrow (\lambda n.\ a\ n * (x - \xi)\ \hat{}\ n)\ sums\ (f\ x)$
and [*simp*]: $f\ \xi = 0$
and *m0*: $a\ m \neq 0$ **and** $m>0$
obtains *s* **where** $0 < s$ **and** $\bigwedge z.\ z \in cball\ \xi\ s - \{\xi\} \Longrightarrow f\ z \neq 0$
⟨*proof*⟩

proposition *isolated-zeros*:
assumes *holf*: *f holomorphic-on S*
and *open S connected S* $\xi \in S$ $f\ \xi = 0$ $\beta \in S$ $f\ \beta \neq 0$
obtains *r* **where** $0 < r$ *ball* $\xi\ r \subseteq S$ $\bigwedge z.\ z \in ball\ \xi\ r - \{\xi\} \Longrightarrow f\ z \neq 0$
⟨*proof*⟩

proposition *analytic-continuation*:
assumes *holf*: *f holomorphic-on S*
and *S*: *open S connected S*
and $U \subseteq S$ $\xi \in S$
and ξ *islimpt U*
and *fU0* [*simp*]: $\bigwedge z.\ z \in U \Longrightarrow f\ z = 0$
and $w \in S$
shows $f\ w = 0$
⟨*proof*⟩

54.2 Open mapping theorem

lemma *holomorphic-contract-to-zero*:
assumes *contf*: *continuous-on* (*cball* $\xi\ r$) *f*
and *holf*: *f holomorphic-on ball* $\xi\ r$
and $0 < r$
and *norm-less*: $\bigwedge z.\ norm(\xi - z) = r \Longrightarrow norm(f\ \xi) < norm(f\ z)$
obtains *z* **where** $z \in ball\ \xi\ r$ $f\ z = 0$
⟨*proof*⟩

theorem *open-mapping-thm*:
assumes *holf*: *f holomorphic-on S*
and *S*: *open S connected S*
and *open U* $U \subseteq S$
and *fne*: $\sim$ *f constant-on S*
shows *open* (*f* ‘ *U*)
⟨*proof*⟩

No need for S to be connected. But the nonconstant condition is stronger.

corollary *open-mapping-thm2*:
assumes *holf*: *f holomorphic-on S*
and *S*: *open S*
and *open U U* $\subseteq$ *S*
and *fnc*: $\bigwedge X$. $[\![$*open X*; *X* $\subseteq$ *S*; *X* $\neq$ {}$]\!] \Longrightarrow$ ~ *f constant-on X*
shows *open* (*f* ' *U*)
⟨*proof*⟩

corollary *open-mapping-thm3*:
assumes *holf*: *f holomorphic-on S*
and *open S* **and** *injf*: *inj-on f S*
shows *open* (*f* ' *S*)
⟨*proof*⟩

54.3 Maximum Modulus Principle

If f is holomorphic, then its norm (modulus) cannot exhibit a true local maximum that is properly within the domain of *f*.

proposition *maximum-modulus-principle*:
assumes *holf*: *f holomorphic-on S*
and *S*: *open S connected S*
and *open U U* $\subseteq$ *S* $\xi \in U$
and *no*: $\bigwedge z.\ z \in U \Longrightarrow$ *norm*(*f z*) $\leq$ *norm*(*f* ξ)
shows *f constant-on S*
⟨*proof*⟩

proposition *maximum-modulus-frontier*:
assumes *holf*: *f holomorphic-on* (*interior S*)
and *contf*: *continuous-on* (*closure S*) *f*
and *bos*: *bounded S*
and *leB*: $\bigwedge z.\ z \in$ *frontier S* $\Longrightarrow$ *norm*(*f z*) $\leq$ *B*
and $\xi \in S$
shows *norm*(*f* ξ) $\leq$ *B*
⟨*proof*⟩

corollary *maximum-real-frontier*:
assumes *holf*: *f holomorphic-on* (*interior S*)
and *contf*: *continuous-on* (*closure S*) *f*
and *bos*: *bounded S*
and *leB*: $\bigwedge z.\ z \in$ *frontier S* $\Longrightarrow$ *Re*(*f z*) $\leq$ *B*
and $\xi \in S$
shows *Re*(*f* ξ) $\leq$ *B*
⟨*proof*⟩

54.4 Factoring out a zero according to its order

lemma *holomorphic-factor-order-of-zero*:

assumes *holf*: f *holomorphic-on* S
and *os*: *open* S
and $\xi \in S$ $0 < n$
and *dnz*: $(deriv\ \hat{}\hat{}\ n)\ f\ \xi \neq 0$
and *dfz*: $\bigwedge i.\ [\![0 < i; i < n]\!] \Longrightarrow (deriv\ \hat{}\hat{}\ i)\ f\ \xi = 0$
obtains $g\ r$ **where** $0 < r$
g *holomorphic-on* $ball\ \xi\ r$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow f\ w - f\ \xi = (w - \xi)\hat{}n * g\ w$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow g\ w \neq 0$
⟨*proof*⟩

lemma *holomorphic-factor-order-of-zero-strong*:
assumes *holf*: f *holomorphic-on* S *open* S $\xi \in S$ $0 < n$
and $(deriv\ \hat{}\hat{}\ n)\ f\ \xi \neq 0$
and $\bigwedge i.\ [\![0 < i; i < n]\!] \Longrightarrow (deriv\ \hat{}\hat{}\ i)\ f\ \xi = 0$
obtains $g\ r$ **where** $0 < r$
g *holomorphic-on* $ball\ \xi\ r$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow f\ w - f\ \xi = ((w - \xi) * g\ w)\ \hat{}\ n$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow g\ w \neq 0$
⟨*proof*⟩

lemma
fixes $k :: 'a{::}wellorder$
assumes *a-def*: $a == LEAST\ x.\ P\ x$ **and** *P*: $P\ k$
shows *def-LeastI*: $P\ a$ **and** *def-Least-le*: $a \leq k$
⟨*proof*⟩

lemma *holomorphic-factor-zero-nonconstant*:
assumes *holf*: f *holomorphic-on* S **and** S: *open* S *connected* S
and $\xi \in S$ $f\ \xi = 0$
and *nonconst*: $\bigwedge c.\ \exists z \in S.\ f\ z \neq c$
obtains $g\ r\ n$
where $0 < n$ $0 < r$ $ball\ \xi\ r \subseteq S$
g *holomorphic-on* $ball\ \xi\ r$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow f\ w = (w - \xi)\hat{}n * g\ w$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow g\ w \neq 0$
⟨*proof*⟩

lemma *holomorphic-lower-bound-difference*:
assumes *holf*: f *holomorphic-on* S **and** S: *open* S *connected* S
and $\xi \in S$ **and** $\varphi \in S$
and *fne*: $f\ \varphi \neq f\ \xi$
obtains $k\ n\ r$
where $0 < k$ $0 < r$
$ball\ \xi\ r \subseteq S$
$\bigwedge w.\ w \in ball\ \xi\ r \Longrightarrow k * norm(w - \xi)\hat{}n \leq norm(f\ w - f\ \xi)$

⟨*proof*⟩

lemma
assumes *holf*: *f holomorphic-on* $(S - \{\xi\})$ **and** ξ: $\xi \in$ *interior S*
shows *holomorphic-on-extend-lim*:
$(\exists g.\ g$ *holomorphic-on* $S \wedge (\forall z \in S - \{\xi\}.\ g\ z = f\ z)) \longleftrightarrow$
$((\lambda z.\ (z - \xi) * f\ z) \longrightarrow 0)\ (at\ \xi)$
(**is** *?P* = *?Q*)
and *holomorphic-on-extend-bounded*:
$(\exists g.\ g$ *holomorphic-on* $S \wedge (\forall z \in S - \{\xi\}.\ g\ z = f\ z)) \longleftrightarrow$
$(\exists B.$ *eventually* $(\lambda z.\ norm(f\ z) \leq B)\ (at\ \xi))$
(**is** *?P* = *?R*)
⟨*proof*⟩

proposition *pole-at-infinity*:
assumes *holf*: *f holomorphic-on UNIV* **and** *lim*: $((inverse\ o\ f) \longrightarrow l)$ *at-infinity*
obtains *a n* **where** $\bigwedge z.\ f\ z = (\sum i{\leq}n.\ a\ i * z\hat{}i)$
⟨*proof*⟩

54.5 Entire proper functions are precisely the non-trivial polynomials

proposition *proper-map-polyfun*:
fixes *c* :: *nat* ⇒ 'a::{*real-normed-div-algebra*,*heine-borel*}
assumes *closed S* **and** *compact K* **and** *c*: $c\ i \neq 0\ 1 \leq i\ i \leq n$
shows *compact* $(S \cap \{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) \in K\})$
⟨*proof*⟩

corollary *proper-map-polyfun-univ*:
fixes *c* :: *nat* ⇒ 'a::{*real-normed-div-algebra*,*heine-borel*}
assumes *compact K c i* $\neq 0\ 1 \leq i\ i \leq n$
shows *compact* $(\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) \in K\})$
⟨*proof*⟩

proposition *proper-map-polyfun-eq*:
assumes *f holomorphic-on UNIV*
shows $(\forall k.$ *compact* $k \longrightarrow$ *compact* $\{z.\ f\ z \in k\}) \longleftrightarrow$
$(\exists c\ n.\ 0 < n \wedge (c\ n \neq 0) \wedge f = (\lambda z.\ \sum i{\leq}n.\ c\ i * z\hat{}i))$
(**is** *?lhs* = *?rhs*)
⟨*proof*⟩

54.6 Relating invertibility and nonvanishing of derivative

proposition *has-complex-derivative-locally-injective*:
assumes *holf*: *f holomorphic-on S*
and *S*: $\xi \in S$ *open S*
and *dnz*: *deriv f* $\xi \neq 0$

obtains r **where** $r > 0$ *ball* ξ $r \subseteq S$ *inj-on* f (*ball* ξ r)
⟨*proof*⟩

proposition *has-complex-derivative-locally-invertible*:
assumes *holf*: f *holomorphic-on* S
and S: $\xi \in S$ *open* S
and *dnz*: *deriv* f $\xi \neq 0$
obtains r **where** $r > 0$ *ball* ξ $r \subseteq S$ *open* (f ‘ (*ball* ξ r)) *inj-on* f (*ball* ξ r)
⟨*proof*⟩

proposition *holomorphic-injective-imp-regular*:
assumes *holf*: f *holomorphic-on* S
and *open* S **and** *injf*: *inj-on* f S
and $\xi \in S$
shows *deriv* f $\xi \neq 0$
⟨*proof*⟩

Hence a nice clean inverse function theorem

proposition *holomorphic-has-inverse*:
assumes *holf*: f *holomorphic-on* S
and *open* S **and** *injf*: *inj-on* f S
obtains g **where** g *holomorphic-on* (f ‘ S)
$\bigwedge z.\ z \in S \Longrightarrow$ *deriv* f z $*$ *deriv* g (f z) $= 1$
$\bigwedge z.\ z \in S \Longrightarrow g(f\ z) = z$
⟨*proof*⟩

54.7 The Schwarz Lemma

lemma *Schwarz1*:
assumes *holf*: f *holomorphic-on* S
and *contf*: *continuous-on* (*closure* S) f
and S: *open* S *connected* S
and *boS*: *bounded* S
and $S \neq \{\}$
obtains w **where** $w \in$ *frontier* S
$\bigwedge z.\ z \in$ *closure* $S \Longrightarrow$ *norm* (f z) $\leq$ *norm* (f w)
⟨*proof*⟩

lemma *Schwarz2*:
⟦f *holomorphic-on ball* 0 r;
$0 < s$; *ball* w $s \subseteq$ *ball* 0 r;
$\bigwedge z.$ *norm* ($w-z$) $< s \Longrightarrow$ *norm*(f z) $\leq$ *norm*(f w)⟧
$\Longrightarrow$ f *constant-on ball* 0 r
⟨*proof*⟩

lemma *Schwarz3*:
assumes *holf*: f *holomorphic-on* (*ball* 0 r) **and** [*simp*]: f 0 $= 0$

obtains h **where** h *holomorphic-on* (*ball 0 r*) **and** $\bigwedge z.\ norm\ z < r \Longrightarrow f\ z = z * (h\ z)$ **and** $deriv\ f\ 0 = h\ 0$
⟨*proof*⟩

proposition *Schwarz-Lemma*:
assumes *holf*: f *holomorphic-on* (*ball 0 1*) **and** [*simp*]: $f\ 0 = 0$
and *no*: $\bigwedge z.\ norm\ z < 1 \Longrightarrow norm\ (f\ z) < 1$
and ξ: $norm\ \xi < 1$
shows $norm\ (f\ \xi) \leq norm\ \xi$ **and** $norm(deriv\ f\ 0) \leq 1$
and $((\exists z.\ norm\ z < 1 \wedge z \neq 0 \wedge norm(f\ z) = norm\ z) \vee norm(deriv\ f\ 0) = 1)$
$\Longrightarrow \exists \alpha.\ (\forall z.\ norm\ z < 1 \longrightarrow f\ z = \alpha * z) \wedge norm\ \alpha = 1$ (**is** *?P* $\Longrightarrow$ *?Q*)
⟨*proof*⟩

54.8 The Schwarz reflection principle

lemma *hol-pal-lem0*:
assumes $d \cdot a \leq k\ k \leq d \cdot b$
obtains c **where**
$c \in$ *closed-segment a b* $d \cdot c = k$
$\bigwedge z.\ z \in$ *closed-segment a c* $\Longrightarrow d \cdot z \leq k$
$\bigwedge z.\ z \in$ *closed-segment c b* $\Longrightarrow k \leq d \cdot z$
⟨*proof*⟩

lemma *hol-pal-lem1*:
assumes *convex S open S*
and *abc*: $a \in S\ b \in S\ c \in S$
$d \neq 0$ **and** *lek*: $d \cdot a \leq k\ d \cdot b \leq k\ d \cdot c \leq k$
and *holf1*: f *holomorphic-on* $\{z.\ z \in S \wedge d \cdot z < k\}$
and *contf*: *continuous-on S f*
shows *contour-integral* (*linepath a b*) f +
contour-integral (*linepath b c*) f +
contour-integral (*linepath c a*) $f = 0$
⟨*proof*⟩

lemma *hol-pal-lem2*:
assumes *S*: *convex S open S*
and *abc*: $a \in S\ b \in S\ c \in S$
and $d \neq 0$ **and** *lek*: $d \cdot a \leq k\ d \cdot b \leq k$
and *holf1*: f *holomorphic-on* $\{z.\ z \in S \wedge d \cdot z < k\}$
and *holf2*: f *holomorphic-on* $\{z.\ z \in S \wedge k < d \cdot z\}$
and *contf*: *continuous-on S f*
shows *contour-integral* (*linepath a b*) f +
contour-integral (*linepath b c*) f +
contour-integral (*linepath c a*) $f = 0$
⟨*proof*⟩

lemma *hol-pal-lem3*:
assumes S: *convex S open S*
and *abc*: $a \in S\ b \in S\ c \in S$
and $d \neq 0$ **and** *lek*: $d \cdot a \leq k$
and *holf1*: *f holomorphic-on* $\{z.\ z \in S \wedge d \cdot z < k\}$
and *holf2*: *f holomorphic-on* $\{z.\ z \in S \wedge k < d \cdot z\}$
and *contf*: *continuous-on S f*
shows *contour-integral* (*linepath a b*) *f* +
contour-integral (*linepath b c*) *f* +
contour-integral (*linepath c a*) *f* = *0*
⟨*proof*⟩

lemma *hol-pal-lem4*:
assumes S: *convex S open S*
and *abc*: $a \in S\ b \in S\ c \in S$ **and** $d \neq 0$
and *holf1*: *f holomorphic-on* $\{z.\ z \in S \wedge d \cdot z < k\}$
and *holf2*: *f holomorphic-on* $\{z.\ z \in S \wedge k < d \cdot z\}$
and *contf*: *continuous-on S f*
shows *contour-integral* (*linepath a b*) *f* +
contour-integral (*linepath b c*) *f* +
contour-integral (*linepath c a*) *f* = *0*
⟨*proof*⟩

proposition *holomorphic-on-paste-across-line*:
assumes S: *open S* **and** $d \neq 0$
and *holf1*: *f holomorphic-on* $(S \cap \{z.\ d \cdot z < k\})$
and *holf2*: *f holomorphic-on* $(S \cap \{z.\ k < d \cdot z\})$
and *contf*: *continuous-on S f*
shows *f holomorphic-on S*
⟨*proof*⟩

proposition *Schwarz-reflection*:
assumes *open S* **and** *cnjs*: *cnj ' S* $\subseteq S$
and *holf*: *f holomorphic-on* $(S \cap \{z.\ 0 < Im\ z\})$
and *contf*: *continuous-on* $(S \cap \{z.\ 0 \leq Im\ z\})$ *f*
and *f*: $\bigwedge z.\ [\![z \in S;\ z \in \mathbb{R}]\!] \Longrightarrow (f\ z) \in \mathbb{R}$
shows $(\lambda z.$ *if* $0 \leq Im\ z$ *then f z else cnj*$(f(cnj\ z)))$ *holomorphic-on S*
⟨*proof*⟩

54.9 Bloch's theorem

lemma *Bloch-lemma-0*:
assumes *holf*: *f holomorphic-on cball 0 r* **and** $0 < r$
and [*simp*]: *f 0* = *0*
and *le*: $\bigwedge z.\ norm\ z < r \Longrightarrow norm(deriv\ f\ z) \leq 2 * norm(deriv\ f\ 0)$
shows *ball 0* $((3 - 2 * sqrt\ 2) * r * norm(deriv\ f\ 0)) \subseteq f$ *' ball 0 r*
⟨*proof*⟩

lemma *Bloch-lemma*:

assumes *holf*: *f holomorphic-on cball a r* **and** *0 < r*
and *le*: $\bigwedge z.\ z \in ball\ a\ r \Longrightarrow norm(deriv\ f\ z) \leq 2 * norm(deriv\ f\ a)$
shows *ball (f a) ((3 − 2 * sqrt 2) * r * norm(deriv f a)) ⊆ f ‘ ball a r*
⟨*proof*⟩

proposition *Bloch-unit*:
assumes *holf*: *f holomorphic-on ball a 1* **and** [*simp*]: *deriv f a = 1*
obtains *b r* **where** *1/12 < r ball b r ⊆ f ‘ (ball a 1)*
⟨*proof*⟩

theorem *Bloch*:
assumes *holf*: *f holomorphic-on ball a r* **and** *0 < r*
and *r′*: $r' \leq r * norm\ (deriv\ f\ a)\ /\ 12$
obtains *b* **where** *ball b r′ ⊆ f ‘ (ball a r)*
⟨*proof*⟩

corollary *Bloch-general*:
assumes *holf*: *f holomorphic-on s* **and** *a ∈ s*
and *tle*: $\bigwedge z.\ z \in frontier\ s \Longrightarrow t \leq dist\ a\ z$
and *rle*: $r \leq t * norm(deriv\ f\ a)\ /\ 12$
obtains *b* **where** *ball b r ⊆ f ‘ s*
⟨*proof*⟩

54.10 Cauchy's residue theorem

Wenda Li (2016)

declare *valid-path-imp-path* [*simp*]

lemma *holomorphic-factor-zero-unique*:
fixes *f*::*complex ⇒ complex* **and** *z*::*complex* **and** *r*::*real*
assumes *r>0*
and *asm*:$\forall\ w{\in}ball\ z\ r.\ f\ w = (w{-}z)\ \hat{}n * g\ w \wedge g\ w{\neq}0 \wedge f\ w = (w - z)\ \hat{}m * h\ w \wedge h\ w{\neq}0$
and *g-holo*:*g holomorphic-on ball z r* **and** *h-holo*:*h holomorphic-on ball z r*
shows *n=m*
⟨*proof*⟩

lemma *holomorphic-factor-zero-Ex1*:
assumes *open s connected s z ∈ s* **and**
holo:*f holomorphic-on s*
and *f z = 0* **and** $\exists\ w{\in}s.\ f\ w \neq 0$
shows $\exists!n.\ \exists\ g\ r.\ 0 < n \wedge 0 < r \wedge$
g holomorphic-on cball z r
$\wedge\ (\forall\ w{\in}cball\ z\ r.\ f\ w = (w{-}z)\ \hat{}n * g\ w \wedge g\ w{\neq}0)$
⟨*proof*⟩

lemma *finite-ball-avoid*:
assumes *open s finite pts*

shows $\forall\, p{\in}s.\ \exists\, e{>}0.\ \forall\, w{\in}ball\ p\ e.\ w{\in}s \wedge (w{\neq}p \longrightarrow w{\notin}pts)$
⟨*proof*⟩

lemma *finite-cball-avoid*:
fixes $s :: 'a :: euclidean\text{-}space\ set$
assumes *open s finite pts*
shows $\forall\, p{\in}s.\ \exists\, e{>}0.\ \forall\, w{\in}cball\ p\ e.\ w{\in}s \wedge (w{\neq}p \longrightarrow w{\notin}pts)$
⟨*proof*⟩

lemma *get-integrable-path*:
assumes *open s connected* $(s-pts)$ *finite pts f holomorphic-on* $(s-pts)$ $a{\in}s-pts$ $b{\in}s-pts$
obtains *g* **where** *valid-path g pathstart g* $= a$ *pathfinish g* $= b$ *path-image g* $\subseteq s-pts$ *f contour-integrable-on g* ⟨*proof*⟩

lemma *Cauchy-theorem-aux*:
assumes *open s connected* $(s-pts)$ *finite pts pts* $\subseteq s$ *f holomorphic-on* $s-pts$
valid-path g pathfinish g $=$ *pathstart g path-image g* $\subseteq s-pts$
$\forall\, z.\ (z \notin s) \longrightarrow$ *winding-number g z* $= 0$
$\forall\, p{\in}s.\ h\ p{>}0 \wedge (\forall\, w{\in}cball\ p\ (h\ p).\ w{\in}s \wedge (w{\neq}p \longrightarrow w \notin pts))$
shows *contour-integral g f* $= (\sum p{\in}pts.$ *winding-number g p* $*$ *contour-integral* $(circlepath\ p\ (h\ p))\ f)$
⟨*proof*⟩

lemma *Cauchy-theorem-singularities*:
assumes *open s connected* $(s-pts)$ *finite pts* **and**
holo:*f holomorphic-on* $s-pts$ **and**
valid-path g **and**
loop:*pathfinish g* $=$ *pathstart g* **and**
path-image g $\subseteq s-pts$ **and**
homo:$\forall\, z.\ (z \notin s) \longrightarrow$ *winding-number g z* $= 0$ **and**
avoid:$\forall\, p{\in}s.\ h\ p{>}0 \wedge (\forall\, w{\in}cball\ p\ (h\ p).\ w{\in}s \wedge (w{\neq}p \longrightarrow w \notin pts))$
shows *contour-integral g f* $= (\sum p{\in}pts.$ *winding-number g p* $*$ *contour-integral* $(circlepath\ p\ (h\ p))\ f)$
(**is** *?L*=*?R*)
⟨*proof*⟩

definition *zorder*::$(complex \Rightarrow complex) \Rightarrow complex \Rightarrow nat$ **where**
zorder f z $= (THE\ n.\ n{>}0 \wedge (\exists\, h\ r.\ r{>}0 \wedge h$ *holomorphic-on cball z r*
$\wedge\ (\forall\, w{\in}cball\ z\ r.\ f\ w = h\ w * (w-z)\ \hat{}\ n \wedge h\ w \neq 0)))$

definition *zer-poly*::$[complex \Rightarrow complex, complex] \Rightarrow complex \Rightarrow complex$ **where**
zer-poly f z $= (SOME\ h.\ \exists\, r\ .\ r{>}0 \wedge h$ *holomorphic-on cball z r*
$\wedge\ (\forall\, w{\in}cball\ z\ r.\ f\ w = h\ w * (w-z)\ \hat{}\ (zorder\ f\ z) \wedge h\ w \neq 0))$

definition *porder*::(*complex* ⇒ *complex*) ⇒ *complex* ⇒ *nat* **where**
porder f z = *zorder* (*inverse o f*) *z*

definition *pol-poly*::[*complex* ⇒ *complex*,*complex*]⇒*complex* ⇒ *complex* **where**
pol-poly f z = *inverse o zer-poly* (*inverse o f*) *z*

definition *residue*::(*complex* ⇒ *complex*) ⇒ *complex* ⇒ *complex* **where**
residue f z = (*let n*=*porder f z*;*h*=*pol-poly f z in* (*deriv* ^^ (*n* − *1*)) *h z* / *fact* (*n*−*1*))

definition *is-pole*:: (*complex* ⇒ *complex*) ⇒ *complex* ⇒*bool* **where**
is-pole f p ≡ *isCont* (*inverse o f*) *p* ∧ *f p* = *0*

lemma *zorder-exist*:
fixes *f*::*complex* ⇒ *complex* **and** *z*::*complex*
defines *n*≡*zorder f z* **and** *h*≡*zer-poly f z*
assumes *open s connected s z*∈*s*
and *holo*: *f holomorphic-on s*
and *f z*=*0* ∃ *w*∈*s*. *f w*≠*0*
shows ∃ *r*. *n*>*0* ∧ *r*>*0* ∧ *cball z r* ⊆ *s* ∧ *h holomorphic-on cball z r*
∧ (∀ *w*∈*cball z r*. *f w* = *h w* ∗ (*w*−*z*) ^*n* ∧ *h w* ≠*0*)
⟨*proof*⟩

lemma *porder-exist*:
fixes *f*::*complex* ⇒ *complex* **and** *z*::*complex*
defines *n*≡*porder f z* **and** *h*≡*pol-poly f z*
assumes *open s connected s z*∈*s*
and *holo*:(*inverse o f*) *holomorphic-on s*
and *f z*=*0* ∃ *w*∈*s*. *f w*≠*0*
shows ∃ *r*. *n*>*0* ∧ *r*>*0* ∧ *cball z r* ⊆ *s* ∧ *h holomorphic-on cball z r*
∧ (∀ *w*∈*cball z r*. *f w* = *h w* / (*w*−*z*) ^*n* ∧ *h w* ≠*0*)
⟨*proof*⟩

lemma *base-residue*:
fixes *f*::*complex* ⇒ *complex* **and** *z*::*complex*
assumes *open s connected s z*∈*s*
and *holo*: (*inverse o f*) *holomorphic-on s*
and *f z*=*0*
and *non-c*:∃ *w*∈*s*. *f w*≠*0*
shows ∃ *r*>*0*. *cball z r* ⊆ *s*
∧ (*f has-contour-integral complex-of-real* (*2* ∗ *pi*) ∗ i ∗ *residue f z*) (*circlepath z r*)
⟨*proof*⟩

theorem *residue-theorem*:
assumes *open s connected* (*s*−*poles*) **and**

holo:*f holomorphic-on s*−*poles* **and**
valid-path γ **and**
loop:*pathfinish* γ = *pathstart* γ **and**
path-image $\gamma \subseteq s-poles$ **and**
homo:$\forall z.\ (z \notin s) \longrightarrow$ *winding-number* $\gamma\ z = 0$ **and**
finite $\{p.\ f\ p=0\}$ **and**
poles:$\forall\, p{\in}poles.$ *is-pole f p*

shows *contour-integral* $\gamma\ f = 2 * pi * \mathrm{i} * (\sum p{\in}poles.$ *winding-number* $\gamma\ p *$ *residue f p*)

⟨*proof*⟩

theorem *argument-principle*:
fixes *f*::*complex* ⇒ *complex* **and** *poles s*:: *complex set*
defines *pts*≡$\{p.\ f\ p=0\}$
defines *zeros*≡*pts* − *poles*
assumes *open s* **and**
connected:*connected* (*s*− *pts*) **and**
f-holo:*f holomorphic-on s*−*poles* **and**
h-holo:*h holomorphic-on s* **and**
valid-path g **and**
loop:*pathfinish g* = *pathstart g* **and**
path-img:*path-image g* ⊆ *s* − *pts* **and**
homo:$\forall z.\ (z \notin s) \longrightarrow$ *winding-number g z* = *0* **and**
finite:*finite pts* **and**
poles:$\forall\, p{\in}poles.$ *is-pole f p*
shows *contour-integral g* ($\lambda x.$ *deriv f x* ∗ *h x* / *f x*) = *2* ∗ *pi* ∗ i ∗
($(\sum p{\in}zeros.$ *winding-number g p* ∗ *h p* ∗ *zorder f p*)
− $(\sum p{\in}poles.$ *winding-number g p* ∗ *h p* ∗ *porder f p*))
(**is** *?L*=*?R*)

⟨*proof*⟩

lemma *holomorphic-imp-diff-on*:
assumes *f-holo*:*f holomorphic-on s* **and** *open s*
shows *f differentiable-on s*

⟨*proof*⟩

theorem *Rouche-theorem*:
fixes *f g*::*complex* ⇒ *complex* **and** *s*:: *complex set*
defines *fg*≡($\lambda p.$ *f p*+ *g p*)
defines *zeros-fg*≡$\{p.\ fg\ p\ =0\}$ **and** *zeros-f*≡$\{p.\ f\ p=0\}$
assumes
open s **and**
connected-fg:*connected* (*s*− *zeros-fg*) **and**
connected-f:*connected* (*s*− *zeros-f*) **and**
finite zeros-fg **and**
finite zeros-f **and**
f-holo:*f holomorphic-on s* **and**
g-holo:*g holomorphic-on s* **and**
valid-path γ **and**

loop:*pathfinish* γ = *pathstart* γ **and**
path-img:*path-image* $\gamma \subseteq s$ **and**
path-less:$\forall z \in$*path-image* γ. *cmod*$(f\ z) >$ *cmod*$(g\ z)$ **and**
homo:$\forall z$. $(z \notin s) \longrightarrow$ *winding-number* $\gamma\ z = 0$
shows $(\sum p \in$*zeros-fg*. *winding-number* $\gamma\ p *$ *zorder fg* $p)$
$= (\sum p \in$*zeros-f*. *winding-number* $\gamma\ p *$ *zorder f* $p)$
⟨*proof*⟩

end

55 Generalised Binomial Theorem

The proof of the Generalised Binomial Theorem and related results. We prove the generalised binomial theorem for complex numbers, following the proof at: https://proofwiki.org/wiki/Binomial_Theorem/General_Binomial_Theorem

theory *Generalised-Binomial-Theorem*
imports
Complex-Main
Complex-Transcendental
Summation
begin

lemma *gbinomial-ratio-limit*:
fixes $a :: 'a ::$ *real-normed-field*
assumes $a \notin \mathbb{N}$
shows $(\lambda n.\ (a\ gchoose\ n)\ /\ (a\ gchoose\ Suc\ n)) \longrightarrow -1$
⟨*proof*⟩

lemma *conv-radius-gchoose*:
fixes $a :: 'a ::$ {*real-normed-field*,*banach*}
shows *conv-radius* $(\lambda n.\ a\ gchoose\ n) = (if\ a \in \mathbb{N}\ then\ \infty\ else\ 1)$
⟨*proof*⟩

lemma *gen-binomial-complex*:
fixes $z ::$ *complex*
assumes *norm* $z < 1$
shows $(\lambda n.\ (a\ gchoose\ n) * z\hat{}n)$ *sums* $(1 + z)$ *powr* a
⟨*proof*⟩

lemma *gen-binomial-complex'*:
fixes $x\ y ::$ *real* **and** $a ::$ *complex*
assumes $|x| < |y|$
shows $(\lambda n.\ (a\ gchoose\ n) *$ *of-real* $x\hat{}n *$ *of-real* y *powr* $(a -$ *of-nat* $n))$ *sums*
of-real $(x + y)$ *powr* a (**is** *?P* $x\ y$)
⟨*proof*⟩

lemma *gen-binomial-complex''*:

fixes $x\ y$ :: *real* **and** a :: *complex*
assumes $|y| < |x|$
shows $(\lambda n.\ (a\ gchoose\ n) * of\text{-}real\ x\ powr\ (a - of\text{-}nat\ n) * of\text{-}real\ y\ \hat{}\ n)\ sums$

$of\text{-}real\ (x + y)\ powr\ a$
⟨*proof*⟩

lemma *gen-binomial-real*:
fixes z :: *real*
assumes $|z| < 1$
shows $(\lambda n.\ (a\ gchoose\ n) * z\hat{}n)\ sums\ (1 + z)\ powr\ a$
⟨*proof*⟩

lemma *gen-binomial-real'*:
fixes $x\ y\ a$ :: *real*
assumes $|x| < y$
shows $(\lambda n.\ (a\ gchoose\ n) * x\hat{}n * y\ powr\ (a - of\text{-}nat\ n))\ sums\ (x + y)\ powr\ a$
⟨*proof*⟩

lemma *one-plus-neg-powr-powser*:
fixes $z\ s$ :: *complex*
assumes $norm\ (z :: complex) < 1$
shows $(\lambda n.\ (-1)\hat{}n * ((s + n - 1)\ gchoose\ n) * z\hat{}n)\ sums\ (1 + z)\ powr\ (-s)$
⟨*proof*⟩

lemma *gen-binomial-real''*:
fixes $x\ y\ a$ :: *real*
assumes $|y| < x$
shows $(\lambda n.\ (a\ gchoose\ n) * x\ powr\ (a - of\text{-}nat\ n) * y\hat{}n)\ sums\ (x + y)\ powr\ a$
⟨*proof*⟩

lemma *sqrt-series'*:
$|z| < a \Longrightarrow (\lambda n.\ ((1/2)\ gchoose\ n) * a\ powr\ (1/2 - real\text{-}of\text{-}nat\ n) * z\ \hat{}\ n)\ sums$

$sqrt\ (a + z :: real)$
⟨*proof*⟩

lemma *sqrt-series*:
$|z| < 1 \Longrightarrow (\lambda n.\ ((1/2)\ gchoose\ n) * z\ \hat{}\ n)\ sums\ sqrt\ (1 + z)$
⟨*proof*⟩

end

56 Integral Test for Summability

theory *Integral-Test*
imports *Integration*
begin

The integral test for summability. We show here that for a decreasing non-negative function, the infinite sum over that function evaluated at the natural numbers converges iff the corresponding integral converges.

As a useful side result, we also provide some results on the difference between the integral and the partial sum. (This is useful e.g. for the definition of the Euler-Mascheroni constant)

locale *antimono-fun-sum-integral-diff* =
  fixes f :: *real* ⇒ *real*
  assumes *dec*: $\bigwedge x\ y.\ x \geq 0 \Longrightarrow x \leq y \Longrightarrow f\ x \geq f\ y$
  assumes *nonneg*: $\bigwedge x.\ x \geq 0 \Longrightarrow f\ x \geq 0$
  assumes *cont*: *continuous-on* $\{0..\}$ f
begin

definition *sum-integral-diff-series* $n = (\sum k{\leq}n.\ f\ (\textit{of-nat}\ k)) - (\textit{integral}\ \{0..\textit{of-nat}\ n\}\ f)$

lemma *sum-integral-diff-series-nonneg*:
  sum-integral-diff-series $n \geq 0$
⟨*proof*⟩

lemma *sum-integral-diff-series-antimono*:
  assumes $m \leq n$
  shows *sum-integral-diff-series* $m \geq$ *sum-integral-diff-series* n
⟨*proof*⟩

lemma *sum-integral-diff-series-Bseq*: *Bseq sum-integral-diff-series*
⟨*proof*⟩

lemma *sum-integral-diff-series-monoseq*: *monoseq sum-integral-diff-series*
  ⟨*proof*⟩

lemma *sum-integral-diff-series-convergent*: *convergent sum-integral-diff-series*
  ⟨*proof*⟩

lemma *integral-test*:
  summable $(\lambda n.\ f\ (\textit{of-nat}\ n)) \longleftrightarrow$ *convergent* $(\lambda n.\ \textit{integral}\ \{0..\textit{of-nat}\ n\}\ f)$
⟨*proof*⟩

end

end

57 Harmonic Numbers

theory *Harmonic-Numbers*
imports
  Complex-Transcendental
  Summation

Integral-Test
begin

The definition of the Harmonic Numbers and the Euler-Mascheroni constant. Also provides a reasonably accurate approximation of *ln 2* and the Euler-Mascheroni constant.

lemma *ln-2-less-1*: *ln 2* < (*1*::*real*)
⟨*proof*⟩

lemma *setsum-Suc-diff'*:
fixes *f* :: *nat* ⇒ *'a*::*ab-group-add*
assumes *m* ≤ *n*
shows (∑ *i* = *m*..<*n*. *f* (*Suc i*) − *f i*) = *f n* − *f m*
⟨*proof*⟩

57.1 The Harmonic numbers

definition *harm* :: *nat* ⇒ *'a* :: *real-normed-field* **where**
harm n = (∑ *k*=*1*..*n*. *inverse* (*of-nat k*))

lemma *harm-altdef*: *harm n* = (∑ *k*<*n*. *inverse* (*of-nat* (*Suc k*)))
⟨*proof*⟩

lemma *harm-Suc*: *harm* (*Suc n*) = *harm n* + *inverse* (*of-nat* (*Suc n*))
⟨*proof*⟩

lemma *harm-nonneg*: *harm n* ≥ (*0* :: *'a* :: {*real-normed-field*,*linordered-field*})
⟨*proof*⟩

lemma *harm-pos*: *n* > *0* ⟹ *harm n* > (*0* :: *'a* :: {*real-normed-field*,*linordered-field*})
⟨*proof*⟩

lemma *of-real-harm*: *of-real* (*harm n*) = *harm n*
⟨*proof*⟩

lemma *norm-harm*: *norm* (*harm n*) = *harm n*
⟨*proof*⟩

lemma *harm-expand*:
harm 0 = *0*
harm (*Suc 0*) = *1*
harm (*numeral n*) = *harm* (*pred-numeral n*) + *inverse* (*numeral n*)
⟨*proof*⟩

lemma *not-convergent-harm*: ¬*convergent* (*harm* :: *nat* ⇒ *'a* :: *real-normed-field*)
⟨*proof*⟩

lemma *harm-pos-iff* [*simp*]: *harm n* > (*0* :: *'a* :: {*real-normed-field*,*linordered-field*})
⟷ *n* > *0*

⟨*proof*⟩

lemma *ln-diff-le-inverse*:
assumes $x \geq (1::real)$
shows $ln\ (x + 1) - ln\ x < 1\ /\ x$
⟨*proof*⟩

lemma *ln-le-harm*: $ln\ (real\ n + 1) \leq (harm\ n :: real)$
⟨*proof*⟩

57.2 The Euler–Mascheroni constant

The limit of the difference between the partial harmonic sum and the natural logarithm (approximately 0.577216). This value occurs e.g. in the definition of the Gamma function.

definition *euler-mascheroni* :: *'a* :: *real-normed-algebra-1* **where**
euler-mascheroni = *of-real* (*lim* (λ*n*. *harm n* − *ln* (*of-nat n*)))

lemma *of-real-euler-mascheroni* [*simp*]: *of-real euler-mascheroni* = *euler-mascheroni*
⟨*proof*⟩

interpretation *euler-mascheroni*: *antimono-fun-sum-integral-diff* λ*x*. *inverse* (*x* + *1*)
⟨*proof*⟩

lemma *euler-mascheroni-sum-integral-diff-series*:
euler-mascheroni.sum-integral-diff-series n = *harm* (*Suc n*) − *ln* (*of-nat* (*Suc n*))
⟨*proof*⟩

lemma *euler-mascheroni-sequence-decreasing*:
$m > 0 \Longrightarrow m \leq n \Longrightarrow harm\ n - ln\ (of\text{-}nat\ n) \leq harm\ m - ln\ (of\text{-}nat\ m :: real)$
⟨*proof*⟩

lemma *euler-mascheroni-sequence-nonneg*:
$n > 0 \Longrightarrow harm\ n - ln\ (of\text{-}nat\ n) \geq (0::real)$
⟨*proof*⟩

lemma *euler-mascheroni-convergent*: *convergent* (λ*n*. *harm n* − *ln* (*of-nat n*) :: *real*)
⟨*proof*⟩

lemma *euler-mascheroni-LIMSEQ*:
(λ*n*. *harm n* − *ln* (*of-nat n*) :: *real*) ⟶ *euler-mascheroni*
⟨*proof*⟩

lemma *euler-mascheroni-LIMSEQ-of-real*:
(λ*n*. *of-real* (*harm n* − *ln* (*of-nat n*))) ⟶
(*euler-mascheroni* :: *'a* :: {*real-normed-algebra-1*, *topological-space*})

⟨*proof*⟩

lemma *euler-mascheroni-sum*:
(λ*n*. *inverse* (*of-nat* (*n*+1)) + *ln* (*of-nat* (*n*+1)) − *ln* (*of-nat* (*n*+2)) :: *real*)
sums euler-mascheroni
⟨*proof*⟩

lemma *alternating-harmonic-series-sums*: (λ*k*. (−1) ^ *k* / *real-of-nat* (*Suc k*)) *sums* *ln 2*
⟨*proof*⟩

lemma *alternating-harmonic-series-sums'*:
(λ*k*. *inverse* (*real-of-nat* (2∗*k*+1)) − *inverse* (*real-of-nat* (2∗*k*+2))) *sums* *ln 2*
⟨*proof*⟩

57.3 Bounds on the Euler–Mascheroni constant

lemma *ln-inverse-approx-le*:
assumes (*x*::*real*) > *0* *a* > *0*
shows *ln* (*x* + *a*) − *ln* *x* ≤ *a* ∗ (*inverse* *x* + *inverse* (*x* + *a*))/*2* (**is** - ≤ *?A*)
⟨*proof*⟩

lemma *ln-inverse-approx-ge*:
assumes (*x*::*real*) > *0* *x* < *y*
shows *ln* *y* − *ln* *x* ≥ *2* ∗ (*y* − *x*) / (*x* + *y*) (**is** - ≥ *?A*)
⟨*proof*⟩

lemma *euler-mascheroni-lower*:
euler-mascheroni ≥ *harm* (*Suc* *n*) − *ln* (*real-of-nat* (*n* + *2*)) + *1*/*real-of-nat* (*2* ∗ (*n* + *2*))
and *euler-mascheroni-upper*:
euler-mascheroni ≤ *harm* (*Suc* *n*) − *ln* (*real-of-nat* (*n* + *2*)) + *1*/*real-of-nat* (*2* ∗ (*n* + *1*))
⟨*proof*⟩

lemma *euler-mascheroni-pos*: *euler-mascheroni* > (*0*::*real*)
⟨*proof*⟩

context
begin

private lemma *ln-approx-aux*:
fixes *n* :: *nat* **and** *x* :: *real*
defines *y* ≡ (*x*−*1*)/(*x*+*1*)
assumes *x*: *x* > *0* *x* ≠ *1*
shows *inverse* (*2*∗*y* ^ (*2*∗*n*+*1*)) ∗ (*ln* *x* − (∑ *k*<*n*. *2*∗*y* ^ (*2*∗*k*+*1*) / *of-nat* (*2*∗*k*+*1*)))
∈
{*0*..(*1* / (*1* − *y* ^ *2*) / *of-nat* (*2*∗*n*+*1*))}

⟨*proof*⟩

lemma
fixes *n* :: *nat* **and** *x* :: *real*
defines *y* ≡ (*x*−*1*)/(*x*+*1*)
defines *approx* ≡ (∑ *k*<*n*. *2*∗*y*^(*2*∗*k*+*1*) / *of-nat* (*2*∗*k*+*1*))
defines *d* ≡ *y*^(*2*∗*n*+*1*) / (*1* − *y*^*2*) / *of-nat* (*2*∗*n*+*1*)
assumes *x*: *x* > *1*
shows *ln-approx-bounds*: *ln x* ∈ {*approx*..*approx* + *2*∗*d*}
and *ln-approx-abs*: *abs* (*ln x* − (*approx* + *d*)) ≤ *d*
⟨*proof*⟩

end

lemma *euler-mascheroni-bounds*:
fixes *n* :: *nat* **assumes** *n* ≥ *1* **defines** *t* ≡ *harm n* − *ln* (*of-nat* (*Suc n*)) :: *real*
shows *euler-mascheroni* ∈ {*t* + *inverse* (*of-nat* (*2*∗(*n*+*1*)))..*t* + *inverse* (*of-nat* (*2*∗*n*))}
⟨*proof*⟩

lemma *euler-mascheroni-bounds'*:
fixes *n* :: *nat* **assumes** *n* ≥ *1* *ln* (*real-of-nat* (*Suc n*)) ∈ {*l*<..<*u*}
shows *euler-mascheroni* ∈
{*harm n* − *u* + *inverse* (*of-nat* (*2*∗(*n*+*1*)))<..<*harm n* − *l* + *inverse* (*of-nat* (*2*∗*n*))}
⟨*proof*⟩

Approximation of *ln* (*2*::′*a*). The lower bound is accurate to about 0.03; the upper bound is accurate to about 0.0015.

lemma *ln2-ge-two-thirds*: *2/3* ≤ *ln* (*2*::*real*)
and *ln2-le-25-over-36*: *ln* (*2*::*real*) ≤ *25/36*
⟨*proof*⟩

Approximation of the Euler–Mascheroni constant. The lower bound is accurate to about 0.0015; the upper bound is accurate to about 0.015.

lemma *euler-mascheroni-gt-19-over-33*: (*euler-mascheroni* :: *real*) > *19/33* (**is** *?th1*)
and *euler-mascheroni-less-13-over-22*: (*euler-mascheroni* :: *real*) < *13/22* (**is** *?th2*)
⟨*proof*⟩

end

58 Periodic Functions

theory *Periodic-Fun*
imports *Complex-Main*
begin

A locale for periodic functions. The idea is that one proves $f(x + p) = f(x)$ for some period p and gets derived results like $f(x - p) = f(x)$ and $f(x + 2p) = f(x)$ for free.

g and gm are "plus/minus k periods" functions. $g1$ and $gn1$ are "plus/minus one period" functions. This is useful e.g. if the period is one; the lemmas one gets are then $f\ (x + (1::'b)) = f\ x$ instead of $f\ (x + (1::'b) * (1::'b)) = f\ x$ etc.

locale *periodic-fun* =
  fixes $f :: ('a :: \{ring\text{-}1\}) \Rightarrow 'b$ **and** $g\ gm :: 'a \Rightarrow 'a \Rightarrow 'a$ **and** $g1\ gn1 :: 'a \Rightarrow 'a$
  assumes *plus-1*: $f\ (g1\ x) = f\ x$
  assumes *periodic-arg-plus-0*: $g\ x\ 0 = x$
  assumes *periodic-arg-plus-distrib*: $g\ x\ (of\text{-}int\ (m + n)) = g\ (g\ x\ (of\text{-}int\ n))\ (of\text{-}int\ m)$
  assumes *plus-1-eq*: $g\ x\ 1 = g1\ x$ **and** *minus-1-eq*: $g\ x\ (-1) = gn1\ x$
    and *minus-eq*: $g\ x\ (-y) = gm\ x\ y$
begin

lemma *plus-of-nat*: $f\ (g\ x\ (of\text{-}nat\ n)) = f\ x$
  ⟨*proof*⟩

lemma *minus-of-nat*: $f\ (gm\ x\ (of\text{-}nat\ n)) = f\ x$
⟨*proof*⟩

lemma *plus-of-int*: $f\ (g\ x\ (of\text{-}int\ n)) = f\ x$
  ⟨*proof*⟩

lemma *minus-of-int*: $f\ (gm\ x\ (of\text{-}int\ n)) = f\ x$
  ⟨*proof*⟩

lemma *plus-numeral*: $f\ (g\ x\ (numeral\ n)) = f\ x$
  ⟨*proof*⟩

lemma *minus-numeral*: $f\ (gm\ x\ (numeral\ n)) = f\ x$
  ⟨*proof*⟩

lemma *minus-1*: $f\ (gn1\ x) = f\ x$
  ⟨*proof*⟩

lemmas *periodic-simps* = *plus-of-nat minus-of-nat plus-of-int minus-of-int*
    plus-numeral minus-numeral plus-1 minus-1

end

Specialised case of the *periodic-fun* locale for periods that are not 1. Gives lemmas $f\ (x - period) = f\ x$ etc.

locale *periodic-fun-simple* =
  fixes $f :: ('a :: \{ring\text{-}1\}) \Rightarrow 'b$ **and** $period :: 'a$
  assumes *plus-period*: $f\ (x + period) = f\ x$

begin
sublocale *periodic-fun f* $\lambda z\ x.\ z + x * period\ \lambda z\ x.\ z - x * period$
$\lambda z.\ z + period\ \lambda z.\ z - period$
⟨*proof*⟩
end

Specialised case of the *periodic-fun* locale for period 1. Gives lemmas $f\ (x - (1::'b)) = f\ x$ etc.

locale *periodic-fun-simple'* =
fixes $f :: ('a :: \{ring\text{-}1\}) \Rightarrow 'b$
assumes *plus-period*: $f\ (x + 1) = f\ x$
begin
sublocale *periodic-fun f* $\lambda z\ x.\ z + x\ \lambda z\ x.\ z - x\ \lambda z.\ z + 1\ \lambda z.\ z - 1$
⟨*proof*⟩

lemma *of-nat*: $f\ (of\text{-}nat\ n) = f\ 0$ ⟨*proof*⟩
lemma *uminus-of-nat*: $f\ (-of\text{-}nat\ n) = f\ 0$ ⟨*proof*⟩
lemma *of-int*: $f\ (of\text{-}int\ n) = f\ 0$ ⟨*proof*⟩
lemma *uminus-of-int*: $f\ (-of\text{-}int\ n) = f\ 0$ ⟨*proof*⟩
lemma *of-numeral*: $f\ (numeral\ n) = f\ 0$ ⟨*proof*⟩
lemma *of-neg-numeral*: $f\ (-numeral\ n) = f\ 0$ ⟨*proof*⟩
lemma *of-1*: $f\ 1 = f\ 0$ ⟨*proof*⟩
lemma *of-neg-1*: $f\ (-1) = f\ 0$ ⟨*proof*⟩

lemmas *periodic-simps'* =
of-nat uminus-of-nat of-int uminus-of-int of-numeral of-neg-numeral of-1 of-neg-1

end

lemma *sin-plus-pi*: $sin\ ((z :: 'a :: \{real\text{-}normed\text{-}field, banach\}) + of\text{-}real\ pi) = -sin\ z$
⟨*proof*⟩

lemma *cos-plus-pi*: $cos\ ((z :: 'a :: \{real\text{-}normed\text{-}field, banach\}) + of\text{-}real\ pi) = -cos\ z$
⟨*proof*⟩

interpretation *sin*: *periodic-fun-simple sin* $2 * of\text{-}real\ pi :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
⟨*proof*⟩

interpretation *cos*: *periodic-fun-simple cos* $2 * of\text{-}real\ pi :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
⟨*proof*⟩

interpretation *tan*: *periodic-fun-simple tan* $2 * of\text{-}real\ pi :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
⟨*proof*⟩

interpretation *cot*: *periodic-fun-simple cot* $2 * of\text{-}real\ pi :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
⟨*proof*⟩

end

59 The Gamma Function

theory *Gamma*
imports
Complex-Transcendental
Summation
Harmonic-Numbers
~~/*src*/*HOL*/*Library*/*Nonpos-Ints*
~~/*src*/*HOL*/*Library*/*Periodic-Fun*
begin

Several equivalent definitions of the Gamma function and its most important properties. Also contains the definition and some properties of the log-Gamma function and the Digamma function and the other Polygamma functions.

Based on the Gamma function, we also prove the Weierstra product form of the sin function and, based on this, the solution of the Basel problem (the sum over all *1 / real* (n^2).

lemma *pochhammer-eq-0-imp-nonpos-Int*:
pochhammer $(x::'a::$*field-char-0*$)\ n = 0 \Longrightarrow x \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *closed-nonpos-Ints* [*simp*]: *closed* $(\mathbb{Z}_{\leq 0} :: 'a ::$ *real-normed-algebra-1 set*$)$
⟨*proof*⟩

lemma *plus-one-in-nonpos-Ints-imp*: $z + 1 \in \mathbb{Z}_{\leq 0} \Longrightarrow z \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *fraction-not-in-ints*:
assumes $\neg(n$ *dvd* $m)\ n \neq 0$
shows *of-int m / of-int n* $\notin (\mathbb{Z} :: 'a ::$ {*division-ring*,*ring-char-0*} *set*$)$
⟨*proof*⟩

lemma *not-in-Ints-imp-not-in-nonpos-Ints*: $z \notin \mathbb{Z} \Longrightarrow z \notin \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *double-in-nonpos-Ints-imp*:
assumes $2 * (z :: 'a ::$ *field-char-0*$) \in \mathbb{Z}_{\leq 0}$
shows $z \in \mathbb{Z}_{\leq 0} \vee z + 1/2 \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *sin-series*: $(\lambda n.\ ((-1)\hat{}n$ */ fact* $(2*n+1))\ *_R\ z\hat{}(2*n+1))$ *sums sin z*
⟨*proof*⟩

lemma *cos-series*: $(\lambda n.\ ((-1)\hat{}n$ */ fact* $(2*n))\ *_R\ z\hat{}(2*n))$ *sums cos z*

⟨*proof*⟩

lemma *sin-z-over-z-series*:
fixes $z :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
assumes $z \neq 0$
shows $(\lambda n.\ (-1)\hat{}n\ /\ fact\ (2*n+1) * z\hat{}(2*n))\ sums\ (sin\ z\ /\ z)$
⟨*proof*⟩

lemma *sin-z-over-z-series'*:
fixes $z :: 'a :: \{real\text{-}normed\text{-}field, banach\}$
assumes $z \neq 0$
shows $(\lambda n.\ sin\text{-}coeff\ (n+1) *_R z\hat{}n)\ sums\ (sin\ z\ /\ z)$
⟨*proof*⟩

lemma *has-field-derivative-sin-z-over-z*:
fixes $A :: 'a :: \{real\text{-}normed\text{-}field, banach\}\ set$
shows $((\lambda z.\ if\ z = 0\ then\ 1\ else\ sin\ z\ /\ z)\ has\text{-}field\text{-}derivative\ 0)\ (at\ 0\ within\ A)$
(**is** (*?f has-field-derivative ?f'*) -)
⟨*proof*⟩

lemma *round-Re-minimises-norm*:
$norm\ ((z::complex) - of\text{-}int\ m) \geq norm\ (z - of\text{-}int\ (round\ (Re\ z)))$
⟨*proof*⟩

lemma *Re-pos-in-ball*:
assumes $Re\ z > 0\ t \in ball\ z\ (Re\ z/2)$
shows $Re\ t > 0$
⟨*proof*⟩

lemma *no-nonpos-Int-in-ball-complex*:
assumes $Re\ z > 0\ t \in ball\ z\ (Re\ z/2)$
shows $t \notin \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *no-nonpos-Int-in-ball*:
assumes $t \in ball\ z\ (dist\ z\ (round\ (Re\ z)))$
shows $t \notin \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *no-nonpos-Int-in-ball'*:
assumes $(z :: 'a :: \{euclidean\text{-}space, real\text{-}normed\text{-}algebra\text{-}1\}) \notin \mathbb{Z}_{\leq 0}$
obtains d **where** $d > 0\ \bigwedge t.\ t \in ball\ z\ d \Longrightarrow t \notin \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *no-nonpos-Real-in-ball*:
assumes z: $z \notin \mathbb{R}_{\leq 0}$ **and** t: $t \in ball\ z\ (if\ Im\ z = 0\ then\ Re\ z\ /\ 2\ else\ abs\ (Im\ z)\ /\ 2)$
shows $t \notin \mathbb{R}_{\leq 0}$
⟨*proof*⟩

59.1 Definitions

We define the Gamma function by first defining its multiplicative inverse *Gamma-inv*. This is more convenient because *Gamma-inv* is entire, which makes proofs of its properties more convenient because one does not have to watch out for discontinuities. (e.g. *Gamma-inv* fulfils $rGamma\ z = z * rGamma\ (z + (1::'a))$ everywhere, whereas *Gamma* does not fulfil the analogous equation on the non-positive integers)

We define the Gamma function (resp. its inverse) in the Euler form. This form has the advantage that it is a relatively simple limit that converges everywhere. The limit at the poles is 0 (due to division by 0). The functional equation $Gamma\ (z + (1::'a)) = z * Gamma\ z$ follows immediately from the definition.

definition *Gamma-series* :: ('a :: {*banach*,*real-normed-field*}) ⇒ *nat* ⇒ 'a **where**
Gamma-series z n = *fact n* ∗ *exp* (*z* ∗ *of-real* (*ln* (*of-nat n*))) / *pochhammer z* (*n*+*1*)

definition *Gamma-series'* :: ('a :: {*banach*,*real-normed-field*}) ⇒ *nat* ⇒ 'a **where**
Gamma-series' z n = *fact* (*n* − *1*) ∗ *exp* (*z* ∗ *of-real* (*ln* (*of-nat n*))) / *pochhammer z n*

definition *rGamma-series* :: ('a :: {*banach*,*real-normed-field*}) ⇒ *nat* ⇒ 'a **where**
rGamma-series z n = *pochhammer z* (*n*+*1*) / (*fact n* ∗ *exp* (*z* ∗ *of-real* (*ln* (*of-nat n*))))

lemma *Gamma-series-altdef*: *Gamma-series z n* = *inverse* (*rGamma-series z n*)
and *rGamma-series-altdef*: *rGamma-series z n* = *inverse* (*Gamma-series z n*)
⟨*proof*⟩

lemma *rGamma-series-minus-of-nat*:
eventually (λ*n*. *rGamma-series* (− *of-nat k*) *n* = *0*) *sequentially*
⟨*proof*⟩

lemma *Gamma-series-minus-of-nat*:
eventually (λ*n*. *Gamma-series* (− *of-nat k*) *n* = *0*) *sequentially*
⟨*proof*⟩

lemma *Gamma-series'-minus-of-nat*:
eventually (λ*n*. *Gamma-series'* (− *of-nat k*) *n* = *0*) *sequentially*
⟨*proof*⟩

lemma *rGamma-series-nonpos-Ints-LIMSEQ*: $z \in \mathbb{Z}_{\leq 0} \Longrightarrow$ *rGamma-series z* ⟶ *0*
⟨*proof*⟩

lemma *Gamma-series-nonpos-Ints-LIMSEQ*: $z \in \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma-series z* ⟶ *0*
⟨*proof*⟩

lemma *Gamma-series'-nonpos-Ints-LIMSEQ*: $z \in \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma-series' z* $\longrightarrow$ *0*
⟨*proof*⟩

lemma *Gamma-series-Gamma-series'*:
assumes *z*: $z \notin \mathbb{Z}_{\leq 0}$
shows (λ*n*. *Gamma-series' z n / Gamma-series z n*) $\longrightarrow$ *1*
⟨*proof*⟩

59.2 Convergence of the Euler series form

We now show that the series that defines the Gamma function in the Euler form converges and that the function defined by it is continuous on the complex halfspace with positive real part.

We do this by showing that the logarithm of the Euler series is continuous and converges locally uniformly, which means that the log-Gamma function defined by its limit is also continuous.

This will later allow us to lift holomorphicity and continuity from the log-Gamma function to the inverse of the Gamma function, and from that to the Gamma function itself.

definition *ln-Gamma-series* :: ('*a* :: {*banach*,*real-normed-field*,*ln*}) ⇒ *nat* ⇒ '*a*
where
ln-Gamma-series z n = *z* ∗ *ln* (*of-nat n*) − *ln z* − (∑*k*=*1*..*n*. *ln* (*z* / *of-nat k* + *1*))

definition *ln-Gamma-series'* :: ('*a* :: {*banach*,*real-normed-field*,*ln*}) ⇒ *nat* ⇒ '*a*
where
ln-Gamma-series' z n =
− *euler-mascheroni*∗*z* − *ln z* + (∑*k*=*1*..*n*. *z* / *of-nat n* − *ln* (*z* / *of-nat k* + *1*))

definition *ln-Gamma* :: ('*a* :: {*banach*,*real-normed-field*,*ln*}) ⇒ '*a* **where**
ln-Gamma z = *lim* (*ln-Gamma-series z*)

We now show that the log-Gamma series converges locally uniformly for all complex numbers except the non-positive integers. We do this by proving that the series is locally Cauchy, adapting this proof: http://math.stackexchange.com/questions/88715
of-gammaz-lim-n-to-infty-fracnzn-prod-m-0nzm

context
begin

private lemma *ln-Gamma-series-complex-converges-aux*:
fixes *z* :: *complex* **and** *k* :: *nat*
assumes *z*: *z* ≠ *0* **and** *k*: *of-nat k* ≥ *2*∗*norm z k* ≥ *2*
shows *norm* (*z* ∗ *ln* (*1* − *1*/*of-nat k*) + *ln* (*z*/*of-nat k* + *1*)) ≤ *2*∗(*norm z* + *norm z*^*2*) / *of-nat k*^*2*

⟨*proof*⟩

lemma *ln-Gamma-series-complex-converges*:
assumes *z*: $z \notin \mathbb{Z}_{\leq 0}$
assumes *d*: $d > 0 \bigwedge n.\ n \in \mathbb{Z}_{\leq 0} \Longrightarrow norm\ (z - of\text{-}int\ n) > d$
shows *uniformly-convergent-on* (*ball z d*) (λ*n z*. *ln-Gamma-series z n* :: *complex*)
⟨*proof*⟩

end

lemma *ln-Gamma-series-complex-converges'*:
assumes *z*: (*z* :: *complex*) $\notin \mathbb{Z}_{\leq 0}$
shows ∃ *d*>*0*. *uniformly-convergent-on* (*ball z d*) (λ*n z*. *ln-Gamma-series z n*)
⟨*proof*⟩

lemma *ln-Gamma-series-complex-converges''*: (*z* :: *complex*) $\notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *convergent* (*ln-Gamma-series z*)
⟨*proof*⟩

lemma *ln-Gamma-complex-LIMSEQ*: (*z* :: *complex*) $\notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *ln-Gamma-series z* ⟶ *ln-Gamma z*
⟨*proof*⟩

lemma *exp-ln-Gamma-series-complex*:
assumes *n* > *0* *z* $\notin \mathbb{Z}_{\leq 0}$
shows *exp* (*ln-Gamma-series z n* :: *complex*) = *Gamma-series z n*
⟨*proof*⟩

lemma *ln-Gamma-series'-aux*:
assumes (*z*::*complex*) $\notin \mathbb{Z}_{\leq 0}$
shows (λ*k*. *z* / *of-nat* (*Suc k*) − *ln* (*1* + *z* / *of-nat* (*Suc k*))) *sums*
(*ln-Gamma z* + *euler-mascheroni* ∗ *z* + *ln z*) (**is** *?f sums ?s*)
⟨*proof*⟩

lemma *uniformly-summable-deriv-ln-Gamma*:
assumes *z*: (*z* :: *'a* :: {*real-normed-field*,*banach*}) ≠ *0* **and** *d*: *d* > *0* *d* ≤ *norm z*/*2*
shows *uniformly-convergent-on* (*ball z d*)
(λ*k z*. $\sum i{<}k$. *inverse* (*of-nat* (*Suc i*)) − *inverse* (*z* + *of-nat* (*Suc i*)))
(**is** *uniformly-convergent-on* - (λ*k z*. $\sum i{<}k$. *?f i z*))
⟨*proof*⟩

lemma *summable-deriv-ln-Gamma*:
z ≠ (*0* :: *'a* :: {*real-normed-field*,*banach*}) ⟹
summable (λ*n*. *inverse* (*of-nat* (*Suc n*)) − *inverse* (*z* + *of-nat* (*Suc n*)))
⟨*proof*⟩

definition *Polygamma* :: *nat* ⇒ ('*a* :: {*real-normed-field*,*banach*}) ⇒ '*a* **where**
Polygamma n z = (*if n* = *0 then*
(∑ *k*. *inverse* (*of-nat* (*Suc k*)) − *inverse* (*z* + *of-nat k*)) − *euler-mascheroni*
else
(−*1*) ^ *Suc n* ∗ *fact n* ∗ (∑ *k*. *inverse* ((*z* + *of-nat k*) ^ *Suc n*)))

abbreviation *Digamma* :: ('*a* :: {*real-normed-field*,*banach*}) ⇒ '*a* **where**
Digamma ≡ *Polygamma 0*

lemma *Digamma-def*:
Digamma z = (∑ *k*. *inverse* (*of-nat* (*Suc k*)) − *inverse* (*z* + *of-nat k*)) −
euler-mascheroni
⟨*proof*⟩

lemma *summable-Digamma*:
assumes (*z* :: '*a* :: {*real-normed-field*,*banach*}) ≠ *0*
shows *summable* (λ*n*. *inverse* (*of-nat* (*Suc n*)) − *inverse* (*z* + *of-nat n*))
⟨*proof*⟩

lemma *summable-offset*:
assumes *summable* (λ*n*. *f* (*n* + *k*) :: '*a* :: *real-normed-vector*)
shows *summable f*
⟨*proof*⟩

lemma *Polygamma-converges*:
fixes *z* :: '*a* :: {*real-normed-field*,*banach*}
assumes *z*: *z* ≠ *0* **and** *n*: *n* ≥ *2*
shows *uniformly-convergent-on* (*ball z d*) (λ*k z*. ∑ *i*<*k*. *inverse* ((*z* + *of-nat*
i) ^ *n*))
⟨*proof*⟩

lemma *Polygamma-converges'*:
fixes *z* :: '*a* :: {*real-normed-field*,*banach*}
assumes *z*: *z* ≠ *0* **and** *n*: *n* ≥ *2*
shows *summable* (λ*k*. *inverse* ((*z* + *of-nat k*) ^ *n*))
⟨*proof*⟩

lemma *has-field-derivative-ln-Gamma-complex* [*derivative-intros*]:
fixes *z* :: *complex*
assumes *z*: *z* ∉ $\mathbb{R}_{\leq 0}$
shows (*ln-Gamma has-field-derivative Digamma z*) (*at z*)
⟨*proof*⟩

declare *has-field-derivative-ln-Gamma-complex*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *Digamma-1* [*simp*]: *Digamma* (*1* :: '*a* :: {*real-normed-field*,*banach*}) = −

euler-mascheroni
⟨*proof*⟩

lemma *Digamma-plus1*:
assumes $z \neq 0$
shows *Digamma* $(z+1)$ = *Digamma* $z + 1/z$
⟨*proof*⟩

lemma *Polygamma-plus1*:
assumes $z \neq 0$
shows *Polygamma* $n\ (z + 1)$ = *Polygamma* $n\ z + (-1)\hat{}n * fact\ n\ /\ (z\ \hat{}\ Suc\ n)$
⟨*proof*⟩

lemma *Digamma-of-nat*:
Digamma (*of-nat* (*Suc* n) :: $'a$ :: {*real-normed-field*,*banach*}) = *harm* n − *euler-mascheroni*
⟨*proof*⟩

lemma *Digamma-numeral*: *Digamma* (*numeral* n) = *harm* (*pred-numeral* n) − *euler-mascheroni*
⟨*proof*⟩

lemma *Polygamma-of-real*: $x \neq 0 \Longrightarrow$ *Polygamma* n (*of-real* x) = *of-real* (*Polygamma* $n\ x$)
⟨*proof*⟩

lemma *Polygamma-Real*: $z \in \mathbb{R} \Longrightarrow z \neq 0 \Longrightarrow$ *Polygamma* $n\ z \in \mathbb{R}$
⟨*proof*⟩

lemma *Digamma-half-integer*:
Digamma (*of-nat* n + 1/2 :: $'a$ :: {*real-normed-field*,*banach*}) =
$(\sum k<n.\ 2\ /\ (of\text{-}nat\ (2*k+1)))$ − *euler-mascheroni* − *of-real* $(2 * ln\ 2)$
⟨*proof*⟩

lemma *Digamma-one-half*: *Digamma* $(1/2)$ = − *euler-mascheroni* − *of-real* $(2 * ln\ 2)$
⟨*proof*⟩

lemma *Digamma-real-three-halves-pos*: *Digamma* (3/2 :: *real*) > 0
⟨*proof*⟩

lemma *has-field-derivative-Polygamma* [*derivative-intros*]:
fixes z :: $'a$ :: {*real-normed-field*,*euclidean-space*}
assumes z: $z \notin \mathbb{Z}_{\leq 0}$
shows (*Polygamma n has-field-derivative Polygamma* (*Suc n*) z) (*at z within A*)
⟨*proof*⟩

declare *has-field-derivative-Polygamma*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *isCont-Polygamma* [*continuous-intros*]:
fixes $f :: - \Rightarrow 'a :: \{real\text{-}normed\text{-}field, euclidean\text{-}space\}$
shows $isCont\ f\ z \Longrightarrow f\ z \notin \mathbb{Z}_{\leq 0} \Longrightarrow isCont\ (\lambda x.\ Polygamma\ n\ (f\ x))\ z$
⟨*proof*⟩

lemma *continuous-on-Polygamma*:
$A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow continuous\text{-}on\ A\ (Polygamma\ n :: - \Rightarrow 'a :: \{real\text{-}normed\text{-}field, euclidean\text{-}space\})$
⟨*proof*⟩

lemma *isCont-ln-Gamma-complex* [*continuous-intros*]:
fixes $f :: 'a :: t2\text{-}space \Rightarrow complex$
shows $isCont\ f\ z \Longrightarrow f\ z \notin \mathbb{R}_{\leq 0} \Longrightarrow isCont\ (\lambda z.\ ln\text{-}Gamma\ (f\ z))\ z$
⟨*proof*⟩

lemma *continuous-on-ln-Gamma-complex* [*continuous-intros*]:
fixes $A :: complex\ set$
shows $A \cap \mathbb{R}_{\leq 0} = \{\} \Longrightarrow continuous\text{-}on\ A\ ln\text{-}Gamma$
⟨*proof*⟩

We define a type class that captures all the fundamental properties of the inverse of the Gamma function and defines the Gamma function upon that. This allows us to instantiate the type class both for the reals and for the complex numbers with a minimal amount of proof duplication.

class *Gamma* = *real-normed-field* + *complete-space* +
fixes $rGamma :: 'a \Rightarrow 'a$
assumes *rGamma-eq-zero-iff-aux*: $rGamma\ z = 0 \longleftrightarrow (\exists n.\ z = -\ of\text{-}nat\ n)$
assumes *differentiable-rGamma-aux1*:
$(\bigwedge n.\ z \neq -\ of\text{-}nat\ n) \Longrightarrow$
let $d = (THE\ d.\ (\lambda n.\ \sum k{<}n.\ inverse\ (of\text{-}nat\ (Suc\ k)) - inverse\ (z + of\text{-}nat\ k)) \longrightarrow d) - scaleR\ euler\text{-}mascheroni\ 1$
in $filterlim\ (\lambda y.\ (rGamma\ y - rGamma\ z + rGamma\ z * d * (y - z)) /_R\ norm\ (y - z))\ (nhds\ 0)\ (at\ z)$
assumes *differentiable-rGamma-aux2*:
let $z = -\ of\text{-}nat\ n$
in $filterlim\ (\lambda y.\ (rGamma\ y - rGamma\ z - (-1)\hat{}n * (setprod\ of\text{-}nat\ \{1..n\}) * (y - z)) /_R\ norm\ (y - z))\ (nhds\ 0)\ (at\ z)$
assumes *rGamma-series-aux*: $(\bigwedge n.\ z \neq -\ of\text{-}nat\ n) \Longrightarrow$
let $fact' = (\lambda n.\ setprod\ of\text{-}nat\ \{1..n\})$;
$exp = (\lambda x.\ THE\ e.\ (\lambda n.\ \sum k{<}n.\ x\hat{}k\ /_R\ fact\ k) \longrightarrow e)$;
$pochhammer' = (\lambda a\ n.\ (\prod n = 0..n.\ a + of\text{-}nat\ n))$
in $filterlim\ (\lambda n.\ pochhammer'\ z\ n\ /\ (fact'\ n * exp\ (z * (ln\ (of\text{-}nat\ n) *_R 1))))$
$(nhds\ (rGamma\ z))\ sequentially$
begin
subclass *banach* ⟨*proof*⟩
end

definition *Gamma z = inverse (rGamma z)*

59.3 Basic properties

lemma *Gamma-nonpos-Int*: $z \in \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma z = 0*
and *rGamma-nonpos-Int*: $z \in \mathbb{Z}_{\leq 0} \Longrightarrow$ *rGamma z = 0*
⟨*proof*⟩

lemma *Gamma-nonzero*: $z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma z ≠ 0*
and *rGamma-nonzero*: $z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *rGamma z ≠ 0*
⟨*proof*⟩

lemma *Gamma-eq-zero-iff*: *Gamma z = 0* $\longleftrightarrow$ $z \in \mathbb{Z}_{\leq 0}$
and *rGamma-eq-zero-iff*: *rGamma z = 0* $\longleftrightarrow$ $z \in \mathbb{Z}_{\leq 0}$
⟨*proof*⟩

lemma *rGamma-inverse-Gamma*: *rGamma z = inverse (Gamma z)*
⟨*proof*⟩

lemma *rGamma-series-LIMSEQ* [*tendsto-intros*]:
rGamma-series z $\longrightarrow$ *rGamma z*
⟨*proof*⟩

lemma *Gamma-series-LIMSEQ* [*tendsto-intros*]:
Gamma-series z $\longrightarrow$ *Gamma z*
⟨*proof*⟩

lemma *Gamma-altdef*: *Gamma z = lim (Gamma-series z)*
⟨*proof*⟩

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

lemma *rGamma-plus1*: *z * rGamma (z + 1) = rGamma z*
⟨*proof*⟩

lemma *pochhammer-rGamma*: *rGamma z = pochhammer z n * rGamma (z + of-nat n)*
⟨*proof*⟩

lemma *Gamma-plus1*: $z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma (z + 1) = z * Gamma z*
⟨*proof*⟩

lemma *pochhammer-Gamma*: $z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *pochhammer z n = Gamma (z + of-nat n) / Gamma z*
⟨*proof*⟩

lemma *Gamma-0* [*simp*]: *Gamma 0 = 0*
and *rGamma-0* [*simp*]: *rGamma 0 = 0*
and *Gamma-neg-1* [*simp*]: *Gamma (− 1) = 0*
and *rGamma-neg-1* [*simp*]: *rGamma (− 1) = 0*
and *Gamma-neg-numeral* [*simp*]: *Gamma (− numeral n) = 0*
and *rGamma-neg-numeral* [*simp*]: *rGamma (− numeral n) = 0*
and *Gamma-neg-of-nat* [*simp*]: *Gamma (− of-nat m) = 0*
and *rGamma-neg-of-nat* [*simp*]: *rGamma (− of-nat m) = 0*
⟨*proof*⟩

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

lemma *Gamma-fact*: *Gamma (of-nat (Suc n)) = fact n*
⟨*proof*⟩

lemma *Gamma-numeral*: *Gamma (numeral n) = fact (pred-numeral n)*
⟨*proof*⟩

lemma *Gamma-of-int*: *Gamma (of-int n) = (if n > 0 then fact (nat (n − 1)) else 0)*
⟨*proof*⟩

lemma *rGamma-of-int*: *rGamma (of-int n) = (if n > 0 then inverse (fact (nat (n − 1))) else 0)*
⟨*proof*⟩

lemma *Gamma-seriesI*:
assumes *(λn. g n / Gamma-series z n) ⟶ 1*
shows *g ⟶ Gamma z*
⟨*proof*⟩

lemma *Gamma-seriesI'*:
assumes *f ⟶ rGamma z*
assumes *(λn. g n * f n) ⟶ 1*
assumes *z ∉ ℤ≤0*
shows *g ⟶ Gamma z*
⟨*proof*⟩

lemma *Gamma-series'-LIMSEQ*: *Gamma-series' z ⟶ Gamma z*
⟨*proof*⟩

59.4 Differentiability

lemma *has-field-derivative-rGamma-no-nonpos-int*:
assumes *z ∉ ℤ≤0*
shows *(rGamma has-field-derivative −rGamma z * Digamma z) (at z within A)*
⟨*proof*⟩

lemma *has-field-derivative-rGamma-nonpos-int*:
(*rGamma has-field-derivative* $(-1)\hat{}n * fact\ n$) (*at* $(-\ of\text{-}nat\ n)$ *within A*)
⟨*proof*⟩

lemma *has-field-derivative-rGamma* [*derivative-intros*]:
(*rGamma has-field-derivative* (*if* $z \in \mathbb{Z}_{\leq 0}$ *then* $(-1)\hat{}(nat\ \lfloor norm\ z \rfloor) * fact\ (nat\ \lfloor norm\ z \rfloor)$
else $-rGamma\ z * Digamma\ z$)) (*at z within A*)
⟨*proof*⟩

declare *has-field-derivative-rGamma-no-nonpos-int* [*THEN DERIV-chain2*, *derivative-intros*]
declare *has-field-derivative-rGamma* [*THEN DERIV-chain2*, *derivative-intros*]
declare *has-field-derivative-rGamma-nonpos-int* [*derivative-intros*]
declare *has-field-derivative-rGamma-no-nonpos-int* [*derivative-intros*]
declare *has-field-derivative-rGamma* [*derivative-intros*]

lemma *has-field-derivative-Gamma* [*derivative-intros*]:
$z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ (*Gamma has-field-derivative Gamma z* $*$ *Digamma z*) (*at z within A*)
⟨*proof*⟩

declare *has-field-derivative-Gamma*[*THEN DERIV-chain2*, *derivative-intros*]

hide-fact *rGamma-eq-zero-iff-aux differentiable-rGamma-aux1 differentiable-rGamma-aux2 differentiable-rGamma-aux2 rGamma-series-aux Gamma-class.rGamma-eq-zero-iff-aux*

59.5 Continuity

lemma *continuous-on-rGamma* [*continuous-intros*]: *continuous-on A rGamma*
⟨*proof*⟩

lemma *continuous-on-Gamma* [*continuous-intros*]: $A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow$ *continuous-on A Gamma*
⟨*proof*⟩

lemma *isCont-rGamma* [*continuous-intros*]:
isCont f z $\Longrightarrow$ *isCont* $(\lambda x.\ rGamma\ (f\ x))\ z$
⟨*proof*⟩

lemma *isCont-Gamma* [*continuous-intros*]:
isCont f z $\Longrightarrow f\ z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *isCont* $(\lambda x.\ Gamma\ (f\ x))\ z$
⟨*proof*⟩

The complex Gamma function

instantiation *complex* :: *Gamma*
begin

definition *rGamma-complex* :: *complex* ⇒ *complex* **where**
rGamma-complex z = *lim* (*rGamma-series z*)

lemma *rGamma-series-complex-converges*:
convergent (*rGamma-series* (*z* :: *complex*)) (**is** *?thesis1*)
and *rGamma-complex-altdef*:
rGamma z = (*if z* ∈ $\mathbb{Z}_{\leq 0}$ *then 0 else exp* (−*ln-Gamma z*)) (**is** *?thesis2*)
⟨*proof*⟩

context
begin

private lemma *rGamma-complex-plus1*: *z* ∗ *rGamma* (*z* + *1*) = *rGamma* (*z* :: *complex*)
⟨*proof*⟩ **lemma** *has-field-derivative-rGamma-complex-no-nonpos-Int*:
assumes (*z* :: *complex*) ∉ $\mathbb{Z}_{\leq 0}$
shows (*rGamma has-field-derivative* − *rGamma z* ∗ *Digamma z*) (*at z*)
⟨*proof*⟩ **lemma** *rGamma-complex-1*: *rGamma* (*1* :: *complex*) = *1*
⟨*proof*⟩ **lemma** *has-field-derivative-rGamma-complex-nonpos-Int*:
(*rGamma has-field-derivative* (−*1*) ^*n* ∗ *fact n*) (*at* (− *of-nat n* :: *complex*))
⟨*proof*⟩

instance ⟨*proof*⟩

end
end

lemma *Gamma-complex-altdef*:
Gamma z = (*if z* ∈ $\mathbb{Z}_{\leq 0}$ *then 0 else exp* (*ln-Gamma* (*z* :: *complex*)))
⟨*proof*⟩

lemma *cnj-rGamma*: *cnj* (*rGamma z*) = *rGamma* (*cnj z*)
⟨*proof*⟩

lemma *cnj-Gamma*: *cnj* (*Gamma z*) = *Gamma* (*cnj z*)
⟨*proof*⟩

lemma *Gamma-complex-real*:
z ∈ ℝ ⟹ *Gamma z* ∈ (ℝ :: *complex set*) **and** *rGamma-complex-real*: *z* ∈ ℝ
⟹ *rGamma z* ∈ ℝ
⟨*proof*⟩

lemma *field-differentiable-rGamma*: *rGamma field-differentiable* (*at z within A*)
⟨*proof*⟩

lemma *holomorphic-on-rGamma*: *rGamma holomorphic-on A*
⟨*proof*⟩

lemma *analytic-on-rGamma*: *rGamma analytic-on A*
⟨*proof*⟩

lemma *field-differentiable-Gamma*: $z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *Gamma field-differentiable* (*at z within A*)
⟨*proof*⟩

lemma *holomorphic-on-Gamma*: $A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow$ *Gamma holomorphic-on A*
⟨*proof*⟩

lemma *analytic-on-Gamma*: $A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow$ *Gamma analytic-on A*
⟨*proof*⟩

lemma *has-field-derivative-rGamma-complex'* [*derivative-intros*]:
(*rGamma has-field-derivative* (*if z* ∈ $\mathbb{Z}_{\leq 0}$ *then* (−1) ^ (*nat* ⌊−*Re z*⌋) ∗ *fact* (*nat* ⌊−*Re z*⌋) *else*
−*rGamma z* ∗ *Digamma z*)) (*at z within A*)
⟨*proof*⟩

declare *has-field-derivative-rGamma-complex'*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *field-differentiable-Polygamma*:
fixes *z*::*complex*
shows
$z \notin \mathbb{Z}_{\leq 0} \Longrightarrow$ *Polygamma n field-differentiable* (*at z within A*)
⟨*proof*⟩

lemma *holomorphic-on-Polygamma*: $A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow$ *Polygamma n holomorphic-on A*
⟨*proof*⟩

lemma *analytic-on-Polygamma*: $A \cap \mathbb{Z}_{\leq 0} = \{\} \Longrightarrow$ *Polygamma n analytic-on A*
⟨*proof*⟩

The real Gamma function

lemma *rGamma-series-real*:
eventually (λ*n*. *rGamma-series x n* = *Re* (*rGamma-series* (*of-real x*) *n*)) *sequentially*
⟨*proof*⟩

instantiation *real* :: *Gamma*
begin

definition *rGamma-real x* = *Re* (*rGamma* (*of-real x* :: *complex*))

instance ⟨*proof*⟩

end

lemma *rGamma-complex-of-real*: *rGamma* (*complex-of-real x*) = *complex-of-real* (*rGamma x*)
⟨*proof*⟩

lemma *Gamma-complex-of-real*: *Gamma* (*complex-of-real x*) = *complex-of-real* (*Gamma x*)
⟨*proof*⟩

lemma *rGamma-real-altdef*: *rGamma x* = *lim* (*rGamma-series* (*x* :: *real*))
⟨*proof*⟩

lemma *Gamma-real-altdef1*: *Gamma x* = *lim* (*Gamma-series* (*x* :: *real*))
⟨*proof*⟩

lemma *Gamma-real-altdef2*: *Gamma x* = *Re* (*Gamma* (*of-real x*))
⟨*proof*⟩

lemma *ln-Gamma-series-complex-of-real*:
x > *0* ⟹ *n* > *0* ⟹ *ln-Gamma-series* (*complex-of-real x*) *n* = *of-real* (*ln-Gamma-series x n*)
⟨*proof*⟩

lemma *ln-Gamma-real-converges*:
assumes (*x*::*real*) > *0*
shows *convergent* (*ln-Gamma-series x*)
⟨*proof*⟩

lemma *ln-Gamma-real-LIMSEQ*: (*x*::*real*) > *0* ⟹ *ln-Gamma-series x* ⟶ *ln-Gamma x*
⟨*proof*⟩

lemma *ln-Gamma-complex-of-real*: *x* > *0* ⟹ *ln-Gamma* (*complex-of-real x*) = *of-real* (*ln-Gamma x*)
⟨*proof*⟩

lemma *Gamma-real-pos-exp*: *x* > (*0* :: *real*) ⟹ *Gamma x* = *exp* (*ln-Gamma x*)
⟨*proof*⟩

lemma *ln-Gamma-real-pos*: *x* > *0* ⟹ *ln-Gamma x* = *ln* (*Gamma x* :: *real*)
⟨*proof*⟩

lemma *Gamma-real-pos*: *x* > (*0*::*real*) ⟹ *Gamma x* > *0*
⟨*proof*⟩

lemma *has-field-derivative-ln-Gamma-real* [*derivative-intros*]:
assumes *x*: *x* > (*0*::*real*)

shows (*ln-Gamma has-field-derivative Digamma x*) (*at x*)
⟨*proof*⟩

declare *has-field-derivative-ln-Gamma-real*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *has-field-derivative-rGamma-real'* [*derivative-intros*]:
(*rGamma has-field-derivative* (*if x* ∈ $\mathbb{Z}_{\leq 0}$ *then* (−*1*) ^ (*nat* ⌊−*x*⌋) ∗ *fact* (*nat* ⌊−*x*⌋) *else*
−*rGamma x* ∗ *Digamma x*)) (*at x within A*)
⟨*proof*⟩

declare *has-field-derivative-rGamma-real'*[*THEN DERIV-chain2*, *derivative-intros*]

lemma *Polygamma-real-odd-pos*:
assumes (*x*::*real*) ∉ $\mathbb{Z}_{\leq 0}$ *odd n*
shows *Polygamma n x* > *0*
⟨*proof*⟩

lemma *Polygamma-real-even-neg*:
assumes (*x*::*real*) > *0 n* > *0 even n*
shows *Polygamma n x* < *0*
⟨*proof*⟩

lemma *Polygamma-real-strict-mono*:
assumes *x* > *0 x* < (*y*::*real*) *even n*
shows *Polygamma n x* < *Polygamma n y*
⟨*proof*⟩

lemma *Polygamma-real-strict-antimono*:
assumes *x* > *0 x* < (*y*::*real*) *odd n*
shows *Polygamma n x* > *Polygamma n y*
⟨*proof*⟩

lemma *Polygamma-real-mono*:
assumes *x* > *0 x* ≤ (*y*::*real*) *even n*
shows *Polygamma n x* ≤ *Polygamma n y*
⟨*proof*⟩

lemma *Digamma-real-ge-three-halves-pos*:
assumes *x* ≥ *3*/*2*
shows *Digamma* (*x* :: *real*) > *0*
⟨*proof*⟩

lemma *ln-Gamma-real-strict-mono*:
assumes *x* ≥ *3*/*2 x* < *y*
shows *ln-Gamma* (*x* :: *real*) < *ln-Gamma y*
⟨*proof*⟩

lemma *Gamma-real-strict-mono*:
assumes $x \geq 3/2$ $x < y$
shows $Gamma\ (x :: real) < Gamma\ y$
⟨*proof*⟩

lemma *log-convex-Gamma-real*: *convex-on* $\{0<..\}$ $(ln \circ Gamma :: real \Rightarrow real)$
⟨*proof*⟩

59.6 Beta function

definition *Beta* **where** $Beta\ a\ b = Gamma\ a * Gamma\ b\ /\ Gamma\ (a + b)$

lemma *Beta-altdef*: $Beta\ a\ b = Gamma\ a * Gamma\ b * rGamma\ (a + b)$
⟨*proof*⟩

lemma *Beta-commute*: $Beta\ a\ b = Beta\ b\ a$
⟨*proof*⟩

lemma *has-field-derivative-Beta1* [*derivative-intros*]:
assumes $x \notin \mathbb{Z}_{\leq 0}$ $x + y \notin \mathbb{Z}_{\leq 0}$
shows $((\lambda x.\ Beta\ x\ y)\ has\text{-}field\text{-}derivative\ (Beta\ x\ y * (Digamma\ x - Digamma\ (x + y))))$
$(at\ x\ within\ A)$ ⟨*proof*⟩

lemma *has-field-derivative-Beta2* [*derivative-intros*]:
assumes $y \notin \mathbb{Z}_{\leq 0}$ $x + y \notin \mathbb{Z}_{\leq 0}$
shows $((\lambda y.\ Beta\ x\ y)\ has\text{-}field\text{-}derivative\ (Beta\ x\ y * (Digamma\ y - Digamma\ (x + y))))$
$(at\ y\ within\ A)$
⟨*proof*⟩

lemma *Beta-plus1-plus1*:
assumes $x \notin \mathbb{Z}_{\leq 0}$ $y \notin \mathbb{Z}_{\leq 0}$
shows $Beta\ (x + 1)\ y + Beta\ x\ (y + 1) = Beta\ x\ y$
⟨*proof*⟩

lemma *Beta-plus1-left*:
assumes $x \notin \mathbb{Z}_{\leq 0}$ $y \notin \mathbb{Z}_{\leq 0}$
shows $(x + y) * Beta\ (x + 1)\ y = x * Beta\ x\ y$
⟨*proof*⟩

lemma *Beta-plus1-right*:
assumes $x \notin \mathbb{Z}_{\leq 0}$ $y \notin \mathbb{Z}_{\leq 0}$
shows $(x + y) * Beta\ x\ (y + 1) = y * Beta\ x\ y$
⟨*proof*⟩

lemma *Gamma-Gamma-Beta*:
assumes $x \notin \mathbb{Z}_{\leq 0}$ $y \notin \mathbb{Z}_{\leq 0}$ $x + y \notin \mathbb{Z}_{\leq 0}$
shows $Gamma\ x * Gamma\ y = Beta\ x\ y * Gamma\ (x + y)$

⟨*proof*⟩

59.7 Legendre duplication theorem

context
begin

private lemma *Gamma-legendre-duplication-aux*:
fixes $z :: 'a :: Gamma$
assumes $z \notin \mathbb{Z}_{\leq 0}$ $z + 1/2 \notin \mathbb{Z}_{\leq 0}$
shows $Gamma\ z * Gamma\ (z + 1/2) = exp\ ((1 - 2{*}z) * of\text{-}real\ (ln\ 2)) * Gamma\ (1/2) * Gamma\ (2{*}z)$
⟨*proof*⟩ **lemma** *Gamma-reflection-aux*:
defines $h \equiv \lambda z{::}complex.\ if\ z \in \mathbb{Z}\ then\ 0\ else$
$(of\text{-}real\ pi * cot\ (of\text{-}real\ pi{*}z) + Digamma\ z - Digamma\ (1 - z))$
defines $a \equiv complex\text{-}of\text{-}real\ pi$
obtains h' **where** $continuous\text{-}on\ UNIV\ h' \bigwedge z.\ (h\ has\text{-}field\text{-}derivative\ (h'\ z))\ (at\ z)$
⟨*proof*⟩

lemma *Gamma-reflection-complex*:
fixes $z :: complex$
shows $Gamma\ z * Gamma\ (1 - z) = of\text{-}real\ pi\ /\ sin\ (of\text{-}real\ pi * z)$
⟨*proof*⟩

lemma *rGamma-reflection-complex*:
$rGamma\ z * rGamma\ (1 - z :: complex) = sin\ (of\text{-}real\ pi * z)\ /\ of\text{-}real\ pi$
⟨*proof*⟩

lemma *rGamma-reflection-complex'*:
$rGamma\ z * rGamma\ (-\ z :: complex) = -z * sin\ (of\text{-}real\ pi * z)\ /\ of\text{-}real\ pi$
⟨*proof*⟩

lemma *Gamma-reflection-complex'*:
$Gamma\ z * Gamma\ (-\ z :: complex) = -\ of\text{-}real\ pi\ /\ (z * sin\ (of\text{-}real\ pi * z))$
⟨*proof*⟩

lemma *Gamma-one-half-real*: $Gamma\ (1/2 :: real) = sqrt\ pi$
⟨*proof*⟩

lemma *Gamma-one-half-complex*: $Gamma\ (1/2 :: complex) = of\text{-}real\ (sqrt\ pi)$
⟨*proof*⟩

lemma *Gamma-legendre-duplication*:
fixes $z :: complex$
assumes $z \notin \mathbb{Z}_{\leq 0}$ $z + 1/2 \notin \mathbb{Z}_{\leq 0}$
shows $Gamma\ z * Gamma\ (z + 1/2) =$

$exp\ ((1\ -\ 2{*}z)\ *\ of\text{-}real\ (ln\ 2))\ *\ of\text{-}real\ (sqrt\ pi)\ *\ Gamma\ (2{*}z)$
⟨*proof*⟩

end

59.8 Limits and residues

The inverse of the Gamma function has simple zeros:

lemma *rGamma-zeros*:
$(\lambda z.\ rGamma\ z\ /\ (z\ +\ of\text{-}nat\ n))\ -\ (-\ of\text{-}nat\ n)\ \longrightarrow\ ((-1)\ \hat{}\ n\ *\ fact\ n\ ::\ 'a\ ::\ Gamma)$
⟨*proof*⟩

The simple zeros of the inverse of the Gamma function correspond to simple poles of the Gamma function, and their residues can easily be computed from the limit we have just proven:

lemma *Gamma-poles*: $filterlim\ Gamma\ at\text{-}infinity\ (at\ (-\ of\text{-}nat\ n\ ::\ 'a\ ::\ Gamma))$
⟨*proof*⟩

lemma *Gamma-residues*:
$(\lambda z.\ Gamma\ z\ *\ (z\ +\ of\text{-}nat\ n))\ -\ (-\ of\text{-}nat\ n)\ \longrightarrow\ ((-1)\ \hat{}\ n\ /\ fact\ n\ ::\ 'a\ ::\ Gamma)$
⟨*proof*⟩

59.9 Alternative definitions

59.9.1 Variant of the Euler form

definition *Gamma-series-euler'* **where**
Gamma-series-euler' z n =
$inverse\ z\ *\ (\prod k{=}1..n.\ exp\ (z\ *\ of\text{-}real\ (ln\ (1\ +\ inverse\ (of\text{-}nat\ k)))))\ /\ (1\ +\ z\ /\ of\text{-}nat\ k))$

context
begin
private lemma *Gamma-euler'-aux1*:
fixes $z\ ::\ 'a\ ::\ \{real\text{-}normed\text{-}field, banach\}$
assumes *n*: $n\ >\ 0$
shows $exp\ (z\ *\ of\text{-}real\ (ln\ (of\text{-}nat\ n\ +\ 1)))\ =\ (\prod k{=}1..n.\ exp\ (z\ *\ of\text{-}real\ (ln\ (1\ +\ 1\ /\ of\text{-}nat\ k))))$
⟨*proof*⟩

lemma *Gamma-series-euler'*:
assumes *z*: $(z\ ::\ 'a\ ::\ Gamma)\ \notin\ \mathbb{Z}_{\leq 0}$
shows $(\lambda n.\ Gamma\text{-}series\text{-}euler'\ z\ n)\ \longrightarrow\ Gamma\ z$
⟨*proof*⟩

end

59.9.2 Weierstrass form

definition *Gamma-series-weierstrass* :: *'a* :: {*banach*,*real-normed-field*} ⇒ *nat* ⇒ *'a* **where**
Gamma-series-weierstrass z n =
exp (−*euler-mascheroni* ∗ *z*) / *z* ∗ (∏ *k*=*1*..*n*. *exp* (*z* / *of-nat k*) / (*1* + *z* / *of-nat k*))

definition *rGamma-series-weierstrass* :: *'a* :: {*banach*,*real-normed-field*} ⇒ *nat* ⇒ *'a* **where**
rGamma-series-weierstrass z n =
exp (*euler-mascheroni* ∗ *z*) ∗ *z* ∗ (∏ *k*=*1*..*n*. (*1* + *z* / *of-nat k*) ∗ *exp* (−*z* / *of-nat k*))

lemma *Gamma-series-weierstrass-nonpos-Ints*:
eventually (λ*k*. *Gamma-series-weierstrass* (− *of-nat n*) *k* = *0*) *sequentially*
⟨*proof*⟩

lemma *rGamma-series-weierstrass-nonpos-Ints*:
eventually (λ*k*. *rGamma-series-weierstrass* (− *of-nat n*) *k* = *0*) *sequentially*
⟨*proof*⟩

lemma *Gamma-weierstrass-complex*: *Gamma-series-weierstrass z* ⟶ *Gamma* (*z* :: *complex*)
⟨*proof*⟩

lemma *tendsto-complex-of-real-iff*: ((λ*x*. *complex-of-real* (*f x*)) ⟶ *of-real c*) *F* = (*f* ⟶ *c*) *F*
⟨*proof*⟩

lemma *Gamma-weierstrass-real*: *Gamma-series-weierstrass x* ⟶ *Gamma* (*x* :: *real*)
⟨*proof*⟩

lemma *rGamma-weierstrass-complex*: *rGamma-series-weierstrass z* ⟶ *rGamma* (*z* :: *complex*)
⟨*proof*⟩

59.9.3 Binomial coefficient form

lemma *Gamma-binomial*:
(λ*n*. ((*z* + *of-nat n*) *gchoose n*) ∗ *exp* (−*z* ∗ *of-real* (*ln* (*of-nat n*)))) ⟶ *rGamma* (*z*+*1*)
⟨*proof*⟩

lemma *fact-binomial-limit*:
(λ*n*. *of-nat* ((*k* + *n*) *choose n*) / *of-nat* (*n* ^ *k*) :: *'a* :: *Gamma*) ⟶ *1* / *fact k*
⟨*proof*⟩

lemma *binomial-asymptotic*:
(λ*n*. *of-nat* ((*k* + *n*) *choose n*) / (*of-nat* (*n* ^ *k*) / *fact k*) :: *'a* :: *Gamma*) ⟶ *1*
⟨*proof*⟩

59.10 The Weierstra product formula for the sine

lemma *sin-product-formula-complex*:
fixes *z* :: *complex*
shows (λ*n*. *of-real pi* ∗ *z* ∗ (∏ *k*=*1*..*n*. *1* − *z*^*2* / *of-nat k*^*2*)) ⟶ *sin* (*of-real pi* ∗ *z*)
⟨*proof*⟩

lemma *sin-product-formula-real*:
(λ*n*. *pi* ∗ (*x*::*real*) ∗ (∏ *k*=*1*..*n*. *1* − *x*^*2* / *of-nat k*^*2*)) ⟶ *sin* (*pi* ∗ *x*)
⟨*proof*⟩

lemma *sin-product-formula-real'*:
assumes *x* ≠ (*0*::*real*)
shows (λ*n*. (∏ *k*=*1*..*n*. *1* − *x*^*2* / *of-nat k*^*2*)) ⟶ *sin* (*pi* ∗ *x*) / (*pi* ∗ *x*)
⟨*proof*⟩

59.11 The Solution to the Basel problem

theorem *inverse-squares-sums*: (λ*n*. *1* / (*n* + *1*)^2) *sums* (*pi*^2 / *6*)
⟨*proof*⟩

end
theory *Multivariate-Analysis*
imports
Fashoda
Extended-Real-Limits
Determinants
Ordered-Euclidean-Space
Bounded-Continuous-Function
Weierstrass
Conformal-Mappings
Generalised-Binomial-Theorem
Gamma
begin

end

60 polynomial functions: extremal behaviour and root counts

theory *PolyRoots*
imports *Complex-Main*

begin

60.1 Geometric progressions

lemma *setsum-gp-basic*:
fixes x :: *'a*::{*comm-ring*,*monoid-mult*}
shows $(1 - x) * (\sum i{\leq}n.\ x\hat{}i) = 1 - x\hat{}Suc\ n$
⟨*proof*⟩

lemma *setsum-gp0*:
fixes x :: *'a*::{*comm-ring*,*division-ring*}
shows $(\sum i{\leq}n.\ x\hat{}i) = (if\ x = 1\ then\ of\text{-}nat(n + 1)\ else\ (1 - x\hat{}Suc\ n) / (1 - x))$
⟨*proof*⟩

lemma *setsum-power-add*:
fixes x :: *'a*::{*comm-ring*,*monoid-mult*}
shows $(\sum i{\in}I.\ x\hat{}(m+i)) = x\hat{}m * (\sum i{\in}I.\ x\hat{}i)$
⟨*proof*⟩

lemma *setsum-power-shift*:
fixes x :: *'a*::{*comm-ring*,*monoid-mult*}
assumes $m \leq n$
shows $(\sum i{=}m..n.\ x\hat{}i) = x\hat{}m * (\sum i{\leq}n{-}m.\ x\hat{}i)$
⟨*proof*⟩

lemma *setsum-gp-multiplied*:
fixes x :: *'a*::{*comm-ring*,*monoid-mult*}
assumes $m \leq n$
shows $(1 - x) * (\sum i{=}m..n.\ x\hat{}i) = x\hat{}m - x\hat{}Suc\ n$
⟨*proof*⟩

lemma *setsum-gp*:
fixes x :: *'a*::{*comm-ring*,*division-ring*}
shows $(\sum i{=}m..n.\ x\hat{}i) =$
$(if\ n < m\ then\ 0$
$\ else\ if\ x = 1\ then\ of\text{-}nat((n + 1) - m)$
$\ else\ (x\hat{}m - x\hat{}Suc\ n) / (1 - x))$
⟨*proof*⟩

lemma *setsum-gp-offset*:
fixes x :: *'a*::{*comm-ring*,*division-ring*}
shows $(\sum i{=}m..m+n.\ x\hat{}i) =$
$(if\ x = 1\ then\ of\text{-}nat\ n + 1\ else\ x\hat{}m * (1 - x\hat{}Suc\ n) / (1 - x))$
⟨*proof*⟩

lemma *setsum-gp-strict*:
fixes x :: *'a*::{*comm-ring*,*division-ring*}
shows $(\sum i{<}n.\ x\hat{}i) = (if\ x = 1\ then\ of\text{-}nat\ n\ else\ (1 - x\hat{}n) / (1 - x))$

⟨*proof*⟩

60.2 Basics about polynomial functions: extremal behaviour and root counts.

lemma *sub-polyfun*:
fixes $x :: 'a::\{comm\text{-}ring,monoid\text{-}mult\}$
shows $(\sum i{\le}n.\ a\ i * x\hat{}i) - (\sum i{\le}n.\ a\ i * y\hat{}i) =$
$(x - y) * (\sum j{<}n.\ \sum k{=}\ Suc\ j..n.\ a\ k * y\hat{}(k - Suc\ j) * x\hat{}j)$
⟨*proof*⟩

lemma *sub-polyfun-alt*:
fixes $x :: 'a::\{comm\text{-}ring,monoid\text{-}mult\}$
shows $(\sum i{\le}n.\ a\ i * x\hat{}i) - (\sum i{\le}n.\ a\ i * y\hat{}i) =$
$(x - y) * (\sum j{<}n.\ \sum k{<}n{-}j.\ a\ (j{+}k{+}1) * y\hat{}k * x\hat{}j)$
⟨*proof*⟩

lemma *polyfun-linear-factor*:
fixes $a :: 'a::\{comm\text{-}ring,monoid\text{-}mult\}$
shows $\exists b.\ \forall z.\ (\sum i{\le}n.\ c\ i * z\hat{}i) =$
$(z{-}a) * (\sum i{<}n.\ b\ i * z\hat{}i) + (\sum i{\le}n.\ c\ i * a\hat{}i)$
⟨*proof*⟩

lemma *polyfun-linear-factor-root*:
fixes $a :: 'a::\{comm\text{-}ring,monoid\text{-}mult\}$
assumes $(\sum i{\le}n.\ c\ i * a\hat{}i) = 0$
shows $\exists b.\ \forall z.\ (\sum i{\le}n.\ c\ i * z\hat{}i) = (z{-}a) * (\sum i{<}n.\ b\ i * z\hat{}i)$
⟨*proof*⟩

lemma *adhoc-norm-triangle*: $a + norm(y) \le b \Longrightarrow norm(x) \le a \Longrightarrow norm(x + y) \le b$
⟨*proof*⟩

lemma *polyfun-extremal-lemma*:
fixes $c :: nat \Rightarrow 'a::real\text{-}normed\text{-}div\text{-}algebra$
assumes $e > 0$
shows $\exists M.\ \forall z.\ M \le norm\ z \longrightarrow norm(\sum i{\le}n.\ c\ i * z\hat{}i) \le e * norm(z)\ \hat{}\ Suc\ n$
⟨*proof*⟩

lemma *norm-lemma-xy*: **assumes** $|b| + 1 \le norm(y) - a\ norm(x) \le a$ **shows** $b \le norm(x + y)$
⟨*proof*⟩

lemma *polyfun-extremal*:
fixes $c :: nat \Rightarrow 'a::real\text{-}normed\text{-}div\text{-}algebra$
assumes $\exists k.\ k \ne 0 \wedge k \le n \wedge c\ k \ne 0$
shows *eventually* $(\lambda z.\ norm(\sum i{\le}n.\ c\ i * z\hat{}i) \ge B)$ *at-infinity*
⟨*proof*⟩

lemma *polyfun-rootbound*:
fixes $c :: nat \Rightarrow 'a::\{comm\text{-}ring,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\exists k.\ k \leq n \wedge c\ k \neq 0$
shows *finite* $\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0\} \wedge$ *card* $\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0\} \leq n$
⟨*proof*⟩

corollary
fixes $c :: nat \Rightarrow 'a::\{comm\text{-}ring,real\text{-}normed\text{-}div\text{-}algebra\}$
assumes $\exists k.\ k \leq n \wedge c\ k \neq 0$
shows *polyfun-rootbound-finite*: *finite* $\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0\}$
and *polyfun-rootbound-card*: *card* $\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0\} \leq n$
⟨*proof*⟩

lemma *polyfun-finite-roots*:
fixes $c :: nat \Rightarrow 'a::\{comm\text{-}ring,real\text{-}normed\text{-}div\text{-}algebra\}$
shows *finite* $\{z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0\} \longleftrightarrow (\exists k.\ k \leq n \wedge c\ k \neq 0)$
⟨*proof*⟩

lemma *polyfun-eq-0*:
fixes $c :: nat \Rightarrow 'a::\{comm\text{-}ring,real\text{-}normed\text{-}div\text{-}algebra\}$
shows $(\forall z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = 0) \longleftrightarrow (\forall k.\ k \leq n \longrightarrow c\ k = 0)$
⟨*proof*⟩

lemma *polyfun-eq-const*:
fixes $c :: nat \Rightarrow 'a::\{comm\text{-}ring,real\text{-}normed\text{-}div\text{-}algebra\}$
shows $(\forall z.\ (\sum i{\leq}n.\ c\ i * z\hat{}i) = k) \longleftrightarrow c\ 0 = k \wedge (\forall k.\ k \neq 0 \wedge k \leq n \longrightarrow c\ k = 0)$
⟨*proof*⟩

end