Disintegration Theorem

Michikazu Hirata

March 17, 2025

Abstract

We formalize mixture and disintegraion of measures. This entry is a formalization of Chapter 14.D of the book by Baccelli et.al. [1]. The main result is the disintegration theorem: let (X, Σ_X) be a measurable space, (Y, Σ_Y) be a standard Borel space, ν be a σ-finite measure on $X \times Y$, and ν_X be the marginal measure on X defined by $\nu_X(A) = \nu(A \times Y)$. Assume that ν_X is σ-finite, then there exists a probability kernel κ from X to Y such that

$$\nu(A \times B) = \int_A \kappa_x(B)\nu_X(\mathrm{d}x),\ A \in \Sigma_X, B \in \Sigma_Y.$$

Such a probability kernel is unique ν_X-almost everywhere.

Contents

1 Lemmas

theory *Lemmas-Disintegration*
imports *Standard-Borel-Spaces.StandardBorel*
begin

1.1 Lemmas

lemma *semiring-of-sets-binary-product-sets*[*simp*]:
semiring-of-sets (*space X* × *space Y*) {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
proof
show {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} ⊆ *Pow* (*space X* × *space Y*)
using *pair-measure-closed* **by** *blast*
next
fix *c d*
assume *c* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} *d* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
then obtain *ac bc ad bd* **where**
c = *ac* × *bc ac* ∈ *sets X bc* ∈ *sets Y d* = *ad* × *bd ad* ∈ *sets X bd* ∈ *sets Y*
by *auto*
thus *c* ∩ *d* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
by(*auto intro*!: *exI*[**where** *x*=*ac* ∩ *ad*] *exI*[**where** *x*=*bc* ∩ *bd*])
next
fix *c d*
assume *c* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} *d* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
then obtain *ac bc ad bd* **where** *cd*:
c = *ac* × *bc ac* ∈ *sets X bc* ∈ *sets Y d* = *ad* × *bd ad* ∈ *sets X bd* ∈ *sets Y*
by *auto*
then have *eq1*:*c* − *d* = ((*ac* − *ad*) × (*bc* − *bd*)) ∪ ((*ac* − *ad*) × (*bc* ∩ *bd*)) ∪ ((*ac* ∩ *ad*) × (*bc* − *bd*))
by *blast*
obtain *a1* **where** *a1*: *a1*⊆*sets X finite a1 disjoint a1 ac* − *ad* = ⋃ *a1*
using *cd sets.Diff-cover*[*of ac X ad*] **by** *auto*
obtain *a2* **where** *a2* : *a2*⊆*sets Y finite a2 disjoint a2 bc* − *bd* = ⋃ *a2*
using *cd sets.Diff-cover*[*of bc Y bd*] **by** *auto*
define *A1 A2 A3*
where *A1-def*:*A1* ≡ {*a* × *b*|*a b*. *a* ∈ *a1* ∧ *b* ∈ *a2*}
and *A2-def*:*A2* ≡ {*a* × (*bc* ∩ *bd*)|*a* . *a* ∈ *a1*}
and *A3-def*:*A3* ≡ {(*ac* ∩ *ad*) × *b*|*b*. *b* ∈ *a2*}
have *disj*:*disjoint* (*A1* ∪ *A2* ∪ *A3*)
proof −
have [*simp*]: *disjoint A1*
proof
fix *x y*
assume *x* ∈ *A1 y* ∈ *A1 x* ≠ *y*
then obtain *xa xb ya yb* **where** *xy*: *x* = *xa* × *xb xa* ∈ *a1 xb* ∈ *a2 y* = *ya* × *yb ya* ∈ *a1 yb* ∈ *a2*
by(*auto simp*: *A1-def*)
with ‹*x* ≠ *y*› **consider** *xa* ≠ *ya* | *xb* ≠ *yb* **by** *auto*
thus *disjnt x y*
proof *cases*
case *1*
then have *xa* ∩ *ya* = {}
using *a1*(*3*) *xy* **by**(*auto simp*: *disjoint-def*)
thus *?thesis*

```
      by(auto simp: xy disjnt-def)
    next
      case 2
      then have xb ∩ yb = {}
        using a2(3) xy by(auto simp: disjoint-def)
      thus ?thesis
        by(auto simp: xy disjnt-def)
    qed
  qed
  have [simp]: disjoint A2
  proof
    fix x y
    assume x ∈ A2 y ∈ A2 x ≠ y
    then obtain xa ya where xy: x = xa × (bc ∩ bd) xa ∈ a1 y = ya × (bc ∩
bd) ya ∈ a1
      by(auto simp: A2-def)
    with a1(3) ‹x ≠ y› have xa ∩ ya = {}
      by(auto simp: disjoint-def)
    thus disjnt x y
      by(auto simp: xy disjnt-def)
  qed
  have [simp]: disjoint A3
  proof
    fix x y
    assume x ∈ A3 y ∈ A3 x ≠ y
    then obtain xb yb where xy:x = (ac ∩ ad) × xb xb ∈ a2 y = (ac ∩ ad) ×
yb yb ∈ a2
      by(auto simp: A3-def)
    with a2(3) ‹x ≠ y› have xb ∩ yb = {}
      by(auto simp: disjoint-def)
    thus disjnt x y
      by(auto simp: xy disjnt-def)
  qed
  show ?thesis
   by(auto intro!: disjoint-union) (insert a1 a2,auto simp: A1-def A2-def A3-def)
qed
have fin: finite (A1 ∪ A2 ∪ A3)
  using a1 a2 by (auto simp: A1-def A2-def A3-def finite-image-set2)
have cdeq:c − d = ⋃ (A1 ∪ A2 ∪ A3)
proof −
  have [simp]:⋃ a1 × ⋃ a2 = ⋃ A1 ⋃ a1 × (bc ∩ bd) = ⋃ A2 (ac ∩ ad)
× ⋃ a2 = ⋃ A3
    by (auto simp: A1-def A2-def A3-def)
  show ?thesis
    using a1(4) a2(4) by(simp add: eq1)
qed
have A1 ∪ A2 ∪ A3 ⊆ {a × b |a b. a ∈ sets X ∧ b ∈ sets Y}
  using a1(1) a2(1) cd by(auto simp: A1-def A2-def A3-def)
with fin disj cdeq show ∃ C⊆{a × b |a b. a ∈ sets X ∧ b ∈ sets Y}. finite C ∧
```

disjoint C ∧ *c* − *d* = ⋃ *C*
by (*auto intro*!: *exI*[**where** *x*=*A1* ∪ *A2* ∪ *A3*])
qed *auto*

lemma *sets-pair-restrict-space*:
sets (*restrict-space X A* ⨂$_M$ *restrict-space Y B*) = *sets* (*restrict-space* (*X* ⨂$_M$ *Y*) (*A* × *B*))
(**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = *sigma-sets* (*space* (*restrict-space X A*) × *space* (*restrict-space Y B*)) {*a* × *b* |*a b*. *a* ∈ *sets* (*restrict-space X A*) ∧ *b* ∈ *sets* (*restrict-space Y B*)}
by(*simp add*: *sets-pair-measure*)
also have ... = *sigma-sets* (*space* (*restrict-space X A*) × *space* (*restrict-space Y B*)) {*a* × *b* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
proof −
have {*a* × *b* |*a b*. *a* ∈ *sets* (*restrict-space X A*) ∧ *b* ∈ *sets* (*restrict-space Y B*)} = {*a* × *b* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
unfolding *space-restrict-space sets-restrict-space*
proof *safe*
fix *xa xb*
show *xa* ∈ *sets X* ⟹ *xb* ∈ *sets Y* ⟹
∃ *a b*. (*A* ∩ *xa*) × (*B* ∩ *xb*) = *a* × *b* ∩ (*A* ∩ *space X*) × (*B* ∩ *space Y*) ∧ *a* ∈ *sets X* ∧ *b* ∈ *sets Y*
by(*auto intro*!: *exI*[**where** *x*=*xa*] *exI*[**where** *x*=*xb*] *dest*:*sets.sets-into-space*)
next
fix *a b*
show *a* ∈ *sets X* ⟹ *b* ∈ *sets Y* ⟹
∃ *aa ba*. *a* × *b* ∩ (*A* ∩ *space X*) × (*B* ∩ *space Y*) = *aa* × *ba* ∧ *aa* ∈ *sets.restricted-space X A* ∧ *ba* ∈ *sets.restricted-space Y B*
by(*auto intro*!: *exI*[**where** *x*=*a* ∩ *A*] *exI*[**where** *x*=*b* ∩ *B*] *dest*:*sets.sets-into-space*)
qed
thus *?thesis* **by** *simp*
qed
also have ... = *sigma-sets* (*space* (*restrict-space X A*) × *space* (*restrict-space Y B*)) {(λ*x*. *x*) −‘ *c* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*c*. *c* ∈ {*a* × *b*| *a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}}
proof −
have {*a* × *b* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} = {(λ*x*. *x*) −‘ *c* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*c*. *c* ∈ {*a* × *b*| *a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}}
by *auto*
thus *?thesis* **by** *simp*
qed
also have ... = {(λ*x*. *x*) −‘ *c* ∩ *space* (*restrict-space X A*) × *space* (*restrict-space Y B*) |*c*. *c* ∈ *sigma-sets* (*space X* × *space Y*) {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}}
by(*rule sigma-sets-vimage-commute*[*symmetric*]) (*auto simp*: *space-restrict-space*)

also have ... = $\{c \cap\ (A \cap \mathit{space}\ X) \times (B \cap \mathit{space}\ Y)\ |c.\ c \in \mathit{sets}\ (X \bigotimes_M\ Y)\}$
by(*simp add*: *space-restrict-space sets-pair-measure*)
also have ... = $\{c \cap A \times B\ |c.\ c \in \mathit{sets}\ (X \bigotimes_M\ Y)\}$
using *sets.sets-into-space*[*of* - $X \bigotimes_M\ Y$,*simplified space-pair-measure*] **by** *blast*
also have ... = *?rhs*
by(*auto simp*: *sets-restrict-space*)
finally show *?thesis* **.**
qed

lemma *restrict-space-space*[*simp*]: *restrict-space M* (*space M*) = *M*
by(*auto intro*!: *measure-eqI simp*: *sets-restrict-space emeasure-restrict-space sets.sets-into-space*)

lemma *atMostq-Int-stable*:
Int-stable $\{\{..r\}\ |r{::}\mathit{real}.\ r \in \mathbb{Q}\}$
by(*auto simp*: *Int-stable-def min-def*)

lemma *rborel-eq-atMostq*:
borel = *sigma UNIV* $\{\ \{..r\}\ |\ r{::}\mathit{real}.\ r \in \mathbb{Q}\}$
proof(*safe intro*!: *borel-eq-sigmaI1*[*OF borel-eq-atMost*,**where** *F*=*id*,*simplified*])
fix *a* :: *real*
interpret *s*: *sigma-algebra UNIV sigma-sets UNIV* $\{\{..r\}\ |r.\ r \in \mathbb{Q}\}$
by(*auto intro*!: *sigma-algebra-sigma-sets*)
have [*simp*]: $\{..a\} = (\bigcap\ ((\lambda r.\ \{..r\})\ `\ \{r.\ r \in \mathbb{Q} \wedge a \leq r\}))$
by *auto* (*metis Rats-dense-in-real less-le-not-le nle-le*)
show $\{..a\} \in$ *sigma-sets UNIV* $\{\{..r\}\ |r.\ r \in \mathbb{Q}\}$
using *countable-Collect countable-rat Rats-no-top-le*
by(*auto intro*!: *s.countable-INT'*)
qed *auto*

corollary *rborel-eq-atMostq-sets*:
sets borel = *sigma-sets UNIV* $\{\{..r\}\ |r{::}\mathit{real}.\ r \in \mathbb{Q}\}$
by(*simp add*: *rborel-eq-atMostq*)

lemma *mono-absolutely-continuous*:
assumes *sets* μ = *sets* ν $\bigwedge A.\ A \in \mathit{sets}\ \mu \Longrightarrow \mu\ A \leq \nu\ A$
shows *absolutely-continuous* ν μ
by(*auto simp*: *absolutely-continuous-def*) (*metis assms*(*1*) *assms*(*2*) *fmeasurableD fmeasurableI-null-sets le-zero-eq null-setsD1 null-setsI*)

lemma *ex-measure-countable-space*:
assumes *countable* (*space X*)
and *sets X* = *Pow* (*space X*)
shows $\exists \mu.$ *sets* μ = *sets X* $\wedge$ $(\forall x{\in}\mathit{space}\ X.\ \mu\ \{x\} = f\ x)$
proof −
define μ **where** $\mu \equiv$ *extend-measure* (*space X*) (*space X*) $(\lambda x.\ \{x\})$ *f*
have *s*:*sets* μ = *sets X*
using *sets-extend-measure*[*of* $\lambda x.\ \{x\}$ *space X space X*] *sigma-sets-singletons*[*OF assms*(*1*)]
by(*auto simp add*: μ-*def assms*(*2*))

```
  show ?thesis
  proof(safe intro!: exI[where x=μ])
    fix x
    assume x:x ∈ space X
    show μ {x} = f x
    proof(cases finite (space X))
      case fin:True
      then have sets-fin:x ∈ sets μ ⟹ finite x for x
        by(auto intro!: rev-finite-subset[OF fin] sets.sets-into-space simp: s)
      define μ' where μ' ≡ (λA. ∑ x∈A. f x)
      show ?thesis
      proof(rule emeasure-extend-measure[of μ space X space X - f μ' x])
        show countably-additive (sets μ) μ'
          using fin sets-fin
         by(auto intro!: sets.countably-additiveI-finite simp: sets-eq-imp-space-eq[OF
s] positive-def μ'-def additive-def comm-monoid-add-class.sum.union-disjoint)
      qed(auto simp: x μ-def μ'-def positive-def)
    next
      case inf:False
       define μ' where μ' ≡ (λA. ∑ n. if from-nat-into (space X) n ∈ A then f
(from-nat-into (space X) n) else 0)
      show ?thesis
      proof(rule emeasure-extend-measure[of μ space X space X - f μ' x])
        fix i
        assume i ∈ space X
        then obtain n where n:from-nat-into (space X) n = i
       using bij-betw-from-nat-into[OF assms(1) inf] by (meson f-the-inv-into-f-bij-betw)
        then have μ' {i} =  (∑ m. if m = n then f (from-nat-into (space X) n)
else 0)
          using from-nat-into-inj-infinite[OF assms(1) inf]
          by(auto simp: μ'-def) metis
        also have ... = (∑ m. if (m + (Suc n)) = n then f (from-nat-into (space
X) n) else 0) + (∑ m<Suc n. if m = n then f (from-nat-into (space X) n) else
0)
          by(rule suminf-offset) auto
        also have ... = f i
          by(auto simp: n)
        finally show μ' {i} = f i .
      next
        show countably-additive (sets μ) μ'
        proof(rule countably-additiveI)
          fix A :: nat ⇒ -
          assume h:range A ⊆ sets μ disjoint-family A ⋃ (range A) ∈ sets μ
          show (∑ i. μ' (A i)) = μ' (⋃ (range A))
          proof −
            have (∑ i. μ' (A i)) = (∫⁺ i. μ' (A i) ∂(count-space UNIV))
              by(simp add: nn-integral-count-space-nat)
            also have ... = (∫⁺ i. (∑ n. if from-nat-into (space X) n ∈ A i then f
(from-nat-into (space X) n) else 0) ∂(count-space UNIV))
```

```
    by(simp add: μ'-def)
  also have ... = (∑ n. (∫⁺ i. (if from-nat-into (space X) n ∈ A i then f
(from-nat-into (space X) n) else 0) ∂(count-space UNIV)))
    by(simp add: nn-integral-suminf)
  also have ... = (∑ n. (∫⁺ i. f (from-nat-into (space X) n) * indicator
(A i) (from-nat-into (space X) n) ∂(count-space UNIV)))
    by(auto intro!: suminf-cong nn-integral-cong)
  also have ... = (∑ n. (∑ i. f (from-nat-into (space X) n) * indicator
(A i) (from-nat-into (space X) n)))
    by(simp add: nn-integral-count-space-nat)
  also have ... = (∑ n. f (from-nat-into (space X) n) * indicator (⋃
(range A)) (from-nat-into (space X) n))
    by(simp add: suminf-indicator[OF h(2)])
  also have ... = μ' (⋃ (range A))
    by(auto simp: μ'-def intro!: suminf-cong)
  finally show ?thesis .
 qed
 qed
 qed(auto simp: x μ-def μ'-def positive-def)
 qed
 qed(simp-all add: s)
qed

lemma ex-prob-space-countable:
 assumes space X ≠ {} countable (space X)
   and sets X = Pow (space X)
 shows ∃μ. sets μ = sets X ∧ prob-space μ
proof(cases finite (space X))
 case fin:True
 define n where n ≡ card (space X)
 with fin assms(1) have n: 0 < n
  by(simp add: card-gt-0-iff)
 obtain μ where μ: sets μ = sets X ⋀x. x ∈ space X ⟹ μ {x} = ennreal (1 /
real n)
  using ex-measure-countable-space[OF assms(2,3)] by meson
 then have sets-fin:x ∈ sets μ ⟹ finite x for x
  by(auto intro!: rev-finite-subset[OF fin] sets.sets-into-space)
 show ?thesis
 proof(safe intro!: exI[where x=μ])
  show prob-space μ
  proof
   have emeasure μ (space μ) = (∑ a∈space μ. ennreal (1/n))
    using emeasure-eq-sum-singleton[OF sets-fin[OF sets.top],of μ] assms(3) μ
    by auto
   also have ... = of-nat n * ennreal (1 / real n)
    using μ(2) sets-eq-imp-space-eq[OF μ(1)] by(simp add: n-def)
   also have ... = 1
     using n by(auto simp: ennreal-of-nat-eq-real-of-nat) (metis ennreal-1 en-
nreal-mult'' mult.commute nonzero-eq-divide-eq not-gr0 of-nat-0-eq-iff of-nat-0-le-iff)
```

finally show *emeasure* μ (*space* μ) = *1* **.**
qed
qed(*use* μ **in** *auto*)
next
case *inf*:*False*
obtain μ **where** μ: *sets* μ = *sets X* $\bigwedge x.\ x \in$ *space X* $\Longrightarrow$ μ {x} = (*1*/*2*)^(*Suc* (*to-nat-on* (*space X*) *x*))
using *ex-measure-countable-space*[*OF assms*(*2*,*3*),*of* $\lambda x.$ (*1*/*2*)^(*Suc* (*to-nat-on* (*space X*) *x*))] **by** *auto*
show *?thesis*
proof(*safe intro*!: *exI*[**where** *x*=μ])
show *prob-space* μ
proof
have *emeasure* μ (*space* μ) = *emeasure* μ ($\bigcup n.$ {*from-nat-into* (*space X*) *n*})
by(*simp add*: *sets-eq-imp-space-eq*[*OF* μ(*1*)] *UNION-singleton-eq-range assms*(*1*) *assms*(*2*))
also have ... = ($\sum n.$ μ {*from-nat-into* (*space X*) *n*})
using *from-nat-into-inj-infinite*[*OF assms*(*2*) *inf*] *from-nat-into*[*OF assms*(*1*)] *assms*(*3*)
by(*auto intro*!: *suminf-emeasure*[*symmetric*] *simp*: μ(*1*) *disjoint-family-on-def*)
also have ... = ($\sum n.$ (*1*/*2*)^(*Suc n*))
by(*simp add*: μ(*2*)[*OF from-nat-into*[*OF assms*(*1*)]] *to-nat-on-from-nat-into-infinite*[*OF assms*(*2*) *inf*])
also have ... = ($\sum i.$ *ennreal* ((*1* / *2*) ^ *Suc i*))
by (*metis* (*mono-tags*, *opaque-lifting*) *divide-ennreal divide-pos-pos ennreal-numeral ennreal-power le-less power-0 zero-less-numeral zero-less-one*)
also have ... = *1*
using *suminf-ennreal-eq*[*OF - power-half-series*]
by (*metis ennreal-1 zero-le-divide-1-iff zero-le-numeral zero-le-power*)
finally show *emeasure* μ (*space* μ) = *1* **.**
qed
qed(*use* μ **in** *auto*)
qed

lemma *AE-I''*:
assumes $N \in$ *null-sets M*
and $\bigwedge x.\ x \in$ *space M* $\Longrightarrow$ $x \notin N \Longrightarrow$ *P x*
shows *AE x in M. P x*
by (*metis* (*no-types*, *lifting*) *assms eventually-ae-filter mem-Collect-eq subsetI*)

lemma *absolutely-continuous-trans*:
assumes *absolutely-continuous L M absolutely-continuous M N*
shows *absolutely-continuous L N*
using *assms* **by**(*auto simp*: *absolutely-continuous-def*)

1.2 Equivalence of Measures

abbreviation *equivalence-measure* :: *'a measure* $\Rightarrow$ *'a measure* $\Rightarrow$ *bool* (**infix** $\langle \sim_M \rangle$ *60*)

where *equivalence-measure M N* ≡ *absolutely-continuous M N* ∧ *absolutely-continuous N M*

lemma *equivalence-measure-refl*: $M \sim_M M$
by(*auto simp*: *absolutely-continuous-def*)

lemma *equivalence-measure-sym*:
assumes $M \sim_M N$
shows $N \sim_M M$
using *assms* **by** *simp*

lemma *equivalence-measure-trans*:
assumes $M \sim_M N$ $N \sim_M L$
shows $M \sim_M L$
using *assms* **by**(*auto simp*: *absolutely-continuous-def*)

lemma *equivalence-measureI*:
assumes *absolutely-continuous M N absolutely-continuous N M*
shows $M \sim_M N$
by(*simp add*: *assms*)

end

2 Disintegration Theorem

theory *Disintegration*
imports *S-Finite-Measure-Monad.Kernels*
Lemmas-Disintegration
begin

2.1 Definition 14.D.2. (Mixture and Disintegration)

context *measure-kernel*
begin

definition *mixture-of* :: [('*a* × '*b*) *measure*, '*a measure*] ⇒ *bool* **where**
mixture-of ν μ ⟷ *sets* ν = *sets* ($X \bigotimes_M Y$) ∧ *sets* μ = *sets* X ∧ (∀ C∈*sets* ($X \bigotimes_M Y$). $\nu\ C = (\int^+ x.\ \int^+ y.\ \mathit{indicator}\ C\ (x,y)\ \partial(\kappa\ x)\ \partial\mu)$)

definition *disintegration* :: [('*a* × '*b*) *measure*, '*a measure*] ⇒ *bool* **where**
disintegration ν μ ⟷ *sets* ν = *sets* ($X \bigotimes_M Y$) ∧ *sets* μ = *sets* X ∧ (∀ A∈*sets* X. ∀ B∈*sets* Y. $\nu\ (A \times B) = (\int^+ x{\in}A.\ (\kappa\ x\ B)\ \partial\mu)$)

lemma *disintegrationI*:
assumes *sets* ν = *sets* ($X \bigotimes_M Y$) *sets* μ = *sets* X
and ⋀A B. $A \in$ *sets* $X \Longrightarrow B \in$ *sets* $Y \Longrightarrow \nu\ (A \times B) = (\int^+ x{\in}A.\ (\kappa\ x\ B)\ \partial\mu)$
shows *disintegration* ν μ
by(*simp add*: *disintegration-def assms*)

lemma *mixture-of-disintegration*:
assumes *mixture-of* ν μ
shows *disintegration* ν μ
unfolding *disintegration-def*
proof *safe*
fix *A B*
assume [*simp*]:$A \in$ *sets X B* $\in$ *sets Y*
have [*simp,measurable-cong*]: *sets* μ = *sets X space* μ = *space X*
using *assms* **by**(*auto simp*: *mixture-of-def intro*!: *sets-eq-imp-space-eq*)
have *A* $\times$ *B* $\in$ *sets* (*X* $\bigotimes_M$ *Y*) **by** *simp*
with *assms* **have** ν (*A* $\times$ *B*) = ($\int^+ x$. $\int^+ y$. *indicator* (*A* $\times$ *B*) (*x*,*y*) $\partial(\kappa\ x)\ \partial\mu$)
by(*simp add*: *mixture-of-def*)
also have ... = ($\int^+ x$. $\int^+ y$. *indicator A x* $*$ *indicator B y* $\partial(\kappa\ x)\ \partial\mu$)
by(*simp add*: *indicator-times*)
also have ... = ($\int^+ x{\in}A$. (κ *x B*) $\partial\mu$)
by(*auto intro*!: *nn-integral-cong simp*: *kernel-sets nn-integral-cmult-indicator mult.commute*)
finally show *emeasure* ν (*A* $\times$ *B*) = ($\int^+ x{\in}A$. *emeasure* (κ *x*) *B*$\partial\mu$) **.**
qed(*use assms*[*simplified mixture-of-def*] **in** *auto*)

lemma
shows *mixture-of-sets-eq*: *mixture-of* ν μ $\Longrightarrow$ *sets* ν = *sets* (*X* $\bigotimes_M$ *Y*) *mixture-of* ν μ $\Longrightarrow$ *sets* μ = *sets X*
and *mixture-of-space-eq*: *mixture-of* ν μ $\Longrightarrow$ *space* ν = *space* (*X* $\bigotimes_M$ *Y*) *mixture-of* ν μ $\Longrightarrow$ *space* μ = *space X*
and *disintegration-sets-eq*: *disintegration* ν μ $\Longrightarrow$ *sets* ν = *sets* (*X* $\bigotimes_M$ *Y*) *disintegration* ν μ $\Longrightarrow$ *sets* μ = *sets X*
and *disintegration-space-eq*: *disintegration* ν μ $\Longrightarrow$ *space* ν = *space* (*X* $\bigotimes_M$ *Y*) *disintegration* ν μ $\Longrightarrow$ *space* μ = *space X*
by(*auto simp*: *mixture-of-def disintegration-def intro*!: *sets-eq-imp-space-eq*)

lemma
shows *mixture-ofD*: *mixture-of* ν μ $\Longrightarrow$ *C* $\in$ *sets* (*X* $\bigotimes_M$ *Y*) $\Longrightarrow$ ν *C* = ($\int^+ x$. $\int^+ y$. *indicator C* (*x*,*y*) $\partial(\kappa\ x)\ \partial\mu$)
and *disintegrationD*: *disintegration* ν μ $\Longrightarrow$ *A* $\in$ *sets X* $\Longrightarrow$ *B* $\in$ *sets Y* $\Longrightarrow$ ν (*A* $\times$ *B*) = ($\int^+ x{\in}A$. (κ *x B*) $\partial\mu$)
by(*auto simp*: *mixture-of-def disintegration-def*)

lemma *disintegration-restrict-space*:
assumes *disintegration* ν μ *A* $\cap$ *space X* $\in$ *sets X*
shows *measure-kernel.disintegration* (*restrict-space X A*) *Y* κ (*restrict-space* ν (*A* $\times$ *space Y*)) (*restrict-space* μ *A*)
proof(*rule measure-kernel.disintegrationI*[*OF restrict-measure-kernel*[*of A*]])
have *sets* (*restrict-space* ν (*A* $\times$ *space Y*)) = *sets* (*restrict-space* (*X* $\bigotimes_M$ *Y*) (*A* $\times$ *space Y*))
by(*auto simp*: *disintegration-sets-eq*[*OF assms*(*1*)] *intro*!: *sets-restrict-space-cong*)
also have ... = *sets* (*restrict-space X A* $\bigotimes_M$ *Y*)
using *sets-pair-restrict-space*[*of X A Y space Y*]

by *simp*
finally show *sets* (*restrict-space* ν ($A \times$ *space* Y)) = *sets* (*restrict-space* X A $\bigotimes_M$ Y) **.**
next
show *sets* (*restrict-space* μ A) = *sets* (*restrict-space* X A)
by(*auto simp*: *disintegration-sets-eq*[*OF assms*(*1*)] *intro*!: *sets-restrict-space-cong*)
next
fix a b
assume *h*:$a \in$ *sets* (*restrict-space* X A) $b \in$ *sets* Y
then have *restrict-space* ν ($A \times$ *space* Y) ($a \times b$) = ν ($a \times b$)
using *sets.sets-into-space*
by(*auto intro*!: *emeasure-restrict-space simp*: *disintegration-space-eq*[*OF assms*(*1*)] *disintegration-sets-eq*[*OF assms*(*1*)] *sets-restrict-space*)
(*metis Sigma-Int-distrib1 assms*(*2*) *pair-measureI sets.top space-pair-measure*)
also have ... = ($\int^+ x \in a$. *emeasure* (κ x) b $\partial\mu$)
using *sets-restrict-space-iff*[*OF assms*(*2*)] *h assms*(*1*)
by(*auto simp*: *disintegration-def*)
also have ... = ($\int^+ x \in A$. (*emeasure* (κ x) b $*$ *indicator* a x) $\partial\mu$)
using *h*(*1*) **by**(*auto intro*!: *nn-integral-cong simp*: *sets-restrict-space*)
(*metis IntD1 indicator-simps*(*1*) *indicator-simps*(*2*) *mult.comm-neutral mult-zero-right*)
also have ... = ($\int^+ x \in a$. *emeasure* (κ x) b ∂*restrict-space* μ A)
by (*metis* (*no-types*, *lifting*) *assms disintegration-sets-eq*(*2*) *disintegration-space-eq*(*2*) *nn-integral-cong nn-integral-restrict-space*)
finally show *restrict-space* ν ($A \times$ *space* Y) ($a \times b$) = ($\int^+ x \in a$. *emeasure* (κ x) b ∂*restrict-space* μ A) **.**
qed

end

context *subprob-kernel*
begin
lemma *countable-disintegration-AE-unique*:
assumes *countable* (*space* Y) **and** [*measurable-cong*]:*sets* Y = *Pow* (*space* Y)
and *subprob-kernel* X Y κ' *sigma-finite-measure* μ
and *disintegration* ν μ *measure-kernel.disintegration* X Y κ' ν μ
shows *AE* x *in* μ. κ x = κ' x
proof −
interpret κ': *subprob-kernel* X Y κ' **by** *fact*
interpret *s*: *sigma-finite-measure* μ **by** *fact*
have *sets-eq*[*measurable-cong*]: *sets* μ = *sets* X *sets* ν = *sets* ($X \bigotimes_M Y$)
using *assms*(*5*) **by**(*auto simp*: *disintegration-def*)
have *1*:*AE* x *in* μ. $\forall y \in$ *space* Y. κ x $\{y\}$ = κ' x $\{y\}$
unfolding *AE-ball-countable*[*OF assms*(*1*)]
proof
fix y
assume y: $y \in$ *space* Y
show *AE* x *in* μ. *emeasure* (κ x) $\{y\}$ = *emeasure* (κ' x) $\{y\}$
proof(*rule s.sigma-finite*)

fix $J :: nat \Rightarrow$ -
assume J:*range* $J \subseteq$ *sets* μ $\bigcup$ (*range* J) = *space* μ $\bigwedge i$. *emeasure* μ (J i) $\neq \infty$
from y **have** [*measurable*]: $(\lambda x.\ \kappa\ x\ \{y\}) \in$ *borel-measurable* X $(\lambda x.\ \kappa'\ x\ \{y\}) \in$ *borel-measurable* X
using *emeasure-measurable* κ'.*emeasure-measurable* **by** *auto*
define A **where** $A \equiv \{x \in$ *space* $\mu.\ \kappa\ x\ \{y\} \leq \kappa'\ x\ \{y\}\}$
have [*measurable*]:$A \in$ *sets* μ
by(*auto simp*: *A-def*)
have A: $\bigwedge x.\ x \in$ *space* $\mu \Longrightarrow x \notin A \Longrightarrow\ \kappa'\ x\ \{y\} \leq \kappa\ x\ \{y\}$
by(*auto simp*: *A-def*)
have *1*: *AE* $x{\in}A$ *in* $\mu.\ \kappa\ x\ \{y\} = \kappa'\ x\ \{y\}$
proof −
have *AE* x *in* $\mu.\ \forall n.\ (x \in A \cap J\ n \longrightarrow \kappa\ x\ \{y\} = \kappa'\ x\ \{y\})$
unfolding *AE-all-countable*
proof
fix n
have *ninf*:$(\int^+ x{\in}A \cap J\ n.\ (\kappa\ x)\ \{y\}\partial\mu) < \infty$
proof −
have $(\int^+ x{\in}A \cap J\ n.\ (\kappa\ x)\ \{y\}\partial\mu) \leq (\int^+ x{\in}A \cap J\ n.\ (\kappa\ x)$ (*space* Y) $\partial\mu)$
using *kernel-sets* y **by**(*auto intro*!: *nn-integral-mono emeasure-mono simp*: *indicator-def disintegration-space-eq*(*2*)[*OF assms*(*5*)])
also have ... $\leq (\int^+ x{\in}A \cap J\ n.\ 1\ \partial\mu)$
using *subprob-space* **by**(*auto intro*!: *nn-integral-mono simp*: *indicator-def disintegration-space-eq*(*2*)[*OF assms*(*5*)])
also have ... $= \mu\ (A \cap J\ n)$
using J **by** *simp*
also have ... $\leq \mu\ (J\ n)$
using J **by** (*auto intro*!: *emeasure-mono*)
also have ... $< \infty$
using *J*(*3*)[*of n*] **by** (*simp add*: *top.not-eq-extremum*)
finally show *?thesis* **.**
qed
have $(\int^+ x.\ (\kappa'\ x)\ \{y\} * indicator\ (A \cap J\ n)\ x - (\kappa\ x)\ \{y\} * indicator\ (A \cap J\ n)\ x\ \partial\mu) = (\int^+ x{\in}A \cap J\ n.\ (\kappa'\ x)\ \{y\}\ \partial\mu) - (\int^+ x{\in}A \cap J\ n.\ (\kappa\ x)\ \{y\}\ \ \partial\mu)$
using J *ninf* **by**(*auto intro*!: *nn-integral-diff simp*: *indicator-def A-def*)
also have ... = *0*
proof −
have *0*: $\nu\ ((A \cap J\ n) \times \{y\}) = (\int^+ x{\in}A \cap J\ n.\ (\kappa\ x)\ \{y\}\ \partial\mu)$
using J y *sets-eq* **by**(*auto intro*!: *disintegrationD*[*OF assms*(*5*),*of A* $\cap$ J n $\{y\}$])
have [*simp*]: $(\int^+ x{\in}A \cap J\ n.\ (\kappa'\ x)\ \{y\}\ \partial\mu) = \nu\ ((A \cap J\ n) \times \{y\})$
using J y *sets-eq* **by**(*auto intro*!: κ'.*disintegrationD*[*OF assms*(*6*),*of A* $\cap$ J n $\{y\}$,*symmetric*])
show *?thesis*
using *ninf* **by** (*simp add*: *0 diff-eq-0-iff-ennreal*)
qed
finally have *assm*:*AE* x *in* $\mu.\ (\kappa'\ x)\ \{y\} * indicator\ (A \cap J\ n)\ x - (\kappa\ x)$

```
{y} * indicator (A ∩ J n) x = 0
          using J by(simp add: nn-integral-0-iff-AE)
        show AE x∈A ∩ J n in μ. (κ x) {y} = (κ' x) {y}
        proof(rule AE-mp[OF assm])
        show AE x in μ. emeasure (κ' x) {y} * indicator (A ∩ J n) x − emeasure
(κ x) {y} * indicator (A ∩ J n) x = 0 ⟶ x ∈ A ∩ J n ⟶ emeasure (κ x) {y}
= emeasure (κ' x) {y}
          proof −
            {
              fix x
              assume h: (κ' x) {y} − (κ x) {y} = 0 x ∈ A
              have (κ x) {y} = (κ' x) {y}
                 using h(2) by(auto intro!: antisym ennreal-minus-eq-0[OF h(1)]
simp: A-def)
            }
            thus ?thesis
              by(auto simp: indicator-def)
          qed
        qed
      qed
      hence AE x∈A ∩ (⋃ (range J))in μ. κ x {y} = κ' x {y}
        by auto
      thus ?thesis
        using J(2) by auto
    qed
    have 2: AE x∈ (space μ − A) in μ. κ x {y} = κ' x {y}
    proof −
      have AE x in μ. ∀ n. x ∈ (space μ − A) ∩ J n ⟶ κ x {y} = κ' x {y}
        unfolding AE-all-countable
      proof
        fix n
        have ninf:(∫⁺x∈(space μ − A) ∩ J n. (κ' x) {y}∂μ) < ∞
        proof −
         have (∫⁺x∈(space μ − A) ∩ J n. (κ' x) {y}∂μ) ≤ (∫⁺x∈(space μ − A)
∩ J n. (κ' x) (space Y) ∂μ)
            using kernel-sets y by(auto intro!: nn-integral-mono emeasure-mono
simp: indicator-def disintegration-space-eq(2)[OF assms(5)])
          also have ... ≤ (∫⁺x∈(space μ − A) ∩ J n. 1 ∂μ)
            using κ'.subprob-space by(auto intro!: nn-integral-mono simp: indica-
tor-def disintegration-space-eq(2)[OF assms(5)])
          also have ... = μ ((space μ − A) ∩ J n)
            using J by simp
          also have ... ≤ μ (J n)
            using J by (auto intro!: emeasure-mono)
          also have ... < ∞
            using J(3)[of n] by (simp add: top.not-eq-extremum)
          finally show ?thesis .
        qed
       have (∫⁺x. (κ x) {y} * indicator ((space μ − A) ∩ J n) x − (κ' x) {y} *
```

indicator ((*space* μ − *A*) ∩ *J n*) *x* ∂μ) = (∫$^+$*x*∈(*space* μ − *A*) ∩ *J n*. (κ *x*) {*y*} ∂μ) − (∫$^+$*x*∈(*space* μ − *A*) ∩ *J n*. (κ' *x*) {*y*} ∂μ)
using *J ninf A* **by**(*auto intro*!: *nn-integral-diff simp*: *indicator-def*)
also have ... = *0*
proof −
have *0*: (∫$^+$*x*∈(*space* μ − *A*) ∩ *J n*. (κ' *x*) {*y*} ∂μ) = ν (((*space* μ − *A*) ∩ *J n*) × {*y*})
using *J y sets-eq* **by**(*auto intro*!: κ'.*disintegrationD*[*OF assms*(*6*),*of* (*space* μ − *A*) ∩ *J n* {*y*},*symmetric*])
have [*simp*]: ν (((*space* μ − *A*) ∩ *J n*) × {*y*}) = (∫$^+$*x*∈(*space* μ − *A*) ∩ *J n*. (κ *x*) {*y*} ∂μ)
using *J y sets-eq* **by**(*auto intro*!: *disintegrationD*[*OF assms*(*5*),*of* (*space* μ − *A*) ∩ *J n* {*y*}])
show *?thesis*
using *ninf* **by** (*simp add*: *0 diff-eq-0-iff-ennreal*)
qed
finally have *assm*:*AE x in* μ. (κ *x*) {*y*} ∗ *indicator* ((*space* μ − *A*) ∩ *J n*) *x* − (κ' *x*) {*y*} ∗ *indicator* ((*space* μ − *A*) ∩ *J n*) *x* = *0*
using *J* **by**(*simp add*: *nn-integral-0-iff-AE*)
show *AE x*∈(*space* μ − *A*) ∩ *J n in* μ. (κ *x*) {*y*} = (κ' *x*) {*y*}
proof(*rule AE-mp*[*OF assm*])
show *AE x in* μ. *emeasure* (κ *x*) {*y*} ∗ *indicator* ((*space* μ − *A*) ∩ *J n*) *x* − *emeasure* (κ' *x*) {*y*} ∗ *indicator* ((*space* μ − *A*) ∩ *J n*) *x* = *0* ⟶ *x* ∈ (*space* μ − *A*) ∩ *J n* ⟶ *emeasure* (κ *x*) {*y*} = *emeasure* (κ' *x*) {*y*}
proof −
{
fix *x*
assume *h*: (κ *x*) {*y*} − (κ' *x*) {*y*} = *0 x* ∈ *space* μ *x* ∉ *A*
have (κ *x*) {*y*} = (κ' *x*) {*y*}
using *A*[*OF h*(*2*,*3*)] **by**(*auto intro*!: *antisym ennreal-minus-eq-0*[*OF h*(*1*)] *simp*: *A-def*)
}
thus *?thesis*
by(*auto simp*: *indicator-def*)
qed
qed
qed
hence *AE x*∈(*space* μ − *A*) ∩ (⋃ (*range J*))*in* μ. κ *x* {*y*} = κ' *x* {*y*}
by *auto*
thus *?thesis*
using *J*(*2*) **by** *auto*
qed
show *AE x in* μ. κ *x* {*y*} = κ' *x* {*y*}
using *1 2* **by**(*auto simp*: *A-def*)
qed
qed
show *?thesis*
proof(*rule AE-mp*[*OF 1*])
{

```
    fix x
    assume x: x ∈ space X
      and h: ∀ y∈space Y. κ x {y} = κ' x {y}
    have κ x = κ' x
      by (simp add: κ'.kernel-sets assms h kernel-sets measure-eqI-countable x)
  }
  thus  AE x in μ. (∀ y∈space Y. emeasure (κ x) {y} = emeasure (κ' x) {y})
⟶ κ x = κ' x
    by(auto simp: sets-eq-imp-space-eq[OF sets-eq(1)])
 qed
qed

end

lemma(in subprob-kernel) nu-mu-spaceY-le:
 assumes disintegration ν μ A ∈ sets X
 shows ν (A × space Y) ≤ μ A
proof −
 have ν (A × space Y) = (∫⁺x∈A. (κ x (space Y)) ∂μ)
   using assms by(simp add: disintegration-def)
 also have ... ≤ (∫⁺x∈A. 1 ∂μ)
   using assms subprob-space by(auto intro!: nn-integral-mono simp:  disintegration-space-eq) (metis dual-order.refl indicator-simps(1) indicator-simps(2) mult.commute mult-1 mult-zero-right)
 also have ... = μ A
   using assms by (simp add: disintegration-def)
 finally show ?thesis .
qed

context prob-kernel
begin

lemma countable-disintegration-AE-unique-prob:
 assumes countable (space Y) and [measurable-cong]:sets Y = Pow (space Y)
     and prob-kernel X Y κ' sigma-finite-measure μ
     and disintegration ν μ measure-kernel.disintegration X Y κ' ν μ
   shows AE x in μ. κ x = κ' x
 by(auto intro!: countable-disintegration-AE-unique[OF assms(1,2) - assms(4−6)] prob-kernel.subprob-kernel assms(3))

end
```

2.2 Lemma 14.D.3.

```
lemma(in prob-kernel) nu-mu-spaceY:
 assumes disintegration ν μ A ∈ sets X
 shows ν (A × space Y) = μ A
proof −
 have ν (A × space Y) = (∫⁺x∈A. (κ x (space Y)) ∂μ)
```

using *assms* **by**(*simp add*: *disintegration-def*)
also have ... = $(\int^+ x{\in}A.\ 1\ \partial\mu)$
using *assms* **by**(*auto intro*!: *nn-integral-cong simp*: *prob-space disintegration-space-eq*)
also have ... = $\mu\ A$
using *assms* **by** (*simp add*: *disintegration-def*)
finally show *?thesis* **.**
qed

corollary(**in** *subprob-kernel*) *nu-finite*:
assumes *disintegration* ν μ *finite-measure* μ
shows *finite-measure* ν
proof
have ν (*space* ν) = ν (*space* ($X \bigotimes_M Y$))
using *assms* **by**(*simp add*: *disintegration-space-eq*)
also have ... $\leq \mu$ (*space* μ)
using *assms* **by**(*simp add*: *nu-mu-spaceY-le disintegration-space-eq space-pair-measure*)
finally show ν (*space* ν) $\neq \infty$
using *assms*(*2*) **by** (*metis finite-measure.emeasure-finite infinity-ennreal-def neq-top-trans*)
qed

corollary(**in** *subprob-kernel*) *nu-subprob-space*:
assumes *disintegration* ν μ *subprob-space* μ
shows *subprob-space* ν
proof
have ν (*space* ν) = ν (*space* ($X \bigotimes_M Y$))
using *assms* **by**(*simp add*: *disintegration-space-eq*)
also have ... $\leq \mu$ (*space* μ)
using *assms* **by**(*simp add*: *nu-mu-spaceY-le disintegration-space-eq space-pair-measure*)
finally show ν (*space* ν) $\leq$ *1*
using *assms*(*2*) *order.trans subprob-space.emeasure-space-le-1* **by** *auto*
next
show *space* $\nu \neq \{\}$
using *Y-not-empty assms* **by**(*auto simp*: *disintegration-space-eq subprob-space-def subprob-space-axioms-def space-pair-measure*)
qed

corollary(**in** *prob-kernel*) *nu-prob-space*:
assumes *disintegration* ν μ *prob-space* μ
shows *prob-space* ν
proof
have ν (*space* ν) = ν (*space* ($X \bigotimes_M Y$))
using *assms* **by**(*simp add*: *disintegration-space-eq*)
also have ... = μ (*space* μ)
using *assms* **by**(*simp add*: *nu-mu-spaceY disintegration-space-eq space-pair-measure*)
finally show ν (*space* ν) = *1*
by (*simp add*: *assms*(*2*) *prob-space.emeasure-space-1*)
qed

lemma(**in** *subprob-kernel*) *nu-sigma-finite*:
assumes *disintegration* ν μ *sigma-finite-measure* μ
shows *sigma-finite-measure* ν
proof
obtain A **where** A:*countable* A $A \subseteq$ *sets* μ $\bigcup A =$ *space* μ $\forall a{\in}A$. *emeasure* μ $a \neq \infty$
using *assms*(*2*) **by** (*meson sigma-finite-measure.sigma-finite-countable*)
have *countable* $\{a \times$ *space* $Y \mid a.\ a \in A\}$
using *countable-image*[*OF A*(*1*),*of* $\lambda a.\ a \times$ *space* Y]
by (*simp add*: *Setcompr-eq-image*)
moreover have $\{a \times$ *space* $Y \mid a.\ a \in A\} \subseteq$ *sets* ν
using *A*(*2*) *assms*(*1*) *disintegration-def* **by** *auto*
moreover have $\bigcup \{a \times$ *space* $Y \mid a.\ a \in A\} =$ *space* ν
using *assms A*(*3*) **by**(*simp add*: *disintegration-space-eq space-pair-measure*) *blast*
moreover have $\forall b{\in}\{a \times$ *space* $Y \mid a.\ a \in A\}$. *emeasure* ν $b \neq \infty$
using *neq-top-trans*[*OF - nu-mu-spaceY-le*[*OF assms*(*1*)]] *A*(*2*,*4*) *assms disintegration-sets-eq*(*2*) **by** *auto*
ultimately show $\exists B.$ *countable* $B \wedge B \subseteq$ *sets* $\nu \wedge \bigcup B =$ *space* $\nu \wedge (\forall b{\in}B.$ *emeasure* ν $b \neq \infty)$
by *blast*
qed

2.3 Theorem 14.D.4. (Measure Mixture Theorem)

lemma(**in** *measure-kernel*) *exist-nu*:
assumes *sets* $\mu =$ *sets* X
shows $\exists \nu.$ *disintegration* ν μ
proof −
define ν **where** $\nu =$ *extend-measure* (*space* $X \times$ *space* Y) $\{(a, b).\ a \in$ *sets* $X \wedge b \in$ *sets* $Y\}$ $(\lambda(a, b).\ a \times b)$ $(\lambda(a, b).\ \int^+ x{\in}a.$ *emeasure* $(\kappa\ x)$ $b\partial\mu)$
have *1*: *sets* $\nu =$ *sets* $(X \bigotimes_M Y)$
proof −
have *sets* $\nu =$ *sigma-sets* (*space* $X \times$ *space* Y) $((\lambda(a, b).\ a \times b)$ ‘ $\{(a, b).\ a \in$ *sets* $X \wedge b \in$ *sets* $Y\})$
unfolding ν-*def*
by(*rule sets-extend-measure*) (*use sets.space-closed*[*of X*] *sets.space-closed*[*of Y*] **in** *blast*)
also have ... = *sigma-sets* (*space* $X \times$ *space* Y) $\{a \times b \mid a\ b.\ a \in$ *sets* $X \wedge b \in$ *sets* $Y\}$
by(*auto intro*!: *sigma-sets-eqI*)
also have ... = *sets* $(X \bigotimes_M Y)$
by(*simp add*: *sets-pair-measure*)
finally show *?thesis* **.**
qed
have *2*: $\nu\ (A \times B) = (\int^+ x{\in}A.\ (\kappa\ x\ B)\ \partial\mu)$ **if** $A \in$ *sets* X $B \in$ *sets* Y **for** A B
proof(*rule extend-measure-caratheodory-pair*[*OF* ν-*def*])
fix i j k l
assume $i \in$ *sets* $X \wedge j \in$ *sets* Y $k \in$ *sets* $X \wedge l \in$ *sets* Y $i \times j = k \times l$

then consider $i = k\ j = l \mid i \times j = \{\}\ k \times l = \{\}$ **by** *blast*
thus $(\int^+ x{\in}i.\ emeasure\ (\kappa\ x)\ j\ \partial\mu) = (\int^+ x{\in}k.\ emeasure\ (\kappa\ x)\ l\ \partial\mu)$
by *cases auto*
next
fix $A\ B\ j\ k$
assume *h*: $\bigwedge n{::}nat.\ A\ n \in sets\ X \wedge B\ n \in sets\ Y\ j \in sets\ X \wedge k \in sets\ Y$ *disjoint-family* $(\lambda n.\ A\ n \times B\ n)\ (\bigcup i.\ A\ i \times B\ i) = j \times k$
show $(\sum n.\ \int^+ x{\in}A\ n.\ emeasure\ (\kappa\ x)\ (B\ n)\partial\mu) = (\int^+ x{\in}j.\ emeasure\ (\kappa\ x)\ k\partial\mu)$
(**is** *?lhs = ?rhs*)
proof −
have *?lhs* $= (\int^+ x.\ (\sum n.\ \kappa\ x\ (B\ n) * indicator\ (A\ n)\ x)\partial\mu)$
proof(*rule nn-integral-suminf*[*symmetric*])
fix *n*
have [*measurable*]:$(\lambda x.\ emeasure\ (\kappa\ x)\ (B\ n)) \in borel\text{-}measurable\ \mu\ indicator\ (A\ n) \in borel\text{-}measurable\ \mu$
using *h*(*1*)[*of n*] *emeasure-measurable*[*of B n*] *assms*(*1*) **by** *auto*
thus $(\lambda x.\ emeasure\ (\kappa\ x)\ (B\ n) * indicator\ (A\ n)\ x) \in borel\text{-}measurable\ \mu$
by *simp*
qed
also have ... = *?rhs*
proof(*safe intro*!: *nn-integral-cong*)
fix *x*
assume $x \in space\ \mu$
consider $j = \{\} \mid k = \{\} \mid j \neq \{\}\ k \neq \{\}$ **by** *auto*
then show $(\sum n.\ emeasure\ (\kappa\ x)\ (B\ n) * indicator\ (A\ n)\ x) = emeasure\ (\kappa\ x)\ k * indicator\ j\ x$
proof *cases*
case *1*
then have $\bigwedge n.\ A\ n \times B\ n = \{\}$
using *h*(*4*) **by** *auto*
have $emeasure\ (\kappa\ x)\ (B\ n) * indicator\ (A\ n)\ x = 0$ **for** *n*
using ‹$A\ n \times B\ n = \{\}$› **by**(*auto simp*: *Sigma-empty-iff*)
thus *?thesis*
by(*simp only*: *1*,*simp*)
next
case *2*
then have $\bigwedge n.\ A\ n \times B\ n = \{\}$
using *h*(*4*) **by** *auto*
have $emeasure\ (\kappa\ x)\ (B\ n) * indicator\ (A\ n)\ x = 0$ **for** *n*
using ‹$A\ n \times B\ n = \{\}$› **by**(*auto simp*: *Sigma-empty-iff*)
thus *?thesis*
by(*simp only*: *2*,*simp*)
next
case *3*
then have *xinjiff*:$x \in j \longleftrightarrow (\exists i.\ \exists y{\in}B\ i.\ (x,y) \in A\ i \times B\ i)$
using *h*(*4*) **by** *blast*
have *bunk*:$\bigcup\ (B\ `\ \{i.\ x \in A\ i\}) = k$ **if** $x \in j$
using *that 3 h*(*4*) **by** *blast*

```
        show ?thesis
        proof(cases x ∈ j)
          case False
          then have ⋀n. x ∉ A n ∨ B n = {}
            using h(4) 3 xinjiff by auto
          have emeasure (κ x) (B n) * indicator (A n) x = 0 for n
            using ‹x ∉ A n ∨ B n = {}› by auto
          thus ?thesis
            by(simp only:)(simp add: False)
        next
          case True
          then have [simp]: emeasure (κ x) k * indicator j x = emeasure (κ x) k
            by simp
         have (∑ n. emeasure (κ x) (B n) * indicator (A n) x) = (∑ n. emeasure
(κ x) (if x ∈ A n then B n else {}))
            by(auto intro!: suminf-cong)
          also have ... = emeasure (κ x) (⋃ n. if x ∈ A n then B n else {})
          proof(rule suminf-emeasure)
            show disjoint-family (λi. if x ∈ A i then B i else {})
          using disjoint-family-onD[OF h(3)] by (auto simp: disjoint-family-on-def)
          next
            show range (λi. if x ∈ A i then B i else {}) ⊆ sets (κ x)
              using h(1) kernel-sets[of x] ‹x ∈ space μ› sets-eq-imp-space-eq[OF
assms(1)] by auto
          qed
          also have ... = emeasure (κ x) k
            using True by(simp add: bunk)
          finally show ?thesis by simp
        qed
      qed
    qed
    finally show ?thesis .
  qed
  qed(use that in auto)
  show ?thesis
    using 1 2 assms
    by(auto simp: disintegration-def)
qed

lemma(in subprob-kernel) exist-unique-nu-sigma-finite':
  assumes sets μ = sets X sigma-finite-measure μ
  shows ∃!ν. disintegration ν μ
proof −
  obtain ν where disi: disintegration ν μ
    using exist-nu[OF assms(1)] by auto
  with assms(2) interpret sf: sigma-finite-measure ν
    by(simp add: nu-sigma-finite)
  interpret μ: sigma-finite-measure μ by fact
  show ?thesis
```

proof(*rule ex1I*[**where** *a*=ν])
fix ν'
assume *disi'*:*disintegration* ν' μ
show $\nu' = \nu$
proof(*rule* μ*.sigma-finite-disjoint*)
fix *A* :: *nat* ⇒ -
assume *A*: *range A* ⊆ *sets* μ ⋃ (*range A*) = *space* μ ⋀*i*. *emeasure* μ (*A i*) ≠ ∞ *disjoint-family A*
define *B* **where** *B* ≡ λ*i*. *A i* × *space Y*
show $\nu' = \nu$
proof(*rule measure-eqI-generator-eq*[**where** *E*= {*a* × *b*|*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} **and** *A*=*B* **and** Ω=*space X* × *space Y*])
show ⋀*C*. *C* ∈ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*} ⟹ *emeasure* ν' *C* = *emeasure* ν *C*
sets ν' = *sigma-sets* (*space X* × *space Y*) {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
sets ν = *sigma-sets* (*space X* × *space Y*) {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
using *disi disi'* **by**(*auto simp*: *disintegration-def sets-pair-measure*)
next
show *range B* ⊆ {*a* × *b* |*a b*. *a* ∈ *sets X* ∧ *b* ∈ *sets Y*}
⋃ (*range B*) = *space X* × *space Y*
using *A*(*1*,*2*) **by**(*auto simp*: *B-def assms*(*1*) *sets-eq-imp-space-eq*[*OF assms*(*1*)])
next
fix *i*
show *emeasure* ν' (*B i*) ≠ ∞
using *A*(*1*) *nu-mu-spaceY-le*[*OF disi'*,*of A i*] *A*(*3*)[*of i*] **by**(*auto simp*: *B-def assms top.extremum-uniqueI*)
qed(*simp-all add*: *Int-stable-pair-measure-generator pair-measure-closed*)
qed
qed *fact*
qed

lemma(**in** *subprob-kernel*) *exist-unique-nu-sigma-finite*:
assumes *sets* μ = *sets X sigma-finite-measure* μ
shows ∃!ν. *disintegration* ν μ ∧ *sigma-finite-measure* ν
using *assms exist-unique-nu-sigma-finite' nu-sigma-finite* **by** *blast*

lemma(**in** *subprob-kernel*) *exist-unique-nu-finite*:
assumes *sets* μ = *sets X finite-measure* μ
shows ∃!ν. *disintegration* ν μ ∧ *finite-measure* ν
using *assms nu-finite finite-measure.sigma-finite-measure*[*OF assms*(*2*)] *exist-unique-nu-sigma-finite'* **by** *blast*

lemma(**in** *subprob-kernel*) *exist-unique-nu-sub-prob-space*:
assumes *sets* μ = *sets X subprob-space* μ
shows ∃!ν. *disintegration* ν μ ∧ *subprob-space* ν
using *assms nu-subprob-space subprob-space-imp-sigma-finite*[*OF assms*(*2*)] *ex-*

ist-unique-nu-sigma-finite' **by** *blast*

lemma(**in** *prob-kernel*) *exist-unique-nu-prob-space*:
assumes *sets* μ = *sets X prob-space* μ
shows $\exists!\nu$. *disintegration* ν μ $\wedge$ *prob-space* ν
using *assms nu-prob-space prob-space-imp-sigma-finite*[*OF assms*(*2*)] *exist-unique-nu-sigma-finite'*
by *blast*

lemma(**in** *subprob-kernel*) *nn-integral-fst-finite'*:
assumes $f \in$ *borel-measurable* $(X \bigotimes_M Y)$ *disintegration* ν μ *finite-measure* μ
shows $(\int^+ z.\ f\ z\ \partial\nu) = (\int^+ x.\ \int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial\mu)$
using *assms*(*1*)
proof *induction*
case (*cong f g*)
have $integral^N\ \nu\ f = integral^N\ \nu\ g$
using *cong*(*3*) **by**(*auto intro*!: *nn-integral-cong simp*: *disintegration-space-eq*(*1*)[*OF assms*(*2*)])
with *cong*(*3*) **show** *?case*
by(*auto simp*: *cong*(*4*) *kernel-space disintegration-space-eq*(*2*)[*OF assms*(*2*)] *space-pair-measure intro*!: *nn-integral-cong*)
next
case (*set A*)
show *?case*
proof(*rule sigma-sets-induct-disjoint*[*of* $\{a \times b | a\ b.\ a \in sets\ X \wedge b \in sets\ Y\}$ *space X* $\times$ *space Y*])
show $A \in$ *sigma-sets* (*space X* $\times$ *space Y*) $\{a \times b\ |a\ b.\ a \in sets\ X \wedge b \in sets\ Y\}$
using *set* **by**(*simp add*: *sets-pair-measure*)
next
fix *A*
assume $A \in \{a \times b\ |a\ b.\ a \in sets\ X \wedge b \in sets\ Y\}$
then obtain *a b* **where** *ab*: $A = a \times b\ a \in sets\ X\ b \in sets\ Y$
by *auto*
with *assms*(*2*) **have** $integral^N\ \nu\ (indicator\ A) = (\int^+ x \in a.\ (\kappa\ x\ b)\ \partial\mu)$
by(*simp add*: *disintegration-def*)
also have ... = $(\int^+ x.\ \int^+ y.\ indicator\ A\ (x, y)\ \partial\kappa\ x\ \partial\mu)$
by(*auto simp*: *ab*(*1*) *indicator-times disintegration-space-eq*(*2*)[*OF assms*(*2*)] *ab*(*3*) *kernel-sets mult.commute nn-integral-cmult-indicator intro*!: *nn-integral-cong*)
finally show $integral^N\ \nu\ (indicator\ A) = (\int^+ x.\ \int^+ y.\ indicator\ A\ (x, y)\ \partial\kappa\ x\ \partial\mu)$ **.**
next
fix *A*
assume *h*: $A \in$ *sigma-sets* (*space X* $\times$ *space Y*) $\{a \times b\ |a\ b.\ a \in sets\ X \wedge b \in sets\ Y\}$ $integral^N\ \nu\ (indicator\ A) = (\int^+ x.\ \int^+ y.\ indicator\ A\ (x, y)\ \partial\kappa\ x\ \partial\mu)$
show $integral^N\ \nu\ (indicator\ (space\ X \times space\ Y - A)) = (\int^+ x.\ \int^+ y.\ indicator\ (space\ X \times space\ Y - A)\ (x, y)\ \partial\kappa\ x\ \partial\mu)$ (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = $(\int^+ z.\ 1 - indicator\ A\ z\ \partial\nu)$

by(*auto intro*!: *nn-integral-cong simp*: *disintegration-space-eq*(*1*)[*OF assms*(*2*)] *space-pair-measure indicator-def*)
also have ... = $(\int^+ z.\ 1\ \partial\nu) - (\int^+ z.\ indicator\ A\ z\ \partial\nu)$
proof(*rule nn-integral-diff*)
show $integral^N\ \nu\ (indicator\ A) \neq \infty$
using *h*(*1*)[*simplified sets-pair-measure*[*symmetric*]] *disintegration-sets-eq*(*1*)[*OF assms*(*2*)] *finite-measure.emeasure-finite*[*OF nu-finite*[*OF assms*(*2*,*3*)]]
by *auto*
next
show *indicator A* ∈ *borel-measurable ν*
using *h*(*1*)[*simplified sets-pair-measure*[*symmetric*]] *disintegration-sets-eq*(*1*)[*OF assms*(*2*)] **by** *simp*
qed(*simp-all add*: *indicator-def*)
also have ... = $(\int^+ z.\ 1\ \partial\nu) - (\int^+ x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
by(*simp add*: *h*(*2*))
also have ... = $\nu\ (space\ X \times space\ Y) - (\int^+ x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
using *nn-integral-indicator*[*OF sets.top*[*of ν*]] **by**(*simp add*: *space-pair-measure disintegration-space-eq*(*1*)[*OF assms*(*2*)])
also have ... = $(\int^+ x.\ \kappa\ x\ (space\ Y)\ \partial\mu) - (\int^+ x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
proof −
have $\nu\ (space\ X \times space\ Y) = (\int^+ x.\ \kappa\ x\ (space\ Y)\ \partial\mu)$
using *assms*(*2*) **by**(*auto simp*: *disintegration-def disintegration-space-eq*(*2*)[*OF assms*(*2*)] *intro*!: *nn-integral-cong*)
thus *?thesis* **by** *simp*
qed
also have ... = $(\int^+ x. \int^+ y.\ indicator\ (space\ X \times space\ Y)\ (x,\ y)\ \partial\kappa\ x\ \partial\mu) - (\int^+ x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
proof −
have $(\int^+ x.\ \kappa\ x\ (space\ Y)\ \partial\mu) = (\int^+ x. \int^+ y.\ indicator\ (space\ X \times space\ Y)\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
using *kernel-sets* **by**(*auto intro*!: *nn-integral-cong simp*: *indicator-times disintegration-space-eq*(*2*)[*OF assms*(*2*)])
thus *?thesis* **by** *simp*
qed
also have ... = $(\int^+ x. (\int^+ y.\ indicator\ (space\ X \times space\ Y)\ (x,\ y)\ \partial\kappa\ x) - (\int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x)\ \partial\mu)$
proof(*rule nn-integral-diff*[*symmetric*])
show $(\lambda x. \int^+ y.\ indicator\ (space\ X \times space\ Y)\ (x,\ y)\ \partial\kappa\ x) \in borel\text{-}measurable\ \mu$
$(\lambda x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x) \in borel\text{-}measurable\ \mu$
by(*use disintegration-sets-eq*[*OF assms*(*2*)] *nn-integral-measurable-f h*(*1*)[*simplified sets-pair-measure*[*symmetric*]] **in** *auto*)+
next
have $(\int^+ x. \int^+ y.\ indicator\ A\ (x,\ y)\ \partial\kappa\ x\ \partial\mu) \leq (\int^+ x. \int^+ y.\ 1\ \partial\kappa\ x\ \partial\mu)$
by(*rule nn-integral-mono*)+ (*simp add*: *indicator-def*)
also have ... ≤ $(\int^+ x.\ 1\ \partial\mu)$

by(*rule nn-integral-mono*) (*simp add*: *subprob-spaces disintegration-space-eq*(*2*)[*OF assms*(*2*)] *subprob-space.subprob-emeasure-le-1*)
also have ... < ∞
using *finite-measure.emeasure-finite*[*OF assms*(*3*)]
by (*simp add*: *top.not-eq-extremum*)
finally show (∫⁺ *x*. ∫⁺ *y*. *indicator A* (*x*, *y*) ∂κ *x* ∂μ) ≠ ∞
by *auto*
next
have *A* ⊆ *space X* × *space Y*
by (*metis h*(*1*) *sets.sets-into-space sets-pair-measure space-pair-measure*)
hence ⋀*x*. (∫⁺ *y*. *indicator A* (*x*, *y*) ∂κ *x*) ≤ (∫⁺ *y*. *indicator* (*space X* × *space Y*) (*x*, *y*) ∂κ *x*)
by(*auto intro*!: *nn-integral-mono*)
thus *AE x in* μ. (∫⁺ *y*. *indicator A* (*x*, *y*) ∂κ *x*) ≤ (∫⁺ *y*. *indicator* (*space X* × *space Y*) (*x*, *y*) ∂κ *x*)
by *simp*
qed
also have ... = (∫⁺ *x*. (∫⁺ *y*. *indicator* (*space X* × *space Y*) (*x*, *y*) − *indicator A* (*x*, *y*) ∂κ *x*) ∂μ)
proof(*intro nn-integral-cong nn-integral-diff*[*symmetric*])
fix *x*
assume *x* ∈ *space* μ
then have *x* ∈ *space X*
by(*auto simp*: *disintegration-space-eq*(*2*)[*OF assms*(*2*)])
with *kernel-sets*[*OF this*] *h*(*1*)[*simplified sets-pair-measure*[*symmetric*]]
show (λ*y*. *indicator* (*space X* × *space Y*) (*x*, *y*)) ∈ *borel-measurable* (κ *x*)
(λ*y*. *indicator A* (*x*, *y*)) ∈ *borel-measurable* (κ *x*)
by *auto*
next
fix *x*
assume *x* ∈ *space* μ
then have *x* ∈ *space X*
by(*auto simp*: *disintegration-space-eq*(*2*)[*OF assms*(*2*)])
have (∫⁺ *y*. *indicator A* (*x*, *y*) ∂κ *x*) ≤ (∫⁺ *y*. *1* ∂κ *x*)
by(*rule nn-integral-mono*) (*simp add*: *indicator-def*)
also have ... ≤ *1*
using *subprob-spaces*[*OF* ‹*x* ∈ *space X*›] **by** (*simp add*: *subprob-space.subprob-emeasure-le-1*)
also have ... < ∞
by *auto*
finally show (∫⁺ *y*. *indicator A* (*x*, *y*) ∂κ *x*) ≠ ∞
by *simp*
have *A* ⊆ *space X* × *space Y*
by (*metis h*(*1*) *sets.sets-into-space sets-pair-measure space-pair-measure*)
thus *AE y in* κ *x*. *indicator A* (*x*, *y*) ≤ (*indicator* (*space X* × *space Y*) (*x*, *y*) :: *ennreal*)
by *auto*
qed
also have ... = *?rhs*
by(*auto simp*: *indicator-def intro*!: *nn-integral-cong*)

```
      finally show ?thesis .
    qed
  next
    fix A
    assume h:disjoint-family A range A ⊆ sigma-sets (space X × space Y) {a ×
b |a b. a ∈ sets X ∧ b ∈ sets Y}
           ⋀i::nat. integral^N ν (indicator (A i)) = (∫⁺ x. ∫⁺ y. indicator (A i)
(x, y) ∂κ x ∂μ)
    show integral^N ν (indicator (⋃ (range A))) = (∫⁺ x. ∫⁺ y. indicator (⋃
(range A)) (x, y) ∂κ x ∂μ) (is ?lhs = ?rhs)
    proof −
      have ?lhs = (∫⁺ z. (∑ i. indicator (A i) z) ∂ν)
        by(simp add: suminf-indicator[OF h(1)])
      also have ... = (∑ i. (∫⁺ z. indicator (A i) z ∂ν))
          by(rule nn-integral-suminf) (use disintegration-sets-eq(1)[OF assms(2)]
h(2)[simplified sets-pair-measure[symmetric]] in simp)
      also have ... = (∑ i. (∫⁺ x. ∫⁺ y. indicator (A i) (x, y) ∂κ x ∂μ))
        by(simp add: h)
      also have ... = (∫⁺ x. (∑ i. (∫⁺ y. indicator (A i) (x, y) ∂κ x)) ∂μ)
      by(rule nn-integral-suminf[symmetric]) (use h(2)[simplified sets-pair-measure[symmetric]]
disintegration-sets-eq(2)[OF assms(2)] nn-integral-measurable-f in simp)
      also have ... = (∫⁺ x. ∫⁺ y. (∑ i. indicator (A i) (x, y)) ∂κ x ∂μ)
        using h(2)[simplified sets-pair-measure[symmetric]] kernel-sets
        by(auto intro!: nn-integral-cong nn-integral-suminf[symmetric] simp: disin-
tegration-space-eq(2)[OF assms(2)])
      also have ... = ?rhs
        by(simp add: suminf-indicator[OF h(1)])
      finally show ?thesis .
    qed
  qed(simp-all add: Int-stable-pair-measure-generator pair-measure-closed)
next
  case (mult u c)
  show ?case (is ?lhs = ?rhs)
  proof −
    have ?lhs = c ∗ (∫⁺ z. u z ∂ν)
      using disintegration-sets-eq(1)[OF assms(2)] mult
      by(simp add: nn-integral-cmult)
    also have ... = c ∗ ( ∫⁺ x. ∫⁺ y. u (x, y) ∂κ x ∂μ)
      by(simp add: mult)
    also have ... = ( ∫⁺ x. c ∗ (∫⁺ y. u (x, y) ∂κ x) ∂μ)
        using nn-integral-measurable-f'[OF mult(2)] disintegration-sets-eq(2)[OF
assms(2)]
      by(simp add: nn-integral-cmult)
    also have ... = ?rhs
     using mult by(auto intro!: nn-integral-cong nn-integral-cmult[symmetric] simp:
disintegration-space-eq(2)[OF assms(2)])
    finally show ?thesis .
  qed
next
```

case (*add u v*)
show *?case* (**is** *?lhs = ?rhs*)
proof −
have *?lhs* = $(\int^+ z.\ v\ z\ \partial\nu) + (\int^+ z.\ u\ z\ \partial\nu)$
using *add disintegration-sets-eq(1)*[*OF assms(2)*] **by** (*simp add*: *nn-integral-add*)
also have ... = $(\int^+ x.\ \int^+ y.\ v\ (x,\ y)\ \partial\kappa\ x\ \partial\mu) + (\int^+ x.\ \int^+ y.\ u\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
by(*simp add*: *add*)
also have ... = $(\int^+ x.\ (\int^+ y.\ v\ (x,\ y)\ \partial\kappa\ x) + (\int^+ y.\ u\ (x,\ y)\ \partial\kappa\ x)\ \partial\mu)$
using *nn-integral-measurable-f'*[*OF add(1)*] *nn-integral-measurable-f'*[*OF add(3)*] *disintegration-sets-eq*[*OF assms(2)*]
by(*auto intro*!: *nn-integral-add*[*symmetric*])
also have ... = $(\int^+ x.\ (\int^+ y.\ v\ (x,\ y) + u\ (x,\ y)\ \partial\kappa\ x)\ \partial\mu)$
using *add* **by**(*auto intro*!: *nn-integral-add*[*symmetric*] *nn-integral-cong simp*: *disintegration-space-eq(2)*[*OF assms(2)*])
finally show *?thesis* **.**
qed
next
case (*seq fi*)
have $(\int^+ y.\ (\bigsqcup\ range\ fi)\ (x,\ y)\ \partial\kappa\ x) = (\bigsqcup i.\ \int^+ y.\ fi\ i\ (x,\ y)\ \partial\kappa\ x)$ (**is** *?lhs = ?rhs*) **if** $x \in space\ X$ **for** *x*
proof −
have *?lhs* = $(\int^+ y.\ (\bigsqcup i.\ fi\ i\ (x,\ y))\ \partial\kappa\ x)$
by(*metis SUP-apply*)
also have ... = *?rhs*
proof(*rule nn-integral-monotone-convergence-SUP*)
show *incseq* $(\lambda i\ y.\ fi\ i\ (x,\ y))$
using *seq mono-compose* **by** *blast*
next
fix *i*
show $(\lambda y.\ fi\ i\ (x,\ y)) \in borel\text{-}measurable\ (\kappa\ x)$
using *seq(1)*[*of i*] *that kernel-sets*[*OF that*] **by** *simp*
qed
finally show *?thesis* **.**
qed
have $integral^N\ \nu\ (\bigsqcup\ range\ fi) = (\int^+ x.\ (\bigsqcup i.\ fi\ i\ x)\ \partial\nu)$
by (*metis SUP-apply*)
also have ... = $(\bigsqcup i.\ integral^N\ \nu\ (fi\ i))$
using *disintegration-sets-eq(1)*[*OF assms(2)*] *seq(1,3)*
by(*auto intro*!: *nn-integral-monotone-convergence-SUP*)
also have ... = $(\bigsqcup i.\ \int^+ x.\ \int^+ y.\ fi\ i\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
by(*simp add*: *seq*)
also have ... = $(\int^+ x.\ (\bigsqcup i.\ \int^+ y.\ fi\ i\ (x,\ y)\ \partial\kappa\ x)\ \partial\mu)$
proof(*safe intro*!: *nn-integral-monotone-convergence-SUP*[*symmetric*])
show *incseq* $(\lambda i\ x.\ \int^+ y.\ fi\ i\ (x,\ y)\ \partial\kappa\ x)$
using *le-funD*[*OF incseq-SucD*[*OF seq(3)*]]
by(*auto intro*!: *incseq-SucI le-funI nn-integral-mono*)
qed(*use disintegration-sets-eq(2)*[*OF assms(2)*] *nn-integral-measurable-f'*[*OF seq(1)*] **in** *auto*)

also have ... = $(\int^+ x.\ \int^+ y.\ (\bigsqcup i.\ fi\ i\ (x,\ y))\ \partial\kappa\ x\ \partial\mu)$
using *kernel-sets seq*(*1*)
by(*auto intro*!: *nn-integral-cong nn-integral-monotone-convergence-SUP*[*symmetric*] *simp*: *disintegration-space-eq*(*2*)[*OF assms*(*2*)] *mono-compose seq*(*3*))
also have ... = $(\int^+ x.\ \int^+ y.\ (\bigsqcup\ range\ fi)\ (x,\ y)\ \partial\kappa\ x\ \partial\mu)$
by(*auto intro*!: *nn-integral-cong simp*: *image-image*)
finally show *?case* **.**
qed

lemma(**in** *prob-kernel*) *nn-integral-fst*:
assumes $f \in$ *borel-measurable* $(X \bigotimes_M Y)$ *disintegration* ν μ *sigma-finite-measure* μ
shows $(\int^+ z.\ f\ z\ \partial\nu) = (\int^+ x.\ \int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial\mu)$
proof(*rule sigma-finite-measure.sigma-finite-disjoint*[*OF assms*(*3*)])
fix *A*
assume *A*:*range A* $\subseteq$ *sets* μ $\bigcup$ (*range A*) = *space* μ $\bigwedge i$::*nat. emeasure* μ (*A i*) $\neq \infty$ *disjoint-family A*
then have *A'*: *range* $(\lambda i.\ A\ i \times space\ Y) \subseteq$ *sets* ν $\bigcup$ (*range* $(\lambda i.\ A\ i \times space\ Y)$) = *space* ν *disjoint-family* $(\lambda i.\ A\ i \times space\ Y)$
by(*auto simp*: *disintegration-sets-eq*[*OF assms*(*2*)] *disjoint-family-on-def disintegration-space-eq*[*OF assms*(*2*)] *space-pair-measure*) *blast*
show *?thesis* (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = $(\int^+ z \in \bigcup$ (*range* $(\lambda i.\ A\ i \times space\ Y)$). $f\ z\ \partial\nu)$
using *A'*(*2*) **by** *auto*
also have ... = $(\sum i.\ \int^+ z \in A\ i \times space\ Y.\ f\ z\ \partial\nu)$
using *A'*(*1*,*3*) *assms*(*1*) *disintegration-sets-eq*[*OF assms*(*2*)]
by(*auto intro*!: *nn-integral-disjoint-family*)
also have ... = $(\sum i.\ \int^+ z.\ f\ z\ \partial$*restrict-space* ν $(A\ i \times space\ Y))$
using *A'*(*1*) **by**(*auto intro*!: *suminf-cong nn-integral-restrict-space*[*symmetric*])
also have ... = $(\sum i.\ \int^+ x.\ \int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial$*restrict-space* μ $(A\ i))$
proof(*safe intro*!: *suminf-cong*)
fix *n*
interpret *pk*: *prob-kernel restrict-space X* (*A n*) *Y* κ
by(*rule restrict-probability-kernel*)
have *An*:*A n* $\cap$ *space X* $\in$ *sets X A n* $\cap$ *space X* = *A n*
using *A*(*1*) **by**(*auto simp*: *disintegration-sets-eq*[*OF assms*(*2*)])
have *f*:*f* $\in$ *borel-measurable* (*restrict-space X* (*A n*) $\bigotimes_M$ *Y*)
proof −
have *1*:*sets* (*restrict-space X* (*A n*) $\bigotimes_M$ *Y*) = *sets* (*restrict-space* (*X* $\bigotimes_M$ *Y*) (*A n* × *space Y*))
using *sets-pair-restrict-space*[**where** *Y*=*Y* **and** *B*=*space Y*] **by** *simp*
show *?thesis*
using *assms*(*1*) **by**(*simp add*: *measurable-cong-sets*[*OF 1 refl*] *measurable-restrict-space1*)
qed
have *fin*: *finite-measure* (*restrict-space* μ (*A n*))
by (*metis A*(*1*) *A*(*3*) *UNIV-I emeasure-restrict-space finite-measureI image-subset-iff space-restrict-space space-restrict-space2 subset-eq*)

show $(\int^+ z.\ f\ z\ \partial$*restrict-space* $\nu\ (A\ n \times$ *space* $Y)) = (\int^+ x.\ \int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial$*restrict-space* $\mu\ (A\ n))$
by(*rule pk.nn-integral-fst-finite'*[*OF f disintegration-restrict-space*[*OF assms*(*2*) *An*(*1*)] *fin*])
qed
also have ... $= (\sum i.\ \int^+ x \in A\ i.\ \int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial\mu)$
using *A*(*1*) **by**(*auto intro*!: *suminf-cong nn-integral-restrict-space*)
also have ... $= (\int^+ x \in \bigcup$ (*range A*). $\int^+ y.\ f\ (x,y)\ \partial(\kappa\ x)\ \partial\mu)$
using *A*(*1*,*4*) *nn-integral-measurable-f'*[*OF assms*(*1*)] *disintegration-sets-eq*[*OF assms*(*2*)]
by(*auto intro*!: *nn-integral-disjoint-family*[*symmetric*])
also have ... = *?rhs*
using *A*(*2*) **by** *simp*
finally show *?thesis* **.**
qed
qed

lemma(**in** *prob-kernel*) *integrable-eq1*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes [*measurable*]:*f* ∈ *borel-measurable* $(X \bigotimes_M Y)$
and *disintegration* $\nu\ \mu$ *sigma-finite-measure* μ
shows $(\int^+ z.$ *ennreal* (*norm* (*f z*)) $\partial\nu) < \infty \longleftrightarrow (\int^+ x.\ \int^+ y.$ *ennreal* (*norm* (*f* (*x*,*y*))) $\partial(\kappa\ x)\ \partial\mu) < \infty$
by(*simp add*: *nn-integral-fst*[*OF* - *assms*(*2*,*3*)])

lemma(**in** *prob-kernel*) *integrable-kernel-integrable*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes *integrable* ν *f disintegration* $\nu\ \mu$ *sigma-finite-measure* μ
shows *AE x in* μ. *integrable* $(\kappa\ x)$ $(\lambda y.\ f\ (x,y))$
proof −
have [*measurable*]:*f* ∈ *borel-measurable* $(X \bigotimes_M Y)$
using *integrable-iff-bounded assms*(*1*) *disintegration-sets-eq*[*OF assms*(*2*)] **by** *simp*
show *?thesis*
unfolding *integrable-iff-bounded*
proof −
have *1*:$(\int^+ x.\ \int^+ y.$ *ennreal* (*norm* (*f* (*x*,*y*))) $\partial\kappa\ x\ \partial\mu) < \infty$
using *assms*(*1*) *integrable-eq1*[*OF* - *assms*(*2*,*3*),*of f*] **by**(*simp add*: *integrable-iff-bounded*)
have *AE x in* μ. $(\int^+ y.$ *ennreal* (*norm* (*f* (*x*,*y*))) $\partial\kappa\ x) \neq \infty$
by(*rule nn-integral-PInf-AE*) (*use 1 disintegration-sets-eq*[*OF assms*(*2*)] *nn-integral-measurable-f* **in** *auto*)
thus *AE x in* μ. $(\lambda y.\ f\ (x,\ y)) \in$ *borel-measurable* $(\kappa\ x) \wedge (\int^+ y.$ *ennreal* (*norm* (*f* (*x*,*y*))) $\partial\kappa\ x) < \infty$
using *top.not-eq-extremum* **by**(*fastforce simp*: *disintegration-space-eq*[*OF assms*(*2*)])
qed
qed

lemma(**in** *prob-kernel*) *integrable-lebesgue-integral-integrable′*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes *integrable ν f disintegration ν μ sigma-finite-measure μ*
shows *integrable μ* (λ*x*. ∫ *y*. *f* (*x*,*y*) ∂(*κ x*))
unfolding *integrable-iff-bounded*
proof
show (λ*x*. ∫ *y*. *f* (*x*,*y*) ∂(*κ x*)) ∈ *borel-measurable μ*
using *disintegration-sets-eq*[*OF assms*(*2*)] *assms*(*1*) *integral-measurable-f′*[*of f*]
by(*auto simp*: *integrable-iff-bounded*)
next
have (∫$^+$ *x*. *ennreal* (*norm* (∫ *y*. *f* (*x*,*y*) ∂(*κ x*))) ∂*μ*) ≤ (∫$^+$ *x*. ∫$^+$*y*. *ennreal* (*norm* (*f* (*x*,*y*))) ∂(*κ x*) ∂*μ*)
using *integral-norm-bound-ennreal integrable-kernel-integrable*[*OF assms*]
by(*auto intro*!: *nn-integral-mono-AE*)
also have ... < ∞
using *integrable-eq1*[*OF - assms*(*2*,*3*),*of f*] *assms*(*1*) *disintegration-sets-eq*[*OF assms*(*2*)]
by(*simp add*: *integrable-iff-bounded*)
finally show (∫$^+$ *x*. *ennreal* (*norm* (∫ *y*. *f* (*x*,*y*) ∂(*κ x*))) ∂*μ*) < ∞ **.**
qed

lemma(**in** *prob-kernel*) *integrable-lebesgue-integral-integrable*:
fixes *f* :: - ⇒- ⇒ -::{*banach*, *second-countable-topology*}
assumes *integrable ν* (λ(*x*,*y*). *f x y*) *disintegration ν μ sigma-finite-measure μ*
shows *integrable μ* (λ*x*. ∫ *y*. *f x y* ∂(*κ x*))
using *integrable-lebesgue-integral-integrable′*[*OF assms*] **by** *simp*

lemma(**in** *prob-kernel*) *integral-fst*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes *integrable ν f disintegration ν μ sigma-finite-measure μ*
shows (∫ *z*. *f z* ∂*ν*) = (∫ *x*. ∫ *y*. *f* (*x*,*y*) ∂(*κ x*) ∂*μ*)
using *assms*(*1*)
proof *induct*
case *b*:(*base A c*)
then have *0*:*integrable ν* (*indicat-real A*)
by *blast*
then have *1*[*measurable*]: *indicat-real A* ∈ *borel-measurable* (*X* ⨂$_M$ *Y*)
using *disintegration-sets-eq*[*OF assms*(*2*)] **by** *auto*
have *eq*:(∫ *z*. *indicat-real A z* ∂*ν*) = (∫ *x*. ∫ *y*. *indicat-real A* (*x*,*y*) ∂*κ x* ∂*μ*) (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = *enn2real* (∫$^+$ *z*. *ennreal* (*indicat-real A z*) ∂*ν*)
by(*rule integral-eq-nn-integral*) (*use b* **in** *auto*)
also have ... = *enn2real* (∫$^+$ *x*. ∫$^+$ *y*. *ennreal* (*indicat-real A* (*x*,*y*)) ∂*κ x* ∂*μ*)
using *nn-integral-fst*[*OF - assms*(*2*,*3*)] *b disintegration-sets-eq*[*OF assms*(*2*)]
by *auto*
also have ... = *enn2real* (∫$^+$ *x*. *ennreal* (∫ *y*. *indicat-real A* (*x*,*y*) ∂*κ x*) ∂*μ*)
proof −
have (∫$^+$ *x*. ∫$^+$ *y*. *ennreal* (*indicat-real A* (*x*,*y*)) ∂*κ x* ∂*μ*) = (∫$^+$ *x*. *ennreal*

```
(∫ y. indicat-real A (x,y) ∂κ x) ∂μ)
    proof(safe intro!: nn-integral-cong nn-integral-eq-integral)
      fix x
      assume x ∈ space μ
      then have x ∈ space X
        by(simp add:  disintegration-space-eq[OF assms(2)])
      hence [simp]:prob-space (κ x) sets (κ x) = sets Y space (κ x) = space Y
        by(auto intro!: prob-spaces sets-eq-imp-space-eq kernel-sets)
      have [simp]:{y. (x, y) ∈ A} ∈ sets Y
      proof −
        have {y. (x, y) ∈ A} = (λy. (x,y)) −‘ A ∩ space Y
          using b(1)[simplified disintegration-sets-eq[OF assms(2)]]
          by auto
        also have ... ∈ sets Y
          using b(1)[simplified disintegration-sets-eq[OF assms(2)]] ‹x ∈ space X›
          by auto
        finally show ?thesis .
      qed
      have [simp]: (λy. indicat-real A (x, y)) = indicat-real {y. (x,y) ∈ A}
        by (auto simp: indicator-def)
      show integrable (κ x) (λy. indicat-real A (x, y))
        using prob-space.emeasure-le-1[of κ x {y. (x, y) ∈ A}]
        by(auto simp add: integrable-indicator-iff order-le-less-trans)
    qed simp
    thus ?thesis by simp
  qed
  also have ... = ?rhs
    using disintegration-sets-eq[OF assms(2)] integral-measurable-f'[OF 1]
    by(auto intro!: integral-eq-nn-integral[symmetric])
  finally show ?thesis .
qed
show ?case (is ?lhs = ?rhs)
proof −
  have ?lhs = (∫ z. indicat-real A z ∂ν) *R c
    using 0 by auto
  also have ... = (∫ x. ∫ y. indicat-real A (x,y) ∂κ x ∂μ) *R c
    by(simp only: eq)
  also have ... = (∫ x. (∫ y. indicat-real A (x,y) ∂κ x) *R c ∂μ)
    using integrable-lebesgue-integral-integrable'[OF 0 assms(2,3)]
    by(auto intro!: integral-scaleR-left[symmetric])
  also have ... = ?rhs
    using integrable-kernel-integrable[OF 0 assms(2,3)] integral-measurable-f'[of
indicat-real A] integral-measurable-f'[of λz. indicat-real A z *R c] disintegration-sets-eq[OF
assms(2)]
    by(auto intro!: integral-cong-AE)
  finally show ?thesis .
qed
next
  case fg:(add f g)
```

note [*measurable*] = *integrable-lebesgue-integral-integrable'*[*OF fg*(*1*) *assms*(*2*,*3*)] *integrable-lebesgue-integral-integrable'*[*OF fg*(*3*) *assms*(*2*,*3*)] *integrable-lebesgue-integral-integrable'*[*OF Bochner-Integration.integrable-add*[*OF fg*(*1*,*3*)] *assms*(*2*,*3*)]
show *?case* (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = (∫ *x*. (∫ *y*. *f* (*x*,*y*) ∂(κ *x*)) + (∫ *y*. *g* (*x*,*y*) ∂(κ *x*)) ∂μ)
by(*simp add*: *Bochner-Integration.integral-add*[*OF fg*(*1*,*3*)] *fg Bochner-Integration.integral-add*[*OF integrable-lebesgue-integral-integrable'*[*OF fg*(*1*) *assms*(*2*,*3*)] *integrable-lebesgue-integral-integrable'*[*OF fg*(*3*) *assms*(*2*,*3*)]])
also have ... = *?rhs*
using *integrable-kernel-integrable*[*OF fg*(*1*) *assms*(*2*,*3*)] *integrable-kernel-integrable*[*OF fg*(*3*) *assms*(*2*,*3*)]
by(*auto intro*!: *integral-cong-AE*)
finally show *?thesis* **.**
qed
next
case (*lim f s*)
then have [*measurable*]: *f* ∈ *borel-measurable* ν ⋀*i*. *s i* ∈ *borel-measurable* ν
by *auto*
show *?case*
proof (*rule LIMSEQ-unique*)
show (λ*i*. *integral*^{L} ν (*s i*)) ⟶ *integral*^{L} ν *f*
proof (*rule integral-dominated-convergence*)
show *integrable* ν (λ*x*. *2* ∗ *norm* (*f x*))
using *lim*(*5*) **by** *auto*
qed(*use lim* **in** *auto*)
next
have (λ*i*. ∫ *x*. ∫ *y*. *s i* (*x*, *y*) ∂(κ *x*) ∂μ) ⟶ ∫ *x*. ∫ *y*. *f* (*x*, *y*) ∂(κ *x*) ∂μ
proof (*rule integral-dominated-convergence*)
have *AE x in* μ. ∀*i*. *integrable* (κ *x*) (λ*y*. *s i* (*x*, *y*))
unfolding *AE-all-countable* **using** *integrable-kernel-integrable*[*OF lim*(*1*) *assms*(*2*,*3*)] **..**
with *AE-space integrable-kernel-integrable*[*OF lim*(*5*) *assms*(*2*,*3*)]
show *AE x in* μ. (λ*i*. ∫ *y*. *s i* (*x*, *y*) ∂(κ *x*)) ⟶ ∫ *y*. *f* (*x*, *y*) ∂(κ *x*)
proof *eventually-elim*
fix *x* **assume** *x*: *x* ∈ *space* μ **and**
s: ∀*i*. *integrable* (κ *x*) (λ*y*. *s i* (*x*, *y*)) **and** *f*: *integrable* (κ *x*) (λ*y*. *f* (*x*, *y*))
show (λ*i*. ∫ *y*. *s i* (*x*, *y*) ∂(κ *x*)) ⟶ ∫ *y*. *f* (*x*, *y*) ∂(κ *x*)
proof (*rule integral-dominated-convergence*)
show *integrable* (κ *x*) (λ*y*. *2* ∗ *norm* (*f* (*x*, *y*)))
using *f* **by** *auto*
show *AE xa in* (κ *x*). (λ*i*. *s i* (*x*, *xa*)) ⟶ *f* (*x*, *xa*)
using *x lim*(*3*) *kernel-space* **by** (*auto simp*: *space-pair-measure disintegration-space-eq*[*OF assms*(*2*)])
show ⋀*i*. *AE xa in* (κ *x*). *norm* (*s i* (*x*, *xa*)) ≤ *2* ∗ *norm* (*f* (*x*, *xa*))
using *x lim*(*4*) *kernel-space* **by** (*auto simp*: *space-pair-measure disintegration-space-eq*[*OF assms*(*2*)])
qed (*use x disintegration-sets-eq*[*OF assms*(*2*)] *disintegration-space-eq*[*OF assms*(*2*)] **in** *auto*)

qed
next
show *integrable* μ $(\lambda x.\ (\int\ y.\ 2 * norm\ (f\ (x,\ y))\ \partial(\kappa\ x)))$
using *integrable-lebesgue-integral-integrable'*[*OF - assms*(*2*,*3*),*of* $\lambda z.\ 2 *$ *norm* (*f* (*fst z*, *snd z*))] *lim*(*5*)
by *auto*
next
fix *i* **show** *AE x in* μ. *norm* $(\int\ y.\ s\ i\ (x,\ y)\ \partial(\kappa\ x)) \leq (\int\ y.\ 2 * norm\ (f\ (x,\ y))\ \partial(\kappa\ x))$
using *AE-space integrable-kernel-integrable*[*OF lim*(*1*) *assms*(*2*,*3*),*of i*] *integrable-kernel-integrable*[*OF lim*(*5*) *assms*(*2*,*3*)]
proof *eventually-elim*
case *sf*:(*elim x*)
from *sf*(*2*) **have** *norm* $(\int\ y.\ s\ i\ (x,\ y)\ \partial(\kappa\ x)) \leq (\int^+ y.\ norm\ (s\ i\ (x,\ y))\ \partial(\kappa\ x))$
by (*rule integral-norm-bound-ennreal*)
also have $\ldots \leq (\int^+ y.\ 2 * norm\ (f\ (x,\ y))\ \partial(\kappa\ x))$
using *sf lim kernel-space* **by** (*auto intro*!: *nn-integral-mono simp*: *space-pair-measure disintegration-space-eq*[*OF assms*(*2*)])
also have $\ldots = (\int y.\ 2 * norm\ (f\ (x,\ y))\ \partial(\kappa\ x))$
using *sf* **by** (*intro nn-integral-eq-integral*) *auto*
finally show *norm* $(\int\ y.\ s\ i\ (x,\ y)\ \partial(\kappa\ x)) \leq (\int\ y.\ 2 * norm\ (f\ (x,\ y))\ \partial(\kappa\ x))$
by *simp*
qed
qed(*use integrable-lebesgue-integral-integrable'*[*OF lim*(*1*) *assms*(*2*,*3*)] *integrable-lebesgue-integral-integrable'*[*OF lim*(*5*) *assms*(*2*,*3*)] *disintegration-sets-eq*[*OF assms*(*2*)] **in** *auto*)
then show $(\lambda i.\ integral^L\ \nu\ (s\ i)) \longrightarrow \int\ x.\ \int\ y.\ f\ (x,\ y)\ \partial(\kappa\ x)\ \partial\mu$
using *lim* **by** *simp*
qed
qed

2.4 Marginal Measure

definition *marginal-measure-on* :: [*'a measure*, *'b measure*, (*'a* × *'b*) *measure*, *'b set*] ⇒ *'a measure* **where**
marginal-measure-on X Y ν *B* = *measure-of* (*space X*) (*sets X*) $(\lambda A.\ \nu\ (A \times B))$

abbreviation *marginal-measure* :: [*'a measure*, *'b measure*, (*'a* × *'b*) *measure*] ⇒ *'a measure* **where**
marginal-measure X Y $\nu \equiv$ *marginal-measure-on X Y* ν (*space Y*)

lemma *space-marginal-measure*: *space* (*marginal-measure-on X Y* ν *B*) = *space X*
and *sets-marginal-measure*: *sets* (*marginal-measure-on X Y* ν *B*) = *sets X*
by(*simp-all add*: *marginal-measure-on-def*)

lemma *emeasure-marginal-measure-on*:
assumes *sets* ν = *sets* $(X \bigotimes_M Y)$ *B* ∈ *sets Y A* ∈ *sets X*

shows *marginal-measure-on X Y ν B A = ν (A × B)*
unfolding *marginal-measure-on-def*
proof(*rule emeasure-measure-of-sigma*)
show *countably-additive (sets X) (λA. emeasure ν (A × B))*
proof(*rule countably-additiveI*)
fix *A :: nat ⇒ -*
assume *h:range A ⊆ sets X disjoint-family A* $\bigcup$ *(range A) ∈ sets X*
have [*simp*]: *(*$\bigcup$ *i. A i × B) = (*$\bigcup$ *(range A) × B)*
by *blast*
have *range (λi. A i × B) ⊆ sets ν disjoint-family (λi. A i × B)*
using *h assms(1,2)* **by**(*auto simp*: *disjoint-family-on-def*)
from *suminf-emeasure*[*OF this*]
show *(*$\sum$ *i. ν (A i × B)) = ν (*$\bigcup$ *(range A) × B)*
by *simp*
qed
qed(*insert assms*, *auto simp*: *positive-def sets.sigma-algebra-axioms*)

lemma *emeasure-marginal-measure*:
assumes *sets ν = sets (X* $\bigotimes_M$ *Y) A ∈ sets X*
shows *marginal-measure X Y ν A = ν (A × space Y)*
using *emeasure-marginal-measure-on*[*OF assms(1) - assms(2)*] **by** *simp*

lemma *finite-measure-marginal-measure-on-finite*:
assumes *finite-measure ν sets ν = sets (X* $\bigotimes_M$ *Y) B ∈ sets Y*
shows *finite-measure (marginal-measure-on X Y ν B)*
by (*simp add*: *assms emeasure-marginal-measure-on finite-measure.emeasure-finite finite-measureI space-marginal-measure*)

lemma *finite-measure-marginal-measure-finite*:
assumes *finite-measure ν sets ν = sets (X* $\bigotimes_M$ *Y)*
shows *finite-measure (marginal-measure X Y ν)*
by(*rule finite-measure-marginal-measure-on-finite*[*OF assms sets.top*])

lemma *marginal-measure-restrict-space*:
assumes *sets ν = sets (X* $\bigotimes_M$ *Y) B ∈ sets Y*
shows *marginal-measure X (restrict-space Y B) (restrict-space ν (space X × B)) = marginal-measure-on X Y ν B*
proof(*rule measure-eqI*)
fix *A*
assume *A ∈ sets (marginal-measure X (restrict-space Y B) (restrict-space ν (space X × B)))*
then have *A ∈ sets X*
by(*simp add*: *sets-marginal-measure*)
have *1*:*sets (restrict-space ν (space X × B)) = sets (X* $\bigotimes_M$ *restrict-space Y B)*
by (*metis assms(1) restrict-space-space sets-pair-restrict-space sets-restrict-space-cong*)
show *emeasure (marginal-measure X (restrict-space Y B) (restrict-space ν (space X × B))) A = emeasure (marginal-measure-on X Y ν B) A*
apply(*simp add*: *emeasure-marginal-measure-on*[*OF assms(1) assms(2) ‹A ∈ sets X›*] *emeasure-marginal-measure*[*OF 1 ‹A ∈ sets X›*])

apply(*simp add*: *space-restrict-space*)
by (*metis Sigma-cong Sigma-mono* ‹*A* ∈ *sets X*› *assms*(*1*) *assms*(*2*) *emeasure-restrict-space inf-le1 pair-measureI sets.Int-space-eq2 sets.sets-into-space sets.top*)
qed(*simp add*: *sets-marginal-measure*)

lemma *restrict-space-marginal-measure-on*:
assumes *sets* ν = *sets* (*X* $\bigotimes_M$ *Y*) *B* ∈ *sets Y A* ∈ *sets X*
shows *restrict-space* (*marginal-measure-on X Y* ν *B*) *A* = *marginal-measure-on* (*restrict-space X A*) *Y* (*restrict-space* ν (*A* × *space Y*)) *B*
proof(*rule measure-eqI*)
fix *A'*
assume *A'* ∈ *sets* (*restrict-space* (*marginal-measure-on X Y* ν *B*) *A*)
then have *h*:*A'* ∈ *sets.restricted-space X A*
by(*simp add*: *sets-marginal-measure sets-restrict-space*)
show *emeasure* (*restrict-space* (*marginal-measure-on X Y* ν *B*) *A*) *A'* = *emeasure* (*marginal-measure-on* (*restrict-space X A*) *Y* (*restrict-space* ν (*A* × *space Y*)) *B*) *A'* (**is** *?lhs* = *?rhs*)
proof −
have *1*:*sets* (*restrict-space* ν (*A* × *space Y*)) = *sets* (*restrict-space X A* $\bigotimes_M$ *Y*)
by (*metis assms*(*1*) *restrict-space-space sets-pair-restrict-space sets-restrict-space-cong*)
have *?lhs* = *emeasure* (*marginal-measure-on X Y* ν *B*) *A'*
using *h* **by**(*auto intro*!: *emeasure-restrict-space simp*: *space-marginal-measure sets-marginal-measure assms*)
also have ... = ν (*A'* × *B*)
using *emeasure-marginal-measure-on*[*OF assms*(*1*,*2*),*of A'*] *h assms*(*3*) **by** *auto*
also have ... = *restrict-space* ν (*A* × *space Y*) (*A'* × *B*)
using *h assms sets.sets-into-space*
by(*auto intro*!: *emeasure-restrict-space*[*symmetric*])
also have ... = *?rhs*
using *emeasure-marginal-measure-on*[*OF 1 assms*(*2*),*simplified sets-restrict-space*,*OF h*] **..**
finally show *?thesis* **.**
qed
qed(*simp add*: *sets-marginal-measure sets-restrict-space*)

lemma *restrict-space-marginal-measure*:
assumes *sets* ν = *sets* (*X* $\bigotimes_M$ *Y*) *A* ∈ *sets X*
shows *restrict-space* (*marginal-measure X Y* ν) *A* = *marginal-measure* (*restrict-space X A*) *Y* (*restrict-space* ν (*A* × *space Y*))
using *restrict-space-marginal-measure-on*[*OF assms*(*1*) - *assms*(*2*)] **by** *simp*

lemma *marginal-measure-mono*:
assumes *sets* ν = *sets* (*X* $\bigotimes_M$ *Y*) *A* ∈ *sets Y B* ∈ *sets Y A* ⊆ *B*
shows *emeasure* (*marginal-measure-on X Y* ν *A*) ≤ *emeasure* (*marginal-measure-on X Y* ν *B*)
proof(*rule le-funI*)
fix *U*

show *emeasure* (*marginal-measure-on X Y ν A*) *U* ≤ *emeasure* (*marginal-measure-on X Y ν B*) *U*
proof −
have *1*:*U* × *A* ⊆ *U* × *B* **using** *assms*(*4*) **by** *auto*
show *?thesis*
proof(*cases U* ∈ *sets X*)
case *True*
then show *?thesis*
by (*simp add*: *1 assms emeasure-marginal-measure-on emeasure-mono*)
next
case *False*
then show *?thesis*
by (*simp add*: *emeasure-notin-sets sets-marginal-measure*)
qed
qed
qed

lemma *marginal-measure-absolutely-countinuous*:
assumes *sets ν* = *sets* (*X* ⨂$_M$ *Y*) *A* ∈ *sets Y B* ∈ *sets Y A* ⊆ *B*
shows *absolutely-continuous* (*marginal-measure-on X Y ν B*) (*marginal-measure-on X Y ν A*)
using *emeasure-marginal-measure*[*OF assms*(*1*)] *assms*(*2*,*3*) *le-funD*[*OF marginal-measure-mono*[*OF assms*]]
by(*auto intro*!: *mono-absolutely-continuous simp*: *sets-marginal-measure*)

lemma *marginal-measure-absolutely-continuous′*:
assumes *sets ν* = *sets* (*X* ⨂$_M$ *Y*) *A* ∈ *sets Y*
shows *absolutely-continuous* (*marginal-measure X Y ν*) (*marginal-measure-on X Y ν A*)
by(*rule marginal-measure-absolutely-countinuous*[*OF assms sets.top sets.sets-into-space*[*OF assms*(*2*)]])

2.5 Lemma 14.D.6.

locale *sigma-finite-measure-on-pair* =
fixes *X* :: *′a measure* **and** *Y* :: *′b measure* **and** *ν* :: (*′a* × *′b*) *measure*
assumes *nu-sets*[*measurable-cong*]: *sets ν* = *sets* (*X* ⨂$_M$ *Y*)
and *sigma-finite*: *sigma-finite-measure ν*
begin

abbreviation *νx* ≡ *marginal-measure X Y ν*

end

locale *projection-sigma-finite* =
fixes *X* :: *′a measure* **and** *Y* :: *′b measure* **and** *ν* :: (*′a* × *′b*) *measure*
assumes *nu-sets*[*measurable-cong*]: *sets ν* = *sets* (*X* ⨂$_M$ *Y*)
and *marginal-sigma-finite*: *sigma-finite-measure* (*marginal-measure X Y ν*)
begin

sublocale νx : *sigma-finite-measure marginal-measure X Y* ν
by(*rule marginal-sigma-finite*)

lemma ν*-sigma-finite*: *sigma-finite-measure* ν
proof(*rule* νx*.sigma-finite*[*simplified sets-marginal-measure space-marginal-measure*])
fix *A* :: *nat* ⇒ -
assume *A*: *range A* ⊆ *sets X* ⋃ (*range A*) = *space X* ⋀*i*. *marginal-measure X Y* ν (*A i*) ≠ ∞
define *C* **where** *C* ≡ *range* (λ*n*. *A n* × *space Y*)
have *1*:*C* ⊆ *sets* ν *countable C* ⋃ *C* = *space* ν
using *nu-sets A*(*1*,*2*) **by**(*auto simp*: *C-def sets-eq-imp-space-eq*[*OF nu-sets*] *space-pair-measure*)
show *sigma-finite-measure* ν
unfolding *sigma-finite-measure-def*
proof(*safe intro*!: *exI*[**where** *x*=*C*,*simplified C-def*])
fix *n*
assume ν (*A n* × *space Y*) = ∞
moreover have ν (*A n* × *space Y*) ≠ ∞
using *A*(*3*)[*of n*] *emeasure-marginal-measure*[*OF nu-sets*,*of A n*] *A*(*1*) **by** *auto*
ultimately show *False* **by** *auto*
qed (*use 1 C-def* **in** *auto*)
qed

sublocale *sigma-finite-measure-on-pair*
using ν*-sigma-finite* **by**(*auto simp*: *sigma-finite-measure-on-pair-def nu-sets*)

definition κ' :: *'a* ⇒ *'b set* ⇒ *ennreal* **where**
κ' *x B* ≡ *RN-deriv* νx (*marginal-measure-on X Y* ν *B*) *x*

lemma *kernel-measurable*[*measurable*]:
(λ*x*. *RN-deriv* (*marginal-measure X Y* ν) (*marginal-measure-on X Y* ν *B*) *x*) ∈ *borel-measurable* νx
by *simp*

corollary κ'*-measurable*[*measurable*]:
(λ*x*. κ' *x B*) ∈ *borel-measurable X*
using *sets-marginal-measure*[*of X Y* ν *space Y*] **by**(*auto simp*: κ'*-def*)

lemma *kernel-RN-deriv*:
assumes *A* ∈ *sets X B* ∈ *sets Y*
shows ν (*A* × *B*) = (∫⁺*x*∈*A*. κ' *x B* ∂νx)
unfolding κ'*-def*
proof −
have *emeasure* ν (*A* × *B*) = *emeasure* (*density* νx (*RN-deriv* νx (*marginal-measure-on X Y* ν *B*))) *A*
by (*simp add*: νx*.density-RN-deriv assms emeasure-marginal-measure-on marginal-measure-absolutely-contin*

nu-sets sets-marginal-measure)
then show *emeasure* ν $(A \times B)$ = *set-nn-integral* νx *A* (*RN-deriv* νx (*marginal-measure-on X Y* ν *B*))
by (*simp add*: *assms*(*1*) *emeasure-density sets-marginal-measure*)
qed

lemma *empty-Y-bot*:
assumes *space Y* = {}
shows $\nu = \bot$
proof −
have *sets* ν = {{}}
using *nu-sets space-empty-iff*[*of X* $\bigotimes_M$ *Y*,*simplified space-pair-measure*] *assms*
by *simp*
thus *?thesis*
by(*simp add*: *sets-eq-bot*)
qed

lemma *empty-Y-nux*:
assumes *space Y* = {}
shows νx *A* = *0*
proof(*cases A* ∈ *sets X*)
case *True*
from *emeasure-marginal-measure*[*OF nu-sets this*]
show *?thesis*
by(*simp add*: *assms*)
next
case *False*
with *sets-marginal-measure*[*of X Y* ν *space Y*]
show *?thesis*
by(*auto intro*!: *emeasure-notin-sets*)
qed

lemma *kernel-empty0-AE*:
AE x in νx. κ' *x* {} = *0*
unfolding κ'*-def* **by**(*rule AE-symmetric*[*OF* νx*.RN-deriv-unique*]) (*auto intro*!: *measure-eqI simp*: *sets-marginal-measure emeasure-density emeasure-marginal-measure-on*[*OF nu-sets*])

lemma *kernel-Y1-AE*:
AE x in νx. κ' *x* (*space Y*) = *1*
unfolding κ'*-def* **by**(*rule AE-symmetric*[*OF* νx*.RN-deriv-unique*]) (*auto intro*!: *measure-eqI simp*: *emeasure-density*)

lemma *kernel-suminf-AE*:
assumes *disjoint-family F*
and $\bigwedge i$. *F i* ∈ *sets Y*
shows *AE x in* νx. $(\sum i.\ \kappa'\ x\ (F\ i)) = \kappa'\ x\ (\bigcup\ (range\ F))$
unfolding κ'*-def*
proof(*rule* νx*.RN-deriv-unique*)

show *density* νx ($\lambda x.$ $\sum i.$ *RN-deriv local.*νx (*marginal-measure-on X Y* ν (*F i*)) *x*) = *marginal-measure-on X Y* ν ($\bigcup$ (*range F*))
proof(*rule measure-eqI*)
fix *A*
assume [*measurable*]:$A \in$ *sets* (*density* νx ($\lambda x.$ $\sum i.$ *RN-deriv* νx (*marginal-measure-on X Y* ν (*F i*)) *x*))
then have [*measurable*]:$A \in$ *sets* νx $A \in$ *sets X* **by**(*auto simp*: *sets-marginal-measure*)
show (*density* νx ($\lambda x.$ $\sum i.$ *RN-deriv* νx (*marginal-measure-on X Y* ν (*F i*)) *x*)) *A* = (*marginal-measure-on X Y* ν ($\bigcup$ (*range F*))) *A*
(**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = ($\int^+ x \in A.$ ($\sum i.$ *RN-deriv* νx (*marginal-measure-on X Y* ν (*F i*)) *x*)$\partial\nu x$)
by(*auto intro*!: *emeasure-density*)
also have ... = ($\int^+ x.$ ($\sum i.$ *RN-deriv* νx (*marginal-measure-on X Y* ν (*F i*)) *x* ∗ *indicator A x*)$\partial\nu x$)
by *simp*
also have ... = ($\sum i.$ ($\int^+ x \in A.$ *RN-deriv* νx (*marginal-measure-on X Y* ν (*F i*)) *x* $\partial\nu x$))
by(*rule nn-integral-suminf*) *auto*
also have ... = ($\sum i.$ ν (*A* × *F i*))
using *kernel-RN-deriv*[*of A F* -] *assms* **by**(*auto intro*!: *suminf-cong simp*: κ'*-def*)
also have ... = ν ($\bigcup i.$ *A* × *F i*)
using *assms nu-sets* **by**(*fastforce intro*!: *suminf-emeasure simp*: *disjoint-family-on-def*)
also have ... = ν (*A* × ($\bigcup i.$ *F i*))
proof −
have ($\bigcup i.$ *A* × *F i*) = (*A* × ($\bigcup i.$ *F i*)) **by** *blast*
thus *?thesis* **by** *simp*
qed
also have ... = *?rhs*
using *nu-sets assms* **by**(*auto intro*!: *emeasure-marginal-measure-on*[*symmetric*])
finally show *?thesis* **.**
qed
qed(*simp add*: *sets-marginal-measure*)
qed *auto*

lemma *kernel-finite-sum-AE*:
assumes *disjoint-family-on F S finite S*
and $\bigwedge i.$ $i \in S \Longrightarrow F\ i \in$ *sets Y*
shows *AE x in* $\nu x.$ ($\sum i \in S.$ κ' *x* (*F i*)) = κ' *x* ($\bigcup i \in S.$ *F i*)
proof −
consider $S = \{\} \mid S \neq \{\}$ **by** *auto*
then show *?thesis*
proof *cases*
case *1*
then show *?thesis*
by(*simp add*: *kernel-empty0-AE*)
next

case *S*:*2*
define F' **where** $F' \equiv (\lambda n.\ \textit{if}\ n < \textit{card}\ S\ \textit{then}\ F\ (\textit{from-nat-into}\ S\ n)\ \textit{else}\ \{\})$
have F'[*simp*]:$\bigwedge i.\ F'\ i \in \textit{sets}\ Y$
using *assms*(*3*)
by (*metis F'-def bot.extremum-strict bot-nat-def card.empty from-nat-into sets.empty-sets*)
have *F'-disj*: *disjoint-family F'*
unfolding *disjoint-family-on-def*
proof *safe*
fix *m n x*
assume *h*:$m \neq n\ x \in F'\ m\ x \in F'\ n$
consider $n < \textit{card}\ S\ m < \textit{card}\ S \mid n \geq \textit{card}\ S \mid m \geq \textit{card}\ S$ **by** *arith*
then show $x \in \{\}$
proof *cases*
case *1*
then have $S \neq \{\}$
by *auto*
with *1* **have** *from-nat-into S n ∈ S from-nat-into S m ∈ S*
using *from-nat-into*[*of S*] **by** *blast*+
moreover have *from-nat-into S n ≠ from-nat-into S m*
by (*metis 1*(*1*) *1*(*2*) *assms*(*2*) *bij-betw-def h*(*1*) *lessThan-iff to-nat-on-finite to-nat-on-from-nat-into*)
ultimately show *?thesis*
using *h assms*(*1*) *1* **by**(*auto simp*: *disjoint-family-on-def F'-def*)
qed(*use h F'-def* **in** *simp-all*)
qed
have *1*:$(\sum i{\in}S.\ \kappa'\ x\ (F\ i)) = (\sum i{<}\textit{card}\ S.\ \kappa'\ x\ (F'\ i))$ **for** *x*
unfolding *F'-def* **by** *auto* (*metis* (*no-types*, *lifting*) *sum.card-from-nat-into sum.cong*)
have *2*: $(\bigcup\ (\textit{range}\ F')) = (\bigcup i{\in}S.\ F\ i)$
proof *safe*
fix *x n*
assume *h*:$x \in F'\ n$
then have $S \neq \{\}\ n < \textit{card}\ S$
by(*auto simp*: *F'-def*) (*meson empty-iff*)
with *h* **show** $x \in \bigcup\ (F\ `\ S)$
by(*auto intro*!: *exI*[**where** *x*=*from-nat-into S n*] *simp*: *F'-def from-nat-into* ‹$S \neq \{\}$›)
next
fix *x s*
assume $s \in S\ x \in F\ s$
with *bij-betwE*[*OF to-nat-on-finite*[*OF assms*(*2*)]]
show $x \in \bigcup\ (\textit{range}\ F')$
by(*auto intro*!: *exI*[**where** *x*=*to-nat-on S s*] *simp*: *F'-def from-nat-into-to-nat-on*[*OF countable-finite*[*OF assms*(*2*)]])
qed
have *AE x in νx*. $(\sum i{<}\textit{card}\ S.\ \kappa'\ x\ (F'\ i)) = (\sum i.\ \kappa'\ x\ (F'\ i))$
proof −
have *AE x in νx*. $\forall\, i{\geq}\textit{card}\ S.\ \kappa'\ x\ (F'\ i) = 0$

```
      using kernel-empty0-AE by(auto simp: F'-def)
    hence AE x in νx. (∑ i. κ' x (F' i)) = (∑ i<card S. κ' x (F' i))
    proof
     show AE x in νx. (∀ i≥card S. κ' x (F' i) = 0) ⟶ (∑ i. κ' x (F' i)) =
(∑ i<card S. κ' x (F' i))
      proof −
       {
        fix x
        assume ∀ i≥card S. κ' x (F' i) = 0
        then have (∑ i. κ' x (F' i)) = (∑ i<card S. κ' x (F' i))
         using suminf-offset[of λi. κ' x (F' i) card S]
         by(auto simp: F'-def)
       }
       thus ?thesis
        by auto
      qed
    qed
    thus ?thesis
     by auto
  qed
  moreover have AE x in νx. (∑ i. κ' x (F' i)) = κ' x (⋃ (range F'))
   using kernel-suminf-AE[OF F'-disj] by simp
  ultimately show ?thesis
   by(auto simp: 1 2)
 qed
qed

lemma kernel-disjoint-sum-AE:
 assumes B ∈ sets Y C ∈ sets Y
    and B ∩ C = {}
  shows AE x in νx. κ' x (B ∪ C) = κ' x B + κ' x C
proof −
 define F where F ≡ λb. if b then B else C
 have [simp]:disjoint-family F ⋀i. F i ∈ sets Y ⋀x. (∑ i∈UNIV. κ' x (F i)) =
κ' x B + κ' x C ⋃ (range F) = B ∪ C
  using assms by(auto simp: F-def disjoint-family-on-def comm-monoid-add-class.sum.Int-Diff[of
UNIV - {True}])
 show ?thesis
  using kernel-finite-sum-AE[of F UNIV] by auto
qed

lemma kernel-mono-AE:
 assumes B ∈ sets Y C ∈ sets Y
    and B ⊆ C
  shows AE x in νx. κ' x B ≤ κ' x C
proof −
 have 1: B ∪ (C − B) = C using assms(3) by auto
 have AE x in νx. κ' x C = κ' x B + κ' x (C − B)
  using assms by(auto intro!: kernel-disjoint-sum-AE[of B C − B,simplified 1])
```

thus *?thesis*
by *auto*
qed

lemma *kernel-incseq-AE*:
assumes *range B* ⊆ *sets Y incseq B*
shows *AE x in νx. incseq* (λ*n*. *κ′ x* (*B n*))
using *assms*(*1*) **by**(*auto simp*: *incseq-Suc-iff AE-all-countable intro*!: *kernel-mono-AE*[*OF* - - *incseq-SucD*[*OF assms*(*2*)]])

lemma *kernel-decseq-AE*:
assumes *range B* ⊆ *sets Y decseq B*
shows *AE x in νx. decseq* (λ*n*. *κ′ x* (*B n*))
using *assms*(*1*) **by**(*auto simp*: *decseq-Suc-iff AE-all-countable intro*!: *kernel-mono-AE*[*OF* - - *decseq-SucD*[*OF assms*(*2*)]])

corollary *kernel-01-AE*:
assumes *B* ∈ *sets Y*
shows *AE x in νx. 0* ≤ *κ′ x B* ∧ *κ′ x B* ≤ *1*
proof −
have {} ⊆ *B B* ⊆ *space Y*
using *assms sets.sets-into-space* **by** *auto*
from *kernel-empty0-AE kernel-Y1-AE kernel-mono-AE*[*OF* - - *this*(*1*)] *kernel-mono-AE*[*OF* - - *this*(*2*)] *assms*
show *?thesis*
by *auto*
qed

lemma *kernel-get-0*: *0* ≤ *κ′ x B*
by *simp*

lemma *kernel-le1-AE*:
assumes *B* ∈ *sets Y*
shows *AE x in νx. κ′ x B* ≤ *1*
using *kernel-01-AE*[*OF assms*] **by** *auto*

corollary *kernel-n-infty*:
assumes *B* ∈ *sets Y*
shows *AE x in νx. κ′ x B* ≠ ⊤
by(*rule AE-mp*[*OF kernel-le1-AE*[*OF assms*]],*standard*) (*auto simp*: *neq-top-trans*[*OF ennreal-one-neq-top*])

corollary *kernel-le-infty*:
assumes *B* ∈ *sets Y*
shows *AE x in νx. κ′ x B* < ⊤
using *kernel-n-infty*[*OF assms*] **by** (*simp add*: *top.not-eq-extremum*)

lemma *kernel-SUP-incseq*:
assumes *range B* ⊆ *sets Y incseq B*

shows *AE x in νx. κ′ x* (⋃ (*range B*)) = (⨆ *n. κ′ x* (*B n*))
proof −
define *Bn* **where** *Bn* ≡ (λ*n. if n* = *0 then* {} *else B* (*n* − *1*))
have *incseq Bn*
using *assms*(*2*) **by**(*auto simp*: *Bn-def incseq-def*)
define *Cn* **where** *Cn* ≡ (λ*n. Bn* (*Suc n*) − *Bn n*)
have *Cn-simp*: *Cn 0* = *B 0 Cn* (*Suc n*) = *B* (*Suc n*) − *B n* **for** *n*
by(*simp-all add*: *Cn-def Bn-def*)
have *Cn-sets*:*Cn n* ∈ *sets Y* **for** *n*
using *assms*(*1*) **by**(*induction n*) (*auto simp*: *Cn-simp*)
have *Cn-disj*: *disjoint-family Cn*
by(*auto intro*!: *disjoint-family-Suc*[*OF*] *incseq-SucD*[*OF* ‹*incseq Bn*›] *simp*: *Cn-def*)
have *Cn-un*: (⋃*k*<*Suc n. Cn k*) = *B n* **for** *n*
using *incseq-SucD*[*OF assms*(*2*)]
by (*induction n*) (*auto simp*: *Cn-simp lessThan-Suc sup-commute*)
have *Cn-sum-Bn*:*AE x in νx.* ∀ *n.* (∑ *i*<*Suc n. κ′ x* (*Cn i*)) = *κ′ x* (*B n*)
unfolding *AE-all-countable*
using *kernel-finite-sum-AE*[*OF disjoint-family-on-mono*[*OF - Cn-disj*],*of* {..<*Suc* -}] *Cn-sets*
by(*auto simp*: *Cn-un*)
have *Cn-bn-un*: (⋃ (*range B*)) = (⋃ (*range Cn*)) (**is** *?lhs* = *?rhs*)
proof *safe*
fix *n x*
assume *x* ∈ *B n*
with *Cn-un*[*of n*] **show** *x* ∈ ⋃ (*range Cn*)
by *blast*
next
fix *n x*
assume *x* ∈ *Cn n*
then show *x* ∈ ⋃ (*range B*)
by(*cases n*,*auto simp*: *Cn-simp*)
qed
hence *AE x in νx. κ′ x* (⋃ (*range B*)) = *κ′ x* (⋃ (*range Cn*))
by *simp*
moreover have *AE x in νx. κ′ x* (⋃ (*range Cn*)) = (∑ *n. κ′ x* (*Cn n*))
by(*rule AE-symmetric*[*OF kernel-suminf-AE*[*OF Cn-disj*]]) (*use Cn-def Bn-def assms*(*1*) **in** *auto*)
moreover have *AE x in νx.* (∑ *n. κ′ x* (*Cn n*)) = (⨆ *n.* ∑ *i*<*n. κ′ x* (*Cn i*))
by(*auto simp*: *suminf-eq-SUP*)
moreover have *AE x in νx.* (⨆ *n.* ∑ *i*<*n. κ′ x* (*Cn i*)) = (⨆ *n.* ∑ *i*<*Suc n. κ′ x* (*Cn i*))
proof(*intro AE-I2 antisym*)
fix *x*
show (⨆ *n.* ∑ *i*<*n. κ′ x* (*Cn i*)) ≤ (⨆ *n.* ∑ *i*<*Suc n. κ′ x* (*Cn i*))
by(*rule complete-lattice-class.Sup-mono*, *auto*, *use le-iff-add* **in** *blast*)
next
fix *x*
show (⨆ *n.* ∑ *i*<*n. κ′ x* (*Cn i*)) ≥ (⨆ *n.* ∑ *i*<*Suc n. κ′ x* (*Cn i*))

by(*rule complete-lattice-class.Sup-mono*) *blast*
qed
moreover have *AE x in νx.* $(\bigsqcup n.\ \sum i{<}Suc\ n.\ \kappa'\ x\ (Cn\ i)) = (\bigsqcup n.\ \kappa'\ x\ (B\ n))$
by(*rule AE-mp*[*OF Cn-sum-Bn*]) (*standard*+, *auto*)
ultimately show *?thesis* **by** *auto*
qed

lemma *kernel-lim-incseq*:
assumes *range B* $\subseteq$ *sets Y incseq B*
shows *AE x in νx.* $(\lambda n.\ \kappa'\ x\ (B\ n)) \longrightarrow \kappa'\ x\ (\bigcup\ (range\ B))$
by(*rule AE-mp*[*OF AE-conjI*[*OF kernel-SUP-incseq*[*OF assms*] *kernel-incseq-AE*[*OF assms*]]],*auto simp*: *LIMSEQ-SUP*)

lemma *kernel-INF-decseq*:
assumes *range B* $\subseteq$ *sets Y decseq B*
shows *AE x in νx.* $\kappa'\ x\ (\bigcap\ (range\ B)) = (\bigsqcap n.\ \kappa'\ x\ (B\ n))$
proof −
define *C* **where** $C \equiv (\lambda k.\ space\ Y - B\ k)$
have *C*:*range C* $\subseteq$ *sets Y incseq C*
using *assms* **by**(*auto simp*: *C-def decseq-def incseq-def*)
have *eq1*: *AE x in νx.* $1 - \kappa'\ x\ (\bigcap\ (range\ B)) = \kappa'\ x\ (\bigcup\ (range\ C))$
proof −
have *AE x in νx.* $\kappa'\ x\ (\bigcup\ (range\ C)) + \kappa'\ x\ (\bigcap\ (range\ B)) - \kappa'\ x\ (\bigcap\ (range\ B)) = \kappa'\ x\ (\bigcup\ (range\ C))$
using *assms*(*1*) *kernel-n-infty*[*of* $\bigcap\ (range\ B)$] **by** *auto*
moreover have *AE x in νx.* $\kappa'\ x\ (\bigcup\ (range\ C)) + \kappa'\ x\ (\bigcap\ (range\ B)) = 1$
proof −
have [*simp*]:$(\bigcup\ (range\ C)) \cup (\bigcap\ (range\ B)) = space\ Y\ (\bigcup\ (range\ C)) \cap (\bigcap\ (range\ B)) = \{\}$
by(*auto simp*: *C-def*) (*meson assms*(*1*) *range-subsetD sets.sets-into-space subsetD*)
from *kernel-disjoint-sum-AE*[*OF - - this*(*2*)] *C*(*1*) *assms*(*1*) *kernel-Y1-AE*
show *?thesis* **by** *auto*
qed
ultimately show *?thesis*
by *auto*
qed
have *eq2*: *AE x in νx.* $\kappa'\ x\ (\bigcup\ (range\ C)) = (\bigsqcup n.\ \kappa'\ x\ (C\ n))$
using *kernel-SUP-incseq*[*OF C*] **by** *auto*
have *eq3*: *AE x in νx.* $(\bigsqcup n.\ \kappa'\ x\ (C\ n)) = (\bigsqcup n.\ 1 - \ \kappa'\ x\ (B\ n))$
proof −
have *AE x in νx.* $\forall n.\ \kappa'\ x\ (C\ n) = 1 - \ \kappa'\ x\ (B\ n)$
unfolding *AE-all-countable*
proof *safe*
fix *n*
have *AE x in νx.* $\kappa'\ x\ (C\ n) + \kappa'\ x\ (B\ n) - \kappa'\ x\ (B\ n) = \kappa'\ x\ (C\ n)$
using *assms*(*1*) *kernel-n-infty*[*of B n*] **by** *auto*
moreover have *AE x in νx.* $\kappa'\ x\ (C\ n) + \kappa'\ x\ (B\ n) = 1$
proof −

```
      have [simp]: C n ∪ B n = space Y C n ∩ B n = {}
        by(auto simp: C-def) (meson assms(1) range-subsetD sets.sets-into-space
subsetD)
      thus ?thesis
        using kernel-disjoint-sum-AE[of C n B n] C(1) assms(1) kernel-Y1-AE
by fastforce
    qed
    ultimately show AE x in νx. κ' x (C n) = 1 − κ' x (B n) by auto
  qed
  thus ?thesis by auto
 qed
 have [simp]: (⨆ n. 1 − κ' x (B n)) = 1 − (⨅ n. κ' x (B n)) for x
  by(auto simp: ennreal-INF-const-minus)
 have eq: AE x in νx. 1 − κ' x (⋂ (range B)) = 1 − (⨅ n. κ' x (B n))
  using eq1 eq2 eq3 by auto
 have le1: AE x in νx. (⨅ n. κ' x (B n)) ≤ 1
 proof −
  have AE x in νx. ∀ n. κ' x (B n) ≤ 1
   using assms(1) by(auto intro!: kernel-le1-AE simp: AE-all-countable)
  thus ?thesis
   by (auto simp: INF-lower2)
 qed
 show ?thesis
   by(rule AE-mp[OF AE-conjI[OF AE-conjI[OF eq le1] kernel-le1-AE[of ⋂
(range B)]]])
    (insert assms(1),auto simp: ennreal-minus-cancel[OF ennreal-one-neq-top])
qed

lemma kernel-lim-decseq:
 assumes range B ⊆ sets Y decseq B
 shows AE x in νx. (λn. κ' x (B n)) ⟶ κ' x (⋂ (range B))
 by(rule AE-mp[OF AE-conjI[OF kernel-INF-decseq[OF assms] kernel-decseq-AE[OF
assms]]],standard,auto simp: LIMSEQ-INF)

end

lemma qlim-eq-lim-mono-at-bot:
 fixes g :: rat ⇒ 'a :: linorder-topology
 assumes mono f (g ⟶ a) at-bot ⋀r::rat. f (real-of-rat r) = g r
 shows (f ⟶ a) at-bot
proof −
 have mono g
  by(metis assms(1,3) mono-def of-rat-less-eq)
 have ga:⋀r. g r ≥ a
 proof(rule ccontr)
  fix r
  assume ¬ a ≤ g r
  then have g r < a by simp
  from order-topology-class.order-tendstoD(1)[OF assms(2) this]
```

```
    obtain Q :: rat where q: ⋀q. q ≤ Q ⟹ g r < g q
      by(auto simp: eventually-at-bot-linorder)
    define q where q ≡ min r Q
    show False
      using q[of q] ‹mono g›
      by(auto simp: q-def mono-def) (meson linorder-not-less min.cobounded1)
  qed
  show ?thesis
  proof(rule decreasing-tendsto)
    show ∀F n in at-bot. a ≤ f n
      unfolding eventually-at-bot-linorder
     by(rule exI[where x=undefined],auto) (metis Ratreal-def assms(1,3) dual-order.trans
ga less-eq-real-def lt-ex monoD of-rat-dense)
  next
    fix x
    assume a < x
    with topological-space-class.topological-tendstoD[OF assms(2),of {..<x}]
    obtain Q :: rat where q: ⋀q. q ≤ Q ⟹ g q < x
      by(auto simp: eventually-at-bot-linorder)
    show ∀F n in at-bot. f n < x
      using q assms(1,3) by(auto intro!: exI[where x=real-of-rat Q] simp: even-
tually-at-bot-linorder) (metis dual-order.refl monoD order-le-less-trans)
  qed
qed

lemma qlim-eq-lim-mono-at-top:
  fixes g :: rat ⇒ 'a :: linorder-topology
  assumes mono f (g ⟶ a) at-top ⋀r::rat. f (real-of-rat r) = g r
  shows (f ⟶ a) at-top
proof −
  have mono g
    by(metis assms(1,3) mono-def of-rat-less-eq)
  have ga:⋀r. g r ≤ a
  proof(rule ccontr)
    fix r
    assume ¬ g r ≤ a
    then have a < g r by simp
    from order-topology-class.order-tendstoD(2)[OF assms(2) this]
    obtain Q :: rat where q: ⋀q. Q ≤ q ⟹ g q < g r
      by(auto simp: eventually-at-top-linorder)
    define q where q ≡ max r Q
    show False
      using q[of q] ‹mono g› by(auto simp: q-def mono-def leD)
  qed
  show ?thesis
  proof(rule increasing-tendsto)
    show ∀F n in at-top. f n ≤ a
      unfolding eventually-at-top-linorder
     by(rule exI[where x=undefined],auto) (metis (no-types, opaque-lifting) assms(1)
```

assms(*3*) *dual-order.trans ga gt-ex monoD of-rat-dense order-le-less*)
next
fix *x*
assume $x < a$
with *topological-space-class.topological-tendstoD*[*OF assms*(*2*),*of* {*x*<..}]
obtain Q :: *rat* **where** *q*: $\bigwedge q.\ Q \leq q \Longrightarrow x < g\ q$
by(*auto simp*: *eventually-at-top-linorder*)
show $\forall_F\ n$ *in at-top*. $x < f\ n$
using *q assms*(*1*,*3*) **by**(*auto simp*: *eventually-at-top-linorder intro*!: *exI*[**where** *x*=*real-of-rat Q*]) (*metis dual-order.refl monoD order-less-le-trans*)
qed
qed

2.6 Theorem 14.D.10. (Measure Disintegration Theorem)

locale *projection-sigma-finite-standard* = *projection-sigma-finite* + *standard-borel-ne Y*
begin

theorem *measure-disintegration*:
$\exists \kappa.$ *prob-kernel X Y* $\kappa \wedge$ *measure-kernel.disintegration X Y* $\kappa\ \nu\ \nu x \wedge$
$(\forall \kappa''.$ *prob-kernel X Y* $\kappa'' \longrightarrow$ *measure-kernel.disintegration X Y* $\kappa''\ \nu\ \nu x \longrightarrow (AE\ x\ in\ \nu x.\ \kappa\ x = \kappa''\ x))$
proof −
have $*$:$\exists \kappa.$ *prob-kernel X* (*borel* :: *real measure*) $\kappa \wedge$ *measure-kernel.disintegration X borel* $\kappa\ \nu$ (*marginal-measure X borel* ν) $\wedge$
$(\forall \kappa''.$ *prob-kernel X borel* $\kappa'' \longrightarrow$ *measure-kernel.disintegration X borel* $\kappa''\ \nu$ (*marginal-measure X borel* ν) $\longrightarrow$ (*AE x in* (*marginal-measure X borel* ν). $\kappa\ x = \kappa''\ x$))
if *nu-sets'*: *sets* ν = *sets* ($X \bigotimes_M$ *borel*) **and** *marginal-sigma-finite'*: *sigma-finite-measure* (*marginal-measure X borel* ν) **for** X :: *'a measure* **and** ν

proof −
interpret *r*: *projection-sigma-finite X borel* ν
using *that* **by**(*auto simp*: *projection-sigma-finite-def*)
define φ :: *'a* ⇒ *rat* ⇒ *real*
where $\varphi \equiv (\lambda x\ r.$ *enn2real* (*r.κ' x* {..*real-of-rat r*}))
have *as1*: *AE x in r.νx*. $\forall r\ s.\ r \leq s \longrightarrow \varphi\ x\ r \leq \varphi\ x\ s$
unfolding *AE-all-countable*
proof(*safe intro*!: *AE-impI*)
fix *r s* :: *rat*
assume $r \leq s$
have *AE x in r.νx*. *r.κ' x* {..*real-of-rat k*} < *top* **for** *k*
using *atMost-borel r.kernel-le-infty* **by** *blast*
from *this*[*of s*] *r.kernel-mono-AE*[*of* {..*real-of-rat r*} {..*real-of-rat s*}] ‹$r \leq s$›
show *AE x in r.νx*. $\varphi\ x\ r \leq \varphi\ x\ s$
by(*auto simp*: φ*-def of-rat-less-eq enn2real-mono*)
qed
have *as2*: *AE x in r.νx*. $\forall r.\ (\lambda n.\ \varphi\ x\ (r + 1\ /\ rat\text{-}of\text{-}nat\ (Suc\ n))) \longrightarrow \varphi$

```
x r
      unfolding AE-all-countable
    proof safe
      fix r
      have 1:(⋂n. {..real-of-rat (r + 1 / rat-of-nat (Suc n))}) = {..real-of-rat r}
      proof safe
        fix x
        assume h: x ∈ (⋂n. {..real-of-rat (r + 1 / rat-of-nat (Suc n))})
        show x ≤ real-of-rat r
        proof(rule ccontr)
          assume ¬ x ≤ real-of-rat r
          then have 0 < x − real-of-rat r by simp
          then obtain n where (1 / (real (Suc n))) < x − real-of-rat r
            using nat-approx-posE by blast
          hence real-of-rat (r + 1 / (1 + rat-of-nat n)) < x
            by (simp add: of-rat-add of-rat-divide)
          with h show False
            using linorder-not-le by fastforce
        qed
      next
        fix x n
        assume x ≤ real-of-rat r
        then  show  x ≤ real-of-rat (r + 1 / rat-of-nat (Suc n))
            by (metis le-add-same-cancel1 of-nat-0-le-iff of-rat-less-eq order-trans
zero-le-divide-1-iff)
      qed
      have AE x in r.νx. (λn. r.κ' x {..real-of-rat (r + 1 / rat-of-nat (Suc n))})
⟶ r.κ' x {..real-of-rat r}
          unfolding 1[symmetric] by(rule r.kernel-lim-decseq) (auto simp: dec-
seq-Suc-iff of-rat-less-eq frac-le)
      from AE-conjI[OF r.kernel-le-infty[of {..real-of-rat r},simplified] this]
      show AE x in r.νx. (λn. φ x (r + 1 / (rat-of-nat (Suc n)))) ⟶ φ x r
        unfolding φ-def by eventually-elim (rule tendsto-enn2real, auto)
    qed

    have as3: AE x in r.νx. (φ x ⟶ 0) at-bot
    proof −
      have 0: range (λn. {..− real n}) ⊆ sets borel decseq (λn. {..− real n})
        by(auto simp: decseq-def)
      show ?thesis
     proof(safe intro!: AE-I2[THEN AE-mp[OF AE-conjI[OF r.kernel-empty0-AE
AE-conjI[OF r.kernel-lim-decseq[OF 0] as1]]]])
        fix x
        assume h: r.κ' x {} = 0  (λn. r.κ' x {..− real n}) ⟶ r.κ' x (⋂n. {..−
real n}) ∀r s. r ≤ s ⟶ φ x r ≤ φ x s
      have [simp]: (⋂n. {..− real n}) = {} by auto (meson le-minus-iff linorder-not-less
reals-Archimedean2)
        show (φ x ⟶ 0) at-bot
        proof(rule decreasing-tendsto)
```

```
        fix r :: real
        assume 0 < r
        with h(2) eventually-sequentially
        obtain N where N:⋀n. n ≥ N ⟹ r.κ' x {..− real n} < r
          by(fastforce simp: order-tendsto-iff h(1))
        show ∀F q in at-bot. φ x q < r
          unfolding eventually-at-bot-linorder
        proof(safe intro!: exI[where x=− rat-of-nat N])
          fix q
          assume q ≤ − rat-of-nat N
          with h(3) have φ x q ≤ φ x (− rat-of-nat N) by simp
          also have ... < r
            by(auto simp: φ-def) (metis N[OF order-refl] ‹0 < r› enn2real-less-iff
enn2real-top of-rat-minus of-rat-of-nat-eq top.not-eq-extremum)
          finally show φ x q < r .
        qed
      qed(simp add: φ-def)
    qed
  qed

  have as4: AE x in r.νx. (φ x ⟶ 1) at-top
  proof −
    have 0: range (λn. {..real n}) ⊆ sets borel incseq (λn. {..real n})
      by(auto simp: incseq-def)
    have [simp]: (⋃n. {..real n}) = UNIV by (auto simp: real-arch-simple)
     have 1: AE x in r.νx. ∀n. r.κ' x {..real n} ≤ 1 AE x in r.νx. ∀q. r.κ' x
{..real-of-rat q} ≤ 1
      by(auto simp: AE-all-countable intro!: r.kernel-le1-AE)
    show ?thesis
     proof(safe intro!: AE-I2[THEN AE-mp[OF AE-conjI[OF AE-conjI[OF 1]
AE-conjI[OF r.kernel-Y1-AE AE-conjI[OF r.kernel-lim-incseq[OF 0] as1]]],simplified]])
      fix x
      assume h: ∀q. r.κ' x {..real-of-rat q} ≤ 1 ∀n. r.κ' x {..real n} ≤ 1
            (λn. r.κ' x {..real n}) ⟶ r.κ' x UNIV ∀r s. r ≤ s ⟶ φ x r ≤
φ x s r.κ' x UNIV = 1
      then have h3: (λn. r.κ' x {..real n}) ⟶ 1
        by auto
      show (φ x ⟶ 1) at-top
      proof(rule increasing-tendsto)
        fix r :: real
        assume r < 1
        with h3 eventually-sequentially
        obtain N where N: ⋀n. n ≥ N ⟹ r < r.κ' x {..real n}
          by(fastforce simp: order-tendsto-iff)
        show ∀F n in at-top. r < φ x n
          unfolding eventually-at-top-linorder
        proof(safe intro!: exI[where x=rat-of-nat N])
          fix q
          assume rat-of-nat N ≤ q
```

```
        have r < φ x (rat-of-nat N)
              by(auto simp: φ-def) (metis N[OF order-refl] h(2) enn2real-1
enn2real-ennreal enn2real-positive-iff ennreal-cases ennreal-leI linorder-not-less zero-less-one)
        also have ... ≤ φ x q
          using h(4) ‹rat-of-nat N ≤ q› by simp
        finally show r < φ x q .
      qed
    qed(use h(1) enn2real-leI φ-def in auto)
  qed
 qed
 from AE-E3[OF AE-conjI[OF as1 AE-conjI[OF as2 AE-conjI[OF as3 as4]]],simplified
space-marginal-measure]
  obtain N where N: N ∈ null-sets r.νx ⋀x r s. x ∈ space X − N ⟹ r ≤ s
⟹ φ x r ≤ φ x s
                ⋀x r. x ∈ space X − N ⟹ (λn. φ x (r + 1 / rat-of-nat (Suc
n))) ⟶ φ x r
                ⋀x. x ∈ space X − N ⟹ (φ x ⟶ 0) at-bot ⋀x. x ∈ space X
− N ⟹ (φ x ⟶ 1) at-top
    by metis
  define F where F ≡ (λx y. indicat-real (space X − N) x * Inf {φ x r |r. y ≤
real-of-rat r} + indicat-real N x * indicat-real {0..} y)
  have [simp]: {φ x r |r. y ≤ real-of-rat r} ≠ {} for x y
    by auto (meson gt-ex less-eq-real-def of-rat-dense)
  have [simp]: bdd-below {φ x r |r. y ≤ real-of-rat r} if x ∈ space X − N for x y
  proof −
    obtain r′ where real-of-rat r′ ≤ y
      by (metis less-eq-real-def lt-ex of-rat-dense)
    from order-trans[OF this] of-rat-less-eq show ?thesis
      by(auto intro!: bdd-belowI[of - φ x r′] N(2)[OF that])
  qed
  have Feq: F x (real-of-rat r) = φ x r if x ∈ space X − N for x r
   using that N(2)[OF that] by(auto intro!: cInf-eq-minimum simp: of-rat-less-eq
F-def)
  have Fmono: mono (F x) if x ∈ space X for x
   by(auto simp: F-def mono-def indicator-def intro!: cInf-superset-mono) (meson
gt-ex less-eq-real-def of-rat-dense)

  have F1: (F x ⟶ 0) at-bot if x ∈ space X for x
  proof(cases x ∈ N)
    case True
    with that show ?thesis
      by(auto simp: F-def tendsto-iff eventually-at-bot-dense indicator-def intro!:
exI[where x=0])
  next
    case False
    with qlim-eq-lim-mono-at-bot[OF Fmono[OF that] N(4)] Feq that
    show ?thesis by auto
  qed
  have F2: (F x ⟶ 1) at-top if x ∈ space X for x
```

```
  proof(cases x ∈ N)
    case True
    with that show ?thesis
      by(auto simp: F-def tendsto-iff eventually-at-top-dense indicator-def intro!:
exI[where x=0])
  next
    case False
    with qlim-eq-lim-mono-at-top[OF Fmono[OF that] N(5)] Feq that
    show ?thesis by auto
  qed
  have F3: continuous (at-right a) (F x) if x ∈ space X for x a
  proof(cases x ∈ N)
    case x:True
    {
      fix e :: real
      assume e:0 < e
      consider a ≥ 0 | a < 0 by fastforce
      then have ∃ d>0. indicat-real {0..} (a + d) − indicat-real {0..} a < e
      proof cases
        case 1
        with e show ?thesis
          by(auto intro!: exI[where x=1])
      next
        case 2
        then obtain b where b > 0 a + b < 0
          by (metis add-less-same-cancel2 of-rat-dense real-add-less-0-iff)
        with e 2 show ?thesis
          by(auto intro!: exI[where x=b])
      qed
    }
    with x show ?thesis
    unfolding continuous-at-right-real-increasing[of F x,OF monoD[OF Fmono[OF
that]],simplified]
      by(auto simp: F-def)
  next
    case x:False
    {
      fix e :: real
      assume e: e > 0
      have ∃ k. a ≤ real-of-rat k ∧ ⋂ {φ x r |r. a ≤ real-of-rat r} + e / 2 ≥ φ
x k
      proof(rule ccontr)
        assume ∄ k. a ≤ real-of-rat k ∧ φ x k ≤ ⋂ {φ x r |r. a ≤ real-of-rat r}
+ e / 2
        then have cont: ⋀k. a ≤ real-of-rat k ⟹ φ x k − e / 2 > ⋂ {φ x r |r.
a ≤ real-of-rat r}
          by auto
        hence a ≤ real-of-rat k ⟹ ∃ r. a ≤ real-of-rat r ∧ φ x r < φ x k − e /
2 for k
```

```
        using cont ‹x ∈ space X› x cInf-less-iff [of {φ x r |r. a ≤ real-of-rat r}
φ x k − e / 2]
        by auto
      then obtain r where r:⋀k. a ≤ real-of-rat k ⟹ a ≤ real-of-rat (r k)
⋀k. a ≤ real-of-rat k ⟹ φ x (r k) < φ x k − e / 2
        by metis
      obtain k where k:a ≤ real-of-rat k
        by (meson gt-ex less-eq-real-def of-rat-dense)
      define f where f ≡ rec-nat k (λn fn. r fn)
      have f-simp: f 0 = k f (Suc n) = r (f n) for n
        by(auto simp: f-def)
      have f1: a ≤ real-of-rat (f n) for n
        using r(1) k by(induction n) (auto simp: f-simp)
      have f2: n ≥ 1 ⟹ φ x (f n) < φ x k − real n * e / 2 for n
      proof(induction n)
        case ih:(Suc n)
        consider n = 0 | n ≥ 1 by fastforce
        then show ?case
        proof cases
          case 1
          with r k show ?thesis
            by(simp add: f-simp)
        next
          case 2
          show ?thesis
              using less-trans[OF r(2)[OF f1[of n]] diff-strict-right-mono[OF
ih(1)[OF 2],of e / 2]]
            by(auto simp: f-simp ring-distribs(2) add-divide-distrib)
        qed
      qed simp
      have ¬ bdd-below {φ x r |r. a ≤ real-of-rat r}
        unfolding bdd-below-def
      proof safe
        fix M
        obtain n where φ x k − M < real n * e / 2
          using f2 e reals-Archimedean3 by fastforce
        then have φ x k − M < real (Suc n) * e / 2
          using divide-strict-right-mono pos-divide-less-eq e by fastforce
        thus Ball {φ x r |r. a ≤ real-of-rat r} ((≤) M) ⟹ False
          using f2[of Suc n] f1[of Suc n] by(auto intro!: exI[where x=φ x (f
(Suc n))])
      qed
      with that x show False
        by simp
    qed
    then obtain k where k: a ≤ real-of-rat k ⨅ {φ x r |r. a ≤ real-of-rat r}
+ e / 2 ≥ φ x k
      by auto
    obtain no where no:⋀n. n≥no ⟹ (φ x (k + 1 / rat-of-nat (Suc n))) −
```

```
(φ x k) < e / 2
      using ‹x ∈ space X› x metric-LIMSEQ-D[OF N(3)[of x k],of e/2] e N(2)[of
x k k + 1 / rat-of-nat (Suc -)]
        by(auto simp: dist-real-def)
        have ∃ d>0. ∏ {φ x r |r. a + d ≤ real-of-rat r} − ∏ {φ x r |r. a ≤
real-of-rat r} < e
      proof(safe intro!: exI[where x=real-of-rat (1 / rat-of-nat (Suc no))])
        have φ x (k + 1 / rat-of-nat (Suc no)) − e < φ x k − e / 2
          using no[OF order-refl] by simp
        also have ... ≤ ∏ {φ x r |r. a ≤ real-of-rat r}
          using k by simp
          finally have φ x (k + 1 / rat-of-nat (Suc no)) − ∏ {φ x r |r. a ≤
real-of-rat r} < e by simp
        moreover have ∏ {φ x r |r. a + real-of-rat (1 / (1 + rat-of-nat no)) ≤
real-of-rat r} ≤ φ x (k + 1 / rat-of-nat (Suc no))
          using k that x by(auto intro!: cInf-lower simp: of-rat-add)
        ultimately show ∏ {φ x r |r. a + real-of-rat (1 / (rat-of-nat (Suc no)))
≤ real-of-rat r} − ∏ {φ x r |r. a ≤ real-of-rat r} < e
          by simp
      qed simp
    }
    with that x show ?thesis
    unfolding continuous-at-right-real-increasing[of F x,OF monoD[OF Fmono[OF
that]],simplified]
      by(auto simp: F-def)
  qed

  define κ where κ ≡ (λx. interval-measure (F x))

  have κ: ⋀x. x ∈ space X ⟹ κ x UNIV = 1
          ⋀x r. x ∈ space X ⟹ κ x {..r} = ennreal (F x r)
 and[simp]: ⋀x. sets (κ x) = sets borel ⋀x. space (κ x) = UNIV
    using emeasure-interval-measure-Iic[OF - F3 F1] interval-measure-UNIV[OF
- F3 F1 F2] Fmono
    by(auto simp: mono-def κ-def)

  interpret κ: prob-kernel X borel κ
    unfolding prob-kernel-def'
  proof(rule measurable-prob-algebra-generated[OF - atMostq-Int-stable,of - UNIV])
    show ⋀a. a ∈ space X ⟹ prob-space (κ a)
      by(auto intro!: prob-spaceI κ(1))
  next
    fix A
    assume A ∈ {{..r} |r::real. r ∈ ℚ}
    then obtain r where r: A = {..real-of-rat r}
      using Rats-cases by blast
    have (λx. ennreal (indicat-real (space X − N) x * φ x r + indicat-real N x
* indicat-real {0..} (real-of-rat r))) ∈ borel-measurable X
    proof −
```

have $N \in sets\ X$
using *null-setsD2*[*OF* $N(1)$] **by**(*auto simp*: *sets-marginal-measure*)
thus *?thesis* **by**(*auto simp*: φ-*def*)
qed
moreover have *indicat-real* (*space* $X - N$) $x * \varphi\ x\ r +$ *indicat-real* $N\ x *$ *indicat-real* $\{0..\}$ (*real-of-rat* r) = *emeasure* ($\kappa\ x$) A **if** $x \in space\ X$ **for** x
using *Feq*[*of* $x\ r$] $\kappa(2)$[*OF that*,*of real-of-rat* r]
by(*cases* $x \in N$) (*auto simp*: r *indicator-def F-def*)
ultimately show $(\lambda x.\ emeasure\ (\kappa\ x)\ A) \in borel\text{-}measurable\ X$
using *measurable-cong*[*of* - $\lambda x.$ *emeasure* ($\kappa\ x$) A $\lambda x.$ *ennreal* (*indicat-real* (*space* $X - N$) $x * \varphi\ x\ r +$ *indicat-real* $N\ x *$ *indicat-real* $\{0..\}$ (*real-of-rat* r))]
by *simp*
qed(*auto simp*: *rborel-eq-atMostq*)
have κ-*AE*:*AE* x *in* $r.\nu x.\ \kappa\ x\ \{..real\text{-}of\text{-}rat\ r\} = r.\kappa'\ x\ \{..real\text{-}of\text{-}rat\ r\}$ **for** r
proof −
have *AE* x *in* $r.\nu x.\ \kappa\ x\ \{..real\text{-}of\text{-}rat\ r\} = ennreal\ (F\ x\ (real\text{-}of\text{-}rat\ r))$
by(*auto simp*: *space-marginal-measure* $\kappa(2)$)
moreover have *AE* x *in* $r.\nu x.$ *ennreal* ($F\ x$ (*real-of-rat* r))= *ennreal* ($\varphi\ x\ r$)
using *Feq*[*of* - r] **by**(*auto simp add*: *space-marginal-measure intro*!: *AE-I''*[*OF* $N(1)$])
moreover have *AE* x *in* $r.\nu x.$ *ennreal* ($\varphi\ x\ r$) = *ennreal* (*enn2real* ($r.\kappa'\ x$ $\{..real\text{-}of\text{-}rat\ r\}$))
by(*simp add*: φ-*def*)
moreover have *AE* x *in* $r.\nu x.$ *ennreal* (*enn2real* ($r.\kappa'\ x\ \{..real\text{-}of\text{-}rat\ r\}$)) = $r.\kappa'\ x\ \{..real\text{-}of\text{-}rat\ r\}$
using r.*kernel-le-infty*[*of* $\{..real\text{-}of\text{-}rat\ r\}$,*simplified*]
by(*auto simp*: *ennreal-enn2real-if*)
ultimately show *?thesis* **by** *auto*
qed
have κ-*dis*: $\kappa.disintegration\ \nu\ r.\nu x$
proof −
interpret *D*: *Dynkin-system UNIV* $\{B \in sets\ borel.\ \forall A \in sets\ X.\ \nu\ (A \times B) = (\int^+ x \in A.\ (\kappa\ x)\ B\partial r.\nu x)\}$
proof
{
fix A
assume h:$A \in sets\ X$
then have $\nu\ (A \times UNIV) = (\int^+ x \in A.\ 1\ \partial r.\nu x)$
using *emeasure-marginal-measure*[*OF nu-sets'* h] *sets-marginal-measure*[*of* X *borel* ν *space borel*] **by** *auto*
also have ... = $(\int^+ x \in A.\ (\kappa\ x)\ UNIV\partial r.\nu x)$
by(*auto intro*!: *nn-integral-cong simp*: κ *space-marginal-measure*)
finally have $\nu\ (A \times UNIV) = (\int^+ x \in A.\ emeasure\ (\kappa\ x)\ UNIV\partial r.\nu x)$ **.**
}
thus $UNIV \in \{B \in sets\ borel.\ \forall A \in sets\ X.\ emeasure\ \nu\ (A \times B) = (\int^+ x \in A.\ emeasure\ (\kappa\ x)\ B\partial r.\nu x)\}$
by *auto*
hence *univ*:$\bigwedge A.\ A \in sets\ X \Longrightarrow \nu\ (A \times UNIV) = (\int^+ x \in A.\ emeasure\ (\kappa\ x)\ UNIV\partial r.\nu x)$ **by** *auto*

show $\bigwedge B.\ B \in \{B \in \mathit{sets\ borel}.\ \forall A{\in}\mathit{sets}\ X.\ \mathit{emeasure}\ \nu\ (A \times B) = (\int^+ x{\in}A.\ \mathit{emeasure}\ (\kappa\ x)\ B\partial r.\nu x)\}$
$\Longrightarrow\ \mathit{UNIV} - B \in \{B \in \mathit{sets\ borel}.\ \forall A{\in}\mathit{sets}\ X.\ \mathit{emeasure}\ \nu\ (A \times B) = (\int^+ x{\in}A.\ \mathit{emeasure}\ (\kappa\ x)\ B\partial r.\nu x)\}$
proof(*rule r.νx.sigma-finite-disjoint*)
fix B **and** $J :: \mathit{nat} \Rightarrow$ -
assume $B \in \{B \in \mathit{sets\ borel}.\ \forall A{\in}\mathit{sets}\ X.\ \mathit{emeasure}\ \nu\ (A \times B) = (\int^+ x{\in}A.\ \mathit{emeasure}\ (\kappa\ x)\ B\ \partial r.\nu x)\}$ *range* $J \subseteq$ *sets* $r.\nu x \bigcup$ (*range* J) = *space* $r.\nu x$
$(\bigwedge i.\ \mathit{emeasure}\ r.\nu x\ (J\ i) \neq \infty)$ *disjoint-family* J
then have B: $B \in \mathit{sets\ borel}\ \ \forall A{\in}\mathit{sets}\ X.\ \nu\ (A \times B) = (\int^+ x{\in}A.\ (\kappa\ x)\ B\partial r.\nu x)$
and J: *range* $J \subseteq$ *sets* $X \bigcup$ (*range* J) = *space* $X\ \bigwedge i.\ \mathit{emeasure}\ r.\nu x\ (J\ i) \neq \infty$ *disjoint-family* J
by (*auto simp*: *sets-marginal-measure space-marginal-measure*)
{
fix A
assume A: $A \in \mathit{sets}\ X$
have $\nu\ (A \times (\mathit{UNIV} - B)) = (\int^+ x{\in}A.\ (\kappa\ x)\ (\mathit{UNIV} - B)\partial r.\nu x)$ (**is** *?lhs* = *?rhs*)
proof −
have *AJi1*: *disjoint-family* $(\lambda i.\ (A \cap J\ i) \times (\mathit{UNIV} - B))$
using *B*(*1*) *J*(*4*) **by**(*fastforce simp*: *disjoint-family-on-def*)
have *AJi2*[*simp*]: $(\bigcup i.\ ((A \cap J\ i) \times (\mathit{UNIV} - B))) = A \times (\mathit{UNIV} - B)$
using *J*(*2*) *sets.sets-into-space*[*OF A*] **by** *blast*
have *AJi3*: (*range* $(\lambda i.\ (A \cap J\ i) \times (\mathit{UNIV} - B))) \subseteq$ *sets* ν
using *B*(*1*) *J*(*1*) *A* **by**(*auto simp*: *nu-sets′*)

have *?lhs* = $(\sum i.\ \nu\ ((A \cap J\ i) \times (\mathit{UNIV} - B)))$
by(*simp add*: *suminf-emeasure*[*OF AJi3 AJi1*])
also have ... = $(\sum i.\ (\int^+ x{\in}\ A \cap J\ i.\ (\kappa\ x)\ (\mathit{UNIV} - B)\ \partial r.\nu x))$
proof(*safe intro*!: *suminf-cong*)
fix n
have *Jn*: $J\ n \in \mathit{sets}\ X$
using J **by** *auto*
have *fin*: $\nu\ ((A \cap J\ n) \times C) \neq \infty$ **for** C
proof(*cases* $(A \cap J\ n) \times C \in \mathit{sets}\ \nu$)
case *True*
then have $\nu\ ((A \cap J\ n) \times C) \leq \nu\ ((A \cap J\ n) \times \mathit{UNIV})$
using *Jn nu-sets′ A* **by**(*intro emeasure-mono*) *auto*
also have $\nu\ ((A \cap J\ n) \times \mathit{UNIV}) \leq \nu\ (J\ n \times \mathit{UNIV})$
using *Jn nu-sets′* **by**(*intro emeasure-mono*) *auto*
also have ... = $r.\nu x\ (J\ n)$
using *emeasure-marginal-measure*[*OF nu-sets′ Jn*] **by** *simp*
finally show *?thesis*
by (*metis J*(*3*)[*of n*] *infinity-ennreal-def neq-top-trans*)
qed(*simp add*: *emeasure-notin-sets*)
show $\nu\ ((A \cap J\ n) \times (\mathit{UNIV} - B)) = (\int^+ x{\in}\ A \cap J\ n.\ (\kappa\ x)\ (\mathit{UNIV} - B)\ \partial r.\nu x)$ (**is** *?lhs* = *?rhs*)

proof −
have *?lhs* = ν (($A \cap J\ n$) × *UNIV*) − ν (($A \cap J\ n$) × B)
proof −
have [*simp*]: *?lhs* + ν (($A \cap J\ n$) × B) = ν (($A \cap J\ n$) × *UNIV*)
proof −
have [*simp*]:(($A \cap J\ n$) × (*UNIV* − B)) ∪ (($A \cap J\ n$) × B) = (($A \cap J\ n$) × *UNIV*) **by** *blast*
show *?thesis*
using *B*(*1*) *A Jn nu-sets*′ **by**(*intro plus-emeasure*[*of* ($A \cap J\ n$) × (*UNIV* − B) - ($A \cap J\ n$) × B,*simplified*]) *auto*
qed
have *?lhs* = *?lhs* + ν (($A \cap J\ n$) × B) − ν (($A \cap J\ n$) × B)
by(*simp only*: *ennreal-add-diff-cancel*[*OF fin*[*of B*]])
also have ... = ν (($A \cap J\ n$) × *UNIV*) − ν (($A \cap J\ n$) × B)
by *simp*
finally show *?thesis* **.**
qed
also have ... = ($\int^+ x \in A \cap J\ n.\ (\kappa\ x)$ *UNIV* $\partial r.\nu x$) − ($\int^+ x \in A \cap J\ n.\ (\kappa\ x)\ B\ \partial r.\nu x$)
using *B*(*2*) *A Jn univ* **by** *auto*
also have ... = ($\int^+ x.$ (($\kappa\ x$) *UNIV* ∗ *indicator* ($A \cap J\ n$) x − ($\kappa\ x$) B ∗ *indicator* ($A \cap J\ n$) x) $\partial r.\nu x$)
proof(*rule nn-integral-diff*[*symmetric*])
show ($\lambda x.\ (\kappa\ x)$ *UNIV* ∗ *indicator* ($A \cap J\ n$) x) ∈ *borel-measurable r*.νx ($\lambda x.\ (\kappa\ x)\ B$ ∗ *indicator* ($A \cap J\ n$) x) ∈ *borel-measurable r*.νx
using *sets-marginal-measure*[*of X borel* ν *space borel*] κ.*emeasure-measurable*[*OF B*(*1*)] κ.*emeasure-measurable*[*of UNIV*] *A Jn*
by(*auto simp del*: *space-borel*)
next
show ($\int^+ x \in A \cap J\ n.\ (\kappa\ x)\ B \partial r.\nu x$) ≠ ∞
using *B*(*2*) *A Jn univ fin*[*of B*] **by** *auto*
next
show *AE x in r*.νx. ($\kappa\ x$) B ∗ *indicator* ($A \cap J\ n$) x ≤ ($\kappa\ x$) *UNIV* ∗ *indicator* ($A \cap J\ n$) x
by(*standard*, *auto simp*: *space-marginal-measure indicator-def intro*!: *emeasure-mono*)
qed
also have ... = ($\int^+ x \in A \cap J\ n.$ (($\kappa\ x$) *UNIV* − ($\kappa\ x$) B) $\partial r.\nu x$)
by(*auto intro*!: *nn-integral-cong simp*: *indicator-def*)
also have ... = *?rhs*
proof(*safe intro*!: *nn-integral-cong*)
fix x
assume $x \in$ *space r*.νx
then have $x \in$ *space X*
by(*simp add*: *space-marginal-measure*)
show (($\kappa\ x$) *UNIV* − ($\kappa\ x$) B) ∗ *indicator* ($A \cap J\ n$) x = ($\kappa\ x$) (*UNIV* − B) ∗ *indicator* ($A \cap J\ n$) x
by(*auto intro*!: *emeasure-compl*[*of B* κ *x*,*simplified*,*symmetric*] *simp*: *B* κ.*prob-spaces* ‹$x \in$ *space X*› *prob-space-imp-subprob-space subprob-space.emeasure-subprob-space-less-top*

indicator-def)
qed
finally show *?thesis* .
qed
qed
also have ... = ($\int^+ x$. ($\sum i$. (κ x) (*UNIV* − *B*) ∗ *indicator* (*A* ∩ *J i*) *x*)∂*r*.ν*x*)
using κ.*emeasure-measurable*[*of UNIV* − *B*] *B*(*1*) *sets-marginal-measure*[*of X borel* ν *space borel*] *A J*
by(*intro nn-integral-suminf*[*symmetric*]) (*auto simp del*: *space-borel*)
also have ... = ($\int^+ x$. (κ x) (*UNIV* − *B*) ∗ *indicator A x* ∗ ($\sum i$. *indicator* (*A* ∩ *J i*) *x*) ∂*r*.ν*x*)
by(*auto simp*: *indicator-def intro*!: *nn-integral-cong*)
also have ... = ($\int^+ x$. (κ x) (*UNIV* − *B*) ∗ *indicator A x* ∗ (*indicator* ($\bigcup i$. *A* ∩ *J i*) *x*) ∂*r*.ν*x*)
proof −
have ($\sum i$. *indicator* (*A* ∩ *J i*) *x*) = (*indicator* ($\bigcup i$. *A* ∩ *J i*) *x* :: *ennreal*) **for** *x*
using *J*(*4*) by(*intro suminf-indicator*) (*auto simp*: *disjoint-family-on-def*)
thus *?thesis*
by(*auto intro*!: *nn-integral-cong*)
qed
also have ... = *?rhs*
using *J*(*2*) by(*auto simp*: *indicator-def space-marginal-measure intro*!: *nn-integral-cong*)
finally show *?thesis* .
qed
}
thus *UNIV* − *B* ∈ {*B* ∈ *sets borel*. ∀ *A*∈*sets X*. *emeasure* ν (*A* × *B*) = ($\int^+ x$∈*A*. *emeasure* (κ x) *B* ∂*r*.ν*x*)}
using *B* by *auto*
qed
next
fix *J* :: *nat* ⇒ -
assume *J1*: *disjoint-family J range J* ⊆ {*B* ∈ *sets borel*. ∀ *A*∈*sets X*. ν (*A* × *B*) = ($\int^+ x$∈*A*. (κ x) *B*∂*r*.ν*x*)}
then have *J2*: *range J* ⊆ *sets borel* $\bigcup$ (*range J*) ∈ *sets borel* $\bigwedge n$ *A*. *A* ∈ *sets X* ⟹ ν (*A* × (*J n*)) = ($\int^+ x$∈*A*. (κ x) (*J n*) ∂*r*.ν*x*)
by *auto*
show $\bigcup$ (*range J*) ∈ {*B* ∈ *sets borel*. ∀ *A*∈*sets X*. ν (*A* × *B*) = ($\int^+ x$∈*A*. (κ x) *B*∂*r*.ν*x*)}
proof −
{
fix *A*
assume *A*:*A* ∈ *sets X*
have ν (*A* × $\bigcup$ (*range J*)) = ($\int^+ x$∈*A*. (κ x) ($\bigcup$ (*range J*)) ∂*r*.ν*x*) (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = ν ($\bigcup n$. *A* × *J n*)

```
            proof −
              have (A × ⋃ (range J)) = (⋃n. A × J n) by blast
              thus ?thesis by simp
            qed
            also have ... = (∑ n. ν (A × J n))
          using J1(1) J2(1) A nu-sets' by(fastforce intro!: suminf-emeasure[symmetric]
simp: disjoint-family-on-def)
            also have ... = (∑ n. (∫⁺x∈A. (κ x) (J n) ∂r.νx))
              by(simp add: J2(3)[OF A])
            also have ... = (∫⁺x. (∑ n. (κ x) (J n) ∗ indicator A x) ∂r.νx)
                using κ.emeasure-measurable J2(1) A sets-marginal-measure[of X
borel ν space borel]
              by(intro nn-integral-suminf[symmetric]) auto
            also have ... = (∫⁺x∈A. (∑ n. (κ x) (J n)) ∂r.νx)
              by auto
            also have ... = (∫⁺x∈A. (κ x) (⋃ (range J)) ∂r.νx)
              using J1 J2 by(auto intro!: nn-integral-cong suminf-emeasure simp:
space-marginal-measure indicator-def)
            finally show ?thesis .
          qed
        }
        thus ?thesis
          using J2(2) by auto
      qed
    qed auto
    have { {..r} | r::real. r ∈ ℚ} ⊆ {B ∈ sets borel. ∀ A∈ sets X. ν (A × B) =
(∫⁺x∈A. (κ x) B∂r.νx)}
    proof −
      {
        fix r ::real and A
        assume h: r ∈ ℚ A ∈ sets X
        then obtain r' where r':r = real-of-rat r'
          using Rats-cases by blast
        have ν (A × {..r}) = (∫⁺x∈A. (κ x) {..r} ∂r.νx) (is ?lhs = ?rhs)
        proof −
          have ?lhs = (∫⁺x∈A. r.κ' x {..r}∂r.νx)
            using h by(simp add: r.kernel-RN-deriv)
          also have ... = ?rhs
            using κ-AE[of r'] by(auto intro!: nn-integral-cong-AE simp: r' simp
del: space-borel)
          finally show ?thesis .
        qed
      }
      thus ?thesis
        by auto
    qed
    from D.Dynkin-subset[OF this] rborel-eq-atMostq[symmetric]
    show ?thesis
    by(auto simp: κ.disintegration-def sets-marginal-measure nu-sets' sigma-eq-Dynkin[OF
```

- atMostq-Int-stable,of UNIV,simplified,symmetric] rborel-eq-atMostq-sets simp del: space-borel)
qed
show *?thesis*
proof(*intro exI conjI strip*)
fix κ''
assume *prob-kernel X* (*borel* :: *real measure*) κ''
interpret κ'': *prob-kernel X borel* κ'' **by** *fact*
assume *disi*: κ''.*disintegration* ν *r.*νx
have *eq-atMostr-AE:AE x in r.*νx. $\forall$ *r.* κ *x* {*..real-of-rat r*} = κ'' *x* {*..real-of-rat r*}
unfolding *AE-all-countable*
proof *safe*
fix *r*
have *AE x in r.*νx. (κ'' *x*) {*..real-of-rat r*} = *r.*κ' *x* {*..real-of-rat r*}
proof(*safe intro!: r.*νx*.RN-deriv-unique*[*of* λx. κ'' *x* {*..real-of-rat r*} *marginal-measure-on X borel* ν {*..real-of-rat r*},*simplified r.*κ'*-def*[*of - {..real-of-rat r},symmetric*]])
show *1*:(λx. *emeasure* (κ'' *x*) {*..real-of-rat r*}) $\in$ *borel-measurable r.*νx
using κ''.*emeasure-measurable*[*of* {*..real-of-rat r*}] *sets-marginal-measure*[*of X borel* ν *space borel*] **by** *simp*
show *density r.*νx (λx. *emeasure* (κ'' *x*) {*..real-of-rat r*}) = *marginal-measure-on X borel* ν {*..real-of-rat r*}
proof(*rule measure-eqI*)
fix *A*
assume *A* $\in$ *sets* (*density r.*νx (λx. (κ'' *x*) {*..real-of-rat r*}))
then have *A* [*measurable*]:*A* $\in$ *sets X*
by(*simp add*: *sets-marginal-measure*)
show *emeasure* (*density r.*νx (λx. *emeasure* (κ'' *x*) {*..real-of-rat r*})) *A* = *emeasure* (*marginal-measure-on X borel* ν {*..real-of-rat r*}) *A* (**is** *?lhs = ?rhs*)
proof −
have *?lhs* = ($\int^+ x \in A$. (κ'' *x*) {*..real-of-rat r*} ∂*r.*νx)
using *emeasure-density*[*OF 1*,*of A*] *A*
by(*simp add*: *sets-marginal-measure*)
also have *...* = ν (*A* × {*..real-of-rat r*})
using *disi A* **by**(*auto simp*: κ''.*disintegration-def*)
also have *...* = *?rhs*
by(*simp add*: *emeasure-marginal-measure-on*[*OF nu-sets' - A*])
finally show *?thesis* **.**
qed
qed(*simp add*: *sets-marginal-measure*)
qed
with κ*-AE*[*of r*]
show *AE x in r.*νx. κ *x* {*..real-of-rat r*} = κ'' *x* {*..real-of-rat r*}
by *auto*
qed
{ fix *x*
assume *h*:*x* $\in$ *space r.*νx $\forall$ *r.* (κ *x*) {*..real-of-rat r*} = (κ'' *x*) {*..real-of-rat r*}
then have *x*: *x* $\in$ *space X*

by(*simp add*: *space-marginal-measure*)
have $\kappa\ x = \kappa''\ x$
proof(*rule measure-eqI-generator-eq*[*OF atMostq-Int-stable*,*of UNIV - - λn. {..real n}*])
show $\bigwedge A.\ A \in \{\{..r\}\ |r.\ r \in \mathbb{Q}\} \Longrightarrow (\kappa\ x)\ A = (\kappa''\ x)\ A$
using *h*(*2*) *Rats-cases* **by** *auto*
next
show $(\bigcup n.\ \{..real\ n\}) = UNIV$
by (*simp add*: *real-arch-simple subsetI subset-antisym*)
next
fix *n*
have $(\kappa\ x)\ \{..real\ n\} \leq \kappa\ x\ UNIV$
by(*auto intro*!: *emeasure-mono*)
also have *... = 1*
by(*rule* κ(*1*)[*OF x*])
finally show $(\kappa\ x)\ \{..real\ n\} \neq \infty$
using *linorder-not-le* **by** *fastforce*
next
show *range* $(\lambda n.\ \{..real\ n\}) \subseteq \{\{..r\}\ |r.\ r \in \mathbb{Q}\}$
using *Rats-of-nat* **by** *blast*
qed(*auto simp*: κ.*kernel-sets*[*OF x*] κ''.*kernel-sets*[*OF x*] *rborel-eq-atMostq-sets*)
}
then show *AE x in r.*ν*x.* $\kappa\ x = \kappa''\ x$
using *eq-atMostr-AE* **by** *fastforce*
qed(*auto simp del*: *space-borel simp add*: κ-*dis* κ.*prob-kernel-axioms*)
qed

show *?thesis*
proof −
define ν' **where** ν' = *distr* ν $(X \bigotimes_M borel)$ (λ(*x*,*y*). (*x*, *to-real y*))
have ν-*distr*:ν = *distr* ν' $(X \bigotimes_M Y)$ (λ(*x*,*y*). (*x*, *from-real y*))
using *nu-sets sets-eq-imp-space-eq*[*OF nu-sets*] *from-real-to-real*
by(*auto simp*: ν'-*def distr-distr space-pair-measure intro*!: *distr-id'*[*symmetric*])
have ν*x-eq*:(*marginal-measure X borel* ν') = ν*x*
using *emeasure-marginal-measure*[*of* ν' *X borel*] *emeasure-marginal-measure*[*OF nu-sets*] *sets-eq-imp-space-eq*[*OF nu-sets*]
by(*auto intro*!: *measure-eqI simp*: *sets-marginal-measure* ν'-*def emeasure-distr map-prod-vimage*[*of id to-real*,*simplified map-prod-def id-def*] *space-pair-measure Times-Int-Times*)
interpret ν' : *projection-sigma-finite X borel* ν'
by(*auto simp*: *projection-sigma-finite-def* ν*x-eq* ν*x.sigma-finite-measure-axioms simp del*: *space-borel*,*auto simp add*: ν'-*def*)
obtain κ' **where** κ': *prob-kernel X borel* κ' *measure-kernel.disintegration X borel* κ' ν' ν'.ν*x*
$\bigwedge$κ''. *prob-kernel X borel* κ'' $\Longrightarrow$ *measure-kernel.disintegration X borel* κ'' ν' ν'.ν*x* $\Longrightarrow$ (*AE x in* ν'.ν*x.* κ' *x* = κ'' *x*)
using ∗[*of* ν' *X*] ν'.*nu-sets* ν'.ν*x.sigma-finite-measure-axioms* **by** *blast*
interpret κ': *prob-kernel X borel* κ' **by** *fact*
define κ **where** κ ≡ (λ*x*. *distr* (κ' *x*) *Y from-real*)

interpret κ: *prob-kernel X Y* κ
by(*auto simp*: *prob-kernel-def'* κ*-def*)
have *disi*: κ*.disintegration* ν νx
proof(*rule* κ*.disintegrationI*)
fix *A B*
assume *A*[*measurable*]:*A* ∈ *sets X* **and** *B*[*measurable*]: *B* ∈ *sets Y*
have [*measurable*]: *from-real* −‘ *B* ∈ *sets borel*
by(*simp add*: *measurable-sets-borel*[*OF - B*])
show ν (*A* × *B*) = (∫ $^+$*x*∈*A*. κ *x B*∂νx) (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = ν' (*A* × (*from-real* −‘ *B*))
by(*auto simp*: ν*-distr emeasure-distr map-prod-vimage*[*of id from-real*,*simplified map-prod-def id-def*])
also have ... = (∫ $^+$*x*∈*A*. κ' *x* (*from-real* −‘ *B*) ∂νx)
using κ'*.disintegrationD*[*OF* κ'(*2*),*of A from-real* −‘ *B*]
by(*auto simp add*: ν*x-eq simp del*: *space-borel*)
also have ... = *?rhs*
by(*auto intro*!: *nn-integral-cong simp*: *space-marginal-measure* κ*-def emeasure-distr*)
finally show *?thesis* **.**
qed
qed(*simp-all add*: *sets-marginal-measure nu-sets*)

show *?thesis*
proof(*safe intro*!: *exI*[**where** *x*=κ])
fix κ''
assume *h*:*prob-kernel X Y* κ''
measure-kernel.disintegration X Y κ'' ν νx
interpret κ'': *prob-kernel X Y* κ'' **by** *fact*
show *AE x in* νx. κ *x* = κ'' *x*
proof −
define κ''' **where** $\kappa''' \equiv (\lambda x.\ distr\ (\kappa''\ x)\ borel\ to\text{-}real)$
interpret κ''': *prob-kernel X borel* κ'''
by(*auto simp*: *prob-kernel-def'* κ'''*-def*)
have κ''*-def*: κ'' *x* = *distr* (κ''' *x*) *Y from-real* **if** *x* ∈ *space X* **for** *x*
using *distr-distr*[*of from-real borel Y to-real* κ'' *x*,*simplified measurable-cong-sets*[*OF* κ''*.kernel-sets*[*OF that*] *refl*,*of borel*]]
by(*auto simp*: κ'''*-def comp-def* κ''*.kernel-sets*[*OF that*] *measurable-cong-sets*[*OF* κ''*.kernel-sets*[*OF that*] κ''*.kernel-sets*[*OF that*]] *sets-eq-imp-space-eq*[*OF* κ''*.kernel-sets*[*OF that*]] *intro*!: *distr-id'*[*symmetric*])
have κ'''*-disi*: κ'''*.disintegration* ν' ν'.νx
proof(*rule* κ'''*.disintegrationI*)
fix *A* **and** *B* :: *real set*
assume *A*[*measurable*]:*A* ∈ *sets X* **and** *B*[*measurable*]:*B* ∈ *sets borel*
show ν' (*A* × *B*) = (∫ $^+$*x*∈*A*. (κ''' *x*) *B* ∂ν'.νx) (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = ν (*A* × (*to-real* −‘ *B* ∩ *space Y*))
by(*auto simp*: ν'*-def emeasure-distr map-prod-vimage*[*of id to-real*,*simplified map-prod-def id-def*] *sets-eq-imp-space-eq*[*OF nu-sets*] *space-pair-measure Times-Int-Times*)

```
    also have ... = (∫⁺x∈A. (κ'' x) (to-real −‘ B ∩ space Y) ∂νx)
      using κ''.disintegrationD[OF h(2) A,of to-real −‘ B ∩ space Y] by
auto
    also have ... = ?rhs
     by(auto simp: νx-eq[symmetric] space-marginal-measure κ'''-def emea-
sure-distr sets-eq-imp-space-eq[OF κ''.kernel-sets] intro!: nn-integral-cong)
    finally show ?thesis .
   qed
  qed(auto simp: ν'-def sets-marginal-measure)
  show ?thesis
    by(rule AE-mp[OF κ'(3)[OF κ'''.prob-kernel-axioms κ'''-disi,simplified
νx-eq]],standard) (auto simp: space-marginal-measure κ''-def κ-def)
 qed
 qed(simp-all add: disi κ.prob-kernel-axioms)
qed
qed

end
```

2.7 Lemma 14.D.12.

```
lemma ex-finite-density-measure:
  fixes A :: nat ⇒ -
  assumes A: range A ⊆ sets M ⋃ (range A) = space M ⋀i. emeasure M (A i)
≠ ∞ disjoint-family A
  defines h ≡ (λx. (∑ n. (1/2)^(Suc n) * (1 / (1 + M (A n))) * indicator (A
n) x))
  shows h ∈ borel-measurable M
        ⋀x. x ∈ space M ⟹ 0 < h x
        ⋀x. x ∈ space M ⟹ h x < 1
        finite-measure (density M h)
proof −
  have less1:0 < 1 / (1 + M (A n)) 1 / (1 + M (A n)) ≤ 1 for n
    using A(3)[of n] ennreal-zero-less-divide[of 1 1 + M (A n)]
    by (auto intro!: divide-le-posI-ennreal simp: add-pos-nonneg)
  show [measurable]:h ∈ borel-measurable M
    using A by(simp add: h-def)
  {
    fix x
    assume x:x ∈ space M
    then obtain i where i: x ∈ A i
      using A(2) by auto
    show 0 < h x
      using A(3)[of i] less1[of i]
     by(auto simp: h-def suminf-pos-iff i ennreal-divide-times ennreal-zero-less-divide
power-divide-distrib-ennreal power-less-top-ennreal intro!: exI[where x=i])
    have h x = (∑ n. (1/2)^(Suc n + 2) * (1 / (1 + M (A (n + 2)))) * indicator
(A (n + 2)) x) + (1/2) * (1 / (1 + M (A 0))) * indicator (A 0) x + (1/2)^2
* (1 / (1 + M (A 1))) * indicator (A 1) x
```

by(*auto simp*: *h-def suminf-split-head suminf-offset*[*of* λ*n*. (*1*/*2*)^(*Suc n*) ∗ (*1* / (*1* + *M* (*A n*))) ∗ *indicator* (*A n*) *x* *2*] *simp del*: *power-Suc sum-mult-indicator*) (*auto simp*: *numeral-2-eq-2*)
also have ... ≤ *1*/*4* + (*1*/*2*) ∗ (*1* / (*1* + *M* (*A* *0*))) ∗ *indicator* (*A* *0*) *x* + (*1*/*2*)^*2* ∗ (*1* / (*1* + *M* (*A* *1*))) ∗ *indicator* (*A* *1*) *x*
proof −
have (∑ *n*. (*1*/*2*)^(*Suc n* + *2*) ∗ (*1* / (*1* + *M* (*A* (*n* + *2*)))) ∗ *indicator* (*A* (*n* + *2*)) *x*) ≤ (∑ *n*. (*1*/*2*)^(*Suc n* + *2*))
using *less1*(*2*)[*of Suc* (*Suc* -)] **by**(*intro suminf-le*,*auto simp*: *indicator-def*) (*metis mult.right-neutral mult-left-mono zero-le*)
also have ... = (∑ *n*. *ennreal* ((*1* / *2*) ^ (*Suc n* + *2*)))
by(*simp only*: *ennreal-power*[*of 1*/*2*,*symmetric*]) (*metis divide-ennreal ennreal-1 ennreal-numeral linorder-not-le not-one-less-zero zero-less-numeral*)
also have ... = *ennreal* (∑ *n*. (*1* / *2*) ^ (*Suc n* + *2*))
by(*rule suminf-ennreal2*) *auto*
also have ... = *ennreal* (*1*/*4*)
using *nsum-of-r*′[*of 1*/*2 Suc* (*Suc* (*Suc* *0*)) *1*] **by** *auto*
also have ... = *1* / *4*
by (*metis ennreal-divide-numeral ennreal-numeral numeral-One zero-less-one-class.zero-le-one*)
finally show *?thesis* **by** *simp*
qed
also have ... < *1* (**is** *?lhs* < -)
proof(*cases x* ∈ *A* *0*)
case *True*
then have *x* ∉ *A* *1*
using *A*(*4*) **by** (*auto simp*: *disjoint-family-on-def*)
hence *?lhs* = *1* / *4* + *1* / *2* ∗ (*1* / (*1* + *emeasure M* (*A* *0*)))
by(*simp add*: *True*)
also have ... ≤ *1* / *4* + *1* / *2*
using *less1*(*2*)[*of 0*] **by** (*simp add*: *divide-right-mono-ennreal ennreal-divide-times*)
also have ... = *1* / *4* + *2* / *4*
using *divide-mult-eq*[*of 2 1 2*] **by** *simp*
also have ... = *3* / *4*
by(*simp add*: *add-divide-distrib-ennreal*[*symmetric*])
also have ... < *1*
by(*simp add*: *divide-less-ennreal*)
finally show *?thesis* **.**
next
case *False*
then have *?lhs* = *1* / *4* + (*1* / *2*)^2 ∗ (*1* / (*1* + *emeasure M* (*A* *1*))) ∗ *indicator* (*A* *1*) *x*
by *simp*
also have ... ≤ *1* / *4* + (*1* / *2*)^2
by (*metis less1*(*2*)[*of 1*] *add-left-mono indicator-eq-0-iff indicator-eq-1-iff mult.right-neutral mult-eq-0-iff mult-left-mono zero-le*)
also have ... = *2* / *4*
by(*simp add*: *power-divide-distrib-ennreal add-divide-distrib-ennreal*[*symmetric*])
also have ... < *1*
by(*simp add*: *divide-less-ennreal*)

```
    finally show ?thesis .
  qed
  finally show h x < 1 .
 }
 show finite-measure (density M h)
 proof
  show emeasure (density M h) (space (density M h)) ≠ ∞
  proof −
   have integral^N M h ≠ ⊤ (is ?lhs ≠ -)
   proof −
    have ?lhs = (∑ n. (∫⁺ x∈A n. ((1/2)^(Suc n) * (1 / (1 + M (A n)))) ∂M))
     using A by(simp add: h-def nn-integral-suminf)
    also have ... = (∑ n. (1/2)^(Suc n) * (1 / (1 + M (A n))) * M (A n))
     by(rule suminf-cong,rule nn-integral-cmult-indicator) (use A in auto)
    also have ... = (∑ n. (1/2)^(Suc n) * ((1 / (1 + M (A n))) * M (A n)))
     by (simp add: mult.assoc)
    also have ... ≤ (∑ n. (1/2)^(Suc n))
    proof −
     have (1 / (1 + M (A n))) * M (A n) ≤ 1 for n
      using A(3)[of n] by (simp add: add-pos-nonneg divide-le-posI-ennreal
ennreal-divide-times)
     thus ?thesis
      by(intro suminf-le) (metis mult.right-neutral mult-left-mono zero-le,auto)
    qed
    also have ... = (∑ n. ennreal ((1/2)^(Suc n)))
     by(simp only: ennreal-power[of 1/2,symmetric]) (metis divide-ennreal
ennreal-1 ennreal-numeral linorder-not-le not-one-less-zero zero-less-numeral)
    also have ... = ennreal (∑ n. (1/2)^(Suc n))
     by(rule suminf-ennreal2) auto
    also have ... = 1
     using nsum-of-r'[of 1/2 1 1] by auto
    finally show ?thesis
     using nle-le by fastforce
   qed
   thus ?thesis
    by(simp add: emeasure-density)
  qed
 qed
qed

lemma(in sigma-finite-measure) finite-density-measure:
 obtains h where h ∈ borel-measurable M
          ⋀x. x ∈ space M ⟹ 0 < h x
          ⋀x. x ∈ space M ⟹ h x < 1
          finite-measure (density M h)
 by (metis (no-types, lifting) sigma-finite-disjoint ex-finite-density-measure)
```

2.8 Lemma 14.D.13.

lemma (**in** *measure-kernel*)
assumes *disintegration* ν μ
defines $\nu x \equiv$ *marginal-measure* X Y ν
shows *disintegration-absolutely-continuous*: *absolutely-continuous* μ νx
and *disintegration-density*: νx = *density* μ ($\lambda x.$ κ x (*space* Y))
and *disintegration-absolutely-continuous-iff*:
absolutely-continuous νx $\mu \longleftrightarrow$ (*AE* x *in* μ. κ x (*space* Y) > *0*)
proof −
note *sets-eq*[*measurable-cong*] = *disintegration-sets-eq*[*OF assms*(*1*)]
note [*measurable*] = *emeasure-measurable*[*OF sets.top*]
have *νx-eq*: νx A = ($\int^+ x{\in}A$. (κ x (*space* Y)) $\partial\mu$) **if** A:$A \in$ *sets* X **for** A
by(*simp add*: *disintegrationD*[*OF assms*(*1*) A *sets.top*] *emeasure-marginal-measure*[*OF sets-eq*(*1*) A] *νx-def*)
thus *1*:νx = *density* μ ($\lambda x.$ κ x (*space* Y))
by(*auto intro*!: *measure-eqI simp*: *sets-marginal-measure νx-def sets-eq emeasure-density*)
hence *sets-νx*:*sets* νx = *sets* X
using *sets-eq* **by** *simp*
show *absolutely-continuous* μ νx
unfolding *absolutely-continuous-def*
proof *safe*
fix A
assume A: $A \in$ *null-sets* μ
have *0* = ($\int^+ x{\in}A$. (κ x (*space* Y)) $\partial\mu$)
by(*simp add*: A *nn-integral-null-set*)
also have ... = νx A
using A *νx-eq*[*of A*,*simplified sets-eq*(*2*)[*symmetric*]]
by *auto*
finally show $A \in$ *null-sets* νx
using A **by**(*auto simp*: *null-sets-def νx-def sets-marginal-measure sets-eq*)
qed
show *absolutely-continuous* νx $\mu \longleftrightarrow$ (*AE* x *in* μ. κ x (*space* Y) > *0*)
proof
assume *h*:*absolutely-continuous* νx μ
define N **where** N = {$x \in$ *space* μ. (κ x) (*space* Y) = *0*}
have $N \in$ *null-sets* μ
proof −
have νx N = ($\int^+ x{\in}N$. (κ x (*space* Y)) $\partial\mu$)
using *νx-eq*[*of N*] **by**(*simp add*: *N-def sets-eq-imp-space-eq*[*OF sets-eq*(*2*)])
also have ... = ($\int^+ x{\in}N$. *0* $\partial\mu$)
by(*rule nn-integral-cong*) (*auto simp*: *N-def indicator-def*)
also have ... = *0* **by** *simp*
finally have $N \in$ *null-sets* νx
by(*auto simp*: *null-sets-def 1 N-def*)
thus *?thesis*
using *h* **by**(*auto simp*: *absolutely-continuous-def*)
qed
then show *AE* x *in* μ. *0* < (κ x) (*space* Y)

by(*auto intro*!: *AE-I'*[*OF - subset-refl*] *simp*: *N-def*)
next
assume *AE x in μ. 0 < (κ x) (space Y)*
then show *absolutely-continuous νx μ*
using *νx-eq* **by**(*auto simp*: *absolutely-continuous-def intro*!: *null-if-pos-func-has-zero-nn-int*[**where** *f*=λ*x. emeasure (κ x) (space Y)*]) (*auto simp*: *null-sets-def sets-νx*)
qed
qed

2.9 Theorem 14.D.14.

locale *sigma-finite-measure-on-pair-standard* = *sigma-finite-measure-on-pair* + *standard-borel-ne Y*

sublocale *projection-sigma-finite-standard* ⊆ *sigma-finite-measure-on-pair-standard*
by (*simp add*: *sigma-finite-measure-on-pair-axioms sigma-finite-measure-on-pair-standard-def standard-borel-ne-axioms*)

context *sigma-finite-measure-on-pair-standard*
begin

lemma *measure-disintegration-extension*:
∃*μ κ. finite-measure μ* ∧ *measure-kernel X Y κ* ∧ *measure-kernel.disintegration X Y κ ν μ* ∧
(∀ *x*∈*space X. sigma-finite-measure* (*κ x*)) ∧
(∀ *x*∈*space X. κ x* (*space Y*) > *0*) ∧
μ ~$_M$ *νx* (**is** *?goal*)
proof(*rule sigma-finite-measure.sigma-finite-disjoint*[*OF sigma-finite*])
fix *A* :: *nat* ⇒ -
assume *A*:*range A* ⊆ *sets ν* ⋃ (*range A*) = *space ν* ⋀*i. emeasure ν* (*A i*) ≠ ∞ *disjoint-family A*
define *h* **where** *h* ≡ (λ*x.* ∑ *n.* (*1* / *2*) ^ *Suc n* ∗ (*1* / (*1* + *emeasure ν* (*A n*))) ∗ *indicator* (*A n*) *x*)
have *h*: *h* ∈ *borel-measurable ν* ⋀*x y. x* ∈ *space X* ⟹ *y* ∈ *space Y* ⟹ *0* < *h* (*x*,*y*) ⋀*x y. x* ∈ *space X* ⟹ *y* ∈ *space Y* ⟹ *h* (*x*,*y*) < *1 finite-measure* (*density ν h*)
using *ex-finite-density-measure*[*OF A*] **by**(*auto simp*: *sets-eq-imp-space-eq*[*OF nu-sets*] *h-def space-pair-measure*)

interpret *psfs-νx*: *finite-measure marginal-measure X Y* (*density ν h*)
by(*rule finite-measure-marginal-measure-finite*[*OF h*(*4*),*simplified*,*OF nu-sets*])

interpret *psfs*: *projection-sigma-finite-standard X Y density ν h*
by(*auto simp*: *projection-sigma-finite-standard-def projection-sigma-finite-def standard-borel-ne-axioms nu-sets finite-measure.sigma-finite-measure*[*OF finite-measure-marginal-measure-finit h*(*4*),*simplified*,*OF nu-sets*]])
from *psfs.measure-disintegration*
obtain *κ'* **where** *κ'*: *prob-kernel X Y κ' measure-kernel.disintegration X Y κ'* (*density ν h*) *psfs.νx* **by** *auto*

interpret *pk*: *prob-kernel X Y* κ' **by** *fact*
define κ **where** $\kappa \equiv (\lambda x.\ density\ (\kappa'\ x)\ (\lambda y.\ 1\ /\ h\ (x,y)))$
have κB: $\kappa\ x\ B = (\int^+ y{\in}B.\ (1\ /\ h\ (x,\ y))\partial\kappa'\ x)$ **if** $x \in space\ X$ **and** [*measurable*]:$B \in sets\ Y$ **for** $x\ B$
using *nu-sets pk.kernel-sets*[*OF that*(*1*)] *that h*(*1*) **by**(*auto simp*: κ*-def emeasure-density*)
interpret *mk*: *measure-kernel X Y* κ
proof
fix *B*
assume [*measurable*]:$B \in sets\ Y$
have *1*:$(\lambda x.\ \int^+ y{\in}B.\ (1\ /\ h\ (x,\ y))\partial\kappa'\ x) \in borel\text{-}measurable\ X$
using *h*(*1*) *nu-sets* **by**(*auto intro*!: *pk.nn-integral-measurable-f*'[*of* $\lambda z.\ (1\ /\ h\ z) * indicator\ B\ (snd\ z)$,*simplified*])
show $(\lambda x.\ (\kappa\ x)\ B) \in borel\text{-}measurable\ X$
by(*rule measurable-cong*[*THEN iffD1*,*OF - 1*],*simp add*: κ*B*)
qed(*simp-all add*: κ*-def pk.kernel-sets space-ne*)

have *disi*: *mk.disintegration* ν *psfs.*ν*x*
proof(*rule mk.disintegrationI*)
fix *A B*
assume *A*[*measurable*]:$A \in sets\ X$ **and** *B*[*measurable*]:$B \in sets\ Y$
show $\nu\ (A \times B) = (\int^+ x{\in}A.\ (\kappa\ x)\ B\partial psfs.\nu x)$ (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = $(\int^+ z{\in}A \times B.\ 1\ \partial\nu)$
by *auto*
also have ... = $(\int^+ z{\in}A \times B.\ (1\ /\ h\ z * h\ z)\ \partial\nu)$
proof −
have *1*: $a * (1\ /\ a) = 1$ **if** $0 < a\ a < 1$ **for** a :: *ennreal*
proof −
have $a * (1\ /\ a) = ennreal\ (enn2real\ a * 1\ /\ (enn2real\ a))$
by (*simp add*: *divide-eq-1-ennreal enn2real-eq-0-iff ennreal-times-divide*)
also have ... = *ennreal 1*
using *enn2real-eq-0-iff that* **by** *fastforce*
finally show *?thesis*
using *ennreal-1* **by** *simp*
qed
show *?thesis*
by(*rule nn-integral-cong*,*auto simp add*: *sets-eq-imp-space-eq*[*OF nu-sets*] *space-pair-measure ennreal-divide-times indicator-def 1*[*OF h*(*2*,*3*)])
qed
also have ... = $(\int^+ z.\ h\ z * ((1\ /\ h\ z) * indicator\ (A \times B)\ z)\ \partial\nu)$
by(*auto intro*!: *nn-integral-cong simp*: *indicator-def mult.commute*)
also have ... = $(\int^+ z{\in}A \times B.\ (1\ /\ h\ z)\ \partial(density\ \nu\ h))$
using *h*(*1*) **by**(*simp add*: *nn-integral-density*)
also have ... = $(\int^+ x.\ \int^+ y.\ (1\ /\ h\ (x,y) * indicator\ (A \times B)\ (x,y))\ \partial\kappa'\ x\ \partial psfs.\nu x)$
using *h*(*1*) **by**(*simp add*: *pk.nn-integral-fst-finite*'[*OF* κ'(*2*) *psfs-*ν*x.finite-measure-axioms*])
also have ... = $(\int^+ x{\in}A.\ (\int^+ y{\in}B.\ (1\ /\ h\ (x,y))\ \partial\kappa'\ x)\ \partial psfs.\nu x)$
by(*auto intro*!: *nn-integral-cong simp*: *indicator-def*)

```
    also have ... = ?rhs
      by(auto intro!: nn-integral-cong simp: κB[OF - B] space-marginal-measure)
    finally show ?thesis .
  qed
 qed(simp-all add: nu-sets sets-marginal-measure)
 have geq0: 0 < (κ x) (space Y) if x ∈ space X for x
 proof −
  have 0 = (∫⁺ y. 0 ∂κ' x) by simp
  also have ... < (∫⁺ y. (1 / h (x,y)) ∂κ' x)
  proof(rule nn-integral-less)
   show ¬ (AE y in κ' x. 1 / h (x, y) ≤ 0)
   proof
    assume AE y in κ' x. 1 / h (x, y) ≤ 0
    moreover have h (x,y) ≠ ⊤ if y ∈ space (κ' x) for y
        using h(3)[OF ‹x ∈ space X› that[simplified sets-eq-imp-space-eq[OF
pk.kernel-sets[OF ‹x ∈ space X›]]]] top.not-eq-extremum
     by fastforce
    ultimately show False
     using prob-space.AE-False[OF pk.prob-spaces[OF that]] by simp
   qed
  qed(use h(1) pk.kernel-sets[OF that] that in auto)
  also have ... = (κ x) (space Y)
     by(simp add: κB[OF that sets.top]) (simp add: sets-eq-imp-space-eq[OF
pk.kernel-sets[OF that],symmetric])
  finally show ?thesis .
 qed

 show ?goal
 proof(safe intro!: exI[where x=psfs.νx] exI[where x=κ] disi)
  show absolutely-continuous νx psfs.νx
   unfolding mk.disintegration-absolutely-continuous-iff[OF disi]
   by standard (simp add: space-marginal-measure geq0)
 next
  fix x
  assume x:x ∈ space X
  define C where C ≡ range (λn. Pair x −‘ (A n) ∩ space Y)
  have 1:countable C C ⊆ sets Y
     using A(1,2) x by (auto simp: nu-sets sets-eq-imp-space-eq[OF nu-sets]
space-pair-measure C-def)
  have 2: ⋃ C = space Y
   using A(1,2) by(auto simp: sets-eq-imp-space-eq[OF nu-sets] space-pair-measure
C-def) (use x in auto)

  show sigma-finite-measure (κ x)
   unfolding sigma-finite-measure-def
  proof(safe intro!: exI[where x=C])
   fix c
   assume c ∈ C (κ x) c = ∞
  then obtain n where c:c = Pair x −‘ (A n) ∩ space Y by(auto simp: C-def)
```

have $(\kappa\ x)\ c = (\int^+ y{\in}c.\ (1\ /\ h\ (x,\ y))\partial\kappa'\ x)$
using $\kappa B[OF\ x, of\ c]\ 1\ \langle c \in C\rangle$ **by** *auto*
also have $... = (\int^+ y{\in}Pair\ x\ -`\ (A\ n).\ (1\ /\ h\ (x,\ y))\partial\kappa'\ x)$
by(*auto intro*!: *nn-integral-cong simp*: *c indicator-def sets-eq-imp-space-eq*[*OF pk.kernel-sets*[*OF x*]])
also have $... = (\int^+ y{\in}Pair\ x\ -`\ (A\ n).\ (1\ /\ ((1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n))))\ \partial\kappa'\ x)$
proof −
{
fix *y*
assume *xy*:$(x,\ y) \in A\ n$
have $1\ /\ h\ (x,\ y) = 1\ /\ ((1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n))))$
proof −
have $h\ (x,\ y) = (1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))$ (**is** *?lhs = ?rhs*)
proof −
have $?lhs = (\sum m.\ (1\ /\ 2)\ \widehat{}\ Suc\ m * (1\ /\ (1 + emeasure\ \nu\ (A\ m))) * indicator\ (A\ m)\ (x,y))$
by(*simp add*: *h-def*)
also have $... = (\sum m.\ if\ m = n\ then\ (1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))\ else\ 0)$
using *xy A*(*4*) **by**(*fastforce intro*!: *suminf-cong simp*: *disjoint-family-on-def indicator-def*)
also have $... = (\sum j.\ if\ j + Suc\ n = n\ then\ (1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))\ else\ 0) + (\sum j{<}Suc\ n.\ if\ j = n\ then\ (1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))\ else\ 0)$
by(*auto simp*: *suminf-offset*[*of* $\lambda m.\ if\ m = n\ then\ (1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))\ else\ 0\ Suc\ n$] *simp del*: *power-Suc*)
also have $... = ?rhs$
by *simp*
finally show *?thesis* **.**
qed
thus *?thesis* **by** *simp*
qed
}
thus *?thesis*
by(*intro nn-integral-cong*) (*auto simp*: *sets-eq-imp-space-eq*[*OF pk.kernel-sets*[*OF x*]] *indicator-def simp del*: *power-Suc*)
qed
also have $... \leq (\int^+ y.\ (1\ /\ ((1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))))\ \partial\kappa'\ x)$
by(*rule nn-integral-mono*) (*auto simp*: *indicator-def*)
also have $... = (1\ /\ ((1\ /\ 2)\ \widehat{}\ Suc\ n * (1\ /\ (1 + emeasure\ \nu\ (A\ n)))))$
by(*simp add*: *prob-space.emeasure-space-1*[*OF pk.prob-spaces*[*OF x*]])
also have $... < \infty$
by (*metis A*(*3*) *ennreal-add-eq-top ennreal-divide-eq-0-iff ennreal-divide-eq-top-iff ennreal-top-neq-one infinity-ennreal-def mult-eq-0-iff power-eq-0-iff top.not-eq-extremum top-neq-numeral*)
finally show *False*

using ‹(κ x) c = ∞› **by** *simp*
qed(*insert 1 2*, *auto simp*: *mk.kernel-sets*[*OF x*] *sets-eq-imp-space-eq*[*OF mk.kernel-sets*[*OF x*]])
qed(*auto simp*: *psfs-νx.finite-measure-axioms geq0 mk.measure-kernel-axioms mk.disintegration-absolutely-co disi*])
qed

end

lemma(**in** *sigma-finite-measure-on-pair*) *measure-disintegration-extension-AE-unique*:
assumes *sigma-finite-measure μ sigma-finite-measure μ'*
measure-kernel X Y κ measure-kernel X Y κ'
measure-kernel.disintegration X Y κ ν μ measure-kernel.disintegration X Y κ' ν μ'
and *absolutely-continuous μ μ' B ∈ sets Y*
shows *AE x in μ. κ' x B * RN-deriv μ μ' x = κ x B*
proof −
interpret *s1*: *sigma-finite-measure μ* **by** *fact*
interpret *s2*: *sigma-finite-measure μ'* **by** *fact*
interpret *mk1*: *measure-kernel X Y κ* **by** *fact*
interpret *mk2*: *measure-kernel X Y κ'* **by** *fact*
have *sets*[*measurable-cong*]:*sets μ = sets X sets μ' = sets X*
using *assms*(*5*,*6*) **by**(*auto dest*: *mk1.disintegration-sets-eq mk2.disintegration-sets-eq*)
have *1*:*AE x in μ. κ x B = RN-deriv μ* (*marginal-measure-on X Y ν B*) *x*
using *sets mk1.emeasure-measurable*[*OF assms*(*8*)] *mk1.disintegrationD*[*OF assms*(*5*) - *assms*(*8*)]
by(*auto intro*!: *measure-eqI s1.RN-deriv-unique simp*: *emeasure-density emeasure-marginal-measure-on*[*OF nu-sets assms*(*8*)] *sets sets-marginal-measure*)
have *2*:*AE x in μ. κ' x B * RN-deriv μ μ' x = RN-deriv μ* (*marginal-measure-on X Y ν B*) *x*
proof −
{
fix *A*
assume *A*: *A ∈ sets X*
have (∫⁺x∈*A*. ((κ' x) *B * RN-deriv μ μ' x*) ∂μ) = (∫⁺x. *RN-deriv μ μ' x * (κ' x B * indicator A x)*∂μ)
by(*auto intro*!: *nn-integral-cong simp*: *indicator-def mult.commute*)
also have ... = (∫⁺x∈*A*. *κ' x B* ∂μ')
using *mk2.emeasure-measurable*[*OF assms*(*8*)] *sets A*
by(*auto intro*!: *s1.RN-deriv-nn-integral*[*OF assms*(*7*),*symmetric*])
also have ... = *ν* (*A* × *B*)
by(*simp add*: *mk2.disintegrationD*[*OF assms*(*6*) *A assms*(*8*)])
finally have (∫⁺x∈*A*. ((κ' x) *B * RN-deriv μ μ' x*) ∂μ) = *ν* (*A* × *B*) .
}
thus *?thesis*
using *sets mk2.emeasure-measurable*[*OF assms*(*8*)]
by(*auto intro*!: *measure-eqI s1.RN-deriv-unique simp*: *emeasure-density emeasure-marginal-measure-on*[*OF nu-sets assms*(*8*)]*sets sets-marginal-measure*)

```
  qed
  show ?thesis
    using 1 2 by auto
qed

end
```

References

[1] F. Baccelli, B. Blaszczyszyn, and M. Karray. *Random Measures, Point Processes, and Stochastic Geometry*. Inria, Jan. 2020.