L^p spaces in Isabelle

Sebastien Gouezel

Abstract

L^p is the space of functions whose p-th power is integrable. It is one of the most fundamental Banach spaces that is used in analysis and probability. We develop a framework for function spaces, and then implement the L^p spaces in this framework using the existing integration theory in Isabelle/HOL. Our development contains most fundamental properties of L^p spaces, notably the Hölder and Minkowski inequalities, completeness of L^p, duality, stability under almost sure convergence, multiplication of functions in L^p and L^q, stability under conditional expectation.

Contents

theory *Functional-Spaces*
imports
HOL−Analysis.Analysis
HOL−Library.Function-Algebras
Ergodic-Theory.SG-Library-Complement
begin

1 Functions as a real vector space

Many functional spaces are spaces of functions. To be able to use the following framework, spaces of functions thus need to be endowed with a vector space structure, coming from pointwise addition and multiplication.

Some instantiations for `fun` are already given in `Function_Algebras.thy` and `Lattices.thy`, we add several.

`minus_fun` is already defined, in `Lattices.thy`, but under the strange name `fun_Compl_def`. We restate the definition so that `unfolding minus_fun_def` works. Same thing for `minus_fun_def`. A better solution would be to have a coherent naming scheme in `Lattices.thy`.

lemmas *uminus-fun-def* = *fun-Compl-def*
lemmas *minus-fun-def* = *fun-diff-def*

lemma *fun-sum-apply*:
fixes *u*::*'i* ⇒ *'a* ⇒ (*'b*::*comm-monoid-add*)
shows (*sum u I*) *x* = *sum* (λ*i*. *u i x*) *I*
by (*induction I rule*: *infinite-finite-induct*, *auto*)

instantiation *fun* :: (*type*, *real-vector*) *real-vector*
begin

definition *scaleR-fun*::*real* ⇒ (*'a* ⇒ *'b*) ⇒ *'a* ⇒ *'b*
where *scaleR-fun* = (λ*c f*. (λ*x*. *c* ∗*R f x*))

lemma *scaleR-apply* [*simp*, *code*]: (*c* ∗*R f*) *x* = *c* ∗*R* (*f x*)
by (*simp add*: *scaleR-fun-def*)

instance by (*standard*, *auto simp add*: *scaleR-add-right scaleR-add-left*)
end

lemmas *divideR-apply* = *scaleR-apply*

lemma [*measurable*]:

$0 \in$ *borel-measurable M*
unfolding *zero-fun-def* **by** *auto*

lemma *borel-measurable-const-scaleR'* [*measurable* (*raw*)]:
$(f::('a \Rightarrow 'b::real\text{-}normed\text{-}vector)) \in$ *borel-measurable M* $\Longrightarrow c *_R f \in$ *borel-measurable M*
unfolding *scaleR-fun-def* **using** *borel-measurable-add* **by** *auto*

lemma *borel-measurable-add'*[*measurable* (*raw*)]:
fixes $f\ g :: 'a \Rightarrow 'b::\{second\text{-}countable\text{-}topology, real\text{-}normed\text{-}vector\}$
assumes *f*: $f \in$ *borel-measurable M*
assumes *g*: $g \in$ *borel-measurable M*
shows $f + g \in$ *borel-measurable M*
unfolding *plus-fun-def* **using** *assms* **by** *auto*

lemma *borel-measurable-uminus'*[*measurable* (*raw*)]:
fixes $f\ g :: 'a \Rightarrow 'b::\{second\text{-}countable\text{-}topology, real\text{-}normed\text{-}vector\}$
assumes *f*: $f \in$ *borel-measurable M*
shows $-f \in$ *borel-measurable M*
unfolding *fun-Compl-def* **using** *assms* **by** *auto*

lemma *borel-measurable-diff'*[*measurable* (*raw*)]:
fixes $f\ g :: 'a \Rightarrow 'b::\{second\text{-}countable\text{-}topology, real\text{-}normed\text{-}vector\}$
assumes *f*: $f \in$ *borel-measurable M*
assumes *g*: $g \in$ *borel-measurable M*
shows $f - g \in$ *borel-measurable M*
unfolding *fun-diff-def* **using** *assms* **by** *auto*

lemma *borel-measurable-sum'*[*measurable* (*raw*)]:
fixes $f::'i \Rightarrow 'a \Rightarrow 'b::\{second\text{-}countable\text{-}topology, real\text{-}normed\text{-}vector\}$
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in$ *borel-measurable M*
shows $(\sum i \in I.\ f\ i) \in$ *borel-measurable M*
using *borel-measurable-sum*[*of I f*, *OF assms*] **unfolding** *fun-sum-apply*[*symmetric*]
by *simp*

lemma *zero-applied-to* [*simp*]:
$(0::('a \Rightarrow ('b::real\text{-}vector)))\ x = 0$
unfolding *zero-fun-def* **by** *simp*

2 Quasinorms on function spaces

A central feature of modern analysis is the use of various functional spaces, and of results of functional analysis on them. Think for instance of L^p spaces, of Sobolev or Besov spaces, or variations around them. Here are several relevant facts about this point of view:

- These spaces typically depend on one or several parameters. This makes it difficult to play with type classes in a system without depen-

dent types.

- The L^p spaces are not spaces of functions (their elements are equivalence classes of functions, where two functions are identified if they coincide almost everywhere). However, in usual analysis proofs, one takes a definite representative and works with it, never going to the equivalence class point of view (which only becomes relevant when one wants to use the fact that one has a Banach space at our disposal, to apply functional analytic tools).
- It is important to describe how the spaces are related to each other, with respect to inclusions or compact inclusions. For instance, one of the most important theorems in analysis is Sobolev embedding theorem, describing when one Sobolev space is included in another one. One also needs to be able to take intersections or sums of Banach spaces, for instance to develop interpolation theory.
- Some other spaces play an important role in analysis, for instance the weak L^1 space. This space only has a quasi-norm (i.e., its norm satisfies the triangular inequality up to a fixed multiplicative constant). A general enough setting should also encompass this kind of space. (One could argue that one should also consider more general topologies such as Frechet spaces, to deal with Gevrey or analytic functions. This is true, but considering quasi-norms already gives a wealth of applications).

Given these points, it seems that the most effective way of formalizing this kind of question in Isabelle/HOL is to think of such a functional space not as an abstract space or type, but as a subset of the space of all functions or of all distributions. Functions that do not belong to the functional space under consideration will then have infinite norm. Then inclusions, intersections, and so on, become trivial to implement. Since the same object contains both the information about the norm and the space where the norm is finite, it conforms to the customary habit in mathematics of identifying the two of them, talking for instance about the L^p space and the L^p norm.

All in all, this approach seems quite promising for "real life analysis".

2.1 Definition of quasinorms

typedef (**overloaded**) ('*a*::*real-vector*) *quasinorm* = {(*C*::*real*, *N*::('*a* ⇒ *ennreal*)). (*C* ≥ *1*)
∧ (∀ *x c*. *N* (*c* ∗$_R$ *x*) = *ennreal* |*c*| ∗ *N*(*x*)) ∧ (∀ *x y*. *N*(*x*+*y*) ≤ *C* ∗ *N x* + *C* ∗ *N y*)}
morphisms *Rep-quasinorm quasinorm-of*
proof
show (*1*,(λ*x*. *0*)) ∈ {(*C*::*real*, *N*::('*a* ⇒ *ennreal*)). (*C* ≥ *1*)

$\wedge$ ($\forall$ x c. N (c $*_R$ x) = $ennreal$ $|c|$ $*$ N x) $\wedge$ ($\forall$ x y. N (x+y) $\leq$ C $*$ N x + C $*$ N y)}
by *auto*
qed

definition *eNorm*::'*a quasinorm* $\Rightarrow$ ('*a*::*real-vector*) $\Rightarrow$ *ennreal*
where *eNorm* N x = (*snd* (*Rep-quasinorm* N)) x

definition *defect*::('*a*::*real-vector*) *quasinorm* $\Rightarrow$ *real*
where *defect* N = *fst* (*Rep-quasinorm* N)

lemma *eNorm-triangular-ineq*:
eNorm N (x + y) $\leq$ *defect* N $*$ *eNorm* N x + *defect* N $*$ *eNorm* N y
unfolding *eNorm-def defect-def* **using** *Rep-quasinorm*[*of* N] **by** *auto*

lemma *defect-ge-1*:
defect N $\geq$ *1*
unfolding *defect-def* **using** *Rep-quasinorm*[*of* N] **by** *auto*

lemma *eNorm-cmult*:
eNorm N (c $*_R$ x) = *ennreal* $|c|$ $*$ *eNorm* N x
unfolding *eNorm-def* **using** *Rep-quasinorm*[*of* N] **by** *auto*

lemma *eNorm-zero* [*simp*]:
eNorm N *0* = *0*
by (*metis eNorm-cmult abs-zero ennreal-0 mult-zero-left real-vector.scale-zero-left*)

lemma *eNorm-uminus* [*simp*]:
eNorm N ($-x$) = *eNorm* N x
using *eNorm-cmult*[*of* N -1 x] **by** *auto*

lemma *eNorm-sum*:
eNorm N ($\sum i{\in}\{..{<}n\}$. u i) $\leq$ ($\sum i{\in}\{..{<}n\}$. (*defect* N)^(*Suc* i) $*$ *eNorm* N (u i))
proof (*cases n=0*)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
then obtain m **where** n = *Suc* m **using** *not0-implies-Suc* **by** *blast*
have $\bigwedge v$. *eNorm* N ($\sum i{\in}\{..n\}$. v i) $\leq$ ($\sum i{\in}\{..{<}n\}$. (*defect* N)^(*Suc* i) $*$ *eNorm* N (v i)) + (*defect* N)^n $*$ *eNorm* N (v n) **for** n
proof (*induction n*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
have $*$: (*defect* N)^(*Suc n*) = (*defect* N)^n $*$ *ennreal*(*defect* N)
by (*metis defect-ge-1 ennreal-le-iff ennreal-neg ennreal-power less-le not-less*

not-one-le-zero semiring-normalization-rules(*28*))
fix *v*::*nat* ⇒ *'a*
define *w* **where** *w* = (λ*i*. *if i* = *n then v n* + *v* (*Suc n*) *else v i*)
have (∑ *i*∈{..*Suc n*}. *v i*) = (∑ *i*∈{..<*n*}. *v i*) + *v n* + *v* (*Suc n*)
using *lessThan-Suc-atMost sum.lessThan-Suc* **by** *auto*
also have ... = (∑ *i*∈{..<*n*}. *w i*) + *w n* **unfolding** *w-def* **by** *auto*
finally have (∑ *i*∈{..*Suc n*}. *v i*) = (∑ *i*∈{..*n*}. *w i*)
by (*metis lessThan-Suc-atMost sum.lessThan-Suc*)
then have *eNorm N* (∑ *i*∈{..*Suc n*}. *v i*) = *eNorm N* (∑ *i*∈{..*n*}. *w i*) **by** *simp*
also have ... ≤ (∑ *i*∈{..<*n*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*w i*)) + (*defect N*) ^ *n* * *eNorm N* (*w n*)
using *Suc.IH* **by** *auto*
also have ... = (∑ *i*∈{..<*n*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*v i*)) + (*defect N*) ^ *n* * *eNorm N* (*v n* + *v* (*Suc n*))
unfolding *w-def* **by** *auto*
also have ... ≤ (∑ *i*∈{..<*n*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*v i*)) +
(*defect N*) ^ *n* * (*defect N* * *eNorm N* (*v n*) + *defect N* * *eNorm N* (*v* (*Suc n*)))
by (*rule add-mono*, *simp*, *rule mult-left-mono*, *auto simp add*: *eNorm-triangular-ineq*)
also have ... = (∑ *i*∈{..<*n*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*v i*))
+ (*defect N*) ^ (*Suc n*) * *eNorm N* (*v n*) + (*defect N*) ^ (*Suc n*) * *eNorm N* (*v* (*Suc n*))
unfolding * **by** (*simp add*: *distrib-left semiring-normalization-rules*(*18*))
also have ... = (∑ *i*∈{..<*Suc n*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*v i*)) + (*defect N*) ^ (*Suc n*) * *eNorm N* (*v* (*Suc n*))
by *auto*
finally show *eNorm N* (∑ *i*∈{..*Suc n*}. *v i*)
≤ (∑ *i*<*Suc n*. *ennreal* (*defect N* ^ *Suc i*) * *eNorm N* (*v i*)) + *ennreal* (*defect N* ^ *Suc n*) * *eNorm N* (*v* (*Suc n*))
by *simp*
qed
then have *eNorm N* (∑ *i*∈{..<*Suc m*}. *u i*)
≤ (∑ *i*∈{..<*m*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*u i*)) + (*defect N*) ^ *m* * *eNorm N* (*u m*)
using *lessThan-Suc-atMost* **by** *auto*
also have ... ≤ (∑ *i*∈{..<*m*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*u i*)) + (*defect N*) ^ (*Suc m*) * *eNorm N* (*u m*)
apply (*rule add-mono*, *auto intro*!: *mult-right-mono ennreal-leI*)
using *defect-ge-1* **by** (*metis atMost-iff le-less lessThan-Suc-atMost lessThan-iff power-Suc power-increasing*)
also have ... = (∑ *i*∈{..<*Suc m*}. (*defect N*) ^ (*Suc i*) * *eNorm N* (*u i*))
by *auto*
finally show *eNorm N* (∑ *i*∈{..<*n*}. *u i*) ≤ (∑ *i*<*n*. *ennreal* (*defect N* ^ *Suc i*) * *eNorm N* (*u i*))
unfolding ‹*n* = *Suc m*› **by** *auto*
qed

Quasinorms are often defined by taking a meaningful formula on a vector

subspace, and then extending by infinity elsewhere. Let us show that this results in a quasinorm on the whole space.

definition *quasinorm-on*::('*a set*) ⇒ *real* ⇒ (('*a*::*real-vector*) ⇒ *ennreal*) ⇒ *bool*
where *quasinorm-on F C N* = (
(∀ *x y*. (*x* ∈ *F* ∧ *y* ∈ *F*) ⟶ (*x* + *y* ∈ *F*) ∧ *N* (*x*+*y*) ≤ *C* ∗ *N x* + *C* ∗ *N y*)
∧ (∀ *c x*. *x* ∈ *F* ⟶ *c* ∗$_R$ *x* ∈ *F* ∧ *N*(*c* ∗$_R$ *x*) = |*c*| ∗ *N x*)
∧ *C* ≥ 1 ∧ 0 ∈ *F*)

lemma *quasinorm-of*:
fixes *N*::('*a*::*real-vector*) ⇒ *ennreal* **and** *C*::*real*
assumes *quasinorm-on UNIV C N*
shows *eNorm* (*quasinorm-of* (*C*,*N*)) *x* = *N x*
defect (*quasinorm-of* (*C*,*N*)) = *C*
using *assms* **unfolding** *eNorm-def defect-def quasinorm-on-def* **by** (*auto simp add*: *quasinorm-of-inverse*)

lemma *quasinorm-onI*:
fixes *N*::('*a*::*real-vector*) ⇒ *ennreal* **and** *C*::*real* **and** *F*::'*a set*
assumes ⋀*x y*. *x* ∈ *F* ⟹ *y* ∈ *F* ⟹ *x* + *y* ∈ *F*
⋀*x y*. *x* ∈ *F* ⟹ *y* ∈ *F* ⟹ *N* (*x* + *y*) ≤ *C* ∗ *N x* + *C* ∗ *N y*
⋀*c x*. *c* ≠ 0 ⟹ *x* ∈ *F* ⟹ *c* ∗$_R$ *x* ∈ *F*
⋀*c x*. *c* ≠ 0 ⟹ *x* ∈ *F* ⟹ *N* (*c* ∗$_R$ *x*) ≤ *ennreal* |*c*| ∗ *N x*
0 ∈ *F N*(0) = 0 *C* ≥ 1
shows *quasinorm-on F C N*
proof −
have *N*(*c* ∗$_R$ *x*) = *ennreal* |*c*| ∗ *N x* **if** *x* ∈ *F* **for** *c x*
proof (*cases c* = 0)
case *True*
then show *?thesis* **using** ‹*N* 0 = 0› **by** *auto*
next
case *False*
have *N*((1/*c*) ∗$_R$ (*c* ∗$_R$ *x*)) ≤ *ennreal* (*abs* (1/*c*)) ∗ *N* (*c* ∗$_R$ *x*)
apply (*rule* ‹⋀*c x*. *c* ≠ 0 ⟹ *x* ∈ *F* ⟹ *N*(*c* ∗$_R$ *x*) ≤ *ennreal* |*c*| ∗ *N x*›)
using *False assms that* **by** *auto*
then have *N x* ≤ *ennreal* (*abs* (1/*c*)) ∗ *N* (*c* ∗$_R$ *x*) **using** *False* **by** *auto*
then have *ennreal* |*c*| ∗ *N x* ≤ *ennreal* |*c*| ∗ *ennreal* (*abs* (1/*c*)) ∗ *N* (*c* ∗$_R$ *x*)
by (*simp add*: *mult.assoc mult-left-mono*)
also have ... = *N* (*c* ∗$_R$ *x*) **using** *ennreal-mult*′ *abs-mult False*
by (*metis abs-ge-zero abs-one comm-monoid-mult-class.mult-1 ennreal-1 eq-divide-eq-1 field-class.field-divide-inverse*)
finally show *?thesis*
using ‹⋀*c x*. *c* ≠ 0 ⟹ *x* ∈ *F* ⟹ *N*(*c* ∗$_R$ *x*) ≤ *ennreal* |*c*| ∗ *N x*›[*OF False* ‹*x* ∈ *F*›] **by** *auto*
qed
then show *?thesis*
unfolding *quasinorm-on-def* **using** *assms* **by** (*auto*, *metis real-vector.scale-zero-left*)
qed

lemma *extend-quasinorm*:

assumes *quasinorm-on F C N*
shows *quasinorm-on UNIV C* (λx. *if* $x \in F$ *then N x else* ∞)
proof −
have $*$: (*if* $x + y \in F$ *then N* $(x + y)$ *else* ∞)
$\leq$ *ennreal C* $*$ (*if* $x \in F$ *then N x else* ∞) $+$ *ennreal C* $*$ (*if* $y \in F$ *then N y else* ∞) **for** *x y*
proof (*cases* $x \in F \wedge y \in F$)
case *True*
then show *?thesis* **using** *assms* **unfolding** *quasinorm-on-def* **by** *auto*
next
case *False*
moreover have $C \geq 1$ **using** *assms* **unfolding** *quasinorm-on-def* **by** *auto*
ultimately have $*$: *ennreal C* $*$ (*if* $x \in F$ *then N x else* ∞) $+$ *ennreal C* $*$ (*if* $y \in F$ *then N y else* ∞) $= \infty$
using *ennreal-mult-eq-top-iff* **by** *auto*
show *?thesis* **by** (*simp add*: $*$)
qed
show *?thesis*
apply (*rule quasinorm-onI*)
using *assms* $*$ **unfolding** *quasinorm-on-def* **apply** (*auto simp add*: *ennreal-top-mult mult.commute*)
by (*metis abs-zero ennreal-0 mult-zero-right real-vector.scale-zero-right*)
qed

2.2 The space and the zero space of a quasinorm

The space of a quasinorm is the vector subspace where it is meaningful, i.e., finite.

definition $space_N$::('*a*::*real-vector*) *quasinorm* $\Rightarrow$ '*a set*
where $space_N$ $N = \{f.$ *eNorm N f* $< \infty\}$

lemma *spaceN-iff*:
$x \in space_N$ $N \longleftrightarrow$ *eNorm N x* $< \infty$
unfolding $space_N$*-def* **by** *simp*

lemma *spaceN-cmult* [*simp*]:
assumes $x \in space_N$ N
shows $c *_R x \in space_N$ N
using *assms* **unfolding** *spaceN-iff* **using** *eNorm-cmult*[*of N c x*] **by** (*simp add*: *ennreal-mult-less-top*)

lemma *spaceN-add* [*simp*]:
assumes $x \in space_N$ N $y \in space_N$ N
shows $x + y \in space_N$ N
proof −
have *eNorm N x* $< \infty$ *eNorm N y* $< \infty$ **using** *assms* **unfolding** $space_N$*-def* **by** *auto*
then have *defect N* $*$ *eNorm N x* $+$ *defect N* $*$ *eNorm N y* $< \infty$
by (*simp add*: *ennreal-mult-less-top*)

then show *?thesis*
unfolding $space_N$*-def* **using** *eNorm-triangular-ineq*[*of N x y*] *le-less-trans* **by** *blast*
qed

lemma *spaceN-diff* [*simp*]:
assumes $x \in space_N\ N\ y \in space_N\ N$
shows $x - y \in space_N\ N$
using *spaceN-add*[*OF assms*(*1*) *spaceN-cmult*[*OF assms*(*2*), *of* -1]] **by** *auto*

lemma *spaceN-contains-zero* [*simp*]:
$0 \in space_N\ N$
unfolding $space_N$*-def* **by** *auto*

lemma *spaceN-sum* [*simp*]:
assumes $\bigwedge i.\ i \in I \Longrightarrow x\ i \in space_N\ N$
shows $(\sum i{\in}I.\ x\ i) \in space_N\ N$
using *assms* **by** (*induction I rule*: *infinite-finite-induct*, *auto*)

The zero space of a quasinorm is the vector subspace of vectors with zero norm. If one wants to get a true metric space, one should quotient the space by the zero space.

definition $zero\text{-}space_N$::('*a*::*real-vector*) *quasinorm* ⇒ '*a set*
where $zero\text{-}space_N\ N = \{f.\ eNorm\ N\ f = 0\}$

lemma *zero-spaceN-iff*:
$x \in zero\text{-}space_N\ N \longleftrightarrow eNorm\ N\ x = 0$
unfolding $zero\text{-}space_N$*-def* **by** *simp*

lemma *zero-spaceN-cmult*:
assumes $x \in zero\text{-}space_N\ N$
shows $c *_R x \in zero\text{-}space_N\ N$
using *assms* **unfolding** *zero-spaceN-iff* **using** *eNorm-cmult*[*of N c x*] **by** *simp*

lemma *zero-spaceN-add*:
assumes $x \in zero\text{-}space_N\ N\ y \in zero\text{-}space_N\ N$
shows $x + y \in zero\text{-}space_N\ N$
proof −
have $eNorm\ N\ x = 0\ eNorm\ N\ y = 0$ **using** *assms* **unfolding** $zero\text{-}space_N$*-def* **by** *auto*
then have $defect\ N * eNorm\ N\ x + defect\ N * eNorm\ N\ y = 0$ **by** *auto*
then show *?thesis*
unfolding *zero-spaceN-iff* **using** *eNorm-triangular-ineq*[*of N x y*] **by** *auto*
qed

lemma *zero-spaceN-diff*:
assumes $x \in zero\text{-}space_N\ N\ y \in zero\text{-}space_N\ N$
shows $x - y \in zero\text{-}space_N\ N$
using *zero-spaceN-add*[*OF assms*(*1*) *zero-spaceN-cmult*[*OF assms*(*2*), *of* -1]] **by**

auto

lemma *zero-spaceN-subset-spaceN*:
zero-space$_N$ *N* ⊆ *space*$_N$ *N*
by (*simp add*: *spaceN-iff zero-spaceN-iff subset-eq*)

On the space, the norms are finite. Hence, it is much more convenient to work there with a real valued version of the norm. We use Norm with a capital N to distinguish it from norms in a (type class) banach space.

definition *Norm*::*'a quasinorm* ⇒ (*'a*::*real-vector*) ⇒ *real*
where *Norm N x* = *enn2real* (*eNorm N x*)

lemma *Norm-nonneg* [*simp*]:
Norm N x ≥ *0*
unfolding *Norm-def* **by** *auto*

lemma *Norm-zero* [*simp*]:
Norm N 0 = *0*
unfolding *Norm-def* **by** *auto*

lemma *Norm-uminus* [*simp*]:
Norm N (−*x*) = *Norm N x*
unfolding *Norm-def* **by** *auto*

lemma *eNorm-Norm*:
assumes *x* ∈ *space*$_N$ *N*
shows *eNorm N x* = *ennreal* (*Norm N x*)
using *assms* **unfolding** *Norm-def* **by** (*simp add*: *spaceN-iff*)

lemma *eNorm-Norm'*:
assumes *x* ∉ *space*$_N$ *N*
shows *Norm N x* = *0*
using *assms* **unfolding** *Norm-def* **apply** (*auto simp add*: *spaceN-iff*)
using *top.not-eq-extremum* **by** *fastforce*

lemma *Norm-cmult*:
Norm N (*c* ∗$_R$ *x*) = *abs c* ∗ *Norm N x*
unfolding *Norm-def* **unfolding** *eNorm-cmult* **by** (*simp add*: *enn2real-mult*)

lemma *Norm-triangular-ineq*:
assumes *x* ∈ *space*$_N$ *N*
shows *Norm N* (*x* + *y*) ≤ *defect N* ∗ *Norm N x* + *defect N* ∗ *Norm N y*
proof (*cases y* ∈ *space*$_N$ *N*)
case *True*
have ∗: *defect N* ∗ *Norm N x* + *defect N* ∗ *Norm N y* ≥ *1* ∗ *0* + *1* ∗ *0*
apply (*rule add-mono*) **by** (*rule mult-mono'*[*OF defect-ge-1 Norm-nonneg*], *simp*, *simp*)+
have *ennreal* (*Norm N* (*x* + *y*)) = *eNorm N* (*x*+*y*)
using *eNorm-Norm*[*OF spaceN-add*[*OF assms True*]] **by** *auto*

```
  also have ... ≤ defect N * eNorm N x + defect N * eNorm N y
    using eNorm-triangular-ineq[of N x y] by auto
  also have ... = defect N * ennreal(Norm N x) + defect N * ennreal(Norm N y)
    using eNorm-Norm assms True by metis
  also have ... = ennreal(defect N * Norm N x + defect N * Norm N y)
    using ennreal-mult ennreal-plus Norm-nonneg defect-ge-1
      by (metis (no-types, opaque-lifting) ennreal-eq-0-iff less-le ennreal-ge-1 en-
nreal-mult' le-less-linear not-one-le-zero semiring-normalization-rules(34))
  finally show ?thesis
    apply (subst ennreal-le-iff[symmetric]) using * by auto
next
  case False
  have x + y ∉ space_N N
  proof (rule ccontr)
    assume ¬ (x + y ∉ space_N N)
    then have x + y ∈ space_N N by simp
   have y ∈ space_N N using spaceN-diff[OF ‹x + y ∈ space_N N› assms] by auto
    then show False using False by simp
  qed
 then have Norm N (x+y) = 0 unfolding Norm-def using spaceN-iff top.not-eq-extremum
by force
  moreover have defect N * Norm N x + defect N * Norm N y ≥ 1 * 0 + 1 * 0
     apply (rule add-mono) by (rule mult-mono'[OF defect-ge-1 Norm-nonneg],
simp, simp)+
  ultimately show ?thesis by simp
qed

lemma Norm-triangular-ineq-diff:
  assumes x ∈ space_N N
  shows Norm N (x − y) ≤ defect N * Norm N x + defect N * Norm N y
using Norm-triangular-ineq[OF assms, of −y] by auto

lemma zero-spaceN-iff':
  x ∈ zero-space_N N ⟷ (x ∈ space_N N ∧ Norm N x = 0)
using eNorm-Norm unfolding space_N-def zero-space_N-def by (auto simp add:
Norm-def, fastforce)

lemma Norm-sum:
  assumes ⋀i. i < n ⟹ u i ∈ space_N N
  shows Norm N (∑ i∈{..<n}. u i) ≤ (∑ i∈{..<n}. (defect N)^(Suc i) * Norm
N (u i))
proof −
  have *: 0 ≤ defect N * defect N ^ i * Norm N (u i) for i
   by (meson Norm-nonneg defect-ge-1 dual-order.trans linear mult-nonneg-nonneg
not-one-le-zero zero-le-power)

  have ennreal (Norm N (∑ i∈{..<n}. u i)) = eNorm N (∑ i∈{..<n}. u i)
    apply (rule eNorm-Norm[symmetric], rule spaceN-sum) using assms by auto
  also have ... ≤ (∑ i∈{..<n}. (defect N)^(Suc i) * eNorm N (u i))
```

using *eNorm-sum* **by** *simp*
also have ... = $(\sum i{\in}\{..{<}n\}.\ (defect\ N)\hat{}(Suc\ i) * ennreal\ (Norm\ N\ (u\ i)))$
using *eNorm-Norm*[*OF assms*] **by** *auto*
also have ... = $(\sum i{\in}\{..{<}n\}.\ ennreal((defect\ N)\hat{}(Suc\ i) * Norm\ N\ (u\ i)))$
by (*subst ennreal-mult''*, *auto*)
also have ... = $ennreal\ (\sum i{\in}\{..{<}n\}.\ (defect\ N)\hat{}(Suc\ i) * Norm\ N\ (u\ i))$
by (*auto intro*!: *sum-ennreal simp add*: $*$)
finally have $**$: $ennreal\ (Norm\ N\ (\sum i{\in}\{..{<}n\}.\ u\ i)) \leq ennreal\ (\sum i{\in}\{..{<}n\}.\ (defect\ N)\hat{}(Suc\ i) * Norm\ N\ (u\ i))$
by *simp*
show *?thesis*
apply (*subst ennreal-le-iff*[*symmetric*], *rule sum-nonneg*) **using** $*$ $**$ **by** *auto*
qed

2.3 An example: the ambient norm in a normed vector space

definition *N-of-norm*::$'a$::*real-normed-vector quasinorm*
where *N-of-norm* = *quasinorm-of* (*1*, $\lambda f.$ *norm f*)

lemma *N-of-norm*:
eNorm N-of-norm f = *ennreal* (*norm f*)
Norm N-of-norm f = *norm f*
defect (*N-of-norm*) = *1*
proof −
have $*$: *quasinorm-on UNIV 1* ($\lambda f.$ *norm f*)
by (*rule quasinorm-onI*, *auto simp add*: *ennreal-mult'*, *metis ennreal-leI ennreal-plus norm-imp-pos-and-ge norm-triangle-ineq*)
show *eNorm N-of-norm f* = *ennreal* (*norm f*)
defect (*N-of-norm*) = *1*
unfolding *N-of-norm-def* **using** *quasinorm-of*[*OF* $*$] **by** *auto*
then show *Norm N-of-norm f* = *norm f* **unfolding** *Norm-def* **by** *auto*
qed

lemma *N-of-norm-space* [*simp*]:
$space_N$ *N-of-norm* = *UNIV*
unfolding $space_N$*-def* **apply** *auto* **unfolding** *N-of-norm*(*1*) **by** *auto*

lemma *N-of-norm-zero-space* [*simp*]:
$zero\text{-}space_N$ *N-of-norm* = {*0*}
unfolding $zero\text{-}space_N$*-def* **apply** *auto* **unfolding** *N-of-norm*(*1*) **by** *auto*

2.4 An example: the space of bounded continuous functions from a topological space to a normed real vector space

The Banach space of bounded continuous functions is defined in `Bounded_Continuous_Function.thy`, as a type `bcontfun`. We import very quickly the results proved in this file to the current framework.

definition $bcontfun_N$::($'a$::*topological-space* ⇒ $'b$::*real-normed-vector*) *quasinorm*

where $bcontfun_N$ = *quasinorm-of* (1, λf. *if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* (∞::*ennreal*))

lemma $bcontfun_N$:
fixes f::('a::*topological-space* $\Rightarrow$ 'b::*real-normed-vector*)
shows *eNorm* $bcontfun_N$ f = (*if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* (∞::*ennreal*))
Norm $bcontfun_N$ f = (*if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* 0)
defect ($bcontfun_N$::(('a $\Rightarrow$ 'b) *quasinorm*)) = 1
proof −
have $*$: *quasinorm-on bcontfun* 1 (λ(f::('a $\Rightarrow$ 'b)). *norm*(*Bcontfun* f))
proof (*rule quasinorm-onI*, *auto*)
fix f g::'a $\Rightarrow$ 'b **assume** H: $f \in$ *bcontfun* $g \in$ *bcontfun*
then show $f + g \in$ *bcontfun* **unfolding** *plus-fun-def* **by** (*simp add*: *plus-cont*)
have $*$: *Bcontfun*($f + g$) = *Bcontfun* f + *Bcontfun* g
using H
by (*auto simp*: *eq-onp-def plus-fun-def bcontfun-def intro*!: *plus-bcontfun.abs-eq*[*symmetric*])
show *ennreal* (*norm* (*Bcontfun* ($f + g$))) $\leq$ *ennreal* (*norm* (*Bcontfun* f)) + *ennreal* (*norm* (*Bcontfun* g))
unfolding $*$ **using** *ennreal-leI*[*OF norm-triangle-ineq*] **by** *auto*
next
fix c::*real* **and** f::'a $\Rightarrow$ 'b **assume** H: $f \in$ *bcontfun*
then show c $*_R$ $f \in$ *bcontfun* **unfolding** *scaleR-fun-def* **by** (*simp add*: *scaleR-cont*)
have $*$: *Bcontfun*(c $*_R$ f) = c $*_R$ *Bcontfun* f
using H
by (*auto simp*: *eq-onp-def scaleR-fun-def bcontfun-def intro*!: *scaleR-bcontfun.abs-eq*[*symmetric*])
show *ennreal* (*norm* (*Bcontfun* (c $*_R$ f))) $\leq$ *ennreal* |c| $*$ *ennreal* (*norm* (*Bcontfun* f))
unfolding $*$ **by** (*simp add*: *ennreal-mult''*)
next
show (0::'$a$$\Rightarrow$'$b$) $\in$ *bcontfun Bcontfun* 0 = 0
unfolding *zero-fun-def zero-bcontfun-def* **by** (*auto simp add*: *const-bcontfun*)
qed
have $**$: *quasinorm-on UNIV* 1 (λ(f::'$a$$\Rightarrow$'$b$). *if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* (∞::*ennreal*))
by (*rule extend-quasinorm*[*OF* $*$])
show *eNorm* $bcontfun_N$ f = (*if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* (∞::*ennreal*))
defect ($bcontfun_N$::('a $\Rightarrow$ 'b) *quasinorm*) = 1
using *quasinorm-of*[*OF* $**$] **unfolding** $bcontfun_N$*-def* **by** *auto*
then show *Norm* $bcontfun_N$ f = (*if* $f \in$ *bcontfun then norm*(*Bcontfun* f) *else* 0)
unfolding *Norm-def* **by** *auto*
qed

lemma $bcontfun_N$*-space*:
$space_N$ $bcontfun_N$ = *bcontfun*
using $bcontfun_N$(1) **by** (*metis* (*no-types*, *lifting*) *Collect-cong bcontfun-def enn2real-top ennreal-0 ennreal-enn2real ennreal-less-top ennreal-zero-neq-top infinity-ennreal-def mem-Collect-eq*

space$_N$-def)

lemma *bcontfun$_N$-zero-space*:
zero-space$_N$ bcontfun$_N$ = {0}
apply (*auto simp add*: *zero-spaceN-iff*)
by (*metis Bcontfun-inject bcontfun$_N$(1) eNorm-zero ennreal-eq-zero-iff ennreal-zero-neq-top infinity-ennreal-def norm-eq-zero norm-imp-pos-and-ge*)

lemma *bcontfun$_N$D*:
assumes *f ∈ space$_N$ bcontfun$_N$*
shows *continuous-on UNIV f*
⋀x. norm(f x) ≤ Norm bcontfun$_N$ f
proof−
have *f ∈ bcontfun* **using** *assms* **unfolding** *bcontfun$_N$-space* **by** *simp*
then show *continuous-on UNIV f* **unfolding** *bcontfun-def* **by** *auto*
show *⋀x. norm(f x) ≤ Norm bcontfun$_N$ f*
using *norm-bounded bcontfun$_N$(2)* ‹*f ∈ bcontfun*› **by** (*metis Bcontfun-inverse*)
qed

lemma *bcontfun$_N$I*:
assumes *continuous-on UNIV f*
⋀x. norm(f x) ≤ C
shows *f ∈ space$_N$ bcontfun$_N$*
Norm bcontfun$_N$ f ≤ C
proof −
have *f ∈ bcontfun* **using** *assms bcontfun-normI* **by** *blast*
then show *f ∈ space$_N$ bcontfun$_N$* **unfolding** *bcontfun$_N$-space* **by** *simp*
show *Norm bcontfun$_N$ f ≤ C* **unfolding** *bcontfun$_N$(2)* **using** ‹*f ∈ bcontfun*›
apply *auto*
using *assms(2)* **by** (*metis apply-bcontfun-cases apply-bcontfun-inverse norm-bound*)
qed

2.5 Continuous inclusions between functional spaces

Continuous inclusions between functional spaces are now defined

instantiation *quasinorm*:: (*real-vector*) *preorder*
begin

definition *less-eq-quasinorm*::*'a quasinorm ⇒ 'a quasinorm ⇒ bool*
where *less-eq-quasinorm N1 N2 = (∃ C≥(0*::*real*). *∀ f. eNorm N2 f ≤ C ∗ eNorm N1 f*)

definition *less-quasinorm*::*'a quasinorm ⇒ 'a quasinorm ⇒ bool*
where *less-quasinorm N1 N2 = (less-eq N1 N2 ∧ (¬ less-eq N2 N1))*

instance proof −
have *E*: *N ≤ N* **for** *N*::*'a quasinorm*
unfolding *less-eq-quasinorm-def* **by** (*rule exI*[*of - 1*], *auto*)
have *T*: *N1 ≤ N3* **if** *N1 ≤ N2 N2 ≤ N3* **for** *N1 N2 N3*::*'a quasinorm*

```
  proof −
    obtain C C' where *: ⋀f. eNorm N2 f ≤ ennreal C * eNorm N1 f
                         ⋀f. eNorm N3 f ≤ ennreal C' * eNorm N2 f
                         C ≥ 0 C' ≥ 0
      using ‹N1 ≤ N2› ‹N2 ≤ N3› unfolding less-eq-quasinorm-def by metis
    {
      fix f
      have eNorm N3 f ≤ ennreal C' * ennreal C * eNorm N1 f
      by (metis *(1)[of f] *(2)[of f] mult.commute mult.left-commute mult-left-mono
order-trans zero-le)
      also have ... = ennreal(C' * C) * eNorm N1 f
        using ‹C ≥ 0› ‹C' ≥ 0› ennreal-mult by auto
      finally have eNorm N3 f ≤ ennreal(C' * C) * eNorm N1 f by simp
    }
    then show ?thesis
      unfolding less-eq-quasinorm-def using ‹C ≥ 0› ‹C' ≥ 0› zero-le-mult-iff by
auto
  qed

  show OFCLASS('a quasinorm, preorder-class)
    apply standard
    unfolding less-quasinorm-def apply simp
    using E apply fast
    using T apply fast
    done
qed
end

abbreviation quasinorm-subset :: ('a::real-vector) quasinorm ⇒ 'a quasinorm ⇒
bool
  where quasinorm-subset ≡ less

abbreviation quasinorm-subset-eq :: ('a::real-vector) quasinorm ⇒ 'a quasinorm
⇒ bool
  where quasinorm-subset-eq ≡ less-eq

notation
  quasinorm-subset (‹'(⊂_N')›) and
  quasinorm-subset (‹(-/ ⊂_N -)› [51, 51] 50) and
  quasinorm-subset-eq (‹'(⊆_N')›) and
  quasinorm-subset-eq (‹(-/ ⊆_N -)› [51, 51] 50)


lemma quasinorm-subsetD:
  assumes N1 ⊆_N N2
  shows ∃ C≥(0::real). ∀f. eNorm N2 f ≤ C * eNorm N1 f
using assms unfolding less-eq-quasinorm-def by auto

lemma quasinorm-subsetI:
```

assumes $\bigwedge f.\ f \in space_N\ N1 \Longrightarrow eNorm\ N2\ f \leq ennreal\ C * eNorm\ N1\ f$
shows $N1 \subseteq_N N2$
proof −
have $eNorm\ N2\ f \leq ennreal\ (max\ C\ 1) * eNorm\ N1\ f$ **for** f
proof (*cases* $f \in space_N\ N1$)
case *True*
then show *?thesis* **using** *assms*[*OF* ‹$f \in space_N\ N1$›]
by (*metis* (*no-types*, *opaque-lifting*) *dual-order.trans ennreal-leI max.cobounded2 max.commute*
mult.commute ordered-comm-semiring-class.comm-mult-left-mono zero-le)
next
case *False*
then show *?thesis* **using** *spaceN-iff*
by (*metis ennreal-ge-1 ennreal-mult-less-top infinity-ennreal-def max.cobounded1*
max.commute not-le not-one-le-zero top.not-eq-extremum)
qed
then show *?thesis* **unfolding** *less-eq-quasinorm-def*
by (*metis ennreal-max-0′ max.cobounded2*)
qed

lemma *quasinorm-subsetI′*:
assumes $\bigwedge f.\ f \in space_N\ N1 \Longrightarrow f \in space_N\ N2$
$\bigwedge f.\ f \in space_N\ N1 \Longrightarrow Norm\ N2\ f \leq C * Norm\ N1\ f$
shows $N1 \subseteq_N N2$
proof (*rule quasinorm-subsetI*)
fix f **assume** $f \in space_N\ N1$
then have $f \in space_N\ N2$ **using** *assms*(*1*) **by** *simp*
then have $eNorm\ N2\ f = ennreal(Norm\ N2\ f)$ **using** *eNorm-Norm* **by** *auto*
also have ... $\leq ennreal(C * Norm\ N1\ f)$
using *assms*(*2*)[*OF* ‹$f \in space_N\ N1$›] *ennreal-leI* **by** *blast*
also have ... $= ennreal\ C * ennreal(Norm\ N1\ f)$
using *ennreal-mult″* **by** *auto*
also have ... $= ennreal\ C * eNorm\ N1\ f$
using *eNorm-Norm*[*OF* ‹$f \in space_N\ N1$›] **by** *auto*
finally show $eNorm\ N2\ f \leq ennreal\ C * eNorm\ N1\ f$
by *simp*
qed

lemma *quasinorm-subset-space*:
assumes $N1 \subseteq_N N2$
shows $space_N\ N1 \subseteq space_N\ N2$
using *assms* **unfolding** $space_N$*-def less-eq-quasinorm-def*
by (*auto*, *metis ennreal-mult-eq-top-iff ennreal-neq-top less-le top.extremum-strict top.not-eq-extremum*)

lemma *quasinorm-subset-Norm-eNorm*:
assumes $f \in space_N\ N1 \Longrightarrow Norm\ N2\ f \leq C * Norm\ N1\ f$
$N1 \subseteq_N N2$
$C > 0$

shows *eNorm N2 f* $\leq$ *ennreal C* $*$ *eNorm N1 f*
proof (*cases f* $\in$ *space*$_N$ *N1*)
case *True*
then have *f* $\in$ *space*$_N$ *N2* **using** *quasinorm-subset-space*[*OF* ‹*N1* $\subseteq_N$ *N2*›] **by** *auto*
then show *?thesis*
using *eNorm-Norm*[*OF True*] *eNorm-Norm assms*(*1*)[*OF True*] **by** (*metis Norm-nonneg ennreal-leI ennreal-mult''*)
next
case *False*
then show *?thesis* **using** ‹*C* $>$ *0*›
by (*metis ennreal-eq-zero-iff ennreal-mult-eq-top-iff infinity-ennreal-def less-imp-le neq-top-trans not-le spaceN-iff*)
qed

lemma *quasinorm-subset-zero-space*:
assumes *N1* $\subseteq_N$ *N2*
shows *zero-space*$_N$ *N1* $\subseteq$ *zero-space*$_N$ *N2*
using *assms* **unfolding** *zero-space*$_N$*-def less-eq-quasinorm-def*
by (*auto*, *metis le-zero-eq mult-zero-right*)

We would like to define the equivalence relation associated to the above order, i.e., the equivalence between norms. This is not equality, so we do not have a true order, but nevertheless this is handy, and not standard in a preorder in Isabelle. The file Library/Preorder.thy defines such an equivalence relation, but including it breaks some proofs so we go the naive way.

definition *quasinorm-equivalent*::(*'a*::*real-vector*) *quasinorm* $\Rightarrow$ *'a quasinorm* $\Rightarrow$ *bool* (**infix** ‹$=_N$› *60*)
where *quasinorm-equivalent N1 N2* = ((*N1* $\subseteq_N$ *N2*) $\wedge$ (*N2* $\subseteq_N$ *N1*))

lemma *quasinorm-equivalent-sym* [*sym*]:
assumes *N1* $=_N$ *N2*
shows *N2* $=_N$ *N1*
using *assms* **unfolding** *quasinorm-equivalent-def* **by** *auto*

lemma *quasinorm-equivalent-trans* [*trans*]:
assumes *N1* $=_N$ *N2 N2* $=_N$ *N3*
shows *N1* $=_N$ *N3*
using *assms order-trans* **unfolding** *quasinorm-equivalent-def* **by** *blast*

2.6 The intersection and the sum of two functional spaces

In this paragraph, we define the intersection and the sum of two functional spaces. In terms of the order introduced above, this corresponds to the minimum and the maximum. More important, these are the first two examples of interpolation spaces between two functional spaces, and they are central as all the other ones are built using them.

definition *quasinorm-intersection*::('a::*real-vector*) *quasinorm* ⇒ 'a *quasinorm* ⇒ 'a *quasinorm* (**infix** ‹∩$_N$› *70*)
where *quasinorm-intersection N1 N2* = *quasinorm-of* (*max* (*defect N1*) (*defect N2*), λ*f*. *eNorm N1 f* + *eNorm N2 f*)

lemma *quasinorm-intersection*:
eNorm (*N1* ∩$_N$ *N2*) *f* = *eNorm N1 f* + *eNorm N2 f*
defect (*N1* ∩$_N$ *N2*) = *max* (*defect N1*) (*defect N2*)
proof −
have *T*: *eNorm N1* (*x* + *y*) + *eNorm N2* (*x* + *y*) ≤
ennreal (*max* (*defect N1*) (*defect N2*)) ∗ (*eNorm N1 x* + *eNorm N2 x*) + *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ (*eNorm N1 y* + *eNorm N2 y*) **for** *x y*
proof −
have *eNorm N1* (*x* + *y*) ≤ *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ *eNorm N1 x* + *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ *eNorm N1 y*
using *eNorm-triangular-ineq*[*of N1 x y*] **by** (*metis* (*no-types*) *max-def distrib-left ennreal-leI mult-right-mono order-trans zero-le*)
moreover have *eNorm N2* (*x* + *y*) ≤ *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ *eNorm N2 x* + *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ *eNorm N2 y*
using *eNorm-triangular-ineq*[*of N2 x y*] **by** (*metis* (*no-types*) *max-def max.commute distrib-left ennreal-leI mult-right-mono order-trans zero-le*)
ultimately have *eNorm N1* (*x* + *y*) + *eNorm N2* (*x* + *y*) ≤ *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ (*eNorm N1 x* + *eNorm N1 y* + (*eNorm N2 x* + *eNorm N2 y*))
by (*simp add*: *add-mono-thms-linordered-semiring*(*1*) *distrib-left*)
then show *?thesis*
by (*simp add*: *ab-semigroup-add-class.add-ac*(*1*) *add.left-commute distrib-left*)
qed

have *H*: *eNorm N1* (*c* ∗$_R$ *x*) + *eNorm N2* (*c* ∗$_R$ *x*) ≤ *ennreal* |*c*| ∗ (*eNorm N1 x* + *eNorm N2 x*) **for** *c x*
by (*simp add*: *eNorm-cmult*[*of N1 c x*] *eNorm-cmult*[*of N2 c x*] *distrib-left*)
have ∗: *quasinorm-on UNIV* (*max* (*defect N1*) (*defect N2*)) (λ*f*. *eNorm N1 f* + *eNorm N2 f*)
apply (*rule quasinorm-onI*) **using** *T H defect-ge-1*[*of N1*] *defect-ge-1*[*of N2*] **by** *auto*
show *defect* (*N1* ∩$_N$ *N2*) = *max* (*defect N1*) (*defect N2*)
eNorm (*N1* ∩$_N$ *N2*) *f* = *eNorm N1 f* + *eNorm N2 f*
unfolding *quasinorm-intersection-def* **using** *quasinorm-of*[*OF* ∗] **by** *auto*
qed

lemma *quasinorm-intersection-commute*:
N1 ∩$_N$ *N2* = *N2* ∩$_N$ *N1*
unfolding *quasinorm-intersection-def max.commute*[*of defect N1*] *add.commute*[*of eNorm N1* -] **by** *simp*

lemma *quasinorm-intersection-space*:
space$_N$ (*N1* ∩$_N$ *N2*) = *space*$_N$ *N1* ∩ *space*$_N$ *N2*
apply *auto* **unfolding** *quasinorm-intersection*(*1*) *spaceN-iff* **by** *auto*

lemma *quasinorm-intersection-zero-space*:
zero-space$_N$ (*N1* $\cap_N$ *N2*) = *zero-space*$_N$ *N1* $\cap$ *zero-space*$_N$ *N2*
apply *auto* **unfolding** *quasinorm-intersection*(*1*) *zero-spaceN-iff* **by** (*auto simp add*: *add-eq-0-iff-both-eq-0*)

lemma *quasinorm-intersection-subset*:
N1 $\cap_N$ *N2* $\subseteq_N$ *N1 N1* $\cap_N$ *N2* $\subseteq_N$ *N2*
by (*rule quasinorm-subsetI*[*of - - 1*], *auto simp add*: *quasinorm-intersection*(*1*))+

lemma *quasinorm-intersection-minimum*:
assumes *N* $\subseteq_N$ *N1 N* $\subseteq_N$ *N2*
shows *N* $\subseteq_N$ *N1* $\cap_N$ *N2*
proof −
obtain *C1 C2*::*real* **where** ∗: $\bigwedge f$. *eNorm N1 f* ≤ *C1* ∗ *eNorm N f*
$\bigwedge f$. *eNorm N2 f* ≤ *C2* ∗ *eNorm N f*
C1 ≥ *0 C2* ≥ *0*
using *quasinorm-subsetD*[*OF assms*(*1*)] *quasinorm-subsetD*[*OF assms*(*2*)] **by** *blast*
have ∗∗: *eNorm* (*N1* $\cap_N$ *N2*) *f* ≤ (*C1* + *C2*) ∗ *eNorm N f* **for** *f*
unfolding *quasinorm-intersection*(*1*) **using** *add-mono*[*OF* ∗(*1*) ∗(*2*)] **by** (*simp add*: *distrib-right* ∗)
show *?thesis*
apply (*rule quasinorm-subsetI*) **using** ∗∗ **by** *auto*
qed

lemma *quasinorm-intersection-assoc*:
(*N1* $\cap_N$ *N2*) $\cap_N$ *N3* =$_N$ *N1* $\cap_N$ (*N2* $\cap_N$ *N3*)
unfolding *quasinorm-equivalent-def* **by** (*meson order-trans quasinorm-intersection-minimum quasinorm-intersection-subset*)

definition *quasinorm-sum*::('*a*::*real-vector*) *quasinorm* ⇒ '*a quasinorm* ⇒ '*a quasinorm* (**infix** ‹+$_N$› *70*)
where *quasinorm-sum N1 N2* = *quasinorm-of* (*max* (*defect N1*) (*defect N2*), λ*f*. *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*})

lemma *quasinorm-sum*:
eNorm (*N1* +$_N$ *N2*) *f* = *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*}
defect (*N1* +$_N$ *N2*) = *max* (*defect N1*) (*defect N2*)
proof −
define *N* **where** *N* = (λ*f*. *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*})
have *T*: *N* (*f*+*g*) ≤
ennreal (*max* (*defect N1*) (*defect N2*)) ∗ *N f* + *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ *N g* **for** *f g*
proof −

have $\exists u.\ (\forall n.\ u\ n \in \{eNorm\ N1\ f1 + eNorm\ N2\ f2 |\ f1\ f2.\ f = f1 + f2\}) \wedge u \longrightarrow Inf\ \{eNorm\ N1\ f1 + eNorm\ N2\ f2 |\ f1\ f2.\ f = f1 + f2\}$
by (*rule Inf-as-limit*, *auto*, *rule exI*[*of - f*], *rule exI*[*of - 0*], *auto*)
then obtain *uf* **where** *uf*: $\bigwedge n.\ uf\ n \in \{eNorm\ N1\ f1 + eNorm\ N2\ f2 |\ f1\ f2.\ f = f1 + f2\}$
$uf \longrightarrow Inf\ \{eNorm\ N1\ f1 + eNorm\ N2\ f2 |\ f1\ f2.\ f = f1 + f2\}$
by *blast*
have $\exists f1\ f2.\ \forall n.\ uf\ n = eNorm\ N1\ (f1\ n) + eNorm\ N2\ (f2\ n) \wedge f = f1\ n + f2\ n$
using *uf*(*1*) **by** (*subst choice-iff*[*symmetric*])+ *blast*
then obtain *f1 f2* **where** *F*: $\bigwedge n.\ uf\ n = eNorm\ N1\ (f1\ n) + eNorm\ N2\ (f2\ n)\ \bigwedge n.\ f = f1\ n + f2\ n$
by *blast*

have $\exists u.\ (\forall n.\ u\ n \in \{eNorm\ N1\ g1 + eNorm\ N2\ g2 |\ g1\ g2.\ g = g1 + g2\}) \wedge u \longrightarrow Inf\ \{eNorm\ N1\ g1 + eNorm\ N2\ g2 |\ g1\ g2.\ g = g1 + g2\}$
by (*rule Inf-as-limit*, *auto*, *rule exI*[*of - g*], *rule exI*[*of - 0*], *auto*)
then obtain *ug* **where** *ug*: $\bigwedge n.\ ug\ n \in \{eNorm\ N1\ g1 + eNorm\ N2\ g2 |\ g1\ g2.\ g = g1 + g2\}$
$ug \longrightarrow Inf\ \{eNorm\ N1\ g1 + eNorm\ N2\ g2 |\ g1\ g2.\ g = g1 + g2\}$
by *blast*
have $\exists g1\ g2.\ \forall n.\ ug\ n = eNorm\ N1\ (g1\ n) + eNorm\ N2\ (g2\ n) \wedge g = g1\ n + g2\ n$
using *ug*(*1*) **by** (*subst choice-iff*[*symmetric*])+ *blast*
then obtain *g1 g2* **where** *G*: $\bigwedge n.\ ug\ n = eNorm\ N1\ (g1\ n) + eNorm\ N2\ (g2\ n)\ \bigwedge n.\ g = g1\ n + g2\ n$
by *blast*

define *h1* **where** $h1 = (\lambda n.\ f1\ n + g1\ n)$
define *h2* **where** $h2 = (\lambda n.\ f2\ n + g2\ n)$
have $*$: $f + g = h1\ n + h2\ n$ **for** *n*
unfolding *h1-def h2-def* **using** *F*(*2*) *G*(*2*) **by** (*auto simp add*: *algebra-simps*)
have $N\ (f+g) \leq ennreal\ (max\ (defect\ N1)\ (defect\ N2)) * (uf\ n + ug\ n)$ **for** *n*
proof −
have $N\ (f+g) \leq eNorm\ N1\ (h1\ n) + eNorm\ N2\ (h2\ n)$
unfolding *N-def* **apply** (*rule Inf-lower*, *auto*, *rule exI*[*of - h1 n*], *rule exI*[*of - h2 n*])
using $*$ **by** *auto*
also have ... $\leq ennreal\ (defect\ N1) * eNorm\ N1\ (f1\ n) + ennreal\ (defect\ N1) * eNorm\ N1\ (g1\ n)$
$+ (ennreal\ (defect\ N2) * eNorm\ N2\ (f2\ n) + ennreal\ (defect\ N2) * eNorm\ N2\ (g2\ n))$
unfolding *h1-def h2-def* **apply** (*rule add-mono*) **using** *eNorm-triangular-ineq* **by** *auto*
also have ... $\leq (ennreal\ (max\ (defect\ N1)\ (defect\ N2)) * eNorm\ N1\ (f1\ n) + ennreal\ (max\ (defect\ N1)\ (defect\ N2)) * eNorm\ N1\ (g1\ n))$
$+ (ennreal\ (max\ (defect\ N1)\ (defect\ N2)) * eNorm\ N2\ (f2\ n) +$

ennreal (*max* (*defect N1*) (*defect N2*)) ∗ *eNorm N2* (*g2 n*))
by (*auto intro*!: *add-mono mult-mono ennreal-leI*)
also have ... = *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ (*uf n* + *ug n*)
unfolding *F*(*1*) *G*(*1*) **by** (*auto simp add*: *algebra-simps*)
finally show *?thesis* **by** *simp*
qed
moreover have ... ⟶ *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ (*N f* + *N g*)
unfolding *N-def* **by** (*auto intro*!: *tendsto-intros simp add*: *uf*(*2*) *ug*(*2*))
ultimately have *N* (*f*+*g*) ≤ *ennreal* (*max* (*defect N1*) (*defect N2*)) ∗ (*N f* + *N g*)
using *LIMSEQ-le-const* **by** *blast*
then show *?thesis* **by** (*auto simp add*: *algebra-simps*)
qed

have *H*: *N* (*c* ∗*R f*) ≤ *ennreal* |*c*| ∗ *N f* **for** *c f*
proof −
have ∃ *u*. (∀ *n*. *u n* ∈ {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*}) ∧ *u* ⟶ *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*}
by (*rule Inf-as-limit*, *auto*, *rule exI*[*of* - *f*], *rule exI*[*of* - *0*], *auto*)
then obtain *uf* **where** *uf*: ⋀*n*. *uf n* ∈ {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*}
uf ⟶ *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *f* = *f1* + *f2*}
by *blast*
have ∃ *f1 f2*. ∀ *n*. *uf n* = *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*) ∧ *f* = *f1 n* + *f2 n*
using *uf*(*1*) **by** (*subst choice-iff*[*symmetric*])+ *blast*
then obtain *f1 f2* **where** *F*: ⋀*n*. *uf n* = *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*) ⋀*n*. *f* = *f1 n* + *f2 n*
by *blast*

have *N* (*c* ∗*R f*) ≤ |*c*| ∗ *uf n* **for** *n*
proof −
have *N* (*c* ∗*R f*) ≤ *eNorm N1* (*c* ∗*R f1 n*) + *eNorm N2* (*c* ∗*R f2 n*)
unfolding *N-def* **apply** (*rule Inf-lower*, *auto*, *rule exI*[*of* - *c* ∗*R f1 n*], *rule exI*[*of* - *c* ∗*R f2 n*])
using *F*(*2*)[*of n*] *scaleR-add-right* **by** *auto*
also have ... = |*c*| ∗ (*eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*))
by (*auto simp add*: *algebra-simps eNorm-cmult*)
finally show *?thesis* **using** *F*(*1*) **by** *simp*
qed
moreover have ... ⟶ |*c*| ∗ *N f*
unfolding *N-def* **by** (*auto intro*!: *tendsto-intros simp add*: *uf*(*2*))
ultimately show *?thesis*
using *LIMSEQ-le-const* **by** *blast*
qed

have *Inf* {*eNorm N1 f1* + *eNorm N2 f2*| *f1 f2*. *0* = *f1* + *f2*} ≤ *0*

by (*rule Inf-lower*, *auto*, *rule exI*[*of - 0*], *auto*)
then have *Z*: *Inf* {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *0 = f1 + f2*} = *0*
by *auto*

have ∗: *quasinorm-on UNIV* (*max* (*defect N1*) (*defect N2*)) (λ*f*. *Inf* {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*})
apply (*rule quasinorm-onI*) **using** *T H Z defect-ge-1*[*of N1*] *defect-ge-1*[*of N2*]
unfolding *N-def* **by** *auto*
show *defect* (*N1* +$_N$ *N2*) = *max* (*defect N1*) (*defect N2*)
eNorm (*N1* +$_N$ *N2*) *f* = *Inf* {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*}
unfolding *quasinorm-sum-def* **using** *quasinorm-of*[*OF* ∗] **by** *auto*
qed

lemma *quasinorm-sum-limit*:
∃*f1 f2*. (∀ *n*. *f = f1 n + f2 n*) ∧ (λ*n*. *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*)) ⟶ *eNorm* (*N1* +$_N$ *N2*) *f*
proof −
have ∃ *u*. (∀ *n*. *u n* ∈ {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*}) ∧ *u* ⟶ *Inf* {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*}
by (*rule Inf-as-limit*, *auto*, *rule exI*[*of - f*], *rule exI*[*of - 0*], *auto*)
then obtain *uf* **where** *uf*: ⋀*n*. *uf n* ∈ {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*}
uf ⟶ *Inf* {*eNorm N1 f1 + eNorm N2 f2*| *f1 f2*. *f = f1 + f2*}
by *blast*
have ∃*f1 f2*. ∀ *n*. *uf n* = *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*) ∧ *f = f1 n + f2 n*
using *uf*(*1*) **by** (*subst choice-iff*[*symmetric*])+ *blast*
then obtain *f1 f2* **where** *F*: ⋀*n*. *uf n* = *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*) ⋀*n*. *f = f1 n + f2 n*
by *blast*
have (λ*n*. *eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*)) ⟶ *eNorm* (*N1* +$_N$ *N2*) *f*
using *F*(*1*) *uf*(*2*) **unfolding** *quasinorm-sum*(*1*) **by** *presburger*
then show *?thesis* **using** *F*(*2*) **by** *auto*
qed

lemma *quasinorm-sum-space*:
space$_N$ (*N1* +$_N$ *N2*) = {*f + g*|*f g*. *f* ∈ *space*$_N$ *N1* ∧ *g* ∈ *space*$_N$ *N2*}
proof (*auto*)
fix *x* **assume** *x* ∈ *space*$_N$ (*N1* +$_N$ *N2*)
then have *Inf* {*eNorm N1 f + eNorm N2 g*| *f g*. *x = f + g*} < ∞
unfolding *quasinorm-sum*(*1*) *spaceN-iff***.**
then have ∃ *z* ∈ {*eNorm N1 f + eNorm N2 g*| *f g*. *x = f + g*}. *z* < ∞
by (*simp add*: *Inf-less-iff*)
then show ∃*f g*. *x = f + g* ∧ *f* ∈ *space*$_N$ *N1* ∧ *g* ∈ *space*$_N$ *N2*
using *spaceN-iff* **by** *force*
next
fix *f g* **assume** *H*: *f* ∈ *space*$_N$ *N1 g* ∈ *space*$_N$ *N2*

have *Inf* $\{eNorm\ N1\ u + eNorm\ N2\ v |\ u\ v.\ f + g = u + v\} \leq eNorm\ N1\ f + eNorm\ N2\ g$
by (*rule Inf-lower*, *auto*)
also have $... < \infty$ **using** *spaceN-iff H* **by** *auto*
finally show $f + g \in space_N\ (N1 +_N N2)$
unfolding *spaceN-iff quasinorm-sum*(*1*)**.**
qed

lemma *quasinorm-sum-zerospace*:
$\{f + g\ |f\ g.\ f \in zero\text{-}space_N\ N1 \wedge g \in zero\text{-}space_N\ N2\} \subseteq zero\text{-}space_N\ (N1 +_N N2)$
proof (*auto*, *unfold zero-spaceN-iff*)
fix *f g* **assume** *H*: *eNorm N1 f = 0 eNorm N2 g = 0*
have *Inf* $\{eNorm\ N1\ f1 + eNorm\ N2\ f2 |\ f1\ f2.\ f + g = f1 + f2\} \leq 0$
by (*rule Inf-lower*, *auto*, *rule exI*[*of - f*], *auto simp add*: *H*)
then show *eNorm* $(N1 +_N N2)\ (f + g) = 0$ **unfolding** *quasinorm-sum*(*1*) **by** *auto*
qed

lemma *quasinorm-sum-subset*:
$N1 \subseteq_N N1 +_N N2\ N2 \subseteq_N N1 +_N N2$
by (*rule quasinorm-subsetI*[*of - - 1*], *auto simp add*: *quasinorm-sum*(*1*), *rule Inf-lower*, *auto*,
metis add.commute add.left-neutral eNorm-zero)+

lemma *quasinorm-sum-maximum*:
assumes $N1 \subseteq_N N\ N2 \subseteq_N N$
shows $N1 +_N N2 \subseteq_N N$
proof −
obtain *C1 C2*::*real* **where** $*$: $\bigwedge f.\ eNorm\ N\ f \leq C1 * eNorm\ N1\ f$
$\bigwedge f.\ eNorm\ N\ f \leq C2 * eNorm\ N2\ f$
$C1 \geq 0\ C2 \geq 0$
using *quasinorm-subsetD*[*OF assms*(*1*)] *quasinorm-subsetD*[*OF assms*(*2*)] **by** *blast*
have $**$: $eNorm\ N\ f \leq (defect\ N * max\ C1\ C2) * eNorm\ (N1 +_N N2)\ f$ **for** *f*
proof −
obtain *f1 f2* **where** *F*: $\bigwedge n.\ f = f1\ n + f2\ n$
$(\lambda n.\ eNorm\ N1\ (f1\ n) + eNorm\ N2\ (f2\ n)) \longrightarrow eNorm\ (N1 +_N N2)\ f$
using *quasinorm-sum-limit* **by** *blast*
have $eNorm\ N\ f \leq ennreal\ (defect\ N * max\ C1\ C2) * (eNorm\ N1\ (f1\ n) + eNorm\ N2\ (f2\ n))$ **for** *n*
proof −
have $eNorm\ N\ f \leq ennreal(defect\ N) * eNorm\ N\ (f1\ n) + ennreal(defect\ N) * eNorm\ N\ (f2\ n)$
unfolding ‹$f = f1\ n + f2\ n$› **using** *eNorm-triangular-ineq* **by** *auto*
also have $... \leq ennreal(defect\ N) * (C1 * eNorm\ N1\ (f1\ n)) + ennreal(defect\ N) * (C2 * eNorm\ N2\ (f2\ n))$
apply (*rule add-mono*) **by** (*rule mult-mono*, *simp*, *simp add*: $*$, *simp*, *simp*)+

also have ... $\leq$ *ennreal*(*defect N*) $*$ (*max C1 C2* $*$ *eNorm N1* (*f1 n*)) + *ennreal*(*defect N*) $*$ (*max C1 C2* $*$ *eNorm N2* (*f2 n*))
by (*auto intro*!:*add-mono mult-mono ennreal-leI*)
also have ... = *ennreal* (*defect N* $*$ *max C1 C2*) $*$ (*eNorm N1* (*f1 n*) + *eNorm N2* (*f2 n*))
apply (*subst ennreal-mult'*) **using** *defect-ge-1 order-trans zero-le-one* **apply** *blast*
by (*auto simp add*: *algebra-simps*)
finally show *?thesis* **by** *simp*
qed
moreover have ... $\longrightarrow$ (*defect N* $*$ *max C1 C2*) $*$ *eNorm* (*N1* $+_N$ *N2*) *f*
by (*auto intro*!:*tendsto-intros F*(*2*))
ultimately show *?thesis*
using *LIMSEQ-le-const* **by** *blast*
qed
then show *?thesis*
using *quasinorm-subsetI* **by** *force*
qed

lemma *quasinorm-sum-assoc*:
(*N1* $+_N$ *N2*) $+_N$ *N3* $=_N$ *N1* $+_N$ (*N2* $+_N$ *N3*)
unfolding *quasinorm-equivalent-def* **by** (*meson order-trans quasinorm-sum-maximum quasinorm-sum-subset*)

2.7 Topology

definition *topology*$_N$::('*a*::*real-vector*) *quasinorm* $\Rightarrow$ '*a topology*
where *topology*$_N$ *N* = *topology* ($\lambda U.\ \forall x{\in}U.\ \exists e{>}0.\ \forall y.\ eNorm\ N\ (y{-}x) < e \longrightarrow y \in U$)

lemma *istopology-topology*$_N$:
istopology ($\lambda U.\ \forall x{\in}U.\ \exists e{>}0.\ \forall y.\ eNorm\ N\ (y{-}x) < e \longrightarrow y \in U$)
unfolding *istopology-def* **by** (*auto*, *metis dual-order.strict-trans less-linear*, *meson*)

lemma *openin-topology*$_N$:
openin (*topology*$_N$ *N*) *U* $\longleftrightarrow$ ($\forall x{\in}U.\ \exists e{>}0.\ \forall y.\ eNorm\ N\ (y{-}x) < e \longrightarrow y \in U$)
unfolding *topology*$_N$*-def* **using** *istopology-topology*$_N$[*of N*] **by** (*simp add*: *topology-inverse'*)

lemma *openin-topology*$_N$*-I*:
assumes $\bigwedge x.\ x \in U \Longrightarrow \exists e{>}0.\ \forall y.\ eNorm\ N\ (y{-}x) < e \longrightarrow y \in U$
shows *openin* (*topology*$_N$ *N*) *U*
using *assms* **unfolding** *openin-topology*$_N$ **by** *auto*

lemma *openin-topology*$_N$*-D*:
assumes *openin* (*topology*$_N$ *N*) *U*
x $\in$ *U*

shows $\exists\, e{>}0.\ \forall\, y.\ eNorm\ N\ (y{-}x) < e \longrightarrow y \in U$
using *assms* **unfolding** *openin-topology*$_N$ **by** *auto*

One should then use this topology to define limits and so on. This is not something specific to quasinorms, but to all topologies defined in this way, not using type classes. However, there is no such body of material (yet?) in Isabelle-HOL, where topology is essentially done with type classes. So, we do not go any further for now.

One exception is the notion of completeness, as it is so important in functional analysis. We give a naive definition, which will be sufficient for the proof of completeness of several spaces. Usually, the most convenient criterion to prove completeness of a normed vector space is in terms of converging series. This criterion is the only nontrivial thing we prove here. We will apply it to prove the completeness of L^p spaces.

definition *cauchy-ine*$_N$::('*a*::*real-vector*) *quasinorm* ⇒ (*nat* ⇒ '*a*) ⇒ *bool*
where *cauchy-ine*$_N$ *N u* = (∀ *e*>*0*. ∃ *M*. ∀ *n*≥*M*. ∀ *m*≥*M*. *eNorm N* (*u n* − *u m*) < *e*)

definition *tendsto-ine*$_N$::('*a*::*real-vector*) *quasinorm* ⇒ (*nat* ⇒ '*a*) ⇒ '*a* => *bool*
where *tendsto-ine*$_N$ *N u x* = (λ*n*. *eNorm N* (*u n* − *x*)) ⟶ *0*

definition *complete*$_N$::('*a*::*real-vector*) *quasinorm* ⇒ *bool*
where *complete*$_N$ *N* = (∀ *u*. *cauchy-ine*$_N$ *N u* ⟶ (∃ *x*. *tendsto-ine*$_N$ *N u x*))

The above definitions are in terms of eNorms, but usually the nice definitions only make sense on the space of the norm, and are expressed in terms of Norms. We formulate the same definitions with norms, they will be more convenient for the proofs.

definition *cauchy-in*$_N$::('*a*::*real-vector*) *quasinorm* ⇒ (*nat* ⇒ '*a*) ⇒ *bool*
where *cauchy-in*$_N$ *N u* = (∀ *e*>*0*. ∃ *M*. ∀ *n*≥*M*. ∀ *m*≥*M*. *Norm N* (*u n* − *u m*) < *e*)

definition *tendsto-in*$_N$::('*a*::*real-vector*) *quasinorm* ⇒ (*nat* ⇒ '*a*) ⇒ '*a* => *bool*
where *tendsto-in*$_N$ *N u x* = (λ*n*. *Norm N* (*u n* − *x*)) ⟶ *0*

lemma *cauchy-ine*$_N$*-I*:
assumes ⋀*e*. *e* > *0* ⟹ (∃ *M*. ∀ *n*≥*M*. ∀ *m*≥*M*. *eNorm N* (*u n* − *u m*) < *e*)
shows *cauchy-ine*$_N$ *N u*
using *assms* **unfolding** *cauchy-ine*$_N$*-def* **by** *auto*

lemma *cauchy-in*$_N$*-I*:
assumes ⋀*e*. *e* > *0* ⟹ (∃ *M*. ∀ *n*≥*M*. ∀ *m*≥*M*. *Norm N* (*u n* − *u m*) < *e*)
shows *cauchy-in*$_N$ *N u*
using *assms* **unfolding** *cauchy-in*$_N$*-def* **by** *auto*

lemma *cauchy-ine-in*:

assumes $\bigwedge n.\ u\ n \in space_N\ N$
shows *cauchy-ine*$_N$ $N\ u \longleftrightarrow$ *cauchy-in*$_N$ $N\ u$
proof
assume *cauchy-in*$_N$ $N\ u$
show *cauchy-ine*$_N$ $N\ u$
proof (*rule cauchy-ine*$_N$*-I*)
fix *e*::*ennreal* **assume** $e > 0$
define *e2* **where** $e2 = min\ e\ 1$
then obtain r **where** $e2 = ennreal\ r\ r > 0$ **unfolding** *e2-def* **using** ‹$e > 0$›
by (*metis ennreal-eq-1 ennreal-less-zero-iff le-ennreal-iff le-numeral-extra*(*1*) *min-def zero-less-one*)
then obtain M **where** $*$: $\forall\, n{\geq}M.\ \forall\, m{\geq}M.\ Norm\ N\ (u\ n - u\ m) < r$
using ‹*cauchy-in*$_N$ $N\ u$› ‹$r > 0$› **unfolding** *cauchy-in*$_N$*-def* **by** *auto*
then have $\forall\, n{\geq}M.\ \forall\, m{\geq}M.\ eNorm\ N\ (u\ n - u\ m) < r$
by (*auto simp add*: *assms eNorm-Norm* ‹$0 < r$› *ennreal-lessI*)
then have $\forall\, n{\geq}M.\ \forall\, m{\geq}M.\ eNorm\ N\ (u\ n - u\ m) < e$
unfolding ‹$e2 = ennreal\ r$›[*symmetric*] *e2-def* **by** *auto*
then show $\exists\, M.\ \forall\, n{\geq}M.\ \forall\, m{\geq}M.\ eNorm\ N\ (u\ n - u\ m) < e$
by *auto*
qed
next
assume *cauchy-ine*$_N$ $N\ u$
show *cauchy-in*$_N$ $N\ u$
proof (*rule cauchy-in*$_N$*-I*)
fix *e*::*real* **assume** $e > 0$
then obtain M **where** $*$: $\forall\, n{\geq}M.\ \forall\, m{\geq}M.\ eNorm\ N\ (u\ n - u\ m) < e$
using ‹*cauchy-ine*$_N$ $N\ u$› ‹$e > 0$› *ennreal-less-zero-iff* **unfolding** *cauchy-ine*$_N$*-def*
by *blast*
then have $\forall\, n{\geq}M.\ \forall\, m{\geq}M.\ Norm\ N\ (u\ n - u\ m) < e$
by (*auto*, *metis Norm-def* ‹$0 < e$› *eNorm-Norm eNorm-Norm′ enn2real-nonneg ennreal-less-iff*)
then show $\exists\, M.\ \forall\, n{\geq}M.\ \forall\, m{\geq}M.\ Norm\ N\ (u\ n - u\ m) < e$
by *auto*
qed
qed

lemma *tendsto-ine-in*:
assumes $\bigwedge n.\ u\ n \in space_N\ N\ x \in space_N\ N$
shows *tendsto-ine*$_N$ $N\ u\ x \longleftrightarrow$ *tendsto-in*$_N$ $N\ u\ x$
proof −
have $*$: $eNorm\ N\ (u\ n - x) = Norm\ N\ (u\ n - x)$ **for** n
using *assms eNorm-Norm spaceN-diff* **by** *blast*
show *?thesis* **unfolding** *tendsto-in*$_N$*-def tendsto-ine*$_N$*-def* $*$
apply (*auto*)
apply (*metis* (*full-types*) *Norm-nonneg ennreal-0 eventually-sequentiallyI order-refl tendsto-ennreal-iff*)
using *tendsto-ennrealI* **by** *fastforce*
qed

lemma *complete$_N$-I*:
assumes $\bigwedge u.$ *cauchy-in$_N$* N $u \Longrightarrow (\forall\, n.\ u\ n \in$ *space$_N$* $N) \Longrightarrow (\exists\, x \in$ *space$_N$* $N.$ *tendsto-in$_N$* N u $x)$
shows *complete$_N$* N
proof −
have $\exists\, x.$ *tendsto-ine$_N$* N u x **if** *cauchy-ine$_N$* N u **for** u
proof −
obtain M::*nat* **where** $*$: $\bigwedge n\ m.\ n \geq M \Longrightarrow m \geq M \Longrightarrow$ *eNorm* N $(u\ n - u\ m) < 1$
using ‹*cauchy-ine$_N$* N u› *ennreal-zero-less-one* **unfolding** *cauchy-ine$_N$-def* **by** *presburger*
define v **where** $v = (\lambda n.\ u\ (n+M) - u\ M)$
have *eNorm* N $(v\ n) < 1$ **for** n **unfolding** *v-def* **using** $*$ **by** *auto*
then have $v\ n \in$ *space$_N$* N **for** n **using** *spaceN-iff*[*of* - N]
by (*metis dual-order.strict-trans ennreal-1 ennreal-less-top infinity-ennreal-def*)
have *cauchy-ine$_N$* N v
proof (*rule cauchy-ine$_N$-I*)
fix e::*ennreal* **assume** $e > 0$
then obtain P::*nat* **where** $*$: $\bigwedge n\ m.\ n \geq P \Longrightarrow m \geq P \Longrightarrow$ *eNorm* N $(u\ n - u\ m) < e$
using ‹*cauchy-ine$_N$* N u› **unfolding** *cauchy-ine$_N$-def* **by** *presburger*
have *eNorm* N $(v\ n - v\ m) < e$ **if** $n \geq P$ $m \geq P$ **for** m n
unfolding *v-def* **by** (*auto*, *rule* $*$, *insert that*, *auto*)
then show $\exists\, M.\ \forall\, n{\geq}M.\ \forall\, m{\geq}M.$ *eNorm* N $(v\ n - v\ m) < e$ **by** *auto*
qed
then have *cauchy-in$_N$* N v **using** *cauchy-ine-in*[*OF* ‹$\bigwedge n.\ v\ n \in$ *space$_N$* N›]
by *auto*
then obtain y **where** *tendsto-in$_N$* N v y $y \in$ *space$_N$* N
using *assms* ‹$\bigwedge n.\ v\ n \in$ *space$_N$* N› **by** *auto*
then have $*$: *tendsto-ine$_N$* N v y
using *tendsto-ine-in* ‹$\bigwedge n.\ v\ n \in$ *space$_N$* N› **by** *auto*
have *tendsto-ine$_N$* N u $(y + u\ M)$
unfolding *tendsto-ine$_N$-def* **apply** (*rule LIMSEQ-offset*[*of* - M])
using $*$ **unfolding** *v-def tendsto-ine$_N$-def* **by** (*auto simp add*: *algebra-simps*)
then show *?thesis* **by** *auto*
qed
then show *?thesis* **unfolding** *complete$_N$-def* **by** *auto*
qed

lemma *cauchy-tendsto-in-subseq*:
assumes $\bigwedge n.\ u\ n \in$ *space$_N$* N
cauchy-in$_N$ N u
strict-mono r
tendsto-in$_N$ N $(u\ o\ r)$ x
shows *tendsto-in$_N$* N u x
proof −
have $\exists\, M.\ \forall\, n{\geq}M.$ *Norm* N $(u\ n - x) < e$ **if** $e > 0$ **for** e
proof −
define f **where** $f = e\ /\ (2 * \mathit{defect}\ N)$

have $f > 0$ **unfolding** *f-def* **using** ‹$e > 0$› *defect-ge-1*[*of N*] **by** (*auto simp add*: *divide-simps*)
obtain *M1* **where** *M1*: $\bigwedge m\ n.\ m \geq M1 \Longrightarrow n \geq M1 \Longrightarrow Norm\ N\ (u\ n - u\ m) < f$
using ‹*cauchy-in*$_N$ $N\ u$› **unfolding** *cauchy-in*$_N$*-def* **using** ‹$f > 0$› **by** *meson*
obtain *M2* **where** *M2*: $\bigwedge n.\ n \geq M2 \Longrightarrow Norm\ N\ ((u\ o\ r)\ n - x) < f$
using ‹*tendsto-in*$_N$ $N\ (u\ o\ r)\ x$› ‹$f > 0$› **unfolding** *tendsto-in*$_N$*-def order-tendsto-iff eventually-sequentially* **by** *blast*
define M **where** $M = max\ M1\ M2$
have $Norm\ N\ (u\ n - x) < e$ **if** $n \geq M$ **for** n
proof −
have $Norm\ N\ (u\ n - x) = Norm\ N\ ((u\ n - u\ (r\ M)) + (u\ (r\ M) - x))$ **by** *auto*
also have ... $\leq$ $defect\ N * Norm\ N\ (u\ n - u\ (r\ M)) + defect\ N * Norm\ N\ (u\ (r\ M) - x)$
apply (*rule Norm-triangular-ineq*) **using** ‹$\bigwedge n.\ u\ n \in space_N\ N$› **by** *simp*
also have ... $<$ $defect\ N * f + defect\ N * f$
apply (*auto intro*!: *add-strict-mono mult-mono simp only*:)
using *defect-ge-1*[*of N*] ‹$n \geq M$› *seq-suble*[*OF* ‹*strict-mono r*›, *of M*] *M1 M2 o-def* **unfolding** *M-def* **by** *auto*
finally show *?thesis*
unfolding *f-def* **using** ‹$e > 0$› *defect-ge-1*[*of N*] **by** (*auto simp add*: *divide-simps*)
qed
then show *?thesis* **by** *auto*
qed
then show *?thesis*
unfolding *tendsto-in*$_N$*-def order-tendsto-iff eventually-sequentially* **using** *Norm-nonneg less-le-trans* **by** *blast*
qed

proposition *complete*$_N$*-I'*:
assumes $\bigwedge n.\ c\ n > 0$
$\bigwedge u.\ (\forall\, n.\ u\ n \in space_N\ N) \Longrightarrow (\forall\, n.\ Norm\ N\ (u\ n) \leq c\ n) \Longrightarrow \exists\, x \in space_N\ N.\ tendsto\text{-}in_N\ N\ (\lambda n.\ (\sum i \in \{0..<n\}.\ u\ i))\ x$
shows *complete*$_N$ N
proof (*rule complete*$_N$*-I*)
fix v **assume** *cauchy-in*$_N$ $N\ v\ \forall\, n.\ v\ n \in space_N\ N$
have $*$: $\exists\, y.\ (\forall\, m \geq y.\ \forall\, p \geq y.\ Norm\ N\ (v\ m - v\ p) < c\ (Suc\ n)) \wedge x < y$ **if** $\forall\, m \geq x.\ \forall\, p \geq x.\ Norm\ N\ (v\ m - v\ p) < c\ n$ **for** $x\ n$
proof −
obtain M **where** i: $\forall\, m \geq M.\ \forall\, p \geq M.\ Norm\ N\ (v\ m - v\ p) < c\ (Suc\ n)$
using ‹*cauchy-in*$_N$ $N\ v$› ‹$c\ (Suc\ n) > 0$› **unfolding** *cauchy-in*$_N$*-def* **by** (*meson zero-less-power*)
then show *?thesis*
apply (*intro exI*[*of - max M* (*x*+*1*)]) **by** *auto*
qed
have $\exists\, r.\ \forall\, n.\ (\forall\, m \geq r\ n.\ \forall\, p \geq r\ n.\ Norm\ N\ (v\ m - v\ p) < c\ n) \wedge r\ n < r\ (Suc\ n)$

apply (*intro dependent-nat-choice*) **using** ‹*cauchy-in*$_N$ *N v*› ‹$\bigwedge n.\ c\ n > 0$› ∗
unfolding *cauchy-in*$_N$*-def* **by** *auto*
then obtain *r* **where** *r*: *strict-mono r* $\bigwedge n.\ \forall\, m{\geq}r\ n.\ \forall\, p{\geq}r\ n.$ *Norm N* (*v m* − *v p*) < *c n*
by (*auto simp*: *strict-mono-Suc-iff*)
define *u* **where** *u* = (λ*n*. *v* (*r* (*Suc n*)) − *v* (*r n*))
have *u n* ∈ *space*$_N$ *N* **for** *n*
unfolding *u-def* **using** ‹$\forall\, n.\ v\ n \in$ *space*$_N$ *N*› **by** *simp*
moreover have *Norm N* (*u n*) ≤ *c n* **for** *n*
unfolding *u-def* **using** *r* **by** (*simp add*: *less-imp-le strict-mono-def*)
ultimately obtain *y* **where** *y*: *y* ∈ *space*$_N$ *N tendsto-in*$_N$ *N* (λ*n*. ($\sum i{\in}\{0..{<}n\}$. *u i*)) *y*
using *assms*(*2*) **by** *blast*
define *x* **where** *x* = *y* + *v* (*r* 0)
have *x* ∈ *space*$_N$ *N*
unfolding *x-def* **using** ‹*y* ∈ *space*$_N$ *N*› ‹$\forall\, n.\ v\ n \in$ *space*$_N$ *N*› **by** *simp*
have *Norm N* (*v* (*r n*) − *x*) = *Norm N* (($\sum i{\in}\{0..{<}n\}$. *u i*) − *y*) **for** *n*
proof −
have *v* (*r n*) = ($\sum i{\in}\{0..{<}n\}$. *u i*) + *v* (*r* 0) **for** *n*
unfolding *u-def* **by** (*induct n*, *auto*)
then show *?thesis* **unfolding** *x-def* **by** (*metis add-diff-cancel-right*)
qed
then have (λ*n*. *Norm N* (*v* (*r n*) − *x*)) ⟶ *0*
using *y*(*2*) **unfolding** *tendsto-in*$_N$*-def* **by** *auto*
then have *tendsto-in*$_N$ *N* (*v* ∘ *r*) *x*
unfolding *tendsto-in*$_N$*-def comp-def* **by** *force*
then have *tendsto-in*$_N$ *N v x*
using ‹$\forall\, n.\ v\ n \in$ *space*$_N$ *N*›
by (*intro cauchy-tendsto-in-subseq*[*OF* - ‹*cauchy-in*$_N$ *N v*› ‹*strict-mono r*›], *auto*)
then show $\exists\, x{\in}$*space*$_N$ *N*. *tendsto-in*$_N$ *N v x*
using ‹*x* ∈ *space*$_N$ *N*› **by** *blast*
qed

Next, we show when the two examples of norms we have introduced before, the ambient norm in a Banach space, and the norm on bounded continuous functions, are complete. We just have to translate in our setting the already known completeness of these spaces.

lemma *complete-N-of-norm*:
complete$_N$ (*N-of-norm*::*'a*::*banach quasinorm*)
proof (*rule complete*$_N$*-I*)
fix *u*::*nat* ⇒ *'a* **assume** *cauchy-in*$_N$ *N-of-norm u*
then have *Cauchy u* **unfolding** *Cauchy-def cauchy-in*$_N$*-def N-of-norm*(*2*) **by** (*simp add*: *dist-norm*)
then obtain *x* **where** *u* ⟶ *x* **using** *convergent-eq-Cauchy* **by** *blast*
then have *tendsto-in*$_N$ *N-of-norm u x* **unfolding** *tendsto-in*$_N$*-def N-of-norm*(*2*)
using *Lim-null tendsto-norm-zero-iff* **by** *fastforce*
moreover have *x* ∈ *space*$_N$ *N-of-norm* **by** *auto*
ultimately show $\exists\, x{\in}$*space*$_N$ *N-of-norm*. *tendsto-in*$_N$ *N-of-norm u x* **by** *auto*

qed

In the next statement, the assumption that `'a` is a metric space is not necessary, a topological space would be enough, but a statement about uniform convergence is not available in this setting. TODO: fix it.

lemma *complete-bcontfunN*:
complete$_N$ (*bcontfun*$_N$::('*a*::*metric-space* ⇒ '*b*::*banach*) *quasinorm*)
proof (*rule complete*$_N$*-I*)
fix *u*::*nat* ⇒ ('*a* ⇒ '*b*) **assume** *H*: *cauchy-in*$_N$ *bcontfun*$_N$ *u* ∀ *n*. *u n* ∈ *space*$_N$ *bcontfun*$_N$
then have *H2*: *u n* ∈ *bcontfun* **for** *n* **using** *bcontfun*$_N$*-space* **by** *auto*
then have ∗∗: *Bcontfun*(*u n* − *u m*) = *Bcontfun* (*u n*) − *Bcontfun* (*u m*) **for** *m n*
unfolding *minus-fun-def minus-bcontfun-def* **by** (*simp add*: *Bcontfun-inverse*)
have ∗: *Norm bcontfun*$_N$ (*u n* − *u m*) = *norm* (*Bcontfun* (*u n* − *u m*)) **for** *n m*
unfolding *bcontfun*$_N$(*2*) **using** *H*(*2*) *bcontfun*$_N$*-space* **by** *auto*
have *Cauchy* (λ*n*. *Bcontfun* (*u n*))
using *H*(*1*) **unfolding** *Cauchy-def cauchy-in*$_N$*-def dist-norm* ∗ ∗∗ **by** *simp*
then obtain *v* **where** *v*: (λ*n*. *Bcontfun* (*u n*)) ⟶ *v*
using *convergent-eq-Cauchy* **by** *blast*
have *v-space*: *apply-bcontfun v* ∈ *space*$_N$ *bcontfun*$_N$ **unfolding** *bcontfun*$_N$*-space* **by** (*simp add*: *apply-bcontfun*)
have ∗∗∗: *Norm bcontfun*$_N$ (*u n* − *v*) = *norm*(*Bcontfun* (*u n*) − *v*) **for** *n*
proof −
have *Norm bcontfun*$_N$ (*u n* − *v*) = *norm* (*Bcontfun*(*u n* − *v*))
unfolding *bcontfun*$_N$(*2*) **using** *H*(*2*) *bcontfun*$_N$*-space v-space* **by** *auto*
moreover have *Bcontfun*(*u n* − *v*) = *Bcontfun* (*u n*) − *v*
unfolding *minus-fun-def minus-bcontfun-def* **by** (*simp add*: *Bcontfun-inverse H2*)
ultimately show *?thesis* **by** *simp*
qed
have *tendsto-in*$_N$ *bcontfun*$_N$ *u v*
unfolding *tendsto-in*$_N$*-def* ∗∗∗ **using** *v Lim-null tendsto-norm-zero-iff* **by** *fastforce*
then show ∃ *v*∈*space*$_N$ *bcontfun*$_N$. *tendsto-in*$_N$ *bcontfun*$_N$ *u v* **using** *v-space* **by** *auto*
qed

end

theory *Lp*
imports *Functional-Spaces*
begin

The material in this file is essentially of analytic nature. However, one of the central proofs (the proof of Holder inequality below) uses a probability space, and Jensen's inequality there. Hence, we need to import `Probability`. Moreover, we use several lemmas from `SG_Library_Complement`.

3 Conjugate exponents

Two numbers p and q are *conjugate* if $1/p + 1/q = 1$. This relation keeps appearing in the theory of L^p spaces, as the dual of L^p is L^q where q is the conjugate of p. This relation makes sense for real numbers, but also for ennreals (where the case $p = 1$ and $q = \infty$ is most important). Unfortunately, manipulating the previous relation with ennreals is tedious as there is no good simproc involving addition and division there. To mitigate this difficulty, we prove once and for all most useful properties of such conjugates exponents in this paragraph.

lemma *Lp-cases-1-PInf*:
assumes $p \geq (1::ennreal)$
obtains (*gr*) *p2* **where** $p = ennreal\ p2\ p2 > 1\ p > 1$
| (*one*) $p = 1$
| (*PInf*) $p = \infty$
using *assms* **by** (*metis* (*full-types*) *antisym-conv ennreal-cases ennreal-le-1 infinity-ennreal-def not-le*)

lemma *Lp-cases*:
obtains (*real-pos*) *p2* **where** $p = ennreal\ p2\ p2 > 0\ p > 0$
| (*zero*) $p = 0$
| (*PInf*) $p = \infty$
by (*metis enn2real-positive-iff ennreal-enn2real-if infinity-ennreal-def not-gr-zero top.not-eq-extremum*)

definition
conjugate-exponent $p = 1 + 1/(p-1)$

lemma *conjugate-exponent-real*:
assumes $p > (1::real)$
shows $1/p + 1/(conjugate\text{-}exponent\ p) = 1$
conjugate-exponent $p > 1$
$conjugate\text{-}exponent(conjugate\text{-}exponent\ p) = p$
$(p-1) * conjugate\text{-}exponent\ p = p$
$p - p\ /\ conjugate\text{-}exponent\ p = 1$
unfolding *conjugate-exponent-def* **using** *assms* **by** (*auto simp add*: *algebra-simps divide-simps*)

lemma *conjugate-exponent-real-iff*:
assumes $p > (1::real)$
shows $q = conjugate\text{-}exponent\ p \longleftrightarrow (1/p + 1/q = 1)$
unfolding *conjugate-exponent-def* **using** *assms* **by** (*auto simp add*: *algebra-simps divide-simps*)

lemma *conjugate-exponent-real-2* [*simp*]:
conjugate-exponent $(2::real) = 2$
unfolding *conjugate-exponent-def* **by** (*auto simp add*: *algebra-simps divide-simps*)

lemma *conjugate-exponent-realI*:
assumes $p > (0::real)$ $q > 0$ $1/p + 1/q = 1$
shows $p > 1$ $q = conjugate\text{-}exponent\ p$ $q > 1$ $p = conjugate\text{-}exponent\ q$
unfolding *conjugate-exponent-def* **using** *assms* **apply** (*auto simp add*: *algebra-simps divide-simps*)
apply (*metis assms*(*3*) *divide-less-eq-1-pos less-add-same-cancel1 zero-less-divide-1-iff*)
using *mult-less-cancel-left-pos* **by** *fastforce*

lemma *conjugate-exponent-real-ennreal*:
assumes $p> (1::real)$
shows $conjugate\text{-}exponent(ennreal\ p) = ennreal(conjugate\text{-}exponent\ p)$
unfolding *conjugate-exponent-def* **using** *assms*
by (*auto*, *metis diff-gt-0-iff-gt divide-ennreal ennreal-1 ennreal-minus zero-le-one*)

lemma *conjugate-exponent-ennreal-1-2-PInf* [*simp*]:
$conjugate\text{-}exponent\ (1::ennreal) = \infty$
$conjugate\text{-}exponent\ (\infty::ennreal) = 1$
$conjugate\text{-}exponent\ (\top::ennreal) = 1$
$conjugate\text{-}exponent\ (2::ennreal) = 2$
using *conjugate-exponent-real-ennreal*[*of 2*] **by** (*auto simp add*: *conjugate-exponent-def*)

lemma *conjugate-exponent-ennreal*:
assumes $p \geq (1::ennreal)$
shows $1/p + 1/(conjugate\text{-}exponent\ p) = 1$
$conjugate\text{-}exponent\ p \geq 1$
$conjugate\text{-}exponent(conjugate\text{-}exponent\ p) = p$
proof −
have $(1/p + 1/(conjugate\text{-}exponent\ p) = 1) \wedge (conjugate\text{-}exponent\ p \geq 1) \wedge conjugate\text{-}exponent(conjugate\text{-}exponent\ p) = p$
using ‹$p \geq 1$› **proof** (*cases rule*: *Lp-cases-1-PInf*)
case (*gr p2*)
then have ∗: $conjugate\text{-}exponent\ p = ennreal\ (conjugate\text{-}exponent\ p2)$ **using** *conjugate-exponent-real-ennreal*[*OF* ‹*p2* > *1*›] **by** *auto*
have *a*: $conjugate\text{-}exponent\ p \geq 1$ **using** ∗ *conjugate-exponent-real*[*OF* ‹*p2* > *1*›] **by** *auto*
have *b*: $conjugate\text{-}exponent(conjugate\text{-}exponent\ p) = p$
using *conjugate-exponent-real*(*3*)[*OF* ‹*p2* > *1*›] *conjugate-exponent-real-ennreal*[*OF* ‹*p2* > *1*›]
conjugate-exponent-real-ennreal[*OF conjugate-exponent-real*(*2*)[*OF* ‹*p2* > *1*›]]
unfolding ∗ ‹$p = ennreal\ p2$› **by** *auto*
have $1 / p + 1 / conjugate\text{-}exponent\ p = ennreal(1/p2 + 1/(conjugate\text{-}exponent\ p2))$ **unfolding** ∗ **unfolding** ‹$p = ennreal\ p2$›
using *conjugate-exponent-real*(*2*)[*OF* ‹*p2* > *1*›] ‹*p2* > *1*›
apply (*subst ennreal-plus*, *auto*) **apply** (*subst divide-ennreal*[*symmetric*], *auto*)
using *divide-ennreal-def inverse-ennreal inverse-eq-divide* **by** *auto*
then have *c*: $1 / p + 1 / conjugate\text{-}exponent\ p = 1$ **using** *conjugate-exponent-real*[*OF* ‹*p2* > *1*›] **by** *auto*
show *?thesis* **using** *a b c* **by** *simp*

qed (*auto*)
then show $1/p + 1/(\textit{conjugate-exponent}\ p) = 1$
$\textit{conjugate-exponent}\ p \geq 1$
$\textit{conjugate-exponent}(\textit{conjugate-exponent}\ p) = p$
by *auto*
qed

lemma *conjugate-exponent-ennreal-iff*:
assumes $p \geq (1::\textit{ennreal})$
shows $q = \textit{conjugate-exponent}\ p \longleftrightarrow (1/p + 1/q = 1)$
using *conjugate-exponent-ennreal*[*OF assms*]
by (*auto*, *metis ennreal-add-diff-cancel-left ennreal-add-eq-top ennreal-top-neq-one one-divide-one-divide-ennreal*)

lemma *conjugate-exponent-ennrealI*:
assumes $1/p + 1/q = (1::\textit{ennreal})$
shows $p \geq 1\ q \geq 1\ p = \textit{conjugate-exponent}\ q\ q = \textit{conjugate-exponent}\ p$
proof −
have $1/p \leq 1$ **using** *assms* **using** *le-iff-add* **by** *fastforce*
then show $p \geq 1$
by (*metis assms divide-ennreal-def ennreal-add-eq-top ennreal-divide-self ennreal-divide-zero ennreal-le-epsilon ennreal-one-neq-top mult.left-neutral mult-left-le zero-le*)
then show $q = \textit{conjugate-exponent}\ p$ **using** *conjugate-exponent-ennreal-iff assms* **by** *auto*
then show $q \geq 1$ **using** *conjugate-exponent-ennreal*[*OF* ‹$p \geq 1$›] **by** *auto*
show $p = \textit{conjugate-exponent}\ q$
using *conjugate-exponent-ennreal-iff*[*OF* ‹$q \geq 1$›, *of p*] *assms* **by** (*simp add*: *add.commute*)
qed

4 Convexity inequalities and integration

In this paragraph, we describe the basic inequalities relating the integral of a function and of its p-th power, for $p > 0$. These inequalities imply in particular that the L^p norm satisfies the triangular inequality, a feature we will need when defining the L^p spaces below. In particular, we prove the Hölder and Minkowski inequalities. The Hölder inequality, especially, is the basis of all further inequalities for L^p spaces.

lemma (**in** *prob-space*) *bound-L1-Lp*:
assumes $p \geq (1::\textit{real})$
$f \in \textit{borel-measurable}\ M$
$\textit{integrable}\ M\ (\lambda x.\ |f\ x|\ \textit{powr}\ p)$
shows $\textit{integrable}\ M\ f$
$\textit{abs}(\int x.\ f\ x\ \partial M)\ \textit{powr}\ p \leq (\int x.\ |f\ x|\ \textit{powr}\ p\ \partial M)$
$\textit{abs}(\int x.\ f\ x\ \partial M) \leq (\int x.\ |f\ x|\ \textit{powr}\ p\ \partial M)\ \textit{powr}\ (1/p)$
proof −

have $*$: *norm x* $\leq$ *1* + (*norm x*) *powr p* **for** *x*::*real*
apply (*cases norm x* $\leq$ *1*)
apply (*meson le-add-same-cancel1 order.trans powr-ge-zero*)
apply (*metis add-le-same-cancel2 assms*(*1*) *less-le-trans linear not-less not-one-le-zero powr-le-cancel-iff powr-one-gt-zero-iff*)
done
show $*$: *integrable M f*
apply (*rule Bochner-Integration.integrable-bound*[*of* - $\lambda x.$ *1* + |*f x*| *powr p*], *auto simp add*: *assms*) **using** $*$ **by** *auto*
show *abs*($\int$ *x. f x* ∂*M*) *powr p* $\leq$ ($\int$ *x.* |*f x*| *powr p* ∂*M*)
by (*rule jensens-inequality*[*OF* $*$ - - *assms*(*3*) *convex-abs-powr*[*OF* ‹*p* $\geq$ *1*›]], *auto*)
then have (*abs*($\int$ *x. f x* ∂*M*) *powr p*) *powr* (*1*/*p*) $\leq$ ($\int$ *x.* |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
using *assms*(*1*) *powr-mono2* **by** *auto*
then show *abs*($\int$ *x. f x* ∂*M*) $\leq$ ($\int$ *x.* |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
using ‹*p* $\geq$ *1*› **by** (*auto simp add*: *powr-powr*)
qed

theorem *Holder-inequality*:
assumes *p* > (*0*::*real*) *q* > *0* *1*/*p* + *1*/*q* = *1*
and [*measurable*]: *f* $\in$ *borel-measurable M g* $\in$ *borel-measurable M*
integrable M ($\lambda x.$ |*f x*| *powr p*)
integrable M ($\lambda x.$ |*g x*| *powr q*)
shows *integrable M* ($\lambda x.$ *f x* $*$ *g x*)
($\int$ *x.* |*f x* $*$ *g x*| ∂*M*) $\leq$ ($\int$ *x.* |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*) $*$ ($\int$ *x.* |*g x*| *powr q* ∂*M*) *powr* (*1*/*q*)
abs($\int$ *x. f x* $*$ *g x* ∂*M*) $\leq$ ($\int$ *x.* |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*) $*$ ($\int$ *x.* |*g x*| *powr q* ∂*M*) *powr* (*1*/*q*)
proof −
have *p* > *1* **using** *conjugate-exponent-realI*(*1*)[*OF* ‹*p*>*0*› ‹*q*>*0*› ‹*1*/*p*+*1*/*q*=*1*›].

have $*$: *x* $*$ *y* $\leq$ *x powr p* + *y powr q* **if** *x* $\geq$ *0* *y* $\geq$ *0* **for** *x y*
proof −
have *x* $*$ *y* = (*x powr p*) *powr* (*1*/*p*) $*$ (*y powr q*) *powr* (*1*/*q*)
using ‹*p* > *0*› ‹*q* > *0*› *powr-powr that*(*1*) *that*(*2*) **by** *auto*
also have ... $\leq$ (*max* (*x powr p*) (*y powr q*)) *powr* (*1*/*p*) $*$ (*max* (*x powr p*) (*y powr q*)) *powr* (*1*/*q*)
apply (*rule mult-mono*, *auto*) **using** *assms*(*1*) *assms*(*2*) *powr-mono2* **by** *auto*
also have ... = *max* (*x powr p*) (*y powr q*)
by (*metis max-def mult.right-neutral powr-add powr-powr assms*(*3*))
also have ... $\leq$ *x powr p* + *y powr q*
by *auto*
finally show *?thesis* **by** *simp*
qed
show [*simp*]: *integrable M* ($\lambda x.$ *f x* $*$ *g x*)
apply (*rule Bochner-Integration.integrable-bound*[*of* - $\lambda x.$ |*f x*| *powr p* + |*g x*| *powr q*], *auto*)

by (*rule Bochner-Integration.integrable-add*, *auto simp add*: *assms* ∗ *abs-mult*)

The proof of the main inequality is done by applying the inequality $(\int |h| d\mu \leq \int |h|^p d\mu)^{1/p}$ to the right function h in the right probability space. One should take $h = f \cdot |g|^{1-q}$, and $d\mu = |g|^q dM/I$, where $I = \int |g|^q$. This readily gives the result.

show ∗: (∫ *x*. |*f x* ∗ *g x*| ∂*M*) ≤ (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (1/*p*) ∗ (∫ *x*. |*g x*| *powr q* ∂*M*) *powr* (1/*q*)
proof (*cases* (∫ *x*. |*g x*| *powr q* ∂*M*) = 0)
case *True*
then have *AE x in M*. |*g x*| *powr q* = 0
by (*subst integral-nonneg-eq-0-iff-AE*[*symmetric*], *auto simp add*: *assms*)
then have ∗: *AE x in M*. *f x* ∗ *g x* = 0
using ‹*q* > 0› **by** *auto*
have (∫ *x*. |*f x* ∗ *g x*| ∂*M*) = (∫ *x*. 0 ∂*M*)
apply (*rule integral-cong-AE*) **using** ∗ **by** *auto*
then show *?thesis* **by** *auto*
next
case *False*
moreover have (∫ *x*. |*g x*| *powr q* ∂*M*) ≥ (∫ *x*. 0 ∂*M*) **by** (*rule integral-mono*, *auto simp add*: *assms*)
ultimately have ∗: (∫ *x*. |*g x*| *powr q* ∂*M*) > 0 **by** (*simp add*: *le-less*)
define *I* **where** *I* = (∫ *x*. |*g x*| *powr q* ∂*M*)
have [*simp*]: *I* > 0 **unfolding** *I-def* **using** ∗ **by** *auto*
define *M2* **where** *M2* = *density M* (λ*x*. |*g x*| *powr q* / *I*)
interpret *prob-space M2*
apply (*standard*, *unfold M2-def*, *auto*, *subst emeasure-density*, *auto*)
apply (*subst divide-ennreal*[*symmetric*], *auto*, *subst nn-integral-divide*, *auto*)
apply (*subst nn-integral-eq-integral*, *auto simp add*: *assms*, *unfold I-def*)
using ∗ **by** *auto*

have [*simp*]: *p* ≥ 1 *p* ≥ 0 **using** ‹*p* > 1› **by** *auto*
have *A*: *q* + (1 − *q*) ∗ *p* = 0 **using** *assms* **by** (*auto simp add*: *divide-simps algebra-simps*)
have *B*: 1 − 1/*p* = 1/*q* **using** ‹1/*p* + 1/*q* = 1› **by** *auto*
define *f2* **where** *f2* = (λ*x*. *f x* ∗ *indicator* {*y*∈ *space M*. *g y* ≠ 0} *x*)
have [*measurable*]: *f2* ∈ *borel-measurable M* **unfolding** *f2-def* **by** *auto*
define *h* **where** *h* = (λ*x*. |*f2 x*| ∗ |*g x*| *powr* (1−*q*))
have [*measurable*]: *h* ∈ *borel-measurable M* **unfolding** *h-def* **by** *auto*
have [*measurable*]: *h* ∈ *borel-measurable M2* **unfolding** *M2-def* **by** *auto*

have *Eq*: (|*g x*| *powr q* / *I*) ∗$_R$ |*h x*| *powr p* = |*f2 x*| *powr p* / *I* **for** *x*
apply (*insert* ‹*I*>0›, *auto simp add*: *divide-simps*, *unfold h-def*)
apply (*auto simp add*: *divide-nonneg-pos divide-simps powr-mult powr-powr powr-add*[*symmetric*] *A*)
unfolding *f2-def* **by** *auto*
have *integrable M2* (λ*x*. |*h x*| *powr p*)
unfolding *M2-def* **apply** (*subst integrable-density*, *simp*, *simp*, *simp add*: *divide-simps*)

apply (*subst Eq*, *rule integrable-divide*, *rule Bochner-Integration.integrable-bound*[*of* - $\lambda x.\ |f\ x|\ powr\ p$], *unfold f2-def*)
by (*unfold indicator-def*, *auto simp add*: ‹*integrable M* ($\lambda x.\ |f\ x|\ powr\ p$)›)
then have *integrable M2* ($\lambda x.\ |h\ x|$)
by (*metis bound-L1-Lp*(*1*) ‹*random-variable borel h*› ‹$p > 1$› *integrable-abs le-less*)

have $(\int x.\ |h\ x|\ powr\ p\ \partial M2) = (\int x.\ (|g\ x|\ powr\ q\ /\ I) *_R (|h\ x|\ powr\ p)\ \partial M)$
unfolding *M2-def* **by** (*rule integral-density*[*of* $\lambda x.\ |h\ x|\ powr\ p$ *M* $\lambda x.\ |g\ x|\ powr\ q\ /\ I$], *auto simp add*: *divide-simps*)
also have ... $= (\int x.\ |f2\ x|\ powr\ p\ /\ I\ \partial M)$
apply (*rule Bochner-Integration.integral-cong*) **using** *Eq* **by** *auto*
also have ... $\leq (\int x.\ |f\ x|\ powr\ p\ /\ I\ \partial M)$
apply (*rule integral-mono'*, *rule integrable-divide*[*OF* ‹*integrable M* ($\lambda x.\ |f\ x|\ powr\ p$)›])
unfolding *f2-def indicator-def* **using** ‹$I > 0$› **by** (*auto simp add*: *divide-simps*)
finally have *C*: $(\int x.\ |h\ x|\ powr\ p\ \partial M2) \leq (\int x.\ |f\ x|\ powr\ p\ /\ I\ \partial M)$ **by** *simp*

have $(\int x.\ |f\ x * g\ x|\ \partial M)\ /\ I = (\int x.\ |f\ x * g\ x|\ /\ I\ \partial M)$
by *auto*
also have ... $= (\int x.\ |f2\ x * g\ x|\ /\ I\ \partial M)$
by (*auto simp add*: *divide-simps*, *rule Bochner-Integration.integral-cong*, *unfold f2-def indicator-def*, *auto*)
also have ... $= (\int x.\ |h\ x|\ \partial M2)$
apply (*unfold M2-def*, *subst integral-density*, *simp*, *simp*, *simp add*: *divide-simps*)
by (*rule Bochner-Integration.integral-cong*, *unfold h-def*, *auto simp add*: *divide-simps algebra-simps powr-add*[*symmetric*] *abs-mult*)
also have ... $\leq abs\ (\int x.\ |h\ x|\ \partial M2)$
by *auto*
also have ... $\leq (\int x.\ abs(|h\ x|)\ powr\ p\ \partial M2)\ powr\ (1/p)$
apply (*rule bound-L1-Lp*(*3*)[*of p* $\lambda x.\ |h\ x|$])
by (*auto simp add*: ‹*integrable M2* ($\lambda x.\ |h\ x|\ powr\ p$)›)
also have ... $\leq (\int x.\ |f\ x|\ powr\ p\ /\ I\ \partial M)\ powr\ (1/p)$
by (*rule powr-mono2*, *insert C*, *auto*)
also have ... $\leq ((\int x.\ |f\ x|\ powr\ p\ \partial M)\ /\ I)\ powr\ (1/p)$
apply (*rule powr-mono2*, *auto simp add*: *divide-simps*) **using** ‹$p \geq 0$› **by** *auto*
also have ... $= (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p) * I\ powr(-1/p)$
by (*auto simp add*: *less-imp-le powr-divide powr-minus-divide*)
finally have $(\int x.\ |f\ x * g\ x|\ \partial M) \leq (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p) * I * I\ powr(-1/p)$
by (*auto simp add*: *divide-simps algebra-simps*)
also have ... $= (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p) * I\ powr\ (1-1/p)$
by (*auto simp add*: *powr-mult-base less-imp-le*)
also have ... $= (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p) * (\int x.\ |g\ x|\ powr\ q\ \partial M)\ powr\ (1/q)$
unfolding *I-def* **using** *B* **by** *auto*
finally show *?thesis*

```
    by simp
  qed
  have abs(∫ x. f x * g x ∂M) ≤ (∫ x. |f x * g x| ∂M) by auto
  then show abs(∫ x. f x * g x ∂M) ≤ (∫ x. |f x| powr p ∂M) powr (1/p) * (∫ x.
|g x| powr q ∂M) powr (1/q)
    using * by linarith
qed

theorem Minkowski-inequality:
  assumes p ≥ (1::real)
      and [measurable, simp]: f ∈ borel-measurable M g ∈ borel-measurable M
          integrable M (λx. |f x| powr p)
          integrable M (λx. |g x| powr p)
  shows integrable M (λx. |f x + g x| powr p)
        (∫ x. |f x + g x| powr p ∂M) powr (1/p)
         ≤ (∫ x. |f x| powr p ∂M) powr (1/p) + (∫ x. |g x| powr p ∂M) powr (1/p)
proof −
  have *: |x + y| powr p ≤ 2 powr p * (|x| powr p + |y| powr p) for x y::real
  proof −
    have |x + y| ≤ |x| + |y| by auto
    also have ... ≤ (max |x| |y|) + max |x| |y| by auto
    also have ... = 2 * max |x| |y| by auto
    finally have |x + y| powr p ≤ (2 * max |x| |y|) powr p
      using powr-mono2 ‹p ≥ 1› by auto
    also have ... = 2 powr p * (max |x| |y|) powr p
      using powr-mult by auto
    also have ... ≤ 2 powr p * (|x| powr p + |y| powr p)
      unfolding max-def by auto
    finally show ?thesis by simp
  qed
  show [simp]: integrable M (λx. |f x + g x| powr p)
    by (rule Bochner-Integration.integrable-bound[of - λx. 2 powr p * (|f x| powr p
+ |g x| powr p)], auto simp add: *)

  show (∫ x. |f x + g x| powr p ∂M) powr (1/p) ≤ (∫ x. |f x| powr p ∂M) powr
(1/p) + (∫ x. |g x| powr p ∂M) powr (1/p)
  proof (cases p=1)
    case True
    then show ?thesis
    apply (auto, subst Bochner-Integration.integral-add[symmetric], insert assms(4)
assms(5), simp, simp)
      by (rule integral-mono', auto)
  next
    case False
    then have [simp]: p > 1 p ≥ 1 p > 0 p ≠ 0 using assms(1) by auto
    define q where q = conjugate-exponent p
    have [simp]: q > 1 q > 0 1/p + 1/q = 1 (p−1) * q = p
      unfolding q-def using conjugate-exponent-real[OF ‹p>1›] by auto
    then have [simp]: (z powr (p−1)) powr q = z powr p for z
```

by (*simp add*: *powr-powr*)
have $(\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M) = (\int x.\ |f\ x + g\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M)$
by (*subst powr-mult-base*, *auto*)
also have ... $\leq (\int x.\ |f\ x| * |f\ x + g\ x|\ powr\ (p-1) + |g\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M)$
apply (*rule integral-mono'*, *rule Bochner-Integration.integrable-add*)
apply (*rule Holder-inequality*(*1*)[*of p q*], *auto*)
apply (*rule Holder-inequality*(*1*)[*of p q*], *auto*)
by (*metis abs-ge-zero abs-triangle-ineq comm-semiring-class.distrib le-less mult-mono' powr-ge-zero*)
also have ... $= (\int x.\ |f\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M) + (\int x.\ |g\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M)$
apply (*rule Bochner-Integration.integral-add*) **by** (*rule Holder-inequality*(*1*)[*of p q*], *auto*)+
also have ... $\leq abs\ (\int x.\ |f\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M) + abs\ (\int x.\ |g\ x| * |f\ x + g\ x|\ powr\ (p-1)\ \partial M)$
by *auto*
also have ... $\leq (\int x.\ abs(|f\ x|)\ powr\ p\ \partial M)\ powr\ (1/p) * (\int x.\ abs(|f\ x + g\ x|\ powr\ (p-1))\ powr\ q\ \partial M)\ powr\ (1/q)$
$+ (\int x.\ abs(|g\ x|)\ powr\ p\ \partial M)\ powr\ (1/p) * (\int x.\ abs(|f\ x + g\ x|\ powr\ (p-1))\ powr\ q\ \partial M)\ powr\ (1/q)$
apply (*rule add-mono*)
apply (*rule Holder-inequality*(*3*)[*of p q*], *simp*, *simp*, *simp*, *simp*, *simp*, *simp*, *simp*)
apply (*rule Holder-inequality*(*3*)[*of p q*], *simp*, *simp*, *simp*, *simp*, *simp*, *simp*, *simp*)
done
also have ... $= (\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)\ powr\ (1/q) *$
$((\int x.\ abs(|f\ x|)\ powr\ p\ \partial M)\ powr\ (1/p) + (\int x.\ abs(|g\ x|)\ powr\ p\ \partial M)\ powr\ (1/p))$
by (*auto simp add*: *algebra-simps*)
finally have $*$: $(\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M) \leq (\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)\ powr\ (1/q) *$
$((\int x.\ abs(|f\ x|)\ powr\ p\ \partial M)\ powr\ (1/p) + (\int x.\ abs(|g\ x|)\ powr\ p\ \partial M)\ powr\ (1/p))$
by *simp*
show *?thesis*
proof (*cases* $(\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M) = 0$)
case *True*
then show *?thesis* **by** *auto*
next
case *False*
then have $**$: $(\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)\ powr\ (1/q) > 0$
by *auto*
have $(\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)\ powr\ (1/q) * (\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$= (\int x.\ |f\ x + g\ x|\ powr\ p\ \partial M)$
by (*auto simp add*: *powr-add*[*symmetric*] *add.commute*)

```
    then have (∫ x. |f x + g x| powr p ∂M) powr (1/q) * (∫ x. |f x + g x| powr
p ∂M) powr (1/p) ≤
            (∫ x. |f x + g x| powr p ∂M) powr (1/q) *
            ((∫ x. abs(|f x|) powr p ∂M) powr (1/p) + (∫ x. abs(|g x|) powr p ∂M)
powr (1/p))
      using * by auto
    then show ?thesis using ** by auto
  qed
 qed
qed
```

When $p < 1$, the function $x \mapsto |x|^p$ is not convex any more. Hence, the L^p "norm" is not a norm any more, but a quasinorm. This is proved using a different convexity argument, as follows.

```
theorem Minkowski-inequality-le-1:
 assumes p > (0::real) p ≤ 1
    and [measurable, simp]: f ∈ borel-measurable M g ∈ borel-measurable M
        integrable M (λx. |f x| powr p)
        integrable M (λx. |g x| powr p)
 shows integrable M (λx. |f x + g x| powr p)
       (∫ x. |f x + g x| powr p ∂M) powr (1/p)
       ≤ 2 powr (1/p-1) * (∫ x. |f x| powr p ∂M) powr (1/p) + 2 powr (1/p-1)
* (∫ x. |g x| powr p ∂M) powr (1/p)
proof −
 have *: |a + b| powr p ≤ |a| powr p + |b| powr p for a b
   using x-plus-y-p-le-xp-plus-yp[OF ‹p > 0› ‹p ≤ 1›, of |a| |b|]
  by (auto, meson abs-ge-zero abs-triangle-ineq assms(1) le-less order.trans powr-mono2)
 show integrable M (λx. |f x + g x| powr p)
   by (rule Bochner-Integration.integrable-bound[of - λx. |f x| powr p + |g x| powr
p], auto simp add: *)

 have (∫ x. |f x + g x| powr p ∂M) powr (1/p) ≤ (∫ x. |f x| powr p + |g x| powr
p ∂M) powr (1/p)
   by (rule powr-mono2, simp add: ‹p > 0› less-imp-le, simp, rule integral-mono′,
auto simp add: *)
 also have ... = 2 powr (1/p) * (((∫ x. |f x| powr p ∂M) + (∫ x. |g x| powr p
∂M)) / 2) powr (1/p)
   by (auto simp add: powr-mult[symmetric] add-divide-distrib)
 also have ... ≤ 2 powr (1/p) * (((∫ x. |f x| powr p ∂M) powr (1/p) + (∫ x. |g
x| powr p ∂M) powr (1/p)) / 2)
    apply (rule mult-mono, simp, rule convex-on-mean-ineq[OF convex-powr[of
1/p]])
   using ‹p ≤ 1› ‹p > 0› by auto
 also have ... = 2 powr (1/p − 1) * ((∫ x. |f x| powr p ∂M) powr (1/p) + (∫ x.
|g x| powr p ∂M) powr (1/p))
   by (simp add: powr-diff)
 finally show (∫ x. |f x + g x| powr p ∂M) powr (1/p)
     ≤ 2 powr (1/p-1) * (∫ x. |f x| powr p ∂M) powr (1/p) + 2 powr (1/p-1)
* (∫ x. |g x| powr p ∂M) powr (1/p)
```

by (*auto simp add*: *algebra-simps*)
qed

5 L^p spaces

We define L^p spaces by giving their defining quasinorm. It is a norm for $p \in [1, \infty]$, and a quasinorm for $p \in (0, 1)$. The construction of a quasinorm from a formula only makes sense if this formula is indeed a quasinorm, i.e., it is homogeneous and satisfies the triangular inequality with the given multiplicative defect. Thus, we have to show that this is indeed the case to be able to use the definition.

definition *Lp-space*::*ennreal* ⇒ *'a measure* ⇒ (*'a* ⇒ *real*) *quasinorm*
where *Lp-space p M* = (
if p = *0 then quasinorm-of* (*1*, (λ*f*. *if* (*f* ∈ *borel-measurable M*) *then 0 else* ∞))
else if p < ∞ *then quasinorm-of* (
if p < *1 then 2 powr* (*1*/*enn2real p* − *1*) *else 1*,
(λ*f*. *if* (*f* ∈ *borel-measurable M* ∧ *integrable M* (λ*x*. |*f x*| *powr* (*enn2real p*)))
then (∫ *x*. |*f x*| *powr* (*enn2real p*) ∂*M*) *powr* (*1*/(*enn2real p*))
else (∞::*ennreal*)))
else quasinorm-of (*1*, (λ*f*. *if f* ∈ *borel-measurable M then esssup M* (λ*x*. *ereal* |*f x*|) *else* (∞::*ennreal*))))

abbreviation 𝔏 == *Lp-space*

5.1 L^∞

Let us check that, for L^∞, the above definition makes sense.

lemma *L-infinity*:
eNorm (𝔏 ∞ *M*) *f* = (*if f* ∈ *borel-measurable M then esssup M* (λ*x*. *ereal* |*f x*|) *else* (∞::*ennreal*))
defect (𝔏 ∞ *M*) = *1*
proof −
have *T*: *esssup M* (λ*x*. *ereal* |(*f* + *g*) *x*|) ≤ *e2ennreal* (*esssup M* (λ*x*. *ereal* |*f x*|)) + *esssup M* (λ*x*. *ereal* |*g x*|)
if [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M* **for** *f g*
proof (*cases emeasure M* (*space M*) = *0*)
case *True*
then have *e2ennreal* (*esssup M* (λ*x*. *ereal* |(*f* + *g*) *x*|)) = *0*
using *esssup-zero-space*[*OF True*] **by** (*simp add*: *e2ennreal-neg*)
then show *?thesis* **by** *simp*
next
case *False*
have ∗: *esssup M* (λ*x*. |*h x*|) ≥ *0* **for** *h*::*'a* ⇒ *real*
proof −
have *esssup M* (λ*x*. *0*) ≤ *esssup M* (λ*x*. |*h x*|) **by** (*rule esssup-mono*, *auto*)

```
    then show ?thesis using esssup-const[OF False, of 0::ereal] by simp
   qed
   have esssup M (λx. ereal |(f + g) x|) ≤ esssup M (λx. ereal |f x| + ereal |g x|)
    by (rule esssup-mono, auto simp add: plus-fun-def)
   also have ... ≤ esssup M (λx. ereal |f x|) + esssup M (λx. ereal |g x|)
    by (rule esssup-add)
   finally show ?thesis
    using ∗ by (simp add: e2ennreal-mono eq-onp-def plus-ennreal.abs-eq)
  qed

  have H: esssup M (λx. ereal |(c ∗R f) x|) ≤ ennreal |c| ∗ esssup M (λx. ereal |f
x|) if c ≠ 0 for f c
  proof −
   have abs c > 0 ereal |c| ≥ 0 using that by auto
   have ∗: esssup M (λx. abs(c ∗R f x)) = abs c ∗ esssup M (λx. |f x|)
    apply (subst esssup-cmult[OF ‹abs c > 0›, of M λx. ereal |f x|, symmetric])
    using times-ereal.simps(1) by (auto simp add: abs-mult)
   show ?thesis
    unfolding e2ennreal-mult[OF ‹ereal |c| ≥ 0›] ∗ scaleR-fun-def
    by simp
  qed

  have esssup M (λx. ereal 0) ≤ 0 using esssup-I by auto
  then have Z: e2ennreal (esssup M (λx. ereal 0)) = 0 using e2ennreal-neg by
auto

  have ∗: quasinorm-on (borel-measurable M) 1 (λ(f::'a⇒real). e2ennreal(esssup
M (λx. ereal |f x|)))
   apply (rule quasinorm-onI) using T H Z by auto
  have ∗∗: quasinorm-on UNIV 1 (λ(f::'a⇒real). if f ∈ borel-measurable M then
e2ennreal(esssup M (λx. ereal |f x|)) else ∞)
   by (rule extend-quasinorm[OF ∗])
  show eNorm (𝔏 ∞ M) f = (if f ∈ borel-measurable M then e2ennreal(esssup M
(λx. |f x|)) else ∞)
       defect (𝔏 ∞ M) = 1
   unfolding Lp-space-def using quasinorm-of[OF ∗∗] by auto
qed

lemma L-infinity-space:
  space_N (𝔏 ∞ M) = {f ∈ borel-measurable M. ∃ C. AE x in M. |f x| ≤ C}
proof (auto simp del: infinity-ennreal-def)
  fix f assume H: f ∈ space_N (𝔏 ∞ M)
  then show f ∈ borel-measurable M
   unfolding space_N-def using L-infinity(1)[of M] top.not-eq-extremum by fast-
force
  then have ∗: esssup M (λx. |f x|) < ∞
   using H unfolding space_N-def L-infinity(1)[of M] by (auto simp add: e2en-
nreal-infty)
  define C where C = real-of-ereal(esssup M (λx. |f x|))
```

```
  have AE x in M. ereal |f x| ≤ ereal C
  proof (cases emeasure M (space M) = 0)
    case True
    then show ?thesis using emeasure-0-AE by simp
  next
    case False
    then have esssup M (λx. |f x|) ≥ 0
     using esssup-mono[of λx. 0 M (λx. |f x|)] esssup-const[OF False, of 0::ereal]
by auto
   then have esssup M (λx. |f x|) = ereal C unfolding C-def using ∗ ereal-real
by auto
   then show ?thesis using esssup-AE[of (λx. ereal |f x|) M] by simp
  qed
  then have AE x in M. |f x| ≤ C by auto
  then show ∃ C. AE x in M. |f x| ≤ C by blast
next
  fix f::'a ⇒ real and C::real
  assume H: f ∈ borel-measurable M AE x in M. |f x| ≤ C
  then have esssup M (λx. |f x|) ≤ C using esssup-I by auto
  then have eNorm (𝔏 ∞ M) f ≤ C unfolding L-infinity(1) using H(1)
    by auto (metis e2ennreal-ereal e2ennreal-mono)
  then show f ∈ space_N (𝔏 ∞ M)
    using spaceN-iff le-less-trans by fastforce
qed

lemma L-infinity-zero-space:
  zero-space_N (𝔏 ∞ M) = {f ∈ borel-measurable M. AE x in M. f x = 0}
proof (auto simp del: infinity-ennreal-def)
  fix f assume H: f ∈ zero-space_N (𝔏 ∞ M)
  then show f ∈ borel-measurable M
    unfolding zero-space_N-def using L-infinity(1)[of M] top.not-eq-extremum by
fastforce
  then have ∗: e2ennreal(esssup M (λx. |f x|)) = 0
    using H unfolding zero-space_N-def using L-infinity(1)[of M] e2ennreal-infty
by auto
  then have esssup M (λx. |f x|) ≤ 0
     by (metis e2ennreal-infty e2ennreal-mult ennreal-top-neq-zero ereal-mult-infty
leI linear mult-zero-left)
  then have f x = 0 if ereal |f x| ≤ esssup M (λx. |f x|) for x
    using that order.trans by fastforce
  then show AE x in M. f x = 0 using esssup-AE[of λx. ereal |f x| M] by auto
next
  fix f::'a ⇒ real
  assume H: f ∈ borel-measurable M AE x in M. f x = 0
  then have esssup M (λx. |f x|) ≤ 0 using esssup-I by auto
  then have eNorm (𝔏 ∞ M) f = 0 unfolding L-infinity(1) using H(1)
    by (simp add: e2ennreal-neg)
  then show f ∈ zero-space_N (𝔏 ∞ M)
    using zero-spaceN-iff by auto
```

qed

lemma *L-infinity-AE-ebound*:
AE x in M. ennreal |f x| ≤ eNorm (𝔏 ∞ M) f
proof (*cases f ∈ borel-measurable M*)
case *False*
then have *eNorm (𝔏 ∞ M) f = ∞*
unfolding *L-infinity(1)* **by** *auto*
then show *?thesis* **by** *simp*
next
case *True*
then have *ennreal |f x| ≤ eNorm (𝔏 ∞ M) f* **if** *|f x| ≤ esssup M (λx. |f x|)* **for** *x*
unfolding *L-infinity(1)* **using** *that e2ennreal-mono*
by *fastforce*
then show *?thesis* **using** *esssup-AE*[*of λx. ereal |f x|*] **by** *force*
qed

lemma *L-infinity-AE-bound*:
assumes *f ∈ space_N (𝔏 ∞ M)*
shows *AE x in M. |f x| ≤ Norm (𝔏 ∞ M) f*
using *L-infinity-AE-ebound*[*of f M*] **unfolding** *eNorm-Norm*[*OF assms*] **by** (*simp*)

In the next lemma, the assumption $C \geq 0$ that might seem useless is in fact necessary for the second statement when the space has zero measure. Indeed, any function is then almost surely bounded by any constant!

lemma *L-infinity-I*:
assumes *f ∈ borel-measurable M*
AE x in M. |f x| ≤ C
C ≥ 0
shows *f ∈ space_N (𝔏 ∞ M)*
Norm (𝔏 ∞ M) f ≤ C
proof −
show *f ∈ space_N (𝔏 ∞ M)*
using *L-infinity-space assms(1) assms(2)* **by** *force*
have *esssup M (λx. |f x|) ≤ C* **using** *assms(1) assms(2) esssup-I* **by** *auto*
then have *eNorm (𝔏 ∞ M) f ≤ ereal C*
unfolding *L-infinity(1)* **using** *assms(1) e2ennreal-mono* **by** *force*
then have *ennreal (Norm (𝔏 ∞ M) f) ≤ ennreal C*
using *eNorm-Norm*[*OF ‹f ∈ space_N (𝔏 ∞ M)›*] *assms(3)* **by** *auto*
then show *Norm (𝔏 ∞ M) f ≤ C* **using** *assms(3)* **by** *auto*
qed

lemma *L-infinity-I′*:
assumes [*measurable*]: *f ∈ borel-measurable M*
and *AE x in M. ennreal |f x| ≤ C*
shows *eNorm (𝔏 ∞ M) f ≤ C*
proof −
have *esssup M (λx. |f x|) ≤ enn2ereal C*

```
    apply (rule esssup-I, auto) using assms(2) less-eq-ennreal.rep-eq by auto
  then show ?thesis unfolding L-infinity using assms apply auto
    using e2ennreal-mono by fastforce
qed

lemma L-infinity-pos-measure:
  assumes [measurable]: f ∈ borel-measurable M
      and eNorm (𝔏 ∞ M) f > (C::real)
  shows emeasure M {x ∈ space M. |f x| > C} > 0
proof −
  have ∗: esssup M (λx. ereal(|f x|)) > ereal C using ‹eNorm (𝔏 ∞ M) f > C›
unfolding L-infinity
  proof (auto)
    assume a1: ennreal C < e2ennreal (esssup M (λx. ereal |f x|))
    have ¬ e2ennreal (esssup M (λa. ereal |f a|)) ≤ e2ennreal (ereal C) if ¬ C <
0
      using a1 that by (metis (no-types) e2ennreal-enn2ereal enn2ereal-ennreal leD
leI)
    then have e2ennreal (esssup M (λa. ereal |f a|)) ≤ e2ennreal (ereal C) ⟶
(∃ e≤esssup M (λa. ereal |f a|). ereal C < e)
      using a1 e2ennreal-neg by fastforce
    then show ?thesis
      by (meson e2ennreal-mono leI less-le-trans)
  qed
  have emeasure M {x ∈ space M. ereal(|f x|) > C} > 0
    by (rule esssup-pos-measure[OF - ∗], auto)
  then show ?thesis by auto
qed

lemma L-infinity-tendsto-AE:
  assumes tendsto-in_N (𝔏 ∞ M) f g
        ⋀n. f n ∈ space_N (𝔏 ∞ M)
        g ∈ space_N (𝔏 ∞ M)
  shows AE x in M. (λn. f n x) ⟶ g x
proof −
  have ∗: AE x in M. |(f n − g) x| ≤ Norm (𝔏 ∞ M) (f n − g) for n
    apply (rule L-infinity-AE-bound) using assms spaceN-diff by blast
  have AE x in M. ∀ n. |(f n − g) x| ≤ Norm (𝔏 ∞ M) (f n − g)
    apply (subst AE-all-countable) using ∗ by auto
  moreover have (λn. f n x) ⟶ g x if ∀ n. |(f n − g) x| ≤ Norm (𝔏 ∞ M)
(f n − g) for x
  proof −
    have (λn. |(f n − g) x|) ⟶ 0
      apply (rule tendsto-sandwich[of λn. 0 - - λn. Norm (𝔏 ∞ M) (f n − g)])
      using that ‹tendsto-in_N (𝔏 ∞ M) f g› unfolding tendsto-in_N-def by auto
    then have (λn. |f n x − g x|) ⟶ 0 by auto
    then show ?thesis
      by (simp add: ‹(λn. |f n x − g x|) ⟶ 0› LIM-zero-cancel tendsto-rabs-zero-cancel)
  qed
```

ultimately show *?thesis* **by** *auto*
qed

As an illustration of the mechanism of spaces inclusion, let us show that bounded continuous functions belong to L^∞.

lemma *bcontfun-subset-L-infinity*:
assumes *sets M = sets borel*
shows $space_N\ bcontfun_N \subseteq space_N\ (\mathfrak{L} \infty M)$
$\bigwedge f.\ f \in space_N\ bcontfun_N \Longrightarrow Norm\ (\mathfrak{L} \infty M)\ f \leq Norm\ bcontfun_N\ f$
$\bigwedge f.\ eNorm\ (\mathfrak{L} \infty M)\ f \leq eNorm\ bcontfun_N\ f$
$bcontfun_N \subseteq_N \mathfrak{L} \infty M$
proof −
have ∗: $f \in space_N\ (\mathfrak{L} \infty M) \wedge Norm\ (\mathfrak{L} \infty M)\ f \leq Norm\ bcontfun_N\ f$ **if** $f \in space_N\ bcontfun_N$ **for** *f*
proof −
have *H*: *continuous-on UNIV f* $\bigwedge x.\ abs(f\ x) \leq Norm\ bcontfun_N\ f$
using $bcontfun_N D[OF\ \langle f \in space_N\ bcontfun_N \rangle]$ **by** *auto*
then have $f \in$ *borel-measurable borel* **using** *borel-measurable-continuous-onI*
by *simp*
then have $f \in$ *borel-measurable M* **using** *assms* **by** *auto*
have ∗: *AE x in M*. $|f\ x| \leq Norm\ bcontfun_N\ f$ **using** *H*(2) **by** *auto*
show *?thesis* **using** *L-infinity-I*[*OF* ‹$f \in$ *borel-measurable M*› ∗ *Norm-nonneg*]
by *auto*
qed
show $space_N\ bcontfun_N \subseteq space_N\ (\mathfrak{L} \infty M)$
$\bigwedge f.\ f \in space_N\ bcontfun_N \Longrightarrow Norm\ (\mathfrak{L} \infty M)\ f \leq Norm\ bcontfun_N\ f$
using ∗ **by** *auto*
show ∗∗: $bcontfun_N \subseteq_N \mathfrak{L} \infty M$
apply (*rule quasinorm-subsetI'*[*of - - 1*]) **using** ∗ **by** *auto*
have $eNorm\ (\mathfrak{L} \infty M)\ f \leq ennreal\ 1 * eNorm\ bcontfun_N\ f$ **for** *f*
apply (*rule quasinorm-subset-Norm-eNorm*) **using** ∗ ∗∗ **by** *auto*
then show $eNorm\ (\mathfrak{L} \infty M)\ f \leq eNorm\ bcontfun_N\ f$ **for** *f* **by** *simp*
qed

5.2 L^p for $0 < p < \infty$

lemma *Lp*:
assumes $p \geq (1::real)$
shows $eNorm\ (\mathfrak{L}\ p\ M)\ f = (if\ (f \in borel\text{-}measurable\ M \wedge integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p))$
$then\ (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$else\ (\infty::ennreal))$
$defect\ (\mathfrak{L}\ p\ M) = 1$
proof −
define *F* **where** $F = \{f \in borel\text{-}measurable\ M.\ integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)\}$
have ∗: *quasinorm-on F 1* $(\lambda(f::'a \Rightarrow real).\ (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p))$
proof (*rule quasinorm-onI*)
fix *f g* **assume** $f \in F\ g \in F$
then show $f + g \in F$

unfolding *F-def plus-fun-def* **apply** (*auto*) **by** (*rule Minkowski-inequality*(*1*), *auto simp add*: ‹*p* ≥ *1*›)
show *ennreal* ((∫ *x*. |(*f* + *g*) *x*| *powr p* ∂*M*) *powr* (*1*/*p*))
≤ *ennreal 1* ∗ (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*) + *ennreal 1* ∗ (∫ *x*. |*g x*| *powr p* ∂*M*) *powr* (*1*/*p*)
apply (*auto*, *subst ennreal-plus*[*symmetric*], *simp*, *simp*, *rule ennreal-leI*)
unfolding *plus-fun-def* **apply** (*rule Minkowski-inequality*(*2*)[*of p f M g*], *auto simp add*: ‹*p* ≥ *1*›)
using ‹*f* ∈ *F*› ‹*g* ∈ *F*› **unfolding** *F-def* **by** *auto*
next
fix *f* **and** *c*::*real* **assume** *f* ∈ *F*
show *c* ∗*R f* ∈ *F* **using** ‹*f* ∈ *F*› **unfolding** *scaleR-fun-def F-def* **by** (*auto simp add*: *abs-mult powr-mult*)
show (∫ *x*. |(*c* ∗*R f*) *x*| *powr p* ∂*M*) *powr* (*1*/*p*) ≤ *ennreal*(*abs*(*c*)) ∗ (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
apply (*rule eq-refl*, *subst ennreal-mult*[*symmetric*], *simp*, *simp*, *rule ennreal-cong*)
apply (*unfold scaleR-fun-def*, *simp add*: *abs-mult powr-mult powr-powr*) **using** ‹*p* ≥ *1*› **by** *auto*
next
show *0* ∈ *F* **unfolding** *zero-fun-def F-def* **by** *auto*
qed (*auto*)

have *p* ≥ *0* **using** ‹*p* ≥ *1*› **by** *auto*
have ∗∗: 𝔏 *p M* = *quasinorm-of* (*1*,
(λ*f*. *if* (*f* ∈ *borel-measurable M* ∧ *integrable M* (λ*x*. |*f x*| *powr p*))
then (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
else (∞::*ennreal*)))
unfolding *Lp-space-def* **using** *enn2real-ennreal*[*OF* ‹*p* ≥ *0*›] ‹*p* ≥ *1*› **apply** *auto*
using *enn2real-ennreal*[*OF* ‹*p* ≥ *0*›] **by** *presburger*
show *eNorm* (𝔏 *p M*) *f* = (*if* (*f* ∈ *borel-measurable M* ∧ *integrable M* (λ*x*. |*f x*| *powr p*))
then (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
else (∞::*ennreal*))
defect (𝔏 *p M*) = *1*
unfolding ∗∗ **using** *quasinorm-of*[*OF extend-quasinorm*[*OF* ∗]] **unfolding** *F-def* **by** *auto*
qed

lemma *Lp-le-1*:
assumes *p* > *0* *p* ≤ (*1*::*real*)
shows *eNorm* (𝔏 *p M*) *f* = (*if* (*f* ∈ *borel-measurable M* ∧ *integrable M* (λ*x*. |*f x*| *powr p*))
then (∫ *x*. |*f x*| *powr p* ∂*M*) *powr* (*1*/*p*)
else (∞::*ennreal*))
defect (𝔏 *p M*) = *2 powr* (*1*/*p* − *1*)
proof −
define *F* **where** *F* = {*f* ∈ *borel-measurable M*. *integrable M* (λ*x*. |*f x*| *powr p*)}

have $*$: *quasinorm-on F (2 powr (1/p−1)) (λ(f::'a⇒real). (∫ x. |f x| powr p ∂M) powr (1/p))*
proof (*rule quasinorm-onI*)
fix *f g* **assume** $f \in F$ $g \in F$
then show $f + g \in F$
unfolding *F-def plus-fun-def* **apply** (*auto*) **by** (*rule Minkowski-inequality-le-1*(*1*), *auto simp add*: ‹*p > 0*› ‹*p ≤ 1*›)
show *ennreal* ((∫ *x*. |(*f* + *g*) *x*| *powr p ∂M*) *powr* (*1*/*p*))
≤ *ennreal* (*2 powr* (*1*/*p*−*1*)) ∗ (∫ *x*. |*f x*| *powr p ∂M*) *powr* (*1*/*p*) + *ennreal* (*2 powr* (*1*/*p*−*1*)) ∗ (∫ *x*. |*g x*| *powr p ∂M*) *powr* (*1*/*p*)
apply (*subst ennreal-mult*[*symmetric*], *auto*)+
apply (*subst ennreal-plus*[*symmetric*], *simp*, *simp*)
apply (*rule ennreal-leI*)
unfolding *plus-fun-def* **apply** (*rule Minkowski-inequality-le-1*(*2*)[*of p f M g*], *auto simp add*: ‹*p > 0*› ‹*p ≤ 1*›)
using ‹*f ∈ F*› ‹*g ∈ F*› **unfolding** *F-def* **by** *auto*
next
fix *f* **and** *c*::*real* **assume** $f \in F$
show $c *_R f \in F$ **using** ‹*f ∈ F*› **unfolding** *scaleR-fun-def F-def* **by** (*auto simp add*: *abs-mult powr-mult*)
show (∫ *x*. |(*c* ∗*R f*) *x*| *powr p ∂M*) *powr* (*1*/*p*) ≤ *ennreal*(*abs*(*c*)) ∗ (∫ *x*. |*f x*| *powr p ∂M*) *powr* (*1*/*p*)
apply (*rule eq-refl*, *subst ennreal-mult*[*symmetric*], *simp*, *simp*, *rule ennreal-cong*)
apply (*unfold scaleR-fun-def*, *simp add*: *abs-mult powr-mult powr-powr*) **using** ‹*p > 0*› **by** *auto*
next
show *0 ∈ F* **unfolding** *zero-fun-def F-def* **by** *auto*
show *1 ≤ 2 powr* (*1 / p − 1*) **using** ‹*p > 0*› ‹*p ≤ 1*› **by** (*auto simp add*: *ge-one-powr-ge-zero*)
qed (*auto*)

have *p ≥ 0* **using** ‹*p > 0*› **by** *auto*
have ∗∗: 𝔏 *p M* = *quasinorm-of* (*2 powr* (*1*/*p*−*1*),
(λ*f*. *if* (*f ∈ borel-measurable M ∧ integrable M* (λ*x*. |*f x*| *powr p*))
then (∫ *x*. |*f x*| *powr p ∂M*) *powr* (*1*/*p*)
else (∞::*ennreal*)))
unfolding *Lp-space-def* **using** ‹*p > 0*› ‹*p ≤ 1*› **using** *enn2real-ennreal*[*OF* ‹*p ≥ 0*›] **apply** *auto*
by (*insert enn2real-ennreal*[*OF* ‹*p ≥ 0*›], *presburger*)+
show *eNorm* (𝔏 *p M*) *f* = (*if* (*f ∈ borel-measurable M ∧ integrable M* (λ*x*. |*f x*| *powr p*))
then (∫ *x*. |*f x*| *powr p ∂M*) *powr* (*1*/*p*)
else (∞::*ennreal*))
defect (𝔏 *p M*) = *2 powr* (*1*/*p*−*1*)
unfolding ∗∗ **using** *quasinorm-of*[*OF extend-quasinorm*[*OF* ∗]] **unfolding** *F-def* **by** *auto*
qed

lemma *Lp-space*:
assumes $p > (0::real)$
shows $space_N\ (\mathfrak{L}\ p\ M) = \{f \in borel\text{-}measurable\ M.\ integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)\}$
apply (*auto simp add*: *spaceN-iff*)
using *Lp*(1) *Lp-le-1*(1) ‹$p > 0$› **apply** (*metis infinity-ennreal-def less-le not-less*)
using *Lp*(1) *Lp-le-1*(1) ‹$p > 0$› **apply** (*metis infinity-ennreal-def less-le not-less*)
using *Lp*(1) *Lp-le-1*(1) ‹$p > 0$› **by** (*metis ennreal-neq-top linear top.not-eq-extremum*)

lemma *Lp-I*:
assumes $p > (0::real)$
$f \in borel\text{-}measurable\ M\ integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)$
shows $f \in space_N\ (\mathfrak{L}\ p\ M)$
$Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
proof −
have ∗: $eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
by (*cases* $p \leq 1$, *insert assms*, *auto simp add*: *Lp-le-1*(1) *Lp*(1))
then show ∗∗: $f \in space_N\ (\mathfrak{L}\ p\ M)$ **unfolding** $space_N$*-def* **by** *auto*
show $Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$ **using** ∗ **unfolding** *Norm-def* **by** *auto*
then show $eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$ **using** *eNorm-Norm*[*OF* ∗∗] **by** *auto*
qed

lemma *Lp-D*:
assumes $p>0\ f \in space_N\ (\mathfrak{L}\ p\ M)$
shows $f \in borel\text{-}measurable\ M$
$integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)$
$Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
proof −
show ∗: $f \in borel\text{-}measurable\ M$
$integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)$
using *Lp-space*[*OF* ‹$p > 0$›] *assms*(2) **by** *auto*
then show $Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
using *Lp-I*[*OF* ‹$p > 0$›] **by** *auto*
qed

lemma *Lp-Norm*:
assumes $p > (0::real)$
$f \in borel\text{-}measurable\ M$
shows $Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
$(Norm\ (\mathfrak{L}\ p\ M)\ f)\ powr\ p = (\int x.\ |f\ x|\ powr\ p\ \partial M)$
proof −
show ∗: $Norm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
proof (*cases integrable M* $(\lambda x.\ |f\ x|\ powr\ p)$)
case *True*

then show *?thesis* **using** *Lp-I*[*OF assms True*] **by** *auto*
next
case *False*
then have $f \notin space_N\ (\mathfrak{L}\ p\ M)$ **using** *Lp-space*[*OF* ‹$p > 0$›, *of M*] **by** *auto*
then have $*$: $Norm\ (\mathfrak{L}\ p\ M)\ f = 0$ **using** *eNorm-Norm'* **by** *auto*
have $(\int x.\ |f\ x|\ powr\ p\ \partial M) = 0$ **using** *False* **by** (*simp add*: *not-integrable-integral-eq*)
then have $(\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p) = 0$ **by** *auto*
then show *?thesis* **using** $*$ **by** *auto*
qed
then show $(Norm\ (\mathfrak{L}\ p\ M)\ f)\ powr\ p = (\int x.\ |f\ x|\ powr\ p\ \partial M)$
unfolding $*$ **using** *powr-powr* ‹$p > 0$› **by** *auto*
qed

lemma *Lp-zero-space*:
assumes $p > (0::real)$
shows $zero\text{-}space_N\ (\mathfrak{L}\ p\ M) = \{f \in borel\text{-}measurable\ M.\ AE\ x\ in\ M.\ f\ x = 0\}$
proof (*auto*)
fix f **assume** H: $f \in zero\text{-}space_N\ (\mathfrak{L}\ p\ M)$
then have $*$: $f \in \{f \in borel\text{-}measurable\ M.\ integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)\}$
using *Lp-space*[*OF assms*] *zero-spaceN-subset-spaceN* **by** *auto*
then show $f \in borel\text{-}measurable\ M$ **by** *auto*
have $eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
by (*cases* $p \leq 1$, *insert* $*$ ‹$p > 0$›, *auto simp add*: *Lp-le-1*(*1*) *Lp*(*1*))
then have $(\int x.\ |f\ x|\ powr\ p\ \partial M) = 0$ **using** H **unfolding** $zero\text{-}space_N\text{-}def$ **by** *auto*
then have $AE\ x\ in\ M.\ |f\ x|\ powr\ p = 0$
by (*subst integral-nonneg-eq-0-iff-AE*[*symmetric*], *insert* $*$, *auto*)
then show $AE\ x\ in\ M.\ f\ x = 0$ **by** *auto*
next
fix $f::'a \Rightarrow real$
assume H [*measurable*]: $f \in borel\text{-}measurable\ M\ AE\ x\ in\ M.\ f\ x = 0$
then have $*$: $AE\ x\ in\ M.\ |f\ x|\ powr\ p = 0$ **by** *auto*
have $integrable\ M\ (\lambda x.\ |f\ x|\ powr\ p)$
using *integrable-cong-AE*[*OF - - *] **by** *auto*
have $**$: $(\int x.\ |f\ x|\ powr\ p\ \partial M) = 0$
using *integral-cong-AE*[*OF - - *] **by** *auto*
have $eNorm\ (\mathfrak{L}\ p\ M)\ f = (\int x.\ |f\ x|\ powr\ p\ \partial M)\ powr\ (1/p)$
by (*cases* $p \leq 1$, *insert* $H(1)$ ‹*integrable M* $(\lambda x.\ |f\ x|\ powr\ p)$› ‹$p > 0$›, *auto simp add*: *Lp-le-1*(*1*) *Lp*(*1*))
then have $eNorm\ (\mathfrak{L}\ p\ M)\ f = 0$ **using** $**$ **by** *simp*
then show $f \in zero\text{-}space_N\ (\mathfrak{L}\ p\ M)$
using *zero-spaceN-iff* **by** *auto*
qed

lemma *Lp-tendsto-AE-subseq*:
assumes $p>(0::real)$
$tendsto\text{-}in_N\ (\mathfrak{L}\ p\ M)\ f\ g$
$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$
$g \in space_N\ (\mathfrak{L}\ p\ M)$

shows $\exists r.$ *strict-mono* $r \wedge$ (*AE* x *in* $M.$ $(\lambda n.\ f\ (r\ n)\ x) \longrightarrow g\ x$)
proof −
have $f\ n - g \in space_N\ (\mathfrak{L}\ p\ M)$ **for** n
using *spaceN-diff*[*OF* ‹$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$› ‹$g \in space_N\ (\mathfrak{L}\ p\ M)$›] **by** *simp*
have *int*: *integrable* M $(\lambda x.\ |f\ n\ x - g\ x|\ powr\ p)$ **for** n
using *Lp-D*(*2*)[*OF* ‹$p > 0$› ‹$f\ n - g \in space_N\ (\mathfrak{L}\ p\ M)$›] **by** *auto*

have $(\lambda n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n - g)) \longrightarrow 0$
using ‹*tendsto-in*$_N$ $(\mathfrak{L}\ p\ M)\ f\ g$› **unfolding** *tendsto-in*$_N$*-def* **by** *auto*
then have $*$: $(\lambda n.\ (\int x.\ |f\ n\ x - g\ x|\ powr\ p\ \partial M)\ powr\ (1/p)) \longrightarrow 0$
using *Lp-D*(*3*)[*OF* ‹$p > 0$› ‹$\bigwedge n.\ f\ n - g \in space_N\ (\mathfrak{L}\ p\ M)$›] **by** *auto*
have $(\lambda n.\ ((\int x.\ |f\ n\ x - g\ x|\ powr\ p\ \partial M)\ powr\ (1/p))\ powr\ p) \longrightarrow 0$
apply (*rule tendsto-zero-powrI*[*of* - - - p]) **using** ‹$p > 0$› $*$ **by** *auto*
then have $**$: $(\lambda n.\ (\int x.\ |f\ n\ x - g\ x|\ powr\ p\ \partial M)) \longrightarrow 0$
using *powr-powr* ‹$p > 0$› **by** *auto*
have $\exists r.$ *strict-mono* $r \wedge$ (*AE* x *in* $M.$ $(\lambda n.\ |f\ (r\ n)\ x - g\ x|\ powr\ p) \longrightarrow 0$)
apply (*rule tendsto-L1-AE-subseq*) **using** *int* $**$ **by** *auto*
then obtain r **where** *strict-mono* r *AE* x *in* $M.$ $(\lambda n.\ |f\ (r\ n)\ x - g\ x|\ powr\ p) \longrightarrow 0$
by *blast*
moreover have $(\lambda n.\ f\ (r\ n)\ x) \longrightarrow g\ x$ **if** $(\lambda n.\ |f\ (r\ n)\ x - g\ x|\ powr\ p) \longrightarrow 0$ **for** x
proof −
have $(\lambda n.\ (|f\ (r\ n)\ x - g\ x|\ powr\ p)\ powr\ (1/p)) \longrightarrow 0$
apply (*rule tendsto-zero-powrI*[*of* - - - *1*/p]) **using** ‹$p > 0$› *that* **by** *auto*
then have $(\lambda n.\ |f\ (r\ n)\ x - g\ x|) \longrightarrow 0$
using *powr-powr* ‹$p > 0$› **by** *auto*
show *?thesis*
by (*simp add*: ‹$(\lambda n.\ |f\ (r\ n)\ x - g\ x|) \longrightarrow 0$› *Limits.LIM-zero-cancel tendsto-rabs-zero-cancel*)
qed
ultimately have *AE* x *in* $M.$ $(\lambda n.\ f\ (r\ n)\ x) \longrightarrow g\ x$ **by** *auto*
then show *?thesis* **using** ‹*strict-mono* r› **by** *auto*
qed

5.3 Specialization to L^1

lemma *L1-space*:
$space_N\ (\mathfrak{L}\ 1\ M) = \{f.\ integrable\ M\ f\}$
unfolding *one-ereal-def* **using** *Lp-space*[*of 1 M*] *integrable-abs-iff* **by** *auto*

lemma *L1-I*:
assumes *integrable* $M\ f$
shows $f \in space_N\ (\mathfrak{L}\ 1\ M)$
$Norm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
$eNorm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
unfolding *one-ereal-def* **using** *Lp-I*[*of 1*, *OF - borel-measurable-integrable*[*OF assms*]] *assms powr-to-1* **by** *auto*

lemma *L1-D*:
assumes $f \in space_N\ (\mathfrak{L}\ 1\ M)$
shows $f \in$ *borel-measurable M*
integrable M f
$Norm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
$eNorm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
using *assms* **by** (*auto simp add*: *L1-space L1-I*)

lemma *L1-int-ineq*:
$abs(\int x.\ f\ x\ \partial M) \leq Norm\ (\mathfrak{L}\ 1\ M)\ f$
proof (*cases integrable M f*)
case *True*
then show *?thesis* **using** *L1-I*(*2*)[*OF True*] **by** *auto*
next
case *False*
then have $(\int x.\ f\ x\ \partial M) = 0$ **by** (*simp add*: *not-integrable-integral-eq*)
then show *?thesis* **using** *Norm-nonneg* **by** *auto*
qed

In L^1, one can give a direct formula for the eNorm of a measurable function, using a nonnegative integral. The same formula holds in L^p for $p > 0$, with additional powers p and $1/p$, but one can not write it down since `powr` is not defined on `ennreal`.

lemma *L1-Norm*:
assumes [*measurable*]: $f \in$ *borel-measurable M*
shows $Norm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
$eNorm\ (\mathfrak{L}\ 1\ M)\ f = (\int^+ x.\ |f\ x|\ \partial M)$
proof −
show ∗: $Norm\ (\mathfrak{L}\ 1\ M)\ f = (\int x.\ |f\ x|\ \partial M)$
using *Lp-Norm*[*of 1*, *OF - assms*] **unfolding** *one-ereal-def* **by** *auto*
show $eNorm\ (\mathfrak{L}\ 1\ M)\ f = (\int^+ x.\ |f\ x|\ \partial M)$
proof (*cases integrable M f*)
case *True*
then have $f \in space_N\ (\mathfrak{L}\ 1\ M)$ **using** *L1-space* **by** *auto*
then have $eNorm\ (\mathfrak{L}\ 1\ M)\ f = ennreal\ (Norm\ (\mathfrak{L}\ 1\ M)\ f)$
using *eNorm-Norm* **by** *auto*
then show *?thesis*
by (*metis* (*mono-tags*) ∗ *AE-I2 True abs-ge-zero integrable-abs nn-integral-eq-integral*)
next
case *False*
then have $eNorm\ (\mathfrak{L}\ 1\ M)\ f = \infty$ **using** *L1-space* $space_N$*-def*
by (*metis ennreal-add-eq-top infinity-ennreal-def le-iff-add le-less-linear mem-Collect-eq*)
moreover have $(\int^+ x.\ |f\ x|\ \partial M) = \infty$
apply (*rule nn-integral-nonneg-infinite*) **using** *False* **by** (*auto simp add*: *integrable-abs-iff*)
ultimately show *?thesis* **by** *simp*
qed
qed

lemma *L1-indicator*:
assumes [*measurable*]: $A \in$ *sets M*
shows *eNorm* ($\mathfrak{L}$ *1 M*) (*indicator A*) = *emeasure M A*
by (*subst L1-Norm*, *auto*, *metis assms ennreal-indicator nn-integral-cong nn-integral-indicator*)

lemma *L1-indicator'*:
assumes [*measurable*]: $A \in$ *sets M*
and *emeasure M A* $\neq \infty$
shows *indicator A* $\in$ *space*$_N$ ($\mathfrak{L}$ *1 M*)
Norm ($\mathfrak{L}$ *1 M*) (*indicator A*) = *measure M A*
unfolding *space*$_N$*-def Norm-def* **using** *L1-indicator*[*OF* ‹$A \in$ *sets M*›] ‹*emeasure M A* $\neq \infty$›
by (*auto simp add*: *top.not-eq-extremum Sigma-Algebra.measure-def*)

5.4 L^0

We have defined L^p for all exponents p, although it does not really make sense for $p = 0$. We have chosen a definition in this case (the space of all measurable functions) so that many statements are true for all exponents. In this paragraph, we show the consistency of this definition.

lemma *L-zero*:
eNorm ($\mathfrak{L}$ *0 M*) *f* = (*if f* $\in$ *borel-measurable M then 0 else* ∞)
defect ($\mathfrak{L}$ *0 M*) = *1*
proof −
have $*$: *quasinorm-on UNIV 1* (λ(*f*::'*a*$\Rightarrow$*real*). (*if f* $\in$ *borel-measurable M then 0 else* ∞))
by (*rule extend-quasinorm*, *rule quasinorm-onI*, *auto*)
show *eNorm* ($\mathfrak{L}$ *0 M*) *f* = (*if f* $\in$ *borel-measurable M then 0 else* ∞)
defect ($\mathfrak{L}$ *0 M*) = *1*
using *quasinorm-of*[*OF* $*$] **unfolding** *Lp-space-def* **by** *auto*
qed

lemma *L-zero-space* [*simp*]:
space$_N$ ($\mathfrak{L}$ *0 M*) = *borel-measurable M*
zero-space$_N$ ($\mathfrak{L}$ *0 M*) = *borel-measurable M*
apply (*auto simp add*: *spaceN-iff zero-spaceN-iff L-zero*(*1*))
using *top.not-eq-extremum* **by** *force*+

5.5 Basic results on L^p for general p

lemma *Lp-measurable-subset*:
space$_N$ ($\mathfrak{L}$ *p M*) $\subseteq$ *borel-measurable M*
proof (*cases rule*: *Lp-cases*[*of p*])
case *zero*
then show *?thesis* **using** *L-zero-space* **by** *auto*
next
case (*real-pos p2*)

```
  then show ?thesis using Lp-space[OF ‹p2 > 0›] by auto
next
  case PInf
  then show ?thesis using L-infinity-space by auto
qed

lemma Lp-measurable:
  assumes f ∈ space_N (𝔏 p M)
  shows f ∈ borel-measurable M
using assms Lp-measurable-subset by auto

lemma Lp-infinity-zero-space:
  assumes p > (0::ennreal)
  shows zero-space_N (𝔏 p M) = {f ∈ borel-measurable M. AE x in M. f x = 0}
proof (cases rule: Lp-cases[of p])
  case PInf
  then show ?thesis using L-infinity-zero-space by auto
next
  case (real-pos p2)
  then show ?thesis using Lp-zero-space[OF ‹p2 > 0›] unfolding ‹p = ennreal
p2› by auto
next
  case zero
  then have False using assms by auto
  then show ?thesis by simp
qed

lemma (in prob-space) Lp-subset-Lq:
  assumes p ≤ q
  shows ⋀f. eNorm (𝔏 p M) f ≤ eNorm (𝔏 q M) f
        𝔏 q M ⊆_N 𝔏 p M
        space_N (𝔏 q M) ⊆ space_N (𝔏 p M)
        ⋀f. f ∈ space_N (𝔏 q M) ⟹ Norm (𝔏 p M) f ≤ Norm (𝔏 q M) f
proof −
  show eNorm (𝔏 p M) f ≤ eNorm (𝔏 q M) f for f
  proof (cases eNorm (𝔏 q M) f < ∞)
    case True
    then have f ∈ space_N (𝔏 q M) using spaceN-iff by auto
    then have f-meas [measurable]: f ∈ borel-measurable M using Lp-measurable
by auto
    consider p = 0 | p = q | p > 0 ∧ p < ∞ ∧ q = ∞ | p > 0 ∧ p < q ∧ q < ∞
      using ‹p ≤ q› apply (simp add: top.not-eq-extremum)
      using not-less-iff-gr-or-eq order.order-iff-strict by fastforce
    then show ?thesis
    proof (cases)
      case 1
      then show ?thesis by (simp add: L-zero(1))
    next
      case 2
```

then show *?thesis* **by** *auto*
next
case *3*
then have $q = \infty$ **by** *simp*
obtain *p2* **where** $p = ennreal\ p2\ p2 > 0$
using *3 enn2real-positive-iff*[*of p*] **by** (*cases p*) *auto*
have $*$: $AE\ x\ in\ M.\ |f\ x| \leq Norm\ (\mathfrak{L}\ \infty\ M)\ f$
using *L-infinity-AE-bound* ‹$f \in space_N\ (\mathfrak{L}\ q\ M)$› ‹$q = \infty$› **by** *auto*
have $**$: *integrable M* ($\lambda x.\ |f\ x|\ powr\ p2$)
apply (*rule Bochner-Integration.integrable-bound*[*of - λx. (Norm* ($\mathfrak{L}\ \infty\ M$) *f) powr p2*], *auto*)
using $*$ *powr-mono2* ‹*p2 > 0*› **by** *force*
then have *eNorm* ($\mathfrak{L}$ *p2 M*) *f* = ($\int x.\ |f\ x|\ powr\ p2\ \partial M$) *powr* (*1/p2*)
using *Lp-I*(*3*)[*OF* ‹*p2 > 0*› *f-meas*] **by** *simp*
also have ... ≤ ($\int x.$ (*Norm* ($\mathfrak{L}\ \infty\ M$) *f*) *powr p2* ∂M) *powr* (*1/p2*)
apply (*rule ennreal-leI*, *rule powr-mono2*, *simp add*: ‹*p2 > 0*› *less-imp-le*, *simp*)
apply (*rule integral-mono-AE*, *auto simp add*: $**$)
using $*$ *powr-mono2* ‹*p2 > 0*› **by** *force*
also have ... = *Norm* ($\mathfrak{L}\ \infty\ M$) *f*
using ‹*p2 > 0*› **by** (*auto simp add*: *prob-space powr-powr*)
finally show *?thesis*
using ‹$p = ennreal\ p2$› ‹$q = \infty$› *eNorm-Norm*[*OF* ‹$f \in space_N\ (\mathfrak{L}\ q\ M)$›]
by *auto*
next
case *4*
then have $0 < p\ p < \infty$ **by** *auto*
then obtain *p2* **where** $p = ennreal\ p2\ p2 > 0$
using *enn2real-positive-iff*[*of p*] **by** (*cases p*) *auto*
have $0 < q\ q < \infty$ **using** *4* **by** *auto*
then obtain *q2* **where** $q = ennreal\ q2\ q2 > 0$
using *enn2real-positive-iff*[*of q*] **by** (*cases q*) *auto*
have *p2 < q2* **using** *4* ‹$p = ennreal\ p2$› ‹$q = ennreal\ q2$›
using *ennreal-less-iff* **by** *auto*
define *r2* **where** *r2 = q2 / p2*
have *r2* ≥ *1* **unfolding** *r2-def* **using** ‹*p2 < q2*› ‹*p2 > 0*› **by** *auto*
have $*$: *abs* (|*z*| *powr p2*) *powr r2* = |*z*| *powr q2* **for** *z*::*real*
unfolding *r2-def* **using** ‹*p2 > 0*› **by** (*simp add*: *powr-powr*)
have *I*: *integrable M* ($\lambda x.$ *abs*(|*f x*| *powr p2*) *powr r2*)
unfolding $*$ **using** ‹$f \in space_N\ (\mathfrak{L}\ q\ M)$› ‹$q = ennreal\ q2$› *Lp-D*(*2*)[*OF* ‹*q2 > 0*›] **by** *auto*
have *J*: *integrable M* ($\lambda x.\ |f\ x|\ powr\ p2$)
by (*rule bound-L1-Lp*(*1*)[*OF* ‹*r2* ≥ *1*› *- I*], *auto*)
have $f \in space_N\ (\mathfrak{L}\ p2\ M)$
by (*rule Lp-I*(*1*)[*OF* ‹*p2 > 0*› *- J*], *simp*)
have ($\int x.\ |f\ x|\ powr\ p2\ \partial M$) *powr* (*1/p2*) = *abs*($\int x.\ |f\ x|\ powr\ p2\ \partial M$) *powr* (*1/p2*)
by *auto*
also have ... ≤ (($\int x.$ *abs* (|*f x*| *powr p2*) *powr r2* ∂M) *powr* (*1/r2*)) *powr*

```
(1/p2)
      apply (subst powr-mono2, simp add: ‹p2 > 0› less-imp-le, simp)
      apply (rule bound-L1-Lp, simp add: ‹r2 ≥ 1›, simp)
       unfolding ∗ using ‹f ∈ space_N (𝔏 q M)› ‹q = ennreal q2› Lp-D(2)[OF
‹q2 > 0›] by auto
     also have ... = (∫ x. |f x| powr q2 ∂M) powr (1/q2)
      unfolding ∗ using ‹p2 > 0› by (simp add: powr-powr r2-def)
     finally show ?thesis
      using ‹f ∈ space_N (𝔏 q M)› Lp-D(4)[OF ‹q2 > 0›] ennreal-leI
      unfolding ‹p = ennreal p2› ‹q = ennreal q2› Lp-D(4)[OF ‹p2 > 0› ‹f ∈
space_N (𝔏 p2 M)›] by force
   qed
  next
   case False
   then have eNorm (𝔏 q M) f = ∞
    using top.not-eq-extremum by fastforce
   then show ?thesis by auto
  qed
  then show 𝔏 q M ⊆_N 𝔏 p M using quasinorm-subsetI[of - - 1] by auto
  then show space_N (𝔏 q M) ⊆ space_N (𝔏 p M) using quasinorm-subset-space
by auto
  then show Norm (𝔏 p M) f ≤ Norm (𝔏 q M) f if f ∈ space_N (𝔏 q M) for f
   using eNorm-Norm that ‹eNorm (𝔏 p M) f ≤ eNorm (𝔏 q M) f› ennreal-le-iff
Norm-nonneg
   by (metis rev-subsetD)
qed

proposition Lp-domination:
  assumes [measurable]: g ∈ borel-measurable M
     and f ∈ space_N (𝔏 p M)
         AE x in M. |g x| ≤ |f x|
  shows g ∈ space_N (𝔏 p M)
       Norm (𝔏 p M) g ≤ Norm (𝔏 p M) f
proof −
  have [measurable]: f ∈ borel-measurable M using Lp-measurable[OF assms(2)]
by simp
  have g ∈ space_N (𝔏 p M) ∧ Norm (𝔏 p M) g ≤ Norm (𝔏 p M) f
  proof (cases rule: Lp-cases[of p])
   case zero
   then have Norm (𝔏 p M) g = 0
    unfolding Norm-def using L-zero(1)[of M] by auto
   then have Norm (𝔏 p M) g ≤ Norm (𝔏 p M) f using Norm-nonneg by auto
   then show ?thesis unfolding ‹p = 0› L-zero-space by auto
  next
   case (real-pos p2)
   have ∗: integrable M (λx. |f x| powr p2)
    using ‹f ∈ space_N (𝔏 p M)› unfolding ‹p = ennreal p2› using Lp-D(2) ‹p2
> 0› by auto
   have ∗∗: integrable M (λx. |g x| powr p2)
```

```
    apply (rule Bochner-Integration.integrable-bound[of - λx. |f x| powr p2]) using
∗ apply auto
      using assms(3) powr-mono2 ‹p2 > 0› by (auto simp add: less-imp-le)
    then have g ∈ space_N (𝔏 p M)
      unfolding ‹p = ennreal p2› using Lp-space[OF ‹p2 > 0›, of M] by auto
    have Norm (𝔏 p M) g = (∫ x. |g x| powr p2 ∂M) powr (1/p2)
      unfolding ‹p = ennreal p2› by (rule Lp-I(2)[OF ‹p2 > 0› - ∗∗], simp)
    also have ... ≤ (∫ x. |f x| powr p2 ∂M) powr (1/p2)
      apply (rule powr-mono2, simp add: ‹p2 > 0› less-imp-le, simp)
      apply (rule integral-mono-AE, auto simp add: ∗ ∗∗)
      using ‹p2 > 0› less-imp-le powr-mono2 assms(3) by auto
    also have ... = Norm (𝔏 p M) f
      unfolding ‹p = ennreal p2› by (rule Lp-I(2)[OF ‹p2 > 0› - ∗, symmetric],
simp)
    finally show ?thesis using ‹g ∈ space_N (𝔏 p M)› by auto
  next
    case PInf
    have AE x in M. |f x| ≤ Norm (𝔏 p M) f
      using ‹f ∈ space_N (𝔏 p M)› L-infinity-AE-bound unfolding ‹p = ∞› by
auto
    then have ∗: AE x in M. |g x| ≤ Norm (𝔏 p M) f
      using assms(3) by auto
    show ?thesis
      using L-infinity-I[OF assms(1) ∗] Norm-nonneg ‹p = ∞› by auto
  qed
  then show g ∈ space_N (𝔏 p M) Norm (𝔏 p M) g ≤ Norm (𝔏 p M) f
    by auto
qed

lemma Lp-Banach-lattice:
  assumes f ∈ space_N (𝔏 p M)
  shows (λx. |f x|) ∈ space_N (𝔏 p M)
        Norm (𝔏 p M) (λx. |f x|) = Norm (𝔏 p M) f
proof −
  have [measurable]: f ∈ borel-measurable M using Lp-measurable[OF assms] by
simp
  show (λx. |f x|) ∈ space_N (𝔏 p M)
    by (rule Lp-domination(1)[OF - assms], auto)
  have Norm (𝔏 p M) (λx. |f x|) ≤ Norm (𝔏 p M) f
    by (rule Lp-domination[OF - assms], auto)
  moreover have Norm (𝔏 p M) f ≤ Norm (𝔏 p M) (λx. |f x|)
    by (rule Lp-domination[OF - ‹(λx. |f x|) ∈ space_N (𝔏 p M)›], auto)
  ultimately show Norm (𝔏 p M) (λx. |f x|) = Norm (𝔏 p M) f
    by auto
qed

lemma Lp-bounded-bounded-support:
  assumes [measurable]: f ∈ borel-measurable M
      and AE x in M. |f x| ≤ C
```

emeasure M $\{x \in$ *space M*. $f\ x \neq 0\} \neq \infty$
shows $f \in$ *space*$_N(\mathfrak{L}\ p\ M)$
proof (*cases rule*: *Lp-cases*[*of p*])
case *zero*
then show *?thesis* **using** *L-zero-space assms* **by** *blast*
next
case *PInf*
then show *?thesis* **using** *L-infinity-space assms* **by** *blast*
next
case (*real-pos p2*)
have $*$: *integrable M* (λx. $|f\ x|$ *powr p2*)
apply (*rule integrableI-bounded-set*[*of* $\{x \in$ *space M*. $f\ x \neq 0\}$ - - *C powr p2*])
using *assms powr-mono2*[*OF less-imp-le*[*OF* ‹*p2* > *0*›]] **by** (*auto simp add*: *top.not-eq-extremum*)
show *?thesis*
unfolding ‹*p* = *ennreal p2*› **apply** (*rule Lp-I*[*OF* ‹*p2* > *0*›]) **using** $*$ **by** *auto*
qed

5.6 L^p versions of the main theorems in integration theory

The space L^p is stable under almost sure convergence, for sequence with bounded norm. This is a version of Fatou's lemma (and it indeed follows from this lemma in the only nontrivial situation where $p \in (0, +\infty)$.

proposition *Lp-AE-limit*:
assumes [*measurable*]: $g \in$ *borel-measurable M*
and *AE x in M*. (λn. *f n x*) $\longrightarrow$ *g x*
shows *eNorm* ($\mathfrak{L}$ *p M*) *g* $\leq$ *liminf* (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f n*))
proof (*cases liminf* (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f n*)) = ∞)
case *True*
then show *?thesis* **by** *auto*
next
case *False*
define *le* **where** *le* = *liminf* (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f n*))
then have *le* $< \infty$ **using** *False* **by** (*simp add*: *top.not-eq-extremum*)
obtain *r0* **where** *r0*: *strict-mono r0* (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f* (*r0 n*))) $\longrightarrow$ *le*
using *liminf-subseq-lim*[*of* λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f n*)]
unfolding *comp-def le-def*
by *blast*
then have *eventually* (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f* (*r0 n*)) $< \infty$) *sequentially*
using *False* **unfolding** *order-tendsto-iff le-def* **by** (*simp add*: *top.not-eq-extremum*)
then obtain *N* **where** *N*: $\bigwedge n.\ n \geq N \Longrightarrow$ *eNorm* ($\mathfrak{L}$ *p M*) (*f* (*r0 n*)) $< \infty$
unfolding *eventually-sequentially* **by** *blast*
define *r* **where** *r* = (λn. *r0* (*n* + *N*))
have *strict-mono r* **unfolding** *r-def* **using** ‹*strict-mono r0*›
by (*simp add*: *strict-mono-Suc-iff*)
have $*$: (λn. *eNorm* ($\mathfrak{L}$ *p M*) (*f* (*r n*))) $\longrightarrow$ *le*
unfolding *r-def* **using** *LIMSEQ-ignore-initial-segment*[*OF r0*(*2*), *of N*]**.**
have *f* (*r n*) $\in$ *space*$_N$ ($\mathfrak{L}$ *p M*) **for** *n*
using *N spaceN-iff* **unfolding** *r-def* **by** *force*

then have [*measurable*]: $f\ (r\ n) \in$ *borel-measurable* M **for** n
using *Lp-measurable* **by** *auto*
define l **where** $l =$ *enn2real le*
have $l \geq 0$ **unfolding** *l-def* **by** *auto*
have $le =$ *ennreal* l **using** ‹$le < \infty$› **unfolding** *l-def* **by** *auto*
have [*tendsto-intros*]: $(\lambda n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ (r\ n))) \longrightarrow l$
apply (*rule tendsto-ennrealD*)
using $*$ ‹$le < \infty$› **unfolding** *eNorm-Norm*[*OF* ‹$\bigwedge n.\ f\ (r\ n) \in space_N\ (\mathfrak{L}\ p\ M)$›] *l-def* **by** *auto*

show *?thesis*
proof (*cases rule*: *Lp-cases*[*of p*])
case *zero*
then have *eNorm* $(\mathfrak{L}\ p\ M)\ g = 0$
using *assms*(*1*) **by** (*simp add*: *L-zero*(*1*))
then show *?thesis* **by** *auto*
next
case (*real-pos p2*)
then have $f\ (r\ n) \in space_N\ (\mathfrak{L}\ p2\ M)$ **for** n
using ‹$\bigwedge n.\ f\ (r\ n) \in space_N(\mathfrak{L}\ p\ M)$› **by** *auto*
have *liminf* $(\lambda n.\ ennreal(|f\ (r\ n)\ x|\ powr\ p2)) = |g\ x|\ powr\ p2$ **if** $(\lambda n.\ f\ n\ x) \longrightarrow g\ x$ **for** x
apply (*rule lim-imp-Liminf*, *auto intro*!: *tendsto-intros simp add*: ‹$p2 > 0$›)
using *LIMSEQ-subseq-LIMSEQ*[*OF that* ‹*strict-mono r*›] **unfolding** *comp-def* **by** *auto*
then have $*$: *AE x in M. liminf* $(\lambda n.\ ennreal(|f\ (r\ n)\ x|\ powr\ p2)) = |g\ x|\ powr\ p2$
using ‹*AE x in M.* $(\lambda n.\ f\ n\ x) \longrightarrow g\ x$› **by** *auto*

have $(\int^+ x.\ ennreal(|f\ (r\ n)\ x|\ powr\ p2)\ \partial M) = ennreal((Norm\ (\mathfrak{L}\ p\ M)\ (f\ (r\ n)))\ powr\ p2)$ **for** n
proof −
have $(\int^+ x.\ ennreal(|f\ (r\ n)\ x|\ powr\ p2)\ \partial M) = ennreal\ (\int x.\ |f\ (r\ n)\ x|\ powr\ p2\ \partial M)$
by (*rule nn-integral-eq-integral*, *auto simp add*: *Lp-D*(*2*)[*OF* ‹$p2 > 0$› ‹$f\ (r\ n) \in space_N\ (\mathfrak{L}\ p2\ M)$›])
also have $... = ennreal((Norm\ (\mathfrak{L}\ p2\ M)\ (f\ (r\ n)))\ powr\ p2)$
unfolding *Lp-D*(*3*)[*OF* ‹$p2 > 0$› ‹$f\ (r\ n) \in space_N\ (\mathfrak{L}\ p2\ M)$›] **using** *powr-powr* ‹$p2 > 0$› **by** *auto*
finally show *?thesis* **using** ‹$p = ennreal\ p2$› **by** *simp*
qed
moreover have $(\lambda n.\ ennreal((Norm\ (\mathfrak{L}\ p\ M)\ (f\ (r\ n)))\ powr\ p2)) \longrightarrow ennreal(l\ powr\ p2)$
by (*auto intro*!:*tendsto-intros simp add*: ‹$p2 > 0$›)
ultimately have $**$: *liminf* $(\lambda n.\ (\int^+ x.\ ennreal(|f\ (r\ n)\ x|\ powr\ p2)\ \partial M)) = ennreal(l\ powr\ p2)$
using *lim-imp-Liminf* **by** *force*

have $(\int^+ x.\ |g\ x|\ powr\ p2\ \partial M) = (\int^+ x.\ liminf\ (\lambda n.\ ennreal(|f\ (r\ n)\ x|\ powr$

p2)) *∂M*)
apply (*rule nn-integral-cong-AE*) **using** ∗ **by** *auto*
also have ... ≤ *liminf* (*λn*. ∫⁺*x*. *ennreal*(|*f* (*r n*) *x*| *powr p2*) *∂M*)
by (*rule nn-integral-liminf*, *auto*)
finally have (∫⁺*x*. |*g x*| *powr p2 ∂M*) ≤ *ennreal*(*l powr p2*) **using** ∗∗ **by** *auto*
then have (∫⁺*x*. |*g x*| *powr p2 ∂M*) < ∞ **using** *le-less-trans* **by** *fastforce*
then have *intg*: *integrable M* (*λx*. |*g x*| *powr p2*)
apply (*intro integrableI-nonneg*) **by** *auto*
then have *g* ∈ *space*$_N$ (𝔏 *p2 M*) **using** *Lp-I*(*1*)[*OF* ‹*p2* > *0*›, *of* - *M*] **by** *fastforce*
have *ennreal*((*Norm* (𝔏 *p2 M*) *g*) *powr p2*) = *ennreal*(∫ *x*. |*g x*| *powr p2 ∂M*)
unfolding *Lp-D*(*3*)[*OF* ‹*p2* > *0*› ‹*g* ∈ *space*$_N$ (𝔏 *p2 M*)›] **using** *powr-powr* ‹*p2* > *0*› **by** *auto*
also have ... = (∫⁺*x*. |*g x*| *powr p2 ∂M*)
by (*rule nn-integral-eq-integral*[*symmetric*], *auto simp add*: *intg*)
finally have *ennreal*((*Norm* (𝔏 *p2 M*) *g*) *powr p2*) ≤ *ennreal*(*l powr p2*)
using ‹(∫⁺*x*. |*g x*| *powr p2 ∂M*) ≤ *ennreal*(*l powr p2*)› **by** *auto*
then have ((*Norm* (𝔏 *p2 M*) *g*) *powr p2*) *powr* (*1*/*p2*) ≤ (*l powr p2*) *powr* (*1*/*p2*)
using *ennreal-le-iff* ‹*l* ≥ *0*› ‹*p2* > *0*› *powr-mono2* **by** *auto*
then have *Norm* (𝔏 *p2 M*) *g* ≤ *l*
using ‹*p2* > *0*› ‹*l* ≥ *0*› **by** (*auto simp add*: *powr-powr*)
then have *eNorm* (𝔏 *p2 M*) *g* ≤ *le*
unfolding *eNorm-Norm*[*OF* ‹*g* ∈ *space*$_N$ (𝔏 *p2 M*)›] ‹*le* = *ennreal l*› **using** *ennreal-leI* **by** *auto*
then show *?thesis* **unfolding** *le-def* ‹*p* = *ennreal p2*› **by** *simp*
next
case *PInf*
then have *AE x in M*. ∀ *n*. |*f* (*r n*) *x*| ≤ *Norm* (𝔏 ∞ *M*) (*f* (*r n*))
apply (*subst AE-all-countable*) **using** *L-infinity-AE-bound* ‹⋀*n*. *f* (*r n*) ∈ *space*$_N$ (𝔏 *p M*)› **by** *blast*
moreover have |*g x*| ≤ *l* **if** ∀ *n*. |*f* (*r n*) *x*| ≤ *Norm* (𝔏 ∞ *M*) (*f* (*r n*)) (*λn*. *f n x*) ⟶ *g x* **for** *x*
proof −
have (*λn*. *f* (*r n*) *x*) ⟶ *g x*
using *that LIMSEQ-subseq-LIMSEQ*[*OF* - ‹*strict-mono r*›] **unfolding** *comp-def* **by** *auto*
then have ∗: (*λn*. |*f* (*r n*) *x*|) ⟶ |*g x*|
by (*auto intro*!:*tendsto-intros*)
show *?thesis*
apply (*rule LIMSEQ-le*[*OF* ∗]) **using** *that*(*1*) ‹(*λn*. *Norm* (𝔏 *p M*) (*f* (*r n*))) ⟶ *l*› **unfolding** *PInf* **by** *auto*
qed
ultimately have *AE x in M*. |*g x*| ≤ *l* **using** ‹*AE x in M*. (*λn*. *f n x*) ⟶ *g x*› **by** *auto*
then have *g* ∈ *space*$_N$ (𝔏 ∞ *M*) *Norm* (𝔏 ∞ *M*) *g* ≤ *l*
using *L-infinity-I*[*OF* ‹*g* ∈ *borel-measurable M*› - ‹*l* ≥ *0*›] **by** *auto*
then have *eNorm* (𝔏 ∞ *M*) *g* ≤ *le*
unfolding *eNorm-Norm*[*OF* ‹*g* ∈ *space*$_N$ (𝔏 ∞ *M*)›] ‹*le* = *ennreal l*› **using**

ennreal-leI **by** *auto*
then show *?thesis* **unfolding** *le-def* ‹$p = \infty$› **by** *simp*
qed
qed

lemma *Lp-AE-limit'*:
assumes $g \in$ *borel-measurable M*
$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$
AE x in M. $(\lambda n.\ f\ n\ x) \longrightarrow g\ x$
$(\lambda n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n)) \longrightarrow l$
shows $g \in space_N\ (\mathfrak{L}\ p\ M)$
$Norm\ (\mathfrak{L}\ p\ M)\ g \leq l$
proof −
have $l \geq 0$ **by** (*rule LIMSEQ-le-const*[*OF* ‹$(\lambda n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n)) \longrightarrow l$›], *auto*)
have $(\lambda n.\ eNorm\ (\mathfrak{L}\ p\ M)\ (f\ n)) \longrightarrow ennreal\ l$
unfolding *eNorm-Norm*[*OF* ‹$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$›] **using** ‹$(\lambda n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n)) \longrightarrow l$› **by** *auto*
then have ∗: $ennreal\ l = liminf\ (\lambda n.\ eNorm\ (\mathfrak{L}\ p\ M)\ (f\ n))$
using *lim-imp-Liminf*[*symmetric*] *trivial-limit-sequentially* **by** *blast*
have $eNorm\ (\mathfrak{L}\ p\ M)\ g \leq ennreal\ l$
unfolding ∗ **apply** (*rule Lp-AE-limit*) **using** *assms* **by** *auto*
then have $eNorm\ (\mathfrak{L}\ p\ M)\ g < \infty$ **using** *le-less-trans* **by** *fastforce*
then show $g \in space_N\ (\mathfrak{L}\ p\ M)$ **using** *spaceN-iff* **by** *auto*
show $Norm\ (\mathfrak{L}\ p\ M)\ g \leq l$
using ‹$eNorm\ (\mathfrak{L}\ p\ M)\ g \leq ennreal\ l$› *ennreal-le-iff*[*OF* ‹$l \geq 0$›]
unfolding *eNorm-Norm*[*OF* ‹$g \in space_N\ (\mathfrak{L}\ p\ M)$›] **by** *auto*
qed

lemma *Lp-AE-limit''*:
assumes $g \in$ *borel-measurable M*
$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$
AE x in M. $(\lambda n.\ f\ n\ x) \longrightarrow g\ x$
$\bigwedge n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n) \leq C$
shows $g \in space_N\ (\mathfrak{L}\ p\ M)$
$Norm\ (\mathfrak{L}\ p\ M)\ g \leq C$
proof −
have $C \geq 0$ **by** (*rule order-trans*[*OF Norm-nonneg*[*of* $\mathfrak{L}$ *p M f 0*] ‹$Norm\ (\mathfrak{L}\ p\ M)\ (f\ 0) \leq C$›])
have ∗: $liminf\ (\lambda n.\ ennreal\ C) = ennreal\ C$
using *Liminf-const trivial-limit-at-top-linorder* **by** *blast*
have $eNorm\ (\mathfrak{L}\ p\ M)\ (f\ n) \leq ennreal\ C$ **for** *n*
unfolding *eNorm-Norm*[*OF* ‹$f\ n \in space_N\ (\mathfrak{L}\ p\ M)$›]
using ‹$Norm\ (\mathfrak{L}\ p\ M)\ (f\ n) \leq C$› **by** (*auto simp add*: *ennreal-leI*)
then have $liminf\ (\lambda n.\ eNorm\ (\mathfrak{L}\ p\ M)\ (f\ n)) \leq ennreal\ C$
using *Liminf-mono*[*of* $(\lambda n.\ eNorm\ (\mathfrak{L}\ p\ M)\ (f\ n))$ *λ-. C sequentially*] ∗ **by** *auto*
then have $eNorm\ (\mathfrak{L}\ p\ M)\ g \leq ennreal\ C$ **using**
Lp-AE-limit[*OF* ‹$g \in$ *borel-measurable M*› ‹*AE x in M*. $(\lambda n.\ f\ n\ x) \longrightarrow g\ x$›, *of p*] **by** *auto*

then have *eNorm* ($\mathfrak{L}$ *p M*) *g* < ∞ **using** *le-less-trans* **by** *fastforce*
then show *g* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*) **using** *spaceN-iff* **by** *auto*
show *Norm* ($\mathfrak{L}$ *p M*) *g* ≤ *C*
using ‹*eNorm* ($\mathfrak{L}$ *p M*) *g* ≤ *ennreal C*› *ennreal-le-iff*[*OF* ‹*C* ≥ *0*›]
unfolding *eNorm-Norm*[*OF* ‹*g* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)›] **by** *auto*
qed

We give the version of Lebesgue dominated convergence theorem in the setting of L^p spaces.

proposition *Lp-domination-limit*:
fixes *p*::*real*
assumes [*measurable*]: *g* ∈ *borel-measurable M*
⋀*n*. *f n* ∈ *borel-measurable M*
and *m* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)
AE x in M. (λ*n*. *f n x*) ⟶ *g x*
⋀*n*. *AE x in M*. |*f n x*| ≤ *m x*
shows *g* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)
tendsto-in$_N$ ($\mathfrak{L}$ *p M*) *f g*
proof −
have [*measurable*]: *m* ∈ *borel-measurable M* **using** *Lp-measurable*[*OF* ‹*m* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)›] **by** *auto*
have *f n* ∈ *space*$_N$($\mathfrak{L}$ *p M*) **for** *n*
apply (*rule Lp-domination*[*OF* - ‹*m* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)›]) **using** ‹*AE x in M*. |*f n x*| ≤ *m x*› **by** *auto*

have *AE x in M*. ∀ *n*. |*f n x*| ≤ *m x*
apply (*subst AE-all-countable*) **using** ‹⋀*n*. *AE x in M*. |*f n x*| ≤ *m x*› **by** *auto*
moreover have |*g x*| ≤ *m x* **if** ∀ *n*. |*f n x*| ≤ *m x* (λ*n*. *f n x*) ⟶ *g x* **for** *x*
apply (*rule LIMSEQ-le-const2*[*of* λ*n*. |*f n x*|]) **using** *that* **by** (*auto intro*!:*tendsto-intros*)
ultimately have ∗: *AE x in M*. |*g x*| ≤ *m x* **using** ‹*AE x in M*. (λ*n*. *f n x*) ⟶ *g x*› **by** *auto*
show *g* ∈ *space*$_N$($\mathfrak{L}$ *p M*)
apply (*rule Lp-domination*[*OF* - ‹*m* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)›]) **using** ∗ **by** *auto*

have (λ*n*. *Norm* ($\mathfrak{L}$ *p M*) (*f n* − *g*)) ⟶ *0*
proof (*cases p* ≤ *0*)
case *True*
then have *ennreal p* = *0* **by** (*simp add*: *ennreal-eq-0-iff*)
then show *?thesis* **unfolding** *Norm-def* **by** (*auto simp add*: *L-zero*(*1*))
next
case *False*
then have *p* > *0* **by** *auto*
have (λ*n*. (∫ *x*. |*f n x* − *g x*| *powr p* ∂*M*)) ⟶ (∫ *x*. |*0*| *powr p* ∂*M*)
proof (*rule integral-dominated-convergence*[*of* - - - (λ*x*. |*2* ∗ *m x*| *powr p*)], *auto*)
show *integrable M* (λ*x*. |*2* ∗ *m x*| *powr p*)
unfolding *abs-mult* **apply** (*subst powr-mult*)
using *Lp-D*(*2*)[*OF* ‹*p* > *0*› ‹*m* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)›] **by** *auto*
have (λ*n*. |*f n x* − *g x*| *powr p*) ⟶ |*0*| *powr p* **if** (λ*n*. *f n x*) ⟶ *g x*

for x
apply (*rule tendsto-powr'*) **using** ‹$p > 0$› *that* **apply** (*auto*)
using *Lim-null tendsto-rabs-zero-iff* **by** *fastforce*
then show *AE x in M*. $(\lambda n.\ |f\ n\ x - g\ x|\ powr\ p) \longrightarrow 0$
using ‹$AE\ x\ in\ M.\ (\lambda n.\ f\ n\ x) \longrightarrow g\ x$› **by** *auto*
have $|f\ n\ x - g\ x|\ powr\ p \leq |2 * m\ x|\ powr\ p$ **if** $|f\ n\ x| \leq m\ x\ |g\ x| \leq m\ x$ **for** $n\ x$
using *powr-mono2* ‹$p > 0$› *that* **by** *auto*
then show *AE x in M*. $|f\ n\ x - g\ x|\ powr\ p \leq |2 * m\ x|\ powr\ p$ **for** n
using ‹$AE\ x\ in\ M.\ |f\ n\ x| \leq m\ x$› ‹$AE\ x\ in\ M.\ |g\ x| \leq m\ x$› **by** *auto*
qed
then have $(\lambda n.\ (Norm\ (\mathfrak{L}\ p\ M)\ (f\ n - g))\ powr\ p) \longrightarrow (Norm\ (\mathfrak{L}\ p\ M)\ 0)\ powr\ p$
unfolding *Lp-D*[*OF* ‹$p > 0$› *spaceN-diff*[*OF* ‹$\bigwedge n.\ f\ n \in space_N(\mathfrak{L}\ p\ M)$› ‹$g \in space_N(\mathfrak{L}\ p\ M)$›]]
using ‹$p > 0$› **by** (*auto simp add*: *powr-powr*)
then have $(\lambda n.\ ((Norm\ (\mathfrak{L}\ p\ M)\ (f\ n - g))\ powr\ p)\ powr\ (1/p)) \longrightarrow ((Norm\ (\mathfrak{L}\ p\ M)\ 0)\ powr\ p)\ powr\ (1/p)$
by (*rule tendsto-powr'*, *auto simp add*: ‹$p > 0$›)
then show *?thesis* **using** *powr-powr* ‹$p > 0$› **by** *auto*
qed
then show $tendsto\text{-}in_N\ (\mathfrak{L}\ p\ M)\ f\ g$
unfolding $tendsto\text{-}in_N$*-def* **by** *auto*
qed

We give the version of the monotone convergence theorem in the setting of L^p spaces.

proposition *Lp-monotone-limit*:
fixes f::$nat \Rightarrow 'a \Rightarrow real$
assumes $p > (0$::$ennreal)$
AE x in M. *incseq* $(\lambda n.\ f\ n\ x)$
$\bigwedge n.\ Norm\ (\mathfrak{L}\ p\ M)\ (f\ n) \leq C$
$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$
shows *AE x in M*. *convergent* $(\lambda n.\ f\ n\ x)$
$(\lambda x.\ lim\ (\lambda n.\ f\ n\ x)) \in space_N\ (\mathfrak{L}\ p\ M)$
$Norm\ (\mathfrak{L}\ p\ M)\ (\lambda x.\ lim\ (\lambda n.\ f\ n\ x)) \leq C$
proof −
have [*measurable*]: $f\ n \in$ *borel-measurable M* **for** n **using** *Lp-measurable*[*OF assms*(*4*)]**.**
show *AE x in M*. *convergent* $(\lambda n.\ f\ n\ x)$
proof (*cases rule*: *Lp-cases*[*of p*])
case *PInf*
have *AE x in M*. $|f\ n\ x| \leq C$ **for** n
using *L-infinity-AE-bound*[*of f n M*] ‹$Norm\ (\mathfrak{L}\ p\ M)\ (f\ n) \leq C$› ‹$f\ n \in space_N\ (\mathfrak{L}\ p\ M)$›
unfolding ‹$p=\infty$› **by** *auto*
then have ∗: *AE x in M*. $\forall n.\ |f\ n\ x| \leq C$
by (*subst AE-all-countable*, *auto*)
have $(\lambda n.\ f\ n\ x) \longrightarrow (SUP\ n.\ f\ n\ x)$ **if** *incseq* $(\lambda n.\ f\ n\ x)$ $\bigwedge n.\ |f\ n\ x| \leq C$

for x
apply (*rule LIMSEQ-incseq-SUP*[*OF* - ‹*incseq* ($\lambda n.$ f n x)›]) **using** *that*(*2*) *abs-le-D1* **by** *fastforce*
then have *convergent* ($\lambda n.$ f n x) **if** *incseq* ($\lambda n.$ f n x) $\bigwedge n.$ $|f\ n\ x| \leq C$ **for** x
unfolding *convergent-def* **using** *that* **by** *auto*
then show *?thesis* **using** ‹AE x *in* $M.$ *incseq* ($\lambda n.$ f n x)› $*$ **by** *auto*
next
case (*real-pos p2*)
define g **where** $g = (\lambda n.\ f\ n - f\ 0)$
have AE x *in* $M.$ *incseq* ($\lambda n.$ g n x)
unfolding *g-def* **using** ‹AE x *in* $M.$ *incseq* ($\lambda n.$ f n x)› **by** (*simp add*: *incseq-def*)
have $g\ n \in space_N\ (\mathfrak{L}\ p2\ M)$ **for** n
unfolding *g-def* **using** ‹$\bigwedge n.$ $f\ n \in space_N\ (\mathfrak{L}\ p\ M)$› **unfolding** ‹$p = ennreal\ p2$› **by** *auto*
then have [*measurable*]: $g\ n \in$ *borel-measurable* M **for** n **using** *Lp-measurable* **by** *auto*
define D **where** $D =$ *defect* ($\mathfrak{L}$ *p2* M) $*$ C $+$ *defect* ($\mathfrak{L}$ *p2* M) $*$ C
have *Norm* ($\mathfrak{L}$ *p2* M) (g n) $\leq D$ **for** n
proof $-$
have $f\ n \in space_N\ (\mathfrak{L}\ p2\ M)$ **using** ‹$f\ n \in space_N\ (\mathfrak{L}\ p\ M)$› **unfolding** ‹$p = ennreal\ p2$› **by** *auto*
have *Norm* ($\mathfrak{L}$ *p2* M) (g n) $\leq$ *defect* ($\mathfrak{L}$ *p2* M) $*$ *Norm* ($\mathfrak{L}$ *p2* M) (f n) $+$ *defect* ($\mathfrak{L}$ *p2* M) $*$ *Norm* ($\mathfrak{L}$ *p2* M) (f *0*)
unfolding *g-def* **using** *Norm-triangular-ineq-diff*[*OF* ‹$f\ n \in space_N\ (\mathfrak{L}\ p2\ M)$›] **by** *auto*
also have ... $\leq D$
unfolding *D-def* **apply**(*rule add-mono*)
using *mult-left-mono defect-ge-1*[*of* $\mathfrak{L}$ *p2* M] ‹$\bigwedge n.$ *Norm* ($\mathfrak{L}$ p M) (f n) $\leq C$› **unfolding** ‹$p = ennreal\ p2$› **by** *auto*
finally show *?thesis* **by** *simp*
qed
have *g-bound*: $(\int^+ x.\ |g\ n\ x|\ powr\ p2\ \partial M) \leq ennreal(D\ powr\ p2)$ **for** n
proof $-$
have $(\int^+ x.\ |g\ n\ x|\ powr\ p2\ \partial M) = ennreal(\int x.\ |g\ n\ x|\ powr\ p2\ \partial M)$
apply (*rule nn-integral-eq-integral*) **using** *Lp-D*(*2*)[*OF* ‹$p2 > 0$› ‹$g\ n \in space_N\ (\mathfrak{L}\ p2\ M)$›] **by** *auto*
also have ... $=$ *ennreal*((*Norm* ($\mathfrak{L}$ *p2* M) (g n)) *powr p2*)
apply (*subst Lp-Norm*(*2*)[*OF* ‹$p2 > 0$›, *of g n*, *symmetric*]) **by** *auto*
also have ... $\leq$ *ennreal*(D *powr p2*)
by (*auto intro*!: *powr-mono2 simp add*: *less-imp-le*[*OF* ‹$p2 > 0$›] ‹*Norm* ($\mathfrak{L}$ *p2* M) (g n) $\leq D$›)
finally show *?thesis* **by** *simp*
qed
have $\forall\, n.\ g\ n\ x \geq 0$ **if** *incseq* ($\lambda n.$ f n x) **for** x
unfolding *g-def* **using** *that* **by** (*auto simp add*: *incseq-def*)
then have AE x *in* $M.$ $\forall\, n.\ g\ n\ x \geq 0$ **using** ‹AE x *in* $M.$ *incseq* ($\lambda n.$ f n x)› **by** *auto*

define h **where** $h = (\lambda n\ x.\ ennreal(|g\ n\ x|\ powr\ p2))$
have [*measurable*]: $h\ n \in borel\text{-}measurable\ M$ **for** n **unfolding** *h-def* **by** *auto*
define H **where** $H = (\lambda x.\ (SUP\ n.\ h\ n\ x))$
have [*measurable*]: $H \in borel\text{-}measurable\ M$ **unfolding** *H-def* **by** *auto*
have $\bigwedge n.\ h\ n\ x \leq h\ (Suc\ n)\ x$ **if** $\forall\, n.\ g\ n\ x \geq 0\ incseq\ (\lambda n.\ g\ n\ x)$ **for** x
unfolding *h-def* **apply** (*auto intro*!:*powr-mono2*)
apply (*auto simp add*: *less-imp-le*[*OF* ‹*p2* > *0*›]) **using** *that incseq-SucD* **by** *auto*
then have $*$: $AE\ x\ in\ M.\ h\ n\ x \leq h\ (Suc\ n)\ x$ **for** n
using ‹$AE\ x\ in\ M.\ \forall\, n.\ g\ n\ x \geq 0$› ‹$AE\ x\ in\ M.\ incseq\ (\lambda n.\ g\ n\ x)$› **by** *auto*
have $(\int^+ x.\ H\ x\ \partial M) = (SUP\ n.\ \int^+ x.\ h\ n\ x\ \partial M)$
unfolding *H-def* **by** (*rule nn-integral-monotone-convergence-SUP-AE*, *auto simp add*: $*$)
also have ... $\leq ennreal(D\ powr\ p2)$
unfolding *H-def h-def* **using** *g-bound* **by** (*simp add*: *SUP-least*)
finally have $(\int^+ x.\ H\ x\ \partial M) < \infty$ **by** (*simp add*: *le-less-trans*)
then have $AE\ x\ in\ M.\ H\ x \neq \infty$
by (*metis* (*mono-tags*, *lifting*) ‹$H \in borel\text{-}measurable\ M$› *infinity-ennreal-def nn-integral-noteq-infinite top.not-eq-extremum*)

have *convergent* $(\lambda n.\ f\ n\ x)$ **if** $H\ x \neq \infty\ incseq\ (\lambda n.\ f\ n\ x)$ **for** x
proof −
define A **where** $A = enn2real(H\ x)$
then have $H\ x = ennreal\ A$ **using** ‹$H\ x \neq \infty$› **by** (*simp add*: *ennreal-enn2real-if*)
have $f\ n\ x \leq f\ 0\ x + A\ powr\ (1/p2)$ **for** n
proof −
have $ennreal(|g\ n\ x|\ powr\ p2) \leq ennreal\ A$
unfolding ‹$H\ x = ennreal\ A$›[*symmetric*] *H-def h-def* **by** (*meson SUP-upper2 UNIV-I order-refl*)
then have $|g\ n\ x|\ powr\ p2 \leq A$
by (*subst ennreal-le-iff*[*symmetric*], *auto simp add*: *A-def*)
have $|g\ n\ x| = (|g\ n\ x|\ powr\ p2)\ powr\ (1/p2)$
using ‹*p2* > *0*› **by** (*simp add*: *powr-powr*)
also have ... $\leq A\ powr\ (1/p2)$
apply (*rule powr-mono2*) **using** ‹*p2* > *0*› ‹$|g\ n\ x|\ powr\ p2 \leq A$› **by** *auto*
finally have $|g\ n\ x| \leq A\ powr\ (1/p2)$ **by** *simp*
then show *?thesis* **unfolding** *g-def* **by** *auto*
qed
then show *convergent* $(\lambda n.\ f\ n\ x)$
using *LIMSEQ-incseq-SUP*[*OF* - ‹*incseq* $(\lambda n.\ f\ n\ x)$›] *convergent-def* **by** (*metis bdd-aboveI2*)
qed
then show $AE\ x\ in\ M.\ convergent\ (\lambda n.\ f\ n\ x)$
using ‹$AE\ x\ in\ M.\ H\ x \neq \infty$› ‹$AE\ x\ in\ M.\ incseq\ (\lambda n.\ f\ n\ x)$› **by** *auto*
qed (*insert* ‹*p*>*0*›, *simp*)
then have *lim*: $AE\ x\ in\ M.\ (\lambda n.\ f\ n\ x) \longrightarrow lim\ (\lambda n.\ f\ n\ x)$
using *convergent-LIMSEQ-iff* **by** *auto*
show $(\lambda x.\ lim\ (\lambda n.\ f\ n\ x)) \in space_N\ (\mathfrak{L}\ p\ M)$
apply (*rule Lp-AE-limit''*[*of* - - *f*, *OF* - ‹$\bigwedge n.\ f\ n \in space_N\ (\mathfrak{L}\ p\ M)$› *lim* ‹$\bigwedge n.$

Norm ($\mathfrak{L}$ *p M*) (*f n*) ≤ *C*›])
by *auto*
show *Norm* ($\mathfrak{L}$ *p M*) (λ*x*. *lim* (λ*n*. *f n x*)) ≤ *C*
apply (*rule Lp-AE-limit''*[*of* - - *f*, *OF* - ‹⋀*n*. *f n* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)› *lim* ‹⋀*n*. *Norm* ($\mathfrak{L}$ *p M*) (*f n*) ≤ *C*›])
by *auto*
qed

5.7 Completeness of L^p

We prove the completeness of L^p.

theorem *Lp-complete*:
complete$_N$ ($\mathfrak{L}$ *p M*)
proof (*cases rule*: *Lp-cases*[*of p*])
case *zero*
show *?thesis*
proof (*rule complete*$_N$*-I*)
fix *u* **assume** ∀(*n*::*nat*). *u n* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)
then have *tendsto-in*$_N$ ($\mathfrak{L}$ *p M*) *u 0*
unfolding *tendsto-in*$_N$*-def Norm-def* ‹*p* = *0*› *L-zero*(*1*) *L-zero-space* **by** *auto*
then show ∃ *x*∈*space*$_N$ ($\mathfrak{L}$ *p M*). *tendsto-in*$_N$ ($\mathfrak{L}$ *p M*) *u x*
by *auto*
qed
next
case (*real-pos p2*)
show *?thesis*
proof (*rule complete*$_N$*-I'*[*of* λ*n*. (*1*/*2*)^*n* ∗ (*1*/(*defect* ($\mathfrak{L}$ *p M*))^(*Suc n*))], *unfold* ‹*p* = *ennreal p2*›)
show *0* < (*1*/*2*) ^ *n* ∗ (*1* / *defect* ($\mathfrak{L}$ (*ennreal p2*) *M*) ^ *Suc n*) **for** *n*
using *defect-ge-1*[*of* $\mathfrak{L}$ (*ennreal p2*) *M*] **by** (*auto simp add*: *divide-simps*)

fix *u* **assume** ∀(*n*::*nat*). *u n* ∈ *space*$_N$ ($\mathfrak{L}$ *p2 M*) ∀ *n*. *Norm* ($\mathfrak{L}$ *p2 M*) (*u n*) ≤ (*1*/*2*)^*n* ∗ (*1*/(*defect* ($\mathfrak{L}$ *p2 M*))^(*Suc n*))
then have *H*: ⋀*n*. *u n* ∈ *space*$_N$ ($\mathfrak{L}$ *p2 M*)
⋀*n*. *Norm* ($\mathfrak{L}$ *p2 M*) (*u n*) ≤ (*1*/*2*) ^ *n* ∗ (*1*/(*defect* ($\mathfrak{L}$ *p2 M*))^(*Suc n*))
unfolding ‹*p* = *ennreal p2*› **by** *auto*
have [*measurable*]: *u n* ∈ *borel-measurable M* **for** *n* **using** *Lp-measurable*[*OF H*(*1*)]**.**

define *w* **where** *w* = (λ*N x*. (∑ *n*∈{..<*N*}. |*u n x*|))
have *w2*: *w* = (λ*N*. *sum* (λ*n x*. |*u n x*|) {..<*N*}) **unfolding** *w-def* **apply** (*rule ext*)+
by (*metis* (*mono-tags*, *lifting*) *sum.cong fun-sum-apply*)
have *incseq* (λ*N*. *w N x*) **for** *x* **unfolding** *w2* **by** (*rule incseq-SucI*, *auto*)
then have *wN-inc*: *AE x in M*. *incseq* (λ*N*. *w N x*) **by** *simp*

have *abs-u-space*: (λ*x*. |*u n x*|) ∈ *space*$_N$ ($\mathfrak{L}$ *p2 M*) **for** *n*
by (*rule Lp-Banach-lattice*[*OF* ‹*u n* ∈ *space*$_N$ ($\mathfrak{L}$ *p2 M*)›])

then have *wN-space*: $w\ N \in space_N\ (\mathfrak{L}\ p2\ M)$ **for** N **unfolding** $w2$ **using** $H(1)$ **by** *auto*
have *abs-u-Norm*: $Norm\ (\mathfrak{L}\ p2\ M)\ (\lambda x.\ |u\ n\ x|) \leq (1/2)\ \hat{}\ n * (1/(defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n))$ **for** n
using *Lp-Banach-lattice*(2)[*OF* ‹$u\ n \in space_N\ (\mathfrak{L}\ p2\ M)$›] $H(2)$ **by** *auto*

have *wN-Norm*: $Norm\ (\mathfrak{L}\ p2\ M)\ (w\ N) \leq 2$ **for** N
proof −
have $*$: $(defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n) \geq 0\ (defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n) > 0$ **for** n
using *defect-ge-1*[*of* $\mathfrak{L}\ p2\ M$] **by** *auto*
have $Norm\ (\mathfrak{L}\ p2\ M)\ (w\ N) \leq (\sum n{<}N.\ (defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n) * Norm\ (\mathfrak{L}\ p2\ M)\ (\lambda x.\ |u\ n\ x|))$
unfolding *w2 lessThan-Suc-atMost*[*symmetric*] **by** (*rule Norm-sum*, *simp add*: *abs-u-space*)
also have $... \leq (\sum n{<}N.\ (defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n) * ((1/2)\ \hat{}\ n * (1/(defect\ (\mathfrak{L}\ p2\ M))\hat{}(Suc\ n))))$
apply (*rule sum-mono*, *rule mult-left-mono*) **using** *abs-u-Norm* $*$ **by** *auto*
also have $... = (\sum n{<}N.\ (1/2)\ \hat{}\ n)$
using $*(2)$ *defect-ge-1*[*of* $\mathfrak{L}\ p2\ M$] **by** (*auto simp add*: *algebra-simps*)
also have $... \leq (\sum n.\ (1/2)\ \hat{}\ n)$
unfolding *lessThan-Suc-atMost*[*symmetric*] **by** (*rule sum-le-suminf*, *rule summable-geometric*[*of* 1/2], *auto*)
also have $... = 2$ **using** *suminf-geometric*[*of* 1/2] **by** *auto*
finally show *?thesis* **by** *simp*
qed

have *AE x in M*. *convergent* $(\lambda N.\ w\ N\ x)$
apply (*rule Lp-monotone-limit*[*OF* ‹$p > 0$›, *of* - - 2], *unfold* ‹$p = ennreal\ p2$›)
using *wN-inc wN-Norm wN-space* **by** *auto*
define m **where** $m = (\lambda x.\ lim\ (\lambda N.\ w\ N\ x))$
have *m-space*: $m \in space_N\ (\mathfrak{L}\ p2\ M)$
unfolding *m-def* ‹$p = ennreal\ p2$›[*symmetric*] **apply** (*rule Lp-monotone-limit*[*OF* ‹$p > 0$›, *of* - - 2], *unfold* ‹$p = ennreal\ p2$›)
using *wN-inc wN-Norm wN-space* **by** *auto*

define v **where** $v = (\lambda x.\ (\sum n.\ u\ n\ x))$
have *v-meas*: $v \in borel\text{-}measurable\ M$ **unfolding** *v-def* **by** *auto*
have *u-meas*: $\bigwedge n.\ (sum\ u\ \{0..{<}n\}) \in borel\text{-}measurable\ M$ **by** *auto*
{
fix x **assume** *convergent* $(\lambda N.\ w\ N\ x)$
then have S: *summable* $(\lambda n.\ |u\ n\ x|)$ **unfolding** *w-def* **using** *summable-iff-convergent* **by** *auto*
then have $m\ x = (\sum n.\ |u\ n\ x|)$ **unfolding** *m-def w-def* **by** (*metis sum-inf-eq-lim*)

have *summable* $(\lambda n.\ u\ n\ x)$ **using** S **by** (*rule summable-rabs-cancel*)
then have $*$: $(\lambda n.\ (sum\ u\ \{..{<}n\})\ x) \longrightarrow v\ x$
unfolding *v-def fun-sum-apply* **by** (*metis convergent-LIMSEQ-iff sum-*

inf-eq-lim summable-iff-convergent)
have $|(sum\ u\ \{..<n\})\ x| \leq m\ x$ **for** n
proof −
have $|(sum\ u\ \{..<n\})\ x| \leq (\sum i{\in}\{..<n\}.\ |u\ i\ x|)$
unfolding *fun-sum-apply* **by** *auto*
also have $... \leq (\sum i.\ |u\ i\ x|)$
apply (*rule sum-le-suminf*) **using** S **by** *auto*
finally show *?thesis* **using** ‹$m\ x = (\sum n.\ |u\ n\ x|)$› **by** *simp*
qed
then have $(\forall n.\ |(sum\ u\ \{0..<n\})\ x| \leq m\ x) \wedge (\lambda n.\ (sum\ u\ \{0..<n\})\ x) \longrightarrow v\ x$
unfolding *atLeast0LessThan* **using** $*$ **by** *auto*
}
then have *m-bound*: $\bigwedge n.\ AE\ x\ in\ M.\ |(sum\ u\ \{0..<n\})\ x| \leq m\ x$
and *u-conv*: $AE\ x\ in\ M.\ (\lambda n.\ (sum\ u\ \{0..<n\})\ x) \longrightarrow v\ x$
using ‹$AE\ x\ in\ M.\ convergent\ (\lambda N.\ w\ N\ x)$› **by** *auto*

have *tendsto-in*$_N$ $(\mathfrak{L}\ p2\ M)\ (\lambda n.\ sum\ u\ \{0..<n\})\ v$
by (*rule Lp-domination-limit*[*OF v-meas u-meas m-space u-conv m-bound*])
moreover have $v \in space_N\ (\mathfrak{L}\ p2\ M)$
by (*rule Lp-domination-limit*[*OF v-meas u-meas m-space u-conv m-bound*])
ultimately show $\exists v \in space_N\ (\mathfrak{L}\ p2\ M).$ *tendsto-in*$_N$ $(\mathfrak{L}\ p2\ M)\ (\lambda n.\ sum\ u\ \{0..<n\})\ v$
by *auto*
qed
next
case *PInf*
show *?thesis*
proof (*rule complete*$_N$*-I'*[*of* $\lambda n.\ (1/2)\hat{}n$])
fix u **assume** $\forall (n::nat).\ u\ n \in space_N\ (\mathfrak{L}\ p\ M)\ \forall n.\ Norm\ (\mathfrak{L}\ p\ M)\ (u\ n) \leq (1/2)\ \hat{}\ n$
then have H: $\bigwedge n.\ u\ n \in space_N\ (\mathfrak{L}\ \infty\ M)\ \bigwedge n.\ Norm\ (\mathfrak{L}\ \infty\ M)\ (u\ n) \leq (1/2)\ \hat{}\ n$ **using** *PInf* **by** *auto*
have [*measurable*]: $u\ n \in borel\text{-}measurable\ M$ **for** n **using** *Lp-measurable*[*OF H(1)*] **by** *auto*
define v **where** $v = (\lambda x.\ \sum n.\ u\ n\ x)$
have [*measurable*]: $v \in borel\text{-}measurable\ M$ **unfolding** *v-def* **by** *auto*
define w **where** $w = (\lambda N\ x.\ (\sum n{\in}\{0..<N\}.\ u\ n\ x))$
have [*measurable*]: $w\ N \in borel\text{-}measurable\ M$ **for** N **unfolding** *w-def* **by** *auto*

have $AE\ x\ in\ M.\ |u\ n\ x| \leq (1/2)\hat{}n$ **for** n
using *L-infinity-AE-bound*[*OF H(1), of n*] *H(2)*[*of n*] **by** *auto*
then have $AE\ x\ in\ M.\ \forall n.\ |u\ n\ x| \leq (1/2)\hat{}n$
by (*subst AE-all-countable*, *auto*)
moreover have $|w\ N\ x - v\ x| \leq (1/2)\hat{}N * 2$ **if** $\forall n.\ |u\ n\ x| \leq (1/2)\hat{}n$ **for** $N\ x$
proof −
have $*$: $\bigwedge n.\ |u\ n\ x| \leq (1/2)\hat{}n$ **using** *that* **by** *auto*
have $**$: *summable* $(\lambda n.\ |u\ n\ x|)$

```
        apply (rule summable-norm-cancel, rule summable-comparison-test'[OF
summable-geometric[of 1/2]])
       using ∗ by auto
      have |w N x − v x| = |(∑ n. u (n + N) x)|
       unfolding v-def w-def
         apply (subst suminf-split-initial-segment[OF summable-rabs-cancel[OF
‹summable (λn. |u n x|)›], of N])
       by (simp add: lessThan-atLeast0)
      also have ... ≤ (∑ n. |u (n + N) x|)
       apply (rule summable-rabs, subst summable-iff-shift) using ∗∗ by auto
      also have ... ≤ (∑ n. (1/2)^(n + N))
      proof (rule suminf-le)
       show ⋀n. |u (n + N) x| ≤ (1/2) ^ (n + N)
        using ∗[of - + N] by simp
       show summable (λn. |u (n + N) x|)
        using ∗∗ by (subst summable-iff-shift) simp
       show summable (λn. (1/2::real) ^ (n + N))
        using summable-geometric [of 1/2] by (subst summable-iff-shift) simp
      qed
      also have ... = (1/2)^N ∗ (∑ n. (1/2)^n)
          by (subst power-add, subst suminf-mult2[symmetric], auto simp add:
summable-geometric[of 1/2])
      also have ... = (1/2)^N ∗ 2
       by (subst suminf-geometric, auto)
      finally show ?thesis by simp
    qed
    ultimately have ∗: AE x in M. |w N x − v x| ≤ (1/2)^N ∗ 2 for N by auto

    have ∗∗: w N − v ∈ space_N (𝔏 ∞ M) Norm (𝔏 ∞ M) (w N − v) ≤ (1/2)^N
∗ 2 for N
      unfolding fun-diff-def using L-infinity-I[OF - ∗] by auto
    have l: (λN. ((1/2)^N) ∗ (2::real)) ⟶ 0 ∗ 2
      by (rule tendsto-mult, auto simp add: LIMSEQ-realpow-zero[of 1/2])
    have tendsto-in_N (𝔏 ∞ M) w v unfolding tendsto-in_N-def
      apply (rule tendsto-sandwich[of λ-. 0 - - λn. (1/2)^n ∗ 2]) using l ∗∗(2)
by auto
    have v = − (w 0 − v) unfolding w-def by auto
    then have v ∈ space_N (𝔏 ∞ M) using ∗∗(1)[of 0] spaceN-add spaceN-diff by
fastforce
    then show ∃ v ∈ space_N (𝔏 p M). tendsto-in_N (𝔏 p M) (λn. sum u {0..<n})
v
     using ‹tendsto-in_N (𝔏 ∞ M) w v› unfolding ‹p = ∞› w-def fun-sum-apply[symmetric]
by auto
  qed (simp)
qed
```

5.8 Multiplication of functions, duality

The next theorem asserts that the multiplication of two functions in L^p and L^q belongs to L^r, where r is determined by the equality $1/r = 1/p + 1/q$. This is essentially a case by case analysis, depending on the kind of L^p space we are considering. The only nontrivial case is when p, q (and r) are finite and nonzero. In this case, it reduces to Hölder inequality.

theorem *Lp-Lq-mult*:
fixes *p q r*::*ennreal*
assumes *1/p + 1/q = 1/r*
and *f* ∈ *space*$_N$ (𝔏 *p M*) *g* ∈ *space*$_N$ (𝔏 *q M*)
shows (λ*x*. *f x* ∗ *g x*) ∈ *space*$_N$ (𝔏 *r M*)
Norm (𝔏 *r M*) (λ*x*. *f x* ∗ *g x*) ≤ *Norm* (𝔏 *p M*) *f* ∗ *Norm* (𝔏 *q M*) *g*
proof −
have [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M* **using** *Lp-measurable assms* **by** *auto*
have (λ*x*. *f x* ∗ *g x*) ∈ *space*$_N$ (𝔏 *r M*) ∧ *Norm* (𝔏 *r M*) (λ*x*. *f x* ∗ *g x*) ≤ *Norm* (𝔏 *p M*) *f* ∗ *Norm* (𝔏 *q M*) *g*
proof (*cases rule*: *Lp-cases*[*of r*])
case *zero*
have ∗: (λ*x*. *f x* ∗ *g x*) ∈ *borel-measurable M* **by** *auto*
then have *Norm* (𝔏 *r M*) (λ*x*. *f x* ∗ *g x*) = *0* **using** *L-zero*[*of M*] **unfolding** *Norm-def zero* **by** *auto*
then have *Norm* (𝔏 *r M*) (λ*x*. *f x* ∗ *g x*) ≤ *Norm* (𝔏 *p M*) *f* ∗ *Norm* (𝔏 *q M*) *g*
using *Norm-nonneg* **by** *auto*
then show *?thesis* **unfolding** *zero* **using** ∗ *L-zero-space*[*of M*] **by** *auto*
next
case (*real-pos r2*)
have *p* > *0 q* > *0* **using** ‹*1/p + 1/q = 1/r*› ‹*r* > *0*›
by (*metis ennreal-add-eq-top ennreal-divide-eq-top-iff ennreal-top-neq-one gr-zeroI zero-neq-one*)+
consider *p* = ∞ | *q* = ∞ | *p* < ∞ ∧ *q* < ∞ **using** *top.not-eq-extremum* **by** *force*
then show *?thesis*
proof (*cases*)
case *1*
then have *q* = *r* **using** ‹*1/p + 1/q = 1/r*›
by (*metis ennreal-divide-top infinity-ennreal-def one-divide-one-divide-ennreal semiring-normalization-rules*(*5*))
have *AE x in M*. |*f x*| ≤ *Norm* (𝔏 *p M*) *f*
using ‹*f* ∈ *space*$_N$ (𝔏 *p M*)› *L-infinity-AE-bound* **unfolding** ‹*p* = ∞› **by** *auto*
then have ∗: *AE x in M*. |*f x* ∗ *g x*| ≤ |*Norm* (𝔏 *p M*) *f* ∗ *g x*|
unfolding *abs-mult* **using** *Norm-nonneg*[*of* 𝔏 *p M f*] *mult-right-mono* **by** *fastforce*
have ∗∗: (λ*x*. *Norm* (𝔏 *p M*) *f* ∗ *g x*) ∈ *space*$_N$ (𝔏 *r M*)
using *spaceN-cmult*[*OF* ‹*g* ∈ *space*$_N$ (𝔏 *q M*)›] **unfolding** ‹*q* = *r*›

scaleR-fun-def **by** *simp*
have ∗∗∗: *Norm* (𝔏 *r M*) (λ*x*. *Norm* (𝔏 *p M*) *f* ∗ *g x*) = *Norm* (𝔏 *p M*) *f* ∗ *Norm* (𝔏 *q M*) *g*
using *Norm-cmult*[*of* 𝔏 *r M*] **unfolding** ‹*q* = *r*› *scaleR-fun-def* **by** *auto*
then show *?thesis*
using *Lp-domination*[*of* λ*x*. *f x* ∗ *g x M* λ*x*. *Norm* (𝔏 *p M*) *f* ∗ *g x r*] **unfolding** ‹*q* = *r*›
using ∗ ∗∗ ∗∗∗ **by** *auto*
next
case *2*
then have *p* = *r* **using** ‹*1*/*p* + *1*/*q* = *1*/*r*›
by (*metis add.right-neutral ennreal-divide-top infinity-ennreal-def one-divide-one-divide-ennreal*)
have *AE x in M*. |*g x*| ≤ *Norm* (𝔏 *q M*) *g*
using ‹*g* ∈ *space*$_N$ (𝔏 *q M*)› *L-infinity-AE-bound* **unfolding** ‹*q* = ∞› **by** *auto*
then have ∗: *AE x in M*. |*f x* ∗ *g x*| ≤ |*Norm* (𝔏 *q M*) *g* ∗ *f x*|
apply (*simp only*: *mult.commute*[*of Norm* (𝔏 *q M*) *g* -])
unfolding *abs-mult* **using** *mult-left-mono Norm-nonneg*[*of* 𝔏 *q M g*] **by** *fastforce*
have ∗∗: (λ*x*. *Norm* (𝔏 *q M*) *g* ∗ *f x*) ∈ *space*$_N$ (𝔏 *r M*)
using *spaceN-cmult*[*OF* ‹*f* ∈ *space*$_N$ (𝔏 *p M*)›] **unfolding** ‹*p* = *r*› *scaleR-fun-def* **by** *simp*
have ∗∗∗: *Norm* (𝔏 *r M*) (λ*x*. *Norm* (𝔏 *q M*) *g* ∗ *f x*) = *Norm* (𝔏 *p M*) *f* ∗ *Norm* (𝔏 *q M*) *g*
using *Norm-cmult*[*of* 𝔏 *r M*] **unfolding** ‹*p* = *r*› *scaleR-fun-def* **by** *auto*
then show *?thesis*
using *Lp-domination*[*of* λ*x*. *f x* ∗ *g x M* λ*x*. *Norm* (𝔏 *q M*) *g* ∗ *f x r*] **unfolding** ‹*p* = *r*›
using ∗ ∗∗ ∗∗∗ **by** *auto*
next
case *3*
obtain *p2* **where** *p* = *ennreal p2 p2* > *0*
using *enn2real-positive-iff*[*of p*] *3* ‹*p* > *0*› **by** (*cases p*) *auto*
obtain *q2* **where** *q* = *ennreal q2 q2* > *0*
using *enn2real-positive-iff*[*of q*] *3* ‹*q* > *0*› **by** (*cases q*) *auto*

have *ennreal*(*1*/*r2*) = *1*/*r*
using ‹*r* = *ennreal r2*› ‹*r2* > *0*› *divide-ennreal zero-le-one* **by** *fastforce*
also have ... = *1*/*p* + *1*/*q* **using** *assms* **by** *auto*
also have ... = *ennreal*(*1*/*p2* + *1*/*q2*) **using** ‹*p* = *ennreal p2*› ‹*p2* > *0*› ‹*q* = *ennreal q2*› ‹*q2* > *0*›
apply (*simp only*: *divide-ennreal ennreal-1*[*symmetric*]) **using** *ennreal-plus*[*of* *1*/*p2* *1*/*q2*, *symmetric*] **by** *auto*
finally have ∗: *1*/*r2* = *1*/*p2* + *1*/*q2*
using *ennreal-inj* ‹*p2* > *0*› ‹*q2* > *0*› ‹*r2* > *0*› **by** (*metis divide-pos-pos ennreal-less-zero-iff le-less zero-less-one*)

define *P* **where** *P* = *p2* / *r2*
define *Q* **where** *Q* = *q2* / *r2*

```
    have [simp]: P > 0 Q > 0 and 1/P + 1/Q = 1
     using ‹p2 > 0› ‹q2 > 0› ‹r2 > 0› ∗ unfolding P-def Q-def by (auto simp
add: divide-simps algebra-simps)
    have Pa: (|z| powr r2) powr P = |z| powr p2 for z
     unfolding P-def powr-powr using ‹r2 > 0› by auto
    have Qa: (|z| powr r2) powr Q = |z| powr q2 for z
     unfolding Q-def powr-powr using ‹r2 > 0› by auto

    have ∗: integrable M (λx. |f x| powr r2 ∗ |g x| powr r2)
     apply (rule Holder-inequality[OF ‹P>0› ‹Q>0› ‹1/P + 1/Q = 1›], auto
simp add: Pa Qa)
    using ‹f ∈ space_N (𝔏 p M)› unfolding ‹p = ennreal p2› using Lp-space[OF
‹p2 > 0›] apply auto
    using ‹g ∈ space_N (𝔏 q M)› unfolding ‹q = ennreal q2› using Lp-space[OF
‹q2 > 0›] by auto
    have (λx. f x ∗ g x) ∈ space_N (𝔏 r M)
     unfolding ‹r = ennreal r2› using Lp-space[OF ‹r2 > 0›, of M] by (auto
simp add: ∗ abs-mult powr-mult)
    have Norm (𝔏 r M) (λx. f x ∗ g x) = (∫ x. |f x ∗ g x| powr r2 ∂M) powr
(1/r2)
     unfolding ‹r = ennreal r2› using Lp-Norm[OF ‹r2 > 0›, of - M] by auto
    also have ... = abs (∫ x. |f x| powr r2 ∗ |g x| powr r2 ∂M) powr (1/r2)
     by (auto simp add: powr-mult abs-mult)
    also have ... ≤ ((∫ x. | |f x| powr r2 | powr P ∂M) powr (1/P) ∗ (∫ x. | |g
x| powr r2 | powr Q ∂M) powr (1/Q)) powr (1/r2)
     apply (rule powr-mono2, simp add: ‹r2 > 0› less-imp-le, simp)
     apply (rule Holder-inequality[OF ‹P>0› ‹Q>0› ‹1/P + 1/Q = 1›], auto
simp add: Pa Qa)
    using ‹f ∈ space_N (𝔏 p M)› unfolding ‹p = ennreal p2› using Lp-space[OF
‹p2 > 0›] apply auto
    using ‹g ∈ space_N (𝔏 q M)› unfolding ‹q = ennreal q2› using Lp-space[OF
‹q2 > 0›] by auto
    also have ... = (∫ x. |f x| powr p2 ∂M) powr (1/p2) ∗ (∫ x. |g x| powr q2
∂M) powr (1/q2)
     apply (auto simp add: powr-mult powr-powr) unfolding P-def Q-def using
‹r2 > 0› by auto
    also have ... = Norm (𝔏 p M) f ∗ Norm (𝔏 q M) g
     unfolding ‹p = ennreal p2› ‹q = ennreal q2›
     using Lp-Norm[OF ‹p2 > 0›, of - M] Lp-Norm[OF ‹q2 > 0›, of - M] by
auto
    finally show ?thesis using ‹(λx. f x ∗ g x) ∈ space_N (𝔏 r M)› by auto
   qed
  next
   case PInf
   then have p = ∞ q = r using ‹1/p + 1/q = 1/r›
    by (metis add-eq-0-iff-both-eq-0 ennreal-divide-eq-0-iff infinity-ennreal-def
not-one-le-zero order.order-iff-strict)+
   have AE x in M. |f x| ≤ Norm (𝔏 p M) f
    using ‹f ∈ space_N (𝔏 p M)› L-infinity-AE-bound unfolding ‹p = ∞› by
```

auto
then have $*$: *AE x in M.* $|f\ x * g\ x| \leq |Norm\ (\mathfrak{L}\ p\ M)\ f * g\ x|$
unfolding *abs-mult* **using** *Norm-nonneg*[*of* $\mathfrak{L}$ *p M f*] *mult-right-mono* **by** *fastforce*
have $**$: $(\lambda x.\ Norm\ (\mathfrak{L}\ p\ M)\ f * g\ x) \in space_N\ (\mathfrak{L}\ r\ M)$
using *spaceN-cmult*[*OF* ‹$g \in space_N\ (\mathfrak{L}\ q\ M)$›] **unfolding** ‹$q = r$› *scaleR-fun-def* **by** *simp*
have $***$: $Norm\ (\mathfrak{L}\ r\ M)\ (\lambda x.\ Norm\ (\mathfrak{L}\ p\ M)\ f * g\ x) = Norm\ (\mathfrak{L}\ p\ M)\ f * Norm\ (\mathfrak{L}\ q\ M)\ g$
using *Norm-cmult*[*of* $\mathfrak{L}$ *r M*] **unfolding** ‹$q = r$› *scaleR-fun-def* **by** *auto*
then show *?thesis*
using *Lp-domination*[*of* $\lambda x.\ f\ x * g\ x\ M\ \lambda x.\ Norm\ (\mathfrak{L}\ p\ M)\ f * g\ x\ r$] **unfolding** ‹$q = r$›
using $*$ $**$ $***$ **by** *auto*
qed
then show $(\lambda x.\ f\ x * g\ x) \in space_N\ (\mathfrak{L}\ r\ M)$
$Norm\ (\mathfrak{L}\ r\ M)\ (\lambda x.\ f\ x * g\ x) \leq Norm\ (\mathfrak{L}\ p\ M)\ f * Norm\ (\mathfrak{L}\ q\ M)\ g$
by *auto*
qed

The previous theorem admits an eNorm version in which one does not assume a priori that the functions under consideration belong to L^p or L^q.

theorem *Lp-Lq-emult*:
fixes *p q r*::*ennreal*
assumes $1/p + 1/q = 1/r$
$f \in$ *borel-measurable M g* $\in$ *borel-measurable M*
shows $eNorm\ (\mathfrak{L}\ r\ M)\ (\lambda x.\ f\ x * g\ x) \leq eNorm\ (\mathfrak{L}\ p\ M)\ f * eNorm\ (\mathfrak{L}\ q\ M)\ g$
proof (*cases* $r = 0$)
case *True*
then have $eNorm\ (\mathfrak{L}\ r\ M)\ (\lambda x.\ f\ x * g\ x) = 0$
using *assms* **by** (*simp add*: *L-zero*(*1*))
then show *?thesis* **by** *auto*
next
case *False*
then have $r > 0$ **using** *not-gr-zero* **by** *blast*
then have $p > 0\ q > 0$ **using** ‹$1/p + 1/q = 1/r$›
by (*metis ennreal-add-eq-top ennreal-divide-eq-top-iff ennreal-top-neq-one gr-zeroI zero-neq-one*)+
then have *Z*: $zero\text{-}space_N\ (\mathfrak{L}\ p\ M) = \{f \in$ *borel-measurable M. AE x in M.* $f\ x = 0\}$
$zero\text{-}space_N\ (\mathfrak{L}\ q\ M) = \{f \in$ *borel-measurable M. AE x in M.* $f\ x = 0\}$
$zero\text{-}space_N\ (\mathfrak{L}\ r\ M) = \{f \in$ *borel-measurable M. AE x in M.* $f\ x = 0\}$
using ‹$r > 0$› *Lp-infinity-zero-space* **by** *auto*
have [*measurable*]: $(\lambda x.\ f\ x * g\ x) \in$ *borel-measurable M* **using** *assms* **by** *auto*
consider $eNorm\ (\mathfrak{L}\ p\ M)\ f = 0 \vee eNorm\ (\mathfrak{L}\ q\ M)\ g = 0$
$|\ (eNorm\ (\mathfrak{L}\ p\ M)\ f > 0 \wedge eNorm\ (\mathfrak{L}\ q\ M)\ g = \infty) \vee (eNorm\ (\mathfrak{L}\ p\ M)\ f = \infty \wedge eNorm\ (\mathfrak{L}\ q\ M)\ g > 0)$
$|\ eNorm\ (\mathfrak{L}\ p\ M)\ f < \infty \wedge eNorm\ (\mathfrak{L}\ q\ M)\ g < \infty$
using *less-top* **by** *fastforce*

then show *?thesis*
proof (*cases*)
case *1*
then have (*AE x in M. f x = 0*) ∨ (*AE x in M. g x = 0*) **using** *Z* **unfolding** *zero-space$_N$-def* **by** *auto*
then have *AE x in M. f x * g x = 0* **by** *auto*
then have *eNorm* (𝔏 *r M*) (λ*x. f x * g x*) = *0* **using** *Z* **unfolding** *zero-space$_N$-def* **by** *auto*
then show *?thesis* **by** *simp*
next
case *2*
then have *eNorm* (𝔏 *p M*) *f * eNorm* (𝔏 *q M*) *g* = ∞ **using** *ennreal-mult-eq-top-iff* **by** *force*
then show *?thesis* **by** *auto*
next
case *3*
then have *∗*: *f* ∈ *space$_N$* (𝔏 *p M*) *g* ∈ *space$_N$* (𝔏 *q M*) **unfolding** *space$_N$-def* **by** *auto*
then have (λ*x. f x * g x*) ∈ *space$_N$* (𝔏 *r M*) **using** *Lp-Lq-mult*(*1*)[*OF assms*(*1*)] **by** *auto*
then show *?thesis*
using *Lp-Lq-mult*(*2*)[*OF assms*(*1*) *∗*] **by** (*simp add*: *eNorm-Norm ∗ ennreal-mult'*[*symmetric*])
qed
qed

lemma *Lp-Lq-duality-bound*:
fixes *p q*::*ennreal*
assumes *1/p + 1/q = 1*
f ∈ *space$_N$* (𝔏 *p M*)
g ∈ *space$_N$* (𝔏 *q M*)
shows *integrable M* (λ*x. f x * g x*)
abs(∫ *x. f x * g x* ∂*M*) ≤ *Norm* (𝔏 *p M*) *f * Norm* (𝔏 *q M*) *g*
proof −
have (λ*x. f x * g x*) ∈ *space$_N$* (𝔏 *1 M*)
apply (*rule Lp-Lq-mult*[*OF - ‹f* ∈ *space$_N$* (𝔏 *p M*)› ‹*g* ∈ *space$_N$* (𝔏 *q M*)›])
using ‹*1/p + 1/q = 1*› **by** *auto*
then show *integrable M* (λ*x. f x * g x*) **using** *L1-space* **by** *auto*

have *abs*(∫ *x. f x * g x* ∂*M*) ≤ *Norm* (𝔏 *1 M*) (λ*x. f x * g x*) **using** *L1-int-ineq* **by** *auto*
also have *...* ≤ *Norm* (𝔏 *p M*) *f * Norm* (𝔏 *q M*) *g*
apply (*rule Lp-Lq-mult*[*OF - ‹f* ∈ *space$_N$* (𝔏 *p M*)› ‹*g* ∈ *space$_N$* (𝔏 *q M*)›])
using ‹*1/p + 1/q = 1*› **by** *auto*
finally show *abs*(∫ *x. f x * g x* ∂*M*) ≤ *Norm* (𝔏 *p M*) *f * Norm* (𝔏 *q M*) *g* **by** *simp*
qed

The next theorem asserts that the norm of an L^p function f can be obtained

by estimating the integrals of fg over all L^q functions g, where $1/p+1/q = 1$. When $p = \infty$, it is necessary to assume that the space is sigma-finite: for instance, if the space is one single atom of infinite mass, then there is no nonzero L^1 function, so taking for f the constant function equal to 1, it has L^∞ norm equal to 1, but $\int fg = 0$ for all L^1 function g.

theorem *Lp-Lq-duality*:
fixes *p q*::*ennreal*
assumes $f \in space_N\ (\mathfrak{L}\ p\ M)$
$1/p + 1/q = 1$
$p = \infty \Longrightarrow$ *sigma-finite-measure M*
shows *bdd-above* $((\lambda g.\ (\int x.\ f\ x * g\ x\ \partial M))`\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\})$
$Norm\ (\mathfrak{L}\ p\ M)\ f = (SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * g\ x\ \partial M))$
proof −
have [*measurable*]: $f \in$ *borel-measurable M* **using** *Lp-measurable*[*OF assms*(*1*)] **by** *auto*
have *B*: $(\int x.\ f\ x * g\ x\ \partial M) \leq Norm\ (\mathfrak{L}\ p\ M)\ f$ **if** $g \in \{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}$ **for** *g*
proof −
have *g*: $g \in space_N\ (\mathfrak{L}\ q\ M)$ $Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1$ **using** *that* **by** *auto*
have $(\int x.\ f\ x * g\ x\ \partial M) \leq abs(\int x.\ f\ x * g\ x\ \partial M)$ **by** *auto*
also have ... $\leq Norm\ (\mathfrak{L}\ p\ M)\ f * Norm\ (\mathfrak{L}\ q\ M)\ g$
using *Lp-Lq-duality-bound*(*2*)[*OF* ‹*1/p + 1/q = 1*› ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$› *g*(*1*)] **by** *auto*
also have ... $\leq Norm\ (\mathfrak{L}\ p\ M)\ f$
using *g*(*2*) *Norm-nonneg*[*of* $\mathfrak{L}\ p\ M\ f$] *mult-left-le* **by** *blast*
finally show $(\int x.\ f\ x * g\ x\ \partial M) \leq Norm\ (\mathfrak{L}\ p\ M)\ f$ **by** *simp*
qed
then show *bdd-above* $((\lambda g.\ (\int x.\ f\ x * g\ x\ \partial M))`\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\})$
by (*meson bdd-aboveI2*)

show $Norm\ (\mathfrak{L}\ p\ M)\ f = (SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * g\ x\ \partial M))$
proof (*rule antisym*)
show $(SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ \int x.\ f\ x * g\ x\ \partial M) \leq Norm\ (\mathfrak{L}\ p\ M)\ f$
by (*rule cSUP-least*, *auto*, *rule exI*[*of - 0*], *auto simp add*: *B*)

have $p \geq 1$ **using** *conjugate-exponent-ennrealI*(*1*)[*OF* ‹*1/p + 1/q = 1*›] **by** *simp*
show $Norm\ (\mathfrak{L}\ p\ M)\ f \leq (SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * g\ x\ \partial M))$
using ‹$p \geq 1$› **proof** (*cases rule*: *Lp-cases-1-PInf*)
case *PInf*
then have $f \in space_N\ (\mathfrak{L}\ \infty\ M)$
using ‹$f \in space_N(\mathfrak{L}\ p\ M)$› **by** *simp*

have $q = 1$ **using** ‹$1/p + 1/q = 1$› ‹$p = \infty$› **by** (*simp add*: *divide-eq-1-ennreal*)
have $c \leq (SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * g\ x\ \partial M))$ **if** $c < Norm\ (\mathfrak{L}\ p\ M)\ f$ **for** c
proof (*cases* $c < 0$)
case *True*
then have $c \leq (\int x.\ f\ x * 0\ x\ \partial M)$ **by** *auto*
also have ... $\leq (SUP\ g{\in}\{g \in space_N\ (\mathfrak{L}\ q\ M).\ Norm\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * g\ x\ \partial M))$
apply (*rule cSUP-upper*, *auto simp add*: *zero-fun-def*[*symmetric*]) **using** *B* **by** (*meson bdd-aboveI2*)
finally show *?thesis* **by** *simp*
next
case *False*
then have *ennreal c* < *eNorm* ($\mathfrak{L}\ \infty\ M$) *f*
using *eNorm-Norm*[*OF* ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$›] *that ennreal-less-iff* **unfolding** ‹$p = \infty$› **by** *auto*
then have $*$: *emeasure M* $\{x \in space\ M.\ |f\ x| > c\} > 0$ **using** *L-infinity-pos-measure*[*of f M c*] **by** *auto*
obtain *A* **where** [*measurable*]: $\bigwedge(n{::}nat).\ A\ n \in sets\ M$ **and** $(\bigcup i.\ A\ i) = space\ M\ \bigwedge i.\ emeasure\ M\ (A\ i) \neq \infty$
using *sigma-finite-measure.sigma-finite*[*OF* ‹$p = \infty \Longrightarrow$ *sigma-finite-measure M*›[*OF* ‹$p = \infty$›]] **by** (*metis UNIV-I sets-range*)
define *Y* **where** $Y = (\lambda n{::}nat.\ \{x \in A\ n.\ |f\ x| > c\})$
have [*measurable*]: $Y\ n \in sets\ M$ **for** *n* **unfolding** *Y-def* **by** *auto*
have $\{x \in space\ M.\ |f\ x| > c\} = (\bigcup n.\ Y\ n)$ **unfolding** *Y-def* **using** ‹$(\bigcup i.\ A\ i) = space\ M$› **by** *auto*
then have *emeasure M* $(\bigcup n.\ Y\ n) > 0$ **using** $*$ **by** *auto*
then obtain *n* **where** *emeasure M* $(Y\ n) > 0$
using *emeasure-pos-unionE*[*of Y*, *OF* ‹$\bigwedge n.\ Y\ n \in sets\ M$›] **by** *auto*
have *emeasure M* $(Y\ n) \leq$ *emeasure M* $(A\ n)$ **apply** (*rule emeasure-mono*) **unfolding** *Y-def* **by** *auto*
then have *emeasure M* $(Y\ n) \neq \infty$ **using** ‹*emeasure M* $(A\ n) \neq \infty$›
by (*metis infinity-ennreal-def neq-top-trans*)
then have *measure M* $(Y\ n) > 0$ **using** ‹*emeasure M* $(Y\ n) > 0$› **unfolding** *measure-def*
by (*simp add*: *enn2real-positive-iff top.not-eq-extremum*)
have $|f\ x| \geq c$ **if** $x \in Y\ n$ **for** *x* **using** *that less-imp-le* **unfolding** *Y-def* **by** *auto*

define *g* **where** $g = (\lambda x.\ indicator\ (Y\ n)\ x * sgn(f\ x))\ /_R\ measure\ M\ (Y\ n)$
have $g \in space_N\ (\mathfrak{L}\ 1\ M)$
apply (*rule Lp-domination*[*of - - indicator* $(Y\ n)\ /_R$ *measure M* $(Y\ n)$]) **unfolding** *g-def*
using *L1-indicator'*[*OF* ‹$Y\ n \in sets\ M$› ‹*emeasure M* $(Y\ n) \neq \infty$›] **by** (*auto simp add*: *abs-mult indicator-def abs-sgn-eq*)
have *Norm* $(\mathfrak{L}\ 1\ M)\ g =$ *Norm* $(\mathfrak{L}\ 1\ M)\ (\lambda x.\ indicator\ (Y\ n)\ x * sgn(f\ x))$ / *abs*(*measure M* $(Y\ n)$)
unfolding *g-def Norm-cmult* **by** (*simp add*: *divide-inverse*)

also have ... ≤ *Norm* (𝔏 *1 M*) (*indicator* (*Y n*)) / *abs*(*measure M* (*Y n*))
using ‹*measure M* (*Y n*) > *0*› **apply** (*auto simp add*: *divide-simps*) **apply** (*rule Lp-domination*)
using *L1-indicator'*[*OF* ‹*Y n* ∈ *sets M*› ‹*emeasure M* (*Y n*) ≠ ∞›] **by** (*auto simp add*: *abs-mult indicator-def abs-sgn-eq*)
also have ... = *measure M* (*Y n*) / *abs*(*measure M* (*Y n*))
using *L1-indicator'*[*OF* ‹*Y n* ∈ *sets M*› ‹*emeasure M* (*Y n*) ≠ ∞›] **by** (*auto simp add*: *abs-mult indicator-def abs-sgn-eq*)
also have ... = *1* **using** ‹*measure M* (*Y n*) > *0*› **by** *auto*
finally have *Norm* (𝔏 *1 M*) *g* ≤ *1* **by** *simp*

have *c* ∗ *measure M* (*Y n*) = (∫ *x*. *c* ∗ *indicator* (*Y n*) *x* ∂*M*)
using ‹*measure M* (*Y n*) > *0*› ‹*emeasure M* (*Y n*) ≠ ∞› **by** *auto*
also have ... ≤ (∫ *x*. |*f x*| ∗ *indicator* (*Y n*) *x* ∂*M*)
apply (*rule integral-mono*)
using ‹*emeasure M* (*Y n*) ≠ ∞› ‹*0* < *Sigma-Algebra.measure M* (*Y n*)› *not-integrable-integral-eq* **apply** *fastforce*
apply (*rule Bochner-Integration.integrable-bound*[*of* - λ*x*. *Norm* (𝔏 ∞ *M*) *f* ∗ *indicator* (*Y n*) *x*])
using ‹*emeasure M* (*Y n*) ≠ ∞› ‹*0* < *Sigma-Algebra.measure M* (*Y n*)› *not-integrable-integral-eq* **apply** *fastforce*
using *L-infinity-AE-bound*[*OF* ‹*f* ∈ *space*$_N$ (𝔏 ∞ *M*)›] **by** (*auto simp add*: *indicator-def Y-def*)
finally have *c* ≤ (∫ *x*. |*f x*| ∗ *indicator* (*Y n*) *x* ∂*M*) / *measure M* (*Y n*)
using ‹*measure M* (*Y n*) > *0*› **by** (*auto simp add*: *divide-simps*)
also have ... = (∫ *x*. *f x* ∗ *indicator* (*Y n*) *x* ∗ *sgn*(*f x*) / *measure M* (*Y n*) ∂*M*)
using ‹*measure M* (*Y n*) > *0*› **by** (*simp add*: *abs-sgn mult.commute mult.left-commute*)
also have ... = (∫ *x*. *f x* ∗ *g x* ∂*M*)
unfolding *divide-inverse g-def divideR-apply* **by** (*auto simp add*: *algebra-simps*)
also have ... ≤ (*SUP g*∈{*g* ∈ *space*$_N$ (𝔏 *q M*). *Norm* (𝔏 *q M*) *g* ≤ *1*}. (∫ *x*. *f x* ∗ *g x* ∂*M*))
unfolding ‹*q* = *1*› **apply** (*rule cSUP-upper*, *auto*)
using ‹*g* ∈ *space*$_N$ (𝔏 *1 M*)› ‹*Norm* (𝔏 *1 M*) *g* ≤ *1*› **apply** *auto* **using** *B* ‹*p* = ∞› ‹*q* = *1*› **by** (*meson bdd-aboveI2*)
finally show *?thesis* **by** *simp*
qed
then show *?thesis* **using** *dense-le* **by** *auto*
next
case *one*
then have *q* = ∞ **using** ‹*1*/*p* + *1*/*q* = *1*› **by** *simp*
define *g* **where** *g* = (λ*x*. *sgn* (*f x*))
have [*measurable*]: *g* ∈ *space*$_N$ (𝔏 ∞ *M*)
apply (*rule L-infinity-I*[*of g M 1*]) **unfolding** *g-def* **by** (*auto simp add*: *abs-sgn-eq*)
have *Norm* (𝔏 ∞ *M*) *g* ≤ *1*
apply (*rule L-infinity-I*[*of g M 1*]) **unfolding** *g-def* **by** (*auto simp add*:

abs-sgn-eq)
have *Norm* ($\mathfrak{L}$ *p M*) *f* = (∫ *x*. |*f x*| ∂*M*)
unfolding ‹*p* = *1*› **apply** (*rule L1-D*(*3*)) **using** ‹*f* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)› **unfolding** ‹*p* = *1*› **by** *auto*
also have ... = (∫ *x*. *f x* ∗ *g x* ∂*M*)
unfolding *g-def* **by** (*simp add*: *abs-sgn*)
also have ... ≤ (*SUP g*∈{*g* ∈ *space*$_N$ ($\mathfrak{L}$ *q M*). *Norm* ($\mathfrak{L}$ *q M*) *g* ≤ *1*}. (∫ *x*. *f x* ∗ *g x* ∂*M*))
unfolding ‹*q* = ∞› **apply** (*rule cSUP-upper*, *auto*)
using ‹*g* ∈ *space*$_N$ ($\mathfrak{L}$ ∞ *M*)› ‹*Norm* ($\mathfrak{L}$ ∞ *M*) *g* ≤ *1*› **apply** *auto*
using *B* ‹*q* = ∞› **by** *fastforce*
finally show *?thesis* **by** *simp*
next
case (*gr p2*)
then have *p2* > *0* **by** *simp*
have *f* ∈ *space*$_N$ ($\mathfrak{L}$ *p2 M*) **using** ‹*f* ∈ *space*$_N$ ($\mathfrak{L}$ *p M*)› ‹*p* = *ennreal p2*› **by** *auto*
define *q2* **where** *q2* = *conjugate-exponent p2*
have *q2* > *1 q2* > *0* **using** *conjugate-exponent-real*(*2*)[*OF* ‹*p2* > *1*›] **unfolding** *q2-def* **by** *auto*
have *q* = *ennreal q2*
unfolding *q2-def conjugate-exponent-real-ennreal*[*OF* ‹*p2* > *1*›, *symmetric*] ‹*p* = *ennreal p2*›[*symmetric*]
using *conjugate-exponent-ennreal-iff*[*OF* ‹*p* ≥ *1*›] ‹*1*/*p* + *1*/*q* = *1*› **by** *auto*

show *?thesis*
proof (*cases Norm* ($\mathfrak{L}$ *p M*) *f* = *0*)
case *True*
then have *Norm* ($\mathfrak{L}$ *p M*) *f* ≤ (∫ *x*. *f x* ∗ *0 x* ∂*M*) **by** *auto*
also have ... ≤ (*SUP g*∈{*g* ∈ *space*$_N$ ($\mathfrak{L}$ *q M*). *Norm* ($\mathfrak{L}$ *q M*) *g* ≤ *1*}. (∫ *x*. *f x* ∗ *g x* ∂*M*))
apply (*rule cSUP-upper*, *auto simp add*: *zero-fun-def*[*symmetric*]) **using** *B* **by** (*meson bdd-aboveI2*)
finally show *?thesis* **by** *simp*
next
case *False*
then have *Norm* ($\mathfrak{L}$ *p2 M*) *f* > *0*
unfolding ‹*p* = *ennreal p2*› **using** *Norm-nonneg*[*of* $\mathfrak{L}$ *p2 M f*] **by** *linarith*

define *h* **where** *h* = (λ*x*. *sgn*(*f x*) ∗ |*f x*| *powr* (*p2* − *1*))
have [*measurable*]: *h* ∈ *borel-measurable M* **unfolding** *h-def* **by** *auto*
have (∫$^+$*x*. |*h x*| *powr q2* ∂*M*) = (∫$^+$*x*. (|*f x*| *powr* (*p2* − *1*)) *powr q2* ∂*M*)
unfolding *h-def* **by** (*rule nn-integral-cong*, *auto simp add*: *abs-mult abs-sgn-eq*)
also have ... = (∫$^+$*x*. |*f x*| *powr p2* ∂*M*)
unfolding *powr-powr q2-def* **using** *conjugate-exponent-real*(*4*)[*OF* ‹*p2* > *1*›] **by** *auto*
also have ... = (*Norm* ($\mathfrak{L}$ *p2 M*) *f*) *powr p2*
apply (*subst Lp-Norm*(*2*), *auto simp add*: ‹*p2* > *0*›)

by (*rule nn-integral-eq-integral*, *auto simp add*: *Lp-D*(*2*)[*OF* ‹*p2* > *0*› ‹*f* ∈ $space_N$ (𝔏 *p2 M*)›])
finally have ∗: ($\int^+ x$. |*h x*| *powr q2* ∂*M*) = (*Norm* (𝔏 *p2 M*) *f*) *powr p2*
by *simp*
have *integrable M* (λ*x*. |*h x*| *powr q2*)
apply (*rule integrableI-bounded*, *auto*) **using** ∗ **by** *auto*
then have ($\int$ *x*. |*h x*| *powr q2* ∂*M*) = ($\int^+ x$. |*h x*| *powr q2* ∂*M*)
by (*rule nn-integral-eq-integral*[*symmetric*], *auto*)
then have ∗∗: ($\int$ *x*. |*h x*| *powr q2* ∂*M*) = (*Norm* (𝔏 *p2 M*) *f*) *powr p2*
using ∗ **by** *auto*

define *g* **where** *g* = (λ*x*. *h x* / (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* / *q2*))
have [*measurable*]: *g* ∈ *borel-measurable M* **unfolding** *g-def* **by** *auto*
have *intg*: *integrable M* (λ*x*. |*g x*| *powr q2*)
unfolding *g-def* **using** ‹*Norm* (𝔏 *p2 M*) *f* > *0*› ‹*q2* > *1*› **apply** (*simp add*: *abs-mult powr-divide powr-powr*)
using ‹*integrable M* (λ*x*. |*h x*| *powr q2*)› *integrable-divide-zero* **by** *blast*
have *g* ∈ $space_N$ (𝔏 *q2 M*) **by** (*rule Lp-I*(*1*)[*OF* ‹*q2* > *0*› - *intg*], *auto*)
have ($\int$ *x*. |*g x*| *powr q2* ∂*M*) = *1*
unfolding *g-def* **using** ‹*Norm* (𝔏 *p2 M*) *f* > *0*› ‹*q2* > *1*› **by** (*simp add*: *abs-mult powr-divide powr-powr* ∗∗)
then have *Norm* (𝔏 *q2 M*) *g* = *1*
apply (*subst Lp-D*[*OF* ‹*q2* > *0*›]) **using** ‹*g* ∈ $space_N$ (𝔏 *q2 M*)› **by** *auto*

have ($\int$ *x*. *f x* ∗ *g x* ∂*M*) = ($\int$ *x*. *f x* ∗ *sgn*(*f x*) ∗ |*f x*| *powr* (*p2* − *1*) / (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* / *q2*) ∂*M*)
unfolding *g-def h-def* **by** (*simp add*: *mult.assoc*)
also have ... = ($\int$ *x*. |*f x*| ∗ |*f x*| *powr* (*p2*−*1*) ∂*M*) / (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* / *q2*)
by (*auto simp add*: *abs-sgn*)
also have ... = ($\int$ *x*. |*f x*| *powr p2* ∂*M*) / (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* / *q2*)
by (*subst powr-mult-base*, *auto*)
also have ... = (*Norm* (𝔏 *p2 M*) *f*) *powr p2* / (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* / *q2*)
by (*subst Lp-Norm*(*2*)[*OF* ‹*p2* > *0*›], *auto*)
also have ... = (*Norm* (𝔏 *p2 M*) *f*) *powr* (*p2* − *p2*/*q2*)
by (*simp add*: *powr-diff* [*symmetric*])
also have ... = *Norm* (𝔏 *p2 M*) *f*
unfolding *q2-def* **using** *conjugate-exponent-real*(*5*)[*OF* ‹*p2* > *1*›] **by** *auto*
finally have *Norm* (𝔏 *p M*) *f* = ($\int$ *x*. *f x* ∗ *g x* ∂*M*)
unfolding ‹*p* = *ennreal p2*› **by** *simp*
also have ... ≤ (*SUP g*∈{*g* ∈ $space_N$ (𝔏 *q M*). *Norm* (𝔏 *q M*) *g* ≤ *1*}. ($\int$ *x*. *f x* ∗ *g x* ∂*M*))
unfolding ‹*q* = *ennreal q2*› **apply** (*rule cSUP-upper*, *auto*)
using ‹*g* ∈ $space_N$ (𝔏 *q2 M*)› ‹*Norm* (𝔏 *q2 M*) *g* = *1*› **apply** *auto*
using *B* ‹*q* = *ennreal q2*› **by** *fastforce*
finally show *?thesis* **by** *simp*
qed

qed
qed
qed

The previous theorem admits a version in which one does not assume a priori that the function under consideration belongs to L^p. This gives an efficient criterion to check if a function is indeed in L^p. In this case, it is always necessary to assume that the measure is sigma-finite.

Note that, in the statement, the Bochner integral $\int fg$ vanishes by definition if fg is not integrable. Hence, the statement really says that the eNorm can be estimated using functions g for which fg is integrable. It is precisely the construction of such functions g that requires the space to be sigma-finite.

theorem *Lp-Lq-duality'*:
fixes *p q*::*ennreal*
assumes *1/p + 1/q = 1*
sigma-finite-measure M
and [*measurable*]: *f* ∈ *borel-measurable M*
shows *eNorm* (𝔏 *p M*) *f* = (*SUP g*∈{*g* ∈ *space*$_N$ (𝔏 *q M*). *Norm* (𝔏 *q M*) *g* ≤ *1*}. *ennreal*(∫ *x*. *f x* ∗ *g x* ∂*M*))
proof (*cases eNorm* (𝔏 *p M*) *f* ≠ ∞)
case *True*
then have *f* ∈ *space*$_N$ (𝔏 *p M*) **unfolding** *space*$_N$*-def* **by** (*simp add*: *top.not-eq-extremum*)
show *?thesis*
unfolding *eNorm-Norm*[*OF* ‹*f* ∈ *space*$_N$ (𝔏 *p M*)›] *Lp-Lq-duality*[*OF* ‹*f* ∈ *space*$_N$ (𝔏 *p M*)› ‹*1/p + 1/q = 1*› ‹*sigma-finite-measure M*›]
apply (*rule SUP-real-ennreal*[*symmetric*], *auto*, *rule exI*[*of - 0*], *auto*)
by (*rule Lp-Lq-duality*[*OF* ‹*f* ∈ *space*$_N$ (𝔏 *p M*)› ‹*1/p + 1/q = 1*› ‹*sigma-finite-measure M*›])
next
case *False*
have *B*: ∃ *g* ∈ {*g* ∈ *space*$_N$ (𝔏 *q M*). *Norm* (𝔏 *q M*) *g* ≤ *1*}. (∫ *x*. *f x* ∗ *g x* ∂*M*) ≥ *C* **if** *C* < ∞ **for** *C*::*ennreal*
proof −
obtain *Cr* **where** *C* = *ennreal Cr Cr* ≥ *0* **using** ‹*C* < ∞› *ennreal-cases less-irrefl* **by** *auto*
obtain *A* **where** *A*: ⋀*n*::*nat*. *A n* ∈ *sets M incseq A* (⋃ *n*. *A n*) = *space M*
⋀*n*. *emeasure M* (*A n*) ≠ ∞
using *sigma-finite-measure.sigma-finite-incseq*[*OF* ‹*sigma-finite-measure M*›] **by** (*metis range-subsetD*)
define *Y* **where** *Y* = (λ*n*. {*x* ∈ *A n*. |*f x*| ≤ *n*})
have [*measurable*]: ⋀*n*. *Y n* ∈ *sets M* **unfolding** *Y-def* **using** ‹⋀*n*::*nat*. *A n* ∈ *sets M*› **by** *auto*
have *incseq Y*
apply (*rule incseq-SucI*) **unfolding** *Y-def* **using** *incseq-SucD*[*OF* ‹*incseq A*›] **by** *auto*
have ∗: ∃ *N*. ∀ *n* ≥ *N*. *f x* ∗ *indicator* (*Y n*) *x* = *f x* **if** *x* ∈ *space M* **for** *x*
proof −
obtain *n0* **where** *n0*: *x* ∈ *A n0* **using** ‹*x* ∈ *space M*› ‹(⋃ *n*. *A n*) = *space*

M› **by** *auto*
obtain *n1*::*nat* **where** *n1*: $|f\ x| \leq$ *n1* **using** *real-arch-simple* **by** *blast*
have $x \in Y$ (*max n0 n1*)
unfolding *Y-def* **using** *n1* **apply** *auto*
using *n0* ‹*incseq A*› *incseq-def max.cobounded1* **by** *blast*
then have $*$: $x \in Y\ n$ **if** $n \geq$ *max n0 n1* **for** *n*
using ‹*incseq Y*› *that incseq-def* **by** *blast*
show *?thesis* **by** (*rule exI*[*of - max n0 n1*], *auto simp add*: $*$)
qed
have $*$: $(\lambda n.\ f\ x * \textit{indicator}\ (Y\ n)\ x) \longrightarrow f\ x$ **if** $x \in$ *space M* **for** *x*
using $*$[*OF that*] **unfolding** *eventually-sequentially*[*symmetric*] **by** (*simp add*: *tendsto-eventually*)
have *liminf* $(\lambda n.$ *eNorm* $(\mathfrak{L}\ p\ M)\ (\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x)) \geq$ *eNorm* $(\mathfrak{L}\ p\ M)\ f$
apply (*rule Lp-AE-limit*) **using** $*$ **by** *auto*
then have *liminf* $(\lambda n.$ *eNorm* $(\mathfrak{L}\ p\ M)\ (\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x)) > Cr$ **using** *False neq-top-trans* **by** *force*
then have *limsup* $(\lambda n.$ *eNorm* $(\mathfrak{L}\ p\ M)\ (\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x)) > Cr$
using *Liminf-le-Limsup less-le-trans trivial-limit-sequentially* **by** *blast*
then obtain *n* **where** *n*: *eNorm* $(\mathfrak{L}\ p\ M)\ (\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) > Cr$
using *Limsup-obtain* **by** *blast*

have $(\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) \in \textit{space}_N\ (\mathfrak{L}\ p\ M)$
apply (*rule Lp-bounded-bounded-support*[*of - - n*], *auto*)
unfolding *Y-def indicator-def* **apply** *auto*
by (*metis* (*mono-tags*, *lifting*) *A*(*1*) *A*(*4*) *emeasure-mono infinity-ennreal-def mem-Collect-eq neq-top-trans subsetI*)
have *Norm* $(\mathfrak{L}\ p\ M)\ (\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) > Cr$
using *n* **unfolding** *eNorm-Norm*[*OF* ‹$(\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) \in \textit{space}_N\ (\mathfrak{L}\ p\ M)$›]
by (*meson ennreal-leI not-le*)
then have $(SUP\ g \in \{g \in \textit{space}_N\ (\mathfrak{L}\ q\ M).\ \textit{Norm}\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * \textit{indicator}\ (Y\ n)\ x * g\ x\ \partial M)) > Cr$
using *Lp-Lq-duality*(*2*)[*OF* ‹$(\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) \in \textit{space}_N\ (\mathfrak{L}\ p\ M)$› ‹$1/p + 1/q = 1$› ‹*sigma-finite-measure M*›]
by *auto*
then have $\exists g \in \{g \in \textit{space}_N\ (\mathfrak{L}\ q\ M).\ \textit{Norm}\ (\mathfrak{L}\ q\ M)\ g \leq 1\}.\ (\int x.\ f\ x * \textit{indicator}\ (Y\ n)\ x * g\ x\ \partial M) > Cr$
apply (*subst less-cSUP-iff*[*symmetric*])
using *Lp-Lq-duality*(*1*)[*OF* ‹$(\lambda x.\ f\ x * \textit{indicator}\ (Y\ n)\ x) \in \textit{space}_N\ (\mathfrak{L}\ p\ M)$› ‹$1/p + 1/q = 1$› ‹*sigma-finite-measure M*›] **apply** *auto*
by (*rule exI*[*of - 0*], *auto*)
then obtain *g* **where** *g*: $g \in \textit{space}_N\ (\mathfrak{L}\ q\ M)$ $\textit{Norm}\ (\mathfrak{L}\ q\ M)\ g \leq 1$ $(\int x.\ f\ x * \textit{indicator}\ (Y\ n)\ x * g\ x\ \partial M) > Cr$
by *auto*
then have [*measurable*]: $g \in$ *borel-measurable M* **using** *Lp-measurable* **by** *auto*
define *h* **where** $h = (\lambda x.\ \textit{indicator}\ (Y\ n)\ x * g\ x)$
have *Norm* $(\mathfrak{L}\ q\ M)\ h \leq \textit{Norm}\ (\mathfrak{L}\ q\ M)\ g$
apply (*rule Lp-domination*[*of - - g*]) **unfolding** *h-def indicator-def* **using** ‹*g*

$\in$ *space*$_N$ ($\mathfrak{L}$ *q M*)› **by** *auto*
then have *a*: *Norm* ($\mathfrak{L}$ *q M*) *h* $\leq$ *1* **using** ‹*Norm* ($\mathfrak{L}$ *q M*) *g* $\leq$ *1*› **by** *auto*
have *b*: *h* $\in$ *space*$_N$ ($\mathfrak{L}$ *q M*)
apply (*rule Lp-domination*[*of* - - *g*]) **unfolding** *h-def indicator-def* **using** ‹*g* $\in$ *space*$_N$ ($\mathfrak{L}$ *q M*)› **by** *auto*
have ($\int$ *x*. *f x* $*$ *h x* ∂*M*) > *Cr* **unfolding** *h-def* **using** *g*(*3*) **by** (*auto simp add*: *mult.assoc*)
then have ($\int$ *x*. *f x* $*$ *h x* ∂*M*) > *C*
unfolding ‹*C* = *ennreal Cr*› **using** ‹*Cr* $\geq$ *0*› **by** (*simp add*: *ennreal-less-iff*)
then show *?thesis* **using** *a b* **by** *auto*
qed
have (*SUP g*$\in${*g* $\in$ *space*$_N$ ($\mathfrak{L}$ *q M*). *Norm* ($\mathfrak{L}$ *q M*) *g* $\leq$ *1*}. *ennreal*($\int$ *x*. *f x* $*$ *g x* ∂*M*)) $\geq$ ∞
apply (*rule dense-le*) **using** *B* **by** (*meson SUP-upper2*)
then show *?thesis* **using** *False neq-top-trans* **by** *force*
qed

5.9 Conditional expectations and L^p

The L^p space with respect to a subalgebra is included in the whole L^p space.

lemma *Lp-subalgebra*:
assumes *subalgebra M F*
shows $\bigwedge$*f*. *eNorm* ($\mathfrak{L}$ *p M*) *f* $\leq$ *eNorm* ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) *f*
($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) $\subseteq_N$ $\mathfrak{L}$ *p M*
space$_N$ (($\mathfrak{L}$ *p* (*restr-to-subalg M F*))) $\subseteq$ *space*$_N$ ($\mathfrak{L}$ *p M*)
$\bigwedge$*f*. *f* $\in$ *space*$_N$ (($\mathfrak{L}$ *p* (*restr-to-subalg M F*))) $\Longrightarrow$ *Norm* ($\mathfrak{L}$ *p M*) *f* = *Norm* ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) *f*
proof −
have $*$: *f* $\in$ *space*$_N$ ($\mathfrak{L}$ *p M*) $\wedge$ *Norm* ($\mathfrak{L}$ *p M*) *f* = *Norm* ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) *f*
if *f* $\in$ *space*$_N$ ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) **for** *f*
proof −
have [*measurable*]: *f* $\in$ *borel-measurable* (*restr-to-subalg M F*) **using** *that Lp-measurable* **by** *auto*
then have [*measurable*]: *f* $\in$ *borel-measurable M*
using *assms measurable-from-subalg measurable-in-subalg*′ **by** *blast*
show *?thesis*
proof (*cases rule*: *Lp-cases*[*of p*])
case *zero*
then show *?thesis* **using** *that* **unfolding** ‹*p* = *0*› *L-zero-space Norm-def L-zero* **by** *auto*
next
case *PInf*
have [*measurable*]: *f* $\in$ *borel-measurable* (*restr-to-subalg M F*) **using** *that Lp-measurable* **by** *auto*
then have [*measurable*]: *f* $\in$ *borel-measurable F* **using** *assms measurable-in-subalg*′ **by** *blast*
then have [*measurable*]: *f* $\in$ *borel-measurable M* **using** *assms measurable-from-subalg* **by** *blast*

```
    have AE x in (restr-to-subalg M F). |f x| ≤ Norm (𝔏 ∞ (restr-to-subalg M
F)) f
      using L-infinity-AE-bound that unfolding ‹p = ∞› by auto
    then have a: AE x in M. |f x| ≤ Norm (𝔏 ∞ (restr-to-subalg M F)) f
      using assms AE-restr-to-subalg by blast
   have ∗: f ∈ space_N (𝔏 ∞ M) Norm (𝔏 ∞ M) f ≤ Norm (𝔏 ∞ (restr-to-subalg
M F)) f
      using L-infinity-I[OF ‹f ∈ borel-measurable M› a] by auto
    then have b: AE x in M. |f x| ≤ Norm (𝔏 ∞ M) f
      using L-infinity-AE-bound by auto
    have c: AE x in (restr-to-subalg M F). |f x| ≤ Norm (𝔏 ∞ M) f
      apply (rule AE-restr-to-subalg2[OF assms]) using b by auto
    have Norm (𝔏 ∞ (restr-to-subalg M F)) f ≤ Norm (𝔏 ∞ M) f
      using L-infinity-I[OF ‹f ∈ borel-measurable (restr-to-subalg M F)› c] by
auto
    then show ?thesis using ∗ unfolding ‹p = ∞› by auto
  next
    case (real-pos p2)
    then have a [measurable]: f ∈ space_N (𝔏 p2 (restr-to-subalg M F))
      using that unfolding ‹p = ennreal p2› by auto
    then have b [measurable]: f ∈ space_N (𝔏 p2 M)
      unfolding Lp-space[OF ‹p2 > 0›] using integrable-from-subalg[OF assms]
by auto
    show ?thesis
      unfolding ‹p = ennreal p2› Lp-D[OF ‹p2 > 0› a] Lp-D[OF ‹p2 > 0› b]
       using integral-subalgebra2[OF assms, symmetric, of f] apply (auto simp
add: b)
       by (metis (mono-tags, lifting) ‹integrable (restr-to-subalg M F) (λx. |f x|
powr p2)› assms integrableD(1) integral-subalgebra2 measurable-in-subalg')
  qed
 qed
 show space_N ((𝔏 p (restr-to-subalg M F))) ⊆ space_N (𝔏 p M) using ∗ by auto
 show Norm (𝔏 p M) f = Norm (𝔏 p (restr-to-subalg M F)) f if f ∈ space_N ((𝔏
p (restr-to-subalg M F))) for f
  using ∗ that by auto
 show eNorm (𝔏 p M) f ≤ eNorm (𝔏 p (restr-to-subalg M F)) f for f
  by (metis ∗ eNorm-Norm eq-iff infinity-ennreal-def less-imp-le spaceN-iff top.not-eq-extremum)
 then show (𝔏 p (restr-to-subalg M F)) ⊆_N 𝔏 p M
  by (metis ennreal-1 mult.left-neutral quasinorm-subsetI)
qed
```

For $p \geq 1$, the conditional expectation of an L^p function still belongs to L^p, with an L^p norm which is bounded by the norm of the original function. This is wrong for $p < 1$. One can prove this separating the cases and using the conditional version of Jensen's inequality, but it is much more efficient to do it with duality arguments, as follows.

```
proposition Lp-real-cond-exp:
 assumes [simp]: subalgebra M F
    and p ≥ (1::ennreal)
```

sigma-finite-measure (*restr-to-subalg M F*)
f ∈ *space*$_N$ (𝔏 *p M*)
shows *real-cond-exp M F f* ∈ *space*$_N$ (𝔏 *p* (*restr-to-subalg M F*))
Norm (𝔏 *p* (*restr-to-subalg M F*)) (*real-cond-exp M F f*) ≤ *Norm* (𝔏 *p M*) *f*
proof −
have [*measurable*]: *f* ∈ *borel-measurable M* **using** *Lp-measurable assms* **by** *auto*
define *q* **where** *q* = *conjugate-exponent p*
have *1/p + 1/q = 1* **unfolding** *q-def* **using** *conjugate-exponent-ennreal*[*OF* ‹*p* ≥ *1*›] **by** *simp*
have *eNorm* (𝔏 *p* (*restr-to-subalg M F*)) (*real-cond-exp M F f*)
= (*SUP g*∈{*g* ∈ *space*$_N$ (𝔏 *q* (*restr-to-subalg M F*)). *Norm* (𝔏 *q* (*restr-to-subalg M F*)) *g* ≤ *1*}. *ennreal*(∫ *x*. (*real-cond-exp M F f*) *x* ∗ *g x* ∂(*restr-to-subalg M F*)))
by (*rule Lp-Lq-duality'*[*OF* ‹*1/p + 1/q = 1*› ‹*sigma-finite-measure* (*restr-to-subalg M F*)›], *simp*)
also have ... ≤ (*SUP g*∈{*g* ∈ *space*$_N$ (𝔏 *q M*). *Norm* (𝔏 *q M*) *g* ≤ *1*}. *ennreal*(∫ *x*. *f x* ∗ *g x* ∂*M*))
proof (*rule SUP-mono*, *auto*)
fix *g* **assume** *H*: *g* ∈ *space*$_N$ (𝔏 *q* (*restr-to-subalg M F*))
Norm (𝔏 *q* (*restr-to-subalg M F*)) *g* ≤ *1*
then have *H2*: *g* ∈ *space*$_N$ (𝔏 *q M*) *Norm* (𝔏 *q M*) *g* ≤ *1*
using *Lp-subalgebra*[*OF* ‹*subalgebra M F*›] **by** (*auto simp add*: *subset-iff*)
have [*measurable*]: *g* ∈ *borel-measurable M g* ∈ *borel-measurable F*
using *Lp-measurable*[*OF H(1)*] *Lp-measurable*[*OF H2(1)*] **by** *auto*
have *int*: *integrable M* (λ*x*. *f x* ∗ *g x*)
using *Lp-Lq-duality-bound*(*1*)[*OF* ‹*1/p + 1/q = 1*› ‹*f* ∈ *space*$_N$ (𝔏 *p M*)› *H2*(*1*)].
have (∫ *x*. (*real-cond-exp M F f*) *x* ∗ *g x* ∂(*restr-to-subalg M F*)) = (∫ *x*. *g x* ∗ (*real-cond-exp M F f*) *x* ∂*M*)
by (*subst mult.commute*, *rule integral-subalgebra2*[*OF* ‹*subalgebra M F*›], *auto*)
also have ... = (∫ *x*. *g x* ∗ *f x* ∂*M*)
apply (*rule sigma-finite-subalgebra.real-cond-exp-intg*, *auto simp add*: *int mult.commute*)
unfolding *sigma-finite-subalgebra-def* **using** *assms* **by** *auto*
finally have *ennreal* (∫ *x*. (*real-cond-exp M F f*) *x* ∗ *g x* ∂(*restr-to-subalg M F*)) ≤ *ennreal* (∫ *x*. *f x* ∗ *g x* ∂*M*)
by (*auto intro*!: *ennreal-leI simp add*: *mult.commute*)
then show ∃ *m*. *m* ∈ *space*$_N$ (𝔏 *q M*) ∧ *Norm* (𝔏 *q M*) *m* ≤ *1*
∧ *ennreal* (*LINT x*|*restr-to-subalg M F*. *real-cond-exp M F f x* ∗ *g x*) ≤ *ennreal* (*LINT x*|*M*. *f x* ∗ *m x*)
using *H2* **by** *blast*
qed
also have ... = *eNorm* (𝔏 *p M*) *f*
apply (*rule Lp-Lq-duality'*[*OF* ‹*1/p + 1/q = 1*›, *symmetric*], *auto intro*!: *sigma-finite-subalgebra-is-sigma-finite*[*of - F*])
unfolding *sigma-finite-subalgebra-def* **using** *assms* **by** *auto*
finally have ∗: *eNorm* (𝔏 *p* (*restr-to-subalg M F*)) (*real-cond-exp M F f*) ≤ *eNorm* (𝔏 *p M*) *f*
by *simp*
then show *a*: *real-cond-exp M F f* ∈ *space*$_N$ (𝔏 *p* (*restr-to-subalg M F*))

by (*meson* ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$› *order-le-less-trans spaceN-iff*)
show *Norm* ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) (*real-cond-exp M F f*) ≤ *Norm* ($\mathfrak{L}$ *p M*) *f*
using ∗ **unfolding** *eNorm-Norm*[*OF* ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$›] *eNorm-Norm*[*OF a*] **by** *simp*
qed

lemma *Lp-real-cond-exp-eNorm*:
assumes [*simp*]: *subalgebra M F*
and *p* ≥ (*1*::*ennreal*)
sigma-finite-measure (*restr-to-subalg M F*)
shows *eNorm* ($\mathfrak{L}$ *p* (*restr-to-subalg M F*)) (*real-cond-exp M F f*) ≤ *eNorm* ($\mathfrak{L}$ *p M*) *f*
proof (*cases eNorm* ($\mathfrak{L}$ *p M*) *f* = ∞)
case *False*
then have ∗: *f* ∈ $space_N$ ($\mathfrak{L}$ *p M*)
unfolding *spaceN-iff* **by** (*simp add*: *top.not-eq-extremum*)
show *?thesis*
using *Lp-real-cond-exp*[*OF assms* ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$›] **by** (*subst eNorm-Norm*, *auto simp*: ‹$f \in space_N\ (\mathfrak{L}\ p\ M)$›)+
qed (*simp*)

end