Riesz Representation Theorem

Michikazu Hirata

March 17, 2025

Abstract

We formalize the Riesz-Markov-Kakutani representation theorem following pp.37-47 of the book *Real and Complex Analysis* by Rudin [1]. This entry also includes formalization of regular measures, tightness of measures, and Urysohn's lemma on locally compact Hausdorff spaces. Roughly speaking, the theorem states that if φ is a positive linear functional from $C(X)$ (the space of continuous functions from X to complex numbers which have compact supports) to complex numbers, then there exists a unique measure μ such that for all $f \in C(X)$,

$$\varphi(f) = \int f \mathrm{d}\mu.$$

Contents

1 Urysohn's Lemma

theory *Urysohn-Locally-Compact-Hausdorff*
imports *Standard-Borel-Spaces.StandardBorel*
begin

We prove Urysohn's lemma for locally compact Hausdorff space (Lemma 2.12 [1])

1.1 Lemmas for Upper/Lower Semi-Continuous Functions

```
lemma
  assumes ⋀x. x ∈ topspace X ⟹ f x = g x
  shows upper-semicontinuous-map-cong:
    upper-semicontinuous-map X f ⟷ upper-semicontinuous-map X g (is ?g1)
    and lower-semicontinuous-map-cong:
    lower-semicontinuous-map X f ⟷ lower-semicontinuous-map X g (is ?g2)
proof −
  have [simp]:⋀a. {x∈topspace X. f x < a} = {x∈topspace X. g x < a}
              ⋀a. {x∈topspace X. f x > a} = {x∈topspace X. g x > a}
    using assms by auto
  show ?g1 ?g2
    by(auto simp: upper-semicontinuous-map-def lower-semicontinuous-map-def)
qed

lemma upper-lower-semicontinuous-map-iff-continuous-map:
 continuous-map X euclidean f ⟷ upper-semicontinuous-map X f ∧ lower-semicontinuous-map
X f
  using continuous-map-upper-lower-semicontinuous-lt
        lower-semicontinuous-map-def upper-semicontinuous-map-def
  by blast

lemma [simp]:
  shows upper-semicontinuous-map-const: upper-semicontinuous-map X (λx. c)
    and lower-semicontinuous-map-const: lower-semicontinuous-map X (λx. c)
  using continuous-map-const[of - euclidean c]
  unfolding upper-lower-semicontinuous-map-iff-continuous-map by auto


lemma upper-semicontinuous-map-c-add-iff:
  fixes c :: real
 shows upper-semicontinuous-map X (λx. c + f x) ⟷ upper-semicontinuous-map
X f
proof −
  have [simp]: c + f x < a ⟷ f x < a − c for x a
    by auto
  show ?thesis
    by(simp add: upper-semicontinuous-map-def) (metis add-diff-cancel-left′)
qed
```

```
corollary upper-semicontinuous-map-add-c-iff:
  fixes c :: real
  shows upper-semicontinuous-map X (λx. f x + c) ⟷ upper-semicontinuous-map
X f
  by(simp add: add.commute upper-semicontinuous-map-c-add-iff)


lemma upper-semicontinuous-map-posreal-cmult-iff:
  fixes c :: real
  assumes c > 0
  shows upper-semicontinuous-map X (λx. c ∗ f x) ⟷ upper-semicontinuous-map
X f
proof −
  have [simp]: c ∗ f x < a ⟷ f x < a / c for x a
    using assms by (simp add: less-divide-eq mult.commute)
  thus ?thesis
    by(simp add: upper-semicontinuous-map-def)
      (metis assms less-numeral-extra(3) nonzero-mult-div-cancel-left)
qed

lemma upper-semicontinuous-map-real-cmult:
  fixes c :: real
  assumes c ≥ 0 upper-semicontinuous-map X f
  shows upper-semicontinuous-map X (λx. c ∗ f x)
  by(cases c = 0)
   (use assms upper-semicontinuous-map-posreal-cmult-iff[simplified dual-order.strict-iff-order]
in auto)


lemma lower-semicontinuous-map-posreal-cmult-iff:
  fixes c :: real
  assumes c > 0
  shows lower-semicontinuous-map X (λx. c ∗ f x) ⟷ lower-semicontinuous-map
X f
proof −
  have [simp]: c ∗ f x > a ⟷ f x > a / c for x a
    by (simp add: assms divide-less-eq mult.commute)
  show ?thesis
    by(simp add: lower-semicontinuous-map-def)
      (metis assms less-numeral-extra(3) nonzero-mult-div-cancel-left)
qed

lemma lower-semicontinuous-map-real-cmult:
  fixes c :: real
  assumes c ≥ 0 lower-semicontinuous-map X f
  shows lower-semicontinuous-map X (λx. c ∗ f x)
  by(cases c = 0)
   (use assms lower-semicontinuous-map-posreal-cmult-iff[simplified dual-order.strict-iff-order]
```

in *auto*)

lemma *upper-semicontinuous-map-INF*:
fixes $f :: - \Rightarrow - \Rightarrow 'a :: \{linorder\text{-}topology, complete\text{-}linorder\}$
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *upper-semicontinuous-map* $X\ (f\ i)$
shows *upper-semicontinuous-map* $X\ (\lambda x.\ \bigsqcap i{\in}I.\ f\ i\ x)$
unfolding *upper-semicontinuous-map-def*
proof
fix a
have $\{x \in topspace\ X.\ (\bigsqcap i{\in}I.\ f\ i\ x) < a\} = (\bigcup i{\in}I.\ \{x{\in}topspace\ X.\ f\ i\ x < a\})$
by(*auto simp*: *Inf-less-iff*)
also have *openin* X ...
using *assms* **by**(*auto simp*: *upper-semicontinuous-map-def*)
finally show *openin* X $\{x \in topspace\ X.\ (\bigsqcap i{\in}I.\ f\ i\ x) < a\}$ **.**
qed

lemma *upper-semicontinuous-map-cInf*:
fixes $f :: - \Rightarrow - \Rightarrow 'a :: \{linorder\text{-}topology, conditionally\text{-}complete\text{-}linorder\}$
assumes $I \neq \{\}$ $\bigwedge x.\ x \in topspace\ X \Longrightarrow bdd\text{-}below\ ((\lambda i.\ f\ i\ x)\ `\ I)$
and $\bigwedge i.\ i \in I \Longrightarrow$ *upper-semicontinuous-map* $X\ (f\ i)$
shows *upper-semicontinuous-map* $X\ (\lambda x.\ \bigsqcap i{\in}I.\ f\ i\ x)$
unfolding *upper-semicontinuous-map-def*
proof
fix a
have [*simp*]:$\bigwedge x.\ x \in topspace\ X \Longrightarrow (\bigsqcap i{\in}I.\ f\ i\ x) < a \longleftrightarrow (\exists x{\in}(\lambda i.\ f\ i\ x)\ `\ I.$
$x < a)$
by(*intro cInf-less-iff*) (*use assms* **in** *auto*)
have $\{x \in topspace\ X.\ (\bigsqcap i{\in}I.\ f\ i\ x) < a\} = (\bigcup i{\in}I.\ \{x{\in}topspace\ X.\ f\ i\ x < a\})$
by *auto*
also have *openin* X ...
using *assms* **by**(*auto simp*: *upper-semicontinuous-map-def*)
finally show *openin* X $\{x \in topspace\ X.\ (\bigsqcap i{\in}I.\ f\ i\ x) < a\}$ **.**
qed

lemma *lower-semicontinuous-map-Sup*:
fixes $f :: - \Rightarrow - \Rightarrow 'a :: \{linorder\text{-}topology, complete\text{-}linorder\}$
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *lower-semicontinuous-map* $X\ (f\ i)$
shows *lower-semicontinuous-map* $X\ (\lambda x.\ \bigsqcup i{\in}I.\ f\ i\ x)$
unfolding *lower-semicontinuous-map-def*
proof
fix a
have $\{x \in topspace\ X.\ (\bigsqcup i{\in}I.\ f\ i\ x) > a\} = (\bigcup i{\in}I.\ \{x{\in}topspace\ X.\ f\ i\ x > a\})$
by(*auto simp*: *less-Sup-iff*)
also have *openin* X ...
using *assms* **by**(*auto simp*: *lower-semicontinuous-map-def*)
finally show *openin* X $\{x \in topspace\ X.\ (\bigsqcup i{\in}I.\ f\ i\ x) > a\}$ **.**
qed

lemma *indicator-closed-upper-semicontinuous-map*:

```
  assumes closedin X C
   shows upper-semicontinuous-map X (indicator C :: - ⇒ 'a :: {zero-less-one,
linorder-topology})
  unfolding upper-semicontinuous-map-def
proof safe
  fix a :: 'a
  consider a ≤ 0 | 0 < a a ≤ 1 | 1 < a
    by fastforce
  then show openin X {x ∈ topspace X. indicator C x < a}
  proof cases
    case 1
    then have [simp]:{x ∈ topspace X. indicator C x < a} = {}
     by(simp add: indicator-def) (meson order.strict-iff-not order.trans zero-less-one-class.zero-le-one)
    show ?thesis
      by simp
  next
    case 2
    then have [simp]:{x ∈ topspace X. indicator C x < a} = topspace X − C
      by(fastforce simp add: indicator-def)
    show ?thesis
      using assms by auto
  next
    case 3
    then have [simp]: {x ∈ topspace X. indicator C x < a} = topspace X
      by (simp add: indicator-def dual-order.strict-trans2)
    show ?thesis
      by simp
  qed
qed

lemma indicator-open-lower-semicontinuous-map:
  assumes openin X U
   shows lower-semicontinuous-map X (indicator U :: - ⇒ 'a :: {zero-less-one,
linorder-topology})
  unfolding lower-semicontinuous-map-def
proof safe
  fix a :: 'a
  consider a < 0 | 0 ≤ a a < 1 | 1 ≤ a
    by fastforce
  then show openin X {x ∈ topspace X. a < indicator U x}
  proof cases
    case 1
    then have [simp]: {x ∈ topspace X. a < indicator U x} = topspace X
      using order-less-trans by (fastforce simp add: indicator-def )
    show ?thesis
      by simp
  next
    case 2
    then have [simp]:{x ∈ topspace X. a < indicator U x} = U
```

using *openin-subset*[*OF assms*] **by**(*simp add*: *indicator-def*) *fastforce*
show *?thesis*
by(*simp add*: *assms*)
next
case *3*
then have [*simp*]:{$x \in$ *topspace* X. $a <$ *indicator* U x} = {}
by(*simp add*: *indicator-def*) (*meson dual-order.strict-trans leD zero-less-one*)
show *?thesis*
by *simp*
qed
qed

lemma *lower-semicontinuous-map-cSup*:
fixes f :: - ⇒ - ⇒ $'a$ :: {*linorder-topology*, *conditionally-complete-linorder*}
assumes $I \neq$ {} ⋀x. $x \in$ *topspace* $X \Longrightarrow$ *bdd-above* ((λi. f i x) ‘ I)
and ⋀i. $i \in I \Longrightarrow$ *lower-semicontinuous-map* X (f i)
shows *lower-semicontinuous-map* X (λx. ⨆i∈I. f i x)
unfolding *lower-semicontinuous-map-def*
proof
fix a
have [*simp*]:⋀x. $x \in$ *topspace* $X \Longrightarrow$ (⨆i∈I. f i x) > a ⟷ (∃x∈(λi. f i x) ‘ I. $x > a$)
by(*intro less-cSup-iff*) (*use assms* **in** *auto*)
have {$x \in$ *topspace* X. (⨆i∈I. f i x) > a} = (⋃i∈I. {x∈*topspace* X. f i $x > a$})
by(*auto simp*: *less-Sup-iff*)
also have *openin* X ...
using *assms* **by**(*auto simp*: *lower-semicontinuous-map-def*)
finally show *openin* X {$x \in$ *topspace* X. (⨆i∈I. f i x) > a} **.**
qed

lemma *openin-continuous-map-less*:
assumes *continuous-map* X (*euclidean* :: ($'a$ :: {*dense-linorder*, *order-topology*}) *topology*) f
and *continuous-map* X *euclidean* g
shows *openin* X {x∈*topspace* X. f $x < g$ x}
proof −
have {x∈*topspace* X. f $x < g$ x} = {x∈*topspace* X. ∃r. f $x < r \wedge r < g$ x}
using *dense order.strict-trans* **by** *blast*
also have ... = (⋃r. {x∈*topspace* X. f $x < r$} ∩ {x∈*topspace* X. $r < g$ x})
by *blast*
also have *openin* X ...
using *assms* **by**(*fastforce simp*: *continuous-map-upper-lower-semicontinuous-lt*)
finally show *?thesis* **.**
qed

corollary *closedin-continuous-map-eq*:
assumes *continuous-map* X (*euclidean* :: ($'a$ :: {*dense-linorder*, *order-topology*}) *topology*) f
and *continuous-map* X *euclidean* g

shows *closedin X* $\{x{\in}topspace\ X.\ f\ x = g\ x\}$
proof −
have $\{x{\in}topspace\ X.\ f\ x = g\ x\} = topspace\ X - (\{x{\in}topspace\ X.\ f\ x < g\ x\} \cup \{x{\in}topspace\ X.\ f\ x > g\ x\})$
by *auto*
also have *closedin X* ...
using *openin-continuous-map-less*[*OF assms*] *openin-continuous-map-less*[*OF assms*(*2*,*1*)]
by *blast*
finally show *?thesis* **.**
qed

1.2 Urysohn's Lemma

lemma *locally-compact-Hausdorff-compactin-openin-subset*:
assumes *locally-compact-space X Hausdorff-space X* $\vee$ *regular-space X*
and *compactin X T openin X V* $T \subseteq V$
shows $\exists U.$ *openin X U* $\wedge$ *compactin X* (*X closure-of U*) $\wedge T \subseteq U \wedge$ (*X closure-of U*) $\subseteq V$
proof −
have $\bigwedge x\ W.$ *openin X W* $\Longrightarrow x \in W$
$\Longrightarrow (\exists U\ V.$ *openin X U* $\wedge$ (*compactin X V* $\wedge$ *closedin X V*) $\wedge x \in U \wedge U \subseteq V \wedge V \subseteq W)$
using *assms*(*1*) **by**(*auto simp*: *locally-compact-space-neighbourhood-base-closedin*[*OF assms*(*2*)] *neighbourhood-base-of*)
from *this*[*OF assms*(*4*)] **have** $\forall x{\in}T.\ \exists U\ W.$ *openin X U* $\wedge$ (*compactin X W* $\wedge$ *closedin X W*) $\wedge x \in U \wedge U \subseteq W \wedge W \subseteq V$
using *assms*(*5*) **by** *blast*
then have $\exists Ux\ Wx.\ \forall x{\in}T.$ *openin X* $(Ux\ x) \wedge$ *compactin X* $(Wx\ x) \wedge$ *closedin X* $(Wx\ x) \wedge x \in Ux\ x \wedge Ux\ x \subseteq Wx\ x \wedge Wx\ x \subseteq V$
by *metis*
then obtain *Ux Wx* **where** *UW*: $\bigwedge x.\ x \in T \Longrightarrow$ *openin X* $(Ux\ x)$ $\bigwedge x.\ x \in T \Longrightarrow$ *compactin X* $(Wx\ x)$ $\bigwedge x.\ x \in T \Longrightarrow$ *closedin X* $(Wx\ x)$
$\bigwedge x.\ x \in T \Longrightarrow x \in Ux\ x$ $\bigwedge x.\ x \in T \Longrightarrow Ux\ x \subseteq Wx\ x$ $\bigwedge x.\ x \in T \Longrightarrow Wx\ x \subseteq V$
by *blast*
have $T \subseteq (\bigcup x{\in}T.\ Ux\ x)$
using *UW* **by** *blast*
hence $\exists \mathcal{F}.$ *finite* $\mathcal{F} \wedge \mathcal{F} \subseteq Ux$ ‘ $T \wedge T \subseteq \bigcup \mathcal{F}$
using *compactinD*[*OF assms*(*3*),*of Ux* ‘ *T*] *UW*(*1*) **by** *auto*
then obtain T' **where** T': *finite* T' $T' \subseteq T$ $T \subseteq (\bigcup x{\in}T'.\ Ux\ x)$
by (*metis finite-subset-image*)
have *1*:*X closure-of* $\bigcup$ $(Ux$ ‘ $T') = (\bigcup x{\in}T'.$ *X closure-of* $(Ux\ x))$
by (*simp add*: T'(*1*) *closure-of-Union*)
have *2*:$\bigwedge x.\ x \in T' \Longrightarrow$ *X closure-of* $(Ux\ x) \subseteq Wx\ x$
by (*meson* T'(*2*) *UW*(*3*) *UW*(*5*) *closure-of-minimal subsetD*)
hence $\bigwedge x.\ x \in T' \Longrightarrow$ *compactin X* (*X closure-of* $(Ux\ x)$)
by (*meson* T'(*2*) *UW*(*2*) *closed-compactin closedin-closure-of subsetD*)
then show *?thesis*

using *T′ 2 UW* **by**(*fastforce intro*!: *exI*[**where** *x*=⋃ *x*∈*T′*. *Ux x*] *compactin-Union simp*: *1*)
qed

lemma *Urysohn-locally-compact-Hausdorff-closed-compact-support*:
  fixes *a b* :: *real* **and** *X* :: *′a topology*
  assumes *locally-compact-space X Hausdorff-space X* ∨ *regular-space X*
    and *a* ≤ *b closedin X S compactin X T disjnt S T*
  obtains *f* **where** *continuous-map X* (*subtopology euclidean* {*a*..*b*}) *f f* ‘ *S* ⊆ {*a*} *f* ‘ *T* ⊆ {*b*} *disjnt* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *a*}) *S compactin X* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *a*})
proof −
  have ∃*f*. *continuous-map X* (*subtopology euclidean* {*0*..*1*::*real*}) *f* ∧ *f* ‘ *S* ⊆ {*0*} ∧ *f* ‘ *T* ⊆ {*1*} ∧ *disjnt* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}) *S* ∧ *compactin X* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*})
  proof −
    define *r* :: *nat* ⇒ *real* **where** *r* ≡ (λ*n*. *if n* = *0 then 0 else if n* = *1 then 1 else from-nat-into* ({*0*<..<*1*} ∩ ℚ) (*n* − *2*))
    have *r-01*: *r 0* = *0 r* (*Suc 0*) = *1*
      by(*simp-all add*: *r-def*)
    have *r-bij*: *bij-betw r UNIV* ({*0*..*1*} ∩ ℚ)
    proof −
      have *1*:*bij-betw* (*from-nat-into* ({*0*<..<*1*::*real*} ∩ ℚ)) *UNIV* ({*0*<..<*1*} ∩ ℚ)
      proof −
        have [*simp*]:*infinite* ({*0*<..<*1*::*real*} ∩ ℚ)
        proof −
          have {*0*<..<*1*::*real*} ∩ ℚ = *of-rat* ‘ {*0*<..<*1*::*rat*}
            by(*auto simp*: *Rats-def*)
          also have *infinite* ...
          proof
            assume *finite* (*real-of-rat* ‘ {*0*<..<*1*})
            moreover have *finite* (*real-of-rat* ‘ {*0*<..<*1*}) ⟷ *finite* {*0*<..<*1*::*rat*}
              by(*auto intro*!: *finite-image-iff inj-onI*)
            ultimately show *False*
              using *infinite-Ioo*[*of 0 1* :: *rat*] **by** *simp*
          qed
          finally show *?thesis* **.**
        qed
        show *?thesis*
          using *countable-rat* **by**(*auto intro*!: *from-nat-into-to-nat-on-product-metric-pair*)
      qed
      have *2*: *bij-betw r* ({*2*..}) ({*0*<..<*1*} ∩ ℚ)
      proof −
        have *3*:*bij-betw* (λ*n*. *n* − *2*) {*2*::*nat*..} *UNIV*
          by(*auto simp*: *bij-betw-def image-def intro*!: *inj-onI bexI*[**where** *x*=- + *2*])
        have *4*:*bij-betw* (λ*n*. *r* (*n* + *2*)) *UNIV* ({*0*<..<*1*} ∩ ℚ)
          using *1* **by**(*auto simp*: *r-def*)
        have *5*: *bij-betw* (λ*n*. *r* (*Suc* (*Suc* (*n* − *2*)))) {*2*..} ({*0*<..<*1*} ∩ ℚ)
          using *bij-betw-comp-iff*[*THEN iffD1*,*OF 3 4*] **by**(*auto simp*: *comp-def*)

show *?thesis*
by(*rule bij-betw-cong*[*THEN iffD1*,*OF - 5*]) (*simp add*: *Suc-diff-Suc numeral-2-eq-2*)
qed
have [*simp*]: *insert* (*Suc 0*) (*insert 0* {*2*..}) = *UNIV insert 1* (*insert 0* ({*0*<..<*1*::*real*} ∩ ℚ)) = {*0*..*1*} ∩ ℚ
by *auto*
show *?thesis*
using *notIn-Un-bij-betw*[*of 1*,*OF - - notIn-Un-bij-betw*[*of 0*,*OF - - 2*]] **by**(*auto simp*: *r-01*)
qed
have *r0-min*: ⋀*n*. *n* ≠ *0* ⟷ *r 0* < *r n*
using *r-bij r-01* **by** (*metis* (*full-types*) *IntE UNIV-I atLeastAtMost-iff bij-betw-iff-bijections linorder-not-le not-less-iff-gr-or-eq*)
have *r1-max*: ⋀*n*. *n* ≠ *1* ⟷ *r n* < *r 1*
using *r-bij r-01*(*2*) **by** (*metis* (*no-types*, *opaque-lifting*) *IntD2 One-nat-def UNIV-I atLeastAtMost-iff bij-betw-iff-bijections inf-commute linorder-less-linear linorder-not-le*)
let *?V* = *topspace X* − *S*
have *openinV*: *openin X ?V*
using *assms*(*4*) **by** *blast*
have *T-sub-V*: *T* ⊆ *?V*
by(*meson DiffI assms*(*5*,*6*) *compactin-subset-topspace disjnt-iff subset-eq*)
obtain *V0* **where** *V0*: *openin X V0 compactin X* (*X closure-of V0*) *T* ⊆ *V0 X closure-of V0* ⊆ *?V*
using *locally-compact-Hausdorff-compactin-openin-subset*[*OF assms*(*1*,*2*) *assms*(*5*) *openinV T-sub-V*] **by** *metis*
obtain *V1* **where** *V1*: *openin X V1 compactin X* (*X closure-of V1*) *T* ⊆ *V1 X closure-of V1* ⊆ *V0*
using *locally-compact-Hausdorff-compactin-openin-subset*[*OF assms*(*1*,*2*) *assms*(*5*) *V0*(*1*,*3*)] **by** *metis*

arg max

have ∃ *i*. *i* < *n* ∧ *r i* < *r n* ∧ (∀ *m*. *m* < *n* ∧ *r m* < *r n* ⟶ *r m* ≤ *r i*) **if** *n*: *n* ≥ *2* **for** *n*
proof −
have *1*:{*m*. *m* < *n* ∧ *r m* < *r n*} ≠ {}
proof −
have *n* ≠ *0*
using *n* **by** *fastforce*
hence *r n* ≠ *r 0*
by (*metis UNIV-I r-bij bij-betw-iff-bijections*)
hence *r n* > *r 0*
by (*metis IntE UNIV-I atLeastAtMost-iff bij-betw-iff-bijections order-less-le r-01*(*1*) *r-bij*)
hence *0* ∈ {*m*. *m* < *n* ∧ *r n* > *r m*}
using *n* **by** *auto*
thus *?thesis*
by *auto*
qed

have 2:*finite* $\{m.\ m < n \wedge r\ n > r\ m\}$
by *auto*
define *ri* **where** $ri \equiv Max\ (r\ `\ \{m.\ m < n \wedge r\ n > r\ m\})$
have *ri-1*: $ri \in r\ `\ \{m.\ m < n \wedge r\ n > r\ m\}$
unfolding *ri-def* **using** *1 2* **by** *auto*
have *ri-2*: $\bigwedge m.\ m < n \Longrightarrow r\ n > r\ m \Longrightarrow r\ m \leq ri$
unfolding *ri-def* **by**(*subst Max-ge-iff*) (*use 1 2* **in** *auto*)
obtain *i* **where** i:$ri = r\ i\ i < n\ r\ n > r\ i$
using *ri-1* **by** *auto*
thus *?thesis*
using *ri-2* **by**(*auto intro*!: *exI*[**where** x=i])
qed
then obtain *i* **where** i: $\bigwedge n.\ n \geq 2 \Longrightarrow i\ n < n\ \bigwedge n.\ n \geq 2 \Longrightarrow r\ (i\ n) < r\ n$
$\bigwedge n\ m.\ n \geq 2 \Longrightarrow m < n \Longrightarrow r\ m < r\ n \Longrightarrow r\ m \leq r\ (i\ n)$
by *metis*

arg min

have $\exists j.\ j < n \wedge r\ n < r\ j \wedge (\forall m.\ m < n \wedge r\ n < r\ m \longrightarrow r\ j \leq r\ m)$ **if** n: $n \geq 2$ **for** n
proof $-$
have 1:$\{m.\ m < n \wedge r\ n < r\ m\} \neq \{\}$
proof $-$
have $n \neq 1$
using n **by** *fastforce*
hence $r\ n \neq r\ 1$
by (*metis UNIV-I r-bij bij-betw-iff-bijections*)
hence $r\ n < r\ 1$
by (*metis IntE One-nat-def UNIV-I atLeastAtMost-iff bij-betw-iff-bijections order-less-le r-01*(*2*) *r-bij*)
hence $1 \in \{m.\ m < n \wedge r\ n < r\ m\}$
using n **by** *auto*
thus *?thesis*
by *auto*
qed
have 2:*finite* $\{m.\ m < n \wedge r\ n < r\ m\}$
by *auto*
define *rj* **where** $rj \equiv Min\ (r\ `\ \{m.\ m < n \wedge r\ n < r\ m\})$
have *rj-1*: $rj \in r\ `\ \{m.\ m < n \wedge r\ n < r\ m\}$
unfolding *rj-def* **using** *1 2* **by** *auto*
have *rj-2*: $\bigwedge m.\ m < n \Longrightarrow r\ n < r\ m \Longrightarrow rj \leq r\ m$
unfolding *rj-def* **by**(*subst Min-le-iff*) (*use 1 2* **in** *auto*)
obtain *j* **where** j:$rj = r\ j\ j < n\ r\ n < r\ j$
using *rj-1* **by** *auto*
thus *?thesis*
using *rj-2* **by**(*auto intro*!: *exI*[**where** x=j])
qed
then obtain *j* **where** j: $\bigwedge n.\ n \geq 2 \Longrightarrow j\ n < n\ \bigwedge n.\ n \geq 2 \Longrightarrow r\ (j\ n) > r\ n\ \bigwedge n\ m.\ n \geq 2 \Longrightarrow m < n \Longrightarrow r\ m > r\ n \Longrightarrow r\ m \geq r\ (j\ n)$
by *metis*

```
  have i2: i 2 = 0
    by (metis i(1,2) One-nat-def dual-order.refl less-2-cases not-less-iff-gr-or-eq
r1-max)
  have j2: j 2 = 1
   by (metis j(1,2) One-nat-def dual-order.refl i(2) i2 less-2-cases not-less-iff-gr-or-eq)
  have ∃ Vn. ∀ n. Vn n = (if n = 0 then V0 else if n = 1 then V1
     else (SOME V. openin X V ∧ compactin X (X closure-of V) ∧ X closure-of
Vn (j n) ⊆ V ∧ X closure-of V ⊆ Vn (i n)))
    (is ∃ Vn. ∀ n. Vn n = ?if n Vn)
  proof(rule dependent-wellorder-choice)
    fix r n and Vn Vn' :: nat ⇒ 'a set
    assume h:⋀y::nat. y < n ⟹ Vn y = Vn' y
    consider n ≥ 2 | n = 0 | n = 1
      by fastforce
    then show r = ?if n Vn ⟷ r = ?if n Vn'
      by cases (use i j h in auto)
  qed auto
  then obtain Vn where Vn-def: ⋀n. Vn n = (if n = 0 then V0 else if n = 1
then V1
     else (SOME V. openin X V ∧ compactin X (X closure-of V) ∧ X closure-of
Vn (j n) ⊆ V ∧ X closure-of V ⊆ Vn (i n)))
    by blast
  have Vn-0: Vn 0 = V0 and Vn-1: Vn 1 = V1
    by(auto simp: Vn-def)
  have Vns: (n ≥ 2 ⟶ openin X (Vn n) ∧ compactin X (X closure-of Vn n) ∧
            X closure-of Vn (j n) ⊆ Vn n ∧ X closure-of Vn n ⊆ Vn (i n)) ∧
            (∀ k≤n. ∀ l≤n. r k < r l ⟶ X closure-of Vn l ⊆ Vn k) (is ?P1 n ∧
?P2 n) for n
  proof(rule nat-less-induct[of - n])
    fix n
    assume h:∀ m<n. ?P1 m ∧ ?P2 m
    show ?P1 n ∧ ?P2 n
    proof
      show P1:?P1 n
      proof
        assume n: 2 ≤ n
        then consider n = 2 | n > 2
          by fastforce
        then show openin X (Vn n) ∧ compactin X (X closure-of Vn n) ∧
                  X closure-of Vn (j n) ⊆ Vn n ∧ X closure-of Vn n ⊆ Vn (i n)
        proof cases
          case 1
         have 2:Vn 2 = (SOME V. openin X V ∧ compactin X (X closure-of V)
∧
                  X closure-of Vn 1 ⊆ V ∧ X closure-of V ⊆ Vn 0)
            by(simp add: Vn-def i2 j2 1)
          show ?thesis
            unfolding 1 i2 j2 Vn-0 Vn-1 2
            by(rule someI-ex)
```

```
          (auto intro!: V0 V1 locally-compact-Hausdorff-compactin-openin-subset[OF
assms(1,2)])
        next
          case 2
        then have 1:Vn n = (SOME V. openin X V ∧ compactin X (X closure-of
V) ∧ X closure-of Vn (j n) ⊆ V ∧ X closure-of V ⊆ Vn (i n))
            by(auto simp: Vn-def)
          show ?thesis
            unfolding 1
          proof(rule someI-ex)
            have ij:j n < n i n < n r (i n) < r (j n)
              using j[of n] i[of n] order.strict-trans 2 by auto
            hence max (j n) (i n) < n
              by auto
            from h[rule-format,OF this] ij(3) have ijsub:X closure-of Vn (j n) ⊆
Vn (i n)
              by auto
            have jc:compactin X (X closure-of Vn (j n))
            proof –
              consider j n ≥ 2 | j n = 0 | j n = 1
                by fastforce
              then show ?thesis
              proof cases
                case 1
                then show ?thesis
                  using ij(1) h by auto
              qed(auto simp: Vn-0 Vn-1[simplified] V0 V1)
            qed
            have io:openin X (Vn (i n))
            proof –
              consider i n ≥ 2 | i n = 0 | i n = 1
                by fastforce
              then show ?thesis
              proof cases
                case 1
                then show ?thesis
                  using ij(2) h by auto
              qed(auto simp: Vn-0 Vn-1[simplified] V0 V1)
            qed
            show ∃x. openin X x ∧ compactin X (X closure-of x) ∧ X closure-of
Vn (j n) ⊆ x ∧ X closure-of x ⊆ Vn (i n)
                  by(rule locally-compact-Hausdorff-compactin-openin-subset[OF
assms(1,2) jc io ijsub])
          qed
        qed
      qed
      show ?P2 n
      proof(intro allI impI)
        fix k l
```

```
assume kl: k ≤ n l ≤ n r k < r l
then consider n = 1 | n ≥ 2
  using r-bij order-neq-le-trans by fastforce
then show X closure-of Vn l ⊆ Vn k
proof cases
  case 1
  then have [simp]: k = 0 l = 1
    using r-01 kl le-Suc-eq by fastforce+
  show ?thesis
    using Vn-0 Vn-1 V0 V1 by simp
next
  case n:2
  consider k < n l < n | k = n l < n | k < n l = n
    using kl order-less-le by auto
  then show ?thesis
  proof cases
    case 1
    with kl(3) h show ?thesis
      by (meson nle-le)
  next
    case k:2
    then have k1:X closure-of Vn (j k) ⊆ Vn k
      using P1 n by simp
    consider r (j k) = r l | r (j k) < r l
      using j(3)[OF - - kl(3)] k n by fastforce
    then show ?thesis
    proof cases
      case 1
      then have j k = l
        using r-bij by(auto simp: bij-betw-def injD)
      with k1 show ?thesis by simp
    next
      case 2
      then have  X closure-of Vn l ⊆ Vn (j k)
        using k h by (meson j(1) n nat-le-linear)
      thus ?thesis
        using k1 closure-of-mono by fastforce
    qed
  next
    case l:3
    consider r k = r (i l) | r k < r (i l)
      using i(3)[OF - - kl(3)] l n by fastforce
    then show ?thesis
    proof cases
      case 1
      then have k = i l
        using r-bij by(auto simp: bij-betw-def injD)
      thus ?thesis
        using P1 l(2) n by blast
```

```
              next
                case 2
                then have  X closure-of Vn (i l) ⊆ Vn k
                  by (metis h i(1) l(1) l(2) n nle-le)
                thus ?thesis
               by (metis P1 closure-of-closure-of closure-of-mono l(2) n subset-trans)
              qed
            qed
          qed
        qed
      qed
    qed

    define Vr where Vr ≡ (λx. let n = THE n. x = r n in Vn n)
    have Vr-Vn: Vr (r n) = Vn n for n
    proof –
      have 1:⋀n m. r n = r m ⟷ n = m
        using r-bij by(auto simp: bij-betw-def injD)
      have [simp]:(THE m. r n = r m) = n
        by(auto simp: 1)
      show ?thesis
        by(simp add: Vr-def)
    qed
    have Vr0: Vr 0 = V0
      using Vr-Vn[of 0] by(auto simp: Vn-0 r-01)
    have Vr1: Vr 1 = V1
      using Vr-Vn[of 1] Vn-1 by(auto simp: r-01)
    have openin-Vr: openin X (Vr s) if s:s ∈ {0..1} ∩ ℚ for s
    proof –
      consider 0 < s s < 1 | s = 0 | s = 1
        using s by fastforce
      then show ?thesis
      proof cases
        case 1
        then obtain n where n ≥ 2 s = r n
               by (metis r0-min r1-max s One-nat-def Suc-1 bij-betw-iff-bijections
bot-nat-0.extremum-unique le-SucE not-less-eq-eq r-bij r-def)
        thus ?thesis
          using Vns Vr-Vn by fastforce
      qed(auto simp: Vr0 Vr1 V0 V1)
    qed
     have compactin-clVr: compactin X (X closure-of (Vr s)) if s:s ∈ {0..1} ∩ ℚ
for s
    proof –
      consider 0 < s s < 1 | s = 0 | s = 1
        using s by fastforce
      then show ?thesis
      proof cases
        case 1
```

then obtain *n* **where** $n \geq 2$ $s = r\ n$
by (*metis r0-min r1-max s One-nat-def Suc-1 bij-betw-iff-bijections bot-nat-0.extremum-unique le-SucE not-less-eq-eq r-bij r-def*)
thus *?thesis*
using *Vns Vr-Vn* **by** *fastforce*
qed(*auto simp*: *Vr0 Vr1 V0 V1*)
qed
have *Vr-antimono*:*X closure-of Vr s* ⊆ *Vr k* **if** *h*:*s* ∈ {*0..1*} ∩ ℚ *k* ∈ {*0..1*} ∩ ℚ *k* < *s* **for** *k s*
proof −
obtain *ns nk* **where** *n*: *s* = *r ns k* = *r nk*
by (*metis h*(*1*,*2*) *bij-betw-iff-bijections r-bij*)
show *?thesis*
using *Vr-Vn Vns*[*of max ns nk*] *h* **by**(*auto simp*: *n*)
qed
define *f* **where** *f* ≡ (λ*x*. ⨆*s*∈{*0..1*} ∩ ℚ. *s* ∗ *indicat-real* (*Vr s*) *x*)
define *g* **where** *g* ≡ (λ*x*. ⨅*s*∈{*0..1*} ∩ ℚ. (*1* − *s*) ∗ *indicat-real* (*X closure-of Vr s*) *x* + *s*)
note [*intro*!] = *bdd-belowI*[**where** *m=0*] *bdd-aboveI*[**where** *M=1*]
note [*simp*] = *mult-le-one*
have *ne*[*simp*]: {*0..1*::*real*} ∩ ℚ ≠ {}
using *Rats-0 atLeastAtMost-iff zero-less-one-class.zero-le-one* **by** *blast*

have *f-lower*:*lower-semicontinuous-map X f*
unfolding *f-def*
by(*auto intro*!: *lower-semicontinuous-map-cSup lower-semicontinuous-map-real-cmult indicator-open-lower-semicontinuous-map openin-Vr*)
have *g-upper*:*upper-semicontinuous-map X g*
unfolding *g-def*
by(*auto intro*!: *upper-semicontinuous-map-cInf upper-semicontinuous-map-real-cmult indicator-closed-upper-semicontinuous-map*
simp: *upper-semicontinuous-map-add-c-iff*)

have *f-01*: ⋀*x*. *0* ≤ *f x* ⋀*x*. *f x* ≤ *1*
proof −
show ⋀*x*. *0* ≤ *f x*
unfolding *f-def* **by**(*subst le-cSup-iff*) (*auto intro*!: *bexI*[**where** *x=0*])
show ⋀*x*. *f x* ≤ *1*
unfolding *f-def* **by**(*subst cSup-le-iff*) (*auto intro*!: *bexI*[**where** *x=0*])
qed
have *g-01*: ⋀*x*. *0* ≤ *g x* ⋀*x*. *g x* ≤ *1*
proof −
show ⋀*x*. *0* ≤ *g x*
unfolding *g-def* **by**(*subst le-cInf-iff*) *auto*
have ⋀*x*. ∀ *y*>*1*. ∃ *a*∈(λ*s*. (*1* − *s*) ∗ *indicat-real* (*X closure-of Vr s*) *x* + *s*) ‘ ({*0..1*} ∩ ℚ). *a* < *y*
by (*metis* (*no-types*, *lifting*) *Int-iff Rats-1 add-0 atLeastAtMost-iff cancel-comm-monoid-add-class.diff-cancel image-eqI less-eq-real-def mult-cancel-left1 zero-less-one-class.zero-le-one*
thus ⋀*x*. *g x* ≤ *1*

unfolding *g-def* **by**(*subst cInf-le-iff*) *auto*
qed

have *disj*: *disjnt* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}) *S*
and *f-csupport*:*compactin X* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*})
proof −
have *1*:{*x*∈*topspace X*. *f x* ≠ *0*} ⊆ *X closure-of V0*
proof −
have {*x*∈*topspace X*. *f x* ≠ *0*} = {*x*∈*topspace X*. *f x* > *0*}
using *f-01* **by** (*simp add*: *order-less-le*)
also have ... ⊆ *X closure-of V0*
proof *safe*
fix *x*
assume *h*:*x* ∈ *topspace X 0* < *f x*
then have ∃ *x*∈(λ*s*. *s* ∗ *indicat-real* (*Vr s*) *x*) ‘ ({*0..1*} ∩ ℚ). *0* < *x*
by(*intro less-cSup-iff*[*THEN iffD1*]) (*auto simp*: *f-def*)
then obtain *s* **where** *s*: *s* ∈ {*0..1*} ∩ ℚ *s* ∗ *indicat-real* (*Vr s*) *x* > *0*
by *fastforce*
hence *1*:*s* > *0 0* < *indicat-real* (*Vr s*) *x*
by (*auto simp add*: *zero-less-mult-iff*)
hence *2*:*x* ∈ *Vr s*
by(*auto simp*: *indicator-def*)
have *Vr s* ⊆ *X closure-of Vr s*
by (*meson closure-of-subset openin-Vr openin-subset s*(*1*))
also have ... ⊆ *X closure-of V0*
using *Vr-antimono*[*OF - - 1*(*1*)] *s*(*1*) **by** (*metis IntI Rats-0 Vr0 atLeastAtMost-iff calculation closure-of-mono order.refl order-trans zero-less-one-class.zero-le-one*)
finally show *x* ∈ *X closure-of V0*
using *2* **by** *auto*
qed
finally show *?thesis* **.**
qed
thus *compactin X* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*})
by (*meson V0*(*2*) *closed-compactin closedin-closure-of closure-of-minimal*)
show *disjnt* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}) *S*
using *1 V0*(*4*) *closure-of-mono* **by**(*fastforce simp*: *disjnt-def*)
qed
have *f-1*: *f x* = *1* **if** *x*: *x* ∈ *T* **for** *x*
proof −
have *xv*:*x* ∈ *V1*
using *V1*(*3*) *x* **by** *blast*
have *1* ≤ *f x*
unfolding *f-def* **by**(*subst le-cSup-iff*) (*auto intro*!: *bexI*[**where** *x*=*1*] *simp*: *Vr1 xv*)
with *f-01* **show** *?thesis*
using *nle-le* **by** *blast*
qed
have *f-0*: *f x* = *0* **if** *x*: *x* ∈ *S* **for** *x*
proof −

```
have x ∉ Vr s if s: s ∈ {0..1} ∩ ℚfor s
proof −
  have x ∉ Vr 0
      using x V0 closure-of-subset[OF openin-subset[of X V0]] by(auto simp:
Vr0)
  moreover have Vr s ⊆ Vr 0
        using Vr-antimono[of s 0] s closure-of-subset[OF openin-subset[OF
openin-Vr[OF s]]]
    by(cases s = 0) auto
  ultimately show ?thesis
    by blast
qed
hence f x ≤ 0
  unfolding f-def by(subst cSup-le-iff) auto
with f-01 show ?thesis
  using nle-le by blast
qed
have fg:f x = g x if x: x ∈ topspace X for x
proof −
  have ¬ f x < g x
  proof
    assume f x < g x
    then obtain r s where rs: r ∈ ℚ s ∈ ℚ f x < r r < s s < g x
      by (meson Rats-dense-in-real)
    hence r:r ∈ {0..1} ∩ ℚ
    using f-01 g-01 by (metis IntI atLeastAtMost-iff inf.orderE inf.strict-coboundedI2
linorder-not-less nle-le)
    hence s:s ∈ {0..1} ∩ ℚ
    using g-01 rs by (metis IntI atLeastAtMost-iff f-01(1) inf.strict-coboundedI2
inf.strict-order-iff less-eq-real-def)
    have x1:x ∉ Vr r
    proof −
      have r ∗ indicat-real (Vr r) x < r
        using r by(auto intro!: cSUP-lessD[OF - rs(3)[simplified f-def]])
      thus ?thesis
        using r by auto
    qed
    have x2:x ∈ X closure-of Vr s
    proof −
      have 1:s < (1 − s) ∗ indicat-real (X closure-of Vr s) x + s
        using s by(intro less-cINF-D[OF - rs(5)[simplified g-def]]) auto
      show ?thesis
        by(rule ccontr) (use s 1 in auto)
    qed
    show False
      using x1 x2 Vr-antimono[OF s r rs(4)] by blast
  qed
  moreover have f x ≤ g x
  proof −
```

have *l* ∗ *indicat-real* (*Vr l*) *x* ≤ (*1* − *s*) ∗ *indicat-real* (*X closure-of Vr s*) *x* + *s*
if *ls*: *l* ∈ {*0..1*} ∩ ℚ *s* ∈ {*0..1*} ∩ ℚ **for** *l s*
proof(*rule ccontr*)
assume *h*:¬ *l* ∗ *indicat-real* (*Vr l*) *x* ≤ (*1* − *s*) ∗ *indicat-real* (*X closure-of Vr s*) *x* + *s*
then have *l* ∗ *indicat-real* (*Vr l*) *x* > (*1* − *s*) ∗ *indicat-real* (*X closure-of Vr s*) *x* + *s*
by *auto*
hence *l* > *s* ∧ *x* ∈ *Vr l* ∧ *x* ∉ *Vr s*
using *ls* **by** (*metis* (*no-types*, *opaque-lifting*) *h Int-iff add.commute add.right-neutral atLeastAtMost-iff closure-of-subset diff-add-cancel in-mono indicator-simps*(*1*) *indicator-simps*(*2*) *mult.commute mult-1 mult-zero-left openin-Vr openin-subset zero-less-one-class.zero-le-one*)
moreover have *Vr l* ⊆ *Vr s*
using *Vr-antimono*[*OF ls*] **by** (*meson calculation closure-of-subset ls*(*1*) *openin-Vr openin-subset order-trans*)
ultimately show *False*
by *blast*
qed
thus *f x* ≤ *g x*
unfolding *f-def g-def* **by**(*auto intro*!: *cSup-le-iff*[*THEN iffD2*] *le-cInf-iff*[*THEN iffD2*])
qed
ultimately show *?thesis*
by *simp*
qed
show *?thesis*
proof(*safe intro*!: *exI*[**where** *x*=*f*])
have *continuous-map X euclideanreal f*
by (*simp add*: *fg f-lower g-upper upper-lower-semicontinuous-map-iff-continuous-map upper-semicontinuous-map-cong*)
thus *continuous-map X* (*top-of-set* {*0..1*}) *f*
using *f-01* **by**(*auto simp*: *continuous-map-in-subtopology*)
qed(*use f-0 f-1 f-csupport disj* **in** *auto*)
qed
then obtain *f* **where** *f*: *continuous-map X* (*top-of-set* {*0..1*}) *f f* ‘ *S* ⊆ {*0*::*real*} *f* ‘ *T* ⊆ {*1*}
disjnt (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}) *S compactin X* (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*})
by *blast*
define *g* **where** *g* ≡ (λ*x*. (*b* − *a*) ∗ *f x* + *a*)
have *continuous-map X* (*top-of-set* {*a..b*}) *g*
proof −
have [*simp*]:*0* ≤ *y* ∧ *y* ≤ *1* ⟹ (*b* − *a*) ∗ *y* + *a* ≤ *b* **for** *y*
using *assms*(*3*) **by** (*meson diff-ge-0-iff-ge le-diff-eq mult-left-le*)
show *?thesis*
using *f*(*1*) *assms*(*3*) **by**(*auto simp*: *image-subset-iff continuous-map-in-subtopology g-def*

intro!: *continuous-map-add continuous-map-real-mult-left*)
qed
moreover have g ‘ $S \subseteq \{a\}$ g ‘ $T \subseteq \{b\}$
using $f(2,3)$ **by**(*auto simp*: *g-def*)
moreover have *disjnt* (X *closure-of* $\{x{\in}$*topspace* $X.\ g\ x \neq a\}$) S
compactin X (X *closure-of* $\{x \in$ *topspace* $X.\ g\ x \neq a\}$)
proof −
consider $a = b \mid a < b$
using *assms* **by** *fastforce*
then have *disjnt* (X *closure-of* $\{x{\in}$*topspace* $X.\ g\ x \neq a\}$) $S \wedge$ *compactin* X (X *closure-of* $\{x \in$ *topspace* $X.\ g\ x \neq a\}$)
proof *cases*
case *1*
then have [*simp*]:$\{x \in$ *topspace* $X.\ g\ x \neq a\} = \{\}$
by(*auto simp*: *g-def*)
thus *?thesis*
by *simp-all*
next
case *2*
then have $\{x \in$ *topspace* $X.\ g\ x \neq a\} = \{x \in$ *topspace* $X.\ f\ x \neq 0\}$
by(*auto simp*: *g-def*)
thus *?thesis*
by(*simp add*: *f*)
qed
thus *disjnt* (X *closure-of* $\{x{\in}$*topspace* $X.\ g\ x \neq a\}$) S *compactin* X (X *closure-of* $\{x \in$ *topspace* $X.\ g\ x \neq a\}$)
by *simp-all*
qed
ultimately show *?thesis*
using *that* **by** *auto*
qed

end

2 Regular Measures

theory *Regular-Measure*
imports *HOL−Probability.Probability*
Standard-Borel-Spaces.StandardBorel
Urysohn-Locally-Compact-Hausdorff
begin

context *Metric-space*
begin

lemma *nbh-add*: $(\bigcup b{\in}(\bigcup a{\in}A.\ mball\ a\ e).\ mball\ b\ f) \subseteq (\bigcup a{\in}A.\ mball\ a\ (e + f))$
proof *clarify*
fix $a\ x\ b$
assume $h{:}a \in A\ \ b \in mball\ a\ e\ \ x \in mball\ b\ f$

```
  show x ∈ (⋃a∈A. mball a (e + f))
  proof(rule UN-I[OF h(1)])
    show x ∈ mball a (e + f)
      using h triangle by fastforce
  qed
qed

lemma nbh-subset:
  assumes A: A ⊆ M and e: e > 0
  shows A ⊆ (⋃a∈A. mball a e)
  using A e by auto

lemma nbh-decseq:
  assumes decseq an
  shows decseq (λn. ⋃a∈A. mball a (an n))
proof(safe intro!: decseq-SucI)
  fix n a b
  assume a ∈ A b ∈ mball a (an (Suc n))
  with decseq-SucD[OF assms] show b ∈ (⋃c∈A. mball c (an n))
    by(auto intro!: bexI[where x=a] simp: frac-le order-less-le-trans)
qed

lemma nbh-Inter-closure-of:
  assumes A: A ≠ {} A ⊆ M
      and an:⋀n. an n > 0 decseq an an ⟶ 0
    shows (⋂n. ⋃a∈A. mball a (an n)) = mtopology closure-of A
proof safe
  fix x
  assume x:x ∈ (⋂n. ⋃a∈A. mball a (an n))
  show x ∈ mtopology closure-of A
    unfolding metric-closure-of
  proof safe
    fix r :: real
    assume 0 < r
    from LIMSEQ-D[OF an(3) this] an(1) obtain N where N: ⋀n. n ≥ N ⟹
an n < r
      by fastforce
    show ∃y∈A. y ∈ mball x r
    proof(rule ccontr)
      assume ¬ (∃y∈A. y ∈ mball x r)
      then have 1:∀y∈A. y ∉ mball x r
        by auto
      obtain a where a:a ∈ A x ∈ mball a (an N)
        using x by auto
      with N[of N] have a ∈ mball x (an N) mball x (an N) ⊆ mball x r
        by (auto simp: commute)
      with a(1) 1 show False by auto
    qed
  qed(use x in auto)
```

next
  fix *x n*
  assume $x \in$ *mtopology closure-of A*
  then have $x \in M$ $\forall\, r{>}0.\ \exists\, y{\in}A.\ y \in$ *mball x r*
    by(*auto simp*: *metric-closure-of*)
  with *an*(*1*)[*of n*] **obtain** *y* **where** *y*:$y \in A$ $y \in$ *mball x* (*an n*)
    by *auto*
  thus $x \in (\bigcup a{\in}A.$ *mball a* (*an n*))
    by(*auto intro*!: *bexI*[**where** *x*=*y*] *simp*: *commute*)
qed

end

lemma(**in** *finite-measure*)
  assumes *range A* $\subseteq$ *sets M disjoint-family A*
  shows *suminf-measure*:($\sum i.$ *measure M* (*A i*)) = *measure M* ($\bigcup i.$ *A i*)
    and *summable-measure*: *summable* ($\lambda i.$ *measure M* (*A i*))
  using *finite-measure-UNION*[*OF assms*] **by**(*auto dest*: *sums-unique simp*: *summable-def*)

We refer to the lecture note [2].

Inner regular and outer regular with abstract topologies.

definition *inner-regular* :: *'a topology* $\Rightarrow$ *'a measure* $\Rightarrow$ *bool* **where**
inner-regular X M $\longleftrightarrow$ *sets M* = *sets* (*borel-of X*) $\wedge$ ($\forall\, A{\in}$*sets M*. *M A* = ($\bigsqcup C{\in}\{C.$
closedin X C $\wedge$ $C \subseteq A\}$. *M C*))

definition *outer-regular* :: *'a topology* $\Rightarrow$ *'a measure* $\Rightarrow$ *bool* **where**
outer-regular X M $\longleftrightarrow$ *sets M* = *sets* (*borel-of X*) $\wedge$ ($\forall\, A{\in}$*sets M*. *M A* = ($\bigsqcap C{\in}\{C.$
openin X C $\wedge$ $A \subseteq C\}$. *M C*))

definition *regular-measure* :: *'a topology* $\Rightarrow$ *'a measure* $\Rightarrow$ *bool* **where**
regular-measure X M $\longleftrightarrow$ *inner-regular X M* $\wedge$ *outer-regular X M*

lemma
  shows *inner-reguarI*: *sets M* = *sets* (*borel-of X*) $\Longrightarrow$ ($\bigwedge A.\ A \in$ *sets M*
      $\Longrightarrow$ *M A* = ($\bigsqcup C{\in}\{C.$ *closedin X C* $\wedge$ $C \subseteq A\}$. *M C*)) $\Longrightarrow$ *inner-regular X*
M
    and *inner-reguarD*: *inner-regular X M* $\Longrightarrow$ *sets M* = *sets* (*borel-of X*)
    inner-regular X M $\Longrightarrow$ $A \in$ *sets M* $\Longrightarrow$ *M A* = ($\bigsqcup C{\in}\{C.$ *closedin X C* $\wedge$ *C*
$\subseteq A\}$. *M C*)
  by(*auto simp*: *inner-regular-def*)

lemma
  shows *outer-reguarI*: *sets M* = *sets* (*borel-of X*)
     $\Longrightarrow$ ($\bigwedge A.\ A \in$ *sets M* $\Longrightarrow$ *M A* = ($\bigsqcap C{\in}\{C.$ *openin X C* $\wedge$ $A \subseteq C\}$. *M C*))
$\Longrightarrow$ *outer-regular X M*
    and *outer-reguarD*: *outer-regular X M* $\Longrightarrow$ *sets M* = *sets* (*borel-of X*)
    outer-regular X M $\Longrightarrow$ $A \in$ *sets M* $\Longrightarrow$ *M A* = ($\bigsqcap C{\in}\{C.$ *openin X C* $\wedge$ $A \subseteq$
$C\}$. *M C*)
  by(*auto simp*: *outer-regular-def*)

lemma
  shows *regular-measureI*: *inner-regular X M* ⟹ *outer-regular X M* ⟹ *regular-measure X M*
    and *regular-measureD*:
   regular-measure X M ⟹ *inner-regular X M regular-measure X M* ⟹ *outer-regular X M*
  by(*auto simp*: *regular-measure-def*)

lemma *inner-regular-finite-measure*:
  assumes *finite-measure M*
  shows *inner-regular X M* ⟷
        sets M = *sets* (*borel-of X*) ∧ (∀ *A*∈*sets M*. *measure M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*))
  unfolding *inner-regular-def*
proof *safe*
  interpret *M*: *finite-measure M* **by** *fact*
  fix *A*
  assume *A* ∈ *sets M* ∀ *A*∈*sets M*. *M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *M C*)
  then have *1*:*M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *M C*)
    by *blast*
  have *ennreal* (*measure M A*) = *ennreal* (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)
  proof −
    have *ennreal* (*measure M A*) = *M A*
      by (*simp add*: *M*.*emeasure-eq-measure*)
    also have ... = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *M C*)
      by *fact*
    also have ... = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *ennreal* (*measure M C*))
      by (*simp add*: *M*.*emeasure-eq-measure*)
    also have ... = *ennreal* (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)
      by(*intro ennreal-SUP*[*symmetric*]) (*use calculation* **in** *fastforce*)+
    finally show *?thesis* **.**
  qed
  moreover have (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*) ≥ *0*
    by(*subst le-cSUP-iff*)
     (*auto intro*!: *bdd-aboveI*[**where** *M*=*measure M* (*space M*)] *M*.*bounded-measure exI*[**where** *x*={}])
  ultimately show *measure M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)
    by *simp*
next
  interpret *M*: *finite-measure M* **by** *fact*
  fix *A*
  assume *A* ∈ *sets M* ∀ *A*∈*sets M*. *measure M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)
  then have *1*:*measure M A* = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)

by *blast*
show $M\ A = (\bigsqcup C{\in}\{C.\ closedin\ X\ C \wedge C \subseteq A\}.\ M\ C)$
proof −
have $M\ A = ennreal\ (measure\ M\ A)$
by(*rule M.emeasure-eq-measure*)
also have ... $= ennreal\ (\bigsqcup C{\in}\{C.\ closedin\ X\ C \wedge C \subseteq A\}.\ measure\ M\ C)$
by (*simp add*: *1*)
also have ... $= (\bigsqcup C{\in}\{C.\ closedin\ X\ C \wedge C \subseteq A\}.\ ennreal\ (measure\ M\ C))$
by(*intro ennreal-SUP*)
(*metis* (*mono-tags*, *lifting*) *M.emeasure-eq-measure M.emeasure-finite SUP-least emeasure-space top.extremum-unique*,*blast*)
finally show *?thesis*
by (*simp add*: *M.emeasure-eq-measure*)
qed
qed

lemma(**in** *finite-measure*)
shows *inner-regularI*: $sets\ M = sets\ (borel\text{-}of\ X) \Longrightarrow (\bigwedge A.\ A \in sets\ M \Longrightarrow measure\ M\ A = (\bigsqcup C{\in}\{C.\ closedin\ X\ C \wedge C \subseteq A\}.\ measure\ M\ C)) \Longrightarrow inner\text{-}regular\ X\ M$
and *inner-regularD*:
$inner\text{-}regular\ X\ M \Longrightarrow A \in sets\ M \Longrightarrow measure\ M\ A = (\bigsqcup C{\in}\{C.\ closedin\ X\ C \wedge C \subseteq A\}.\ measure\ M\ C)$
by(*auto simp*: *inner-regular-finite-measure finite-measure-axioms*)

lemma *outer-regular-finite-measure*:
assumes *finite-measure M*
shows $outer\text{-}regular\ X\ M \longleftrightarrow sets\ M = sets\ (borel\text{-}of\ X) \wedge (\forall A{\in}sets\ M.\ measure\ M\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ measure\ M\ C))$
unfolding *outer-regular-def*
proof *safe*
interpret *M*: *finite-measure M* **by** *fact*
fix *A*
assume $A{:}A \in sets\ M\ \forall A{\in}sets\ M.\ measure\ M\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ measure\ M\ C)$
and *sets-M*: $sets\ M = sets\ (borel\text{-}of\ X)$
then have $1{:}measure\ M\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ measure\ M\ C)$
by *blast*
have [*simp*]:*openin X* (*space M*)
by (*simp add*: *sets-M sets-eq-imp-space-eq space-borel-of*)
show $M\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ M\ C)$
proof −
have $enn2ereal\ (M\ A) = ereal\ (measure\ M\ A)$
by (*simp add*: *M.emeasure-eq-measure*)
also have ... $= ereal\ (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ measure\ M\ C)$
by (*simp add*: *1*)
also have ... $= (\bigsqcap\ (ereal\ `\ measure\ M\ `\ \{C.\ openin\ X\ C \wedge A \subseteq C\}))$
by(*intro ereal-Inf'*) (*auto intro*!: *bdd-belowI*[**where** *m=0*] *exI*[**where** *x=space M*] *sets.sets-into-space*[*OF A*(*1*)])

also have ... = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *enn2ereal* (*M C*))
by (*metis* (*no-types*, *lifting*) *M*.*emeasure-eq-measure enn2ereal-ennreal image-cong image-image measure-nonneg*)
also have ... = *enn2ereal* (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *M C*)
by (*simp add*: *Inf-ennreal.rep-eq image-image*)
finally show *?thesis*
using *enn2ereal-inject* **by** *blast*
qed
next
interpret *M*: *finite-measure M* **by** *fact*
fix *A*
assume *A*:*A* ∈ *sets M* ∀ *A*∈*sets M*. *M A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *M C*)
and *sets-M*: *sets M* = *sets* (*borel-of X*)
then have *1*:*M A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *M C*)
by *blast*
have [*simp*]:*openin X* (*space M*)
by (*simp add*: *sets-M sets-eq-imp-space-eq space-borel-of*)
show *measure M A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *measure M C*)
proof −
have *ereal* (*measure M A*) = *enn2ereal* (*M A*)
by (*simp add*: *M*.*emeasure-eq-measure*)
also have ... = *enn2ereal* (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *M C*)
by(*simp add*: *1*)
also have ... = (⨅ (*ereal* ‘ *measure M* ‘ {*C*. *openin X C* ∧ *A* ⊆ *C*}))
by(*auto simp*: *Inf-ennreal.rep-eq image-image M*.*emeasure-eq-measure*)
also have ... = *ereal* (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *measure M C*)
by(*intro ereal-Inf'*[*symmetric*]) (*auto intro*!: *bdd-belowI*[**where** *m*=*0*] *exI*[**where** *x*=*space M*] *sets.sets-into-space*[*OF A*(*1*)])
finally show *?thesis*
by *blast*
qed
qed

lemma(**in** *finite-measure*)
shows *outer-regularI*: *sets M* = *sets* (*borel-of X*) ⟹ (⋀*A*. *A* ∈ *sets M* ⟹ *measure M A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *measure M C*)) ⟹ *outer-regular X M*
and *outer-regularD*: *outer-regular X M* ⟹ *A* ∈ *sets M* ⟹ *measure M A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *measure M C*)
by(*auto simp*: *outer-regular-finite-measure finite-measure-axioms*)

Abstract version of ⟦*sets ?M* = *sets borel*; *emeasure ?M* (*space ?M*) ≠ ∞; *?B* ∈ *sets borel*⟧ ⟹ *emeasure ?M ?B* = ⨆ (*emeasure ?M* ‘ {*K*. *K* ⊆ *?B* ∧ *compact K*}) and ⟦*sets ?M* = *sets borel*; *emeasure ?M* (*space ?M*) ≠ ∞; *?B* ∈ *sets borel*⟧ ⟹ *emeasure ?M ?B* = ⨅ (*emeasure ?M* ‘ {*U*. *?B* ⊆ *U* ∧ *open U*}).

lemma(**in** *finite-measure*)
assumes *metrizable-space X sets* (*borel-of X*) = *sets M*

shows *inner-regular'*:*inner-regular X M*
and *outer-regular'*:*outer-regular X M*
proof −
let *?Sup* = λ*A*. (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A*}. *measure M C*)
let *?Inf* = λ*A*. (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *measure M C*)
{
fix *A*
assume *A*[*measurable*]: *A* ∈ *sets M*
obtain *d* **where** *d*: *Metric-space* (*topspace X*) *d Metric-space.mtopology* (*topspace X*) *d* = *X*
by (*metis Metric-space.topspace-mtopology assms*(*1*) *metrizable-space-def*)
then interpret *d*: *Metric-space topspace X d* **by** *simp*
have *sets*[*measurable* (*raw*)]: ⋀*A*. *openin X A* ⟹ *A* ∈ *sets M* ⋀*A*. *closedin X A* ⟹ *A* ∈ *sets M*
⋀*A*. *openin d.mtopology A* ⟹ *A* ∈ *sets M* ⋀*A*. *closedin d.mtopology A* ⟹ *A* ∈ *sets M*
by(*auto simp*: *d assms*(*2*)[*symmetric*] *dest*: *borel-of-open borel-of-closed*)
have *bdd*[*simp*]: ⋀*A*. *bdd-above* (*measure M* ‘ {*C*. *closedin X C* ∧ *C* ⊆ *A*})
⋀*A*. *bdd-below* (*measure M* ‘ {*C*. *closedin X C* ∧ *C* ⊆ *A*})
⋀*A*. *bdd-above* (*measure M* ‘ {*C*. *openin X C* ∧ *A* ⊆ *C*})
⋀*A*. *bdd-below* (*measure M* ‘ {*C*. *openin X C* ∧ *A* ⊆ *C*})
by(*auto intro*!: *bdd-aboveI*[**where** *M*=*measure M* (*space M*)] *bdd-belowI*[**where** *m*=*0*] *bounded-measure*)
have *ne*[*simp*]: {*C*. *closedin X C* ∧ *C* ⊆ *A*} ≠ {} *A* ∈ *sets M* ⟹ {*C*. *openin X C* ∧ *A* ⊆ *C*} ≠ {} **for** *A*
using *sets.sets-into-space*[*of A M*,*simplified space-borel-of*]
sets-eq-imp-space-eq[*OF assms*(*2*),*simplified space-borel-of*] **by** *blast*+
have *1*:*measure M A* ≤ *?Inf A measure M A* ≥ *?Sup A*
using *sets.sets-into-space*[*OF A*[*simplified assms*(*2*)[*symmetric*]],*simplified space-borel-of*]
openin-topspace closedin-topspace sets.sets-into-space[*OF A*]
by(*fastforce intro*!: *le-cInf-iff*[**where** *a*=*measure M A*
and *S*=*measure M* ‘ {*C*. *openin X C* ∧ *A* ⊆ *C*},*THEN iffD2*]
cSup-le-iff[**where** *a*=*measure M A*
and *S*=*measure M* ‘ {*C*. *closedin X C* ∧ *C* ⊆ *A*},*THEN iffD2*]
bdd-aboveI[**where** *M*=*measure M* (*space M*)] *bdd-belowI*[**where** *m*=*0*] *finite-measure-mono*)+
have *setsM*: *sigma-sets* (*topspace X*) {*U*. *closedin X U*} = *sets M*
using *sets-eq-imp-space-eq*[*OF assms*(*2*)] **by**(*auto simp*: *assms*(*2*)[*symmetric*] *sets-borel-of-closed*)
have *2*:*Int-stable* {*U*. *closedin X U*} {*U*. *closedin X U*} ⊆ *Pow* (*topspace X*)
by(*auto dest*: *closedin-subset intro*!: *Int-stableI*)

have *measure M A* ≤ *?Sup A* ∧ *measure M A* ≥ *?Inf A*
proof(*rule sigma-sets-induct-disjoint*[*OF 2 A*[*simplified setsM*[*symmetric*]]])
fix *a*
assume *a* ∈ {*U*. *closedin X U*}

then have *a*[*measurable*]: *closedin X a a* ∈ *sets M*
by(*auto simp*: *assms*(*2*)[*symmetric*] *borel-of-closed*)
show *measure M a* ≤ *?Sup a* ∧ *measure M a* ≥ *?Inf a*
proof (*cases a* = {})
case *empty*:*True*
thus *?thesis*
by(*auto intro*!: *cINF-lower*[**where** *f*=*measure M* **and** *x*={},*simplified*] *bdd-belowI*[**where** *m*=*0*]
simp: *empty*)
next
case *ne*:*False*
show *?thesis*
proof
have *measure M a* = *?Sup a*
by(*rule cSup-eq-maximum*[*symmetric*],*insert a*(*1*),*auto intro*!: *finite-measure-mono*)
thus *measure M a* ≤ *?Sup a* **by** *simp*
next
show *measure M a* ≥ *?Inf a*
proof −
have *?Inf a* ≤ (⨅*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / *Suc n*)))
proof(*rule cInf-superset-mono*)
show *range* (λ*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / *real* (*Suc n*)))) ⊆ *measure M* ‘ {*C*. *openin X C* ∧ *a* ⊆ *C*}
proof *clarify*
fix *n*
have (⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real n*))) ∈ {*C*. *openin X C* ∧ *a* ⊆ *C*}
using *d*.*openin-mball*[*simplified d*(*2*)] *closedin-subset*[*OF a*(*1*)] **by** *auto*
thus *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / (*Suc n*))) ∈ *measure M* ‘ {*C*. *openin X C* ∧ *a* ⊆ *C*}
by *auto*
qed
qed *auto*
also have ... = *measure M a*
proof −
have [*measurable*]: (⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real n*))) ∈ *sets M* **for** *n*
by(*auto simp*: *assms*(*2*)[*symmetric*] *d*.*openin-mball*[*simplified d*] *intro*!: *borel-of-open openin-clauses*(*3*))
have *0*:*decseq* (λ*n*. ⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real n*)))
by(*rule d*.*nbh-decseq*) (*auto intro*!: *decseq-SucI simp*: *frac-le*)
have *1*:*decseq* (λ*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real n*))))
by(*rule decseq-SucI*,*rule finite-measure-mono*) (*use decseq-SucD*[*OF 0*] **in** *auto*)
have *2*:(λ*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real n*)))) ⟶ (⨅*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / *Suc n*)))
by(*auto intro*!: *LIMSEQ-decseq-INF*[*OF* - *1*] *bdd-belowI*[**where** *m*=*0*])
moreover have (λ*n*. *measure M* (⋃*x*∈*a*. *d*.*mball x* (*1* / (*1* + *real*

n)))) $\longrightarrow$ *measure M a*
proof −
have ($\bigcap$*n*. ($\bigcup$*x*∈*a*. *d.mball x* (*1* / (*1* + *real n*)))) = *d.mtopology closure-of a*
by(*rule d.nbh-Inter-closure-of*[*OF ne*])
(*auto simp*: *closedin-subset*[*OF a*(*1*)] *frac-le*
intro!: *decseq-SucI LIMSEQ-inverse-real-of-nat*[*simplified inverse-eq-divide*,*simplified*])
also have ... = *a*
by(*auto simp*: *closure-of-eq d a*)
finally have ($\bigcap$*n*. ($\bigcup$*x*∈*a*. *d.mball x* (*1* / (*1* + *real n*)))) = *a* **.**
moreover have (λ*n*. *measure M* ($\bigcup$*x*∈*a*. *d.mball x* (*1* / (*1* + *real n*))))
$\longrightarrow$ *measure M* ($\bigcap$*n*. ($\bigcup$*x*∈*a*. *d.mball x* (*1* / (*1* + *real n*))))
by(*auto intro*!: *finite-Lim-measure-decseq simp*: *0*)
ultimately show *?thesis* **by** *simp*
qed
ultimately show *?thesis*
by(*auto dest*: *LIMSEQ-unique*)
qed
finally show *?Inf a* ≤ *measure M a* **.**
qed
qed
qed
next
show *measure M* {} ≤ *?Sup* {} ∧ *measure M* {} ≥ *?Inf* {}
by(*auto intro*!: *cINF-lower*[**where** *f*=*measure M* **and** *x*={},*simplified*] *bdd-belowI*[**where** *m*=*0*])
next
fix *a*
assume *a* ∈ *sigma-sets* (*topspace X*) {*U*. *closedin X U*}
and *ih*:*measure M a* ≤ *?Sup a* ∧ *measure M a* ≥ *?Inf a*
then have [*measurable*]:*a* ∈ *sets M*
by(*simp add*: *setsM*)
show *measure M* (*topspace X* − *a*) ≤ *?Sup* (*topspace X* − *a*) ∧ *measure M* (*topspace X* − *a*) ≥ *?Inf* (*topspace X* − *a*)
proof
show *measure M* (*topspace X* − *a*) ≤ *?Sup* (*topspace X* − *a*)
proof(*safe intro*!: *le-cSup-iff-less*[*THEN iffD2*])
fix *y*
assume *y* < *measure M* (*topspace X* − *a*)
then have *measure M a* < *measure M* (*space M*) − *y*
by(*auto simp*: *sets-eq-imp-space-eq*[*OF assms*(*2*),*simplified space-borel-of*] *finite-measure-compl*)
then obtain *U* **where** *U*: *openin X U a* ⊆ *U measure M U* ≤ *measure M* (*space M*) − *y*
using *ih* **by**(*auto simp*: *cInf-le-iff-less*[*OF ne*(*2*) *bdd*(*4*)])
show ∃ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *topspace X* − *a*}. *y* ≤ *Sigma-Algebra.measure*

```
M C
          proof(safe intro!: bexI[where x=topspace X − U])
            have [arith]:measure M a ≤ measure M U
              using U by(auto intro!: finite-measure-mono)
            show y ≤ measure M (topspace X − U)
                using U by(auto simp: sets-eq-imp-space-eq[OF assms(2),simplified
space-borel-of] finite-measure-compl)
          qed(use U in auto)
        qed auto
      next
        show ?Inf (topspace X − a) ≤ measure M (topspace X − a)
        proof(rule cInf-le-iff-less[THEN iffD2])
          show ∀y>measure M (topspace X − a). ∃C∈{C. openin X C ∧ topspace
X − a ⊆ C}. measure M C ≤ y
          proof safe
            fix y
            assume measure M (topspace X − a) < y
            then have measure M (space M) − y < measure M a
            by(auto simp: sets-eq-imp-space-eq[OF assms(2),simplified space-borel-of]
finite-measure-compl)
            then obtain C where C: closedin X C C ⊆ a measure M (space M) −
y ≤ measure M C
              using ih by(auto simp: le-cSup-iff-less[OF ne(1) bdd(1)])
            show ∃C∈{C. openin X C ∧ topspace X − a ⊆ C}. measure M C ≤ y
            proof(safe intro!: bexI[where x=topspace X − C])
              have [arith]:measure M C ≤ measure M a
                using C by(auto intro!: finite-measure-mono)
              show measure M (topspace X − C) ≤ y
                using C by(auto simp: sets-eq-imp-space-eq[OF assms(2),simplified
space-borel-of] finite-measure-compl)
            qed(use C in auto)
          qed
        qed auto
      qed
    next
      fix a :: nat ⇒ -
      assume h: disjoint-family a range a ⊆ sigma-sets (topspace X) {U. closedin
X U}
        and ih: ⋀i. measure M (a i) ≤ ?Sup (a i) ∧ ?Inf (a i) ≤ measure M (a i)
      then have a[measurable]: ⋀i. a i ∈ sets M
        by(simp add: setsM)
      show measure M (⋃i. a i) ≤ ?Sup (⋃i. a i) ∧ ?Inf (⋃i. a i) ≤ measure M
(⋃i. a i)
      proof
        show measure M (⋃i. a i) ≤ ?Sup (⋃i. a i)
        proof(rule le-cSup-iff-less[THEN iffD2])
          show ∀y< measure M (⋃ (range a)). ∃C∈{C. closedin X C ∧ C ⊆ ⋃
(range a)}. y ≤ measure M C
          proof safe
```

fix *y*
assume $y < measure\ M\ (\bigcup i.\ a\ i)$
also have ... = $(\sum i.\ measure\ M\ (a\ i))$
by(*rule suminf-measure*[*OF - h*(*1*),*symmetric*]) *auto*
finally obtain *N* **where** *N*: $y < (\sum i{<}N.\ measure\ M\ (a\ i))$
by (*meson linorder-not-less measure-nonneg suminf-le-const summableI-nonneg-bounded*)
consider $N = 0 \mid N > 0$ **by** *auto*
then show $\exists C\in\{C.\ closedin\ X\ C \wedge C \subseteq \bigcup\ (range\ a)\}.\ y \leq measure$ $M\ C$
proof *cases*
case *1*
with *N* **show** *?thesis* **by**(*auto intro*!: *exI*[**where** *x*={}])
next
case [*arith*]:*2*
define *e* **where** $e \equiv ((\sum i{<}N.\ measure\ M\ (a\ i)) - y)\ /\ N$
have *e*[*arith*]: $e > 0$
using *N* **by**(*auto simp*: *e-def*)
hence $\bigwedge i.\ measure\ M\ (a\ i) - e < measure\ M\ (a\ i)$ **by** *auto*
hence $\forall i.\ \exists Ci.\ closedin\ X\ Ci \wedge Ci \subseteq a\ i \wedge measure\ M\ (a\ i) - e \leq$ $measure\ M\ Ci$
using *ih*[*simplified le-cSup-iff-less*[*OF ne*(*1*) *bdd*(*1*)]] **by** *auto*
then obtain *Ci* **where** *Ci*: $\bigwedge i.\ closedin\ X\ (Ci\ i)$
$\bigwedge i.\ Ci\ i \subseteq a\ i$ $\bigwedge i.\ measure\ M\ (a\ i) - e \leq measure\ M\ (Ci\ i)$
by *metis*
with *h* **have** *Ci-d*:*disjoint-family-on Ci* {..<*N*}
by(*auto simp*: *disjoint-family-on-def*) *blast*
show *?thesis*
proof(*safe intro*!: *bexI*[**where** $x{=}\bigcup\ (Ci\ `\ \{..{<}N\})$])
have $y \leq (\sum i{<}N.\ measure\ M\ (a\ i)) - ((\sum i{<}N.\ measure\ M\ (a\ i))$ $-\ y)$ **by** *auto*
also have ... $\leq (\sum i{<}N.\ measure\ M\ (a\ i) - e)$
by(*auto simp*: *e-def sum-subtractf*)
also have ... $\leq (\sum i{<}N.\ measure\ M\ (Ci\ i))$
using *Ci* **by**(*auto intro*!: *sum-mono*)
also have ... $= measure\ M\ (\bigcup\ (Ci\ `\ \{..{<}N\}))$
by(*rule finite-measure-finite-Union*[*OF - - Ci-d*,*symmetric*]) (*use Ci* **in** *auto*)
finally show $y \leq measure\ M\ (\bigcup\ (Ci\ `\ \{..{<}N\}))$ **.**
qed(*insert Ci*,*auto intro*!: *closedin-Union*)
qed
qed
qed *auto*
next
show $?Inf\ (\bigcup i.\ a\ i) \leq measure\ M\ (\bigcup i.\ a\ i)$
proof(*rule cInf-le-iff-less*[*THEN iffD2*])
show $\forall y{>}\ measure\ M\ (\bigcup\ (range\ a)).\ \exists C\in\{C.\ openin\ X\ C \wedge \bigcup\ (range$ $a) \subseteq C\}.\ measure\ M\ C \leq y$
proof *safe*
fix *y*

assume *1*:*measure M* (⋃*i. a i*) < *y*
define *en* **where** *en* ≡ (λ*n.* (*y* − *measure M* (⋃*i. a i*)) ∗ (*1* / *2*) ^ (*Suc n*))
with *1* **have** [*arith*]:*en n* > *0* **for** *n* **by** *auto*
hence *measure M* (*a i*) < *measure M* (*a i*) + *en i* **for** *i* **by** *auto*
hence ∃ *Ui. openin X Ui* ∧ *a i* ⊆ *Ui* ∧ *measure M Ui* ≤ *measure M* (*a i*) + *en i* **for** *i*
using *ih*[*of i*,*simplified cInf-le-iff-less*[*OF ne*(*2*)[*OF* ‹*a i* ∈ *sets M*›] *bdd*(*4*)]] **by** *auto*
then obtain *Ui* **where** *Ui*: ⋀*i. openin X* (*Ui i*) ⋀*i. a i* ⊆ *Ui i*
⋀*i. measure M* (*Ui i*) ≤ *measure M* (*a i*) + *en i*
by *metis*
have [*simp*]: *summable en summable* (λ*n. measure M* (*a n*))
by(*auto simp*: *en-def intro*!: *summable-measure h*)
hence [*simp*]: *summable* (λ*n. measure M* (*a n*) + *en n*)
by(*auto intro*!: *summable-add*)
have [*simp*]:*summable* (λ*n. measure M* (*Ui n*))
using *Ui* **by**(*auto intro*!: *summable-comparison-test-ev*[*OF* - ‹*summable* (λ*n. measure M* (*a n*) + *en n*)›])
show ∃ *C*∈{*C. openin X C* ∧ ⋃ (*range a*) ⊆ *C*}. *measure M C* ≤ *y*
proof(*safe intro*!: *bexI*[**where** *x*=⋃*i. Ui i*])
have *measure M* (⋃*i. Ui i*) ≤ (∑*i. measure M* (*Ui i*))
using *Ui* **by**(*auto intro*!: *finite-measure-subadditive-countably*)
also have ... ≤ (∑*i. measure M* (*a i*) + *en i*)
by(*auto intro*!: *suminf-le Ui*)
also have ... = (∑*i. measure M* (*a i*)) + (∑*i. en i*)
by(*simp add*: *suminf-add*)
also have ... = *measure M* (⋃*i. a i*) + (*y* − *measure M* (⋃*i. a i*))
proof −
have [*simp*]:(∑*i. measure M* (*a i*)) = *measure M* (⋃*i. a i*)
by(*auto intro*!: *suminf-measure h*)
have (∑*i. en i*) = (*y* − *Sigma-Algebra.measure M* (⋃ (*range a*))) / *2* ∗ (∑*n.* (*1* / *2*) ^ *n*)
by(*simp only*: *suminf-mult*[*of* λ*n.* (*1* / *2*) ^ *n* :: *real*,*simplified*,*symmetric*]) (*simp add*: *en-def*)
also have ... = (*y* − *measure M* (⋃*i. a i*))
by(*simp add*: *suminf-geometric*)
finally show *?thesis* **by** *simp*
qed
finally show *measure M* (⋃*i. Ui i*) ≤ *y* **by** *simp*
qed(*use Ui* **in** *auto*)
qed
show {*C. openin X C* ∧ ⋃ (*range a*) ⊆ *C*} ≠ {}
using *sets.sets-into-space*[*OF a*]
by(*force intro*!: *exI*[**where** *x*=*topspace X*] *simp*: *sets-eq-imp-space-eq*[*OF assms*(*2*),*simplified space-borel-of*])
qed *auto*
qed
qed

```
    note 1 this
  }
  with assms(2) show inner-regular X M outer-regular X M
    by (fastforce intro!: inner-regularI outer-regularI)+
qed

definition tight-on-set :: 'a topology ⇒ 'a measure set ⇒ bool where
tight-on-set X Γ ⟷ (∀ M∈Γ. finite-measure M ∧ sets (borel-of X) = sets M) ∧
                    (∀ e>0. ∃ K. compactin X K ∧ (∀ M∈Γ. measure M (space M −
K) < e))

abbreviation tight-on :: 'a topology ⇒ 'a measure ⇒ bool where
tight-on X M ≡ tight-on-set X {M}

lemma tight-on-def:
tight-on X M ⟷ finite-measure M ∧ sets (borel-of X) = sets M ∧
                (∀ e>0. ∃ K. compactin X K ∧ measure M (space M − K) < e)
  by(auto simp: tight-on-set-def)

lemma tight-on-set-subset: A ⊆ B ⟹ tight-on-set X B ⟹ tight-on-set X A
  unfolding tight-on-set-def by blast

lemma tight-on-tight: tight-on-set euclidean (Mi ‘ UNIV) ∧ (∀ i. real-distribution
(Mi i)) ⟷ tight Mi
proof safe
  assume h:tight-on-set euclideanreal (range Mi) ∀ i. real-distribution (Mi i)
  show tight Mi
    unfolding tight-def
  proof safe
    fix e :: real
    assume e: e > 0
    with h(1) obtain K where K:
     compact K ⋀i. measure (Mi i) (space (Mi i) − K) < e
      by(auto simp: tight-on-set-def)
    obtain r where r:
      r > 0 K ⊆ ball 0 r
      by(metis bounded-subset-ballD[OF compact-imp-bounded[OF K(1)]])
    show ∃ a b. a < b ∧ (∀ n. 1 − e < measure (Mi n) {a<..b})
    proof(rule exI[where x=−r])
      show ∃ b>− r. ∀ n. 1 − e < measure (Mi n) {− r<..b}
      proof(safe intro!: exI[where x=r])
        fix n
        interpret real-distribution Mi n
          using h by simp
        have [measurable]: K ∈ sets (Mi n)
          by (simp add: K(1) borel-compact)
        hence 1 − e < prob K
          using K(2)[of n] by(simp add: prob-compl del: borel-UNIV)
        also have ... ≤ prob {− r<..<r}
```

```
      using r by(auto intro!: finite-measure-mono simp: ball-eq-greaterThanLessThan)
       also have ... ≤ prob {−r<..r}
         by(auto intro!: finite-measure-mono)
       finally show 1 − e < prob {−r<..r} .
     qed(use r in auto)
   qed
 qed(use h in simp)
next
 assume h:tight Mi
 show tight-on-set euclideanreal (range Mi)
   unfolding tight-on-set-def
 proof safe
   fix e :: real
   assume e: e > 0
   with h obtain a b where ab: a < b ⋀n. measure (Mi n) {a<..b} > 1 − e
     by(auto simp: tight-def)
   show ∃K. compactin euclideanreal K ∧ (∀M∈range Mi. measure M (space M
− K) < e)
   proof(safe intro!: exI[where x={a..b}])
     fix n
     interpret real-distribution Mi n
       using h by(auto simp: tight-def)
     have prob (space (Mi n) − {a..b}) = 1 − prob {a..b}
       by(rule prob-compl) simp
     also have ... ≤ 1 − prob {a<..b}
       by(auto intro!: finite-measure-mono)
     also have ... < e
       using ab(2)[of n] by auto
     finally show prob (space (Mi n) − {a..b}) < e .
   qed simp
 qed(insert h,auto simp: borel-of-euclidean tight-def real-distribution-def real-distribution-axioms-def
prob-space-def)
qed(auto simp: tight-def)

lemma inner-regular'':
 assumes metrizable-space X tight-on X M
     and [measurable]:A ∈ sets M
   shows measure M A = (⨆K∈{K. compactin X K ∧ K ⊆ A}. measure M K)
(is - = ?rhs)
proof −
 have sets: sets (borel-of X) = sets M
   using assms(2) by(simp add: tight-on-def)
 interpret M: finite-measure M
   using assms(2) by(simp add: tight-on-def)
 have measure M A ≥ ?rhs
   using sets.sets-into-space[OF assms(3)]
  by(auto intro!: cSup-le-iff[THEN iffD2] M.finite-measure-mono bdd-aboveI[where
M=measure M (space M)])
 moreover have measure M A ≤ ?rhs
```

proof −
have *measure M A* − *e* < *?rhs* **if** *e*[*arith*]: *e* > *0* **for** *e*
proof −
obtain *K* **where** *K*: *compactin X K measure M* (*space M* − *K*) < *e*
using *assms*(*2*)[*simplified tight-on-def*] *e* **by** *metis*
hence [*measurable*]: *K* ∈ *sets M*
by(*auto simp*: *sets*[*symmetric*]
intro!: *borel-of-closed compactin-imp-closedin*[*OF metrizable-imp-Hausdorff-space*[*OF assms*(*1*)]]])
have *measure M A* − *e* < *measure M A* − *measure M* (*space M* − *K*)
using *K* **by** *auto*
also have ... ≤ *measure M* (*A* ∩ *K*)
by (*metis Diff-mono M.finite-measure-Diff′ M.finite-measure-mono* ‹*K* ∈ *sets M*› *assms*(*3*) *cancel-ab-semigroup-add-class.diff-right-commute dual-order.refl le-iff-diff-le-0 sets.Diff sets.sets-into-space sets.top*)
also have ... = (⨆ *C*∈{*C*. *closedin X C* ∧ *C* ⊆ *A* ∩ *K*}. *measure M C*)
by(*rule M.inner-regularD*[*OF M.inner-regular′*[*OF assms*(*1*) *sets*]]) *measurable*
also have ... ≤ *?rhs*
proof(*rule cSup-mono*)
show ⋀*b*. *b* ∈ *Sigma-Algebra.measure M* ‘ {*C*. *closedin X C* ∧ *C* ⊆ *A* ∩ *K*}
⟹ ∃ *a*∈*Sigma-Algebra.measure M* ‘ {*K*. *compactin X K* ∧ *K* ⊆ *A*}. *b* ≤ *a*
proof *safe*
fix *C*
assume *closedin X C C* ⊆ *A* ∩ *K*
then show ∃ *a*∈*Sigma-Algebra.measure M* ‘ {*K*. *compactin X K* ∧ *K* ⊆ *A*}. *measure M C* ≤ *a*
by(*auto intro*!: *closed-compactin*[*OF K*(*1*)])
qed
qed(*auto intro*!: *bdd-aboveI*[**where** *M*=*measure M* (*space M*)] *M.bounded-measure*)
finally show *?thesis* **.**
qed
thus *?thesis*
by (*metis* (*full-types*) *cancel-ab-semigroup-add-class.diff-right-commute dual-order.refl le-iff-diff-le-0 less-iff-diff-less-0 linorder-not-less*)
qed
ultimately show *?thesis* **by** *simp*
qed

lemma(**in** *finite-measure*) *tight-on-compact-space*:
assumes *metrizable-space X compact-space X sets* (*borel-of X*) = *sets M*
shows *tight-on X M*
using *assms*(*1*,*2*)
by(*auto simp*: *tight-on-def assms finite-measure-axioms sets-eq-imp-space-eq*[*OF assms*(*3*)[*symmetric*]]
compact-space-def space-borel-of
intro!: *exI*[**where** *x*=*space M*])

lemma(**in** *finite-measure*) *tight-on-finite-space*:
  assumes *metrizable-space X sets* (*borel-of X*) = *sets M finite* (*space M*)
  shows *tight-on X M*
proof −
  from *assms*(*3*) **have** *compact-space X*
   by(*auto simp*: *assms compact-space-def sets-eq-imp-space-eq*[*OF assms*(*2*)[*symmetric*]] *space-borel-of*
       intro!: *finite-imp-compactin-eq*[*THEN iffD2*])
  from *tight-on-compact-space*[*OF assms*(*1*) *this assms*(*2*)] **show** *?thesis* **.**
qed

lemma(**in** *finite-measure*) *tight-on-Polish*:
  assumes *Polish-space X sets* (*borel-of X*) = *sets M*
  shows *tight-on X M*
proof(*cases finite* (*space M*))
  case *True*
  then show *?thesis*
    by(*auto intro*!: *tight-on-finite-space assms Polish-space-imp-metrizable-space*)
next
  case *inf*:*False*
  then have *inf2*: *infinite* (*topspace X*)
    by(*auto simp*: *sets-eq-imp-space-eq*[*OF assms*(*2*)[*symmetric*]] *space-borel-of*)
  obtain *d* **where** *d*:*Metric-space* (*topspace X*) *d Metric-space.mtopology* (*topspace X*) *d* = *X*
    Metric-space.mcomplete (*topspace X*) *d*
   by (*metis Metric-space.topspace-mtopology assms*(*1*) *completely-metrizable-space-def Polish-space-imp-completely-metrizable-space*)
  interpret *d*: *Metric-space topspace X d* **by** *fact*
  have [*measurable*]:$\bigwedge$*a e*. *d.mball a e* ∈ *sets M* $\bigwedge$*a e*. *d.mcball a e* ∈ *sets M*
     using *d.openin-mball d.closedin-mcball* **by**(*auto simp*: *assms*(*2*)[*symmetric*] *borel-of-open borel-of-closed d*)
  show *?thesis*
    unfolding *tight-on-def*
  proof *safe*
    fix *e* :: *real*
    assume *e*: *e* > *0*
    from *assms* **obtain** *U* **where** *U*: *countable U dense-in X U*
      by(*auto simp*: *separable-space-def2 Polish-space-def*)
    have *U-ne*: *U* ≠ {}
      by (*metis U*(*2*) *dense-in-nonempty inf2 infinite-imp-nonempty*)
    let *?an* = *from-nat-into U*
    have *an*:$\bigwedge$*n*. *?an n* ∈ *U*
      by (*simp add*: *U-ne from-nat-into*)
    have *anU*: ($\bigcup$*n*. *d.mball* (*?an n*) *e'*) = *topspace X* **if** *e'* > *0* **for** *e'*
    proof −
      have ($\bigcup$*n*. *d.mball* (*?an n*) *e'*) = ($\bigcup$*u*∈*U*. *d.mball u e'*)
        by(*auto simp*: *UN-from-nat-into*[*OF U*(*1*) *U-ne*])
      also have ... = *topspace X*
        by(*rule d.mdense-balls-cover*[*simplified d*,*OF U*(*2*) *that*])

finally show *?thesis* **.**
qed
have ∃ *n. measure M* (⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*)) > *measure M* (*space M*) − *e* ∗ (*1* / *2*)^*Suc m* **for** *m*
proof −
have *1*:(λ*n. measure M* (⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*)))
⟶ *measure M* (⋃ *n*. ⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*))
by(*rule finite-Lim-measure-incseq*) (*fastforce simp*: *incseq-def*)+
have (⋃ *n*. ⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*)) = (⋃ *n*. *d.mball* (*?an n*) (*1* / *Suc m*)) **by** *blast*
also have ... = *topspace X*
by(*rule anU*) *auto*
also have ... = *space M*
by(*simp add*: *sets-eq-imp-space-eq*[*OF assms*(*2*),*simplified space-borel-of*])
finally have (λ*n. measure M* (⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*))) ⟶ *measure M* (*space M*)
using *1* **by** *simp*
moreover have *e* ∗ (*1* / *2*)^*Suc m* > *0* **using** *e* **by** *auto*
ultimately have ∃ *N*. ∀ *n*≥*N*. |*measure M* (⋃ *i*∈{..<*n*}. *d.mball* (*?an i*) (*1* / *Suc m*)) − *measure M* (*space M*)| < *e* ∗ (*1*/*2*)^*Suc m*
unfolding *LIMSEQ-def dist-real-def* **by** *metis*
then obtain *N* **where** *measure M* (*space M*) − *measure M* (⋃ *i*∈{..<*N*}. *d.mball* (*?an i*) (*1* / *Suc m*)) < *e* ∗ (*1*/*2*)^*Suc m*
using *bounded-measure* **by** *auto*
thus *?thesis*
by(*auto intro*!: *exI*[**where** *x*=*N*])
qed
then obtain *n* **where** *n*: ⋀*m*. *measure M* (⋃ *i*∈{..<*n m*}. *d.mball* (*?an i*) (*1* / *Suc m*)) > *measure M* (*space M*) − *e* ∗ (*1* / *2*)^*Suc m*
by *metis*
have *n*′: ⋀*m*. *measure M* (⋃ *i*∈{..<*n m*}. *d.mcball* (*?an i*) (*1* / *Suc m*)) > *measure M* (*space M*) − *e* ∗ (*1* / *2*)^*Suc m*
by(*rule order.strict-trans2*[*OF n*]) (*auto intro*!: *finite-measure-mono*)
define *K* **where** *K* ≡ ⋂ *m*. ⋃ *k*∈{..<*n m*}. *d.mcball* (*?an k*) (*1* / *Suc m*)
have *K-closed*: *closedin d.mtopology K*
by(*auto intro*!: *closedin-Union simp*: *K-def*)
have *K-compact*: *compactin d.mtopology K*
proof −
have *d.mtotally-bounded K*
unfolding *d.mtotally-bounded-def2*
proof *safe*
fix *e*′ :: *real*
assume [*arith*]:*e*′ > *0*
then obtain *m* **where** *m*[*arith*]: *1* / *Suc m* < *e*′
using *nat-approx-posE* **by** *blast*
have *K* ⊆ (⋃ *k*∈{..<*n m*}. *d.mcball* (*?an k*) (*1* / *Suc m*))
by(*auto simp*: *K-def*)
also have ... ⊆ (⋃ *k*∈{..<*n m*}. *d.mball* (*?an k*) *e*′)
using *m* **by** *auto*

```
        finally show ∃Ka. finite Ka ∧ Ka ⊆ topspace X ∧ K ⊆ (⋃x∈Ka. d.mball x e')
          using an dense-in-subset[OF U(2)] by(fastforce intro!: exI[where x=?an ` {..<n m}])
      qed
      thus ?thesis
      by(simp add: d.mtotally-bounded-eq-compact-closedin[OF d(3) K-closed,simplified])
    qed
    show ∃K. compactin X K ∧ measure M (space M − K) < e
    proof(safe intro!: exI[where x=K])
      have sum:summable (λm. measure M (space M − (⋃k∈{..<n m}. d.mcball (?an k) (1 / Suc m))))
        apply(intro summable-comparison-test-ev[OF - summable-mult[OF complete-algebra-summable-geometric[where x=1 / 2]],of - e] exI[where x=1])
        apply(simp add: eventually-sequentially finite-measure-compl)
        apply(intro exI[where x=1] allI)
        subgoal for l
          using n'[of l] e bounded-measure
          apply(auto intro!: order.strict-implies-order[OF order.strict-trans[where b=e∗ (1 / 2)^Suc l]])
          done
        by simp
      have measure M (space M − K) = measure M (⋃m. (space M − (⋃k∈{..<n m}. d.mcball (?an k) (1 / Suc m))))
        by(auto simp: K-def)
      also have ... ≤ (∑m. measure M (space M − (⋃k∈{..<n m}. d.mcball (?an k) (1 / Suc m))))
        by(rule finite-measure-subadditive-countably) (use sum in auto)
      also have ... = measure M (space M − (⋃k∈{..<n 0}. d.mcball (?an k) (1 / Suc 0)))
              + (∑m. measure M (space M − (⋃k∈{..<n (Suc m)}. d.mcball (?an k) (1 / Suc (Suc m)))))
        using suminf-split-initial-segment[OF sum,of 1] by simp
      also have ... < e ∗ (1 / 2)
              + (∑m. measure M (space M − (⋃k∈{..<n (Suc m)}. d.mcball (?an k) (1 / Suc (Suc m)))))
        using n'[of 0] by(simp add: finite-measure-compl)
      also have ... ≤  e ∗ (1 / 2) + (∑m. e ∗ (1 / 2) ^(Suc (Suc m)))
      proof −
        have (∑m. measure M (space M − (⋃k∈{..<n (Suc m)}. d.mcball (?an k) (1 / Suc (Suc m))))) ≤ (∑m. e ∗ (1 / 2) ^(Suc (Suc m)))
        proof(rule suminf-le)
          fix l
          show measure M (space M − (⋃k<n (Suc l). d.mcball (?an k) (1 / real (Suc (Suc l))))) ≤ e ∗ (1 / 2) ^Suc (Suc l)
            using n'[of Suc l] by (auto simp: finite-measure-compl)
        qed(use summable-Suc-iff[THEN iffD2,OF sum] in auto)
        thus ?thesis by simp
      qed
```

also have ... = *e*
by(*simp add*: *suminf-geometric*[*of 1 / 2* :: *real*] *suminf-mult suminf-divide*)
finally show *measure M* (*space M* − *K*) < *e* **.**
qed(*use K-compact d* **in** *auto*)
qed(*use finite-measure-axioms assms* **in** *auto*)
qed

corollary(**in** *finite-measure*) *inner-regular-Polish*:
assumes *Polish-space X sets* (*borel-of X*) = *sets M A* ∈ *sets M*
shows *measure M A* = ($\bigsqcup$*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *measure M K*)
by(*auto intro*!: *tight-on-Polish inner-regular′′ simp*: *assms Polish-space-imp-metrizable-space*)

end

3 The Riesz Representation Theorem

theory *Riesz-Representation*
imports *Regular-Measure*
Urysohn-Locally-Compact-Hausdorff
begin

3.1 Lemmas for Complex-Valued Continuous Maps

lemma *continuous-map-Re′*[*simp*,*continuous-intros*]: *continuous-map euclidean euclideanreal Re*
and *continuous-map-Im′*[*simp*,*continuous-intros*]: *continuous-map euclidean euclideanreal Im*
and *continuous-map-complex-of-real′*[*simp*,*continuous-intros*]: *continuous-map euclideanreal euclidean complex-of-real*
by(*auto simp*: *continuous-on tendsto-Re tendsto-Im*)

corollary
assumes *continuous-map X euclidean f*
shows *continuous-map-Re*[*simp*,*continuous-intros*]: *continuous-map X euclideanreal* (*λx*. *Re* (*f x*))
and *continuous-map-Im*[*simp*,*continuous-intros*]: *continuous-map X euclideanreal* (*λx*. *Im* (*f x*))
by(*auto intro*!: *continuous-map-compose*[*OF assms*,*simplified comp-def*] *continuous-map-Re′ continuous-map-Im′*)

lemma *continuous-map-of-real-iff* [*simp*]:
continuous-map X euclidean (*λx*. *of-real* (*f x*) :: - :: *real-normed-div-algebra*) ⟷ *continuous-map X euclideanreal f*
by(*auto simp*: *continuous-map-atin tendsto-of-real-iff*)

lemma *continuous-map-complex-mult* [*continuous-intros*]:
fixes *f* :: ′*a* ⇒ *complex*
shows ⟦*continuous-map X euclidean f*; *continuous-map X euclidean g*⟧ ⟹ *continuous-map X euclidean* (*λx*. *f x* ∗ *g x*)

by (*simp add*: *continuous-map-atin tendsto-mult*)

lemma *continuous-map-complex-mult-left*:
fixes *f* :: *'a* ⇒ *complex*
shows *continuous-map X euclidean f* ⟹ *continuous-map X euclidean* (*λx. c* ∗ *f x*)
by(*simp add*: *continuous-map-atin tendsto-mult*)

lemma *complex-continuous-map-iff*:
continuous-map X euclidean f ⟷ *continuous-map X euclideanreal* (*λx. Re* (*f x*)) ∧ *continuous-map X euclideanreal* (*λx. Im* (*f x*))
proof *safe*
assume *continuous-map X euclideanreal* (*λx. Re* (*f x*)) *continuous-map X euclideanreal* (*λx. Im* (*f x*))
then have *continuous-map X euclidean* (*λx. Re* (*f x*) + i ∗ *Im* (*f x*))
by(*auto intro*!: *continuous-map-add continuous-map-complex-mult-left continuous-map-compose*[*of X euclideanreal*,*simplified comp-def*])
thus *continuous-map X euclidean f*
using *complex-eq* **by** *auto*
qed(*use continuous-map-compose*[*OF - continuous-map-Re'*,*simplified comp-def*] *continuous-map-compose*[*OF - continuous-map-Im'*,*simplified comp-def*] **in** *auto*)

lemma *complex-integrable-iff*: *complex-integrable M f* ⟷ *integrable M* (*λx. Re* (*f x*)) ∧ *integrable M* (*λx. Im* (*f x*))
proof *safe*
assume *h*[*measurable*]:*integrable M* (*λx. Re* (*f x*)) *integrable M* (*λx. Im* (*f x*))
show *complex-integrable M f*
unfolding *integrable-iff-bounded*
proof *safe*
show *f*[*measurable*]:*f* ∈ *borel-measurable M*
using *borel-measurable-complex-iff h* **by** *blast*
have ($\int^+$ *x. ennreal* (*cmod* (*f x*)) ∂*M*) ≤ ($\int^+$ *x. ennreal* (|*Re* (*f x*)| + |*Im* (*f x*)|) ∂*M*)
by(*intro nn-integral-mono ennreal-leI*) (*use cmod-le* **in** *auto*)
also have ... = ($\int^+$ *x. ennreal* |*Re* (*f x*)| ∂*M*) + ($\int^+$ *x. ennreal* |*Im* (*f x*)| ∂*M*)
by(*auto intro*!: *nn-integral-add*)
also have ... < ∞
using *h* **by**(*auto simp*: *integrable-iff-bounded*)
finally show ($\int^+$ *x. ennreal* (*cmod* (*f x*)) ∂*M*) < ∞ **.**
qed
qed(*auto dest*: *integrable-Re integrable-Im*)

3.2 Compact Supports

definition *has-compact-support-on* :: (*'a* ⇒ *'b* :: *monoid-add*) ⇒ *'a topology* ⇒ *bool*
(**infix** ‹*has'-compact'-support'-on*› *60*) **where**
f has-compact-support-on X ⟷ *compactin X* (*X closure-of support-on* (*topspace*

X) *f*)

lemma *has-compact-support-on-iff*:
f has-compact-support-on X ⟷ *compactin X* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*})
by(*simp add*: *has-compact-support-on-def support-on-def*)

lemma *has-compact-support-on-zero*[*simp*]: (λ*x*. *0*) *has-compact-support-on X*
by(*simp add*: *has-compact-support-on-iff*)

lemma *has-compact-support-on-compact-space*[*simp*]: *compact-space X* ⟹ *f has-compact-support-on X*
by(*auto simp*: *has-compact-support-on-def closedin-compact-space*)

lemma *has-compact-support-on-add*[*simp*,*intro*!]:
assumes *f has-compact-support-on X g has-compact-support-on X*
shows (λ*x*. *f x* + *g x*) *has-compact-support-on X*
proof −
have *support-on* (*topspace X*) (λ*x*. *f x* + *g x*)
⊆ *support-on* (*topspace X*) *f* ∪ *support-on* (*topspace X*) *g*
by(*auto simp*: *in-support-on*)
moreover have *compactin X* (*X closure-of* ...)
using *assms* **by**(*simp add*: *has-compact-support-on-def compactin-Un*)
ultimately show *?thesis*
unfolding *has-compact-support-on-def* **by** (*meson closed-compactin closedin-closure-of closure-of-mono*)
qed

lemma *has-compact-support-on-sum*:
assumes *finite I* ⋀*i*. *i* ∈ *I* ⟹ *f i has-compact-support-on X*
shows (λ*x*. (∑ *i*∈*I*. *f i x*)) *has-compact-support-on X*
proof −
have *support-on* (*topspace X*) (λ*x*. (∑ *i*∈*I*. *f i x*)) ⊆ (⋃ *i*∈*I*. *support-on* (*topspace X*) (*f i*))
by(*simp add*: *subset-eq*) (*meson in-support-on sum.neutral*)
moreover have *compactin X* (*X closure-of* ...)
using *assms* **by**(*auto simp*: *has-compact-support-on-def closure-of-Union intro*!: *compactin-Union*)
ultimately show *?thesis*
unfolding *has-compact-support-on-def* **by** (*meson closed-compactin closedin-closure-of closure-of-mono*)
qed

lemma *has-compact-support-on-mult-left*:
fixes *g* :: - ⇒ - :: *mult-zero*
assumes *g has-compact-support-on X*
shows (λ*x*. *f x* ∗ *g x*) *has-compact-support-on X*
proof −
have *support-on* (*topspace X*) (λ*x*. *f x* ∗ *g x*) ⊆ *support-on* (*topspace X*) *g*

by(*auto simp add*: *in-support-on*)
thus *?thesis*
using *assms* **unfolding** *has-compact-support-on-def*
by (*meson closed-compactin closedin-closure-of closure-of-mono*)
qed

lemma *has-compact-support-on-mult-right*:
fixes *f* :: - ⇒ - :: *mult-zero*
assumes *f has-compact-support-on X*
shows (*λx. f x* ∗ *g x*) *has-compact-support-on X*
proof −
have *support-on* (*topspace X*) (*λx. f x* ∗ *g x*) ⊆ *support-on* (*topspace X*) *f*
by(*auto simp add*: *in-support-on*)
thus *?thesis*
using *assms* **unfolding** *has-compact-support-on-def*
by (*meson closed-compactin closedin-closure-of closure-of-mono*)
qed

lemma *has-compact-support-on-uminus-iff* [*simp*]:
fixes *f* :: - ⇒ - :: *group-add*
shows (*λx.* − *f x*) *has-compact-support-on X* ⟷ *f has-compact-support-on X*
by(*auto simp*: *has-compact-support-on-def support-on-def*)

lemma *has-compact-support-on-diff* [*simp*,*intro*!]:
fixes *f* :: - ⇒ - :: *group-add*
shows *f has-compact-support-on X* ⟹ *g has-compact-support-on X*
⟹ (*λx. f x* − *g x*) *has-compact-support-on X*
unfolding *diff-conv-add-uminus* **by**(*intro has-compact-support-on-add*) *auto*

lemma *has-compact-support-on-max*[*simp*,*intro*!]:
assumes *f has-compact-support-on X g has-compact-support-on X*
shows (*λx. max* (*f x*) (*g x*)) *has-compact-support-on X*
proof −
have *support-on* (*topspace X*) (*λx. max* (*f x*) (*g x*))
⊆ *support-on* (*topspace X*) *f* ∪ *support-on* (*topspace X*) *g*
by (*simp add*: *in-support-on max-def-raw unfold-simps*(*2*))
moreover have *compactin X* (*X closure-of* ...)
using *assms* **by**(*simp add*: *has-compact-support-on-def compactin-Un*)
ultimately show *?thesis*
unfolding *has-compact-support-on-def* **by** (*meson closed-compactin closedin-closure-of closure-of-mono*)
qed

lemma *has-compact-support-on-ext-iff* [*iff*]:
(*λx*∈*topspace X. f x*) *has-compact-support-on X* ⟷ *f has-compact-support-on X*
by(*auto intro*!: *arg-cong2*[**where** *f*=*compactin*] *arg-cong2*[**where** *f*=(*closure-of*)]
simp: *has-compact-support-on-def in-support-on*)

lemma *has-compact-support-on-of-real-iff* [*iff*]:

(λx. *of-real* (f x)) *has-compact-support-on* X = f *has-compact-support-on* X
by(*auto simp*: *has-compact-support-on-iff*)

lemma *has-compact-support-on-complex-iff*:
f *has-compact-support-on* X ⟷
(λx. *Re* (f x)) *has-compact-support-on* X ∧ (λx. *Im* (f x)) *has-compact-support-on* X
proof *safe*
assume h:(λx. *Re* (f x)) *has-compact-support-on* X (λx. *Im* (f x)) *has-compact-support-on* X
have *support-on* (*topspace* X) f ⊆ *support-on* (*topspace* X) (λx. *Re* (f x)) ∪ *support-on* (*topspace* X) (λx. *Im* (f x))
using *complex.expand* **by**(*auto simp*: *in-support-on*)
hence X *closure-of support-on* (*topspace* X) f
⊆ X *closure-of support-on* (*topspace* X) (λx. *Re* (f x)) ∪ X *closure-of support-on* (*topspace* X) (λx. *Im* (f x))
by (*metis* (*no-types*, *lifting*) *closure-of-Un sup.absorb-iff2*)
thus f *has-compact-support-on* X
using h **unfolding** *has-compact-support-on-def*
by (*meson closed-compactin closedin-closure-of compactin-Un*)
next
assume h:f *has-compact-support-on* X
have *support-on* (*topspace* X) (λx. *Re* (f x)) ⊆ *support-on* (*topspace* X) f
support-on (*topspace* X) (λx. *Im* (f x)) ⊆ *support-on* (*topspace* X) f
by(*auto simp*: *in-support-on*)
thus (λx. *Re* (f x)) *has-compact-support-on* X (λx. *Im* (f x)) *has-compact-support-on* X
using h **by**(*auto simp*: *closed-compactin closure-of-mono has-compact-support-on-def*)
qed

lemma [*simp*]:
assumes f *has-compact-support-on* X
shows *has-compact-support-on-Re*:(λx. *Re* (f x)) *has-compact-support-on* X
and *has-compact-support-on-Im*:(λx. *Im* (f x)) *has-compact-support-on* X
using *assms* **by**(*auto simp*: *has-compact-support-on-complex-iff*)

3.3 Positive Linear Functionsls

definition *positive-linear-functional-on-CX* :: 'a *topology* ⇒ (('a ⇒ 'b :: {*ring*, *order*, *topological-space*}) ⇒ 'b) ⇒ *bool*
where *positive-linear-functional-on-CX* X φ ≡
(∀f. *continuous-map* X *euclidean* f ⟶ f *has-compact-support-on* X
⟶ (∀x∈*topspace* X. f x ≥ 0) ⟶ φ (λx∈*topspace* X. f x) ≥ 0) ∧
(∀f a. *continuous-map* X *euclidean* f ⟶ f *has-compact-support-on* X
⟶ φ (λx∈*topspace* X. a ∗ f x) = a ∗ φ (λx∈*topspace* X. f x)) ∧
(∀f g. *continuous-map* X *euclidean* f ⟶ f *has-compact-support-on* X
⟶ *continuous-map* X *euclidean* g ⟶ g *has-compact-support-on* X
⟶ φ (λx∈*topspace* X. f x + g x) = φ (λx∈*topspace* X. f x) + φ (λx∈*topspace* X. g x))

lemma
assumes *positive-linear-functional-on-CX X φ*
shows *pos-lin-functional-on-CX-pos*:
$\bigwedge f$. *continuous-map X euclidean f* $\Longrightarrow$ *f has-compact-support-on X*
$\Longrightarrow$ ($\bigwedge x$. *x*∈*topspace X* $\Longrightarrow$ *f x* ≥ *0*) $\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *f x*) ≥ *0*
and *pos-lin-functional-on-CX-lin*:
$\bigwedge f$ *a*. *continuous-map X euclidean f* $\Longrightarrow$ *f has-compact-support-on X*
$\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *a* ∗ *f x*) = *a* ∗ *φ* (λ*x*∈*topspace X*. *f x*)
$\bigwedge f$ *g*. *continuous-map X euclidean f* $\Longrightarrow$ *f has-compact-support-on X*
$\Longrightarrow$ *continuous-map X euclidean g* $\Longrightarrow$ *g has-compact-support-on X*
$\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *f x* + *g x*) = *φ* (λ*x*∈*topspace X*. *f x*) + *φ* (λ*x*∈*topspace X*. *g x*)
using *assms* **by**(*auto simp*: *positive-linear-functional-on-CX-def*)

lemma *pos-lin-functional-on-CX-pos-complex*:
assumes *positive-linear-functional-on-CX X φ*
shows *continuous-map X euclidean f* $\Longrightarrow$ *f has-compact-support-on X*
$\Longrightarrow$ ($\bigwedge x$. *x*∈*topspace X* $\Longrightarrow$ *Re* (*f x*) ≥ *0*) $\Longrightarrow$ ($\bigwedge x$. *x* ∈ *topspace X* $\Longrightarrow$ *f x* ∈ ℝ)
$\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *f x*) ≥ *0*
by(*intro pos-lin-functional-on-CX-pos*[*OF assms*]) (*simp-all add*: *complex-is-Real-iff less-eq-complex-def*)

lemma *positive-linear-functional-on-CX-compact*:
assumes *compact-space X*
shows *positive-linear-functional-on-CX X φ* ⟷
(∀ *f*. *continuous-map X euclidean f* ⟶ (∀ *x*∈*topspace X*. *f x* ≥ *0*) ⟶ *φ* (λ*x*∈*topspace X*. *f x*) ≥ *0*) ∧
(∀ *f a*. *continuous-map X euclidean f* ⟶ *φ* (λ*x*∈*topspace X*. *a* ∗ *f x*) = *a* ∗ *φ* (λ*x*∈*topspace X*. *f x*)) ∧
(∀ *f g*. *continuous-map X euclidean f* ⟶ *continuous-map X euclidean g*
⟶ *φ* (λ*x*∈*topspace X*. *f x* + *g x*) = *φ* (λ*x*∈*topspace X*. *f x*) + *φ* (λ*x*∈*topspace X*. *g x*))
by(*auto simp*: *positive-linear-functional-on-CX-def assms*)

lemma
assumes *positive-linear-functional-on-CX X φ compact-space X*
shows *pos-lin-functional-on-CX-compact-pos*:
$\bigwedge f$. *continuous-map X euclidean f*
$\Longrightarrow$ ($\bigwedge x$. *x*∈*topspace X* $\Longrightarrow$ *f x* ≥ *0*) $\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *f x*) ≥ *0*
and *pos-lin-functional-on-CX-compact-lin*:
$\bigwedge f$ *a*. *continuous-map X euclidean f*
$\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *a* ∗ *f x*) = *a* ∗ *φ* (λ*x*∈*topspace X*. *f x*)
$\bigwedge f$ *g*. *continuous-map X euclidean f* $\Longrightarrow$ *continuous-map X euclidean g*
$\Longrightarrow$ *φ* (λ*x*∈*topspace X*. *f x* + *g x*) = *φ* (λ*x*∈*topspace X*. *f x*) + *φ* (λ*x*∈*topspace X*. *g x*)
using *assms*(*1*) **by**(*auto simp*: *positive-linear-functional-on-CX-compact assms*(*2*))

lemma *pos-lin-functional-on-CX-diff*:
fixes f :: - ⇒ - :: {*real-normed-vector*, *ring-1*}
assumes *positive-linear-functional-on-CX X* φ
and *cont*:*continuous-map X euclidean f continuous-map X euclidean g*
and *csupp*: *f has-compact-support-on X g has-compact-support-on X*
shows φ ($\lambda x \in$*topspace X*. $f\,x - g\,x$) = φ ($\lambda x \in$*topspace X*. $f\,x$) − φ ($\lambda x \in$*topspace X*. $g\,x$)
using *pos-lin-functional-on-CX-lin*(*2*)[*OF assms*(*1*),*of f* $\lambda x.\ -\ g\ x$] *cont csupp pos-lin-functional-on-CX-lin*(*1*)[*OF assms*(*1*) *cont*(*2*) *csupp*(*2*),*of* − *1*] **by** *simp*

lemma *pos-lin-functional-on-CX-compact-diff*:
fixes f :: - ⇒ - :: {*real-normed-vector*, *ring-1*}
assumes *positive-linear-functional-on-CX X* φ *compact-space X*
and *continuous-map X euclidean f continuous-map X euclidean g*
shows φ ($\lambda x \in$*topspace X*. $f\,x - g\,x$) = φ ($\lambda x \in$*topspace X*. $f\,x$) − φ ($\lambda x \in$*topspace X*. $g\,x$)
using *assms*(*2*) **by**(*auto intro*!: *pos-lin-functional-on-CX-diff assms*)

lemma *pos-lin-functional-on-CX-mono*:
fixes f :: - ⇒ - :: {*real-normed-vector*, *ring-1*, *ordered-ab-group-add*}
assumes *positive-linear-functional-on-CX X* φ
and *mono*:$\bigwedge x.\ x \in$ *topspace X* $\Longrightarrow f\,x \le g\,x$
and *cont*:*continuous-map X euclidean f continuous-map X euclidean g*
and *csupp*: *f has-compact-support-on X g has-compact-support-on X*
shows φ ($\lambda x \in$*topspace X*. $f\,x$) ≤ φ ($\lambda x \in$*topspace X*. $g\,x$)
proof −
have φ ($\lambda x \in$*topspace X*. $f\,x$) ≤ φ ($\lambda x \in$*topspace X*. $f\,x$) + φ ($\lambda x \in$*topspace X*. $g\,x - f\,x$)
by(*auto intro*!: *pos-lin-functional-on-CX-pos*[*OF assms*(*1*)] *assms continuous-map-diff*)
also have ... = φ ($\lambda x \in$*topspace X*. $f\,x + (g\,x - f\,x)$)
by(*intro pos-lin-functional-on-CX-lin*(*2*)[*symmetric*]) (*auto intro*!: *assms continuous-map-diff*)
also have ... = φ ($\lambda x \in$*topspace X*. $g\,x$)
by *simp*
finally show *?thesis* **.**
qed

lemma *pos-lin-functional-on-CX-compact-mono*:
fixes f :: - ⇒ - :: {*real-normed-vector*, *ring-1*, *ordered-ab-group-add*}
assumes *positive-linear-functional-on-CX X* φ *compact-space X*
and $\bigwedge x.\ x \in$ *topspace X* $\Longrightarrow f\,x \le g\,x$
and *continuous-map X euclidean f continuous-map X euclidean g*
shows φ ($\lambda x \in$*topspace X*. $f\,x$) ≤ φ ($\lambda x \in$*topspace X*. $g\,x$)
using *assms*(*2*) **by**(*auto intro*!: *assms pos-lin-functional-on-CX-mono*)

lemma *pos-lin-functional-on-CX-zero*:
assumes *positive-linear-functional-on-CX X* φ
shows φ ($\lambda x \in$*topspace X*. *0*) = *0*
proof −

have φ ($\lambda x{\in}$*topspace* X. *0*) = φ ($\lambda x{\in}$*topspace* X. *0* $*$ *0*)
  by *simp*
also have ... = *0* $*$ φ ($\lambda x{\in}$*topspace* X. *0*)
  by(*intro pos-lin-functional-on-CX-lin*) (*auto simp*: *assms*)
finally show *?thesis*
  by *simp*
qed

lemma *pos-lin-functional-on-CX-uminus*:
  fixes *f* :: - ⇒ - :: {*real-normed-vector*, *ring-1*}
  assumes *positive-linear-functional-on-CX X* φ
    and *continuous-map X euclidean f*
    and *csupp*: *f has-compact-support-on X*
  shows φ ($\lambda x{\in}$*topspace* X. $-$ *f x*) = $-$ φ ($\lambda x{\in}$*topspace* X. *f x*)
  using *pos-lin-functional-on-CX-diff*[*of X* φ λx. *0 f*]
  by(*auto simp*: *assms pos-lin-functional-on-CX-zero*)

lemma *pos-lin-functional-on-CX-compact-uminus*:
  fixes *f* :: - ⇒ - :: {*real-normed-vector*, *ring-1*}
  assumes *positive-linear-functional-on-CX X* φ *compact-space X*
    and *continuous-map X euclidean f*
  shows φ ($\lambda x{\in}$*topspace* X. $-$ *f x*) = $-$ φ ($\lambda x{\in}$*topspace* X. *f x*)
  using *pos-lin-functional-on-CX-diff*[*of X* φ λx. *0 f*]
  by(*auto simp*: *assms pos-lin-functional-on-CX-zero*)

lemma *pos-lin-functional-on-CX-sum*:
  fixes *f* :: - ⇒ - ⇒ - :: {*real-normed-vector*}
  assumes *positive-linear-functional-on-CX X* φ
    and *finite I* $\bigwedge i$. $i \in I \Longrightarrow$ *continuous-map X euclidean* (*f i*)
    and $\bigwedge i$. $i \in I \Longrightarrow$ *f i has-compact-support-on X*
  shows φ ($\lambda x{\in}$*topspace* X. ($\sum i{\in}I$. *f i x*)) = ($\sum i{\in}I$. φ ($\lambda x{\in}$*topspace* X. *f i x*))
  using *assms*(*2*,*3*,*4*)
proof *induction*
  case *empty*
  show *?case*
    using *pos-lin-functional-on-CX-zero*[*OF assms*(*1*)] **by**(*simp add*: *restrict-def*)
next
  case *ih*:(*insert a F*)
  show *?case* (**is** *?lhs* = *?rhs*)
  proof $-$
    have *?lhs* = φ ($\lambda x{\in}$*topspace* X. *f a x* + ($\sum i{\in}F$. *f i x*))
      by(*simp add*: *sum.insert-if*[*OF ih*(*1*)] *ih*(*2*) *restrict-def*)
    also have ... = φ ($\lambda x{\in}$*topspace* X. *f a x*) + φ ($\lambda x{\in}$*topspace* X. ($\sum i{\in}F$. *f i x*))
      by (*auto intro*!: *pos-lin-functional-on-CX-lin*[*OF assms*(*1*)]
              has-compact-support-on-sum ih continuous-map-sum)
    also have ... = *?rhs*
      by(*simp add*: *ih*) (*simp add*: *restrict-def*)
    finally show *?thesis* **.**
  qed

qed

lemma *pos-lin-functional-on-CX-pos-is-real*:
fixes *f* :: - ⇒ *complex*
assumes *positive-linear-functional-on-CX X φ*
and *continuous-map X euclidean f f has-compact-support-on X*
and ⋀*x*. *x* ∈ *topspace X* ⟹ *f x* ∈ ℝ
shows *φ* (*λx*∈*topspace X*. *f x*) ∈ ℝ
proof −
have *φ* (*λx*∈*topspace X*. *f x*) = *φ* (*λx*∈*topspace X*. *complex-of-real* (*Re* (*f x*)))
by (*metis* (*no-types*, *lifting*) *assms*(*4*) *of-real-Re restrict-ext*)
also have ... = *φ* (*λx*∈*topspace X*. *complex-of-real* (*max 0* (*Re* (*f x*))) − *complex-of-real* (*max 0* (− *Re* (*f x*))))
by (*metis* (*no-types*, *opaque-lifting*) *diff-0 diff-0-right equation-minus-iff max.absorb-iff2 max.order-iff neg-0-le-iff-le nle-le of-real-diff*)
also have ... = *φ* (*λx*∈*topspace X*. *complex-of-real* (*max 0* (*Re* (*f x*)))) − *φ* (*λx*∈*topspace X*. *complex-of-real* (*max 0* (− *Re* (*f x*))))
using *assms* **by**(*auto intro*!: *pos-lin-functional-on-CX-diff continuous-map-real-max*)
also have ... ∈ ℝ
using *assms* **by**(*intro Reals-diff*)
(*auto intro*!: *nonnegative-complex-is-real pos-lin-functional-on-CX-pos*[*OF assms*(*1*)] *continuous-map-real-max*
simp: *less-eq-complex-def*)
finally show *?thesis* **.**
qed

lemma
fixes *φ X*
defines *φ*′ ≡ (*λf*. *Re* (*φ* (*λx*∈*topspace X*. *complex-of-real* (*f x*))))
assumes *plf*:*positive-linear-functional-on-CX X φ*
shows *pos-lin-functional-on-CX-complex-decompose*:
⋀*f*. *continuous-map X euclidean f* ⟹ *f has-compact-support-on X*
⟹ *φ* (*λx*∈*topspace X*. *f x*)
= *complex-of-real* (*φ*′ (*λx*∈*topspace X*. *Re* (*f x*))) + i ∗ *complex-of-real* (*φ*′ (*λx*∈*topspace X*. *Im* (*f x*)))
and *pos-lin-functional-on-CX-complex-decompose-plf*:
positive-linear-functional-on-CX X φ′
proof −
fix *f* :: - ⇒ *complex*
assume *f*:*continuous-map X euclidean f f has-compact-support-on X*
show *φ* (*λx*∈*topspace X*. *f x*)
= *complex-of-real* (*φ*′ (*λx*∈*topspace X*. *Re* (*f x*))) + i ∗ *complex-of-real* (*φ*′ (*λx*∈*topspace X*. *Im* (*f x*)))
(**is** *?lhs* = *?rhs*)
proof −
have *φ* (*λx*∈*topspace X*. *f x*) = *φ* (*λx*∈*topspace X*. *Re* (*f x*) + i ∗ *Im* (*f x*))
using *complex-eq* **by** *presburger*
also have ... = *φ* (*λx*∈*topspace X*. *complex-of-real* (*Re* (*f x*))) + *φ* (*λx*∈*topspace X*. i ∗ *complex-of-real* (*Im* (*f x*)))

using *f* **by**(*auto intro*!: *pos-lin-functional-on-CX-lin*[*OF plf*] *has-compact-support-on-mult-left continuous-map-complex-mult-left*)
also have ... = φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Re* (*f x*))) + i ∗ φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Im* (*f x*)))
using *f* **by**(*auto intro*!: *pos-lin-functional-on-CX-lin*[*OF plf*])
also have ... = *complex-of-real* (φ' ($\lambda x\in$*topspace X*. (*Re* (*f x*)))) + i ∗ *complex-of-real* (φ' ($\lambda x\in$*topspace X*. *Im* (*f x*)))
proof −
have [*simp*]: *complex-of-real* (φ' ($\lambda x\in$*topspace X*. *Re* (*f x*))) = φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Re* (*f x*)))
(**is** *?l* = *?r*)
proof −
have *?l* = *complex-of-real* (*Re* (φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Re* (*f x*)))))
by (*metis* (*mono-tags*, *lifting*) φ'*-def restrict-apply'* *restrict-ext*)
also have ... = *?r*
by(*intro of-real-Re pos-lin-functional-on-CX-pos-is-real*[*OF plf*]) (*use f* **in** *auto*)
finally show *?thesis* **.**
qed
have [*simp*]: *complex-of-real* (φ' ($\lambda x\in$*topspace X*. *Im* (*f x*))) = φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Im* (*f x*)))
(**is** *?l* = *?r*)
proof −
have *?l* = *complex-of-real* (*Re* (φ ($\lambda x\in$*topspace X*. *complex-of-real* (*Im* (*f x*)))))
by (*metis* (*mono-tags*, *lifting*) φ'*-def restrict-apply'* *restrict-ext*)
also have ... = *?r*
by(*intro of-real-Re pos-lin-functional-on-CX-pos-is-real*[*OF plf*]) (*use f* **in** *auto*)
finally show *?thesis* **.**
qed
show *?thesis* **by** *simp*
qed
finally show *?thesis* **.**
qed
next
show *positive-linear-functional-on-CX X* φ'
unfolding *positive-linear-functional-on-CX-def*
proof *safe*
fix *f*
assume *f*:*continuous-map X euclideanreal f f has-compact-support-on X* ∀ *x*∈*topspace X*. *0* ≤ *f x*
show φ' ($\lambda x\in$*topspace X*. *f x*) ≥ *0*
proof −
have *0* ≤ φ ($\lambda x\in$*topspace X*. *complex-of-real* (*f x*))
using *f* **by**(*intro pos-lin-functional-on-CX-pos*[*OF plf*]) (*simp-all add*: *less-eq-complex-def*)
hence *0* ≤ *Re* (φ ($\lambda x\in$*topspace X*. *complex-of-real* (*f x*)))

by (*simp add*: *less-eq-complex-def*)
also have ... = φ′ (λx∈*topspace* X. f x)
by (*metis* (*mono-tags*, *lifting*) φ′-*def restrict-apply*′ *restrict-ext*)
finally show *?thesis* **.**
qed
next
fix *a f*
assume *f*:*continuous-map X euclideanreal f f has-compact-support-on X*
show φ′ (λx∈*topspace* X. a ∗ f x) = a ∗ φ′ (λx∈*topspace* X. f x)
proof −
have ∗:φ (λx∈*topspace* X. *complex-of-real* a ∗ *complex-of-real* (f x)) = *complex-of-real* a ∗ φ (λx∈*topspace* X. *complex-of-real* (f x))
using *f* **by**(*auto intro*!: *pos-lin-functional-on-CX-lin*[*OF plf*])
have φ′ (λx∈*topspace* X. a ∗ f x) = *Re* (φ (λx∈*topspace* X. *complex-of-real* a ∗ *complex-of-real* (f x)))
by (*metis* (*mono-tags*, *lifting*) φ′-*def of-real-mult restrict-apply*′ *restrict-ext*)
also have ... = a ∗ (*Re* (φ (λx∈*topspace* X. *complex-of-real* (f x))))
unfolding ∗ **by** *simp*
also have ... = a ∗ φ′ (λx∈*topspace* X. f x)
by (*metis* (*mono-tags*, *lifting*) φ′-*def restrict-apply*′ *restrict-ext*)
finally show *?thesis* **.**
qed
next
fix *f g*
assume *fg*:*continuous-map X euclideanreal f f has-compact-support-on X*
continuous-map X euclideanreal g g has-compact-support-on X
show φ′ (λx∈*topspace* X. f x + g x) = φ′ (λx∈*topspace* X. f x) + φ′ (λx∈*topspace* X. g x)
proof −
have ∗: φ (λx∈*topspace* X. *complex-of-real* (f x) + *complex-of-real* (g x)) = φ (λx∈*topspace* X. *complex-of-real* (f x)) + φ (λx∈*topspace* X. *complex-of-real* (g x))
using *fg* **by**(*auto intro*!: *pos-lin-functional-on-CX-lin*[*OF plf*])
have φ′ (λx∈*topspace* X. f x + g x) = *Re* (φ (λx∈*topspace* X. *complex-of-real* (f x + g x)))
by (*metis* (*mono-tags*, *lifting*) φ′-*def restrict-apply*′ *restrict-ext*)
also have ... = *Re* (φ (λx∈*topspace* X. *complex-of-real* (f x)) + φ (λx∈*topspace* X. *complex-of-real* (g x)))
unfolding *of-real-add* ∗ **by** *simp*
also have ... = *Re* (φ (λx∈*topspace* X. *complex-of-real* (f x))) + *Re* (φ (λx∈*topspace* X. *complex-of-real* (g x)))
by *simp*
also have ... = φ′ (λx∈*topspace* X. f x) + φ′ (λx∈*topspace* X. g x)
by (*metis* (*mono-tags*, *lifting*) φ′-*def restrict-apply*′ *restrict-ext*)
finally show *?thesis* **.**
qed
qed
qed

3.4 Lemmas for Uniqueness

lemma *rep-measures-real-unique*:
assumes *locally-compact-space X Hausdorff-space X*
assumes *N*: *subalgebra N* (*borel-of X*)
$\bigwedge f$. *continuous-map X euclideanreal f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ *integrable N f*
$\bigwedge A$. *A*$\in$*sets N* $\Longrightarrow$ *emeasure N A* = ($\bigsqcap C\in\{C$. *openin X C* $\wedge$ $A \subseteq C\}$. *emeasure N C*)
$\bigwedge A$. *openin X A* $\Longrightarrow$ *emeasure N A* = ($\bigsqcup K\in\{K$. *compactin X K* $\wedge$ $K \subseteq A\}$. *emeasure N K*)
$\bigwedge A$. *A*$\in$*sets N* $\Longrightarrow$ *emeasure N A* $< \infty \Longrightarrow$ *emeasure N A* = ($\bigsqcup K\in\{K$. *compactin X K* $\wedge$ $K \subseteq A\}$. *emeasure N K*)
$\bigwedge K$. *compactin X K* $\Longrightarrow$ *N K* $< \infty$
assumes *M*: *subalgebra M* (*borel-of X*)
$\bigwedge f$. *continuous-map X euclideanreal f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ *integrable M f*
$\bigwedge A$. *A*$\in$*sets M* $\Longrightarrow$ *emeasure M A* = ($\bigsqcap C\in\{C$. *openin X C* $\wedge$ $A \subseteq C\}$. *emeasure M C*)
$\bigwedge A$. *openin X A* $\Longrightarrow$ *emeasure M A* = ($\bigsqcup K\in\{K$. *compactin X K* $\wedge$ $K \subseteq A\}$. *emeasure M K*)
$\bigwedge A$. *A*$\in$*sets M* $\Longrightarrow$ *emeasure M A* $< \infty \Longrightarrow$ *emeasure M A* = ($\bigsqcup K\in\{K$. *compactin X K* $\wedge$ $K \subseteq A\}$. *emeasure M K*)
$\bigwedge K$. *compactin X K* $\Longrightarrow$ *M K* $< \infty$
and *sets-eq*: *sets N* = *sets M*
and *integ-eq*: $\bigwedge f$. *continuous-map X euclideanreal f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ ($\int x.\ f\ x\ \partial N$) = ($\int x.\ f\ x\ \partial M$)
shows *N* = *M*
proof(*intro measure-eqI sets-eq*)
have *space-N*: *space N* = *topspace X* **and** *space-M*: *space M* = *topspace X*
using *N*(*1*) *M*(*1*) **by**(*auto simp*: *subalgebra-def space-borel-of*)
have *N K* = *M K* **if** *K*:*compactin X K* **for** *K*
proof −
have *kc*: *kc-space X*
using *Hausdorff-imp-kc-space assms*(*2*) **by** *blast*
have *K-sets*[*measurable*]: *K* $\in$ *sets N K* $\in$ *sets M*
using *N*(*1*) *M*(*1*) *compactin-imp-closedin-gen*[*OF kc K*]
by(*auto simp*: *borel-of-closed subalgebra-def*)
show *?thesis*
proof(*rule antisym*[*OF ennreal-le-epsilon ennreal-le-epsilon*])
fix *e* :: *real*
assume *e*: *e* > *0*
show *emeasure N K* $\le$ *emeasure M K* + *ennreal e*
proof −
have *emeasure M K* $\ge$ $\bigsqcap$ (*emeasure M* ‘ $\{C$. *openin X C* $\wedge$ $K \subseteq C\}$)
by(*simp add*: *M*(*3*)[*OF K-sets*(*2*)])
from *Inf-le-iff*[*THEN iffD1*,*OF this*,*rule-format*,*of emeasure M K* + *e*]
obtain *U* **where** *U*:*openin X U K* $\subseteq$ *U emeasure M U* < *emeasure M K* + *ennreal e*
using *K M*(*6*) *e* **by** *fastforce*

then have [*measurable*]: $U \in$ *sets M*
using *M*(*1*) **by**(*auto simp*: *subalgebra-def borel-of-open*)
then obtain *f* **where** *f*:*continuous-map X* (*top-of-set* {*0*..*1*::*real*}) *f*
f ' (*topspace X* − *U*) ⊆ {*0*} *f* ' *K* ⊆ {*1*}
disjnt (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*}) (*topspace X* − *U*)
compactin X (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*})
using *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF assms*(*1*) *disjI1*[*OF assms*(*2*)],*of 0 1 topspace X* − *U K*] *U K*
by(*simp add*: *closedin-def disjnt-iff*) *blast*
have *f-int*: *integrable N f integrable M f*
using *f* **by**(*auto intro*!: *N M simp*: *continuous-map-in-subtopology has-compact-support-on-iff*)
have *f-01*: *x* ∈ *topspace X* ⟹ *0* ≤ *f x x* ∈ *topspace X* ⟹ *f x* ≤ *1* **for** *x*
using *continuous-map-image-subset-topspace*[*OF f*(*1*)] **by** *auto*
have *emeasure N K* = (∫⁺*x*. *indicator K x* ∂*N*)
by *simp*
also have ... ≤ (∫⁺*x*. *f x* ∂*N*)
using *f*(*3*) **by**(*intro nn-integral-mono*) (*auto simp*: *indicator-def*)
also have ... = *ennreal* (∫ *x*. *f x* ∂*N*)
by(*rule nn-integral-eq-integral*) (*use f-int continuous-map-image-subset-topspace*[*OF f*(*1*)] *f-01 space-N* **in** *auto*)
also have ... = *ennreal* (∫ *x*. *f x* ∂*M*)
using *f* **by**(*auto intro*!: *ennreal-cong integ-eq simp*: *continuous-map-in-subtopology has-compact-support-on-iff*)
also have ... = (∫⁺*x*. *f x* ∂*M*)
by(*rule nn-integral-eq-integral*[*symmetric*])
(*use f-int continuous-map-image-subset-topspace*[*OF f*(*1*)] *f-01 space-M* **in** *auto*)
also have ... ≤ (∫⁺*x*. *indicator U x* ∂*M*)
using *f*(*2*) *f-01* **by**(*intro nn-integral-mono*) (*auto simp*: *indicator-def space-M*)
also have ... = *emeasure M U*
by *simp*
also have ... < *emeasure M K* + *ennreal e*
by *fact*
finally show *?thesis*
by *simp*
qed
next
fix *e* :: *real*
assume *e*: *e* > *0*
show *emeasure M K* ≤ *emeasure N K* + *ennreal e*
proof −
have *emeasure N K* ≥ ⨅ (*emeasure N* ' {*C*. *openin X C* ∧ *K* ⊆ *C*})
by(*simp add*: *N*(*3*)[*OF K-sets*(*1*)])
from *Inf-le-iff*[*THEN iffD1*,*OF this*,*rule-format*,*of emeasure N K* + *e*]
obtain *U* **where** *U*:*openin X U K* ⊆ *U emeasure N U* < *emeasure N K* + *ennreal e*
using *K N*(*6*) *e* **by** *fastforce*
then have [*measurable*]: *U* ∈ *sets N*

using $N(1)$ **by**(*auto simp*: *subalgebra-def borel-of-open*)
then obtain f **where** f:*continuous-map* X (*top-of-set* $\{0..1::real\}$) f
f ‘ (*topspace* $X - U$) $\subseteq \{0\}$ f ‘ $K \subseteq \{1\}$
disjnt (X *closure-of* $\{x \in$ *topspace* $X.\ f\ x \neq 0\}$) (*topspace* $X - U$)
compactin X (X *closure-of* $\{x \in$ *topspace* $X.\ f\ x \neq 0\}$)
using *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF assms*(*1*) *disjI1*[*OF assms*(*2*)],*of 0 1 topspace* $X - U\ K$] $U\ K$
by(*simp add*: *closedin-def disjnt-iff*) *blast*
have *f-int*: *integrable N f integrable M f*
using f **by**(*auto intro*!: *N M simp*: *continuous-map-in-subtopology has-compact-support-on-iff*)
have *f-01*: $x \in$ *topspace* $X \Longrightarrow 0 \leq f\ x\ x \in$ *topspace* $X \Longrightarrow f\ x \leq 1$ **for** x
using *continuous-map-image-subset-topspace*[*OF* $f(1)$] **by** *auto*
have *emeasure* $M\ K = (\int^+ x.$ *indicator* $K\ x\ \partial M)$
by *simp*
also have ... $\leq (\int^+ x.\ f\ x\ \partial M)$
using $f(3)$ **by**(*intro nn-integral-mono*) (*auto simp*: *indicator-def*)
also have ... = *ennreal* $(\int x.\ f\ x\ \partial M)$
by(*rule nn-integral-eq-integral*) (*use f-int continuous-map-image-subset-topspace*[*OF* $f(1)$] *f-01 space-M* **in** *auto*)
also have ... = *ennreal* $(\int x.\ f\ x\ \partial N)$
using f **by**(*auto intro*!: *ennreal-cong integ-eq*[*symmetric*] *simp*: *continuous-map-in-subtopology has-compact-support-on-iff*)
also have ... $= (\int^+ x.\ f\ x\ \partial N)$
by(*rule nn-integral-eq-integral*[*symmetric*])
(*use f-int continuous-map-image-subset-topspace*[*OF* $f(1)$] *f-01 space-N* **in** *auto*)
also have ... $\leq (\int^+ x.$ *indicator* $U\ x\ \partial N)$
using $f(2)$ *f-01* **by**(*intro nn-integral-mono*) (*auto simp*: *indicator-def space-N*)
also have ... = *emeasure N U*
by *simp*
also have ... < *emeasure N K* + *ennreal e*
by *fact*
finally show *?thesis*
by *simp*
qed
qed
qed
hence $\bigwedge A.$ *openin X A* $\Longrightarrow$ *emeasure N A* = *emeasure M A*
by(*auto simp*: $N(4)$ $M(4)$)
thus $\bigwedge A.\ A \in$ *sets N* $\Longrightarrow$ *emeasure N A* = *emeasure M A*
using $N(3)$ $M(3)$ **by**(*auto simp*: *sets-eq*)
qed

lemma *rep-measures-complex-unique*:
fixes X :: *'a topology*
assumes *locally-compact-space X Hausdorff-space X*
assumes N: *subalgebra N* (*borel-of X*)
$\bigwedge f.$ *continuous-map X euclidean f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ *com-*

plex-integrable N f
$\bigwedge A.\ A{\in}sets\ N \Longrightarrow emeasure\ N\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ emeasure\ N\ C)$
$\bigwedge A.\ openin\ X\ A \Longrightarrow emeasure\ N\ A = (\bigsqcup K{\in}\{K.\ compactin\ X\ K \wedge K \subseteq A\}.\ emeasure\ N\ K)$
$\bigwedge A.\ A{\in}sets\ N \Longrightarrow emeasure\ N\ A < \infty \Longrightarrow emeasure\ N\ A = (\bigsqcup K{\in}\{K.\ compactin\ X\ K \wedge K \subseteq A\}.\ emeasure\ N\ K)$
$\bigwedge K.\ compactin\ X\ K \Longrightarrow N\ K < \infty$
assumes *M*: *subalgebra M* (*borel-of X*)
$\bigwedge f.\ continuous\text{-}map\ X\ euclidean\ f \Longrightarrow f\ has\text{-}compact\text{-}support\text{-}on\ X \Longrightarrow complex\text{-}integrable\ M\ f$
$\bigwedge A.\ A{\in}sets\ M \Longrightarrow emeasure\ M\ A = (\bigsqcap C{\in}\{C.\ openin\ X\ C \wedge A \subseteq C\}.\ emeasure\ M\ C)$
$\bigwedge A.\ openin\ X\ A \Longrightarrow emeasure\ M\ A = (\bigsqcup K{\in}\{K.\ compactin\ X\ K \wedge K \subseteq A\}.\ emeasure\ M\ K)$
$\bigwedge A.\ A{\in}sets\ M \Longrightarrow emeasure\ M\ A < \infty \Longrightarrow emeasure\ M\ A = (\bigsqcup K{\in}\{K.\ compactin\ X\ K \wedge K \subseteq A\}.\ emeasure\ M\ K)$
$\bigwedge K.\ compactin\ X\ K \Longrightarrow M\ K < \infty$
and *sets-eq*: *sets N* = *sets M*
and *integ-eq*: $\bigwedge f{::}'a \Rightarrow complex.\ continuous\text{-}map\ X\ euclidean\ f \Longrightarrow f\ has\text{-}compact\text{-}support\text{-}on\ X$
$\Longrightarrow (\int x.\ f\ x\ \partial N) = (\int x.\ f\ x\ \partial M)$
shows $N = M$
proof(*rule rep-measures-real-unique*[*OF assms*(*1*,*2*)])
fix *f*
assume *f*:*continuous-map X euclideanreal f f has-compact-support-on X*
show $(\int x.\ f\ x\ \partial N) = (\int x.\ f\ x\ \partial M)$
proof −
have $(\int x.\ f\ x\ \partial N) = Re\ (\int x.\ (complex\text{-}of\text{-}real\ (f\ x))\ \partial N)$
by *simp*
also have ... $= Re\ (\int x.\ (complex\text{-}of\text{-}real\ (f\ x))\ \partial M)$
proof −
have *1*:$(\int x.\ (complex\text{-}of\text{-}real\ (f\ x))\ \partial N) = (\int x.\ (complex\text{-}of\text{-}real\ (f\ x))\ \partial M)$
by(*rule integ-eq*) (*auto intro*!: *f*)
show *?thesis*
unfolding *1* **by** *simp*
qed
finally show *?thesis*
by *simp*
qed
next
fix *f*
assume *continuous-map X euclideanreal f f has-compact-support-on X*
hence *complex-integrable N* ($\lambda x.$ *complex-of-real* (*f x*)) *complex-integrable M* ($\lambda x.$ *complex-of-real* (*f x*))
by (*auto intro*!: *M N*)
thus *integrable N f integrable M f*
using *complex-of-real-integrable-eq* **by** *auto*
qed *fact*+

```
lemma finite-tight-measure-eq:
  assumes locally-compact-space X metrizable-space X tight-on X N tight-on X M
     and integ-eq: ⋀f. continuous-map X euclideanreal f ⟹ f ∈ topspace X →
{0..1} ⟹ (∫ x. f x ∂N) = (∫ x. f x ∂M)
  shows N = M
proof(rule measure-eqI)
  interpret N: finite-measure N
    using assms(3) tight-on-def by blast
  interpret M: finite-measure M
    using assms(4) tight-on-def by blast
  have integ-N: ⋀A. A ∈ sets N ⟹ integrable N (indicat-real A)
   and integ-M: ⋀A. A ∈ sets M ⟹ integrable M (indicat-real A)
    by (auto simp add: N.emeasure-eq-measure M.emeasure-eq-measure)
  have sets-N: sets N = borel-of X and space-N: space N = topspace X
   and sets-M: sets M = borel-of X and space-M: space M = topspace X
    using assms(3,4) sets-eq-imp-space-eq[of - borel-of X]
    by(auto simp: tight-on-def space-borel-of)
  fix A
  assume [measurable]:A ∈ sets N
  then have [measurable]: A ∈ sets M
    using sets-M sets-N by blast
  have measure M A = ⨆ (Sigma-Algebra.measure M ` {K. compactin X K ∧ K
⊆ A})
    by(auto intro!: inner-regular''[OF assms(2,4)])
  also have ... = ⨆ (Sigma-Algebra.measure N ` {K. compactin X K ∧ K ⊆ A})
  proof –
    have measure M K = measure N K if K:compactin X K K ⊆ A for K
    proof –
      have [measurable]: K ∈ sets M K ∈ sets N
       by(auto simp: sets-M sets-N intro!: borel-of-closed compactin-imp-closedin K
metrizable-imp-Hausdorff-space assms)
      show ?thesis
      proof(rule antisym[OF field-le-epsilon field-le-epsilon])
        fix e :: real
        assume e:e > 0
        have ∀ y>measure N K. ∃ a∈measure N ` {C. openin X C ∧ K ⊆ C}. a <
y
       by(intro cInf-le-iff[THEN iffD1] eq-refl[OF N.outer-regularD[OF N.outer-regular'[OF
assms(2) sets-N[symmetric]],symmetric]])
             (auto intro!: bdd-belowI[where m=0] compactin-subset-topspace[OF
K(1)])
        from this[rule-format,of measure N K + e] obtain U where U: openin X
U K ⊆ U measure N U < measure N K + e
          using e by auto
        then have [measurable]: U ∈ sets M U ∈ sets N
          by(auto simp: sets-N sets-M intro!: borel-of-open)
        obtain f where f:continuous-map X (top-of-set {0..1::real}) f
          f ` (topspace X – U) ⊆ {0} f ` K ⊆ {1}
```

disjnt (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*}) (*topspace X* − *U*)
compactin X (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*})
using *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF assms*(*1*) *disjI1*[*OF metrizable-imp-Hausdorff-space*[*OF assms*(*2*)]],*of 0 1 topspace X* − *U K*] *U K*
by(*simp add*: *closedin-def disjnt-iff*) *blast*
hence *f'*: *continuous-map X euclideanreal f*
$\bigwedge$*x*. *x* ∈ *topspace X* ⟹ *f x* ≥ *0* $\bigwedge$*x*. *x* ∈ *topspace X* ⟹ *f x* ≤ *1*
by (*auto simp add*: *continuous-map-in-subtopology*)
have [*measurable*]: *f* ∈ *borel-measurable M f* ∈ *borel-measurable N*
using *continuous-map-measurable*[*OF f'*(*1*)]
by(*auto simp*: *borel-of-euclidean sets-N sets-M cong*: *measurable-cong-sets*)
from *f'*(*2*,*3*) **have** *f-int*[*simp*]: *integrable M f integrable N f*
by(*auto intro*!: *M*.*integrable-const-bound*[**where** *B*=*1*] *N*.*integrable-const-bound*[**where** *B*=*1*] *simp*: *space-N space-M*)
have *measure M K* = (∫ *x*. *indicator K x* ∂*M*)
by *simp*
also have ... ≤ (∫ *x*. *f x* ∂*M*)
using *f*(*3*) *f'*(*2*) **by**(*intro integral-mono integ-M*) (*auto simp*: *space-M indicator-def*)
also have ... = (∫ *x*. *f x* ∂*N*)
by(*auto intro*!: *integ-eq*[*symmetric*] *f'*)
also have ... ≤ (∫ *x*. *indicator U x* ∂*N*)
using *f*(*2*) *f'*(*3*) **by**(*intro integral-mono integ-N*) (*auto simp*: *space-N indicator-def*)
also have ... ≤ *measure N K* + *e*
using *U*(*3*) **by** *fastforce*
finally show *measure M K* ≤ *measure N K* + *e* **.**
next
fix *e* :: *real*
assume *e*:*e* > *0*
have ∀ *y*>*measure M K*. ∃ *a*∈*measure M* ‘ {*C*. *openin X C* ∧ *K* ⊆ *C*}. *a* < *y*
by(*intro cInf-le-iff*[*THEN iffD1*] *eq-refl*[*OF M*.*outer-regularD*[*OF M*.*outer-regular'*[*OF assms*(*2*) *sets-M*[*symmetric*]],*symmetric*]])
(*auto intro*!: *bdd-belowI*[**where** *m*=*0*] *compactin-subset-topspace*[*OF K*(*1*)])
from *this*[*rule-format*,*of measure M K* + *e*] **obtain** *U* **where** *U*: *openin X U K* ⊆ *U measure M U* < *measure M K* + *e*
using *e* **by** *auto*
then have [*measurable*]: *U* ∈ *sets M U* ∈ *sets N*
by(*auto simp*: *sets-N sets-M intro*!: *borel-of-open*)
obtain *f* **where** *f*:*continuous-map X* (*top-of-set* {*0*..*1*::*real*}) *f*
f ‘ (*topspace X* − *U*) ⊆ {*0*} *f* ‘ *K* ⊆ {*1*}
disjnt (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*}) (*topspace X* − *U*)
compactin X (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*})
using *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF assms*(*1*) *disjI1*[*OF metrizable-imp-Hausdorff-space*[*OF assms*(*2*)]],*of 0 1 topspace X* − *U K*] *U K*

by(*simp add*: *closedin-def disjnt-iff*) *blast*
hence *f'*: *continuous-map X euclideanreal f*
$\bigwedge x.\ x \in topspace\ X \Longrightarrow f\ x \geq 0$ $\bigwedge x.\ x \in topspace\ X \Longrightarrow f\ x \leq 1$
by (*auto simp add*: *continuous-map-in-subtopology*)
have [*measurable*]: $f \in$ *borel-measurable M f* $\in$ *borel-measurable N*
using *continuous-map-measurable*[*OF f'*(*1*)]
by(*auto simp*: *borel-of-euclidean sets-N sets-M cong*: *measurable-cong-sets*)
from *f'*(*2*,*3*) have *f-int*[*simp*]: *integrable M f integrable N f*
by(*auto intro*!: *M.integrable-const-bound*[where *B=1*] *N.integrable-const-bound*[where *B=1*] *simp*: *space-N space-M*)
have *measure N K* = ($\int x.$ *indicator K x* ∂N)
by *simp*
also have ... $\leq$ ($\int x.\ f\ x\ \partial N$)
using *f*(*3*) *f'*(*2*) by(*intro integral-mono integ-N*) (*auto simp*: *space-N indicator-def*)
also have ... = ($\int x.\ f\ x\ \partial M$)
by(*auto intro*!: *integ-eq f'*)
also have ... $\leq$ ($\int x.$ *indicator U x* ∂M)
using *f*(*2*) *f'*(*3*) by(*intro integral-mono integ-M*) (*auto simp*: *space-M indicator-def*)
also have ... $\leq$ *measure M K* + *e*
using *U*(*3*) by *fastforce*
finally show *measure N K* $\leq$ *measure M K* + *e* .
qed
qed
thus *?thesis*
by *simp*
qed
also have ... = *measure N A*
by(*auto intro*!: *inner-regular''*[*symmetric*,*OF assms*(*2*,*3*)])
finally show *emeasure N A* = *emeasure M A*
using *M.emeasure-eq-measure N.emeasure-eq-measure* by *presburger*
qed(*insert assms*(*3*,*4*), *auto simp*: *tight-on-def*)

3.5 Riesez Representation Theorem for Real Numbers

theorem *Riesz-representation-real-complete*:
fixes *X* :: *'a topology* and φ :: (*'a* $\Rightarrow$ *real*) $\Rightarrow$ *real*
assumes *lh*:*locally-compact-space X Hausdorff-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
shows $\exists M.\ \exists!N.$ *sets N* = *M* $\wedge$ *subalgebra N* (*borel-of X*)
$\wedge$ ($\forall A \in$ *sets N*. *emeasure N A* = ($\bigsqcap C \in \{C.$ *openin X C* $\wedge A \subseteq C\}$. *emeasure N C*))
$\wedge$ ($\forall A.$ *openin X A* $\longrightarrow$ *emeasure N A* = ($\bigsqcup K \in \{K.$ *compactin X K* $\wedge K \subseteq A\}$. *emeasure N K*))
$\wedge$ ($\forall A \in$ *sets N*. *emeasure N A* $< \infty$
$\longrightarrow$ *emeasure N A* = ($\bigsqcup K \in \{K.$ *compactin X K* $\wedge K \subseteq A\}$. *emeasure N K*))
$\wedge$ ($\forall K.$ *compactin X K* $\longrightarrow$ *emeasure N K* $< \infty$)

$\wedge$ ($\forall f$. *continuous-map X euclideanreal f* $\longrightarrow$ *f has-compact-support-on X* $\longrightarrow$ φ ($\lambda x\in$*topspace X*. *f x*) = ($\int x$. *f x* ∂N))
$\wedge$ ($\forall f$. *continuous-map X euclideanreal f* $\longrightarrow$ *f has-compact-support-on X* $\longrightarrow$ *integrable N f*)
$\wedge$ *complete-measure N*
proof –
let *?iscont* = λf. *continuous-map X euclideanreal f*
let *?csupp* = λf. *f has-compact-support-on X*
let *?fA* = λA *f*. *?iscont f* $\wedge$ *?csupp f* $\wedge$ *X closure-of* {$x\in$*topspace X*. *f x* $\neq$ *0*} $\subseteq$ *A*
$\wedge$ *f* $\in$ *topspace X* $\rightarrow$ {*0..1*} $\wedge$ *f* $\in$ *topspace X* $-$ *A* $\rightarrow$ {*0*}
let *?fK* = λK *f*. *?iscont f* $\wedge$ *?csupp f* $\wedge$ *f* $\in$ *topspace X* $\rightarrow$ {*0..1*} $\wedge$ *f* $\in$ *K* $\rightarrow$ {*1*}

have *ext-sup*[*simp*]:
$\bigwedge$*P Q*. {$x\in$*topspace X*. (*if x* $\in$ *topspace X then P x else Q x*) $\neq$ *0*} = {$x\in$*topspace X*. *P x* $\neq$ *0*}
by *fastforce*
have *times-unfold*[*simp*]: $\bigwedge$*P Q*. {$x\in$*topspace X*. *P x* $\wedge$ *Q x*} = {$x\in$*topspace X*. *P x*} $\cap$ {$x\in$*topspace X*. *Q x*}
by *fastforce*
note *pos* = *pos-lin-functional-on-CX-pos*[*OF plf*]
note *linear* = *pos-lin-functional-on-CX-lin*[*OF plf*]
note *φdiff* = *pos-lin-functional-on-CX-diff*[*OF plf*]
note *φmono* = *pos-lin-functional-on-CX-mono*[*OF plf*]
note *φ-0* = *pos-lin-functional-on-CX-zero*[*OF plf*]

Lemma 2.13 [1].

have *fApartition*: $\exists$ *hi*. ($\forall$ *i*$\leq$*n*. (*?fA* (*Vi i*) (*hi i*))) $\wedge$
($\forall x\in K$. ($\sum$ *i*$\leq$*n*. *hi i x*) = *1*) $\wedge$ ($\forall x\in$*topspace X*. *0* $\leq$ ($\sum$ *i*$\leq$*n*. *hi i x*)) $\wedge$
($\forall x\in$*topspace X*. ($\sum$ *i*$\leq$*n*. *hi i x*) $\leq$ *1*)
if *K*:*compactin X K* $\bigwedge$*i*::*nat*. *i* $\leq$ *n* $\Longrightarrow$ *openin X* (*Vi i*) *K* $\subseteq$ ($\bigcup$ *i*$\leq$*n*. *Vi i*) **for** *K Vi n*
proof –
{
fix *x*
assume *x*:*x* $\in$ *K*
have $\exists$ *i*$\leq$*n*. *x* $\in$ *Vi i* $\wedge$ ($\exists$ *U V*. *openin X U* $\wedge$ (*compactin X V*) $\wedge$ *x* $\in$ *U* $\wedge$ *U* $\subseteq$ *V* $\wedge$ *V* $\subseteq$ *Vi i*)
proof –
obtain *i* **where** *i*: *i* $\leq$ *n x* $\in$ *Vi i*
using *K x* **by** *blast*
thus *?thesis*
using *locally-compact-space-neighbourhood-base*[*of X*] *neighbourhood-base-of*[*of* λU. *compactin X U X*] *lh K*
by(*fastforce intro*!: *exI*[**where** *x*=*i*])
qed
}

hence $\exists$ *ix Ux Vx*. $\forall$ *x*$\in$*K*. *ix x* $\leq$ *n* $\wedge$ *x* $\in$ *Vi* (*ix x*) $\wedge$ *openin X* (*Ux x*) $\wedge$
compactin X (*Vx x*) $\wedge$ *x* $\in$ *Ux x* $\wedge$ *Ux x* $\subseteq$ *Vx x* $\wedge$ *Vx x* $\subseteq$ *Vi* (*ix x*)
by *metis*
then obtain *ix Ux Vx* **where** *xinK*: $\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *ix x* $\leq$ *n* $\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *x* $\in$ *Vi* (*ix x*)
$\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *openin X* (*Ux x*) $\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *compactin X* (*Vx x*) $\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *x* $\in$ *Ux x*
$\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *Ux x* $\subseteq$ *Vx x* $\bigwedge$*x*. *x* $\in$ *K* $\Longrightarrow$ *Vx x* $\subseteq$ *Vi* (*ix x*)
by *blast*
hence *K* $\subseteq$ ($\bigcup$*x*$\in$*K*. *Ux x*)
by *fastforce*
from *compactinD*[*OF K*(*1*) - *this*] *xinK*(*3*) **obtain** *K'* **where** *K'*: *finite K' K'* $\subseteq$ *K K* $\subseteq$ ($\bigcup$*x*$\in$*K'*. *Ux x*)
by (*metis* (*no-types*, *lifting*) *finite-subset-image imageE*)

define *Hi* **where** *Hi* $\equiv$ (λ*i*. $\bigcup$ (*Vx* ‘ {*x*. *x* $\in$ *K'* $\wedge$ *ix x* = *i*}))
have *Hi-Vi*: $\bigwedge$*i*. *i* $\leq$ *n* $\Longrightarrow$ *Hi i* $\subseteq$ *Vi i*
using *xinK K'* **by**(*fastforce simp*: *Hi-def*)
have *K-unHi*: *K* $\subseteq$ ($\bigcup$*i*$\leq$*n*. *Hi i*)
proof
fix *x*
assume *x* $\in$ *K*
then obtain *y* **where** *y*:*y* $\in$ *K' x* $\in$ *Ux y*
using *K'* **by** *auto*
then have *x* $\in$ *Vx y ix y* $\leq$ *n*
using *K' xinK*[*of y*] **by** *auto*
with *y* **show** *x* $\in$ ($\bigcup$*i*$\leq$*n*. *Hi i*)
by(*fastforce simp*: *Hi-def*)
qed
have *compactin-Hi*: $\bigwedge$*i*. *i* $\leq$ *n* $\Longrightarrow$ *compactin X* (*Hi i*)
using *xinK K'* **by**(*auto intro*!: *compactin-Union simp*: *Hi-def*)
{
fix *i*
assume *i* $\in$ {..*n*}
then have *i*: *i* $\leq$ *n* **by** *auto*
have *closedin X* (*topspace X* $-$ *Vi i*) *disjnt* (*topspace X* $-$ *Vi i*) (*Hi i*)
using *Hi-Vi*[*OF i*] *K*(*2*)[*OF i*] **by** (*auto simp*: *disjnt-def*)
from *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*of - 0 1*,*OF lh*(*1*) *disjI1*[*OF lh*(*2*)] - *this*(*1*) *compactin-Hi*[*OF i*] *this*(*2*)]
have $\exists$ *hi*. *continuous-map X* (*top-of-set* {*0..1*::*real*}) *hi* $\wedge$ *hi* ‘ (*topspace X* $-$ *Vi i*) $\subseteq$ {*0*} $\wedge$
hi ‘ *Hi i* $\subseteq$ {*1*} $\wedge$ *disjnt* (*X closure-of* {*x*$\in$*topspace X*. *hi x* $\neq$ *0*}) (*topspace X* $-$ *Vi i*) $\wedge$
?csupp hi
unfolding *has-compact-support-on-iff* **by** *fastforce*
hence $\exists$ *hi*. *?iscont hi* $\wedge$ *hi* ‘ *topspace X* $\subseteq$ {*0..1*} $\wedge$ *hi* ‘ (*topspace X* $-$ *Vi i*) $\subseteq$ {*0*} $\wedge$

hi ‘ Hi i ⊆ {*1*} ∧ *disjnt* (*X closure-of* {*x*∈*topspace X*. *hi x* ≠ *0*}) (*topspace X* − *Vi i*) ∧
?csupp hi
by (*simp add*: *continuous-map-in-subtopology disjnt-def has-compact-support-on-def*)
}
hence ∃ *hi*. ∀ *i*∈{..*n*}. *?iscont* (*hi i*) ∧ *hi i ‘ topspace X* ⊆ {*0*..*1*} ∧
hi i ‘ (*topspace X* − *Vi i*) ⊆ {*0*} ∧ *hi i ‘ Hi i* ⊆ {*1*} ∧
disjnt (*X closure-of* {*x*∈*topspace X*. *hi i x* ≠ *0*}) (*topspace X* − *Vi i*) ∧ *?csupp* (*hi i*)
by(*intro bchoice*) *auto*
hence ∃ *hi*. ∀ *i*≤*n*. *?iscont* (*hi i*) ∧ *hi i ‘ topspace X* ⊆ {*0*..*1*} ∧ *hi i ‘* (*topspace X* − *Vi i*) ⊆ {*0*} ∧
hi i ‘ Hi i ⊆ {*1*} ∧ *disjnt* (*X closure-of* {*x*∈*topspace X*. *hi i x* ≠ *0*}) (*topspace X* − *Vi i*) ∧ *?csupp* (*hi i*)
by (*meson atMost-iff*)
then obtain *gi* **where** *gi*: ⋀*i*. *i* ≤ *n* ⟹ *?iscont* (*gi i*)
⋀*i*. *i* ≤ *n* ⟹ *gi i ‘ topspace X* ⊆ {*0*..*1*} ⋀*i*. *i* ≤ *n* ⟹ *gi i ‘* (*topspace X* − *Vi i*) ⊆ {*0*}
⋀*i*. *i* ≤ *n* ⟹ *gi i ‘ Hi i* ⊆ {*1*}
⋀*i*. *i* ≤ *n* ⟹ *disjnt* (*X closure-of* {*x*∈*topspace X*. *gi i x* ≠ *0*}) (*topspace X* − *Vi i*)
⋀*i*. *i* ≤ *n* ⟹ *?csupp* (*gi i*)
by *fast*
define *hi* **where** *hi* ≡ (λ*n*. λ*x*∈*topspace X*. (∏ *i*<*n*. (*1* − *gi i x*)) ∗ *gi n x*)
show *?thesis*
proof(*safe intro*!: *exI*[**where** *x*=*hi*])
fix *i*
assume *i*: *i* ≤ *n*
then show *?iscont* (*hi i*)
using *gi*(*1*) **by**(*auto simp*: *hi-def intro*!: *continuous-map-real-mult continuous-map-prod continuous-map-diff*)
show *?csupp* (*hi i*)
proof −
have {*x* ∈ *topspace X*. *hi i x* ≠ *0*} = {*x*∈*topspace X*. *gi i x* ≠ *0*} ∩ (⋂ *j*<*i*. {*x*∈*topspace X*. *gi j x* ≠ *1*})
using *gi* **by**(*auto simp*: *hi-def*)
also have ... ⊆ {*x*∈*topspace X*. *gi i x* ≠ *0*}
by *blast*
finally show *?thesis*
using *gi*(*6*)[*OF i*] *closure-of-mono closed-compactin*
by(*fastforce simp*: *has-compact-support-on-iff*)
qed
next
fix *i x*
assume *i*: *i* ≤ *n* **and** *x*: *x* ∈ *topspace X*
{
assume *x* ∉ *Vi i*
with *i x gi*(*3*)[*of i*] **show** *hi i x* = *0*
by(*auto simp*: *hi-def*)

}
show *hi i x* ∈ {*0..1*}
using *i x gi*(*2*) **by**(*auto simp*: *hi-def image-subset-iff intro*!: *mult-nonneg-nonneg mult-le-one prod-le-1 prod-nonneg*)
next
fix *x*
have *1*:($\sum i \le n.\ hi\ i\ x$) = *1* − ($\prod i \le n.\ (1 - gi\ i\ x)$) **if** *x*:*x* ∈ *topspace X*
proof −
have ($\sum i \le n.\ hi\ i\ x$) = ($\sum j \le n.\ (\prod i<j.\ (1 - gi\ i\ x)) * gi\ j\ x$)
using *x* **by** (*simp add*: *hi-def*)
also have ... = *1* − ($\prod i \le n.\ (1 - gi\ i\ x)$)
proof −
have [*simp*]: ($\prod i<m.\ 1 - gi\ i\ x$) ∗ (*1* − *gi m x*) = ($\prod i \le m.\ 1 - gi\ i\ x$) **for** *m*
by (*metis lessThan-Suc-atMost prod.lessThan-Suc*)
show *?thesis*
by(*induction n*, *simp-all*) (*simp add*: *right-diff-distrib*)
qed
finally show *?thesis* **.**
qed
{
assume *x*:*x* ∈ *K*
then obtain *i* **where** *i*: *i* ≤ *n x* ∈ *Hi i*
using *K-unHi* **by** *auto*
have *x* ∈ *topspace X*
using *K*(*1*) *x compactin-subset-topspace* **by** *auto*
hence ($\sum i \le n.\ hi\ i\ x$) = *1* − ($\prod i \le n.\ (1 - gi\ i\ x)$)
by(*simp add*: *1*)
also have ... = *1*
using *i gi*(*4*)[*OF i*(*1*)] **by**(*auto intro*!: *prod-zero bexI*[**where** *x*=*i*])
finally show ($\sum i \le n.\ hi\ i\ x$) = *1* **.**
}
assume *x*: *x* ∈ *topspace X*
then show *0* ≤ ($\sum i \le n.\ hi\ i\ x$) ($\sum i \le n.\ hi\ i\ x$) ≤ *1*
using *gi*(*2*) **by**(*auto simp*: *1 image-subset-iff intro*!: *prod-nonneg prod-le-1*)
next
fix *i x*
assume *h*:*i* ≤ *n x* ∈ *X closure-of* {*x* ∈ *topspace X*. *hi i x* ≠ *0*}
have {*x* ∈ *topspace X*. *hi i x* ≠ *0*} = {*x*∈*topspace X*. *gi i x* ≠ *0*} ∩ ($\bigcap j<i.$ {*x*∈*topspace X*. *gi j x* ≠ *1*})
using *gi* **by**(*auto simp*: *hi-def*)
also have ... ⊆ {*x*∈*topspace X*. *gi i x* ≠ *0*}
by *blast*
finally have *X closure-of* {*x* ∈ *topspace X*. *hi i x* ≠ *0*} ⊆ *X closure-of* {*x*∈*topspace X*. *gi i x* ≠ *0*}
by(*rule closure-of-mono*)
thus *x* ∈ *Vi i*
using *gi*(*5*)[*OF h*(*1*)] *h*(*2*) *closure-of-subset-topspace* **by**(*fastforce simp*: *disjnt-def*)

qed
qed
note [*simp*, *intro*!] = *continuous-map-add continuous-map-diff continuous-map-real-mult*
define μ' **where** $\mu' \equiv (\lambda A. \bigsqcup\ (ennreal\ `\ \varphi\ `\ \{(\lambda x\in topspace\ X.\ f\ x)\ |f.\ ?fA\ A\ f\}))$
define μ **where** $\mu \equiv (\lambda A. \bigsqcap\ (\mu'\ `\ \{V.\ A \subseteq V \wedge openin\ X\ V\}))$

define *Mf* **where** $Mf \equiv \{E.\ E \subseteq topspace\ X \wedge \mu\ E < \top \wedge \mu\ E = (\bigsqcup\ (\mu\ `\ \{K.\ K \subseteq E \wedge compactin\ X\ K\}))\}$
define *M* **where** $M \equiv \{E.\ E \subseteq topspace\ X \wedge (\forall K.\ compactin\ X\ K \longrightarrow E \cap K \in Mf)\}$

have μ'*-mono*: $\bigwedge A\ B.\ A \subseteq B \Longrightarrow \mu'\ A \leq \mu'\ B$
unfolding μ'*-def* **by**(*fastforce intro*!: *SUP-subset-mono imageI*)
have μ*-open*: $\mu\ A = \mu'\ A$ **if** *openin X A* **for** *A*
unfolding μ*-def* **by** (*metis* (*mono-tags*, *lifting*) *INF-eqI* μ'*-mono dual-order.refl mem-Collect-eq that*)
have μ*-mono*: $\bigwedge A\ B.\ A \subseteq B \Longrightarrow \mu\ A \leq \mu\ B$
unfolding μ*-def* **by**(*auto intro*!: *INF-superset-mono*)
have μ*-fin-subset*: $\mu\ A < \infty \Longrightarrow A \subseteq topspace\ X$ **for** *A*
by (*metis* (*mono-tags*, *lifting*) *INF-less-iff* μ*-def mem-Collect-eq openin-subset order.trans*)

have μ'*-empty*: $\mu'\ \{\} = 0$ **and** μ*-empty*: $\mu\ \{\} = 0$
proof −
have *1*: $\{(\lambda x\in topspace\ X.\ f\ x)\ |f.\ ?fA\ \{\}\ f\} = \{\lambda x\in topspace\ X.\ 0\}$
by(*fastforce intro*!: *exI*[**where** $x=\lambda x\in topspace\ X.\ 0$])
thus $\mu'\ \{\} = 0\ \ \mu\ \{\} = 0$
by(*auto simp*: μ'*-def* φ*-0* μ*-open*)
qed
have *empty-in-Mf*: $\{\} \in Mf$
by(*auto simp*: *Mf-def* μ*-empty*)

have *step1*: $\mu\ (\bigcup\ (range\ Ei)) \leq (\sum i.\ \mu\ (Ei\ i))$ **for** *Ei*
proof −
have *1*: $\mu'\ (V \cup U) \leq \mu'\ V + \mu'\ U$ **if** *VU*: *openin X V openin X U* **for** *V U*
proof −
have $\mu'\ (V \cup U) = \bigsqcup\ (ennreal\ `\ \varphi\ `\ \{(\lambda x\in topspace\ X.\ f\ x)\ |f.\ ?fA\ (V \cup U)\ f\})$
by(*simp add*: μ'*-def*)
also have ... $\leq \mu'\ V + \mu'\ U$
unfolding *Sup-le-iff*
proof *safe*
fix *g*
assume *g*: *?iscont g ?csupp g* $g \in topspace\ X \to \{0..1\}$ $g \in topspace\ X - (V \cup U) \to \{0\}$
$X\ closure\text{-}of\ \{x \in topspace\ X.\ g\ x \neq 0\} \subseteq V \cup U$
have $\exists hi.\ (\forall i\leq Suc\ 0.\ ?iscont\ (hi\ i) \wedge ?csupp\ (hi\ i) \wedge$
$X\ closure\text{-}of\ \{x \in topspace\ X.\ hi\ i\ x \neq 0\} \subseteq (case\ i\ of\ 0 \Rightarrow V\ |$

Suc - ⇒ *U*) ∧
hi i ∈ *topspace X* → {*0*..*1*} ∧
hi i ∈ *topspace X* − (*case i of 0* ⇒ *V* | *Suc* - ⇒ *U*) → {*0*}) ∧
(∀ *x*∈*X closure-of* {*x*∈*topspace X*. *g x* ≠ *0*}. (∑ *i*≤*Suc 0*. *hi i x*) = *1*) ∧
(∀ *x*∈*topspace X*. *0* ≤ (∑ *i*≤*Suc 0*. *hi i x*)) ∧ (∀ *x*∈*topspace X*. (∑ *i*≤*Suc 0*. *hi i x*) ≤ *1*)
proof(*safe intro*!: *fApartition*[*of* - *Suc 0* λ*i*. *case i of 0* ⇒ *V* | - ⇒ *U*])
have *1*:(⋃ *i*≤*Suc 0*. *case i of 0* ⇒ *V* | *Suc* - ⇒ *U*) = *U* ∪ *V*
by(*fastforce simp*: *le-Suc-eq*)
show ⋀*x*. *x* ∈ *X closure-of* {*x* ∈ *topspace X*. *g x* ≠ *0*} ⟹ *x* ∈ (⋃ *i*≤*Suc 0*. *case i of 0* ⇒ *V* | *Suc* - ⇒ *U*)
unfolding *1* **using** *g* **by** *blast*
next
show *compactin X* (*X closure-of* {*x* ∈ *topspace X*. *g x* ≠ *0*})
using *g* **by**(*simp add*: *has-compact-support-on-iff*)
qed(*use g VU le-Suc-eq* **in** *auto*)
then obtain *hi* **where**
(∀ *i*≤*Suc 0*. *?iscont* (*hi i*) ∧ *?csupp* (*hi i*) ∧
X closure-of {*x* ∈ *topspace X*. *hi i x* ≠ *0*} ⊆ (*case i of 0* ⇒ *V* | *Suc* - ⇒ *U*) ∧
hi i ∈ *topspace X* → {*0*..*1*} ∧ *hi i* ∈ *topspace X* − (*case i of 0* ⇒ *V* | *Suc* - ⇒ *U*) → {*0*}) ∧
(∀ *x*∈*X closure-of* {*x*∈*topspace X*. *g x* ≠ *0*}. (∑ *i*≤*Suc 0*. *hi i x*) = *1*) ∧
(∀ *x*∈*topspace X*. *0* ≤ (∑ *i*≤*Suc 0*. *hi i x*)) ∧ (∀ *x*∈*topspace X*. (∑ *i*≤*Suc 0*. *hi i x*) ≤ *1*)
by *blast*
hence *h0*: *?iscont* (*hi 0*) *?csupp* (*hi 0*) *X closure-of* {*x* ∈ *topspace X*. *hi 0 x* ≠ *0*} ⊆ *V*
hi 0 ∈ *topspace X* → {*0*..*1*} *hi 0* ∈ *topspace X* − *V* → {*0*}
and *h1*: *?iscont* (*hi* (*Suc 0*)) *?csupp* (*hi* (*Suc 0*)) *X closure-of* {*x* ∈ *topspace X*. *hi* (*Suc 0*) *x* ≠ *0*} ⊆ *U*
hi (*Suc 0*) ∈ *topspace X* → {*0*..*1*} *hi* (*Suc 0*) ∈ *topspace X* − *U* → {*0*}
and *h01-sum*: ⋀*x*. *x* ∈ *X closure-of* {*x*∈*topspace X*. *g x* ≠ *0*} ⟹ (∑ *i*≤*Suc 0*. *hi i x*) = *1*
unfolding *le-Suc-eq le-0-eq* **by** *auto*
have *ennreal* (φ (λ*x*∈*topspace X*. *g x*)) = *ennreal* (φ (λ*x*∈*topspace X*. *g x* ∗ (*hi 0 x* + *hi* (*Suc 0*) *x*)))
proof −
have [*simp*]: (λ*x*∈*topspace X*. *g x*) = (λ*x*∈*topspace X*. *g x* ∗ (*hi 0 x* + *hi* (*Suc 0*) *x*))
proof
fix *x*
consider *g x* ≠ *0 x* ∈ *topspace X* | *g x* = *0* | *x* ∉ *topspace X*
by *fastforce*
then show *restrict g* (*topspace X*) *x* = (λ*x*∈*topspace X*. *g x* ∗ (*hi 0 x* + *hi* (*Suc 0*) *x*)) *x*
proof *cases*
case *1*

then have $x \in X$ *closure-of* $\{x{\in}topspace\ X.\ g\ x \neq 0\}$
using *in-closure-of* **by** *fastforce*
from *h01-sum*[*OF this*] **show** *?thesis*
by *simp*
qed *simp-all*
qed
show *?thesis*
by *simp*
qed
also have ... $=$ *ennreal* (φ ($\lambda x{\in}topspace\ X.\ g\ x * hi\ 0\ x + g\ x * hi\ (Suc\ 0)\ x$))
by (*simp add*: *ring-class.ring-distribs*(*1*))
also have ... $=$ *ennreal* (φ ($\lambda x{\in}topspace\ X.\ g\ x * hi\ 0\ x$) $+$ φ ($\lambda x{\in}topspace\ X.\ g\ x * hi\ (Suc\ 0)\ x$))
by(*intro ennreal-cong linear*(*2*) *has-compact-support-on-mult-left continuous-map-real-mult g h0 h1*)
also have ... $=$ *ennreal* (φ ($\lambda x{\in}topspace\ X.\ g\ x * hi\ 0\ x$)) $+$ *ennreal* (φ ($\lambda x{\in}topspace\ X.\ g\ x * hi\ (Suc\ 0)\ x$))
using *g*(*3*) *h0*(*4*) *h1*(*4*)
by(*auto intro*!: *ennreal-plus pos g h0 h1 has-compact-support-on-mult-left mult-nonneg-nonneg*)
also have ... $\leq \mu'\ V + \mu'\ U$
unfolding μ'*-def*
proof(*safe intro*!: *add-mono Sup-upper imageI*)
show $\exists f.\ (\lambda x{\in}topspace\ X.\ g\ x * hi\ 0\ x) = restrict\ f\ (topspace\ X) \wedge\ ?iscont\ f \wedge\ ?csupp\ f \wedge$
X *closure-of* $\{x \in topspace\ X.\ f\ x \neq 0\} \subseteq V \wedge f \in topspace\ X \rightarrow \{0..1\} \wedge f \in topspace\ X - V \rightarrow \{0\}$
using *Pi-mem*[*OF g*(*3*)] *Pi-mem*[*OF h0*(*4*)] *in-mono*[*OF closure-of-mono*[*OF inf-sup-ord*(*2*)[*of* $\{x \in topspace\ X.\ g\ x \neq 0\}$]]] *h0*(*3*,*5*)
by(*auto intro*!: *exI*[**where** $x{=}\lambda x{\in}topspace\ X.\ \ g\ x * hi\ 0\ x$] *g*(*1*,*2*) *h0*(*1*,*2*) *has-compact-support-on-mult-left mult-le-one mult-nonneg-nonneg*)
show $\exists f.\ (\lambda x{\in}topspace\ X.\ g\ x * hi\ (Suc\ 0)\ x) = restrict\ f\ (topspace\ X) \wedge\ ?iscont\ f \wedge$
$?csupp\ f \wedge X$ *closure-of* $\{x \in topspace\ X.\ f\ x \neq 0\} \subseteq U \wedge f \in topspace\ X \rightarrow \{0..1\} \wedge f \in topspace\ X - U \rightarrow \{0\}$
using *Pi-mem*[*OF g*(*3*)] *Pi-mem*[*OF h1*(*4*)] *in-mono*[*OF closure-of-mono*[*OF inf-sup-ord*(*2*)[*of* $\{x \in topspace\ X.\ g\ x \neq 0\}$]]] *h1*(*3*,*5*)
by(*auto intro*!: *exI*[**where** $x{=}\lambda x{\in}topspace\ X.\ \ g\ x * hi\ 1\ x$] *g*(*1*,*2*) *h1*(*1*,*2*) *has-compact-support-on-mult-left mult-le-one mult-nonneg-nonneg*)
qed
finally show *ennreal* (φ (*restrict g* (*topspace X*))) $\leq \mu'\ V + \mu'\ U$ **.**
qed
finally show $\mu'\ (V \cup U) \leq \mu'\ V + \mu'\ U$ **.**
qed

consider $\exists i.\ \mu\ (Ei\ i) = \infty \mid \bigwedge i.\ \mu\ (Ei\ i) < \infty$
using *top.not-eq-extremum* **by** *auto*
then show *?thesis*

proof *cases*
case *1*
then show *?thesis*
by (*metis μ-mono ennreal-suminf-lessD infinity-ennreal-def linorder-not-le subset-UNIV top.not-eq-extremum*)
next
case *fin:2*
show *?thesis*
proof(*rule ennreal-le-epsilon*)
fix *e* :: *real*
assume *e*: $0 < e$
have $\exists$ *Vi*. *Ei i* $\subseteq$ *Vi* $\wedge$ *openin X Vi* $\wedge$ μ' *Vi* $\leq$ μ (*Ei i*) + *ennreal* ((*1* / *2*)^*Suc i*) $*$ *ennreal e* **for** *i*
proof −
have *1*:μ (*Ei i*) < μ (*Ei i*) + *ennreal* ((*1* / *2*) ^ *Suc i*) $*$ *ennreal e*
using *e fin less-le* **by** *fastforce*
have *0* < *ennreal* ((*1* / *2*)^*Suc i*) $*$ *ennreal e*
using *e* **by** (*simp add*: *ennreal-zero-less-mult-iff*)
have ($\bigsqcap$ (μ' ‘ {*V*. *Ei i* $\subseteq$ *V* $\wedge$ *openin X V*})) $\leq$ μ (*Ei i*)
by(*auto simp*: *μ-def*)
from *Inf-le-iff*[*THEN iffD1*,*OF this*,*rule-format*,*OF 1*]
show *?thesis*
by *auto*
qed
then obtain *Vi* **where** *Vi*: $\bigwedge$*i*. *Vi i* $\supseteq$ *Ei i* $\bigwedge$*i*. *openin X* (*Vi i*)
$\bigwedge$*i*. μ (*Vi i*) $\leq$ μ (*Ei i*) + *ennreal* ((*1* / *2*)^*Suc i*) $*$ *ennreal e*
by (*metis μ-open*)
hence μ ($\bigcup$ (*range Ei*)) $\leq$ μ ($\bigcup$ (*range Vi*))
by(*auto intro*!: *μ-mono*)
also have ... = μ' ($\bigcup$ (*range Vi*))
using *Vi* **by**(*auto intro*!: *μ-open*)
also have ... = ($\bigsqcup$ (*ennreal* ‘ φ ‘ {($\lambda x \in$*topspace X*. *f x*) |*f*. *?fA* ($\bigcup$ (*range Vi*)) *f*}))
by(*simp add*: *μ′-def*)
also have ... $\leq$ ($\sum$ *i*. μ (*Ei i*)) + *ennreal e*
unfolding *Sup-le-iff*
proof *safe*
fix *f*
assume *f*: *?iscont f ?csupp f X closure-of* {*x* $\in$ *topspace X*. *f x* $\neq$ *0*} $\subseteq$ $\bigcup$ (*range Vi*) *f* $\in$ *topspace X* $\to$ {*0*..*1*} *f* $\in$ *topspace X* − $\bigcup$ (*range Vi*) $\to$ {*0*}
have $\exists$ *n*. *f* $\in$ *topspace X* − ($\bigcup$*i*$\leq$*n*. *Vi i*) $\to$ {*0*} $\wedge$ *X closure-of* {*x* $\in$ *topspace X*. *f x* $\neq$ *0*} $\subseteq$ ($\bigcup$*i*$\leq$*n*. *Vi i*)
proof −
obtain *K* **where** *K*:*finite K K* $\subseteq$ *range Vi X closure-of* {*x* $\in$ *topspace X*. *f x* $\neq$ *0*} $\subseteq$ $\bigcup$ *K*
using *compactinD*[*OF f*(*2*)[*simplified has-compact-support-on-iff*]] *Vi*(*2*) *f*(*3*)
by (*metis* (*mono-tags*, *lifting*) *imageE*)
hence $\exists$ *n*. *K* $\subseteq$ *Vi* ‘ {..*n*}

by (*metis* (*no-types*, *lifting*) *finite-nat-iff-bounded-le finite-subset-image image-mono*)
then obtain *n* **where** *n*: *X closure-of* $\{x \in$ *topspace X*. *f x* $\neq$ *0*$\}$ $\subseteq$ ($\bigcup$*i*$\leq$*n*. *Vi i*)
using *K*(*3*) **by** *fastforce*
show *?thesis*
using *in-closure-of n subsetD* **by**(*fastforce intro*!: *exI*[**where** *x*=*n*])
qed
then obtain *n* **where** *n*:*f* $\in$ *topspace X* $-$ ($\bigcup$*i*$\leq$*n*. *Vi i*) $\to$ $\{$*0*$\}$ *X closure-of* $\{x \in$ *topspace X*. *f x* $\neq$ *0*$\}$ $\subseteq$ ($\bigcup$*i*$\leq$*n*. *Vi i*)
by *blast*
have *ennreal* (φ (*restrict f* (*topspace X*))) $\leq$ μ' ($\bigcup$*i*$\leq$*n*. *Vi i*)
using *f*(*4*) *f n* **by**(*auto intro*!: *imageI exI*[**where** *x*=*f*] *Sup-upper simp*: μ'*-def*)
also have ... $\leq$ ($\sum$*i*$\leq$*n*. μ' (*Vi i*))
proof(*induction n*)
case *ih*:(*Suc n'*)
have [*simp*]:μ' ($\bigcup$ (*Vi* ‘ $\{$..*Suc n'*$\}$)) = μ' ($\bigcup$ (*Vi* ‘ $\{$..*n'*$\}$) $\cup$ *Vi* (*Suc n'*))
by(*rule arg-cong*[*of - -* μ']) (*fastforce simp*: *le-Suc-eq*)
also have ... $\leq$ μ' ($\bigcup$ (*Vi* ‘ $\{$..*n'*$\}$)) + μ' (*Vi* (*Suc n'*))
using *Vi*(*2*) **by**(*auto intro*!: *1*)
also have ... $\leq$ ($\sum$*i*$\leq$*Suc n'*. μ' (*Vi i*))
using *ih* **by** *fastforce*
finally show *?case* **.**
qed *simp*
also have ... = ($\sum$*i*$\leq$*n*. μ (*Vi i*))
using *Vi*(*2*) **by**(*simp add*: μ*-open*)
also have ... $\leq$ ($\sum$*i*. μ (*Vi i*))
by (*auto intro*!: *incseq-SucI incseq-le*[*OF - summable-LIMSEQ'*])
also have ... $\leq$ ($\sum$*i*. μ (*Ei i*) + *ennreal* ((*1* / *2*)^*Suc i*) $*$ *ennreal e*)
by(*intro suminf-le Vi*(*3*)) *auto*
also have ... = ($\sum$*i*. μ (*Ei i*)) + ($\sum$*i*. *ennreal* ((*1* / *2*)^*Suc i*) $*$ *ennreal e*)
by(*rule suminf-add*[*symmetric*]) *auto*
also have ... = ($\sum$*i*. μ (*Ei i*)) + ($\sum$*i*. *ennreal* ((*1* / *2*)^*Suc i*)) $*$ *ennreal e*
by *simp*
also have ... = ($\sum$*i*. μ (*Ei i*)) + *ennreal 1* $*$ *ennreal e*
proof $-$
have ($\sum$*i*. *ennreal* ((*1* / *2*)^*Suc i*)) = *ennreal 1*
by(*rule suminf-ennreal-eq*) (*use power-half-series* **in** *auto*)
thus *?thesis*
by *presburger*
qed
also have ... = ($\sum$*i*. μ (*Ei i*)) + *ennreal e*
by *simp*
finally show *ennreal* (φ (*restrict f* (*topspace X*))) $\leq$ ($\sum$*i*. μ (*Ei i*)) + *ennreal e* **.**
qed

```
      finally show μ (⋃ (range Ei)) ≤ (∑ i. μ (Ei i)) + ennreal e .
    qed
  qed
qed
have step1': μ (E1 ∪ E2) ≤ μ E1 + μ E2 for E1 E2
proof –
  define En where En ≡ (λn::nat. if n = 0 then E1 else if n = 1 then E2 else
{})
  have 1: (⋃ (range En)) = (E1 ∪ E2)
    by(auto simp: En-def)
  have 2: (∑ i. μ (En i)) = μ E1 + μ E2
    using suminf-offset[of λi. μ (En i),of Suc (Suc 0)]
    by(auto simp: En-def μ-empty)
  show ?thesis
    using step1[of En] by(simp add: 1 2)
qed
have step2: K ∈ Mf μ K = (⨅ (ennreal ‘ φ ‘ {(λx∈topspace X. f x) | f. ?fK K
f})) if K: compactin X K for K
proof –
  have le1: μ K ≤ ennreal (φ (λx∈topspace X. f x)) if f:?iscont f ?csupp f f ∈
topspace X → {0..1} f ∈ K → {1} for f
  proof –
    have f: continuous-map X (top-of-set {0..1::real}) f f ‘ K ⊆ {1} ?csupp f
      using f by (auto simp: continuous-map-in-subtopology)
    hence f-cont: ?iscont f f ∈ topspace X → {0..1}
      by (auto simp add: continuous-map-in-subtopology)
    have 1:μ K ≤ ennreal (1 / ((real n + 1) / (real n + 2)) ∗ φ (λx∈topspace
X. f x)) for n
    proof –
      let ?a = (real n + 1) / (real n + 2)
      define V where V ≡ {x∈topspace X. ?a < f x}
      have openinV:openin X V
     using f(1)by (auto simp: V-def continuous-map-upper-lower-semicontinuous-lt-gen)
      have KV: K ⊆ V
        using f(2) compactin-subset-topspace[OF K] by(auto simp: V-def)
      hence μ K ≤ μ V
        by(rule μ-mono)
      also have ... = μ' V
        by(simp add: μ-open openinV)
      also have ... = (⨆ (ennreal ‘ φ ‘ {(λx∈topspace X. f x) |f. ?fA V f}))
        by(simp add: μ'-def)
      also have ... ≤ (1 / ?a) ∗ φ (λx∈topspace X. f x)
        unfolding Sup-le-iff
      proof (safe intro!: ennreal-leI)
        fix g
        assume g: ?iscont g ?csupp g X closure-of {x∈topspace X. g x ≠ 0} ⊆ V
          g ∈ topspace X → {0..1} g ∈ topspace X − V → {0}
        show φ (restrict g (topspace X)) ≤ 1 / ?a ∗ φ (restrict f (topspace X))
(is ?l ≤ ?r)
```

proof −
have *?l* ≤ *φ* (*λx*∈*topspace X*. *1* / *?a* ∗ *f x*)
proof(*rule φmono*)
fix *x*
assume *x*: *x* ∈ *topspace X*
consider *g x* ≠ *0* | *g x* = *0*
by *fastforce*
then show *g x* ≤ *1* / ((*real n* + *1*) / (*real n* + *2*)) ∗ *f x*
proof *cases*
case *1*
then have *x* ∈ *V*
using *g*(*5*) *x* **by** *auto*
hence *?a* < *f x*
by(*auto simp*: *V-def x*)
hence *1* < *1* / *?a* ∗ *f x*
by (*simp add*: *divide-less-eq mult.commute*)
thus *?thesis*
by(*intro order.strict-implies-order*[*OF order.strict-trans1*[*of g x 1 1* / *?a* ∗ *f x*]]) (*use g*(*4*) *x* **in** *auto*)
qed(*use Pi-mem*[*OF f-cont*(*2*)] *x* **in** *auto*)
qed(*intro g f-cont f has-compact-support-on-mult-left continuous-map-real-mult continuous-map-canonical-const*)+
also have ... = *?r*
by(*intro linear f f-cont*)
finally show *?thesis* **.**
qed
qed
finally show *?thesis* **.**
qed
have *2*:(*λn*. *ennreal* (*1* / ((*real n* + *1*) / (*real n* + *2*)) ∗ *φ* (*restrict f* (*topspace X*))))
⟶ *ennreal* (*φ* (*restrict f* (*topspace X*)))
proof(*intro tendsto-ennrealI tendsto-mult-right*[**where** *l*=*1*::*real*,*simplified*])
have *1*: (*λn*. *1* / ((*real n* + *1*) / (*real n* + *2*))) = (*λn*. *real* (*Suc* (*Suc n*)) / *real* (*Suc n*))
by (*simp add*: *add.commute*)
show (*λn*. *1* / ((*real n* + *1*) / (*real n* + *2*))) ⟶ *1*
unfolding *1* **by**(*rule LIMSEQ-Suc*[*OF LIMSEQ-Suc-n-over-n*])
qed
show *μ K* ≤ *ennreal* (*φ* (*λx*∈*topspace X*. *f x*))
by(*rule Lim-bounded2*[**where** *N*=*0*,*OF 2*]) (*use 1* **in** *auto*)
qed
have *muK-fin*:*μ K* < ⊤
proof −
obtain *f* **where** *f*: *continuous-map X* (*top-of-set* {*0*..*1*::*real*}) *f f* ‘ *K* ⊆ {*1*} *?csupp f*
using *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF lh*(*1*) *disjI1*[*OF lh*(*2*)] *zero-le-one closedin-empty K*] **by**(*auto simp*: *has-compact-support-on-iff*)

hence *?iscont f ?csupp f f* ∈ *topspace X* → {*0..1*} *f* ∈ *K* → {*1*}
by(*auto simp*: *continuous-map-in-subtopology*)
from *le1*[*OF this*]
show *?thesis*
using *dual-order.strict-trans2 ennreal-less-top* **by** *blast*
qed
moreover have μ *K* = (⨆ (μ ' {*K'*. *K'* ⊆ *K* ∧ *compactin X K'*}))
by (*metis* (*no-types*, *lifting*) *SUP-eqI* μ*-mono mem-Collect-eq subset-refl K*)
ultimately show *K* ∈ *Mf*
using *compactin-subset-topspace*[*OF K*] **by**(*simp add*: *Mf-def*)

show μ *K* = (⨅ (*ennreal* ' φ ' {(λ*x*∈*topspace X*. *f x*) |*f*. *?fK K f*}))
proof(*safe intro*!: *antisym le-Inf-iff*[*THEN iffD2*] *Inf-le-iff*[*THEN iffD2*])
fix *g*
assume *?iscont g ?csupp g g* ∈ *topspace X* → {*0..1*} *g* ∈ *K* → {*1*}
from *le1*[*OF this*(*1*−*4*)]
show μ *K* ≤ *ennreal* (φ (λ*x*∈*topspace X*. *g x*))
by *force*
next
fix *y*
assume μ *K* < *y*
then obtain *V* **where** *V*: *openin X V K* ⊆ *V* μ' *V* < *y*
by (*metis* (*mono-tags*, *lifting*) *INF-less-iff* μ*-def mem-Collect-eq*)
hence *closedin X* (*topspace X* − *V*) *disjnt* (*topspace X* − *V*) *K*
by (*auto simp*: *disjnt-def*)
from *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF lh*(*1*) *disjI1*[*OF lh*(*2*)] *zero-le-one this*(*1*) *K this*(*2*)]
obtain *f* **where** *f'*:*continuous-map X* (*subtopology euclidean* {*0..1*}) *f f* ' (*topspace X* − *V*) ⊆ {*0*::*real*}
f ' *K* ⊆ {*1*} *disjnt* (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}) (*topspace X* − *V*)
compactin X (*X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*})
by *blast*
hence *f*:*?iscont f ?csupp f* ⋀*x*. *x* ∈ *topspace X* ⟹ *f x* ≥ *0*
⋀*x*. *x* ∈ *topspace X* ⟹ *f x* ≤ *1* ⋀*x*. *x* ∈ *K* ⟹ *f x* = *1*
by(*auto simp*: *has-compact-support-on-iff continuous-map-in-subtopology*)
have *ennreal* (φ (*restrict f* (*topspace X*))) < *y*
proof(*rule order.strict-trans1*)
show *ennreal* (φ (*restrict f* (*topspace X*))) ≤ μ' *V*
unfolding μ'*-def* **using** *f' f in-closure-of*
by (*fastforce intro*!: *Sup-upper imageI exI*[**where** *x*=λ*x*∈*topspace X*. *f x*] *simp*: *disjnt-iff*)
qed *fact*
thus ∃ *a*∈*ennreal* ' φ ' {(λ*x*∈*topspace X*. *f x*)|*f*. *?fK K f*}. *a* < *y*
using *f compactin-subset-topspace*[*OF K*] **by**(*auto intro*!: *exI*[**where** *x*=λ*x*∈*topspace X*. *f x*])
qed
qed
have μ*-K*: μ *K* ≤ *ennreal* (φ (λ*x*∈*topspace X*. *f x*)) **if** *K*: *compactin X K* **and**

f:?fK K f **for** K f
  using *le-Inf-iff*[*THEN iffD1*,*OF eq-refl*[*OF step2*(*2*)[*OF K*]]] *f* **by** *blast*
 have *step3*: μ *A* = ($\bigsqcup$*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. μ *K*) μ *A* < ∞ ⟹ *A* ∈ *Mf* **if** *A*:*openin X A* **for** *A*
 proof −
  show μ *A* = ($\bigsqcup$*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. μ *K*)
  proof(*safe intro*!: *antisym le-Sup-iff*[*THEN iffD2*] *Sup-le-iff*[*THEN iffD2*])
   fix *y*
   assume *y*: *y* < μ *A*
  from *less-SUP-iff*[*THEN iffD1*,*OF less-INF-D*[*OF y*[*simplified* μ*-def*],*simplified* μ'*-def*],*of A*]
   obtain *f* **where** *f*: *?iscont f ?csupp f X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*} ⊆ *A*
    f ∈ *topspace X* → {*0*..*1*} *f* ∈ *topspace X* − *A* → {*0*} *y* < *ennreal* (φ (λ*x*∈*topspace X*. *f x*))
    using *A* **by** *blast*
   show ∃ *a*∈μ ‘ {*K*. *compactin X K* ∧ *K* ⊆ *A*}. *y* < *a*
   proof(*rule bexI*[**where** *x*=μ (*X closure-of* {*x* ∈ *topspace X*. *f x* ≠ *0*})])
    show *y* < μ (*X closure-of* {*a* ∈ *topspace X*. *f a* ≠ *0*})
    proof(*rule order.strict-trans2*)
     show *ennreal* (φ (λ*x*∈*topspace X*. *f x*)) ≤ μ (*X closure-of* {*a* ∈ *topspace X*. *f a* ≠ *0*})
      using *f in-closure-of in-mono*
     by(*fastforce intro*!: *Sup-upper imageI exI*[**where** *x*=*f*] *simp*: μ*-def le-Inf-iff* μ'*-def*)
    qed *fact*
   qed(*use f*(*2*,*3*) *has-compact-support-on-iff* **in** *auto*)
  qed(*auto intro*!: μ*-mono*)
  thus μ *A* < ∞ ⟹ *A* ∈ *Mf*
   unfolding *Mf-def* **using** *openin-subset*[*OF A*] **by** *simp metis*
 qed
 have *step4*: μ ($\bigcup$*n*. *En n*) = ($\sum$*n*. μ (*En n*)) μ ($\bigcup$*n*. *En n*) < ∞ ⟹ ($\bigcup$*n*. *En n*) ∈ *Mf*
  if *En*: $\bigwedge$*n*. *En n* ∈ *Mf disjoint-family En* **for** *En*
 proof −
  have *compacts*: μ (*K1* ∪ *K2*) = μ *K1* + μ *K2* **if** *K*: *compactin X K1 compactin X K2 disjnt K1 K2* **for** *K1 K2*
  proof(*rule antisym*)
   show μ (*K1* ∪ *K2*) ≤ μ *K1* + μ *K2*
    by(*rule step1'*)
  next
   show μ *K1* + μ *K2* ≤ μ (*K1* ∪ *K2*)
   proof(*rule ennreal-le-epsilon*)
    fix *e* :: *real*
    assume *e*: *0* < *e* μ (*K1* ∪ *K2*) < ⊤
    from *Urysohn-locally-compact-Hausdorff-closed-compact-support*[*OF lh*(*1*) *disjI1*[*OF lh*(*2*)]
     zero-le-one compactin-imp-closedin[*OF lh*(*2*) *K*(*1*)] *K*(*2*,*3*)]
    obtain *f* **where** *f*: *continuous-map X* (*top-of-set* {*0*..*1*::*real*}) *f f* ‘ *K1* ⊆

{0} f ` K2 ⊆ {1}
disjnt (X closure-of {x ∈ topspace X. f x ≠ 0}) K1 compactin X (X closure-of {x ∈ topspace X. f x ≠ 0})
by *blast*
hence *f′: ?iscont f ?csupp f ⋀x. x ∈ topspace X ⟹ f x ≥ 0 ⋀x. x ∈ topspace X ⟹ f x ≤ 1*
by(*auto simp*: *has-compact-support-on-iff continuous-map-in-subtopology*)
from *Inf-le-iff*[*THEN iffD1*,*OF eq-refl*[*OF step2*(*2*)[*symmetric*,*OF compactin-Un*[*OF K*(*1*,*2*)]]],*rule-format*,*of μ (K1 ∪ K2) + ennreal e*]
obtain *g* **where** *g*: *?iscont g ?csupp g g ∈ topspace X → {0..1} g ∈ K1 ∪ K2 → {1}*
ennreal (φ (λx∈topspace X. g x)) < μ (K1 ∪ K2) + ennreal e
using *e* **by** *fastforce*
have *μ K1 + μ K2 ≤ ennreal (φ (λx∈topspace X. (1 − f x) ∗ g x)) + ennreal (φ (λx∈topspace X. f x ∗ g x))*
proof(*rule add-mono*)
show *μ K1 ≤ ennreal (φ (λx∈topspace X. (1 − f x) ∗ g x))*
using *f′ Pi-mem*[*OF g*(*3*)] *g*(*1*,*2*,*4*,*5*) *f*(*2*) *compactin-subset-topspace*[*OF K*(*1*)]
by(*auto intro*!: *μ-K has-compact-support-on-mult-left mult-nonneg-nonneg mult-le-one K*(*1*) *mult-eq-1*[*THEN iffD2*])
show *μ K2 ≤ ennreal (φ (λx∈topspace X. f x ∗ g x))*
using *g f Pi-mem*[*OF g*(*3*)] *f′ compactin-subset-topspace*[*OF K*(*2*)]
by(*auto intro*!: *μ-K*[*OF K*(*2*)] *has-compact-support-on-mult-left mult-nonneg-nonneg mult-le-one mult-eq-1*[*THEN iffD2*])
qed
also have ... = *ennreal (φ (λx∈topspace X. (1 − f x)∗g x) + φ (λx∈topspace X. f x ∗ g x))*
using *f′ g* **by**(*auto intro*!: *ennreal-plus*[*symmetric*] *pos has-compact-support-on-mult-left mult-nonneg-nonneg*)
also have ... = *ennreal (φ (λx∈topspace X. (1 − f x) ∗ g x + f x ∗ g x))*
by(*auto intro*!: *ennreal-cong linear*[*symmetric*] *has-compact-support-on-mult-left f′ g*)
also have ... = *ennreal (φ (λx∈topspace X. g x))*
by (*simp add*: *Groups.mult-ac*(*2*) *right-diff-distrib*)
also have ... < *μ (K1 ∪ K2) + ennreal e*
by *fact*
finally show *μ K1 + μ K2 ≤ μ (K1 ∪ K2) + ennreal e*
by *order*
qed
qed
have *Hn*:*∃ Hn. ∀ n. compactin X (Hn n) ∧ (Hn n) ⊆ En n ∧ μ (En n) < μ (Hn n) + ennreal ((1 / 2)^Suc n) ∗ ennreal e′*
if *e′*: *e′ > 0* **for** *e′*
proof(*safe intro*!: *choice*)
show *∃ Hn. compactin X Hn ∧ Hn ⊆ En n ∧ μ (En n) < μ Hn + ennreal ((1 / 2)^Suc n) ∗ ennreal e′* **for** *n*
proof(*cases μ (En n) < ennreal ((1 / 2)^Suc n) ∗ ennreal e′*)
case *True*

```
    then show ?thesis
    using e' by(auto intro!: exI[where x={}] simp: μ-empty ennreal-zero-less-mult-iff)
  next
    case False
    then have le: μ (En n) ≥ ennreal ((1 / 2) ^ Suc n) ∗ ennreal e'
      by order
    hence pos:0 < μ (En n)
      using e' zero-less-power by fastforce
    have fin: μ (En n) < ⊤
      using En Mf-def by blast
    hence 1:μ (En n) − ennreal ((1 / 2)^Suc n) ∗ ennreal e' < μ (En n)
      using pos by(auto intro!: ennreal-between simp: ennreal-zero-less-mult-iff
e')
    have μ (En n) = ⨆ (μ ‘ {K. K ⊆ (En n) ∧ compactin X K})
      using En by(auto simp: Mf-def)
    from le-Sup-iff[THEN iffD1,OF eq-refl[OF this],rule-format,OF 1]
    obtain Hn where Hn: Hn ⊆ En n compactin X Hn μ (En n) − ennreal ((1
/ 2)^Suc n) ∗ ennreal e' < μ Hn
      by blast
    hence μ (En n) < μ Hn + ennreal ((1 / 2)^Suc n) ∗ ennreal e'
      by (metis diff-diff-ennreal' diff-gt-0-iff-gt-ennreal fin le order-less-le)
    with Hn(1,2) show ?thesis
      by blast
  qed
qed
show 1:μ (⋃n. En n) = (∑n. μ (En n))
proof(rule antisym)
  show (∑n. μ (En n)) ≤ μ (⋃ (range En))
  proof(rule ennreal-le-epsilon)
    fix e :: real
    assume fin: μ (⋃ (range En)) < ⊤ and e:0 < e
    from Hn[OF e] obtain Hn where Hn: ⋀n. compactin X (Hn n) ⋀n. Hn
n ⊆ En n
      ⋀n. μ (En n) < μ (Hn n) + ennreal ((1 / 2) ^ Suc n) ∗ ennreal e
      by blast
    have (∑n≤N. μ (En n)) ≤ μ (⋃ (range En)) + ennreal e for N
    proof −
      have (∑n≤N. μ (En n)) ≤ (∑n≤N. μ (Hn n) + ennreal ((1 / 2) ^ Suc
n) ∗ ennreal e)
        by(rule sum-mono) (use Hn(3) order-less-le in auto)
      also have ... = (∑n≤N. μ (Hn n)) + (∑n≤N. ennreal ((1 / 2) ^ Suc
n) ∗ ennreal e)
        by(rule sum.distrib)
      also have ... = μ (⋃n≤N. Hn n) + (∑n≤N. ennreal ((1 / 2) ^ Suc n)
∗ ennreal e)
      proof −
        have (∑n≤N. μ (Hn n)) = μ (⋃n≤N. Hn n)
        proof(induction N)
          case ih:(Suc N')
```

show *?case* (**is** *?l* = *?r*)
proof −
have *?l* = μ ($\bigcup$ (*Hn* ‘ {..*N'*})) + μ (*Hn* (*Suc N'*))
by(*simp add*: *ih*)
also have ... = μ (($\bigcup$ (*Hn* ‘ {..*N'*})) ∪ *Hn* (*Suc N'*))
proof(*rule compacts*[*symmetric*])
show *disjnt* ($\bigcup$ (*Hn* ‘ {..*N'*})) (*Hn* (*Suc N'*))
using *En*(*2*) *Hn*(*2*) **unfolding** *disjoint-family-on-def disjnt-iff*
by (*metis Int-iff Suc-n-not-le-n UNIV-I UN-iff atMost-iff empty-iff in-mono*)
qed(*auto intro*!: *compactin-Union Hn*)
also have ... = *?r*
by (*simp add*: *Un-commute atMost-Suc*)
finally show *?thesis* **.**
qed
qed *simp*
thus *?thesis*
by *simp*
qed
also have ... ≤ μ ($\bigcup$ (*range En*)) + ($\sum$ *n*≤*N*. *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*)
using *Hn*(*2*) **by**(*auto intro*!: *μ-mono*)
also have ... ≤ μ ($\bigcup$ (*range En*)) + *ennreal e*
proof −
have ($\sum$ *n*≤*N*. *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*) = *ennreal* ($\sum$ *n*≤*N*. ((*1* / *2*) ^ *Suc n*)) ∗ *ennreal e*
unfolding *sum-distrib-right*[*symmetric*] **by** *simp*
also have ... = *ennreal e* ∗ *ennreal* ($\sum$ *n*≤*N*. ((*1* / *2*) ^ *Suc n*))
using *mult.commute* **by** *blast*
also have ... ≤ *ennreal e* ∗ *ennreal* ($\sum$ *n*. ((*1* / *2*) ^ *Suc n*))
using *e* **by**(*auto intro*!: *ennreal-mult-le-mult-iff*[*THEN iffD2*] *ennreal-leI sum-le-suminf*)
also have ... = *ennreal e*
using *power-half-series sums-unique* **by** *fastforce*
finally show *?thesis*
by *fastforce*
qed
finally show *?thesis* **.**
qed
thus ($\sum$ *n*. μ (*En n*)) ≤ μ ($\bigcup$ (*range En*)) + *ennreal e*
by(*auto intro*!: *LIMSEQ-le-const2*[*OF summable-LIMSEQ'*] *exI*[**where** *x*=*0*])
qed
qed *fact*
show $\bigcup$ (*range En*) ∈ *Mf* **if** μ ($\bigcup$ (*range En*)) < ∞
proof −
have μ ($\bigcup$ (*range En*)) = ($\bigsqcup$ (μ ‘ {*K*. *K* ⊆ ($\bigcup$ (*range En*)) ∧ *compactin X K*}))
proof(*rule antisym*)

show μ ($\bigcup$ (*range En*)) ≤ $\bigsqcup$ (μ ‘ {*K*. *K* ⊆ $\bigcup$ (*range En*) ∧ *compactin X K*})
unfolding *le-Sup-iff*
proof *safe*
fix *y*
assume *y* < μ ($\bigcup$ (*range En*))
from *order-tendstoD*(*1*)[*OF summable-LIMSEQ′ this*[*simplified 1*]]
obtain *N* **where** *N*: *y* < ($\sum$ *n*≤*N*. μ (*En n*))
by *fastforce*
obtain *e* **where** *e*: *e* > *0 y* < ($\sum$ *n*≤*N*. μ (*En n*)) − *ennreal e*
by (*metis N ennreal-le-epsilon ennreal-less-top less-diff-eq-ennreal linorder-not-le*)
from *Hn*[*OF e*(*1*)] **obtain** *Hn* **where** *Hn*: $\bigwedge$*n*. *compactin X* (*Hn n*) $\bigwedge$*n*. *Hn n* ⊆ *En n*
$\bigwedge$*n*. μ (*En n*) < μ (*Hn n*) + *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*
by *blast*
have *y* < ($\sum$ *n*≤*N*. μ (*En n*)) − *ennreal e*
by *fact*
also have ... ≤ ($\sum$ *n*≤*N*. μ (*Hn n*) + *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*) − *ennreal e*
by(*intro ennreal-minus-mono sum-mono*) (*use Hn*(*3*) *order-less-le* **in** *auto*)
also have ... = ($\sum$ *n*≤*N*. μ (*Hn n*)) + ($\sum$ *n*≤*N*. *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*) − *ennreal e*
by (*simp add*: *sum.distrib*)
also have ... = μ ($\bigcup$*n*≤*N*. *Hn n*) + ($\sum$ *n*≤*N*. *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*) − *ennreal e*
proof −
have ($\sum$ *n*≤*N*. μ (*Hn n*)) = μ ($\bigcup$*n*≤*N*. *Hn n*)
proof(*induction N*)
case *ih*:(*Suc N′*)
show *?case* (**is** *?l* = *?r*)
proof −
have *?l* = μ ($\bigcup$ (*Hn* ‘ {..*N′*})) + μ (*Hn* (*Suc N′*))
by(*simp add*: *ih*)
also have ... = μ (($\bigcup$ (*Hn* ‘ {..*N′*})) ∪ *Hn* (*Suc N′*))
proof(*rule compacts*[*symmetric*])
show *disjnt* ($\bigcup$ (*Hn* ‘ {..*N′*})) (*Hn* (*Suc N′*))
using *En*(*2*) *Hn*(*2*) **unfolding** *disjoint-family-on-def disjnt-iff*
by (*metis Int-iff Suc-n-not-le-n UNIV-I UN-iff atMost-iff empty-iff in-mono*)
qed(*auto intro*!: *compactin-Union Hn*)
also have ... = *?r*
by (*simp add*: *Un-commute atMost-Suc*)
finally show *?thesis* **.**
qed
qed *simp*
thus *?thesis*
by *simp*

qed
also have ... ≤ μ (⋃n≤N. *Hn n*) + (∑n. *ennreal* ((*1* / *2*) ^ *Suc n*) ∗ *ennreal e*) − *ennreal e*
by(*intro ennreal-minus-mono add-mono sum-le-suminf*) (*use e* **in** *auto*)
also have ... = μ (⋃n≤N. *Hn n*) + (∑n. *ennreal* ((*1* / *2*) ^ *Suc n*)) ∗ *ennreal e* − *ennreal e*
using *ennreal-suminf-multc* **by** *presburger*
also have ... = μ (⋃n≤N. *Hn n*) + *ennreal e* − *ennreal e*
proof −
have (∑n. *ennreal* ((*1* / *2*) ^ *Suc n*)) = *ennreal 1*
by(*rule suminf-ennreal-eq*) (*use power-half-series* **in** *auto*)
thus *?thesis*
by *fastforce*
qed
also have ... = μ (⋃n≤N. *Hn n*)
by *simp*
finally show *Bex* (μ ‘ {*K*. *K* ⊆ ⋃ (*range En*) ∧ *compactin X K*}) ((<) *y*)
using *Hn* **by**(*auto intro*!: *exI*[**where** *x*=⋃n≤N. *Hn n*] *compactin-Union*)
qed
qed(*auto intro*!: *Sup-le-iff* [*THEN iffD2*] μ*-mono*)
moreover have (⋃ (*range En*)) ⊆ *topspace X*
using *En* **by**(*auto simp*: *Mf-def*)
ultimately show *?thesis*
using *that* **by**(*auto simp*: *Mf-def*)
qed
qed
have *step4'*: μ (*E1* ∪ *E2*) = μ *E1* + μ *E2* μ(*E1* ∪ *E2*) < ∞ ⟹ *E1* ∪ *E2* ∈ *Mf*
if *E*: *E1* ∈ *Mf* *E2* ∈ *Mf* *disjnt E1 E2* **for** *E1 E2*
proof −
define *En* **where** *En* ≡ (λn::*nat*. *if n* = *0 then E1 else if n* = *1 then E2 else* {})
have *1*: (⋃ (*range En*)) = (*E1* ∪ *E2*)
by(*auto simp*: *En-def*)
have *2*: (∑i. μ (*En i*)) = μ *E1* + μ *E2*
using *suminf-offset*[*of* λi. μ (*En i*),*of Suc* (*Suc 0*)]
by(*auto simp*: *En-def* μ*-empty*)
have *3*:*disjoint-family En*
using *E*(*3*) **by**(*auto simp*: *disjoint-family-on-def disjnt-def En-def*)
have *4*: ⋀n. *En n* ∈ *Mf*
using *E*(*1*,*2*) **by**(*auto simp*: *En-def empty-in-Mf*)
show μ (*E1* ∪ *E2*) = μ *E1* + μ *E2* μ(*E1* ∪ *E2*) < ∞ ⟹ *E1* ∪ *E2* ∈ *Mf*
using *step4*[*of En*] *E*(*1*) **by**(*simp-all add*: *1 2 3 4*)
qed

have *step5*: ∃ *V K*. *openin X V* ∧ *compactin X K* ∧ *K* ⊆ *E* ∧ *E* ⊆ *V* ∧ μ (*V* − *K*) < *ennreal e*
if *E*: *E* ∈ *Mf* **and** *e*: *e* > *0* **for** *E e*

proof−
  have *1*:μ *E* < μ *E* + *ennreal* (*e* / *2*)
    using *E e* **by**(*simp add*: *Mf-def*) (*metis* μ*-mono linorder-not-le*)
  hence *2*: μ *E* + *ennreal* (*e* / *2*) < μ *E* + *ennreal* (*e* / *2*) + *ennreal* (*e* / *2*)
    by *simp*
  from *Inf-le-iff*[*THEN iffD1*,*OF eq-refl*,*rule-format*,*OF* - *1*]
  obtain *V* **where** *V*: *openin X V E* ⊆ *V* μ *V* < μ *E* + *ennreal* (*e* / *2*)
    using μ*-def* μ*-open* **by** *force*
  have μ *E* + *ennreal* (*e* / *2*) + *ennreal* (*e* / *2*) ≤ (⨆*K*∈{*K*. *K* ⊆ *E* ∧ *compactin X K*}. μ *K* + *ennreal e*)
    by(*subst ennreal-SUP-add-left*,*insert E e*) (*auto simp*: *ennreal-plus-if Mf-def*)
  from *le-Sup-iff*[*THEN iffD1*,*OF this*,*rule-format*,*OF 2*]
  obtain *K* **where** *K*: *compactin X K K* ⊆ *E* μ *E* + *ennreal* (*e* / *2*) < μ *K* + *ennreal e*
    by *blast*
  have μ (*V* − *K*) < ∞
   by (*metis Diff-subset V*(*3*) μ*-mono dual-order.strict-trans1 infinity-ennreal-def order-le-less-trans top-greatest*)
  hence μ *K* + μ (*V* − *K*) = μ (*K* ∪ (*V* − *K*))
      by(*intro step4*′(*1*)[*symmetric*,*OF step2*(*1*)[*OF K*(*1*)] *step3*(*2*)] *openin-diff V*(*1*) *compactin-imp-closedin K*(*1*) *lh*(*2*))
      (*auto simp*: *disjnt-iff*)
  also have ... = μ *V*
    by (*metis Diff-partition K*(*2*) *V*(*2*) *order-trans*)
  also have ... < μ *K* + *ennreal e*
    by(*auto intro*!: *order.strict-trans*[*OF V*(*3*)] *K*)
  finally have μ (*V* − *K*) < *ennreal e*
    by(*simp add*: *ennreal-add-left-cancel-less*)
  thus *?thesis*
    using *V K* **by** *blast*
qed
have *step6*: ⋀*A B*. *A* ∈ *Mf* ⟹ *B* ∈ *Mf* ⟹ *A* − *B* ∈ *Mf* ⋀*A B*. *A* ∈ *Mf* ⟹ *B* ∈ *Mf* ⟹ *A* ∪ *B* ∈ *Mf*
  ⋀*A B*. *A* ∈ *Mf* ⟹ *B* ∈ *Mf* ⟹ *A* ∩ *B* ∈ *Mf*
proof −
  {
    fix *A B*
    assume *AB*: *A* ∈ *Mf B* ∈ *Mf*
    have *dif1*: μ (*A* − *B*) < ∞
       by (*metis* (*no-types*, *lifting*) *AB*(*1*) *Diff-subset Mf-def* μ*-mono infinity-ennreal-def mem-Collect-eq order-le-less-trans*)
    have μ (*A* − *B*) = (⨆ (μ ‘ {*K*. *K* ⊆ (*A* − *B*) ∧ *compactin X K*}))
    proof(*rule antisym*)
      show μ (*A* − *B*) ≤ ⨆ (μ ‘ {*K*. *K* ⊆ *A* − *B* ∧ *compactin X K*})
        unfolding *le-Sup-iff*
      proof *safe*
        fix *y*
        assume *y*:*y* < μ (*A* − *B*)
        then obtain *e* **where** *e*: *e* > *0* *ennreal e* = μ (*A* − *B*) − *y*

by (*metis dif1 diff-gt-0-iff-gt-ennreal diff-le-self-ennreal ennreal-cases ennreal-less-zero-iff neq-top-trans order-less-le*)
from *step5*[*OF AB*(*1*) *half-gt-zero*[*OF e*(*1*)]] *step5*[*OF AB*(*2*) *half-gt-zero*[*OF e*(*1*)]]
obtain *V1 V2 K1 K2* **where** *VK*:
openin X V1 compactin X K1 K1 ⊆ *A A* ⊆ *V1* μ (*V1* − *K1*) < *ennreal* (*e* / *2*)
openin X V2 compactin X K2 K2 ⊆ *B B* ⊆ *V2* μ (*V2* − *K2*) < *ennreal* (*e* / *2*)
by *auto*
have *K1V2*:*compactin X* (*K1* − *V2*)
by(*auto intro*!: *closed-compactin*[*OF VK*(*2*)] *compactin-imp-closedin*[*OF lh*(*2*) *VK*(*2*)] *VK*(*6*))
have μ (*A* − *B*) ≤ μ ((*K1* − *V2*) ∪ (*V1* − *K1*) ∪ (*V2* − *K2*))
using *VK* **by**(*auto intro*!: μ*-mono*)
also have ... ≤ μ ((*K1* − *V2*) ∪ (*V1* − *K1*)) + μ (*V2* − *K2*)
by *fact*
also have ... ≤ μ (*K1* − *V2*) + μ (*V1* − *K1*) + μ (*V2* − *K2*)
by(*auto intro*!: *step1'*)
also have ... < μ (*K1* − *V2*) + μ (*V1* − *K1*) + *ennreal* (*e* / *2*)
unfolding *add.assoc ennreal-add-left-cancel-less ennreal-add-left-cancel-less*
using *step2*(*1*)[*OF K1V2*] *VK*(*5*,*10*) *Mf-def* **by** *fastforce*
also have ... ≤ μ (*K1* − *V2*) + *ennreal* (*e* / *2*) + *ennreal* (*e* / *2*)
using *order.strict-implies-order*[*OF VK*(*5*)] **by**(*auto simp*: *add-mono*)
also have ... = μ (*K1* − *V2*) + *ennreal e*
using *e*(*1*) *ennreal-plus-if* **by** *auto*
finally have *1*:μ (*A* − *B*) < μ (*K1* − *V2*) + *ennreal e* **.**
show ∃ *a*∈(μ ‘ {*K*. *K* ⊆ *A* − *B* ∧ *compactin X K*}). (*y* < *a*)
proof(*safe intro*!: *bexI*[**where** *x*=μ (*K1* − *V2*)] *imageI*)
have *y* < μ (*K1* − *V2*) + *ennreal e* − *ennreal e*
by (*metis 1 add-diff-self-ennreal e*(*2*) *ennreal-less-top less-diff-eq-ennreal order-less-imp-le y*)
also have ... = μ (*K1* − *V2*)
by *simp*
finally show *y* < μ (*K1* − *V2*) **.**
qed(*use K1V2 VK* **in** *auto*)
qed
qed(*auto intro*!: μ*-mono simp*: *Sup-le-iff*)
with *dif1* **show** *A* − *B* ∈ *Mf*
using *Mf-def* μ*-fin-subset* **by** *auto*
}
note *diff*=*this*
fix *A B*
assume *AB*: *A* ∈ *Mf B* ∈ *Mf*
show *un*: *A* ∪ *B* ∈ *Mf*
proof −
have *A* ∪ *B* = (*A* − *B*) ∪ *B*
by *fastforce*
also have ... ∈ *Mf*

proof(*rule step4′*(*2*))
have μ ($A - B \cup B$) = μ ($A - B$) + μ B
by(*rule step4′*(*1*)) (*auto simp*: *diff AB disjnt-iff*)
also have ... < ∞
using *Mf-def diff*[*OF AB*] *AB*(*2*) **by** *fastforce*
finally show μ ($A - B \cup B$) < ∞ **.**
qed(*auto simp*: *diff AB disjnt-iff*)
finally show *?thesis* **.**
qed
show *int*: $A \cap B \in$ *Mf*
proof −
have $A \cap B = A - (A - B)$
by *blast*
also have ... ∈ *Mf*
by(*auto intro*!: *diff AB*)
finally show *?thesis* **.**
qed
qed
have *step6′*: ($\bigcup$ *i*∈*I*. *Ai i*) ∈ *Mf* **if** *finite I* ($\bigwedge$*i*. *i* ∈ *I* ⟹ *Ai i* ∈ *Mf*) **for** *Ai*
and *I* :: *nat set*
proof −
have (∀ *i*∈*I*. *Ai i* ∈ *Mf*) ⟶ ($\bigcup$ *i*∈*I*. *Ai i*) ∈ *Mf*
by(*rule finite-induct*[*OF that*(*1*)]) (*auto intro*!: *step6*(*2*) *empty-in-Mf*)
with *that* **show** *?thesis*
by *blast*
qed
have *step7*: *sigma-algebra* (*topspace X*) *M sets* (*borel-of X*) ⊆ *M*
proof −
show *sa*:*sigma-algebra* (*topspace X*) *M*
unfolding *sigma-algebra-iff2*
proof(*intro conjI ballI allI impI*)
show {} ∈ *M*
using *empty-in-Mf* **by**(*auto simp*: *M-def*)
next
show *M-subspace*:*M* ⊆ *Pow* (*topspace X*)
by(*auto simp*: *M-def*)
{
fix *s*
assume *s*:*s* ∈ *M*
show *topspace X* − *s* ∈ *M*
unfolding *M-def*
proof(*intro conjI CollectI allI impI*)
fix *K*
assume *K*: *compactin X K*
have (*topspace X* − *s*) ∩ *K* = *K* − (*s* ∩ *K*)
using *M-subspace s compactin-subset-topspace*[*OF K*] **by** *fast*
also have ... ∈ *Mf*
by(*intro step6*(*1*) *step2*(*1*)[*OF K*]) (*use s K M-def* **in** *blast*)
finally show (*topspace X* − *s*) ∩ *K* ∈ *Mf* **.**

```
  qed blast
}
{
  fix An :: nat ⇒ -
  assume An: range An ⊆ M
  show (⋃ (range An)) ∈ M
    unfolding M-def
  proof(intro CollectI conjI allI impI)
    fix K
    assume K: compactin X K
    have ∃ Bn. ∀ n. Bn n = (An n ∩ K) − (⋃ i<n. Bn i)
      by(rule dependent-wellorder-choice) auto
    then obtain Bn where Bn: ⋀n. Bn n = (An n ∩ K) − (⋃ i<n. Bn i)
      by blast
    have Bn-disj:disjoint-family Bn
      unfolding disjoint-family-on-def
    proof safe
      fix m n x
      assume h:m ≠ n x ∈ Bn m x ∈ Bn n
      then consider m < n | n < m
        by linarith
      then show x ∈ {}
      proof cases
        case 1
        with h(3) have x ∉ Bn m
          by(auto simp: Bn[of n])
        with h(2) show ?thesis by blast
      next
        case 2
        with h(2) have x ∉ Bn n
          by(auto simp: Bn[of m])
        with h(3) show ?thesis by blast
      qed
    qed
    have un:(⋃ (range An) ∩ K) = (⋃ n. Bn n)
    proof −
      have 1:An n ∩ K ⊆ (⋃ i≤n. Bn i) for n
      proof safe
        fix x
        assume x:x ∈ An n x ∈ K
        define m where m = (LEAST m. x ∈ An m)
        have m1:⋀l. l < m ⟹ x ∈ An m ⟹ x ∉ An l
          using m-def not-less-Least by blast
        hence x-nBn:l < m ⟹ x ∉ Bn l for l
          by (metis Bn Diff-Diff-Int Diff-iff m-def not-less-Least)
        have m2: x ∈ An m
          by (metis LeastI-ex x(1) m-def)
        have m3: m ≤ n
          using m1 m2 not-le-imp-less x(1) by blast
```

```
          have x ∈ Bn m
            unfolding Bn[of m]
            using x-nBn m2 x(2) by fast
          thus x ∈ ⋃ (Bn ` {..n})
            using m3 by blast
        qed
        have 2:(⋃n. An n ∩ K) = (⋃n. Bn n)
        proof(rule antisym)
          show (⋃n. An n ∩ K) ⊆ ⋃ (range Bn)
          proof safe
            fix n x
            assume x ∈ An n x ∈ K
            then have x ∈ (⋃i≤n. Bn i)
              using 1 by fast
            thus x ∈ ⋃ (range Bn)
              by fast
          qed
        next
          show ⋃ (range Bn) ⊆ (⋃n. An n ∩ K)
          proof(rule SUP-mono)
            show ∃ m∈UNIV. Bn i ⊆ An m ∩ K for i
              by(auto intro!: bexI[where x=i] simp: Bn[of i])
          qed
        qed
        thus ?thesis
          by simp
      qed
      also have ... ∈ Mf
      proof(safe intro!: step4(2) Bn-disj)
        fix n
        show Bn n ∈ Mf
        proof(rule less-induct)
          fix n
          show (⋀m. m < n ⟹ Bn m ∈ Mf) ⟹ Bn n ∈ Mf
            using An K by(auto intro!: step6′ step6(1) simp :Bn[of n] M-def)
        qed
      next
        have μ (⋃ (range Bn)) ≤ μ K
          unfolding un[symmetric] by(auto intro!: μ-mono)
        also have ... < ∞
          using step2(1)[OF K] by(auto simp: Mf-def)
        finally show μ (⋃ (range Bn)) < ∞ .
      qed
      finally show ⋃ (range An) ∩ K ∈ Mf .
    qed(use An M-def in auto)
  }
qed
show sets (borel-of X) ⊆ M
  unfolding sets-borel-of-closed
```

proof(*safe intro*!: *sigma-algebra.sigma-sets-subset*[*OF sa*])
fix *T*
assume *closedin X T*
then show $T \in M$
by (*simp add*: *Int-commute M-def closedin-subset compact-Int-closedin step2*(*1*))
qed
qed
have *step8*: $A \in Mf \longleftrightarrow A \in M \wedge \mu\ A < \infty$ **for** *A*
proof *safe*
assume *A*: $A \in Mf$
then have $A \subseteq$ *topspace X*
by(*auto simp*: *Mf-def*)
thus $A \in M$
by(*auto simp*: *M-def intro*!:*step6*(*3*)[*OF A step2*(*1*)])
show $\mu\ A < \infty$
using *A* **by**(*auto simp*: *Mf-def*)
next
assume *A*: $A \in M\ \mu\ A < \infty$
hence $A \subseteq$ *topspace X*
using *M-def* **by** *blast*
moreover have $\mu\ A = (\bigsqcup\ (\mu\ `\ \{K.\ K \subseteq A \wedge \mathit{compactin}\ X\ K\}))$
proof(*rule antisym*)
show $\mu\ A \le \bigsqcup\ (\mu\ `\ \{K.\ K \subseteq A \wedge \mathit{compactin}\ X\ K\})$
unfolding *le-Sup-iff*
proof *safe*
fix *y*
assume *y*:$y < \mu\ A$
then obtain *e* **where** *e*: $e > 0$ *ennreal* $e = \mu\ A - y$
by (*metis A*(*2*) *diff-gt-0-iff-gt-ennreal diff-le-self-ennreal ennreal-cases ennreal-less-zero-iff neq-top-trans order-less-le*)
obtain *U* **where** *U*: *openin X U* $A \subseteq U\ \mu\ U < \infty$
using *Inf-less-iff*[*THEN iffD1*,*OF A*(*2*)[*simplified* μ*-def*]] μ*-open* **by** *force*
from *step5*[*OF step3*(*2*)[*OF U*(*1*,*3*)] *half-gt-zero*[*OF e*(*1*)]]
obtain *V K* **where** *VK*:
openin X V compactin X K $K \subseteq U\ U \subseteq V\ \mu\ (V - K) <$ *ennreal* $(e\ /\ 2)$
by *blast*
have *AK*: $A \cap K \in Mf$
using *step2*(*1*) *VK*(*2*) *A* **by**(*auto simp*: *M-def*)
hence *e'*: $\mu\ (A \cap K) < \mu\ (A \cap K) +$ *ennreal* $(e\ /\ 2)$
by (*metis Diff-Diff-Int Diff-subset Int-commute U*(*3*) *VK*(*3*) *VK*(*5*) μ*-mono add.commute diff-gt-0-iff-gt-ennreal ennreal-add-diff-cancel infinity-ennreal-def order-le-less-trans top.not-eq-extremum zero-le*)
have $\mu\ (A \cap K) + \mathit{ennreal}\ (e\ /\ 2) = (\bigsqcup K{\in}\{L.\ L \subseteq (A \cap K) \wedge \mathit{compactin}\ X\ L\}.\ \mu\ K + \mathit{ennreal}\ (e\ /\ 2))$
by(*subst ennreal-SUP-add-left*) (*use AK Mf-def* **in** *auto*)
from *le-Sup-iff*[*THEN iffD1*,*OF this*[*THEN eq-refl*],*rule-format*,*OF e'*]
obtain *H* **where** *H*: *compactin X H* $H \subseteq A \cap K\ \mu\ (A \cap K) < \mu\ H +$ *ennreal* $(e\ /\ 2)$

```
        by blast
      show ∃a∈μ ` {K. K ⊆ A ∧ compactin X K}. y < a
      proof(safe intro!: bexI[where x=μ H] imageI H(1))
        have μ A ≤ μ ((A ∩ K) ∪ (V − K))
          using VK U by(auto intro!: μ-mono)
        also have ... ≤ μ (A ∩ K) + μ (V − K)
          by(auto intro!: step1'(1))
        also have ... < μ H + ennreal (e / 2) + ennreal (e / 2)
          using H(3) VK(5) add-strict-mono by blast
        also have ... = μ H + ennreal e
          using e(1) ennreal-plus-if by fastforce
        finally have 1: μ A < μ H + ennreal e .
        have y = μ A − ennreal e
          using A(2) diff-diff-ennreal e(2) y by fastforce
        also have ... < μ H + ennreal e − ennreal e
          using 1
              by (metis diff-le-self-ennreal e(2) ennreal-add-diff-cancel-right en-
nreal-less-top minus-less-iff-ennreal top-neq-ennreal)
        also have ... = μ H
          by simp
        finally show y < μ H .
      qed(use H in auto)
    qed
  qed(auto simp: Sup-le-iff intro!: μ-mono)
  ultimately show A ∈ Mf
    using A(2) Mf-def by auto
qed
define N where N ≡ measure-of (topspace X) M μ
have step9: measure-space (topspace X) M μ
  unfolding measure-space-def
proof safe
  show countably-additive M μ
    unfolding countably-additive-def
   by (metis Sup-upper UNIV-I μ-mono image-eqI image-subset-iff infinity-ennreal-def
linorder-not-less neq-top-trans step1 step4(1) step8)
qed(auto simp: step7 positive-def μ-empty)
have space-N: space N = topspace X and sets-N: sets N = M and emeasure-N:
A ∈ sets N ⟹ emeasure N A = μ A for A
proof −
  show space N = topspace X
    by (simp add: N-def space-measure-of-conv)
  show 1:sets N = M
    by (simp add: N-def sigma-algebra.sets-measure-of-eq step7(1))
  have ⋀x. x ∈ M ⟹ x ⊆ topspace X
    by(auto simp: M-def)
  thus A ∈ sets N ⟹ emeasure N A = μ A
      unfolding N-def using step9 by(auto intro!: emeasure-measure-of simp:
measure-space-def 1[simplified N-def])
qed
```

have *g1*:*subalgebra N* (*borel-of X*) (**is** *?g1*)
and *g2*:($\forall$ *A*$\in$*sets N*. *emeasure N A* = ($\bigsqcap$ *C*$\in${*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}. *emeasure N C*)) (**is** *?g2*)
and *g3*:($\forall$ *A*. *openin X A* $\longrightarrow$ *emeasure N A* = ($\bigsqcup$ *K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*)) (**is** *?g3*)
and *g4*:($\forall$ *A*$\in$*sets N*. *emeasure N A* < ∞ $\longrightarrow$ *emeasure N A* = ($\bigsqcup$ *K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*)) (**is** *?g4*)
and *g5*:($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* < ∞) (**is** *?g5*)
and *g6*:*complete-measure N* (**is** *?g6*)
proof −
have *1*: $\bigwedge$*P*. ($\bigwedge$*C*. *P C* $\Longrightarrow$ *C* $\in$ *sets N*) $\Longrightarrow$ *emeasure N* ‘ {*C*. *P C*} = μ ‘ {*C*. *P C*}
using *emeasure-N* **by** *auto*
show *?g1*
by(*auto simp*: *subalgebra-def sets-N space-N space-borel-of step7*)
show *?g2*
proof −
have *emeasure N* ‘ {*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*} = μ ‘ {*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*} **for** *A*
using *step7*(*2*) **by**(*auto intro*!: *1 simp*: *sets-N dest*: *borel-of-open*)
hence *emeasure N* ‘ {*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*} = μ' ‘ {*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*} **for** *A*
using *μ-open* **by** *auto*
thus *?thesis*
by(*simp add*: *emeasure-N sets-N μ-def*) (*metis* (*no-types*, *lifting*) *Collect-cong*)
qed
show *?g3*
by (*metis* (*no-types*, *lifting*) *1 borel-of-open emeasure-N sets-N step2*(*1*) *step3*(*1*) *step7*(*2*) *step8 subsetD*)
show *?g4*
proof *safe*
fix *A*
assume *A*[*measurable*]: *A* $\in$ *sets N emeasure N A* < ∞
then have *Mf*:*A* $\in$ *Mf*
by (*simp add*: *emeasure-N sets-N step8*)
have *emeasure N A* = μ *A*
by (*simp add*: *emeasure-N*)
also have ... = $\bigsqcup$ (μ ‘ {*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*})
using *Mf* **unfolding** *Mf-def* **by** *simp metis*
also have ... = $\bigsqcup$ (*emeasure N* ‘ {*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*})
using *emeasure-N sets-N step2*(*1*) *step8* **by** *auto*
finally show *emeasure N A* = $\bigsqcup$ (*emeasure N* ‘ {*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}) **.**
qed
show *?g5*
using *emeasure-N sets-N step2*(*1*) *step8* **by** *auto*
show *?g6*
proof

fix *A B*
assume *AB*:*B* ⊆ *A A* ∈ *null-sets N*
then have *μ A* = *0*
by (*metis emeasure-N null-setsD1 null-setsD2*)
hence *1*:*μ B* = *0*
using *μ-mono*[*OF AB*(*1*)] **by** *fastforce*
have *B* ∈ *Mf*
proof −
have *B* ⊆ *topspace X*
by (*metis AB gfp.leq-trans null-setsD2 sets.sets-into-space space-N*)
moreover have *μ B* = ⨆ (*μ* ‘ {*K*. *K* ⊆ *B* ∧ *compactin X K*})
proof(*rule antisym*)
show ⨆ (*μ* ‘ {*K*. *K* ⊆ *B* ∧ *compactin X K*}) ≤ *μ B*
by(*auto simp*: *Sup-le-iff μ-mono*)
qed(*simp add*: *1*)
moreover have *μ B* < ⊤
by(*simp add*: *1*)
ultimately show *?thesis*
unfolding *Mf-def* **by** *blast*
qed
thus *B* ∈ *sets N*
by(*simp add*: *step8 sets-N*)
qed
qed

have *g7*: (∀ *f*. *?iscont f* ⟶ *?csupp f* ⟶ *integrable N f*)
unfolding *integrable-iff-bounded*
proof *safe*
fix *f*
assume *f*:*?iscont f ?csupp f*
then show [*measurable*]:*f* ∈ *borel-measurable N*
by(*auto intro*!: *measurable-from-subalg*[*OF g1*]
simp: *lower-semicontinuous-map-measurable upper-lower-semicontinuous-map-iff-continuous-map*)
let *?K* = *X closure-of* {*x*∈*topspace X*. *f x* ≠ *0*}
have *K*[*measurable*]: *compactin X ?K ?K* ∈ *sets N*
using *f*(*2*) *g1 sets-N step2*(*1*) *step8* **by**(*auto simp*: *has-compact-support-on-iff subalgebra-def*)
have *bounded* (*f* ‘ *?K*)
using *image-compactin*[*of X ?K euclideanreal f*] *f*
by(*auto simp*: *has-compact-support-on-iff intro*!: *compact-imp-bounded*)
then obtain *B* **where** *B*:⋀*x*. *x* ∈ *?K* ⟹ |*f x*| ≤ *B*
by (*meson bounded-real imageI*)
show (∫⁺ *x*. *ennreal* (*norm* (*f x*)) ∂*N*) < ∞
proof −
have (∫⁺ *x*. *ennreal* (*norm* (*f x*)) ∂*N*) ≤ (∫⁺ *x*. *ennreal* (*indicator ?K x* ∗|*f x*|) ∂*N*)
using *in-closure-of* **by**(*fastforce intro*!: *nn-integral-mono simp*: *indicator-def space-N*)
also have ... ≤ (∫⁺ *x*. *ennreal* (*B* ∗ *indicator ?K x*) ∂*N*)

```
        using B by(auto intro!: nn-integral-mono ennreal-leI simp: indicator-def)
      also have ... = (∫+ x. ennreal B ∗ indicator ?K x ∂N)
        by(auto intro!: nn-integral-cong simp: indicator-def)
      also have ... = ennreal B ∗ (∫+ x. indicator ?K x ∂N)
        by(simp add: nn-integral-cmult)
      also have ... = ennreal B ∗ emeasure N ?K
        by simp
      finally show ?thesis
       using g5 K(1) ennreal-mult-less-top linorder-not-le top.not-eq-extremum by
fastforce
    qed
  qed
  have g8: ∀f. ?iscont f ⟶ ?csupp f ⟶ φ (λx∈topspace X. f x) = (∫ x. f x ∂N)
  proof safe
    have 1: φ (λx∈topspace X. f x) ≤ (∫ x. f x ∂N) if f:?iscont f ?csupp f for f
    proof −
      let ?K = X closure-of {x∈topspace X. f x ≠ 0}
      have K[measurable]: compactin X ?K ?K ∈ sets N
      using f(2) g1 sets-N step2(1) step8 by(auto simp: has-compact-support-on-iff
subalgebra-def)
      have f-meas[measurable]: f ∈ borel-measurable N
        using f by(auto intro!: measurable-from-subalg[OF g1]
         simp: lower-semicontinuous-map-measurable upper-lower-semicontinuous-map-iff-continuous-map)
      have bounded (f ‘ ?K)
        using image-compactin[of X ?K euclideanreal f] f
        by(auto simp: has-compact-support-on-iff intro!: compact-imp-bounded)
      then obtain B′ where B′:⋀x. x ∈ ?K ⟹ |f x| ≤ B′
        by (meson bounded-real imageI)
      define B where B ≡ max 1 B′
      have B-pos: B > 0 and B: ⋀x. x ∈ ?K ⟹ |f x| ≤ B
        using B′ by(auto simp add: B-def intro!: max.coboundedI2)
     have 1:φ (λx∈topspace X. f x) ≤ (∫ x. f x ∂N) +  1 / (Suc n) ∗ (2 ∗ measure
N ?K + (1 / Suc n) + 2 ∗ B + 1) for n
      proof −
       have ∃yn. ∀m::nat. yn m = (if m = 0 then − B − 1 else  1 / 2 ∗ 1 / Suc
n + yn (m − 1))
          by(rule dependent-wellorder-choice) auto
        then obtain yn′ where yn′:⋀m::nat. yn′ m = (if m = 0 then − B − 1
else  1 / 2 ∗ 1 / Suc n + yn′ (m − 1))
          by blast
        hence yn′-0: yn′ 0 = − B − 1 and yn′-Suc: ⋀m. yn′ (Suc m) = 1 / 2 ∗
1 / Suc n + yn′ m
          by simp-all
        have yn′-accum: yn′ m = m ∗ (1 / 2 ∗ 1 / Suc n) + yn′ 0 for m
          by(induction m) (auto simp: yn′-Suc add-divide-distrib)

        define L :: nat where L = (LEAST k. B ≤ yn′ k)
        define yn where yn ≡ (λn. if n = L then B else yn′ n)
        have L-least: ⋀i. i < L ⟹ yn′ i < B
```

by (*metis L-def linorder-not-less not-less-Least*)
have *yn-L*: *yn L* = *B*
by(*auto simp*: *yn-def*)
have *yn′-L*: *yn′ L* ≥ *B*
unfolding *L-def*
proof(*rule LeastI-ex*)
show ∃ *x*. *B* ≤ *yn′ x*
proof(*safe intro*!: *exI*[**where** *x*=*nat* (*ceiling* ((*2* ∗ *B* + *2*) / ((*1*/*2*) ∗ *1* / *real* (*Suc n*))))])
have *B* ≤ *2* ∗ *B* + *2* + (− *B* − *1*)
using *B-pos* **by** *fastforce*
also have ... = (*2* ∗ *B* + *2*) / ((*1*/*2*) ∗ *1* / *real* (*Suc n*)) ∗ (*1* / *2* ∗ *1* / *Suc n*) + *yn′ 0*
by(*auto simp*: *yn′-0*)
also have ... ≤ *real* (*nat* (*ceiling* ((*2* ∗ *B* + *2*) / ((*1*/*2*) ∗ *1* / *real* (*Suc n*))))) ∗ (*1* / *2* ∗ *1* / *Suc n*) + *yn′ 0*
by(*intro add-mono real-nat-ceiling-ge mult-right-mono*) *auto*
also have ... = *yn′* (*nat* (*ceiling* ((*2* ∗ *B* + *2*) / ((*1*/*2*) ∗ *1* / *real* (*Suc n*)))))
by (*metis yn′-accum*)
finally show *B* ≤ *yn′* (*nat* ⌈(*2* ∗ *B* + *2*) / (*1* / *2* ∗ *1* / *real* (*Suc n*))⌉)
.
qed
qed
have *L-pos*: *0* < *L*
proof(*rule ccontr*)
assume ¬ *0* < *L*
then have [*simp*]:*L* = *0*
by *blast*
show *False*
using *yn′-L yn′-0 B-pos* **by** *auto*
qed
have *yn-0*: *yn 0* = − *B* − *1*
using *L-pos* **by**(*auto simp*: *yn-def yn′-0*)
have *strict-mono-yn*:*strict-mono yn*
proof(*rule strict-monoI-Suc*)
fix *m*
consider *m* = *L* | *Suc m* = *L* | *m* < *L Suc m* < *L* | *L* < *m L* < *Suc m*
by *linarith*
then show *yn m* < *yn* (*Suc m*)
proof *cases*
case *1*
then have *yn m* = *B*
by(*simp add*: *yn-L*)
also have ... ≤ *yn′ m*
using *yn′-L* **by**(*simp add*: *1*)
also have ... < *yn′* (*Suc m*)
by (*simp add*: *yn′-Suc*)
also have ... = *yn* (*Suc m*)

using *1* **by**(*auto simp*: *yn-def*)
finally show *?thesis* **.**
next
case *2*
then have *yn m* = *yn′ m*
using *yn-def* **by** *force*
also have ... < *B*
using *L-least*[*of m*] *2* **by** *blast*
also have ... = *yn* (*Suc m*)
by(*simp add*: *2 yn-L*)
finally show *?thesis* **.**
qed(*auto simp*: *yn-def yn′-Suc*)
qed
have *yn-le-L*: ⋀*i*. *i* ≤ *L* ⟹ *yn i* ≤ *B*
using *L-least less-eq-real-def yn-def* **by** *auto*
have *yn-ge-L*: ⋀*i*. *L* < *i* ⟹ *B* < *yn i*
using *strict-mono-yn*[*THEN strict-monoD*] *yn-L* **by** *blast*
have *yn-ge*: ⋀*i*. − *B* − *1* ≤ *yn i*
using *monoD*[*OF strict-mono-mono*[*OF strict-mono-yn*],*of 0*] *yn-0* **by** *auto*
have *yn-Suc-le*: *yn* (*Suc i*) < *1* / *real* (*Suc n*) + *yn i* **for** *i*
proof −
consider *i* = *L* | *Suc i* = *L* | *i* < *L Suc i* < *L* | *L* < *i L* < *Suc i*
by *linarith*
then show *?thesis*
proof *cases*
case *1*
then have *yn* (*Suc i*) = *yn′* (*Suc L*)
by(*simp add*: *yn-def*)
also have ... = *1* / *2* ∗ *1* / *Suc n* + *yn′ L*
by(*simp add*: *yn′-Suc*)
also have ... = (*1* / *2*) ∗ (*1* / *Suc n*) + (*1* / *2*) ∗ (*1* / *Suc n*) + *yn′* (*L* − *1*)
using *L-pos yn′* **by** *fastforce*
also have ... = *1* / *Suc n* + *yn′* (*L* − *1*)
unfolding *semiring-normalization-rules*(*1*) **by** *simp*
also have ... < *1* / *Suc n* + *B*
by (*simp add*: *L-least L-pos less-eq-real-def*)
finally show *?thesis*
by(*simp add*: *1 yn-L*)
next
case *2*
then have *yn* (*Suc i*) = *B*
by(*simp add*: *yn-L*)
also have ... ≤ *yn′ L*
using *yn′-L* **.**
also have ... = *1* / *2* ∗ *1* / *Suc n* + *yn′* (*L* − *1*)
using *yn′ L-pos* **by** *simp*
also have ... = *1* / *2* ∗ *1* / *Suc n* + *yn i*

using *2 yn-def* **by** *force*
also have ... $<$ *1* / *Suc n* + *yn i*
by (*simp add*: *pos-less-divide-eq*)
finally show *?thesis* **.**
qed(*auto simp*: *yn-def yn'-Suc pos-less-divide-eq*)
qed

have *f-bound*: *f x* ∈ {*yn 0*<..*yn L*} **if** *x*:*x* ∈ *?K* **for** *x*
using *B*[*OF x*] *yn-L yn-0* **by** *auto*
define *En* **where** *En* ≡ (λ*m*. {*x*∈*topspace X*. *yn m* < *f x* ∧ *f x* ≤ *yn* (*Suc m*)} ∩ *?K*)
have *En-sets*[*measurable*]: *En m* ∈ *sets N* **for** *m*
proof −
have {*x*∈*topspace X*. *yn m* < *f x* ∧ *f x* ≤ *yn* (*Suc m*)} = *f* −‘ {*yn m*<..*yn* (*Suc m*)} ∩ *space N*
by(*auto simp*: *space-N*)
also have ... ∈ *sets N*
by *simp*
finally show *?thesis*
by(*simp add*: *En-def*)
qed
have *En-disjnt*: *disjoint-family En*
unfolding *disjoint-family-on-def*
proof *safe*
fix *m n x*
assume *m* ≠ *n* **and** *x*: *x* ∈ *En n x* ∈ *En m*
then consider *m* < *n* | *n* < *m*
by *linarith*
thus *x* ∈ {}
proof *cases*
case *1*
hence *1*:*Suc m* ≤ *n*
by *simp*
from *x* **have** *f x* ≤ *yn* (*Suc m*) *yn n* < *f x*
by(*auto simp*: *En-def*)
with *1* **show** *?thesis*
using *monoD*[*OF strict-mono-mono*[*OF strict-mono-yn*] *1*] **by** *linarith*
next
case *2*
hence *1*:*Suc n* ≤ *m*
by *simp*
from *x* **have** *f x* ≤ *yn* (*Suc n*) *yn m* < *f x*
by(*auto simp*: *En-def*)
with *1* **show** *?thesis*
using *monoD*[*OF strict-mono-mono*[*OF strict-mono-yn*] *1*] **by** *linarith*
qed
qed
have *K-eq-un-En*: *?K* = (⋃ *i*≤*L*. *En i*)
proof *safe*

fix *x*
assume *x*:*x* ∈ *?K*
have ∃ *m*∈{..*L*}. *yn m* < *f x* ∧ *x* ∈ *topspace X* ∧ *f x* ≤ *yn* (*Suc m*)
proof(*rule ccontr*)
assume ¬ (∃ *m*∈{..*L*}. *yn m* < *f x* ∧ *x* ∈ *topspace X* ∧ *f x* ≤ *yn* (*Suc m*))
then have *1*:⋀*m*. *m* ≤ *L* ⟹ *yn* (*Suc m*) < *f x* ∨ *f x* ≤ *yn m*
using *compactin-subset-topspace*[*OF K*(*1*)] *x* **by** *force*
then have *m* ≤ *L* ⟹ *yn* (*Suc m*) < *f x* **for** *m*
by(*induction m*) (*use B x yn-0* **in** *fastforce*)+
hence *yn* (*Suc L*) < *f x*
by *force*
with *yn-ge-L*[*of Suc L*] *f-bound x B* **show** *False*
by *fastforce*
qed
thus *x* ∈ (⋃*i*≤*L*. *En i*)
using *x* **by**(*auto simp*: *En-def*)
qed(*auto simp*: *En-def*)
have *emeasure-En-fin*: *emeasure N* (*En i*) < ∞ **for** *i*
proof −
have *emeasure N* (*En i*) ≤ *μ ?K*
unfolding *emeasure-N*[*OF En-sets*[*of i*]] **by**(*auto intro*!: *μ-mono simp*: *En-def*)
also have ... < ∞
using *step2*(*1*)[*OF K*(*1*)] *step8* **by** *blast*
finally show *?thesis* **.**
qed
have ∃ *Vi*. *openin X Vi* ∧ *En i* ⊆ *Vi* ∧ *measure N Vi* < *measure N* (*En i*) + (*1* / *Suc n*) / *Suc L* ∧
(∀ *x*∈*Vi*. *f x* < (*1* / *real* (*Suc n*) + *yn i*)) ∧ *emeasure N Vi* < ∞ **for** *i*
proof −
have *1*:*emeasure N* (*En i*) < *emeasure N* (*En i*) + *ennreal* (*1* / *real* (*Suc n*) / *real* (*Suc L*))
unfolding *ennreal-add-left-cancel-less*[**where** *b*=*0*,*simplified add-0-right*]
using *emeasure-En-fin* **by** (*simp add*: *order-less-le*)
from *Inf-le-iff*[*THEN iffD1*,*OF eq-refl*[*OF g2*[*rule-format*,*OF En-sets*[*of i*],*symmetric*]],*rule-format*,*OF this*]
obtain *Vi* **where** *Vi*:*openin X Vi Vi* ⊇ *En i*
emeasure N Vi < *emeasure N* (*En i*) + *ennreal* (*1* / *real* (*Suc n*) / *real* (*Suc L*))
by *blast*
hence *ennreal* (*measure N Vi*) = *emeasure N Vi*
unfolding *measure-def* **using** *ennreal-enn2real-if* **by** *fastforce*
also have ... < *ennreal* (*measure N* (*En i*)) + *ennreal* (*1* / *real* (*Suc n*) / *real* (*Suc L*))
using *ennreal-enn2real-if emeasure-En-fin Vi* **by** (*metis emeasure-eq-ennreal-measure top.extremum-strict*)
also have ... = *ennreal* (*measure N* (*En i*) + *1* / *real* (*Suc n*) / *real* (*Suc*

L))
by *simp*
finally have *1*:*measure N Vi* < *measure N* (*En i*) + *1* / *real* (*Suc n*) / *real* (*Suc L*)
by(*auto intro*!: *ennreal-less-iff* [*THEN iffD1*])
define *Vi′* **where** *Vi′* = *Vi* ∩ {*x*∈*topspace X*. *yn i* < *f x* ∧ *f x* < *1* / *real* (*Suc n*) + *yn i*}
have *En i* ⊆ *Vi′*
proof −
have *En i* = *En i* ∩ {*x*∈*topspace X*. *yn i* < *f x* ∧ *f x* < *1* / *real* (*Suc n*) + *yn i*}
unfolding *En-def* **using** *order.strict-trans1* [*OF - yn-Suc-le*] **by** *fast*
also have ... ⊆ *Vi′*
using *Vi*(*2*) **by**(*auto simp*: *Vi′-def*)
finally show *?thesis* **.**
qed
moreover have *openin X Vi′*
proof −
have {*x* ∈ *topspace X*. *yn i*<*f x* ∧ *f x*< *1*/*real* (*Suc n*) + *yn i*} = (*f* −‘ {*yn i*<..<*1*/*real* (*Suc n*) + *yn i*} ∩ *topspace X*)
by *fastforce*
also have *openin X* ...
using *continuous-map-open*[*OF f*(*1*)] **by** *simp*
finally show *?thesis*
using *Vi*(*1*) **by**(*auto simp*: *Vi′-def*)
qed
moreover have *measure N Vi′* < *measure N* (*En i*) + (*1* / *real* (*Suc n*) / *real* (*Suc L*)) (**is** *?l* < *?r*)
proof −
have *?l* ≤ *measure N Vi*
unfolding *measure-def*
proof(*safe intro*!: *enn2real-mono emeasure-mono*)
show *Vi* ∈ *sets N*
using *Vi*(*1*) *borel-of-open sets-N step7*(*2*) **by** *blast*
show *emeasure N Vi* < ⊤
by (*metis* ‹*ennreal* (*Sigma-Algebra.measure N Vi*) = *emeasure N Vi*› *ennreal-less-top*)
qed(*auto simp*: *Vi′-def*)
with *1* **show** *?thesis*
by *fastforce*
qed
moreover have ⋀*x*. *x* ∈ *Vi′* ⟹ *f x* < (*1* / *real* (*Suc n*) + *yn i*)
by(*auto simp*: *Vi′-def*)
moreover have *emeasure N Vi′* < ∞
by (*metis* (*no-types*, *lifting*) *Diff-Diff-Int Diff-subset Vi′-def Vi*(*1*) ‹*ennreal* (*measure N Vi*) = *emeasure N Vi*› *borel-of-open emeasure-mono ennreal-less-top infinity-ennreal-def linorder-not-less sets-N step7*(*2*) *subsetD top.not-eq-extremum*)
ultimately show *?thesis*

by *blast*
qed
then obtain *Vi* **where**
Vi: $\bigwedge i.$ *openin X* (*Vi i*) $\bigwedge i.$ *En i* $\subseteq$ *Vi i*
$\bigwedge i.$ *measure N* (*Vi i*) < *measure N* (*En i*) + (*1* / *Suc n*) / *Suc L*
$\bigwedge i\ x.$ $x \in$ *Vi i* $\Longrightarrow$ *f x* < (*1* / *real* (*Suc n*) + *yn i*)
$\bigwedge i.$ *emeasure N* (*Vi i*) < ∞
by *metis*
have *?K* $\subseteq$ ($\bigcup$ *i*$\leq$*L*. *Vi i*)
using *K-eq-un-En Vi*(*2*) **by** *blast*
from *fApartition*[*OF K*(*1*) *Vi*(*1*) *this*]
obtain *hi* **where** *hi*: $\bigwedge i.$ *i* $\leq$ *L* $\Longrightarrow$ *?iscont* (*hi i*) $\bigwedge i.$ *i* $\leq$ *L* $\Longrightarrow$ *?csupp* (*hi i*)
$\bigwedge i.$ *i* $\leq$ *L* $\Longrightarrow$ *X closure-of* {$x \in$ *topspace X*. *hi i x* $\neq$ *0*} $\subseteq$ *Vi i*
$\bigwedge i.$ *i* $\leq$ *L* $\Longrightarrow$ *hi i* $\in$ *topspace X* $\to$ {*0..1*} $\bigwedge i.$ *i* $\leq$ *L* $\Longrightarrow$ *hi i* $\in$ *topspace X* − *Vi i* $\to$ {*0*}
$\bigwedge x.$ $x \in$ *?K* $\Longrightarrow$ ($\sum$ *i*$\leq$*L*. *hi i x*) = *1* $\bigwedge x.$ $x \in$ *topspace X* $\Longrightarrow$ *0* $\leq$ ($\sum$ *i*$\leq$*L*. *hi i x*)
$\bigwedge x.$ $x \in$ *topspace X* $\Longrightarrow$ ($\sum$ *i*$\leq$*L*. *hi i x*) $\leq$ *1*
by *blast*
have *f-sum-hif*: ($\sum$ *i*$\leq$*L*. *f x* ∗ *hi i x*) = *f x* **if** *x*:$x \in$ *topspace X* **for** *x*
proof(*cases f x* = *0*)
case *False*
then have $x \in$ *?K*
using *in-closure-of x* **by** *fast*
with *hi*(*6*)[*OF this*] **show** *?thesis*
by(*simp add*: *sum-distrib-left*[*symmetric*])
qed *simp*
have *sum-muEi*:($\sum$ *i*$\leq$*L*. *measure N* (*En i*)) = *measure N ?K*
proof −
have ($\sum$ *i*$\leq$*L*. *measure N* (*En i*)) = *measure N* ($\bigcup$ *i*$\leq$*L*. *En i*)
using *emeasure-En-fin En-disjnt*
by(*fastforce intro*!: *measure-UNION'*[*symmetric*] *fmeasurableI pairwiseI simp*: *disjnt-iff disjoint-family-on-def*)
also have ... = *measure N ?K*
by(*simp add*: *K-eq-un-En*)
finally show *?thesis* **.**
qed
have *measure-K-le*: *measure N ?K* $\leq$ ($\sum$ *i*$\leq$*L*. φ ($\lambda x \in$*topspace X*. *hi i x*))
proof −
have *ennreal* (*measure N ?K*) = μ *?K*
by (*metis* (*mono-tags, lifting*) *K*(*1*) *K*(*2*) *Sigma-Algebra.measure-def emeasure-N ennreal-enn2real g5 infinity-ennreal-def*)
also have μ *?K* $\leq$ *ennreal* (φ ($\lambda x \in$*topspace X*. $\sum$ *i*$\leq$*L*. *hi i x*))
by(*auto intro*!: *le-Inf-iff*[*THEN iffD1*,*OF eq-refl*[*OF step2*(*2*)[*OF K*(*1*)]],*rule-format*]
imageI exI[**where** *x*=$\lambda x.$ $\sum$ *i*$\leq$*L*. *hi i x*] *has-compact-support-on-sum hi continuous-map-sum*)
also have ... = *ennreal* ($\sum$ *i*$\leq$*L*. φ ($\lambda x \in$*topspace X*. *hi i x*))

by(*auto intro*!: *pos-lin-functional-on-CX-sum assms ennreal-cong hi*)
finally show *?thesis*
using *Pi-mem*[*OF hi*(*4*)] **by**(*auto intro*!: *ennreal-le-iff*[*of - measure N ?K*,*THEN iffD1*] *sum-nonneg pos hi*)
qed
have φ (*restrict f* (*topspace X*)) = φ ($\lambda x \in$*topspace X*. $\sum i \leq L$. *f x* $*$ *hi i x*)
using *f-sum-hif restrict-ext* **by** *force*
also have ... = ($\sum i \leq L$. φ ($\lambda x \in$*topspace X*. *f x* $*$ *hi i x*))
using *f hi* **by**(*auto intro*!: *pos-lin-functional-on-CX-sum assms has-compact-support-on-mult-right*)
also have ... $\leq$ ($\sum i \leq L$. φ ($\lambda x \in$*topspace X*. (*1* / (*Suc n*) + *yn i*) $*$ *hi i x*))
proof(*safe intro*!: *sum-mono* φ*mono*)
fix *i x*
assume *i*:*i* $\leq$ *L x* $\in$ *topspace X*
show *f x* $*$ *hi i x* $\leq$ (*1* / (*Suc n*) + *yn i*) $*$ *hi i x*
proof(*cases x* $\in$ *Vi i*)
case *True*
hence *f x* < *1* / (*Suc n*) + *yn i*
by *fact*
thus *?thesis*
using *Pi-mem*[*OF hi*(*4*)[*OF i*(*1*)] *i*(*2*)] **by**(*intro mult-right-mono*) *auto*
next
case *False*
then show *?thesis*
using *Pi-mem*[*OF hi*(*5*)[*OF i*(*1*)]] *i*(*2*) **by** *force*
qed
qed(*auto intro*!: *f hi has-compact-support-on-mult-left*)
also have ... = ($\sum i \leq L$. (*1* / (*Suc n*) + *yn i*) $*$ φ ($\lambda x \in$*topspace X*. *hi i x*))
by(*intro Finite-Cartesian-Product.sum-cong-aux linear hi*) *auto*
also have ... = ($\sum i \leq L$. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) $*$ φ ($\lambda x \in$*topspace X*. *hi i x*))
− ($\sum i \leq L$. (*B* + *1*) $*$ φ ($\lambda x \in$*topspace X*. *hi i x*))
by(*simp add*: *sum-subtractf*[*symmetric*] *distrib-right*)
also have ... = ($\sum i \leq L$. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) $*$ φ ($\lambda x \in$*topspace X*. *hi i x*))
− (*B* + *1*) $*$ ($\sum i \leq L$. φ ($\lambda x \in$*topspace X*. *hi i x*))
by (*simp add*: *sum-distrib-left*)
also have ... $\leq$ ($\sum i \leq L$. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) $*$ (*measure N* (*En i*) + (*1* / *Suc n* / *Suc L*)))
− (*B* + *1*) $*$ *measure N ?K*
proof(*safe intro*!: *diff-mono*[*OF sum-mono*[*OF mult-left-mono*]])
fix *i*
assume *i*: *i* $\leq$ *L*
show φ (*restrict* (*hi i*) (*topspace X*)) $\leq$ *measure N* (*En i*) + *1* / (*Suc n*) / (*Suc L*) (**is** *?l* $\leq$ *?r*)
proof −
have *?l* $\leq$ *measure N* (*Vi i*)
proof −
have *ennreal* (φ (*restrict* (*hi i*) (*topspace X*))) $\leq$ μ' (*Vi i*)
using *hi*(*1*,*2*,*3*,*4*,*5*)[*OF i*] **by**(*auto intro*!: *SUP-upper imageI exI*[**where**

x=*hi i*] *simp*: μ'-*def*)
also have ... = *emeasure N* (*Vi i*)
by (*metis Vi*(*1*) μ-*open borel-of-open emeasure-N sets-N step7*(*2*) *subsetD*)
also have ... = *ennreal* (*measure N* (*Vi i*))
using *Vi*(*5*)[*of i*] **by**(*auto simp*: *measure-def intro*!: *ennreal-enn2real*[*symmetric*])
finally show φ (*restrict* (*hi i*) (*topspace X*)) $\leq$ *measure N* (*Vi i*)
using *ennreal-le-iff measure-nonneg* **by** *blast*
qed
with *Vi*(*3*)[*of i*] **show** *?thesis*
by *linarith*
qed
show *0* $\leq$ *1* / *real* (*Suc n*) + *yn i* + (*B* + *1*)
using *yn-ge*[*of i*] **by**(*simp add*: *add.assoc*)
qed(*use B-pos measure-K-le* **in** *fastforce*)
also have ... = ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) ∗ *measure N* (*En i*)) + *2* ∗ ($\sum$ *i*$\leq$*L*. ((*1* / *Suc n*)) ∗ *measure N* (*En i*))
+ ($\sum$ *i*$\leq$*L*. (*B* + *1*) ∗ *measure N* (*En i*))
+ ($\sum$ *i*$\leq$*L*. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) ∗ (*1* / *Suc n* / *Suc L*)) − (*B* + *1*) ∗ *measure N ?K*
by(*simp add*: *distrib-left distrib-right sum.distrib sum-subtractf left-diff-distrib*)
also have ... = ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) ∗ *measure N* (*En i*)) + *1* / *Suc n* ∗ *2*∗ *measure N ?K*
+ ($\sum$ *i*$\leq$*L*. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) ∗ (*1* / *Suc n* / *Suc L*))
by(*simp add*: *sum-distrib-left*[*symmetric*] *sum-muEi del*: *times-divide-eq-left*)
also have ... $\leq$ ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) ∗ *measure N* (*En i*)) + *1* / *Suc n* ∗ *2*∗ *measure N ?K*
+ ($\sum$ *i*$\leq$*L*. (*1* / (*Suc n*) + *B* + (*B* + *1*)) ∗ (*1* / *Suc n* / *Suc L*))
proof −
have ($\sum$ *i*$\leq$*L*. (*1* / (*Suc n*) + *yn i* + (*B* + *1*)) ∗ (*1* / *Suc n* / *Suc L*))
$\leq$ ($\sum$ *i*$\leq$*L*. (*1* / (*Suc n*) + *B* + (*B* + *1*)) ∗ (*1* / *Suc n* / *Suc L*))
proof(*safe intro*!: *sum-mono mult-right-mono*)
fix *i*
assume *i*: *i* $\leq$ *L*
show *1* / (*Suc n*) + *yn i* + (*B* + *1*) $\leq$ *1* / (*Suc n*) + *B* + (*B* + *1*)
using *yn-le-L*[*OF i*] **by** *fastforce*
qed *auto*
thus *?thesis*
by *argo*
qed
also have ... = ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) ∗ *measure N* (*En i*)) + *1* / *Suc n* ∗ *2*∗ *measure N ?K*
+ (*1* / (*Suc n*) + *B* + (*B* + *1*)) ∗ (*1* / *Suc n*)
by *simp*
also have ... = ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) ∗ *measure N* (*En i*))
+ *1* / *Suc n* ∗ (*2* ∗ *measure N ?K* + (*1* / *Suc n*) + *2* ∗ *B* + *1*)

by *argo*
also have ... $\leq$ ($\int$ *x*. *f x* ∂*N*) + *1* / (*Suc n*) $*$ (*2* $*$ *measure N ?K* + (*1* / *Suc n*) + *2* $*$ *B* + *1*)
proof −
have ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) $*$ *measure N* (*En i*)) $\leq$ ($\int$ *x*. *f x* ∂*N*) (**is** *?l* $\leq$ *?r*)
proof −
have *?l* = ($\sum$ *i*$\leq$*L*. ($\int$ *x*. (*yn i* − *1* / (*Suc n*)) $*$ *indicator* (*En i*) *x* ∂*N*))
by *simp*
also have ... = ($\int$ *x*. ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / (*Suc n*)) $*$ *indicator* (*En i*) *x*) ∂*N*)
by(*rule Bochner-Integration.integral-sum*[*symmetric*]) (*use emeasure-En-fin* **in** *simp*)
also have ... $\leq$ *?r*
proof(*rule integral-mono*)
fix *x*
assume *x*: *x* $\in$ *space N*
consider $\bigwedge$*i*. *i* $\leq$ *L* $\Longrightarrow$ *x* $\notin$ *En i* | $\exists$ *i*$\leq$*L*. *x* $\in$ *En i*
by *blast*
then show ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / *real* (*Suc n*)) $*$ *indicat-real* (*En i*) *x*) $\leq$ *f x*
proof *cases*
case *1*
then have *x* $\notin$ *?K*
by(*simp add*: *K-eq-un-En*)
hence *f x* = *0*
using *x in-closure-of* **by**(*fastforce simp*: *space-N*)
with *1* **show** *?thesis*
by *force*
next
case *2*
then obtain *i* **where** *i*: *i* $\leq$ *L x* $\in$ *En i*
by *blast*
with *En-disjnt* **have** $\bigwedge$*j*. *j* $\neq$ *i* $\Longrightarrow$ *x* $\notin$ *En j*
by(*auto simp*: *disjoint-family-on-def*)
hence ($\sum$ *i*$\leq$*L*. (*yn i* − *1* / *real* (*Suc n*)) $*$ *indicat-real* (*En i*) *x*)
= ($\sum$ *j*$\leq$*L*. *if j* = *i then* (*yn i* − *1* / *real* (*Suc n*)) *else 0*)
by(*intro Finite-Cartesian-Product.sum-cong-aux*) (*use i* **in** *auto*)
also have ... = *yn i* − *1* / *real* (*Suc n*)
using *i* **by** *auto*
also have ... $\leq$ *f x*
using *i*(*2*) **by**(*auto simp*: *En-def diff-less-eq order-less-le-trans intro*!: *order.strict-implies-order*)
finally show *?thesis* **.**
qed
next
show *integrable N* (λ*x*. $\sum$ *i*$\leq$*L*. (*yn i* − *1* / *real* (*Suc n*)) $*$ *indicat-real* (*En i*) *x*)
using *emeasure-En-fin* **by** *fastforce*

qed(*use g7 f* **in** *auto*)
finally show *?thesis* **.**
qed
thus *?thesis*
by *fastforce*
qed
finally show *?thesis* **.**
qed
show *?thesis*
proof(*rule Lim-bounded2*)
show (λn. ($\int x.\ f\ x\ \partial N$) + *1* / *real* (*Suc n*) $*$ (*2* $*$ *measure N ?K* + *1* / *real* (*Suc n*) + *2* $*$ *B* + *1*)) $\longrightarrow$ ($\int x.\ f\ x\ \partial N$)
apply(*rule tendsto-add*[**where** *b=0*,*simplified*])
apply *simp*
apply(*rule tendsto-mult*[**where** *a* = *0*::*real*, *simplified*,**where** *b=2* $*$ *measure N ?K* + *2* $*$ *B* + *1*])
apply(*intro LIMSEQ-Suc*[*OF lim-inverse-n'*] *tendsto-add*[*OF tendsto-const*,*of - 0*,*simplified*] *tendsto-add*[*OF - tendsto-const*])+
done
qed(*use 1* **in** *auto*)
qed
fix *f*
assume *f*: *?iscont f ?csupp f*
show φ ($\lambda x{\in}$*topspace X*. *f x*) = ($\int x.\ f\ x\ \partial N$)
proof(*rule antisym*)
have $-\ \varphi$ ($\lambda x{\in}$*topspace X*. *f x*) = φ ($\lambda x{\in}$*topspace X*. $-$ *f x*)
using *f* **by**(*auto intro*!: φ*diff*[*of* λx. *0 f*,*simplified* φ*-0*,*simplified*,*symmetric*])
also have ... $\leq$ ($\int x.\ -\ f\ x\ \partial N$)
by(*intro 1*) (*auto simp*: *f*)
also have ... = $-$ ($\int x.\ f\ x\ \partial N$)
by *simp*
finally show φ ($\lambda x{\in}$*topspace X*. *f x*) $\geq$ ($\int x.\ f\ x\ \partial N$)
by *linarith*
qed(*intro f 1*)
qed
show *?thesis*
apply(*intro exI*[**where** *x=M*] *ex1I*[**where** *a=N*] *rep-measures-real-unique*[*OF lh*(*1*,*2*),*of - N*])
using *sets-N g1 g2 g3 g4 g5 g6 g7 g8* **by** *auto*
qed

3.6 Riesz Representation Theorem for Complex Numbers

theorem *Riesz-representation-complex-complete*:
fixes *X* :: *'a topology* **and** φ :: (*'a* $\Rightarrow$ *complex*) $\Rightarrow$ *complex*
assumes *lh*:*locally-compact-space X Hausdorff-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
shows $\exists$ *M*. $\exists!$*N*. *sets N* = *M* $\wedge$ *subalgebra N* (*borel-of X*)
$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* = ($\bigsqcap$ *C*$\in${*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}. *emeasure*

N C))
∧ (∀ A. *openin* X A ⟶ *emeasure* N A = (⨆ K∈{K. *compactin* X K ∧ K ⊆ A}. *emeasure* N K))
∧ (∀ A∈*sets* N. *emeasure* N A < ∞ ⟶ *emeasure* N A = (⨆ K∈{K. *compactin* X K ∧ K ⊆ A}. *emeasure* N K))
∧ (∀ K. *compactin* X K ⟶ *emeasure* N K < ∞)
∧ (∀ f. *continuous-map* X *euclidean* f ⟶ f *has-compact-support-on* X ⟶ φ (λx∈*topspace* X. f x) = (∫ x. f x ∂N))
∧ (∀ f. *continuous-map* X *euclidean* f ⟶ f *has-compact-support-on* X ⟶ *complex-integrable* N f)
∧ *complete-measure* N
proof −
let *?φ′* = λf. *Re* (φ (λx∈*topspace* X. *complex-of-real* (f x)))
from *Riesz-representation-real-complete*[*OF lh pos-lin-functional-on-CX-complex-decompose-plf*[*OF plf*]]
obtain M N **where** *MN*:
sets N = M *subalgebra* N (*borel-of* X) (∀ A∈*sets* N. *emeasure* N A = ⨅ (*emeasure* N ‘ {C. *openin* X C ∧ A ⊆ C}))
(∀ A. *openin* X A ⟶ *emeasure* N A = ⨆ (*emeasure* N ‘ {K. *compactin* X K ∧ K ⊆ A}))
(∀ A∈*sets* N. *emeasure* N A < ∞ ⟶ *emeasure* N A = ⨆ (*emeasure* N ‘ {K. *compactin* X K ∧ K ⊆ A}))
(∀ K. *compactin* X K ⟶ *emeasure* N K < ∞)
⋀f. *continuous-map* X *euclideanreal* f ⟹ f *has-compact-support-on* X ⟹ *?φ′* (λx∈*topspace* X. f x) = (∫ x. f x ∂N)
⋀f. *continuous-map* X *euclideanreal* f ⟹ f *has-compact-support-on* X ⟹ *integrable* N f *complete-measure* N
by *fastforce*
have *MN1*: *complex-integrable* N f **if** f:*continuous-map* X *euclidean* f f *has-compact-support-on* X **for** f
using f **unfolding** *complex-integrable-iff*
by(*auto intro*!: *MN*(*8*))
have *MN2*: φ (λx∈*topspace* X. f x) = (∫ x. f x ∂N)
if f:*continuous-map* X *euclidean* f f *has-compact-support-on* X **for** f
proof −
have φ (λx∈*topspace* X. f x)
= *complex-of-real* (*?φ′* (λx∈*topspace* X. *Re* (f x))) + i ∗ *complex-of-real* (*?φ′* (λx∈*topspace* X. *Im* (f x)))
using f **by**(*intro pos-lin-functional-on-CX-complex-decompose*[*OF plf*])
also have ... = *complex-of-real* (∫ x. *Re* (f x) ∂N) + i ∗ *complex-of-real* (∫ x. *Im* (f x) ∂N)
proof −
have ∗:*?φ′* (λx∈*topspace* X. *Re* (f x)) = (∫ x. *Re* (f x) ∂N)
using f **by**(*intro MN*(*7*)) *auto*
have ∗∗:*?φ′* (λx∈*topspace* X. *Im* (f x)) = (∫ x. *Im* (f x) ∂N)
using f **by**(*intro MN*(*7*)) *auto*
show *?thesis*
unfolding ∗ ∗∗ **..**
qed

also have ... = *complex-of-real* (*Re* ($\int$ *x*. *f x* ∂*N*)) + i $*$ *complex-of-real* (*Im* ($\int$ *x*. *f x* ∂*N*))
by(*simp add*: *integral-Im*[*OF MN1*[*OF that*]] *integral-Re*[*OF MN1*[*OF that*]])
also have ... = ($\int$ *x*. *f x* ∂*N*)
using *complex-eq* **by** *auto*
finally show *?thesis* **.**
qed
show *?thesis*
apply(*intro exI*[**where** *x*=*M*] *ex1I*[**where** *a*=*N*] *rep-measures-complex-unique*[*OF lh*])
using *MN*(*1*−*6*,*9*) *MN1 MN2*
by *auto*
qed

3.7 Other Forms of the Theorem

In the case when the representation measure is on X.

theorem *Riesz-representation-real*:
assumes *lh*:*locally-compact-space X Hausdorff-space X*
and *positive-linear-functional-on-CX X* φ
shows $\exists$!*N*. *sets N* = *sets* (*borel-of X*)
$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* = ($\prod$ *C*$\in${*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}. *emeasure N C*))
$\wedge$ ($\forall$ *A*. *openin X A* $\longrightarrow$ *emeasure N A* = ($\bigsqcup$ *K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* < ∞ $\longrightarrow$ *emeasure N A* = ($\bigsqcup$ *K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
$\wedge$ ($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* < ∞)
$\wedge$ ($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ *f has-compact-support-on X* $\longrightarrow$ φ (λ*x*$\in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
$\wedge$ ($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ *f has-compact-support-on X* $\longrightarrow$ *integrable N f*)
proof −
from *Riesz-representation-real-complete*[*OF assms*] **obtain** *M N* **where** *MN*:
sets N = *M subalgebra N* (*borel-of X*) ($\forall$ *A*$\in$*sets N*. *emeasure N A* = $\prod$ (*emeasure N* ‘ {*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}))
($\forall$ *A*. *openin X A* $\longrightarrow$ *emeasure N A* = $\bigsqcup$ (*emeasure N* ‘ {*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}))
($\forall$ *A*$\in$*sets N*. *emeasure N A* < ∞ $\longrightarrow$ *emeasure N A* = $\bigsqcup$ (*emeasure N* ‘ {*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}))
($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* < ∞)
$\bigwedge$*f*. *continuous-map X euclideanreal f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ φ (λ*x*$\in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*)
$\bigwedge$*f*. *continuous-map X euclideanreal f* $\Longrightarrow$ *f has-compact-support-on X* $\Longrightarrow$ *integrable N f complete-measure N*
by *fastforce*

define *N′* **where** *N′* $\equiv$ *restr-to-subalg N* (*borel-of X*)
have *g1*: *sets N′* = *sets* (*borel-of X*) (**is** *?g1*)

and *g2*: ∀*A*∈*sets N′*. *emeasure N′ A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N′ C*) (**is** *?g2*)
and *g3*: ∀*A*. *openin X A* ⟶ *emeasure N′ A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N′ K*) (**is** *?g3*)
and *g4*: ∀*A*∈*sets N′*. *emeasure N′ A* < ∞ ⟶ *emeasure N′ A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N′ K*) (**is** *?g4*)
and *g5*: ∀*K*. *compactin X K* ⟶ *emeasure N′ K* < ∞ (**is** *?g5*)
and *g6*: ∀*f*. *continuous-map X euclideanreal f* ⟶ *f has-compact-support-on X* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N′*) (**is** *?g6*)
and *g7*: ∀*f*. *continuous-map X euclideanreal f* ⟶ *f has-compact-support-on X* ⟶ *integrable N′ f* (**is** *?g7*)
proof −
have *sets-N′*: *sets N′* = *borel-of X*
using *sets-restr-to-subalg*[*OF MN*(*2*)] **by**(*auto simp*: *N′-def*)
have *emeasure-N′*: ⋀*A*. *A* ∈ *sets N′* ⟹ *emeasure N′ A* = *emeasure N A*
by (*simp add*: *MN*(*2*) *N′-def emeasure-restr-to-subalg sets-restr-to-subalg*)
have *setsN′*[*measurable*]: ⋀*A*. *openin X A* ⟹ *A* ∈ *sets N′* ⋀*A*. *compactin X A* ⟹ *A* ∈ *sets N′*
by(*auto simp*: *sets-N′ dest*: *borel-of-open borel-of-closed*[*OF compactin-imp-closedin*[*OF lh*(*2*)]])
have *sets-N′-sets-N*[*simp*]: ⋀*A*. *A* ∈ *sets N′* ⟹ *A* ∈ *sets N*
using *MN*(*2*) *sets-N′ subalgebra-def* **by** *blast*
show *?g1*
by (*simp add*: *MN*(*2*) *N′-def sets-restr-to-subalg*)
show *?g2*
using *MN*(*3*) **by**(*auto simp*: *emeasure-N′*)
show *?g3*
using *MN*(*4*) **by**(*auto simp*: *emeasure-N′*)
show *?g4*
using *MN*(*5*) **by**(*auto simp*: *emeasure-N′*)
show *?g5*
using *MN*(*6*) **by**(*auto simp*: *emeasure-N′*)
show *?g6 ?g7*
proof *safe*
fix *f*
assume *f*:*continuous-map X euclideanreal f f has-compact-support-on X*
then have [*measurable*]: *f* ∈ *borel-measurable* (*borel-of X*)
by (*simp add*: *continuous-lower-semicontinuous lower-semicontinuous-map-measurable*)
from *MN*(*7*,*8*) *f* **show** φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N′*) *integrable N′ f*
by(*auto simp*: *N′-def integral-subalgebra2*[*OF MN*(*2*)] *intro*!: *integrable-in-subalg*[*OF MN*(*2*)])
qed
qed
have *g8*: ⋀*L*. *sets L* = *sets* (*borel-of X*) ⟹ *subalgebra L* (*borel-of X*)
by (*metis sets-eq-imp-space-eq subalgebra-def subset-refl*)

show *?thesis*
apply(*intro ex1I*[**where** *a*=*N′*] *rep-measures-real-unique*[*OF lh*])

using *g1 g2 g3 g4 g5 g6 g7 g8* **by** *auto*
qed

theorem *Riesz-representation-complex*:
fixes X :: *'a topology* **and** φ :: (*'a* ⇒ *complex*) ⇒ *complex*
assumes *lh*:*locally-compact-space X Hausdorff-space X*
and *positive-linear-functional-on-CX X* φ
shows ∃!*N*. *sets N* = *sets* (*borel-of X*)
∧ (∀ *A*∈*sets N*. *emeasure N A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))
∧ (∀ *A*. *openin X A* ⟶ *emeasure N A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
∧ (∀ *A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
∧ (∀ *K*. *compactin X K* ⟶ *emeasure N K* < ∞)
∧ (∀ *f*. *continuous-map X euclidean f* ⟶ *f has-compact-support-on X* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
∧ (∀ *f*. *continuous-map X euclidean f* ⟶ *f has-compact-support-on X* ⟶ *complex-integrable N f*)
proof −
from *Riesz-representation-complex-complete*[*OF assms*] **obtain** *M N* **where** *MN*:
sets N = *M subalgebra N* (*borel-of X*) (∀ *A*∈*sets N*. *emeasure N A* = ⨅ (*emeasure N* ‘ {*C*. *openin X C* ∧ *A* ⊆ *C*}))
(∀ *A*. *openin X A* ⟶ *emeasure N A* = ⨆ (*emeasure N* ‘ {*K*. *compactin X K* ∧ *K* ⊆ *A*}))
(∀ *A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = ⨆ (*emeasure N* ‘ {*K*. *compactin X K* ∧ *K* ⊆ *A*}))
(∀ *K*. *compactin X K* ⟶ *emeasure N K* < ∞)
⋀*f*. *continuous-map X euclidean f* ⟹ *f has-compact-support-on X* ⟹ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*)
⋀*f*. *continuous-map X euclidean f* ⟹ *f has-compact-support-on X* ⟹ *complex-integrable N f complete-measure N*
by *fastforce*

define *N′* **where** *N′* ≡ *restr-to-subalg N* (*borel-of X*)
have *g1*: *sets N′* = *sets* (*borel-of X*) (**is** *?g1*)
and *g2*: ∀ *A*∈*sets N′*. *emeasure N′ A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N′ C*) (**is** *?g2*)
and *g3*: ∀ *A*. *openin X A* ⟶ *emeasure N′ A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N′ K*) (**is** *?g3*)
and *g4*: ∀ *A*∈*sets N′*. *emeasure N′ A* < ∞ ⟶ *emeasure N′ A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N′ K*) (**is** *?g4*)
and *g5*: ∀ *K*. *compactin X K* ⟶ *emeasure N′ K* < ∞ (**is** *?g5*)
and *g6*: ∀ *f*. *continuous-map X euclidean f* ⟶ *f has-compact-support-on X* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N′*) (**is** *?g6*)
and *g7*: ∀ *f*. *continuous-map X euclidean f* ⟶ *f has-compact-support-on X* ⟶ *complex-integrable N′ f* (**is** *?g7*)
proof −
have *sets-N′*: *sets N′* = *borel-of X*

using *sets-restr-to-subalg*[*OF MN*(*2*)] **by**(*auto simp*: *N′-def*)
have *emeasure-N′*: ⋀*A*. *A* ∈ *sets N′* ⟹ *emeasure N′ A* = *emeasure N A*
by (*simp add*: *MN*(*2*) *N′-def emeasure-restr-to-subalg sets-restr-to-subalg*)
have *setsN′*[*measurable*]: ⋀*A*. *openin X A* ⟹ *A* ∈ *sets N′* ⋀*A*. *compactin X A* ⟹ *A* ∈ *sets N′*
by(*auto simp*: *sets-N′ dest*: *borel-of-open borel-of-closed*[*OF compactin-imp-closedin*[*OF lh*(*2*)]])
have *sets-N′-sets-N*[*simp*]: ⋀*A*. *A* ∈ *sets N′* ⟹ *A* ∈ *sets N*
using *MN*(*2*) *sets-N′ subalgebra-def* **by** *blast*
show *?g1*
by (*simp add*: *MN*(*2*) *N′-def sets-restr-to-subalg*)
show *?g2*
using *MN*(*3*) **by**(*auto simp*: *emeasure-N′*)
show *?g3*
using *MN*(*4*) **by**(*auto simp*: *emeasure-N′*)
show *?g4*
using *MN*(*5*) **by**(*auto simp*: *emeasure-N′*)
show *?g5*
using *MN*(*6*) **by**(*auto simp*: *emeasure-N′*)
show *?g6 ?g7*
proof *safe*
fix *f* :: - ⇒ *complex*
assume *f*:*continuous-map X euclidean f f has-compact-support-on X*
then have [*measurable*]: *f* ∈ *borel-measurable* (*borel-of X*)
by (*metis borel-of-euclidean continuous-map-measurable*)
show *φ* (*λx*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N′*) *integrable N′ f*
using *MN*(*7*,*8*) *f* **by**(*auto simp*: *N′-def integral-subalgebra2*[*OF MN*(*2*)] *intro*!: *integrable-in-subalg*[*OF MN*(*2*)])
qed
qed
have *g8*: ⋀*L*. *sets L* = *sets* (*borel-of X*) ⟹ *subalgebra L* (*borel-of X*)
by (*metis sets-eq-imp-space-eq subalgebra-def subset-refl*)

show *?thesis*
apply(*intro ex1I*[**where** *a*=*N′*] *rep-measures-complex-unique*[*OF lh*])
using *g1 g2 g3 g4 g5 g6 g7 g8* **by** *auto*
qed

3.7.1 Theorem for Compact Hausdorff Spaces

theorem *Riesz-representation-real-compact-Hausdorff*:
fixes *X* :: *′a topology* **and** *φ* :: (*′a* ⇒ *real*) ⇒ *real*
assumes *lh*:*compact-space X Hausdorff-space X*
and *positive-linear-functional-on-CX X φ*
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *finite-measure N*
∧ (∀ *A*∈*sets N*. *emeasure N A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))
∧ (∀ *A*. *openin X A* ⟶ *emeasure N A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))

$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* $<$ ∞ $\longrightarrow$ *emeasure N A* = ($\bigsqcup$*K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
$\wedge$ ($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* $<$ ∞)
$\wedge$ ($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ φ (λ*x*$\in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
$\wedge$ ($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ *integrable N f*)
proof −
have [*simp*]: *compactin X* (*X closure-of A*) **for** *A*
by (*simp add*: *closedin-compact-space lh*(*1*))
from *Riesz-representation-real*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *assms*(*2*,*3*)] **obtain** *N* **where** *N*:
sets N = *sets* (*borel-of X*)
($\forall$ *A*$\in$*sets N*. *emeasure N A* = ($\bigsqcap$ *C*$\in${*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}. *emeasure N C*))
($\forall$ *A*. *openin X A* $\longrightarrow$ *emeasure N A* = ($\bigsqcup$*K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
($\forall$ *A*$\in$*sets N*. *emeasure N A* $<$ ∞ $\longrightarrow$ *emeasure N A* = ($\bigsqcup$*K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* $<$ ∞)
($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ φ (λ*x*$\in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
($\forall$ *f*. *continuous-map X euclideanreal f* $\longrightarrow$ *integrable N f*)
by(*fastforce simp*: *assms*(*1*))
have *space-N*:*space N* = *topspace X*
by (*simp add*: *N*(*1*) *sets-eq-imp-space-eq space-borel-of*)
have *fin*:*finite-measure N*
using *N*(*5*)[*rule-format*,*of topspace X*] *lh*(*1*)
by(*auto intro*!: *finite-measureI simp*: *space-N compact-space-def*)
have *1*:$\bigwedge$*L*. *sets L* = *sets* (*borel-of X*) $\Longrightarrow$ *subalgebra L* (*borel-of X*)
by (*metis sets-eq-imp-space-eq subalgebra-def subset-refl*)
show *?thesis*
by(*intro ex1I*[**where** *a*=*N*] *rep-measures-real-unique*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *lh*(*2*)])
(*use N fin 1* **in** *auto*)
qed

theorem *Riesz-representation-complex-compact-Hausdorff*:
fixes *X* :: *'a topology* **and** φ :: (*'a* $\Rightarrow$ *complex*) $\Rightarrow$ *complex*
assumes *lh*:*compact-space X Hausdorff-space X*
and *positive-linear-functional-on-CX X* φ
shows $\exists$!*N*. *sets N* = *sets* (*borel-of X*) $\wedge$ *finite-measure N*
$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* = ($\bigsqcap$ *C*$\in${*C*. *openin X C* $\wedge$ *A* $\subseteq$ *C*}. *emeasure N C*))
$\wedge$ ($\forall$ *A*. *openin X A* $\longrightarrow$ *emeasure N A* = ($\bigsqcup$*K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
$\wedge$ ($\forall$ *A*$\in$*sets N*. *emeasure N A* $<$ ∞ $\longrightarrow$ *emeasure N A* = ($\bigsqcup$*K*$\in${*K*. *compactin X K* $\wedge$ *K* $\subseteq$ *A*}. *emeasure N K*))
$\wedge$ ($\forall$ *K*. *compactin X K* $\longrightarrow$ *emeasure N K* $<$ ∞)
$\wedge$ ($\forall$ *f*. *continuous-map X euclidean f* $\longrightarrow$ φ (λ*x*$\in$*topspace X*. *f x*) = ($\int$ *x*.

f x ∂N))
∧ (∀*f*. *continuous-map X euclidean f* ⟶ *complex-integrable N f*)
proof −
have [*simp*]: *compactin X* (*X closure-of A*) **for** *A*
by (*simp add*: *closedin-compact-space lh*(*1*))
from *Riesz-representation-complex*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *assms*(*2*,*3*)] **obtain** *N* **where** *N*:
sets N = *sets* (*borel-of X*)
(∀*A*∈*sets N*. *emeasure N A* = (⨅*C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))
(∀*A*. *openin X A* ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*K*. *compactin X K* ⟶ *emeasure N K* < ∞)
(∀*f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
(∀*f*. *continuous-map X euclidean f* ⟶ *complex-integrable N f*)
by (*fastforce simp*: *lh*(*1*))
have *space-N*:*space N* = *topspace X*
by (*simp add*: *N*(*1*) *sets-eq-imp-space-eq space-borel-of*)
have *fin*:*finite-measure N*
using *N*(*5*)[*rule-format*,*of topspace X*] *lh*(*1*)
by(*auto intro*!: *finite-measureI simp*: *space-N compact-space-def*)
have *1*:⋀*L*. *sets L* = *sets* (*borel-of X*) ⟹ *subalgebra L* (*borel-of X*)
by (*metis sets-eq-imp-space-eq subalgebra-def subset-refl*)
show *?thesis*
by(*intro ex1I*[**where** *a*=*N*] *rep-measures-complex-unique*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *lh*(*2*)])
(*use N fin 1* **in** *auto*)
qed

3.7.2 Theorem for Compact Metrizable Spaces

theorem *Riesz-representation-real-compact-metrizable*:
fixes *X* :: *'a topology* **and** φ :: (*'a* ⇒ *real*) ⇒ *real*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *finite-measure N*
∧ (∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
proof −
have *hd*: *Hausdorff-space X*
by (*simp add*: *lh*(*2*) *metrizable-imp-Hausdorff-space*)

from *Riesz-representation-real-compact-Hausdorff*[*OF lh*(*1*) *hd plf*] **obtain** *N* **where** *N*:
sets N = *sets* (*borel-of X*) *finite-measure N*
(∀*A*∈*sets N*. *emeasure N A* = (⨅*C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))

(∀*A*. *openin X A* ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*K*. *compactin X K* ⟶ *emeasure N K* < ∞)
(∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
(∀*f*. *continuous-map X euclideanreal f* ⟶ *integrable N f*)
by *fastforce*
then have *tight-on-N*:*tight-on X N*
using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *metis*

show *?thesis*
proof(*safe intro*!: *ex1I*[**where** *a*=*N*])
fix *M*
assume *M*:*sets M* = *sets* (*borel-of X*) *finite-measure M* (∀*f*. *continuous-map X euclideanreal f* ⟶ φ (*restrict f* (*topspace X*)) = *integral*^{L} *M f*)
then have *tight-on X M*
using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *blast*
thus *M* = *N*
using *N*(*7*) *M*(*3*) **by**(*auto intro*!: *finite-tight-measure-eq*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *lh*(*2*)] *tight-on-N*)
qed(*use N* **in** *auto*)
qed

theorem *Riesz-representation-real-compact-metrizable-le1*:
fixes *X* :: ′*a topology* **and** φ :: (′*a* ⇒ *real*) ⇒ *real*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *finite-measure N*
∧ (∀*f*. *continuous-map X euclideanreal f* ⟶ *f* ∈ *topspace X* → {*0*..*1*} ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
proof −
have *hd*: *Hausdorff-space X*
by (*simp add*: *lh*(*2*) *metrizable-imp-Hausdorff-space*)

from *Riesz-representation-real-compact-Hausdorff*[*OF lh*(*1*) *hd plf*] **obtain** *N*
where *N*:
sets N = *sets* (*borel-of X*) *finite-measure N*
(∀*A*∈*sets N*. *emeasure N A* = (⨅*C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))
(∀*A*. *openin X A* ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = (⨆*K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*K*. *compactin X K* ⟶ *emeasure N K* < ∞)
(∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
(∀*f*. *continuous-map X euclideanreal f* ⟶ *integrable N f*)

by *fastforce*
then have *tight-on-N*:*tight-on X N*
using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *metis*

show *?thesis*
proof(*safe intro*!: *ex1I*[**where** *a*=*N*])
fix *M*
assume *M*:*sets M* = *sets* (*borel-of X*) *finite-measure M* (∀*f*. *continuous-map X euclideanreal f* ⟶ *f* ∈ *topspace X* → {*0..1*} ⟶ φ (*restrict f* (*topspace X*)) = *integral*^{L} *M f*)
then have *tight-on X M*
using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *blast*
thus *M* = *N*
using *N*(*7*) *M*(*3*) **by**(*auto intro*!: *finite-tight-measure-eq*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *lh*(*2*)] *tight-on-N*)
qed(*use N* **in** *auto*)
qed

theorem *Riesz-representation-complex-compact-metrizable*:
fixes *X* :: *'a topology* **and** φ :: (*'a* ⇒ *complex*) ⇒ *complex*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *finite-measure N* ∧ (∀*f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
proof−
have *hd*: *Hausdorff-space X*
by (*simp add*: *lh*(*2*) *metrizable-imp-Hausdorff-space*)

from *Riesz-representation-complex-compact-Hausdorff*[*OF lh*(*1*) *hd plf*] **obtain** *N* **where** *N*:
sets N = *sets* (*borel-of X*) *finite-measure N*
(∀*A*∈*sets N*. *emeasure N A* = (⨅ *C*∈{*C*. *openin X C* ∧ *A* ⊆ *C*}. *emeasure N C*))
(∀*A*. *openin X A* ⟶ *emeasure N A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*A*∈*sets N*. *emeasure N A* < ∞ ⟶ *emeasure N A* = (⨆ *K*∈{*K*. *compactin X K* ∧ *K* ⊆ *A*}. *emeasure N K*))
(∀*K*. *compactin X K* ⟶ *emeasure N K* < ∞)
(∀*f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
(∀*f*. *continuous-map X euclidean f* ⟶ *complex-integrable N f*)
by *fastforce*
then have *tight-on-N*:*tight-on X N*
using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *metis*

show *?thesis*
proof(*safe intro*!: *ex1I*[**where** *a*=*N*])
fix *M*
assume *M*:*sets M* = *sets* (*borel-of X*) *finite-measure M* (∀*f*. *continuous-map*

X euclidean f ⟶ φ (*restrict f* (*topspace X*)) = ($\int$ *x*. *f x* ∂*M*))
 then have *tight-on-M*:*tight-on X M*
 using *finite-measure.tight-on-compact-space lh*(*1*) *lh*(*2*) **by** *blast*
 have ($\int$ *x*. *f x* ∂*N*) = ($\int$ *x*. *f x* ∂*M*) **if** *f*:*continuous-map X euclideanreal f* **for** *f*
 proof −
 have ($\int$ *x*. *f x* ∂*N*) = *Re* ($\int$ *x*. *complex-of-real* (*f x*) ∂*N*)
 by *simp*
 also have ... = *Re* (φ (λ*x*∈*topspace X*. *complex-of-real* (*f x*)))
 by(*intro arg-cong*[**where** *f*=*Re*] *N*(*7*)[*rule-format*,*symmetric*]) (*simp add*: *f*)
 also have ... = *Re* ($\int$ *x*. *complex-of-real* (*f x*) ∂*M*)
 by(*intro arg-cong*[**where** *f*=*Re*] *M*(*3*)[*rule-format*]) (*simp add*: *f*)
 also have ... = ($\int$ *x*. *f x* ∂*M*)
 by *simp*
 finally show *?thesis* **.**
 qed
 thus *M* = *N*
 by(*auto intro*!: *finite-tight-measure-eq*[*OF compact-imp-locally-compact-space*[*OF lh*(*1*)] *lh*(*2*)] *tight-on-N tight-on-M*)
 qed(*use N* **in** *auto*)
qed

theorem *Riesz-representation-real-compact-metrizable-subprob*:
 fixes *X* :: *'a topology* **and** φ :: (*'a* ⇒ *real*) ⇒ *real*
 assumes *lh*:*compact-space X metrizable-space X*
 and *plf*:*positive-linear-functional-on-CX X* φ
 and *le1*: φ (λ*x*∈*topspace X*. *1*) ≤ *1* **and** *ne*: *X* ≠ *trivial-topology*
 shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *subprob-space N*
 ∧ (∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
proof −
 from *Riesz-representation-real-compact-metrizable*[*OF assms*(*1*−*3*)]
 obtain *N* **where** *N*: *sets N* = *sets* (*borel-of X*) *finite-measure N* (∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
 ⋀*M*. *sets M* = *sets* (*borel-of X*) ⟹ *finite-measure M* ⟹ (∀*f*. *continuous-map X euclideanreal f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*M*)) ⟹ *M* = *N*
 by *fastforce*
 then interpret *finite-measure N*
 by *blast*
 have *subN*:*subprob-space N*
 proof
 have *measure N* (*space N*) = ($\int$ *x*. *1* ∂*N*)
 by *simp*
 also have ... = φ (λ*x*∈*topspace X*. *1*)
 by(*intro N*(*3*)[*rule-format*,*symmetric*]) *simp*
 also have ... ≤ *1*
 by *fact*
 finally show *emeasure N* (*space N*) ≤ *1*
 by (*simp add*: *emeasure-eq-measure*)

next
show *space N* ≠ {}
using *sets-eq-imp-space-eq*[*OF N*(*1*)] *ne* **by**(*auto simp*: *space-borel-of*)
qed
show *?thesis*
using *N*(*4*)[*OF - subprob-space.axioms*(*1*)] *subN N*(*1*,*3*) **by**(*auto intro*!: *ex1I*[**where** *a*=*N*])
qed

theorem *Riesz-representation-real-compact-metrizable-subprob-le1*:
fixes *X* :: *'a topology* **and** φ :: (*'a* ⇒ *real*) ⇒ *real*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
and *le1*: φ (λ*x*∈*topspace X*. *1*) ≤ *1* **and** *ne*: *X* ≠ *trivial-topology*
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *subprob-space N*
∧ (∀ *f*. *continuous-map X euclideanreal f* ⟶ *f* ∈ *topspace X* → {*0*..*1*} ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
proof −
from *Riesz-representation-real-compact-metrizable-le1*[*OF lh plf*]
obtain *N* **where** *N*: *sets N* = *sets* (*borel-of X*) *finite-measure N* (∀ *f*. *continuous-map X euclideanreal f* ⟶ *f* ∈ *topspace X* → {*0*..*1*} ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*N*))
⋀*M*. *sets M* = *sets* (*borel-of X*) ⟹ *finite-measure M* ⟹ (∀ *f*. *continuous-map X euclideanreal f* ⟶ *f* ∈ *topspace X* → {*0*..*1*} ⟶ φ (λ*x*∈*topspace X*. *f x*) = (∫ *x*. *f x* ∂*M*)) ⟹ *M* = *N*
by *fastforce*
then interpret *finite-measure N*
by *blast*
have *subN*:*subprob-space N*
proof
have *measure N* (*space N*) = (∫ *x*. *1* ∂*N*)
by *simp*
also have ... = φ (λ*x*∈*topspace X*. *1*)
by(*intro N*(*3*)[*rule-format*,*symmetric*]) *simp-all*
also have ... ≤ *1*
by *fact*
finally show *emeasure N* (*space N*) ≤ *1*
by (*simp add*: *emeasure-eq-measure*)
next
show *space N* ≠ {}
using *sets-eq-imp-space-eq*[*OF N*(*1*)] *ne* **by**(*auto simp*: *space-borel-of*)
qed
show *?thesis*
using *N*(*4*)[*OF - subprob-space.axioms*(*1*)] *subN N*(*1*,*3*) **by**(*auto intro*!: *ex1I*[**where** *a*=*N*])
qed

theorem *Riesz-representation-real-compact-metrizable-prob*:
fixes *X* :: *'a topology* **and** φ :: (*'a* ⇒ *real*) ⇒ *real*

assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
and φ ($\lambda x \in$*topspace X*. *1*) = *1*
shows $\exists!N$. *sets N* = *sets* (*borel-of X*) $\wedge$ *prob-space N*
$\wedge$ ($\forall f$. *continuous-map X euclideanreal f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂N))
proof –
from *Riesz-representation-real-compact-metrizable*[*OF lh plf*]
obtain *N* **where** *N*: *sets N* = *sets* (*borel-of X*) *finite-measure N* ($\forall f$. *continuous-map X euclideanreal f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂N))
$\bigwedge M$. *sets M* = *sets* (*borel-of X*) $\Longrightarrow$ *finite-measure M* $\Longrightarrow$ ($\forall f$. *continuous-map X euclideanreal f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂M)) $\Longrightarrow$ *M* = *N*
by *fastforce*
then interpret *finite-measure N*
by *blast*
have *probN*:*prob-space N*
proof
have *measure N* (*space N*) = ($\int$ *x*. *1* ∂N)
by *simp*
also have ... = φ ($\lambda x \in$*topspace X*. *1*)
by(*intro N*(*3*)[*rule-format*,*symmetric*]) *simp*
also have ... = *1*
by *fact*
finally show *emeasure N* (*space N*) = *1*
by (*simp add*: *emeasure-eq-measure*)
qed
show *?thesis*
using *N*(*4*)[*OF - prob-space.finite-measure*] *probN N*(*1*,*3*) **by**(*auto intro*!: *ex1I*[**where** *a*=*N*])
qed

theorem *Riesz-representation-complex-compact-metrizable-subprob*:
fixes *X* :: *'a topology* **and** φ :: (*'a* $\Rightarrow$ *complex*) $\Rightarrow$ *complex*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
and *le1*: *Re* (φ ($\lambda x \in$*topspace X*. *1*)) $\leq$ *1* **and** *ne*: *X* $\neq$ *trivial-topology*
shows $\exists!N$. *sets N* = *sets* (*borel-of X*) $\wedge$ *subprob-space N*
$\wedge$ ($\forall f$. *continuous-map X euclidean f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂N))
proof –
from *Riesz-representation-complex-compact-metrizable*[*OF lh plf*]
obtain *N* **where** *N*: *sets N* = *sets* (*borel-of X*) *finite-measure N* ($\forall f$. *continuous-map X euclidean f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂N))
$\bigwedge M$. *sets M* = *sets* (*borel-of X*) $\Longrightarrow$ *finite-measure M* $\Longrightarrow$ ($\forall f$. *continuous-map X euclidean f* $\longrightarrow$ φ ($\lambda x \in$*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂M)) $\Longrightarrow$ *M* = *N*
by *fastforce*
then interpret *finite-measure N*
by *blast*
have *subN*:*subprob-space N*

proof
have *measure N* (*space N*) = ($\int$ *x*. *1* ∂*N*)
by *simp*
also have ... = *Re* ($\int$ *x*. *1* ∂*N*)
by *simp*
also have ... = *Re* (φ (λ*x*∈*topspace X*. *1*))
by(*intro arg-cong*[**where** *f*=*Re*] *N*(*3*)[*rule-format*,*symmetric*]) *simp*
also have ... ≤ *1*
by *fact*
finally show *emeasure N* (*space N*) ≤ *1*
by (*simp add*: *emeasure-eq-measure*)
next
show *space N* ≠ {}
using *sets-eq-imp-space-eq*[*OF N*(*1*)] *ne* **by**(*auto simp*: *space-borel-of*)
qed
show *?thesis*
using *N*(*4*)[*OF* - *subprob-space.axioms*(*1*)] *subN N*(*1*,*3*) **by**(*auto intro*!: *ex1I*[**where** *a*=*N*])
qed

theorem *Riesz-representation-complex-compact-metrizable-prob*:
fixes *X* :: ′*a topology* **and** φ :: (′*a* ⇒ *complex*) ⇒ *complex*
assumes *lh*:*compact-space X metrizable-space X*
and *plf*:*positive-linear-functional-on-CX X* φ
and *Re* (φ (λ*x*∈*topspace X*. *1*)) = *1*
shows ∃!*N*. *sets N* = *sets* (*borel-of X*) ∧ *prob-space N*
∧ (∀ *f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
proof −
from *Riesz-representation-complex-compact-metrizable*[*OF lh plf*]
obtain *N* **where** *N*: *sets N* = *sets* (*borel-of X*) *finite-measure N* (∀ *f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*N*))
⋀*M*. *sets M* = *sets* (*borel-of X*) ⟹ *finite-measure M* ⟹ (∀ *f*. *continuous-map X euclidean f* ⟶ φ (λ*x*∈*topspace X*. *f x*) = ($\int$ *x*. *f x* ∂*M*)) ⟹ *M* = *N*
by *fastforce*
then interpret *finite-measure N*
by *blast*
have *probN*:*prob-space N*
proof
have *measure N* (*space N*) = ($\int$ *x*. *1* ∂*N*)
by *simp*
also have ... = *Re* ($\int$ *x*. *1* ∂*N*)
by *simp*
also have ... = *Re* (φ (λ*x*∈*topspace X*. *1*))
by(*intro arg-cong*[**where** *f*=*Re*] *N*(*3*)[*rule-format*,*symmetric*]) *simp*
also have ... = *1*
by *fact*
finally show *emeasure N* (*space N*) = *1*
by (*simp add*: *emeasure-eq-measure*)

```
  qed
  show ?thesis
    using N(4)[OF - prob-space.finite-measure] probN N(1,3) by(auto intro!:
ex1I[where a=N])
qed

end
```

References

[1] W. Rudin. *Real and Complex Analysis, 3rd Ed.* McGraw-Hill, Inc., USA, 1987.

[2] O. van Gaans. Probability measures on metric spaces. https://www.math.leidenuniv.nl/~vangaans/jancol1.pdf. Accessed: March 2. 2023.