The Hahn and Jordan Decomposition Theorems

Marie Cousin   Mnacho Echenim   Hervé Guiol

March 17, 2025

Contents

1 Introduction

Signed measures are a generalization of measures that can map measurable sets to negative values. In this work we formalize the Hahn decomposition theorem for signed measures, namely that if $(\Omega, \mathcal{A}, \mu)$ is a measure space for a signed measure μ, then Ω can be decomposed as $\Omega^+ \cup \Omega^-$, where every measurable subset of Ω^+ has a positive measure, and every measurable subset of Ω^- has a negative measure. We then prove that this decomposition is essentially unique, meaning that if $X^+ \cup X^-$ is another such decomposition, then any measurable subset in $(\Omega^+ \triangle X^+) \cup (\Omega^- \triangle X^-)$ has a zero measure.

We also formalize the Jordan decomposition theorem as a corollary, which states that the signed measure μ admits a unique decomposition into a difference $\mu = \mu^+ - \mu^-$ of two positive measures, at least one of which is finite, and such that for any Hahn decomposition $\Omega^+ \cup \Omega^-$ and measurable set A, if $A \subseteq \Omega^-$ then $\mu^+(A) = 0$ and if $A \subseteq \Omega^+$ then $\mu^-(A) = 0$. The formalization is mostly based on [1], Section 16 of Chapter 4.

2 Signed measures

In this section we define signed measures. These are generalizations of measures that can also take negative values but cannot contain both ∞ and $-\infty$ in their range.

2.1 Basic definitions

theory *Hahn-Jordan-Decomposition* **imports**
HOL−Probability.Probability
Hahn-Jordan-Prelims
begin

definition *signed-measure*::$'a$ *measure* $\Rightarrow$ ($'a$ *set* $\Rightarrow$ *ereal*) $\Rightarrow$ *bool* **where**
$signed\text{-}measure\ M\ \mu \longleftrightarrow \mu\ \{\} = 0 \wedge (-\infty \notin range\ \mu \vee \infty \notin range\ \mu) \wedge$
$(\forall A.\ range\ A \subseteq sets\ M \longrightarrow disjoint\text{-}family\ A \longrightarrow \bigcup\ (range\ A) \in sets\ M \longrightarrow$
$(\lambda i.\ \mu\ (A\ i))\ sums\ \mu\ (\bigcup\ (range\ A))) \wedge$
$(\forall A.\ range\ A \subseteq sets\ M \longrightarrow disjoint\text{-}family\ A \longrightarrow \bigcup\ (range\ A) \in sets\ M \longrightarrow$
$|\mu\ (\bigcup\ (range\ A))| < \infty \longrightarrow summable\ (\lambda i.\ real\text{-}of\text{-}ereal\ |\mu\ (A\ i)|))$

lemma *signed-measure-empty*:
assumes *signed-measure* $M\ \mu$
shows $\mu\ \{\} = 0$ **using** *assms* **unfolding** *signed-measure-def* **by** *simp*

lemma *signed-measure-sums*:
assumes *signed-measure* $M\ \mu$
and $range\ A \subseteq M$
and *disjoint-family* A
and $\bigcup\ (range\ A) \in sets\ M$
shows $(\lambda i.\ \mu\ (A\ i))\ sums\ \mu\ (\bigcup\ (range\ A))$
using *assms* **unfolding** *signed-measure-def* **by** *simp*

lemma *signed-measure-summable*:
assumes *signed-measure* $M\ \mu$
and $range\ A \subseteq M$
and *disjoint-family* A
and $\bigcup\ (range\ A) \in sets\ M$
and $|\mu\ (\bigcup\ (range\ A))| < \infty$
shows $summable\ (\lambda i.\ real\text{-}of\text{-}ereal\ |\mu\ (A\ i)|)$
using *assms* **unfolding** *signed-measure-def* **by** *simp*

lemma *signed-measure-inf-sum*:
assumes *signed-measure* $M\ \mu$
and $range\ A \subseteq M$
and *disjoint-family* A
and $\bigcup\ (range\ A) \in sets\ M$
shows $(\sum i.\ \mu\ (A\ i)) = \mu\ (\bigcup\ (range\ A))$ **using** *sums-unique assms*
signed-measure-sums **by** (*metis*)

lemma *signed-measure-abs-convergent*:
  assumes *signed-measure M μ*
    and *range A* ⊆ *sets M*
    and *disjoint-family A*
    and ⋃ (*range A*) ∈ *sets M*
    and |μ (⋃ (*range A*))| < ∞
  shows *summable* (λ*i*. *real-of-ereal* |μ (*A i*)|) **using** *assms*
  unfolding *signed-measure-def* **by** *simp*

lemma *signed-measure-additive*:
  assumes *signed-measure M μ*
  shows *additive M μ*
proof (*auto simp add*: *additive-def*)
  fix *x y*
  assume *x*: *x* ∈ *M* **and** *y*: *y* ∈ *M* **and** *x* ∩ *y* = {}
  hence *disjoint-family* (*binaryset x y*)
    by (*auto simp add*: *disjoint-family-on-def binaryset-def*)
  have (λ*i*. μ ((*binaryset x y*) *i*)) *sums* (μ *x* + μ *y*) **using** *binaryset-sums*
      signed-measure-empty[*of M μ*] *assms* **by** *simp*
  have *range* (*binaryset x y*) = {*x*, *y*, {}} **using** *range-binaryset-eq* **by** *simp*
  moreover have {*x*, *y*, {}} ⊆ *M* **using** *x y* **by** *auto*
  moreover have *x*∪*y* ∈ *sets M* **using** *x y* **by** *simp*
  moreover have (⋃ (*range* (*binaryset x y*))) = *x*∪ *y*
    by (*simp add*: *calculation*(*1*))
  ultimately have (λ*i*. μ ((*binaryset x y*) *i*)) *sums* μ (*x* ∪ *y*) **using** *assms x y*
      signed-measure-empty[*of M μ*] *signed-measure-sums*[*of M μ*]
      ‹*disjoint-family* (*binaryset x y*)› **by** (*metis*)
  then show μ (*x* ∪ *y*) = μ *x* + μ *y*
    using ‹(λ*i*. μ ((*binaryset x y*) *i*)) *sums* (μ *x* + μ *y*)› *sums-unique2* **by** *force*
qed

lemma *signed-measure-add*:
  assumes *signed-measure M μ*
    and *a*∈ *sets M*
    and *b*∈ *sets M*
    and *a*∩ *b* = {}
  shows μ (*a*∪ *b*) = μ *a* + μ *b* **using** *additiveD*[*OF signed-measure-additive*]
    assms **by** *auto*

lemma *signed-measure-disj-sum*:
  shows *finite I* ⟹ *signed-measure M μ* ⟹ *disjoint-family-on A I* ⟹
    (⋀*i*. *i* ∈ *I* ⟹ *A i* ∈ *sets M*) ⟹ μ (⋃ *i*∈ *I*. *A i*) = (∑ *i*∈ *I*. μ (*A i*))
proof (*induct rule*:*finite-induct*)
  case *empty*
  then show *?case* **unfolding** *signed-measure-def* **by** *simp*
next
  case (*insert x F*)
  have μ (⋃ (*A* ‘ *insert x F*)) = μ ((⋃ (*A* ‘*F*)) ∪ *A x*)
    by (*simp add*: *Un-commute*)

```
  also have ... = μ (⋃ (A ‘F)) + μ (A x)
  proof –
    have (⋃ (A ‘F)) ∩ (A x) = {} using insert
      by (metis disjoint-family-on-insert inf-commute)
    moreover have ⋃ (A ‘F) ∈ sets M using insert by auto
    moreover have A x ∈ sets M using insert by simp
    ultimately show ?thesis by (meson insert.prems(1) signed-measure-add)
  qed
  also have ... = (∑ i∈ F. μ (A i)) + μ (A x) using insert
    by (metis disjoint-family-on-insert insert-iff)
  also have ... = (∑ i∈insert x F. μ (A i))
    by (simp add: add.commute insert.hyps(1) insert.hyps(2))
  finally show ?case .
qed

lemma pos-signed-measure-count-additive:
  assumes signed-measure M μ
    and ∀ E ∈ sets M. 0 ≤ μ E
  shows countably-additive (sets M) (λA. e2ennreal (μ A))
  unfolding countably-additive-def
proof (intro allI impI)
  fix A::nat ⇒ 'a set
  assume range A ⊆ sets M
    and disjoint-family A
    and ⋃ (range A) ∈ sets M note Aprops = this
  have eq: ⋀i. μ (A i) = enn2ereal (e2ennreal (μ (A i)))
    using assms enn2ereal-e2ennreal Aprops by simp
  have (λn. ∑ i≤n. μ (A i)) ⟶ μ (⋃ (range A)) using
      sums-def-le[of λi. μ (A i) μ (⋃ (range A))] assms
      signed-measure-sums[of M] Aprops by simp
  hence ((λn. e2ennreal (∑ i≤n. μ (A i))) ⟶
      e2ennreal (μ (⋃ (range A)))) sequentially
    using tendsto-e2ennrealI[of (λn. ∑ i≤n. μ (A i)) μ (⋃ (range A))]
    by simp
  moreover have ⋀n. e2ennreal (∑ i≤n. μ (A i)) = (∑ i≤n. e2ennreal (μ (A
i)))
    using e2ennreal-finite-sum by (metis enn2ereal-nonneg eq finite-atMost)
  ultimately have ((λn. (∑ i≤n. e2ennreal (μ (A i)))) ⟶
    e2ennreal (μ (⋃ (range A)))) sequentially by simp
  hence (λi. e2ennreal (μ (A i))) sums e2ennreal (μ (⋃ (range A)))
    using sums-def-le[of λi. e2ennreal (μ (A i)) e2ennreal (μ (⋃ (range A)))]
    by simp
  thus (∑ i. e2ennreal (μ (A i))) = e2ennreal (μ (⋃ (range A)))
    using sums-unique assms by (metis)
qed

lemma signed-measure-minus:
  assumes signed-measure M μ
  shows signed-measure M (λA. – μ A) unfolding signed-measure-def
```

proof (*intro conjI*)
show $-\mu$ {} = *0* **using** *assms* **unfolding** *signed-measure-def* **by** *simp*
show $-\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) $\vee$ $\infty \notin$ *range* ($\lambda A.\ -\mu\ A$)
proof (*cases* $\infty \in$ *range* μ)
case *True*
hence $-\infty \notin$ *range* μ **using** *assms* **unfolding** *signed-measure-def* **by** *simp*
hence $\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) **using** *ereal-uminus-eq-reorder* **by** *blast*
thus $-\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) $\vee$ $\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) **by** *simp*
next
case *False*
hence $-\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) **using** *ereal-uminus-eq-reorder*
by (*simp add*: *image-iff*)
thus $-\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) $\vee$ $\infty \notin$ *range* ($\lambda A.\ -\mu\ A$) **by** *simp*
qed
show $\forall A$. *range* $A \subseteq$ *sets M* $\longrightarrow$ *disjoint-family A* $\longrightarrow$ $\bigcup$ (*range A*) $\in$ *sets M*
$\longrightarrow$
$|-\mu$ ($\bigcup$ (*range A*))$| < \infty$ $\longrightarrow$ *summable* (λi. *real-of-ereal* $|-\mu$ (*A i*)$|$)
proof (*intro allI impI*)
fix *A*::*nat* $\Rightarrow$ *'a set*
assume *range* $A \subseteq$ *sets M* **and** *disjoint-family A* **and** $\bigcup$ (*range A*) $\in$ *sets M*
and $|-\mu$ ($\bigcup$ (*range A*))$| < \infty$
thus *summable* (λi. *real-of-ereal* $|-\mu$ (*A i*)$|$) **using** *assms*
unfolding *signed-measure-def* **by** *simp*
qed
show $\forall A$. *range* $A \subseteq$ *sets M* $\longrightarrow$ *disjoint-family A* $\longrightarrow$ $\bigcup$ (*range A*) $\in$ *sets M*
$\longrightarrow$
($\lambda i.\ -\mu$ (*A i*)) *sums* $-\mu$ ($\bigcup$ (*range A*))
proof $-$
{
fix *A*::*nat* $\Rightarrow$ *'a set*
assume *range* $A \subseteq$ *sets M* **and** *disjoint-family A* **and**
$\bigcup$ (*range A*) $\in$ *sets M* **note** *Aprops* = *this*
have $-\infty \notin$ *range* ($\lambda i.\ \mu$ (*A i*)) $\vee$ $\infty \notin$ *range* ($\lambda i.\ \mu$ (*A i*))
proof $-$
have *range* ($\lambda i.\ \mu$ (*A i*)) $\subseteq$ *range* μ **by** *auto*
thus *?thesis* **using** *assms* **unfolding** *signed-measure-def* **by** *auto*
qed
moreover have ($\lambda i.\ \mu$ (*A i*)) *sums* μ ($\bigcup$ (*range A*))
using *signed-measure-sums*[*of M*] *Aprops assms* **by** *simp*
ultimately have ($\lambda i.\ -\mu$ (*A i*)) *sums* $-\mu$ ($\bigcup$ (*range A*))
using *sums-minus'*[*of* $\lambda i.\ \mu$ (*A i*)] **by** *simp*
}
thus *?thesis* **by** *auto*
qed
qed

locale *near-finite-function* =
fixes μ:: *'b set* $\Rightarrow$ *ereal*
assumes *inf-range*: $-\infty \notin$ *range* μ $\vee$ $\infty \notin$ *range* μ

```
lemma (in near-finite-function) finite-subset:
  assumes |μ E| < ∞
    and A⊆ E
    and μ E = μ A + μ (E − A)
  shows |μ A| < ∞
proof (cases ∞ ∈ range μ)
  case False
  show ?thesis
  proof (cases 0 < μ A)
    case True
    hence |μ A| = μ A by simp
    also have ... < ∞ using False by (metis ereal-less-PInfty rangeI)
    finally show ?thesis .
  next
    case False
    hence |μ A| = −μ A using not-less-iff-gr-or-eq by fastforce
    also have ... = μ (E − A) − μ E
    proof −
      have μ E = μ A + μ (E − A) using assms by simp
      hence μ E − μ A = μ (E − A)
        by (metis abs-ereal-uminus assms(1) calculation ereal-diff-add-inverse
            ereal-infty-less(2) ereal-minus(5) ereal-minus-less-iff
            ereal-minus-less-minus ereal-uminus-uminus less-ereal.simps(2)
            minus-ereal-def plus-ereal.simps(3))
      thus ?thesis using assms(1) ereal-add-uminus-conv-diff ereal-eq-minus
        by auto
    qed
    also have ... ≤ μ (E − A) + |μ E|
      by (metis ‹− μ A = μ (E − A) − μ E› abs-ereal-less0 abs-ereal-pos
          ereal-diff-le-self add-increasing2 less-eq-ereal-def
          minus-ereal-def not-le-imp-less)
    also have ... < ∞ using assms ‹∞ ∉ range μ›
      by (metis UNIV-I ereal-less-PInfty ereal-plus-eq-PInfty image-eqI)
    finally show ?thesis .
  qed
next
  case True
  hence −∞ ∉ range μ using inf-range by simp
  hence −∞ < μ A by (metis ereal-infty-less(2) rangeI)
  show ?thesis
  proof (cases μ A < 0)
    case True
    hence |μ A| = −μ A using not-less-iff-gr-or-eq by fastforce
    also have ... < ∞ using ‹−∞ < μ A› using ereal-uminus-less-reorder
      by blast
    finally show ?thesis .
  next
    case False
```

```
    hence |μ A| = μ A by simp
    also have ... = μ E − μ (E − A)
    proof −
      have μ E = μ A + μ (E − A) using assms by simp
      thus μ A = μ E − μ (E − A) by (metis add.right-neutral assms(1)
            add-diff-eq-ereal calculation ereal-diff-add-eq-diff-diff-swap
            ereal-diff-add-inverse ereal-infty-less(1) ereal-plus-eq-PInfty
            ereal-x-minus-x)
    qed
    also have ... ≤ |μ E| − μ (E − A)
      by (metis ‹|μ A| = μ A› ‹μ A = μ E − μ (E − A)› abs-ereal-ge0
          abs-ereal-pos abs-ereal-uminus antisym-conv ereal-0-le-uminus-iff
          ereal-abs-diff ereal-diff-le-mono-left ereal-diff-le-self le-cases
          less-eq-ereal-def minus-ereal-def)
    also have ... < ∞
    proof −
      have −∞ < μ (E − A) using ‹−∞ ∉ range μ›
        by (metis ereal-infty-less(2) rangeI)
      hence − μ (E − A) < ∞ using ereal-uminus-less-reorder by blast
      thus ?thesis using assms by (simp add: ereal-minus-eq-PInfty-iff
            ereal-uminus-eq-reorder)
    qed
    finally show ?thesis .
  qed
qed

locale signed-measure-space=
  fixes M::'a measure and μ
  assumes sgn-meas: signed-measure M μ

sublocale signed-measure-space ⊆ near-finite-function
proof (unfold-locales)
  show − ∞ ∉ range μ ∨ ∞ ∉ range μ using sgn-meas
    unfolding signed-measure-def by simp
qed

context signed-measure-space
begin
lemma signed-measure-finite-subset:
  assumes E ∈ sets M
    and |μ E| < ∞
    and A∈ sets M
    and A⊆ E
  shows |μ A| < ∞
proof (rule finite-subset)
  show |μ E| < ∞ A⊆ E using assms by auto
  show μ E = μ A + μ (E − A) using assms
      sgn-meas signed-measure-add[of M μ A E − A]
    by (metis Diff-disjoint Diff-partition sets.Diff)
```

qed

lemma *measure-space-e2ennreal* :
assumes *measure-space* (*space M*) (*sets M*) $m \wedge (\forall E \in sets\ M.\ m\ E < \infty) \wedge$
$(\forall E \in sets\ M.\ m\ E \geq 0)$
shows $\forall E \in sets\ M.\ e2ennreal\ (m\ E) < \infty$
proof
fix *E*
assume $E \in sets\ M$
show $e2ennreal\ (m\ E) < \infty$
proof −
have $m\ E < \infty$ **using** *assms* ‹$E \in sets\ M$›
by *blast*
then have $e2ennreal\ (m\ E) < \infty$ **using** *e2ennreal-less-top*
using ‹$m\ E < \infty$› **by** *auto*
thus *?thesis* **by** *simp*
qed
qed

2.2 Positive and negative subsets

The Hahn decomposition theorem is based on the notions of positive and negative measurable sets. A measurable set is positive (resp. negative) if all its measurable subsets have a positive (resp. negative) measure by μ. The decomposition theorem states that any measure space for a signed measure can be decomposed into a positive and a negative measurable set.

definition *pos-meas-set* **where**
pos-meas-set $E \longleftrightarrow E \in sets\ M \wedge (\forall A \in sets\ M.\ A \subseteq E \longrightarrow 0 \leq \mu\ A)$

definition *neg-meas-set* **where**
neg-meas-set $E \longleftrightarrow E \in sets\ M \wedge (\forall A \in sets\ M.\ A \subseteq E \longrightarrow \mu\ A \leq 0)$

lemma *pos-meas-setI*:
assumes $E \in sets\ M$
and $\bigwedge A.\ A \in sets\ M \Longrightarrow A \subseteq E \Longrightarrow 0 \leq \mu\ A$
shows *pos-meas-set E* **unfolding** *pos-meas-set-def* **using** *assms* **by** *simp*

lemma *pos-meas-setD1* :
assumes *pos-meas-set E*
shows $E \in sets\ M$
using *assms* **unfolding** *pos-meas-set-def*
by *simp*

lemma *neg-meas-setD1* :
assumes *neg-meas-set E*
shows $E \in sets\ M$ **using** *assms* **unfolding** *neg-meas-set-def* **by** *simp*

lemma *neg-meas-setI*:

assumes *E* ∈ *sets M*
and ⋀*A*. *A* ∈ *sets M* ⟹ *A* ⊆ *E* ⟹ *μ A* ≤ *0*
shows *neg-meas-set E* **unfolding** *neg-meas-set-def* **using** *assms* **by** *simp*

lemma *pos-meas-self*:
assumes *pos-meas-set E*
shows *0* ≤ *μ E* **using** *assms* **unfolding** *pos-meas-set-def* **by** *simp*

lemma *empty-pos-meas-set*:
shows *pos-meas-set* {}
by (*metis bot.extremum-uniqueI eq-iff pos-meas-set-def sets.empty-sets sgn-meas signed-measure-empty*)

lemma *empty-neg-meas-set*:
shows *neg-meas-set* {}
by (*metis neg-meas-set-def order-refl sets.empty-sets sgn-meas signed-measure-empty subset-empty*)

lemma *pos-measure-meas*:
assumes *pos-meas-set E*
and *A*⊆ *E*
and *A*∈ *sets M*
shows *0* ≤ *μ A* **using** *assms* **unfolding** *pos-meas-set-def* **by** *simp*

lemma *pos-meas-subset*:
assumes *pos-meas-set A*
and *B*⊆ *A*
and *B*∈ *sets M*
shows *pos-meas-set B* **using** *assms pos-meas-set-def* **by** *auto*

lemma *neg-meas-subset*:
assumes *neg-meas-set A*
and *B*⊆ *A*
and *B*∈ *sets M*
shows *neg-meas-set B* **using** *assms neg-meas-set-def* **by** *auto*

lemma *pos-meas-set-Union*:
assumes ⋀(*i*::*nat*). *pos-meas-set* (*A i*)
and ⋀*i*. *A i* ∈ *sets M*
and |*μ* (⋃ *i*. *A i*)| < ∞
shows *pos-meas-set* (⋃ *i*. *A i*)
proof (*rule pos-meas-setI*)
show ⋃ (*range A*) ∈ *sets M* **using** *sigma-algebra.countable-UN assms* **by** *simp*
obtain *B* **where** *disjoint-family B* **and** (⋃(*i*::*nat*). *B i*) = (⋃(*i*::*nat*). *A i*)
and ⋀*i*. *B i* ∈ *sets M* **and** ⋀*i*. *B i* ⊆ *A i* **using** *disj-Union2 assms* **by** *auto*
fix *C*
assume *C* ∈ *sets M* **and** *C*⊆ (⋃ *i*. *A i*)
hence *C* = *C* ∩ (⋃ *i*. *A i*) **by** *auto*
also have ... = *C* ∩ (⋃ *i*. *B i*) **using** ‹(⋃*i*. *B i*) = (⋃*i*. *A i*)› **by** *simp*

```
also have ... = (⋃ i. C ∩ B i) by auto
finally have C = (⋃ i. C ∩ B i) .
hence μ C = μ (⋃ i. C ∩ B i) by simp
also have ... = (∑ i. μ (C ∩ (B i)))
proof (rule signed-measure-inf-sum[symmetric])
  show signed-measure M μ using sgn-meas by simp
  show disjoint-family (λi. C ∩ B i) using ‹disjoint-family B›
    by (meson Int-iff disjoint-family-subset subset-iff)
  show range (λi. C ∩ B i) ⊆ sets M using ‹C∈ sets M› ‹⋀i. B i ∈ sets M›
    by auto
  show (⋃i. C ∩ B i) ∈ sets M using ‹C = (⋃ i. C ∩ B i)› ‹C∈ sets M›
    by simp
qed
also have ... ≥ 0
proof (rule suminf-nonneg)
  show ⋀n. 0 ≤ μ (C ∩ B n)
  proof −
    fix n
    have C∩ B n ⊆ A n using ‹⋀i. B i ⊆ A i› by auto
    moreover have C ∩ B n ∈ sets M using ‹C∈ sets M› ‹⋀i. B i ∈ sets M›
      by simp
    ultimately show 0 ≤ μ (C ∩ B n) using assms pos-measure-meas[of A n]
      by simp
  qed
  have summable (λi. real-of-ereal (μ (C ∩ B i)))
  proof (rule summable-norm-cancel)
    have ⋀n. norm (real-of-ereal (μ (C ∩ B n))) =
      real-of-ereal |μ (C ∩ B n)| by simp
    moreover have summable (λi. real-of-ereal |μ (C ∩ B i)|)
    proof (rule signed-measure-abs-convergent)
      show signed-measure M μ using sgn-meas by simp
      show range (λi. C ∩ B i) ⊆ sets M using ‹C∈ sets M›
          ‹⋀i. B i ∈ sets M› by auto
      show disjoint-family (λi. C ∩ B i) using ‹disjoint-family B›
        by (meson Int-iff disjoint-family-subset subset-iff)
      show (⋃i. C ∩ B i) ∈ sets M using ‹C = (⋃ i. C ∩ B i)› ‹C∈ sets M›
        by simp
      have |μ C| < ∞
      proof (rule signed-measure-finite-subset)
        show (⋃ i. A i) ∈ sets M using assms by simp
        show |μ (⋃ (range A))| < ∞ using assms by simp
        show C ∈ sets M using ‹C ∈ sets M› .
        show C ⊆ ⋃ (range A) using ‹C ⊆ ⋃ (range A) › .
      qed
      thus |μ (⋃i. C ∩ B i)| < ∞ using ‹C = (⋃ i. C ∩ B i)› by simp
    qed
    ultimately show summable (λn. norm (real-of-ereal (μ (C ∩ B n))))
      by auto
  qed
```

thus *summable* (λ*i*. μ (*C* ∩ *B i*)) **by** (*simp add*: ‹⋀*n*. *0* ≤ μ (*C* ∩ *B n*)›
summable-ereal-pos)
qed
finally show *0* ≤ μ *C* **.**
qed

lemma *pos-meas-set-pos-lim*:
assumes ⋀(*i*::*nat*). *pos-meas-set* (*A i*)
and ⋀*i*. *A i* ∈ *sets M*
shows *0* ≤ μ (⋃ *i*. *A i*)
proof −
obtain *B* **where** *disjoint-family B* **and** (⋃(*i*::*nat*). *B i*) = (⋃(*i*::*nat*). *A i*)
and ⋀*i*. *B i* ∈ *sets M* **and** ⋀*i*. *B i* ⊆ *A i* **using** *disj-Union2 assms* **by** *auto*
note *Bprops* = *this*
have *sums*: (λ*n*. μ (*B n*)) *sums* μ (⋃*i*. *B i*)
proof (*rule signed-measure-sums*)
show *signed-measure M* μ **using** *sgn-meas* **.**
show *range B* ⊆ *sets M* **using** *Bprops* **by** *auto*
show *disjoint-family B* **using** *Bprops* **by** *simp*
show ⋃ (*range B*) ∈ *sets M* **using** *Bprops* **by** *blast*
qed
hence *summable* (λ*n*. μ (*B n*)) **using** *sums-summable*[*of* λ*n*. μ (*B n*)] **by** *simp*
hence *suminf* (λ*n*. μ (*B n*)) = μ (⋃*i*. *B i*) **using** *sums sums-iff* **by** *auto*
thus *?thesis* **using** *suminf-nonneg*
by (*metis Bprops*(*2*) *Bprops*(*3*) *Bprops*(*4*) ‹*summable* (λ*n*. μ (*B n*))› *assms*(*1*)

pos-measure-meas)
qed

lemma *pos-meas-disj-union*:
assumes *pos-meas-set A*
and *pos-meas-set B*
and *A*∩ *B* = {}
shows *pos-meas-set* (*A* ∪ *B*) **unfolding** *pos-meas-set-def*
proof (*intro conjI ballI impI*)
show *A*∪ *B* ∈ *sets M*
by (*metis assms*(*1*) *assms*(*2*) *pos-meas-set-def sets.Un*)
next
fix *C*
assume *C*∈ *sets M* **and** *C*⊆ *A*∪ *B*
define *DA* **where** *DA* = *C*∩ *A*
define *DB* **where** *DB* = *C*∩ *B*
have *DA*∈ *sets M* **using** *DA-def* ‹*C* ∈ *sets M*› *assms*(*1*) *pos-meas-set-def*
by *blast*
have *DB*∈ *sets M* **using** *DB-def* ‹*C* ∈ *sets M*› *assms*(*2*) *pos-meas-set-def*
by *blast*
have *DA* ∩ *DB* = {} **unfolding** *DA-def DB-def* **using** *assms* **by** *auto*
have *C* = *DA* ∪ *DB* **unfolding** *DA-def DB-def* **using** ‹*C*⊆ *A*∪ *B*› **by** *auto*
have *0* ≤ μ *DB* **using** *assms* **unfolding** *DB-def pos-meas-set-def*

```
    by (metis  DB-def Int-lower2‹DB ∈ sets M›)
  also have ... ≤ μ DA + μ DB using assms unfolding  pos-meas-set-def
    by (metis DA-def Diff-Diff-Int Diff-subset Int-commute ‹DA ∈ sets M›
        ereal-le-add-self2)
  also have ... = μ C using signed-measure-add sgn-meas ‹DA ∈ sets M›
      ‹DB ∈ sets M› ‹DA ∩ DB = {}› ‹C = DA ∪ DB› by metis
  finally show 0 ≤ μ C .
qed

lemma pos-meas-set-union:
  assumes pos-meas-set A
    and pos-meas-set B
  shows pos-meas-set (A ∪ B)
proof −
  define C where C = B − A
  have A∪ C = A∪ B unfolding C-def by auto
  moreover have pos-meas-set (A∪ C)
  proof (rule pos-meas-disj-union)
    show pos-meas-set C unfolding C-def
      by (meson Diff-subset assms(1) assms(2) sets.Diff
          signed-measure-space.pos-meas-set-def
          signed-measure-space.pos-meas-subset signed-measure-space-axioms)
    show pos-meas-set A using assms by simp
    show A ∩ C = {} unfolding C-def by auto
  qed
  ultimately show ?thesis by simp
qed

lemma neg-meas-disj-union:
  assumes neg-meas-set A
    and neg-meas-set B
    and A∩ B = {}
  shows neg-meas-set (A ∪ B) unfolding neg-meas-set-def
proof (intro conjI ballI impI)
  show A∪ B ∈ sets M
    by (metis assms(1) assms(2) neg-meas-set-def sets.Un)
next
  fix C
  assume C∈ sets M and C⊆ A∪ B
  define DA where DA = C∩ A
  define DB where DB = C∩ B
  have DA∈ sets M using DA-def ‹C ∈ sets M› assms(1) neg-meas-set-def
    by blast
  have DB∈ sets M using DB-def ‹C ∈ sets M› assms(2) neg-meas-set-def
    by blast
  have DA ∩ DB = {} unfolding DA-def DB-def using assms by auto
  have C = DA ∪ DB unfolding DA-def DB-def using ‹C⊆ A∪ B› by auto
  have μ C = μ DA + μ DB using signed-measure-add sgn-meas ‹DA ∈ sets M›
      ‹DB ∈ sets M› ‹DA ∩ DB = {}› ‹C = DA ∪ DB› by metis
```

also have ... ≤ *μ DB* **using** *assms* **unfolding** *neg-meas-set-def*
by (*metis DA-def Int-lower2* ‹*DA* ∈ *sets M*› *add-decreasing dual-order.refl*)
also have ... ≤ *0* **using** *assms* **unfolding** *DB-def neg-meas-set-def*
by (*metis DB-def Int-lower2*‹*DB* ∈ *sets M*›)
finally show *μ C* ≤ *0* **.**
qed

lemma *neg-meas-set-union*:
assumes *neg-meas-set A*
and *neg-meas-set B*
shows *neg-meas-set* (*A* ∪ *B*)
proof −
define *C* **where** *C* = *B* − *A*
have *A*∪ *C* = *A*∪ *B* **unfolding** *C-def* **by** *auto*
moreover have *neg-meas-set* (*A*∪ *C*)
proof (*rule neg-meas-disj-union*)
show *neg-meas-set C* **unfolding** *C-def*
by (*meson Diff-subset assms*(*1*) *assms*(*2*) *sets.Diff neg-meas-set-def*
neg-meas-subset signed-measure-space-axioms)
show *neg-meas-set A* **using** *assms* **by** *simp*
show *A* ∩ *C* = {} **unfolding** *C-def* **by** *auto*
qed
ultimately show *?thesis* **by** *simp*
qed

lemma *neg-meas-self* :
assumes *neg-meas-set E*
shows *μ E* ≤ *0* **using** *assms* **unfolding** *neg-meas-set-def* **by** *simp*

lemma *pos-meas-set-opp*:
assumes *signed-measure-space.pos-meas-set M* (λ *A*. − *μ A*) *A*
shows *neg-meas-set A*
proof −
have *m-meas-pos* : *signed-measure M* (λ *A*. − *μ A*)
using *assms signed-measure-space-def*
by (*simp add*: *sgn-meas signed-measure-minus*)
thus *?thesis*
by (*metis assms ereal-0-le-uminus-iff neg-meas-setI*
signed-measure-space.intro signed-measure-space.pos-meas-set-def)
qed

lemma *neg-meas-set-opp*:
assumes *signed-measure-space.neg-meas-set M* (λ *A*. − *μ A*) *A*
shows *pos-meas-set A*
proof −
have *m-meas-neg* : *signed-measure M* (λ *A*. − *μ A*)
using *assms signed-measure-space-def*
by (*simp add*: *sgn-meas signed-measure-minus*)
thus *?thesis*

```
    by (metis assms ereal-uminus-le-0-iff m-meas-neg pos-meas-setI
        signed-measure-space.intro signed-measure-space.neg-meas-set-def)
qed
end

lemma signed-measure-inter:
  assumes signed-measure M μ
    and A ∈ sets M
  shows signed-measure M (λE. μ (E ∩ A)) unfolding signed-measure-def
proof (intro conjI)
  show μ ({} ∩ A) = 0 using assms(1) signed-measure-empty by auto
  show − ∞ ∉ range (λE. μ (E ∩ A)) ∨ ∞ ∉ range (λE. μ (E ∩ A))
  proof (rule ccontr)
    assume ¬ (− ∞ ∉ range (λE. μ (E ∩ A)) ∨ ∞ ∉ range (λE. μ (E ∩ A)))
    hence − ∞ ∈ range (λE. μ (E ∩ A)) ∧ ∞ ∈ range (λE. μ (E ∩ A)) by simp
    hence − ∞ ∈ range μ ∧ ∞ ∈ range μ by auto
    thus False using assms unfolding signed-measure-def by simp
  qed
  show ∀ E. range E ⊆ sets M ⟶ disjoint-family E ⟶ ⋃ (range E) ∈ sets M
⟶
    (λi. μ (E i ∩ A)) sums μ (⋃ (range E) ∩ A)
  proof (intro allI impI)
    fix E::nat ⇒ 'a set
    assume range E ⊆ sets M and disjoint-family E and ⋃ (range E) ∈ sets M
    note Eprops = this
    define F where F = (λi. E i ∩ A)
    have (λi. μ (F i)) sums μ (⋃ (range F))
    proof (rule signed-measure-sums)
      show signed-measure M μ using assms by simp
      show range F ⊆ sets M using Eprops F-def assms by blast
      show disjoint-family F using Eprops F-def assms
        by (metis disjoint-family-subset inf.absorb-iff2 inf-commute
            inf-right-idem)
      show ⋃ (range F) ∈ sets M using Eprops assms unfolding F-def
        by (simp add: Eprops assms countable-Un-Int(1) sets.Int)
    qed
    moreover have ⋃ (range F) = A ∩ ⋃ (range E) unfolding F-def by auto
    ultimately show (λi. μ (E i ∩ A)) sums μ (⋃ (range E) ∩ A)
      unfolding F-def by simp
  qed
  show ∀ E. range E ⊆ sets M ⟶
        disjoint-family E ⟶
        ⋃ (range E) ∈ sets M ⟶ |μ (⋃ (range E) ∩ A)| < ∞ ⟶
        summable (λi. real-of-ereal |μ (E i ∩ A)|)
  proof (intro allI impI)
    fix E::nat ⇒ 'a set
    assume range E ⊆ sets M and disjoint-family E and
      ⋃ (range E) ∈ sets M and |μ (⋃ (range E) ∩ A)| < ∞ note Eprops = this
    show summable (λi. real-of-ereal |μ (E i ∩ A)|)
```

```
    proof (rule signed-measure-summable)
      show signed-measure M μ using assms by simp
      show range (λi. E i ∩ A) ⊆ sets M using Eprops assms by blast
      show disjoint-family (λi. E i ∩ A) using Eprops assms
          disjoint-family-subset inf.absorb-iff2 inf-commute inf-right-idem
        by fastforce
      show (⋃ i. E i ∩ A) ∈ sets M using Eprops assms
        by (simp add: Eprops assms countable-Un-Int(1) sets.Int)
      show |μ (⋃ i. E i ∩ A)| < ∞ using Eprops by auto
    qed
  qed
qed

context signed-measure-space
begin
lemma pos-signed-to-meas-space :
  assumes pos-meas-set M1
    and m1 = (λA. μ (A ∩ M1))
  shows measure-space (space M) (sets M) m1 unfolding measure-space-def
proof (intro conjI)
  show sigma-algebra (space M) (sets M)
    by (simp add: sets.sigma-algebra-axioms)
  show positive (sets M) m1 using assms unfolding pos-meas-set-def
    by (metis Sigma-Algebra.positive-def Un-Int-eq(4)
        e2ennreal-neg neg-meas-self sup-bot-right empty-neg-meas-set)
  show countably-additive (sets M) m1
  proof (rule pos-signed-measure-count-additive)
    show ∀ E∈sets M. 0 ≤ m1 E by (metis assms inf.cobounded2
          pos-meas-set-def sets.Int)
    show signed-measure M m1 using assms pos-meas-set-def
        signed-measure-inter[of M μ M1] sgn-meas by blast
  qed
qed

lemma neg-signed-to-meas-space :
  assumes neg-meas-set M2
    and m2 = (λA. −μ (A ∩ M2))
  shows measure-space (space M) (sets M) m2 unfolding measure-space-def
proof (intro conjI)
  show sigma-algebra (space M) (sets M)
    by (simp add: sets.sigma-algebra-axioms)
  show positive (sets M) m2 using assms unfolding neg-meas-set-def
    by (metis Sigma-Algebra.positive-def e2ennreal-neg ereal-uminus-zero
        inf.absorb-iff2 inf.orderE inf-bot-right neg-meas-self pos-meas-self
        empty-neg-meas-set empty-pos-meas-set)
  show countably-additive (sets M) m2
  proof (rule pos-signed-measure-count-additive)
    show ∀ E∈sets M. 0 ≤ m2 E
      by (metis assms ereal-uminus-eq-reorder ereal-uminus-le-0-iff
```

inf.cobounded2 neg-meas-set-def sets.Int)
have *signed-measure M* (*λA. μ* (*A* ∩ *M2*)) **using** *assms neg-meas-set-def signed-measure-inter*[*of M μ M2*] *sgn-meas* **by** *blast*
thus *signed-measure M m2* **using** *signed-measure-minus assms* **by** *simp*
qed
qed

lemma *pos-part-meas-nul-neg-set* :
assumes *pos-meas-set M1*
and *neg-meas-set M2*
and *m1* = (*λA. μ* (*A* ∩ *M1*))
and *E* ∈ *sets M*
and *E* ⊆ *M2*
shows *m1 E* = *0*
proof −
have *m1 E* ≥ *0* **using** *assms* **unfolding** *pos-meas-set-def* **by** (*simp add*: ‹*E* ∈ *sets M*› *sets.Int*)
have *μ E* ≤ *0* **using** ‹*E* ⊆ *M2*› *assms* **unfolding** *neg-meas-set-def* **using** ‹*E* ∈ *sets M*› **by** *blast*
then have *m1 E* ≤ *0* **using** ‹*μ E* ≤ *0*› *assms* **by** (*metis Int-Un-eq*(*1*) *Un-subset-iff* ‹*E* ∈ *sets M*› ‹*E* ⊆ *M2*› *pos-meas-setD1 sets.Int signed-measure-space.neg-meas-set-def signed-measure-space-axioms*)
thus *m1 E* = *0* **using** ‹*m1 E* ≥ *0*› ‹*m1 E* ≤ *0*› **by** *auto*
qed

lemma *neg-part-meas-nul-pos-set* :
assumes *pos-meas-set M1*
and *neg-meas-set M2*
and *m2* = (*λA.* −*μ* (*A* ∩ *M2*))
and *E* ∈ *sets M*
and *E* ⊆ *M1*
shows *m2 E* = *0*
proof −
have *m2 E* ≥ *0* **using** *assms* **unfolding** *neg-meas-set-def* **by** (*simp add*: ‹*E* ∈ *sets M*› *sets.Int*)
have *μ E* ≥ *0* **using** *assms* **unfolding** *pos-meas-set-def* **by** *blast*
then have *m2 E* ≤ *0* **using** ‹*μ E* ≥ *0*› *assms* **by** (*metis* ‹*E* ∈ *sets M*› ‹*E* ⊆ *M1*› *ereal-0-le-uminus-iff ereal-uminus-uminus inf-sup-ord*(*1*) *neg-meas-setD1 pos-meas-set-def pos-meas-subset sets.Int*)
thus *m2 E* = *0* **using** ‹*m2 E* ≥ *0*› ‹*m2 E* ≤ *0*› **by** *auto*
qed

definition *pos-sets* **where**
pos-sets = {*A. A* ∈ *sets M* ∧ *pos-meas-set A*}

definition *pos-img* **where**
pos-img = {*μ A*|*A. A*∈ *pos-sets*}

2.3 Essential uniqueness

In this part, under the assumption that a measure space for a signed measure admits a decomposition into a positive and a negative set, we prove that this decomposition is essentially unique; in other words, that if two such decompositions (P, N) and (X, Y) exist, then any measurable subset of $(P \triangle X) \cup (N \triangle Y)$ has a null measure.

definition *hahn-space-decomp* **where**
hahn-space-decomp M1 M2 ≡ (*pos-meas-set M1*) ∧ (*neg-meas-set M2*) ∧
(*space M* = *M1* ∪ *M2*) ∧ (*M1* ∩ *M2* = {})

lemma *pos-neg-null-set*:
assumes *pos-meas-set A*
and *neg-meas-set A*
shows *μ A* = *0* **using** *assms pos-meas-self*[*of A*] *neg-meas-self*[*of A*] **by** *simp*

lemma *pos-diff-neg-meas-set*:
assumes (*pos-meas-set M1*)
and (*neg-meas-set N2*)
and (*space M* = *N1* ∪ *N2*)
and *N1* ∈ *sets M*
shows *neg-meas-set* ((*M1* − *N1*) ∩ *space M*) **using** *assms neg-meas-subset*
by (*metis Diff-subset-conv Int-lower2 pos-meas-setD1 sets.Diff*
sets.Int-space-eq2)

lemma *neg-diff-pos-meas-set*:
assumes (*neg-meas-set M2*)
and (*pos-meas-set N1*)
and (*space M* = *N1* ∪ *N2*)
and *N2* ∈ *sets M*
shows *pos-meas-set* ((*M2* − *N2*) ∩ *space M*)
proof −
have (*M2* − *N2*) ∩ *space M* ⊆ *N1* **using** *assms* **by** *auto*
thus *?thesis* **using** *assms pos-meas-subset neg-meas-setD1* **by** *blast*
qed

lemma *pos-sym-diff-neg-meas-set*:
assumes *hahn-space-decomp M1 M2*
and *hahn-space-decomp N1 N2*
shows *neg-meas-set* ((*sym-diff M1 N1*) ∩ *space M*) **using** *assms*
unfolding *hahn-space-decomp-def*
by (*metis Int-Un-distrib2 neg-meas-set-union pos-meas-setD1*
pos-diff-neg-meas-set)

lemma *neg-sym-diff-pos-meas-set*:
assumes *hahn-space-decomp M1 M2*
and *hahn-space-decomp N1 N2*
shows *pos-meas-set* ((*sym-diff M2 N2*) ∩ *space M*) **using** *assms*
neg-diff-pos-meas-set **unfolding** *hahn-space-decomp-def*

```
  by (metis (no-types, lifting) Int-Un-distrib2 neg-meas-setD1
      pos-meas-set-union)

lemma pos-meas-set-diff:
  assumes pos-meas-set A
    and B∈ sets M
  shows pos-meas-set ((A − B) ∩ (space M)) using pos-meas-subset
  by (metis Diff-subset assms(1) assms(2) pos-meas-setD1 sets.Diff
      sets.Int-space-eq2)

lemma pos-meas-set-sym-diff:
  assumes pos-meas-set A
    and pos-meas-set B
  shows pos-meas-set ((sym-diff A B) ∩ space M) using pos-meas-set-diff
  by (metis Int-Un-distrib2 assms(1) assms(2) pos-meas-setD1
      pos-meas-set-union)

lemma neg-meas-set-diff:
  assumes neg-meas-set A
    and B∈ sets M
  shows neg-meas-set ((A − B) ∩ (space M)) using neg-meas-subset
  by (metis Diff-subset assms(1) assms(2) neg-meas-setD1 sets.Diff
      sets.Int-space-eq2)

lemma neg-meas-set-sym-diff:
  assumes neg-meas-set A
    and neg-meas-set B
  shows neg-meas-set ((sym-diff A B) ∩ space M) using neg-meas-set-diff
  by (metis Int-Un-distrib2 assms(1) assms(2) neg-meas-setD1
      neg-meas-set-union)

lemma hahn-decomp-space-diff:
  assumes hahn-space-decomp M1 M2
    and hahn-space-decomp N1 N2
  shows pos-meas-set ((sym-diff M1 N1 ∪ sym-diff M2 N2) ∩ space M)
    neg-meas-set ((sym-diff M1 N1 ∪ sym-diff M2 N2) ∩ space M)
proof −
  show pos-meas-set ((sym-diff M1 N1 ∪ sym-diff M2 N2) ∩ space M)
    by (metis Int-Un-distrib2 assms(1) assms(2) hahn-space-decomp-def
        neg-sym-diff-pos-meas-set pos-meas-set-sym-diff pos-meas-set-union)
  show neg-meas-set ((sym-diff M1 N1 ∪ sym-diff M2 N2) ∩ space M)
    by (metis Int-Un-distrib2 assms(1) assms(2) hahn-space-decomp-def
        neg-meas-set-sym-diff neg-meas-set-union pos-sym-diff-neg-meas-set)
qed

lemma hahn-decomp-ess-unique:
  assumes hahn-space-decomp M1 M2
    and hahn-space-decomp N1 N2
    and C ⊆ sym-diff M1 N1 ∪ sym-diff M2 N2
```

and *C*∈ *sets M*
shows *μ C* = *0*
proof −
have *C*⊆ (*sym-diff M1 N1* ∪ *sym-diff M2 N2*) ∩ *space M* **using** *assms*
by (*simp add*: *sets.sets-into-space*)
thus *?thesis* **using** *assms hahn-decomp-space-diff pos-neg-null-set*
by (*meson neg-meas-subset pos-meas-subset*)
qed

3 Existence of a positive subset

The goal of this part is to prove that any measurable set of finite and positive measure must contain a positive subset with a strictly positive measure.

3.1 A sequence of negative subsets

definition *inf-neg* **where**
inf-neg A = (*if* (*A* ∉ *sets M* ∨ *pos-meas-set A*) *then* (*0*::*nat*)
else Inf {*n*|*n*. (*1*::*nat*) ≤ *n* ∧ (∃ *B* ∈ *sets M*. *B* ⊆ *A* ∧ *μ B* < *ereal*(−*1*/*n*))})

lemma *inf-neg-ne*:
assumes *A* ∈ *sets M*
and ¬ *pos-meas-set A*
shows {*n*::*nat*|*n*. (*1*::*nat*) ≤ *n* ∧
(∃ *B* ∈ *sets M*. *B* ⊆ *A* ∧ *μ B* < *ereal* (−*1*/*n*))} ≠ {}
proof −
define *N* **where** *N* = {*n*::*nat*|*n*. (*1*::*nat*) ≤ *n* ∧
(∃ *B* ∈ *sets M*. *B* ⊆ *A* ∧ *μ B* < *ereal* (−*1*/*n*))}
have ∃ *B* ∈ *sets M*. *B*⊆ *A* ∧ *μ B* < *0* **using** *assms* **unfolding** *pos-meas-set-def*
by *auto*
from *this* **obtain** *B* **where** *B*∈ *sets M* **and** *B*⊆ *A* **and** *μ B* < *0* **by** *auto*
hence ∃ *n*::*nat*. (*1*::*nat*) ≤ *n* ∧ *μ B* < *ereal* (−*1*/*n*)
proof (*cases μ B* = −∞)
case *True*
hence *μ B* < −*1*/(*2*::*nat*) **by** *simp*
thus *?thesis* **using** *numeral-le-real-of-nat-iff one-le-numeral* **by** *blast*
next
case *False*
hence *real-of-ereal* (*μ B*) < *0* **using** ‹*μ B* < *0*›
by (*metis Infty-neq-0*(*3*) *ereal-mult-eq-MInfty ereal-zero-mult*
less-eq-ereal-def less-eq-real-def less-ereal.simps(*2*)
real-of-ereal-eq-0 real-of-ereal-le-0)
hence ∃ *n*::*nat*. *Suc 0* ≤ *n* ∧ *real-of-ereal* (*μ B*) < −*1*/*n*
proof −
define *nw* **where** *nw* = *Suc* (*nat* (*floor* (−*1*/ (*real-of-ereal* (*μ B*)))))
have *Suc 0* ≤ *nw* **unfolding** *nw-def* **by** *simp*
have *0* < −*1*/ (*real-of-ereal* (*μ B*)) **using** ‹*real-of-ereal* (*μ B*) < *0*›
by *simp*

```
    have −1/ (real-of-ereal (μ B)) < nw unfolding nw-def by linarith
    hence 1/nw < 1/(−1/ (real-of-ereal (μ B)))
      using ‹0 < −1/ (real-of-ereal (μ B))› by (metis frac-less2
          le-eq-less-or-eq of-nat-1 of-nat-le-iff zero-less-one)
    also have ... = − (real-of-ereal (μ B)) by simp
    finally have 1/nw < − (real-of-ereal (μ B)) .
    hence real-of-ereal (μ B) < −1/nw by simp
    thus ?thesis using ‹Suc 0 ≤ nw› by auto
  qed
  from this obtain n1::nat where Suc 0 ≤ n1
    and real-of-ereal (μ B) < −1/n1 by auto
  hence ereal (real-of-ereal (μ B)) < −1/n1 using real-ereal-leq[of μ B]
    ‹μ B < 0› by simp
  moreover have μ B = real-of-ereal (μ B) using ‹μ B < 0› False
    by (metis less-ereal.simps(2) real-of-ereal.elims zero-ereal-def)
  ultimately show ?thesis  using ‹Suc 0 ≤ n1› by auto
 qed
 from this obtain n0::nat where (1::nat) ≤ n0 and μ B < −1/n0 by auto
 hence n0 ∈ {n::nat|n. (1::nat) ≤ n ∧
   (∃ B ∈ sets M. B ⊆ A ∧ μ B <  ereal(−1/n))}
   using ‹B∈ sets M› ‹B⊆ A› by auto
 thus ?thesis by auto
qed

lemma inf-neg-ge-1:
 assumes A ∈ sets M
   and ¬ pos-meas-set A
 shows (1::nat) ≤ inf-neg A
proof −
 define N where N = {n::nat|n. (1::nat) ≤ n ∧
   (∃ B ∈ sets M. B ⊆ A ∧ μ B < ereal (−1/n))}
 have N ≠ {} unfolding N-def using assms inf-neg-ne by auto
 moreover have ⋀n. n∈ N ⟹ (1::nat) ≤ n unfolding N-def by simp
 ultimately show 1 ≤ inf-neg A unfolding inf-neg-def N-def
   using Inf-nat-def1 assms(1) assms(2) by presburger
qed

lemma inf-neg-pos:
 assumes A ∈ sets M
   and ¬ pos-meas-set A
 shows ∃ B ∈ sets M. B⊆ A ∧ μ B < −1/(inf-neg A)
proof −
 define N where N = {n::nat|n. (1::nat) ≤ n ∧
   (∃ B ∈ sets M. B ⊆ A ∧ μ B < ereal (−1/n))}
 have N ≠ {} unfolding N-def using assms inf-neg-ne by auto
 hence Inf N ∈ N using Inf-nat-def1[of N] by simp
 hence inf-neg A ∈ N unfolding N-def inf-neg-def using assms by auto
 thus ?thesis unfolding N-def by auto
qed
```

definition *rep-neg* **where**
  rep-neg A = (*if* (*A* ∉ *sets M* ∨ *pos-meas-set A*) *then* {} *else*
  SOME B. *B* ∈ *sets M* ∧ *B* ⊆ *A* ∧ *μ B* ≤ *ereal* (−*1* / (*inf-neg A*)))

lemma *g-rep-neg*:
  assumes *A*∈ *sets M*
    and ¬ *pos-meas-set A*
  shows *rep-neg A* ∈ *sets M rep-neg A* ⊆ *A*
    μ (*rep-neg A*) ≤ *ereal* (−*1* / (*inf-neg A*))
proof −
  have ∃ *B*. *B* ∈ *sets M* ∧ *B*⊆ *A* ∧ *μ B* ≤ −*1* / (*inf-neg A*) **using** *assms*
      inf-neg-pos[*of A*] **by** *auto*
  from *someI-ex*[*OF this*] **show** *rep-neg A* ∈ *sets M rep-neg A* ⊆ *A*
    μ (*rep-neg A*) ≤ −*1* / (*inf-neg A*)
    unfolding *rep-neg-def* **using** *assms* **by** *auto*
qed

lemma *rep-neg-sets*:
  shows *rep-neg A* ∈ *sets M*
proof (*cases A* ∉ *sets M* ∨ *pos-meas-set A*)
  case *True*
  then show *?thesis* **unfolding** *rep-neg-def* **by** *simp*
next
  case *False*
  then show *?thesis* **using** *g-rep-neg*(*1*) **by** *blast*
qed

lemma *rep-neg-subset*:
  shows *rep-neg A* ⊆ *A*
proof (*cases A* ∉ *sets M* ∨ *pos-meas-set A*)
  case *True*
  then show *?thesis* **unfolding** *rep-neg-def* **by** *simp*
next
  case *False*
  then show *?thesis* **using** *g-rep-neg*(*2*) **by** *blast*
qed

lemma *rep-neg-less*:
  assumes *A*∈ *sets M*
    and ¬ *pos-meas-set A*
  shows *μ* (*rep-neg A*) ≤ *ereal* (−*1* / (*inf-neg A*)) **using** *assms g-rep-neg*(*3*)
  by *simp*

lemma *rep-neg-leq*:
  shows *μ* (*rep-neg A*) ≤ *0*
proof (*cases A* ∉ *sets M* ∨ *pos-meas-set A*)
  case *True*
  hence *rep-neg A* = {} **unfolding** *rep-neg-def* **by** *simp*

then show *?thesis* **using** *sgn-meas signed-measure-empty* **by** *force*
next
case *False*
then show *?thesis* **using** *rep-neg-less* **by** (*metis le-ereal-le minus-divide-left neg-le-0-iff-le of-nat-0 of-nat-le-iff zero-ereal-def zero-le zero-le-divide-1-iff*)
qed

3.2 Construction of the positive subset

fun *pos-wtn*
where
pos-wtn-base: *pos-wtn E 0 = E*|
pos-wtn-step: *pos-wtn E* (*Suc n*) = *pos-wtn E n* − *rep-neg* (*pos-wtn E n*)

lemma *pos-wtn-subset*:
shows *pos-wtn E n* ⊆ *E*
proof (*induct n*)
case *0*
then show *?case* **using** *pos-wtn-base* **by** *simp*
next
case (*Suc n*)
hence *rep-neg* (*pos-wtn E n*) ⊆ *pos-wtn E n* **using** *rep-neg-subset* **by** *simp*
then show *?case* **using** *Suc* **by** *auto*
qed

lemma *pos-wtn-sets*:
assumes *E*∈ *sets M*
shows *pos-wtn E n* ∈ *sets M*
proof (*induct n*)
case *0*
then show *?case* **using** *assms* **by** *simp*
next
case (*Suc n*)
then show *?case* **using** *pos-wtn-step rep-neg-sets* **by** *auto*
qed

definition *neg-wtn* **where**
neg-wtn E (*n*::*nat*) = *rep-neg* (*pos-wtn E n*)

lemma *neg-wtn-neg-meas*:
shows *μ* (*neg-wtn E n*) ≤ *0* **unfolding** *neg-wtn-def* **using** *rep-neg-leq* **by** *simp*

lemma *neg-wtn-sets*:
shows *neg-wtn E n* ∈ *sets M* **unfolding** *neg-wtn-def* **using** *rep-neg-sets* **by** *simp*

lemma *neg-wtn-subset*:
shows *neg-wtn E n* ⊆ *E* **unfolding** *neg-wtn-def*
using *pos-wtn-subset*[*of E n*] *rep-neg-subset*[*of pos-wtn E n*] **by** *simp*

```
lemma neg-wtn-union-subset:
  shows (⋃ i ≤ n. neg-wtn E i) ⊆ E using neg-wtn-subset by auto

lemma pos-wtn-Suc:
  shows pos-wtn E (Suc n) = E − (⋃ i ≤ n. neg-wtn E i) unfolding neg-wtn-def
proof (induct n)
  case 0
  then show ?case using pos-wtn-base pos-wtn-step by simp
next
  case (Suc n)
  have pos-wtn E (Suc (Suc n)) = pos-wtn E (Suc n) −
    rep-neg (pos-wtn E (Suc n))
    using pos-wtn-step by simp
  also have ... = (E − (⋃ i ≤ n. rep-neg (pos-wtn E i))) −
    rep-neg (pos-wtn E (Suc n))
    using Suc by simp
  also have ... = E − (⋃ i ≤ (Suc n). rep-neg (pos-wtn E i))
    using diff-union[of E λi. rep-neg (pos-wtn E i) n] by auto
  finally show pos-wtn E (Suc (Suc n)) =
    E − (⋃ i ≤ (Suc n). rep-neg (pos-wtn E i)) .
qed

definition pos-sub where
  pos-sub E = (⋂ n. pos-wtn E n)

lemma pos-sub-sets:
  assumes E∈ sets M
  shows pos-sub E ∈ sets M unfolding pos-sub-def using pos-wtn-sets assms
  by auto

lemma pos-sub-subset:
  shows pos-sub E ⊆ E unfolding pos-sub-def using pos-wtn-subset by blast

lemma pos-sub-infty:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows |μ (pos-sub E)| < ∞ using signed-measure-finite-subset assms
    pos-sub-sets pos-sub-subset by simp

lemma neg-wtn-djn:
  shows disjoint-family (λn. neg-wtn E n) unfolding disjoint-family-on-def
proof −
  {
    fix n
    fix m::nat
    assume n < m
    hence ∃ p. m = Suc p using old.nat.exhaust by auto
    from this obtain p where m = Suc p by auto
```

have *neg-wtn E m* ⊆ *pos-wtn E m* **unfolding** *neg-wtn-def*
by (*simp add*: *rep-neg-subset*)
also have ... = *E* − (⋃ *i* ≤ *p*. *neg-wtn E i*) **using** *pos-wtn-Suc* ‹*m* = *Suc p*›
by *simp*
finally have *neg-wtn E m* ⊆ *E* − (⋃ *i* ≤ *p*. *neg-wtn E i*) **.**
moreover have *neg-wtn E n* ⊆ (⋃ *i* ≤ *p*. *neg-wtn E i*) **using** ‹*n* < *m*›
‹*m* = *Suc p*› **by** (*simp add*: *UN-upper*)
ultimately have *neg-wtn E n* ∩ *neg-wtn E m* = {} **by** *auto*
}
thus ∀ *m*∈*UNIV*. ∀ *n*∈*UNIV*. *m* ≠ *n* ⟶ *neg-wtn E m* ∩ *neg-wtn E n* = {}
by (*metis inf-commute linorder-neqE-nat*)
qed
end

lemma *disjoint-family-imp-on*:
assumes *disjoint-family A*
shows *disjoint-family-on A S*
using *assms disjoint-family-on-mono subset-UNIV* **by** *blast*

context *signed-measure-space*
begin
lemma *neg-wtn-union-neg-meas*:
shows μ (⋃ *i* ≤ *n*. *neg-wtn E i*) ≤ *0*
proof −
have μ (⋃ *i* ≤ *n*. *neg-wtn E i*) = (∑ *i*∈{.. *n*}. μ (*neg-wtn E i*))
proof (*rule signed-measure-disj-sum*, *simp*+)
show *signed-measure M* μ **using** *sgn-meas* **.**
show *disjoint-family-on* (*neg-wtn E*) {..*n*} **using** *neg-wtn-djn*
disjoint-family-imp-on[*of neg-wtn E*] **by** *simp*
show ⋀*i*. *i* ∈ {..*n*} ⟹ *neg-wtn E i* ∈ *sets M* **using** *neg-wtn-sets* **by** *simp*
qed
also have ... ≤ *0* **using** *neg-wtn-neg-meas* **by** (*simp add*: *sum-nonpos*)
finally show *?thesis* **.**
qed

lemma *pos-wtn-meas-gt*:
assumes *0* < μ *E*
and *E*∈ *sets M*
shows *0* < μ (*pos-wtn E n*)
proof (*cases n* = *0*)
case *True*
then show *?thesis* **using** *assms* **by** *simp*
next
case *False*
hence ∃ *m*. *n* = *Suc m* **by** (*simp add*: *not0-implies-Suc*)
from *this* **obtain** *m* **where** *n* = *Suc m* **by** *auto*
hence *eq*: *pos-wtn E n* = *E* − (⋃ *i* ≤ *m*. *neg-wtn E i*) **using** *pos-wtn-Suc*
by *simp*
hence *pos-wtn E n* ∩ (⋃ *i* ≤ *m*. *neg-wtn E i*) = {} **by** *auto*

moreover have $E = pos\text{-}wtn\ E\ n \cup (\bigcup\ i \le m.\ neg\text{-}wtn\ E\ i)$
using *eq neg-wtn-union-subset*[*of E m*] **by** *auto*
ultimately have $\mu\ E = \mu\ (pos\text{-}wtn\ E\ n) + \mu\ (\bigcup\ i \le m.\ neg\text{-}wtn\ E\ i)$
using *signed-measure-add*[*of M μ pos-wtn E n* ⋃ *i ≤ m. neg-wtn E i*]
pos-wtn-sets neg-wtn-sets assms sgn-meas **by** *auto*
hence $0 < \mu\ (pos\text{-}wtn\ E\ n) + \mu\ (\bigcup\ i \le m.\ neg\text{-}wtn\ E\ i)$ **using** *assms* **by** *simp*
thus *?thesis* **using** *neg-wtn-union-neg-meas*
by (*metis add.right-neutral add-mono not-le*)
qed

definition *union-wit* **where**
$union\text{-}wit\ E = (\bigcup\ n.\ neg\text{-}wtn\ E\ n)$

lemma *union-wit-sets*:
shows *union-wit E* ∈ *sets M* **unfolding** *union-wit-def*
proof (*intro sigma-algebra.countable-nat-UN*)
show *sigma-algebra* (*space M*) (*sets M*)
by (*simp add*: *sets.sigma-algebra-axioms*)
show *range* (*neg-wtn E*) ⊆ *sets M*
proof −
{
fix *n*
have *neg-wtn E n* ∈ *sets M* **unfolding** *neg-wtn-def*
by (*simp add*: *rep-neg-sets*)
}
thus *?thesis* **by** *auto*
qed
qed

lemma *union-wit-subset*:
shows *union-wit E* ⊆ *E*
proof −
{
fix *n*
have *neg-wtn E n* ⊆ *E* **unfolding** *neg-wtn-def* **using** *pos-wtn-subset*
rep-neg-subset[*of pos-wtn E n*] **by** *auto*
}
thus *?thesis* **unfolding** *union-wit-def* **by** *auto*
qed

lemma *pos-sub-diff*:
shows *pos-sub E* = *E* − *union-wit E*
proof
show *pos-sub E* ⊆ *E* − *union-wit E*
proof −
have *pos-sub E* ⊆ *E* **using** *pos-sub-subset* **by** *simp*
moreover have *pos-sub E* ∩ *union-wit E* = {}
proof (*rule ccontr*)
assume *pos-sub E* ∩ *union-wit E* ≠ {}

hence ∃ *a*. *a*∈ *pos-sub E* ∩ *union-wit E* **by** *auto*
from *this* **obtain** *a* **where** *a*∈ *pos-sub E* ∩ *union-wit E* **by** *auto*
hence *a*∈ *union-wit E* **by** *simp*
hence ∃ *n*. *a* ∈ *rep-neg* (*pos-wtn E n*) **unfolding** *union-wit-def neg-wtn-def*
by *auto*
from *this* **obtain** *n* **where** *a* ∈ *rep-neg* (*pos-wtn E n*) **by** *auto*
have *a* ∈ *pos-wtn E* (*Suc n*) **using** ‹*a*∈ *pos-sub E* ∩ *union-wit E*›
unfolding *pos-sub-def* **by** *blast*
hence *a*∉ *rep-neg* (*pos-wtn E n*) **using** *pos-wtn-step* **by** *simp*
thus *False* **using** ‹*a* ∈ *rep-neg* (*pos-wtn E n*)› **by** *simp*
qed
ultimately show *?thesis* **by** *auto*
qed
next
show *E* − *union-wit E* ⊆ *pos-sub E*
proof
fix *a*
assume *a* ∈ *E* − *union-wit E*
show *a* ∈ *pos-sub E* **unfolding** *pos-sub-def*
proof
fix *n*
show *a* ∈ *pos-wtn E n*
proof (*cases n* = *0*)
case *True*
thus *?thesis* **using** *pos-wtn-base* ‹*a*∈ *E* − *union-wit E*› **by** *simp*
next
case *False*
hence ∃ *m*. *n* = *Suc m* **by** (*simp add*: *not0-implies-Suc*)
from *this* **obtain** *m* **where** *n* = *Suc m* **by** *auto*
have (⋃ *i* ≤ *m*. *rep-neg* (*pos-wtn E i*)) ⊆
(⋃ *n*. (*rep-neg* (*pos-wtn E n*))) **by** *auto*
hence *a* ∈ *E* − (⋃ *i* ≤ *m*. *rep-neg* (*pos-wtn E i*))
using ‹*a* ∈ *E* − *union-wit E*› **unfolding** *union-wit-def neg-wtn-def*
by *auto*
thus *a*∈ *pos-wtn E n* **using** *pos-wtn-Suc* ‹*n* = *Suc m*›
unfolding *neg-wtn-def* **by** *simp*
qed
qed
qed
qed

definition *num-wtn* **where**
num-wtn E n = *inf-neg* (*pos-wtn E n*)

lemma *num-wtn-geq*:
shows *μ* (*neg-wtn E n*) ≤ *ereal* (−*1*/(*num-wtn E n*))
proof (*cases* (*pos-wtn E n*) ∉ *sets M* ∨ *pos-meas-set* (*pos-wtn E n*))
case *True*
hence *neg-wtn E n* = {} **unfolding** *neg-wtn-def rep-neg-def* **by** *simp*

moreover have *num-wtn E n* = *0* **using** *True* **unfolding** *num-wtn-def inf-neg-def*

by *simp*
ultimately show *?thesis* **using** *sgn-meas signed-measure-empty* **by** *force*
next
case *False*
then show *?thesis* **using** *g-rep-neg*(*3*)[*of pos-wtn E n*] **unfolding** *neg-wtn-def num-wtn-def* **by** *simp*
qed

lemma *neg-wtn-infty*:
assumes $E \in$ *sets M*
and $|\mu\ E| < \infty$
shows $|\mu$ (*neg-wtn E i*)$| < \infty$
proof (*rule signed-measure-finite-subset*)
show $E \in$ *sets M* $|\mu\ E| < \infty$ **using** *assms* **by** *auto*
show *neg-wtn E i* $\in$ *sets M*
proof (*cases pos-wtn E i* $\notin$ *sets M* $\vee$ *pos-meas-set* (*pos-wtn E i*))
case *True*
then show *?thesis* **unfolding** *neg-wtn-def rep-neg-def* **by** *simp*
next
case *False*
then show *?thesis* **unfolding** *neg-wtn-def*
using *g-rep-neg*(*1*)[*of pos-wtn E i*] **by** *simp*
qed
show *neg-wtn E i* $\subseteq$ *E* **unfolding** *neg-wtn-def* **using** *pos-wtn-subset*[*of E*] *rep-neg-subset*[*of pos-wtn E i*] **by** *auto*
qed

lemma *union-wit-infty*:
assumes $E \in$ *sets M*
and $|\mu\ E| < \infty$
shows $|\mu$ (*union-wit E*)$| < \infty$ **using** *union-wit-subset union-wit-sets signed-measure-finite-subset assms* **unfolding** *union-wit-def* **by** *simp*

lemma *neg-wtn-summable*:
assumes $E \in$ *sets M*
and $|\mu\ E| < \infty$
shows *summable* ($\lambda i. -$ *real-of-ereal* (μ (*neg-wtn E i*)))
proof $-$
have *signed-measure M* μ **using** *sgn-meas* **.**
moreover have *range* (*neg-wtn E*) $\subseteq$ *sets M* **unfolding** *neg-wtn-def*
using *rep-neg-sets* **by** *auto*
moreover have *disjoint-family* (*neg-wtn E*) **using** *neg-wtn-djn* **by** *simp*
moreover have $\bigcup$ (*range* (*neg-wtn E*)) $\in$ *sets M* **using** *union-wit-sets*
unfolding *union-wit-def* **by** *simp*
moreover have $|\mu$ ($\bigcup$ (*range* (*neg-wtn E*)))$| < \infty$
using *union-wit-subset signed-measure-finite-subset union-wit-sets assms*
unfolding *union-wit-def* **by** *simp*

```
  ultimately have summable (λi. real-of-ereal |μ (neg-wtn E i)|)
    using signed-measure-abs-convergent[of M ] by simp
  moreover have ⋀i. |μ (neg-wtn E i)| = −(μ (neg-wtn E i))
  proof −
    fix i
    have μ (neg-wtn E i) ≤ 0 using rep-neg-leq[of pos-wtn E i]
      unfolding neg-wtn-def .
    thus |μ (neg-wtn E i)| = −μ (neg-wtn E i) using less-eq-ereal-def by auto
  qed
  ultimately show ?thesis by simp
qed

lemma inv-num-wtn-summable:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows summable (λn. 1/(num-wtn E n))
proof (rule summable-bounded)
  show ⋀i. 0 ≤ 1 / real (num-wtn E i) by simp
  show ⋀i. 1 / real (num-wtn E i) ≤ (λn. −real-of-ereal (μ (neg-wtn E n))) i
  proof −
    fix i
    have |μ (neg-wtn E i)| < ∞ using assms neg-wtn-infty by simp
    have ereal (1/(num-wtn E i)) ≤ −μ (neg-wtn E i) using num-wtn-geq[of E i]
        ereal-minus-le-minus by fastforce
    also have ... = ereal(− real-of-ereal (μ (neg-wtn E i)))
      using ‹|μ (neg-wtn E i)| < ∞› ereal-real′ by auto
    finally have ereal (1/(num-wtn E i)) ≤
      ereal(− real-of-ereal (μ (neg-wtn E i))) .
    thus 1 / real (num-wtn E i) ≤ −real-of-ereal (μ (neg-wtn E i)) by simp
  qed
  show summable (λi. − real-of-ereal (μ (neg-wtn E i)))
    using assms neg-wtn-summable by simp
qed

lemma inv-num-wtn-shift-summable:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows summable (λn. 1/(num-wtn E n − 1))
proof (rule sum-shift-denum)
  show summable (λn. 1 / real (num-wtn E n)) using assms inv-num-wtn-summable
    by simp
qed

lemma neg-wtn-meas-sums:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows (λi. − (μ (neg-wtn E i))) sums
  suminf (λi. − real-of-ereal (μ (neg-wtn E i)))
proof −
```

have ($\lambda i.$ *ereal* ($-$ *real-of-ereal* (μ (*neg-wtn E i*)))) *sums*
suminf ($\lambda i.$ $-$ *real-of-ereal* (μ (*neg-wtn E i*)))
proof (*rule sums-ereal*[*THEN iffD2*])
have *summable* ($\lambda i.$ $-$ *real-of-ereal* (μ (*neg-wtn E i*)))
using *neg-wtn-summable assms* **by** *simp*
thus ($\lambda x.$ $-$ *real-of-ereal* (μ (*neg-wtn E x*)))
sums ($\sum i.$ $-$ *real-of-ereal* (μ (*neg-wtn E i*)))
by *auto*
qed
moreover have $\bigwedge i.$ μ (*neg-wtn E i*) = *ereal* (*real-of-ereal* (μ (*neg-wtn E i*)))
proof $-$
fix *i*
show μ (*neg-wtn E i*) = *ereal* (*real-of-ereal* (μ (*neg-wtn E i*)))
using *assms*(*1*) *assms*(*2*) *ereal-real′ neg-wtn-infty* **by** *auto*
qed
ultimately show *?thesis*
by (*metis* (*no-types*, *lifting*) *sums-cong uminus-ereal.simps*(*1*))
qed

lemma *neg-wtn-meas-suminf-le*:
assumes $E \in$ *sets M*
and $|\mu\ E| < \infty$
shows *suminf* ($\lambda i.$ μ (*neg-wtn E i*)) $\leq -$ *suminf* ($\lambda n.$ *1*/(*num-wtn E n*))
proof $-$
have *suminf* ($\lambda n.$ *1*/(*num-wtn E n*)) $\leq$
suminf ($\lambda i.$ $-$*real-of-ereal* (μ (*neg-wtn E i*)))
proof (*rule suminf-le*)
show *summable* ($\lambda n.$ *1* / *real* (*num-wtn E n*)) **using** *assms*
inv-num-wtn-summable[*of E*]
summable-minus[*of* $\lambda n.$ *1* / *real* (*num-wtn E n*)] **by** *simp*
show *summable* ($\lambda i.$ $-$*real-of-ereal* (μ (*neg-wtn E i*)))
using *neg-wtn-summable assms*
summable-minus[*of* $\lambda i.$ *real-of-ereal* (μ (*neg-wtn E i*))]
by (*simp add*: *summable-minus-iff*)
show $\bigwedge n.$ *1* / *real* (*num-wtn E n*) $\leq -$*real-of-ereal* (μ (*neg-wtn E n*))
proof $-$
fix *n*
have μ (*neg-wtn E n*) $\leq$ *ereal* ($-$ *1* / *real* (*num-wtn E n*))
using *num-wtn-geq* **by** *simp*
hence *ereal* (*1*/ *real* (*num-wtn E n*)) $\leq - \mu$ (*neg-wtn E n*)
by (*metis add.inverse-inverse eq-iff ereal-uminus-le-reorder linear*
minus-divide-left uminus-ereal.simps(*1*))
have *real-of-ereal* (*ereal* (*1* / *real* (*num-wtn E n*))) $\leq$
real-of-ereal ($- \mu$ (*neg-wtn E n*))
proof (*rule real-of-ereal-positive-mono*)
show *0* $\leq$ *ereal* (*1* / *real* (*num-wtn E n*)) **by** *simp*
show *ereal* (*1* / *real* (*num-wtn E n*)) $\leq - \mu$ (*neg-wtn E n*)
using ‹*ereal* (*1* / *real* (*num-wtn E n*)) $\leq - \mu$ (*neg-wtn E n*)› **.**
show $- \mu$ (*neg-wtn E n*) $\neq \infty$ **using** *neg-wtn-infty*[*of E n*] *assms* **by** *auto*

```
      qed
      thus (1 / real (num-wtn E n)) ≤ −real-of-ereal ( μ (neg-wtn E n))
        by simp
    qed
  qed
  also have ... = − suminf (λi. real-of-ereal (μ (neg-wtn E i)))
  proof (rule suminf-minus)
    show summable (λn. real-of-ereal (μ (neg-wtn E n)))
      using neg-wtn-summable assms
        summable-minus[of λi. real-of-ereal (μ (neg-wtn E i))]
      by (simp add: summable-minus-iff)
  qed
  finally have suminf (λn. 1/(num-wtn E n)) ≤
    − suminf (λi. real-of-ereal (μ (neg-wtn E i))) .
  hence a: suminf (λi. real-of-ereal (μ (neg-wtn E i))) ≤
    − suminf (λn. 1/(num-wtn E n)) by simp
  show suminf (λi. (μ (neg-wtn E i))) ≤ ereal (−suminf (λn. 1/(num-wtn E n)))

  proof −
    have sumeq: suminf (λi. ereal (real-of-ereal (μ (neg-wtn E i)))) =
      suminf (λi. (real-of-ereal (μ (neg-wtn E i))))
    proof (rule sums-suminf-ereal)
      have summable (λi. −real-of-ereal (μ (neg-wtn E i)))
        using neg-wtn-summable assms
          summable-minus[of λi. real-of-ereal (μ (neg-wtn E i))]
        by (simp add: summable-minus-iff)
      thus (λi. real-of-ereal (μ (neg-wtn E i))) sums
        (∑ i. real-of-ereal (μ (neg-wtn E i)))
        using neg-wtn-summable[of E] assms summable-minus-iff by blast
    qed
    hence suminf (λi. μ (neg-wtn E i)) =
      suminf (λi. (real-of-ereal (μ (neg-wtn E i))))
    proof −
      have ⋀i. ereal (real-of-ereal (μ (neg-wtn E i))) = μ (neg-wtn E i)
      proof −
        fix i
        show ereal (real-of-ereal (μ (neg-wtn E i))) = μ (neg-wtn E i)
          using neg-wtn-infty[of E] assms by (simp add: ereal-real′)
      qed
      thus ?thesis using sumeq by auto
    qed
    thus ?thesis using a by simp
  qed
qed

lemma union-wit-meas-le:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows μ (union-wit E) ≤ − suminf (λn. 1 / real (num-wtn E n))
```

proof −
  have *μ* (*union-wit E*) = *μ* ($\bigcup$ (*range* (*neg-wtn E*))) **unfolding** *union-wit-def*
    by *simp*
  also have ... = ($\sum$ *i*. *μ* (*neg-wtn E i*))
  proof (*rule signed-measure-inf-sum*[*symmetric*])
    show *signed-measure M μ* **using** *sgn-meas* **.**
    show *range* (*neg-wtn E*) ⊆ *sets M*
      by (*simp add*: *image-subset-iff neg-wtn-def rep-neg-sets*)
    show *disjoint-family* (*neg-wtn E*) **using** *neg-wtn-djn* **by** *simp*
    show $\bigcup$ (*range* (*neg-wtn E*)) ∈ *sets M* **using** *union-wit-sets*
      unfolding *union-wit-def* **by** *simp*
  qed
  also have ... ≤ − *suminf* (*λn*. *1* / *real* (*num-wtn E n*))
    using *assms neg-wtn-meas-suminf-le* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *pos-sub-pos-meas*:
  assumes *E* ∈ *sets M*
    and |*μ E*| < ∞
    and *0* < *μ E*
    and ¬ *pos-meas-set E*
  shows *0* < *μ* (*pos-sub E*)
proof −
  have *0* < *μ E* **using** *assms* **by** *simp*
  also have ... = *μ* (*pos-sub E*) + *μ* (*union-wit E*)
  proof −
    have *E* = *pos-sub E* ∪ (*union-wit E*)
      using *pos-sub-diff*[*of E*] *union-wit-subset* **by** *force*
    moreover have *pos-sub E* ∩ *union-wit E* = {}
      using *pos-sub-diff* **by** *auto*
    ultimately show *?thesis*
      using *signed-measure-add*[*of M μ pos-sub E union-wit E*]
        pos-sub-sets union-wit-sets assms sgn-meas **by** *simp*
  qed
  also have ... ≤ *μ* (*pos-sub E*) + (− *suminf* (*λn*. *1* / *real* (*num-wtn E n*)))
  proof −
    have *μ* (*union-wit E*) ≤ − *suminf* (*λn*. *1* / *real* (*num-wtn E n*))
      using *union-wit-meas-le*[*of E*] *assms* **by** *simp*
    thus *?thesis* **using** *union-wit-infty assms* **using** *add-left-mono* **by** *blast*
  qed
  also have ... = *μ* (*pos-sub E*) − *suminf* (*λn*. *1* / *real* (*num-wtn E n*))
    by (*simp add*: *minus-ereal-def*)
  finally have *0* < *μ* (*pos-sub E*) − *suminf* (*λn*. *1* / *real* (*num-wtn E n*)) **.**
  moreover have *0* < *suminf* (*λn*. *1* / *real* (*num-wtn E n*))
  proof (*rule suminf-pos2*)
    show *0* < *1* / *real* (*num-wtn E 0*)
      using *inf-neg-ge-1*[*of E*] *assms pos-wtn-base* **unfolding** *num-wtn-def* **by** *simp*
    show $\bigwedge$*n*. *0* ≤ *1* / *real* (*num-wtn E n*) **by** *simp*

```
    show summable (λn. 1 / real (num-wtn E n))
      using assms inv-num-wtn-summable by simp
  qed
  ultimately show ?thesis using pos-sub-infty assms by fastforce
qed

lemma num-wtn-conv:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows (λn. 1/(num-wtn E n)) ⟶ 0
proof (rule summable-LIMSEQ-zero)
  show summable (λn. 1 / real (num-wtn E n))
    using assms inv-num-wtn-summable by simp
qed

lemma num-wtn-shift-conv:
  assumes E ∈ sets M
    and |μ E| < ∞
  shows (λn. 1/(num-wtn E n − 1)) ⟶ 0
proof (rule summable-LIMSEQ-zero)
  show summable (λn. 1 / real (num-wtn E n − 1))
    using assms inv-num-wtn-shift-summable by simp
qed

lemma inf-neg-E-set:
  assumes 0 < inf-neg E
  shows E ∈ sets M using assms unfolding inf-neg-def by presburger

lemma inf-neg-pos-meas:
  assumes 0 < inf-neg E
  shows ¬ pos-meas-set E using assms unfolding inf-neg-def by presburger

lemma inf-neg-mem:
  assumes 0 < inf-neg E
  shows inf-neg E ∈ {n::nat|n. (1::nat) ≤ n ∧
    (∃ B ∈ sets M. B ⊆ E ∧ μ B < ereal (−1/n))}
proof −
  have E ∈ sets M using assms unfolding inf-neg-def by presburger
  moreover have ¬ pos-meas-set E using assms unfolding inf-neg-def
    by presburger
  ultimately have {n::nat|n. (1::nat) ≤ n ∧
    (∃ B ∈ sets M. B ⊆ E ∧ μ B < ereal (−1/n))} ≠ {}
    using inf-neg-ne[of E] by simp
  thus ?thesis unfolding inf-neg-def
    by (meson Inf-nat-def1 ‹E ∈ sets M› ‹¬ pos-meas-set E›)
qed

lemma prec-inf-neg-pos:
  assumes 0 < inf-neg E − 1
```

```
    and B ∈ sets M
    and B⊆ E
  shows −1/(inf-neg E − 1) ≤ μ B
proof (rule ccontr)
  define S where S = {p::nat|p. (1::nat) ≤ p ∧
    (∃ B ∈ sets M. B ⊆ E ∧ μ B < ereal (−1/p))}
  assume ¬ ereal (− 1 / real (inf-neg E − 1)) ≤ μ B
  hence μ B < −1/(inf-neg E − 1) by auto
  hence inf-neg E − 1∈ S unfolding S-def using assms by auto
  have Suc 0 < inf-neg E using assms by simp
  hence inf-neg E ∈ S unfolding  S-def using inf-neg-mem[of E] by simp
  hence S ≠ {} by auto
  have inf-neg E = Inf S unfolding S-def inf-neg-def
    using assms inf-neg-E-set inf-neg-pos-meas by auto
  have  inf-neg E − 1 < inf-neg E using assms by simp
  hence inf-neg E −1 ∉ S
    using cInf-less-iff[of S] ‹S ≠ {}› ‹inf-neg E = Inf S› by auto
  thus False using ‹inf-neg E − 1 ∈ S› by simp
qed

lemma pos-wtn-meas-ge:
  assumes E ∈ sets M
    and |μ E| < ∞
    and C∈ sets M
    and ⋀n. C⊆ pos-wtn E n
    and ⋀n. 0 < num-wtn E n
  shows ∃ N. ∀ n≥ N. − 1/ (num-wtn E n − 1) ≤ μ C
proof −
  have ∃ N. ∀ n≥ N. 1/(num-wtn E n) < 1/2 using num-wtn-conv[of E]
      conv-0-half[of λn. 1 / real (num-wtn E n)] assms by simp
  from this obtain N where ∀ n≥ N. 1/(num-wtn E n) < 1/2 by auto
  {
    fix n
    assume N ≤ n
    hence 1/(num-wtn E n) < 1/2 using ‹∀ n≥ N. 1/(num-wtn E n) < 1/2› by
simp
    have 1/(1/2) < 1/(1/(num-wtn E n))
    proof (rule frac-less2, auto)
      show 2 / real (num-wtn E n) < 1 using ‹1/(num-wtn E n) < 1/2›
        by linarith
      show 0 < num-wtn E n unfolding num-wtn-def using inf-neg-ge-1 assms
        by (simp add: num-wtn-def)
    qed
    hence 2 < (num-wtn E n) by simp
    hence Suc 0 < num-wtn E n − 1 unfolding num-wtn-def by simp
    hence − 1/ (num-wtn E n − 1) ≤ μ C using assms prec-inf-neg-pos
      unfolding num-wtn-def by simp
  }
  thus ?thesis by auto
```

qed

lemma *pos-sub-pos-meas-subset*:
  assumes $E \in$ *sets M*
    and $|\mu\ E| < \infty$
    and $C \in$ *sets M*
    and $C \subseteq$ (*pos-sub E*)
    and $\bigwedge n.\ 0 <$ *num-wtn E n*
  shows $0 \le \mu\ C$
proof −
  have $\bigwedge n.\ C \subseteq$ *pos-wtn E n* **using** *assms* **unfolding** *pos-sub-def* **by** *auto*
  hence $\exists N.\ \forall n \ge N.\ -1/$ (*num-wtn E n* − *1*) $\le \mu\ C$ **using** *assms*
    pos-wtn-meas-ge[*of E C*] **by** *simp*
  from *this* **obtain** *N* **where** *Nprop*: $\forall n \ge N.\ -1/$ (*num-wtn E n* − *1*) $\le \mu\ C$
by *auto*
  show $0 \le \mu\ C$
  proof (*rule lim-mono*)
    show $\bigwedge n.\ N \le n \Longrightarrow -1/$ (*num-wtn E n* − *1*) $\le (\lambda n.\ \mu\ C)\ n$
      using *Nprop* **by** *simp*
    have ($\lambda n.$ (*1* / *real* (*num-wtn E n* − *1*))) $\longrightarrow$ *0*
      using *assms num-wtn-shift-conv*[*of E*] **by** *simp*
    hence ($\lambda n.$ (− *1* / *real* (*num-wtn E n* − *1*))) $\longrightarrow$ *0*
      using *tendsto-minus*[*of* $\lambda n.$ *1* / *real* (*num-wtn E n* − *1*) *0*] **by** *simp*
    thus ($\lambda n.$ *ereal* (− *1* / *real* (*num-wtn E n* − *1*))) $\longrightarrow$ *0*
      by (*simp add*: *zero-ereal-def*)
    show ($\lambda n.\ \mu\ C$) $\longrightarrow \mu\ C$ **by** *simp*
  qed
qed

lemma *pos-sub-pos-meas'*:
  assumes $E \in$ *sets M*
    and $|\mu\ E| < \infty$
    and $0 < \mu\ E$
    and $\forall n.\ 0 <$ *num-wtn E n*
  shows $0 < \mu$ (*pos-sub E*)
proof −
  have $0 < \mu\ E$ **using** *assms* **by** *simp*
  also have ... $= \mu$ (*pos-sub E*) $+ \mu$ (*union-wit E*)
  proof −
    have $E =$ *pos-sub E* $\cup$ (*union-wit E*)
      using *pos-sub-diff*[*of E*] *union-wit-subset* **by** *force*
    moreover have *pos-sub E* $\cap$ *union-wit E* = {}
      using *pos-sub-diff* **by** *auto*
    ultimately show *?thesis*
      using *signed-measure-add*[*of M* μ *pos-sub E union-wit E*]
        pos-sub-sets union-wit-sets assms sgn-meas **by** *simp*
  qed
  also have ... $\le \mu$ (*pos-sub E*) + (− *suminf* ($\lambda n.$ *1* / *real* (*num-wtn E n*)))
  proof −

have *μ* (*union-wit E*) ≤ − *suminf* (*λn. 1* / *real* (*num-wtn E n*))
using *union-wit-meas-le*[*of E*] *assms* **by** *simp*
thus *?thesis* **using** *union-wit-infty assms* **using** *add-left-mono* **by** *blast*
qed
also have ... = *μ* (*pos-sub E*) − *suminf* (*λn. 1* / *real* (*num-wtn E n*))
by (*simp add*: *minus-ereal-def*)
finally have *0* < *μ* (*pos-sub E*) − *suminf* (*λn. 1* / *real* (*num-wtn E n*)) **.**
moreover have *0* < *suminf* (*λn. 1* / *real* (*num-wtn E n*))
proof (*rule suminf-pos2*)
show *0* < *1* / *real* (*num-wtn E 0*) **using** *assms* **by** *simp*
show ⋀*n. 0* ≤ *1* / *real* (*num-wtn E n*) **by** *simp*
show *summable* (*λn. 1* / *real* (*num-wtn E n*))
using *assms inv-num-wtn-summable* **by** *simp*
qed
ultimately show *?thesis* **using** *pos-sub-infty assms* **by** *fastforce*
qed

We obtain the main result of this part on the existence of a positive subset.

lemma *exists-pos-meas-subset*:
assumes *E* ∈ *sets M*
and |*μ E*| < ∞
and *0* < *μ E*
shows ∃ *A. A* ⊆ *E* ∧ *pos-meas-set A* ∧ *0* < *μ A*
proof (*cases* ∀ *n. 0* < *num-wtn E n*)
case *True*
have *pos-meas-set* (*pos-sub E*)
proof (*rule pos-meas-setI*)
show *pos-sub E* ∈ *sets M* **by** (*simp add*: *assms*(*1*) *pos-sub-sets*)
fix *A*
assume *A* ∈ *sets M* **and** *A*⊆ *pos-sub E*
thus *0* ≤ *μ A* **using** *assms True pos-sub-pos-meas-subset*[*of E*] **by** *simp*
qed
moreover have *0* < *μ* (*pos-sub E*)
using *pos-sub-pos-meas'*[*of E*] *True assms* **by** *simp*
ultimately show *?thesis* **using** *pos-meas-set-def* **by** (*metis pos-sub-subset*)
next
case *False*
hence ∃ *n. num-wtn E n* = *0* **by** *simp*
from *this* **obtain** *n* **where** *num-wtn E n* = *0* **by** *auto*
hence *pos-wtn E n* ∉ *sets M* ∨ *pos-meas-set* (*pos-wtn E n*)
using *inf-neg-ge-1* **unfolding** *num-wtn-def* **by** *fastforce*
hence *pos-meas-set* (*pos-wtn E n*) **using** *assms*
by (*simp add*: ‹*E* ∈ *sets M*› *pos-wtn-sets*)
moreover have *0* < *μ* (*pos-wtn E n*) **using** *pos-wtn-meas-gt assms* **by** *simp*
ultimately show *?thesis* **using** *pos-meas-set-def* **by** (*meson pos-wtn-subset*)
qed

4 The Hahn decomposition theorem

definition *seq-meas* **where**
  seq-meas = (*SOME f*. *incseq f* ∧ *range f* ⊆ *pos-img* ∧ ⨆ *pos-img* = ⨆ *range f*)

lemma *seq-meas-props*:
  shows *incseq seq-meas* ∧ *range seq-meas* ⊆ *pos-img* ∧
    ⨆ *pos-img* = ⨆ *range seq-meas*
proof –
  have *ex*: ∃*f*. *incseq f* ∧ *range f* ⊆ *pos-img* ∧ ⨆ *pos-img* = ⨆ *range f*
  proof (*rule Extended-Real.Sup-countable-SUP*)
    show *pos-img* ≠ {}
    proof –
      have {} ∈ *pos-sets* **using** *empty-pos-meas-set* **unfolding** *pos-sets-def*
        by *simp*
      hence μ {} ∈ *pos-img* **unfolding** *pos-img-def* **by** *auto*
      thus *?thesis* **by** *auto*
    qed
  qed
  let *?V* = *SOME f*. *incseq f* ∧ *range f* ⊆ *pos-img* ∧ ⨆ *pos-img* = ⨆ *range f*
  have *vprop*: *incseq ?V* ∧ *range ?V* ⊆ *pos-img* ∧ ⨆ *pos-img* = ⨆ *range ?V*
    using *someI-ex*[*of* λ*f*. *incseq f* ∧ *range f* ⊆ *pos-img* ∧
      ⨆ *pos-img* = ⨆ *range f*] *ex* **by** *blast*
  show *?thesis* **using** *seq-meas-def vprop* **by** *presburger*
qed

definition *seq-meas-rep* **where**
  seq-meas-rep n = (*SOME A*. *A*∈ *pos-sets* ∧ *seq-meas n* = μ *A*)

lemma *seq-meas-rep-ex*:
  shows *seq-meas-rep n* ∈ *pos-sets* ∧ μ (*seq-meas-rep n*) = *seq-meas n*
proof –
  have *ex*: ∃ *A*. *A* ∈ *pos-sets* ∧ *seq-meas n* = μ *A* **using** *seq-meas-props*
    by (*smt* (*verit*) *UNIV-I image-subset-iff mem-Collect-eq pos-img-def*)
  let *?V* = *SOME A*. *A*∈ *pos-sets* ∧ *seq-meas n* = μ *A*
  have *vprop*: *?V*∈ *pos-sets* ∧ *seq-meas n* = μ *?V* **using**
      someI-ex[*of* λ*A*. *A*∈ *pos-sets* ∧ *seq-meas n* = μ *A*] **using** *ex* **by** *blast*
  show *?thesis* **using** *seq-meas-rep-def vprop* **by** *fastforce*
qed

lemma *seq-meas-rep-pos*:
  assumes ∀ *E* ∈ *sets M*. μ *E* < ∞
  shows *pos-meas-set* (⋃ *i*. *seq-meas-rep i*)
proof (*rule pos-meas-set-Union*)
  show ⋀*i*. *pos-meas-set* (*seq-meas-rep i*)
    using *seq-meas-rep-ex signed-measure-space.pos-sets-def*
      signed-measure-space-axioms **by** *auto*
  then show ⋀*i*. *seq-meas-rep i* ∈ *sets M*
    by (*simp add*: *pos-meas-setD1*)

```
  show |μ (⋃ (range seq-meas-rep))| < ∞
  proof −
    have (⋃ (range seq-meas-rep)) ∈ sets M
    proof (rule sigma-algebra.countable-Union)
      show sigma-algebra (space M) (sets M)
        by (simp add: sets.sigma-algebra-axioms)
      show countable (range seq-meas-rep) by simp
      show range seq-meas-rep ⊆ sets M
        by (simp add: ‹⋀i. seq-meas-rep i ∈ sets M› image-subset-iff)
    qed
    hence μ (⋃ (range seq-meas-rep)) ≥ 0
      using ‹⋀i. pos-meas-set (seq-meas-rep i)› ‹⋀i. seq-meas-rep i ∈ sets M›
        signed-measure-space.pos-meas-set-pos-lim signed-measure-space-axioms
      by blast
    thus ?thesis using assms ‹⋃ (range seq-meas-rep) ∈ sets M› abs-ereal-ge0
      by simp
  qed
qed

lemma sup-seq-meas-rep:
  assumes ∀ E ∈ sets M. μ E < ∞
    and S = (⨆ pos-img)
    and A = (⋃ i. seq-meas-rep i)
  shows μ A = S
proof −
  have pms: pos-meas-set (⋃ i. seq-meas-rep i)
    using assms seq-meas-rep-pos by simp
  hence μ A ≤ S
    by (metis (mono-tags, lifting) Sup-upper ‹S = ⨆ pos-img› mem-Collect-eq
        pos-img-def pos-meas-setD1 pos-sets-def assms(2) assms(3))
  have ∀ n. (μ A = μ (A − seq-meas-rep n) + μ (seq-meas-rep n))
  proof
    fix n
    have A = (A − seq-meas-rep n) ∪ seq-meas-rep n
      using ‹A = ⋃ (range seq-meas-rep)› by blast
    hence μ A = μ ((A − seq-meas-rep n) ∪ seq-meas-rep n) by simp
    also have ... = μ (A − seq-meas-rep n) + μ (seq-meas-rep n)
    proof (rule signed-measure-add)
      show signed-measure M μ using sgn-meas by simp
      show seq-meas-rep n ∈ sets M
        using pos-sets-def seq-meas-rep-ex by auto
      then show A − seq-meas-rep n ∈ sets M
        by (simp add: assms pms pos-meas-setD1 sets.Diff)
      show (A − seq-meas-rep n) ∩ seq-meas-rep n = {} by auto
    qed
    finally show μ A = μ (A − seq-meas-rep n) + μ (seq-meas-rep n).
  qed
  have ∀ n. μ A ≥ μ (seq-meas-rep n)
  proof
```

```
    fix n
    have μ A ≥ 0 using pms assms unfolding pos-meas-set-def by auto
    have (A − seq-meas-rep n) ⊆ A by simp
    hence pos-meas-set (A − seq-meas-rep n)
    proof −
      have (A − seq-meas-rep n) ∈ sets M
        using pms assms pos-meas-setD1 pos-sets-def seq-meas-rep-ex by auto
      thus ?thesis using pms assms unfolding pos-meas-set-def by auto
    qed
    hence μ (A − seq-meas-rep n) ≥ 0 unfolding pos-meas-set-def by auto
    thus μ (seq-meas-rep n) ≤ μ A
      using ‹∀ n. (μ A = μ (A − seq-meas-rep n) + μ (seq-meas-rep n))›
      by (metis ereal-le-add-self2)
  qed
  hence μ A ≥ (⨆ range seq-meas) by (simp add: Sup-le-iff seq-meas-rep-ex)
  moreover have S = (⨆ range seq-meas)
    using seq-meas-props ‹S = (⨆ pos-img)› by simp
  ultimately have μ A ≥ S by simp
  thus μ A = S using ‹μ A ≤ S› by simp
qed

lemma seq-meas-rep-compl:
  assumes ∀ E ∈ sets M. μ E < ∞
    and A = (⋃ i. seq-meas-rep i)
  shows neg-meas-set ((space M) − A) unfolding neg-meas-set-def
proof (rule ccontr)
  assume asm: ¬ (space M − A ∈ sets M ∧
    (∀ Aa∈sets M. Aa ⊆ space M − A ⟶ μ Aa ≤ 0))
  define S where S = (⨆ pos-img)
  have pos-meas-set A  using assms seq-meas-rep-pos by simp
  have μ A = S using sup-seq-meas-rep assms  S-def by simp
  hence S < ∞ using assms ‹pos-meas-set A› pos-meas-setD1 by blast
  have (space M − A ∈ sets M)
    by (simp add: ‹pos-meas-set A› pos-meas-setD1 sets.compl-sets)
  hence  ¬(∀ Aa∈sets M. Aa ⊆ space M − A ⟶ μ Aa ≤ 0) using asm by blast
  hence ∃  E ∈ sets M. E ⊆ ((space M) − A) ∧ μ E > 0
    by (metis less-eq-ereal-def linear)
  from this obtain E where E ∈ sets M and E ⊆ ((space M) − A) and
    μ E > 0 by auto
  have ∃  A0 ⊆ E. pos-meas-set A0 ∧ μ A0 > 0
  proof (rule exists-pos-meas-subset)
    show E ∈ sets M using ‹E ∈ sets M› by simp
    show 0 < μ E using ‹μ E > 0› by simp
    show |μ E| < ∞
    proof −
      have μ E < ∞ using assms ‹E ∈ sets M› by simp
      moreover have − ∞ < μ E using ‹0 < μ E› by simp
      ultimately show ?thesis
        by (meson ereal-infty-less(1) not-inftyI)
```

```
    qed
  qed
  from this obtain A0 where A0 ⊆ E and pos-meas-set A0 and  μ A0 > 0
    by auto
  have pos-meas-set (A ∪ A0)
    using pos-meas-set-union ‹pos-meas-set A0› ‹pos-meas-set A› by simp
  have μ (A ∪ A0) = μ A + μ A0
  proof (rule signed-measure-add)
    show signed-measure M μ using sgn-meas by simp
    show A ∈ sets M using ‹pos-meas-set A›
      unfolding pos-meas-set-def by simp
    show A0 ∈ sets M using ‹pos-meas-set A0›
      unfolding pos-meas-set-def by simp
    show (A ∩ A0) = {} using ‹A0 ⊆ E› ‹E ⊆ ((space M) − A)› by auto
  qed
  then have μ (A ∪ A0) > S
    using ‹μ A = S› ‹μ A0 > 0›
    by (metis ‹S < ∞› ‹pos-meas-set (A ∪ A0)› abs-ereal-ge0 ereal-between(2)
        not-inftyI not-less-iff-gr-or-eq pos-meas-self)
  have (A ∪ A0) ∈ pos-sets
  proof −
    have  (A ∪ A0) ∈ sets M using sigma-algebra.countable-Union
      by (simp add: ‹pos-meas-set (A ∪ A0)› pos-meas-setD1)
     moreover have pos-meas-set (A ∪ A0) using ‹pos-meas-set (A ∪ A0)› by
simp
    ultimately show ?thesis unfolding pos-sets-def by simp
  qed
  then have μ (A ∪ A0) ∈ pos-img unfolding pos-img-def by auto
 show False using ‹μ (A ∪ A0) > S› ‹μ (A ∪ A0) ∈ pos-img› ‹S = (⨆ pos-img)›
    by (metis Sup-upper sup.absorb-iff2 sup.strict-order-iff)
qed

lemma hahn-decomp-finite:
  assumes ∀ E ∈ sets M. μ E < ∞
  shows ∃ M1 M2. hahn-space-decomp M1 M2 unfolding hahn-space-decomp-def
proof −
  define S where S = (⨆ pos-img)
  define A where A = (⋃ i. seq-meas-rep i)
  have pos-meas-set A unfolding A-def using assms seq-meas-rep-pos by simp
  have neg-meas-set ((space M) − A)
    using seq-meas-rep-compl assms unfolding A-def by simp
  show ∃ M1 M2. pos-meas-set M1 ∧ neg-meas-set M2 ∧ space M = M1 ∪ M2 ∧

    M1 ∩ M2 = {}
  proof (intro exI conjI)
    show pos-meas-set A using ‹pos-meas-set A› .
    show neg-meas-set (space M − A) using ‹neg-meas-set (space M − A)› .
    show space M = A ∪ (space M − A)
      by (metis Diff-partition ‹pos-meas-set A› inf.absorb-iff2 pos-meas-setD1
```

```
        sets.Int-space-eq1)
    show A ∩ (space M − A) = {} by auto
  qed
qed

theorem hahn-decomposition:
  shows ∃ M1 M2. hahn-space-decomp M1 M2
proof (cases ∀ E ∈ sets M. μ E < ∞)
  case True
  thus ?thesis using hahn-decomp-finite by simp
next
  case False
  define m where m = (λA . − μ A)
  have ∃ M1 M2. signed-measure-space.hahn-space-decomp M m M1 M2
  proof (rule signed-measure-space.hahn-decomp-finite)
    show signed-measure-space M m
      using signed-measure-minus sgn-meas ‹m = (λA . − μ A)›
      by (unfold-locales, simp)
    show ∀ E∈sets M. m E < ∞
    proof
      fix E
      assume E ∈ sets M
      show m E < ∞
      proof
        show m E ≠ ∞
        proof (rule ccontr)
          assume ¬ m E ≠ ∞
          have m E = ∞
            using ‹¬ m E ≠ ∞› by auto
          have signed-measure M m
            using ‹signed-measure-space M m› signed-measure-space-def by auto
          moreover have m E = − μ E using ‹m = (λA . − μ A)› by auto
          then have ∞ ∉ range m using ‹signed-measure M m›
            by (metis (no-types, lifting) False ereal-less-PInfty
                ereal-uminus-eq-reorder image-iff inf-range m-def rangeI)
          show False using ‹m E = ∞› ‹∞ ∉ range m›
            by (metis rangeI)
        qed
      qed
    qed
  qed
  hence ∃ M1 M2. (neg-meas-set M1) ∧ (pos-meas-set M2) ∧ (space M = M1 ∪
M2) ∧
    (M1 ∩ M2 = {})
    using pos-meas-set-opp neg-meas-set-opp unfolding m-def
    by (metis sgn-meas signed-measure-minus signed-measure-space-def
        signed-measure-space.hahn-space-decomp-def)
  thus ?thesis using hahn-space-decomp-def by (metis inf-commute sup-commute)
qed
```

5 The Jordan decomposition theorem

definition *jordan-decomp* **where**
jordan-decomp m1 m2 $\longleftrightarrow$ ((*measure-space* (*space M*) (*sets M*) *m1*) $\wedge$
(*measure-space* (*space M*) (*sets M*) *m2*) $\wedge$
($\forall$ *A*$\in$ *sets M*. *0* $\leq$ *m1 A*) $\wedge$
($\forall$ *A*$\in$ *sets M*. *0* $\leq$ *m2 A*) $\wedge$
($\forall$ *A* $\in$ *sets M*. μ *A* = (*m1 A*) $-$ (*m2 A*)) $\wedge$
($\forall$ *P N A*. *hahn-space-decomp P N* $\longrightarrow$
(*A* $\in$ *sets M* $\longrightarrow$ *A* $\subseteq$ *P* $\longrightarrow$ (*m2 A*) = *0*) $\wedge$
(*A* $\in$ *sets M* $\longrightarrow$ *A* $\subseteq$ *N* $\longrightarrow$ (*m1 A*) = *0*)) $\wedge$
(($\forall$ *A* $\in$ *sets M*. *m1 A* < ∞) $\vee$ ($\forall$ *A* $\in$ *sets M*. *m2 A* < ∞)))

lemma *jordan-decomp-pos-meas*:
assumes *jordan-decomp m1 m2*
and *hahn-space-decomp P N*
and *A* $\in$ *sets M*
shows *m1 A* = μ (*A* $\cap$ *P*)
proof $-$
have *A*$\cap$*P* $\in$ *sets M* **using** *assms* **unfolding** *hahn-space-decomp-def*
by (*simp add*: *pos-meas-setD1 sets.Int*)
have *A*$\cap$ *N* $\in$ *sets M* **using** *assms* **unfolding** *hahn-space-decomp-def*
by (*simp add*: *neg-meas-setD1 sets.Int*)
have (*A* $\cap$ *P*) $\cap$ (*A*$\cap$ *N*) = {} **using** *assms* **unfolding** *hahn-space-decomp-def*
by *auto*
have *A* = (*A* $\cap$ *P*) $\cup$ (*A*$\cap$ *N*) **using** *assms* **unfolding** *hahn-space-decomp-def*
by (*metis Int-Un-distrib sets.Int-space-eq2*)
hence *m1 A* = *m1* ((*A* $\cap$ *P*) $\cup$ (*A*$\cap$ *N*)) **by** *simp*
also have ... = *m1* (*A* $\cap$ *P*) + *m1* (*A* $\cap$ *N*)
using *assms pos-e2ennreal-additive*[*of M m1*] ‹*A*$\cap$*P* $\in$ *sets M*› ‹*A*$\cap$*N* $\in$ *sets M*›
‹*A* $\cap$ *P* $\cap$ (*A* $\cap$ *N*) = {}›
unfolding *jordan-decomp-def additive-def* **by** *simp*
also have ... = *m1* (*A* $\cap$ *P*) **using** *assms* **unfolding** *jordan-decomp-def*
by (*metis Int-lower2* ‹*A* $\cap$ *N* $\in$ *sets M*› *add.right-neutral*)
also have ... = *m1* (*A* $\cap$ *P*) $-$ *m2* (*A* $\cap$ *P*)
using *assms* **unfolding** *jordan-decomp-def*
by (*metis Int-subset-iff* ‹*A* $\cap$ *P* $\in$ *sets M*› *ereal-minus*(*7*)
local.pos-wtn-base pos-wtn-subset)
also have ... = μ (*A* $\cap$ *P*) **using** *assms* ‹*A* $\cap$ *P* $\in$ *sets M*›
unfolding *jordan-decomp-def* **by** *simp*
finally show *?thesis* **.**
qed

lemma *jordan-decomp-neg-meas*:
assumes *jordan-decomp m1 m2*
and *hahn-space-decomp P N*
and *A* $\in$ *sets M*
shows *m2 A* = $-\mu$ (*A* $\cap$ *N*)

```
proof –
  have A∩P ∈ sets M using assms unfolding hahn-space-decomp-def
    by (simp add: pos-meas-setD1 sets.Int)
  have A∩ N ∈ sets M using assms unfolding hahn-space-decomp-def
    by (simp add: neg-meas-setD1 sets.Int)
  have (A ∩ P) ∩ (A∩ N) = {}
    using assms unfolding hahn-space-decomp-def by auto
  have A = (A ∩ P) ∪ (A∩ N)
    using assms unfolding hahn-space-decomp-def
    by (metis Int-Un-distrib sets.Int-space-eq2)
  hence m2 A = m2 ((A ∩ P) ∪ (A∩ N)) by simp
  also have ... = m2 (A ∩ P) + m2 (A ∩ N)
    using pos-e2ennreal-additive[of M m2] assms
      ‹A∩P ∈ sets M› ‹A∩N ∈ sets M› ‹A ∩ P ∩ (A ∩ N) = {}›
    unfolding jordan-decomp-def additive-def by simp
  also have ... = m2 (A ∩ N) using assms unfolding jordan-decomp-def
    by (metis Int-lower2 ‹A ∩ P ∈ sets M› add.commute add.right-neutral)
  also have ... = m2 (A ∩ N) − m1 (A ∩ N)
    using assms unfolding jordan-decomp-def
    by (metis Int-lower2 ‹A ∩ N ∈ sets M› ereal-minus(7))
  also have ... = −μ (A ∩ N) using assms ‹A ∩ P ∈ sets M›
    unfolding jordan-decomp-def
    by (metis Diff-cancel Diff-eq-empty-iff Int-Un-eq(2) ‹A ∩ N ∈ sets M›
        ‹m2 (A ∩ N) = m2 (A ∩ N) − m1 (A ∩ N)› ereal-minus(8)
        ereal-uminus-eq-reorder sup.bounded-iff)
  finally show ?thesis .
qed

lemma pos-inter-neg-0:
  assumes hahn-space-decomp M1 M2
    and hahn-space-decomp P N
    and A ∈ sets M
    and A ⊆ N
  shows μ (A ∩ M1) = 0
proof –
  have μ (A ∩ M1) = μ (A ∩ ((M1 ∩ P) ∪ (M1 ∩ (sym-diff M1 P))))
    by (metis Diff-subset-conv Int-Un-distrib Un-upper1 inf.orderE)
  also have ... = μ ((A ∩ (M1 ∩ P)) ∪ (A ∩ (M1 ∩ (sym-diff M1 P))))
    by (simp add: Int-Un-distrib)
  also have ... = μ (A ∩ (M1 ∩ P)) + μ (A ∩ (M1 ∩ (sym-diff M1 P)))
  proof (rule signed-measure-add)
    show signed-measure M μ using sgn-meas .
    show A ∩ (M1 ∩ P) ∈ sets M
      by (meson assms(1) assms(2) assms(3) hahn-space-decomp-def sets.Int
          signed-measure-space.pos-meas-setD1 signed-measure-space-axioms)
    show A ∩ (M1 ∩ sym-diff M1 P) ∈ sets M
      by (meson Diff-subset assms(1) assms(2) assms(3) hahn-space-decomp-def
          pos-meas-setD1 pos-meas-set-union pos-meas-subset sets.Diff sets.Int)
    show A ∩ (M1 ∩ P) ∩ (A ∩ (M1 ∩ sym-diff M1 P)) = {} by auto
```

```
  qed
  also have ... = μ (A ∩ (M1 ∩ (sym-diff M1 P)))
  proof −
    have A ∩ (M1 ∩ P) = {} using assms hahn-space-decomp-def by auto
    thus ?thesis using signed-measure-empty[OF sgn-meas] by simp
  qed
  also have ... = 0
  proof (rule hahn-decomp-ess-unique[OF assms(1) assms(2)])
    show A ∩ (M1 ∩ sym-diff M1 P) ⊆ sym-diff M1 P ∪ sym-diff M2 N by auto
    show A ∩ (M1 ∩ sym-diff M1 P) ∈ sets M
    proof −
      have sym-diff M1 P ∈ sets M using assms
        by (meson hahn-space-decomp-def sets.Diff sets.Un
            signed-measure-space.pos-meas-setD1 signed-measure-space-axioms)
      hence M1 ∩ sym-diff M1 P ∈ sets M
        by (meson assms(1) hahn-space-decomp-def pos-meas-setD1 sets.Int)
      thus ?thesis by (simp add: assms sets.Int)
    qed
  qed
  finally show ?thesis .
qed

lemma neg-inter-pos-0:
  assumes hahn-space-decomp M1 M2
    and hahn-space-decomp P N
    and A ∈ sets M
    and A ⊆ P
  shows μ (A ∩ M2) = 0
proof −
  have μ (A ∩ M2) = μ (A ∩ ((M2 ∩ N) ∪ (M2 ∩ (sym-diff M2 N))))
    by (metis Diff-subset-conv Int-Un-distrib Un-upper1 inf.orderE)
  also have ... = μ ((A ∩ (M2 ∩ N)) ∪ (A ∩ (M2 ∩ (sym-diff M2 N))))
    by (simp add: Int-Un-distrib)
  also have ... = μ (A ∩ (M2 ∩ N)) + μ (A ∩ (M2 ∩ (sym-diff M2 N)))
  proof (rule signed-measure-add)
    show signed-measure M μ using sgn-meas .
    show A ∩ (M2 ∩ N) ∈ sets M
      by (meson assms(1) assms(2) assms(3) hahn-space-decomp-def sets.Int
          signed-measure-space.neg-meas-setD1 signed-measure-space-axioms)
    show A ∩ (M2 ∩ sym-diff M2 N) ∈ sets M
      by (meson Diff-subset assms(1) assms(2) assms(3) hahn-space-decomp-def
          neg-meas-setD1 neg-meas-set-union neg-meas-subset sets.Diff sets.Int)
    show A ∩ (M2 ∩ N) ∩ (A ∩ (M2 ∩ sym-diff M2 N)) = {} by auto
  qed
  also have ... = μ (A ∩ (M2 ∩ (sym-diff M2 N)))
  proof −
    have A ∩ (M2 ∩ N) = {} using assms hahn-space-decomp-def by auto
    thus ?thesis using signed-measure-empty[OF sgn-meas] by simp
  qed
```

```
  also have ... = 0
  proof (rule hahn-decomp-ess-unique[OF assms(1) assms(2)])
    show A ∩ (M2 ∩ sym-diff M2 N) ⊆ sym-diff M1 P ∪ sym-diff M2 N by auto
    show A ∩ (M2 ∩ sym-diff M2 N) ∈ sets M
    proof –
      have sym-diff M2 N ∈ sets M using assms
        by (meson hahn-space-decomp-def sets.Diff sets.Un
            signed-measure-space.neg-meas-setD1 signed-measure-space-axioms)
      hence M2 ∩ sym-diff M2 N ∈ sets M
        by (meson assms(1) hahn-space-decomp-def neg-meas-setD1 sets.Int)
      thus ?thesis by (simp add: assms sets.Int)
    qed
  qed
  finally show ?thesis .
qed

lemma jordan-decomposition :
  shows ∃ m1 m2. jordan-decomp m1 m2
proof –
  have ∃ M1 M2. hahn-space-decomp M1 M2 using hahn-decomposition
    unfolding hahn-space-decomp-def by simp
  from this obtain M1 M2 where hahn-space-decomp M1 M2 by auto
  note Mprops = this
  define m1 where m1 = (λA. μ (A ∩ M1))
  define m2 where m2 = (λA. −μ (A ∩ M2))
  show ?thesis unfolding jordan-decomp-def
  proof (intro exI allI impI conjI ballI)
    show measure-space (space M) (sets M) (λx. e2ennreal (m1 x))
      using pos-signed-to-meas-space Mprops m1-def
      unfolding hahn-space-decomp-def by auto
  next
    show measure-space (space M) (sets M) (λx. e2ennreal (m2 x))
      using neg-signed-to-meas-space Mprops m2-def
      unfolding hahn-space-decomp-def by auto
  next
    fix A
    assume A∈ sets M
    thus 0 ≤ m1 A unfolding m1-def using Mprops
      unfolding hahn-space-decomp-def
      by (meson inf-sup-ord(2) pos-meas-setD1 sets.Int
          signed-measure-space.pos-measure-meas signed-measure-space-axioms)
  next
    fix A
    assume A∈ sets M
    thus 0 ≤ m2 A unfolding m2-def using Mprops
      unfolding hahn-space-decomp-def
      by (metis ereal-0-le-uminus-iff inf-sup-ord(2) neg-meas-self
          neg-meas-setD1 neg-meas-subset sets.Int)
  next
```

```
  fix A
  assume A ∈ sets M
  have μ A = μ ((A ∩ M1) ∪ (A ∩ M2)) using Mprops
    unfolding hahn-space-decomp-def
    by (metis Int-Un-distrib ‹A ∈ sets M› sets.Int-space-eq2)
  also have ... = μ (A ∩ M1) + μ (A ∩ M2)
  proof (rule signed-measure-add)
    show signed-measure M μ using sgn-meas .
    show A ∩ M1 ∈ sets M using Mprops ‹A ∈ sets M›
      unfolding hahn-space-decomp-def
      by (simp add: pos-meas-setD1 sets.Int)
    show A ∩ M2 ∈ sets M using Mprops ‹A ∈ sets M›
      unfolding hahn-space-decomp-def
      by (simp add: neg-meas-setD1 sets.Int)
    show A ∩ M1 ∩ (A ∩ M2) = {} using Mprops
      unfolding hahn-space-decomp-def by auto
  qed
  also have ... = m1 A − m2 A using m1-def m2-def by simp
  finally show μ A = m1 A − m2 A .
next
  fix P N A
  assume hahn-space-decomp P N and A ∈ sets M and A ⊆ N
  note hn = this
  have μ (A ∩ M1) = 0
  proof (rule pos-inter-neg-0[OF - hn])
    show hahn-space-decomp M1 M2 using Mprops
      unfolding hahn-space-decomp-def by simp
  qed
  thus m1 A = 0 unfolding m1-def by simp
next
  fix P N A
  assume hahn-space-decomp P N and A ∈ sets M and A ⊆ P
  note hp = this
  have μ (A ∩ M2) = 0
  proof (rule neg-inter-pos-0[OF - hp])
    show hahn-space-decomp M1 M2 using Mprops
      unfolding hahn-space-decomp-def by simp
  qed
  thus m2 A = 0 unfolding m2-def by simp
next
  show (∀ E∈sets M. m1 E < ∞) ∨ (∀ E∈sets M. m2 E < ∞)
  proof (cases ∀ E ∈ sets M. m1 E < ∞)
    case True
    thus ?thesis by simp
  next
    case False
    have ∀ E ∈ sets M. m2 E < ∞
    proof
      fix E
```

```
      assume E ∈ sets M
      show m2 E < ∞
      proof −
        have (m2 E) = −μ (E ∩ M2) using m2-def by simp
        also have ... ≠ ∞ using False sgn-meas inf-range
          by (metis ereal-less-PInfty ereal-uminus-uminus m1-def rangeI)
        finally have m2 E ≠ ∞ .
        thus ?thesis by (simp add: top.not-eq-extremum)
      qed
    qed
    thus ?thesis by simp
  qed
 qed
qed

lemma jordan-decomposition-unique :
 assumes jordan-decomp m1 m2
   and jordan-decomp n1 n2
   and A ∈ sets M
 shows m1 A = n1 A m2 A = n2 A
proof −
 have ∃ M1 M2. hahn-space-decomp M1 M2 using hahn-decomposition by simp
 with jordan-decomp-neg-meas jordan-decomp-pos-meas
 show m1 A = n1 A m2 A = n2 A by (metis assms)+
qed

end

end
```

References

[1] E. DiBenedetto. *Real Analysis*. Birkhäuser Advanced Texts. Birkhäuser.