

Coproduct Measure

Michikazu Hirata

February 6, 2026

Abstract

This entry formalizes the coproduct measure. Let I be a set and $\{M_i\}_{i \in I}$ measurable spaces. The σ -algebra on $\coprod_{i \in I} M_i = \{(i, x) \mid i \in I \wedge x \in M_i\}$ is defined as the least one making $(\lambda x. (i, x))$ measurable for all $i \in I$. Let μ_i be measures on M_i for all $i \in I$ and A a measurable set of $\coprod_{i \in I} M_i$. The coproduct measure $\coprod_{i \in I} \mu_i$ is defined as follows:

$$\left(\coprod_{i \in I} \mu_i\right)(A) = \sum_{i \in I} \mu_i(A_i), \quad \text{where } A_i = \{x \mid (i, x) \in A\}.$$

We also prove the relationship with coproduct quasi-Borel spaces: the functor $R : \mathbf{Meas} \rightarrow \mathbf{QBS}$ preserves countable coproducts.

Contents

1	Preliminaries	2
1.1	Metrics and Metrizable Spaces	2
1.2	Copy of Extended non-negative reals	3
1.3	Lemmas for Infinite Sum	9
2	Binary Coproduct Measures	15
2.1	The Measurable Space and Measurability	15
2.2	Measures	20
2.3	Finiteness	22
2.4	σ -Finiteness	22
2.5	Non-Negative Integral	22
2.6	Integrability	24
2.7	The Lebesgue Integral	24
3	Coproduct Measures	25
3.1	The Measurable Space and Measurability	25
3.2	Measures	29
3.3	Non-Negative Integral	32
3.4	Integrability	33

3.5	The Lebesgue Integral	33
3.6	Finite Coproduct Measures	35
3.7	Countable Infinite Coproduct Measures	35
3.8	Finiteness	39
3.9	σ -Finiteness	39
4	Additional Properties	40
4.1	s-Finiteness	40
4.2	Standardness	41
4.3	Relationships with Quasi-Borel Spaces	46

1 Preliminaries

```

theory Lemmas-Coproduct-Measure
  imports HOL-Probability.Probability
           Standard-Borel-Spaces.Abstract-Metrizable-Topology
begin

```

1.1 Metrics and Metrizability

```

lemma metrizable-space-metric-space:
  assumes  $d$ :Metric-space UNIV  $d$  Metric-space.mtopology UNIV  $d$  = euclidean
  shows class.metric-space  $d$  ( $\bigcap e \in \{0 < ..\}$ . principal  $\{(x, y). d\ x\ y < e\}$ ) open
proof -
  interpret Metric-space UNIV  $d$  by fact
  show ?thesis
proof
  show open  $U \longleftrightarrow (\forall x \in U. \forall_F (x', y) \text{ in } \bigcap e \in \{0 < ..\}. \text{ principal } \{(F, y). d\ F\ y < e\}. x' = x \longrightarrow y \in U)$  for  $U$ 
  proof(subst eventually-INF-base)
    show  $a \in \{0 < ..\} \implies b \in \{0 < ..\} \implies \exists x \in \{0 < ..\}. \text{ principal } \{(F, y). d\ F\ y < a\} \leq \text{ principal } \{(F, y). d\ F\ y < b\} \square \text{ principal } \{(F, y). d\ F\ y < b\}$  for  $a\ b$ 
    by(auto intro!: be_xI[where  $x = \min\ a\ b$ ])
  next
    show open  $U \longleftrightarrow (\forall x \in U. \exists b \in \{0 < ..\}. \forall_F (x', y) \text{ in } \text{ principal } \{(F, y). d\ F\ y < b\}. x' = x \longrightarrow y \in U)$ 
    by(fastforce simp: open_in_mtopology[simplified  $d(2)$ ,simplified] eventually-principal)
  qed simp
qed(auto simp: triangle')
qed

```

```

corollary metrizable-space-metric-space-ex:
  assumes metrizable-space (euclidean :: 'a :: topological-space topology)
  shows  $\exists (d :: 'a \Rightarrow 'a \Rightarrow \text{real})\ F. \text{ class.metric-space } d\ F\ \text{ open}$ 
proof -
  from assms obtain  $d :: 'a \Rightarrow 'a \Rightarrow \text{real}$  where Metric-space UNIV  $d$  Metric-space.mtopology UNIV  $d$  = euclidean

```

```

    by (metis Metric-space.topspace-mtopology metrizable-space-def topspace-euclidean)
  from metrizable-space-metric-space[OF this] show ?thesis
  by blast
qed

```

```

lemma completely-metrizable-space-metric-space:
  assumes Metric-space (UNIV :: 'a :: topological-space set) d Metric-space.mtopology
  UNIV d = euclidean Metric-space.mcomplete UNIV d
  shows class.complete-space d ( $\prod e \in \{0 < ..\}$ . principal  $\{(x,y). d x y < e\}$ ) open
proof -
  interpret Metric-space UNIV d by fact
  interpret m:metric-space d  $\prod e \in \{0 < ..\}$ . principal  $\{(x,y). d x y < e\}$  open
  by(auto intro!: metrizable-space-metric-space assms)

```

```

  have [simp]:topological-space.convergent (open :: 'a set  $\Rightarrow$  bool) = convergent
proof
  fix x :: nat  $\Rightarrow$  'a
  have *:class.topological-space (open :: 'a set  $\Rightarrow$  bool)
  by standard auto
  show topological-space.convergent open x = convergent x
  by(simp add: topological-space.convergent-def[OF *] topological-space.nhds-def[OF
*] convergent-def nhds-def)
qed
  show ?thesis
  apply unfold-locales
  using assms(3) by(auto simp: mcomplete-def assms(2) MCauchy-def m.Cauchy-def
convergent-def)
qed

```

```

lemma completely-metrizable-space-metric-space-ex:
  assumes completely-metrizable-space (euclidean :: 'a :: topological-space topology)
  shows  $\exists (d :: 'a \Rightarrow 'a \Rightarrow \text{real}) F$ . class.complete-space d F open
proof -
  from assms obtain d :: 'a  $\Rightarrow$  'a  $\Rightarrow$  real where Metric-space UNIV d Met-
ric-space.mtopology UNIV d = euclidean Metric-space.mcomplete UNIV d
  by (metis Metric-space.topspace-mtopology completely-metrizable-space-def topspace-euclidean)
  from completely-metrizable-space-metric-space[OF this] show ?thesis
  by blast
qed

```

1.2 Copy of Extended non-negative reals

In the proof of the change of ordering of the infinite sum (*infsum*) for *ennreal*, we use `infsum_Sigma` and `compact_uniformly_continuous`. Thus, we need to interpret *ennreal* as a metric space. However, there is no standard metric on *ennreal* even though it is a Polish space (thus, a metrizable space). Hence, we do not want to give a metric on *ennreal* globally. Instead of defining a metric on *ennreal*, we define a type copy of *ennreal*, then define a metric on

the copy and prove the change of ordering of the infinite sum. Finally, we transfer the theorems to the ones for *ennreal*.

```

typedef ennreal' = UNIV :: ennreal set
  by simp

lemma bij-Abs-ennreal': bij Abs-ennreal'
  by (metis Abs-ennreal'-cases Abs-ennreal'-inject UNIV-I bij-iff)

lemma inj-Abs-ennreal': inj Abs-ennreal'
  by (simp add: Abs-ennreal'-inject inj-on-def)

setup-lifting type-definition-ennreal'

instantiation ennreal' :: complete-linorder
begin

lift-definition top-ennreal' :: ennreal' is top .
lift-definition bot-ennreal' :: ennreal' is 0 .
lift-definition sup-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  ennreal' is sup .
lift-definition inf-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  ennreal' is inf .
lift-definition Inf-ennreal' :: ennreal' set  $\Rightarrow$  ennreal' is Inf .
lift-definition Sup-ennreal' :: ennreal' set  $\Rightarrow$  ennreal' is sup 0  $\circ$  Sup .

lift-definition less-eq-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  bool is ( $\leq$ ) .
lift-definition less-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  bool is ( $<$ ) .

instance
  by standard
  (transfer, auto simp: Inf-lower Inf-greatest Sup-upper sup.coboundedI2 Sup-least) +

end

instantiation ennreal' :: infinity
begin

definition infinity-ennreal' :: ennreal'
where
  [simp]:  $\infty = (top::ennreal')$ 

instance ..

end

instantiation ennreal' :: {semiring-1-no-zero-divisors, comm-semiring-1}
begin

lift-definition one-ennreal' :: ennreal' is 1 .
lift-definition zero-ennreal' :: ennreal' is 0 .
lift-definition plus-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  ennreal' is (+) .

```

lift-definition *times-ennreal'* :: *ennreal'* \Rightarrow *ennreal'* \Rightarrow *ennreal'* **is** (*) .

instance
by *standard* (*transfer*; *auto simp: field-simps*)+

end

instantiation *ennreal'* :: *minus*
begin

lift-definition *minus-ennreal'* :: *ennreal'* \Rightarrow *ennreal'* \Rightarrow *ennreal'* **is** *minus* .

instance ..

end

instance *ennreal'* :: *numeral* ..

instance *ennreal'* :: *ordered-comm-monoid-add*
by (*standard*, *transfer*) (*use ennreal-add-left-cancel-le in auto*)

lemma *ennreal'-nonneg[simp]*: $\bigwedge r :: \text{ennreal}'. 0 \leq r$
by *transfer simp*

lemma *sum-Rep-ennreal'[simp]*: $(\sum i \in I. \text{Rep-ennreal}' (f i)) = \text{Rep-ennreal}' (\text{sum } f I)$
by (*induction I rule: infinite-finite-induct*) (*auto simp: sum-nonneg zero-ennreal'.rep-eq plus-ennreal'.rep-eq*)

lemma *transfer-sum-ennreal'* [*transfer-rule*]:
rel-fun (*rel-fun* (=) *pcr-ennreal'*) (*rel-fun* (=) *pcr-ennreal'*) *sum sum*
using *rel-funD* **by** (*fastforce simp: comp-def ennreal'.pcr-cr-eq cr-ennreal'-def*)

lemma *pcr-ennreal'-eq:pcr-ennreal'* *a b* $\longleftrightarrow b = \text{Abs-ennreal}' a$
by (*metis Abs-ennreal'-inverse Rep-ennreal'-inverse UNIV-I cr-ennreal'-def ennreal'.pcr-cr-eq*)

lemma *rel-set-pcr-ennreal'-eq:rel-set pcr-ennreal'* *A B* $\longleftrightarrow B = \text{Abs-ennreal}' ` A$
by (*auto simp: rel-set-def pcr-ennreal'-eq*)

lemma *transfer-lessThan-ennreal'* [*transfer-rule*]:
rel-fun pcr-ennreal' (*rel-set pcr-ennreal'*) *lessThan lessThan*

proof –
have [*simp*]: $\bigwedge x xa. xa < \text{Abs-ennreal}' x \implies xa \in \text{Abs-ennreal}' ` \{..<x\}$
by (*metis Abs-ennreal'-cases imageI lessThan-iff less-ennreal'.abs-eq*)
show *?thesis*
by (*fastforce simp: rel-set-pcr-ennreal'-eq pcr-ennreal'-eq less-ennreal'.abs-eq*)

qed

lemma *transfer-greaterThan-ennreal'[transfer-rule]:*
rel-fun pcr-ennreal' (rel-set pcr-ennreal') greaterThan greaterThan
proof –
 have [*simp*]: $\bigwedge x xa. \text{Abs-ennreal}' x < xa \implies xa \in \text{Abs-ennreal}' \{x < ..\}$
 by (*metis Abs-ennreal'-cases greaterThan-iff image-eqI less-ennreal'.abs-eq*)
 show ?*thesis*
 by(*fastforce simp: rel-set-pcr-ennreal'-eq pcr-ennreal'-eq less-ennreal'.abs-eq*)
qed

The transfer rule for *generate-topology*.

lemma *homeomorphism-generating-topology-imp:*
 assumes *bj: bij f*
 and *generate-topology S a*
 shows *generate-topology ((\cdot) f \cdot S) (f \cdot a)*
proof –
 have [*simp*]: $f \cdot \text{UNIV} = \text{UNIV}$
 by (*simp add: assms(1) bij-betw-imp-surj-on*)
 have [*simp*]: $f \cdot (a \cap b) = f \cdot a \cap f \cdot b$ **for** *a b*
 by(*intro image-Int bij-betw-imp-inj-on[OF bj]*)
 have [*simp*]: $(f \cdot \bigcup K) = \bigcup ((\cdot) f \cdot K)$ **for** *K*
 by *blast*
 show ?*thesis*
 using *assms(2)*
proof(*induct rule: generate-topology.induct*)
 case (*Basis s*)
 then show ?*case*
 by(*auto intro!: generate-topology.Basis*)
qed (*auto intro!: generate-topology.Int generate-topology.UNIV generate-topology.UN*)
qed

corollary *homeomorphism-generating-topology-eq:*
 assumes *bjf: bij f*
 shows *generate-topology S a = generate-topology ((\cdot) f \cdot S) (f \cdot a)*
proof –
 define *g* where *g \equiv the-inv f*
 have *bjg: bij g*
 using *assms bij-betw-the-inv-into g-def* **by** *blast*
 have *gf: g (f x) = x* **for** *x*
 by (*metis assms bij-betw-imp-inj-on g-def the-inv-f-f*)
 show ?*thesis*
proof
 assume *generate-topology ((\cdot) f \cdot S) (f \cdot a)*
 then have *generate-topology ((\cdot) g \cdot ((\cdot) f \cdot S)) (g \cdot f \cdot a)*
 by(*auto intro!: homeomorphism-generating-topology-imp[OF bjg]*)
 moreover have $((\cdot) g \cdot ((\cdot) f \cdot S)) = S g \cdot f \cdot a = a$
 using *gf* **by**(*auto simp: image-comp*)
 ultimately show *generate-topology S a*
 by *argo*
qed(*auto intro!: bjf homeomorphism-generating-topology-imp*)

qed

lemma *transfer-generate-topology-ennreal'*[*transfer-rule*]:

rel-fun (rel-set (rel-set pcr-ennreal')) (rel-fun (rel-set pcr-ennreal') (=)) generate-topology generate-topology

proof(*intro rel-funI*)

fix *S S' a b*

assume *h:rel-set (rel-set pcr-ennreal') S S' rel-set pcr-ennreal' a b*

then have [*simp*]:*S' = (·) Abs-ennreal' ' S*

by(*auto simp: rel-set-def[of rel-set pcr-ennreal'] rel-set-pcr-ennreal'-eq*)

show *generate-topology S a = generate-topology S' b*

using *h(2) by(auto simp: rel-set-pcr-ennreal'-eq homeomorphism-generating-topology-eq[OF bij-Abs-ennreal'])*

qed

instantiation *ennreal' :: topological-space*

begin

lift-definition *open-ennreal' :: ennreal' set \Rightarrow bool is open .*

instance

by *standard (transfer, auto)+*

end

instance *ennreal' :: second-countable-topology*

proof

obtain *B :: ennreal set set where B:*

countable B open = generate-topology B

using *ex-countable-subbasis by blast*

have *open = generate-topology ((·) Abs-ennreal' ' B)*

using *B(2) by transfer auto*

with *B(1) show $\exists B':: ennreal' set set. countable B' \wedge open = generate-topology B'$*

by(*auto intro!: exI[where x=($\lambda b. Abs-ennreal' ' b$) ' B]*)

qed

instance *ennreal' :: linorder-topology*

by (*standard, transfer*) (*use open-ennreal-def in auto*)

lemma *continuous-map-Abs-ennreal':continuous-on UNIV Abs-ennreal'*

by (*metis continuous-on-open-vimage image-vimage-eq open-Int open-UNIV open-ennreal'.abs-eq*)

lemma *continuous-map-Rep-ennreal':continuous-on UNIV Rep-ennreal'*

by (*metis continuous-on-open-vimage image-vimage-eq open-Int open-UNIV open-ennreal'.rep-eq*)

corollary *continuous-map-ennreal'-eq: continuous-on A f \longleftrightarrow continuous-on A ($\lambda x. Rep-ennreal' (f x)$)*

proof

```

have ( $\lambda x. \text{Abs-ennreal}' (\text{Rep-ennreal}' (f x))$ ) =  $f$ 
  by (simp add: Rep-ennreal'-inverse)
thus continuous-on A f if h:continuous-on A ( $\lambda x. \text{Rep-ennreal}' (f x)$ )
  using continuous-on-compose[OF h continuous-on-subset[OF continuous-map-Abs-ennreal']]
  by(simp add: comp-def)
qed(simp add: continuous-on-compose[OF - continuous-on-subset[OF continuous-map-Rep-ennreal']],simplified
comp-def)

```

```

lemma ennreal-ennreal'-homeomorphic:
  (euclidean :: ennreal topology) homeomorphic-space (euclidean :: ennreal' topology)
  by(auto simp: homeomorphic-space-def homeomorphic-maps-def continuous-map-Abs-ennreal'
continuous-map-Rep-ennreal' Abs-ennreal'-inverse Rep-ennreal'-inverse
intro!: exI[where x=Rep-ennreal'] exI[where x=Abs-ennreal'])

```

```

corollary Polish-space-ennreal': Polish-space (euclidean :: ennreal' topology)
  using Polish-space-ennreal ennreal-ennreal'-homeomorphic homeomorphic-Polish-space-aux
by blast

```

```

instantiation ennreal' :: metric-space
begin

```

```

definition dist-ennreal' :: ennreal'  $\Rightarrow$  ennreal'  $\Rightarrow$  real
  where dist-ennreal'  $\equiv$  SOME d. Metric-space UNIV d  $\wedge$ 
Metric-space.mtopology UNIV d = euclidean  $\wedge$ 
Metric-space.mcomplete UNIV d

```

```

definition uniformity-ennreal' :: (ennreal'  $\times$  ennreal') filter
  where uniformity-ennreal'  $\equiv$   $\prod$  e $\in$ {0<..}. principal {(x,y). dist x y < e}

```

```

instance

```

```

proof –

```

```

  let ?open = open :: ennreal' set  $\Rightarrow$  bool

```

```

  interpret c:complete-space dist uniformity ?open

```

```

  proof –

```

```

    have  $\exists d. \text{Metric-space} (\text{UNIV} :: \text{ennreal}' \text{ set}) d \wedge$ 
      Metric-space.mtopology UNIV d = euclidean  $\wedge$ 
      Metric-space.mcomplete UNIV d

```

```

  by (metis Polish-space-ennreal' Metric-space.topspace-mtopology Polish-space-def
completely-metrizable-space-def topspace-euclidean)

```

```

  hence Metric-space (UNIV :: ennreal' set) dist  $\wedge$ 
Metric-space.mtopology (UNIV :: ennreal' set) dist = euclidean  $\wedge$ 
Metric-space.mcomplete (UNIV :: ennreal' set) dist

```

```

  unfolding dist-ennreal'-def by(rule someI-ex)

```

```

  with completely-metrizable-space-metric-space show class.complete-space dist
uniformity ?open

```

```

  by(fastforce simp: uniformity-ennreal'-def)

```

```

qed

```

```

  have [simp]:topological-space.convergent ?open = convergent

```

```

proof

```

```

fix x :: nat ⇒ ennreal'
have *:class.topological-space ?open
  by standard auto
show topological-space.convergent open x = convergent x
by(simp add: topological-space.convergent-def[OF *] topological-space.nhds-def[OF
*] convergent-def nhds-def)
qed
show OFCLASS(ennreal', metric-space-class)
  by standard (use uniformity-ennreal'-def c.open-uniformity c.dist-triangle2
c.Cauchy-convergent in auto)
qed

end

```

1.3 Lemmas for Infinite Sum

```

lemma transfer-nhds-ennreal'[transfer-rule]: rel-fun pcr-ennreal' (rel-filter pcr-ennreal')
nhds nhds
proof(rule rel-funI)
  fix x x'
  assume h:pcr-ennreal' x x'
  then have b:nhds (x, x') ⊓ principal {(y, y'). pcr-ennreal' y y'} ≠ ⊥ (is ?F ≠
-)
  by(auto simp: eventually-False[symmetric] eventually-inf-principal dest: even-
tually-nhds-x-imp-x)
  have ev-eq1:(∀F xx' in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (fst
xx'))
    ⇔ eventually P (nhds x) for P
  proof
    assume ∀F xx' in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (fst xx')
    then obtain S where
      S:open S (x, x') ∈ S ∧ xx'. xx' ∈ S ⇒ pcr-ennreal' (fst xx') (snd xx') ⇒ P
(fst xx')
    unfolding eventually-nhds by blast
    then obtain A B where AB: open A open (Abs-ennreal' ' B) (x,x') ∈ A ×
Abs-ennreal' ' B A × Abs-ennreal' ' B ⊆ S
    by (metis ennreal'.type-definition-ennreal' open-prod-elim surj-image-vimage-eq
type-definition.univ)
    have AB1:open (A ∩ B) x ∈ A ∩ B
    using AB h by(auto simp: open-ennreal'.abs-eq pcr-ennreal'-eq dest: injD[OF
inj-Abs-ennreal'])
    have AB2:(y, Abs-ennreal' y) ∈ S pcr-ennreal' (fst (y, Abs-ennreal' y)) (snd
(y, Abs-ennreal' y))
    if y:y ∈ A y ∈ B for y
    using AB y by(auto simp: pcr-ennreal'-eq)
    show eventually P (nhds x)
    using S(3)[OF AB2] AB1 by(auto intro!: exI[where x=A ∩ B] simp:
eventually-nhds)
  next

```

assume *eventually P (nhds x)*
then obtain *S* **where** *open S x ∈ S ∧ y. y ∈ S ⇒ P y*
by(*auto simp: eventually-nhds*)
thus $\forall_F xx'$ *in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (fst xx')*
unfolding *eventually-nhds* **by**(*auto intro!: exI[where x=S × UNIV] simp:*
open-Times)
qed
have *ev-eq2:($\forall_F xx'$ in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (snd xx'))*
 \longleftrightarrow *eventually P (nhds x') for P*
proof
assume $\forall_F xx'$ *in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (snd xx')*
then obtain *S* **where**
 $S:open S (x, x') ∈ S ∧ xx'. xx' ∈ S ⇒ pcr-ennreal' (fst xx') (snd xx') ⇒ P$
 $(snd xx')$
unfolding *eventually-nhds* **by** *blast*
then obtain *A B* **where** *AB: open A open (Abs-ennreal' ' B) (x,x') ∈ A ×*
Abs-ennreal' ' B A × Abs-ennreal' ' B ⊆ S
by (*metis ennreal'.type-definition-ennreal' open-prod-elim surj-image-vimage-eq*
type-definition.univ)
have *AB1:open (A ∩ B) x ∈ A ∩ B*
using *AB h* **by**(*auto simp: open-ennreal'.abs-eq pcr-ennreal'-eq dest: injD[OF*
inj-Abs-ennreal'])
have *AB2: (y, Abs-ennreal' y) ∈ S pcr-ennreal' (fst (y, Abs-ennreal' y)) (snd*
(y, Abs-ennreal' y))
if *y:y ∈ A y ∈ B* **for** *y*
using *AB y* **by**(*auto simp: pcr-ennreal'-eq*)
show *eventually P (nhds x')*
using *S(3)[OF AB2] AB1 h*
by(*auto intro!: exI[where x=Abs-ennreal' '(A ∩ B)] simp: eventually-nhds*
pcr-ennreal'-eq open-ennreal'.abs-eq)
next
assume *eventually P (nhds x')*
then obtain *S* **where** *open S x' ∈ S ∧ y. y ∈ S ⇒ P y*
by(*auto simp: eventually-nhds*)
thus $\forall_F xx'$ *in nhds (x, x'). pcr-ennreal' (fst xx') (snd xx') → P (snd xx')*
unfolding *eventually-nhds* **by**(*auto intro!: exI[where x=UNIV × S] simp:*
open-Times)
qed
show *rel-filter pcr-ennreal' (nhds x) (nhds x')*
proof(*rule rel-filter.intros*)
show $\forall_F (x, y)$ *in nhds (x, x') □ principal {(y, y'). pcr-ennreal' y y'}*.
pcr-ennreal' x y
unfolding *eventually-inf-principal* **using** *h* **by**(*auto intro!: eventuallyI simp:*
pcr-ennreal'-eq)
qed (*auto intro!: filter-eqI simp: eventually-inf-principal eventually-map-filter-on*
split-beta' ev-eq1 ev-eq2)
qed

lemmas *transfer-filtermap-ennreal'*[*transfer-rule*] = *filtermap-parametric*[**where** *A=HOL.eq*
and *B=pcr-ennreal'*]

lemma *transfer-filterlim-ennreal'*[*transfer-rule*]:
rel-fun (rel-fun (=) pcr-ennreal') (*rel-fun (rel-filter pcr-ennreal')* (*rel-fun (rel-filter*
(=)) (=))) *filterlim filterlim*
unfolding *filterlim-def* **by** *transfer-prover*

lemma *transfer-The-ennreal*:
assumes *P:∃!x. P x*
and *rel-fun pcr-ennreal' (=) P P'*
shows *The P' = Abs-ennreal' (The P)*
proof –
have *P':∃!x'. P' x'*
by (*metis Rep-ennreal'-inverse pcr-ennreal'-eq rel-funD[OF assms(2)] P*)
show *?thesis*
proof(*rule the1I2*)
fix *x*
assume *h:P x*
show (*THE a. P' a*) = *Abs-ennreal' x*
by(*rule the1I2[OF P']*) (*metis (full-types) h P' assms(2) ennreal'.id-abs-transfer*
rel-funD)
qed fact
qed

lemma *transfer-infsum-ennreal'*[*transfer-rule*]:
rel-fun (rel-fun (=) pcr-ennreal') (*rel-fun (=) pcr-ennreal'*) *infsum (infsum :: ('a*
 \Rightarrow $-) \Rightarrow - \Rightarrow -)$
proof –
have **:rel-fun pcr-ennreal' (=) (λl. (sum x \longrightarrow l) (finite-subsets-at-top A))*
(λl. (sum y \longrightarrow l) (finite-subsets-at-top A))
if [*transfer-rule*]: *rel-fun (=) pcr-ennreal' x y* **for** *x :: 'a \Rightarrow ennreal* **and** *y* **and**
A
by *transfer-prover*
show *?thesis*
apply(*simp add: nonneg-summable-on-complete infsum-def[abs-def]*)
apply(*intro rel-funI*)
apply(*simp add: pcr-ennreal'-eq Topological-Spaces.Lim-def*)
apply(*intro transfer-The-ennreal*)
apply (*meson has-sum-def has-sum-unique nonneg-has-sum-complete zero-le*)
using *** **by** *auto*
qed

lemma *inf-sum-Rep-Abs-ennreal'*:*infsum f A = Rep-ennreal' (infsum ((λx. Abs-ennreal'*
(f x))) A)
by *transfer (simp add: comp-def)*

lemma *continuous-on-add-ennreal'*:
fixes *f g :: 'a::topological-space \Rightarrow ennreal'*

shows *continuous-on A f* \implies *continuous-on A g* \implies *continuous-on A* ($\lambda x. f x + g x$)
unfolding *continuous-map-ennreal'-eq plus-ennreal'.rep-eq*
by(*rule continuous-on-add-ennreal*)

lemma *uniformly-continuous-add-ennreal'*: *isUCont* ($\lambda(x::ennreal', y). x + y$)

proof(*safe intro!*: *compact-uniformly-continuous*)

have *compact* ($UNIV \times UNIV :: (ennreal' \times ennreal')$ *set*)

by(*intro compact-Times compact-UNIV*)

thus *compact* ($UNIV :: (ennreal' \times ennreal')$ *set*)

by *simp*

qed(*auto intro!*: *continuous-on-add-ennreal'* *continuous-on-fst* *continuous-on-snd*
simp: split-beta')

lemma *infsun-eq-suminf*:

assumes *f summable-on UNIV*

shows ($\sum_{n \in UNIV} f n$) = *suminf f*

using *has-sum-imp-sums*[*OF has-sum-infsun*[*OF assms*]]

by (*simp add: sums-iff*)

lemma *infsun-Sigma-ennreal'*:

fixes *f* :: $- \Rightarrow ennreal'$

shows *infsun f* (*Sigma A B*) = *infsun* ($\lambda x. infsun (\lambda y. f (x, y)) (B x)$) *A*

by(*auto intro!*: *uniformly-continuous-add-ennreal'* *infsun-Sigma nonneg-summable-on-complete*)

lemma *infsun-swap-ennreal'*:

fixes *f* :: $- \Rightarrow - \Rightarrow ennreal'$

shows *infsun* ($\lambda x. infsun (\lambda y. f x y) B$) *A* = *infsun* ($\lambda y. infsun (\lambda x. f x y) A$)

B

by(*auto intro!*: *infsun-swap uniformly-continuous-add-ennreal'* *nonneg-summable-on-complete*)

lemma *infsun-Sigma-ennreal*:

fixes *f* :: $- \Rightarrow ennreal$

shows *infsun f* (*Sigma A B*) = *infsun* ($\lambda x. infsun (\lambda y. f (x, y)) (B x)$) *A*

by (*simp add: inf-sum-Rep-Abs-ennreal'* *infsun-Sigma-ennreal'* *Rep-ennreal'-inverse*)

lemma *infsun-swap-ennreal*:

fixes *f* :: $- \Rightarrow - \Rightarrow ennreal$

shows *infsun* ($\lambda x. infsun (\lambda y. f x y) B$) *A* = *infsun* ($\lambda y. infsun (\lambda x. f x y) A$)

B

by (*simp add: inf-sum-Rep-Abs-ennreal'* *Rep-ennreal'-inverse infsun-swap-ennreal'*[**where**
A=A])

lemma *has-sum-cmult-right-ennreal*:

fixes *f* :: $- \Rightarrow ennreal$

assumes $c < \top$ (*f has-sum a*) *A*

shows ($\lambda x. c * f x$) *has-sum* $c * a$) *A*

using *ennreal-tendsto-cmult*[*OF assms*(1)] *assms*(2)

by (*force simp add: has-sum-def sum-distrib-left*)

lemma *infsum-cmult-right-ennreal*:
fixes $f :: - \Rightarrow \text{ennreal}$
assumes $c < \top$
shows $(\sum_{\infty} x \in A. c * f x) = c * \text{infsum } f A$
by (*simp add: assms has-sum-cmult-right-ennreal infsumI nonneg-summable-on-complete*)

lemma *ennreal-sum-SUP-eq*:
fixes $f :: \text{nat} \Rightarrow - \Rightarrow \text{ennreal}$
assumes $\text{finite } A \wedge x. x \in A \implies \text{incseq } (\lambda j. f j x)$
shows $(\sum_{i \in A. \bigsqcup n. f n i}) = (\bigsqcup n. \sum_{i \in A. f n i})$
using *assms*
proof *induction*
case *empty*
then show *?case*
by *simp*
next
case *ih:(insert x F)*
show *?case (is ?lhs = ?rhs)*
proof –
have $?lhs = (\bigsqcup n. f n x) + (\bigsqcup n. \text{sum } (f n) F)$
using *ih by simp*
also have $\dots = (\bigsqcup n. f n x + \text{sum } (f n) F)$
using *ih by (auto intro!: incseq-sumI2 ennreal-SUP-add[symmetric])*
also have $\dots = ?rhs$
using *ih by simp*
finally show *?thesis .*
qed
qed

lemma *ennreal-infsum-Sup-eq*:
fixes $f :: \text{nat} \Rightarrow - \Rightarrow \text{ennreal}$
assumes $\wedge x. x \in A \implies \text{incseq } (\lambda j. f j x)$
shows $(\sum_{\infty} x \in A. (\text{SUP } j. f j x)) = (\text{SUP } j. (\sum_{\infty} x \in A. f j x))$ (**is** *?lhs = ?rhs*)
proof –
have $?lhs = (\bigsqcup (\text{sum } (\lambda x. \bigsqcup j. f j x) \text{ ' } \{F. \text{finite } F \wedge F \subseteq A\}))$
by (*auto intro!: nonneg-infsum-complete simp: SUP-upper2 assms*)
also have $\dots = (\bigsqcup A \in \{F. \text{finite } F \wedge F \subseteq A\}. \bigsqcup j. \text{sum } (f j) A)$
using *assms by (auto intro!: SUP-cong ennreal-sum-SUP-eq)*
also have $\dots = (\bigsqcup j. \bigsqcup A \in \{F. \text{finite } F \wedge F \subseteq A\}. \text{sum } (f j) A)$
using *SUP-commute by fast*
also have $\dots = ?rhs$
by (*subst nonneg-infsum-complete*) (*use assms in auto*)
finally show *?thesis .*
qed

lemma *bounded-infsum-summable*:
assumes $\wedge x. x \in A \implies f x \geq 0$ $(\sum_{\infty} x \in A. \text{ennreal } (f x)) < \text{top}$
shows *f summable-on A*

proof(*rule nonneg-bdd-above-summable-on*)
from *assms(2)* **obtain** K **where** $K: (\sum_{\infty x \in A. \text{ennreal } (f x)} \leq \text{ennreal } K \ K \geq 0$
using *less-top-ennreal* **by** *force*
show *bdd-above* ($\text{sum } f \text{ ' } \{F. F \subseteq A \wedge \text{finite } F\}$)
proof(*safe intro!*: *bdd-aboveI*[**where** $M=K$])
fix A'
assume $A': A' \subseteq A \ \text{finite } A'$
have $(\sum_{\infty x \in A. \text{ennreal } (f x)} = (\bigsqcup (\text{sum } (\lambda x. \text{ennreal } (f x)) \text{ ' } \{F. \text{finite } F \wedge F \subseteq A\}))$
by (*simp add: nonneg-infsum-complete*)
also have $\dots = (\bigsqcup ((\lambda F. \text{ennreal } (\text{sum } f F)) \text{ ' } \{F. \text{finite } F \wedge F \subseteq A\}))$
by(*auto intro!*: *SUP-cong sum-ennreal assms*)
finally have $(\bigsqcup ((\lambda F. \text{ennreal } (\text{sum } f F)) \text{ ' } \{F. \text{finite } F \wedge F \subseteq A\})) \leq \text{ennreal } K$
using K **by** *order*
hence $\text{ennreal } (\text{sum } f A') \leq \text{ennreal } K$
by (*simp add: A' SUP-le-iff*)
thus $\text{sum } f A' \leq K$
by (*simp add: K(2)*)
qed
qed fact

lemma *infsum-less-top-dest*:

fixes $f :: - \Rightarrow - :: \{\text{ordered-comm-monoid-add, topological-comm-monoid-add, t2-space, complete-linorder, linorder-topology}\}$
assumes $(\sum_{\infty x \in A. f x) < \text{top} \wedge x. x \in A \implies f x \geq 0 \ x \in A$
shows $f x < \text{top}$
proof(*rule ccontr*)
assume $f: \neg f x < \text{top}$
have $(\sum_{\infty x \in A. f x) = (\sum_{\infty y \in A - \{x\} \cup \{x\}. f y)$
by(*rule arg-cong*[**where** $f=\text{infsum } _$]) (*use assms in auto*)
also have $\dots = (\sum_{\infty y \in A - \{x\}. f y) + (\sum_{\infty y \in \{x\}. f y)$
using *assms(2)* **by**(*intro infsum-Un-disjoint*) (*auto intro!*: *nonneg-summable-on-complete*)
also have $\dots = (\sum_{\infty y \in A - \{x\}. f y) + \text{top}$
using $f \text{ top.not-eq-extremum}$ **by** *fastforce*
also have $\dots = \text{top}$
by(*auto intro!*: *add-top infsum-nonneg assms*)
finally show *False*
using *assms(1)* **by** *simp*
qed

lemma *infsum-ennreal-eq*:

assumes $f \text{ summable-on } A \wedge x. x \in A \implies f x \geq 0$
shows $(\sum_{\infty x \in A. \text{ennreal } (f x)} = \text{ennreal } (\sum_{\infty x \in A. f x)$
proof –
have $(\sum_{\infty x \in A. \text{ennreal } (f x)} = (\bigsqcup (\text{sum } (\lambda x. \text{ennreal } (f x)) \text{ ' } \{F. \text{finite } F \wedge F \subseteq A\}))$
by (*simp add: nonneg-infsum-complete*)

also have ... = (\bigsqcup ((λF . *ennreal* (*sum* *f* *F*)) ‘ {*F*. *finite* *F* \wedge *F* \subseteq *A*}))
by(*auto intro!*: *SUP-cong sum-ennreal assms*)
also have ... = *ennreal* ($\sum_{\infty} x \in A$. *f* *x*)
using *infsum-nonneg-is-SUPREMUM-ennreal[OF assms]* **by** *simp*
finally show ?*thesis* .
qed

lemma *abs-summable-on-integrable-iff*:
fixes *f* :: - \Rightarrow - :: {*banach*, *second-countable-topology*}
shows *Infinite-Sum.abs-summable-on* *f* *A* \longleftrightarrow *integrable* (*count-space* *A*) *f*
by (*simp add: abs-summable-equivalent abs-summable-on-def*)

lemma *infsum-eq-integral*:
fixes *f* :: - \Rightarrow - :: {*banach*, *second-countable-topology*}
assumes *Infinite-Sum.abs-summable-on* *f* *A*
shows *infsum* *f* *A* = *integral^L* (*count-space* *A*) *f*
using *assms infssetsum-infsum[of f A,symmetric]*
by(*auto simp: abs-summable-on-integrable-iff abs-summable-on-def infssetsum-def*)

end

theory *Coproduct-Measure*
imports *Lemmas-Coproduct-Measure*
HOL-Analysis.Analysis
begin

2 Binary Coproduct Measures

definition *copair-measure* :: [*'a measure*, *'b measure*] \Rightarrow (*'a* + *'b*) *measure* (**infixr** $\langle \bigoplus_M \rangle$ 65) **where**
 $M \bigoplus_M N = \text{measure-of } (\text{space } M \langle + \rangle \text{ space } N)$
 $(\{\text{Inl } \cdot A \mid A. A \in \text{sets } M\} \cup \{\text{Inr } \cdot A \mid A. A \in \text{sets } N\})$
 $(\lambda A. \text{emeasure } M (\text{Inl } \cdot A) + \text{emeasure } N (\text{Inr } \cdot A))$

2.1 The Measurable Space and Measurability

lemma
shows *space-copair-measure*: *space* (*copair-measure* *M* *N*) = *space* *M* $\langle + \rangle$ *space* *N*
and *sets-copair-measure-sigma*:
 $\text{sets } (\text{copair-measure } M \ N)$
 $= \text{sigma-sets } (\text{space } M \langle + \rangle \text{ space } N) (\{\text{Inl } \cdot A \mid A. A \in \text{sets } M\} \cup \{\text{Inr } \cdot A \mid A. A \in \text{sets } N\})$
and *Inl-measurable[measurable]*: *Inl* \in *M* \rightarrow_M $M \bigoplus_M N$
and *Inr-measurable[measurable]*: *Inr* \in *N* \rightarrow_M $M \bigoplus_M N$
proof –
have 1: $(\{\text{Inl } \cdot A \mid A. A \in \text{sets } M\} \cup \{\text{Inr } \cdot A \mid A. A \in \text{sets } N\}) \subseteq \text{Pow } (\text{space } M \langle + \rangle \text{ space } N)$

using *sets.sets-into-space*[*of - M*] *sets.sets-into-space*[*of - N*] **by** *fastforce*
show *space (copair-measure M N) = space M <+> space N*
and *2:sets (copair-measure M N)*
 $= \text{sigma-sets (space M <+> space N) (\{Inl \text{ ` } A \mid A. A \in \text{sets } M\} \cup \{Inr \text{ ` } A \mid A. A \in \text{sets } N\})}$
by(*simp-all add: copair-measure-def sets-measure-of[OF 1] space-measure-of[OF 1]*)
show $Inl \in M \rightarrow_M M \oplus_M N \quad Inr \in N \rightarrow_M M \oplus_M N$
by(*auto intro!: measurable-sigma-sets[OF 2 1] simp: vimage-def image-def*)
qed

lemma *sets-copair-measure-cong*:
 $\text{sets } M1 = \text{sets } M2 \implies \text{sets } N1 = \text{sets } N2 \implies \text{sets } (M1 \oplus_M N1) = \text{sets } (M2 \oplus_M N2)$
by(*simp cong: sets-eq-imp-space-eq add: sets-copair-measure-sigma*)

lemma *measurable-image-Inl*[*measurable*]: $A \in \text{sets } M \implies Inl \text{ ` } A \in \text{sets } (M \oplus_M N)$
using *sets-copair-measure-sigma* **by** *fastforce*

lemma *measurable-image-Inr*[*measurable*]: $A \in \text{sets } N \implies Inr \text{ ` } A \in \text{sets } (M \oplus_M N)$
using *sets-copair-measure-sigma* **by** *fastforce*

lemma *measurable-vimage-Inl*:
assumes [*measurable*]: $A \in \text{sets } (M \oplus_M N)$
shows $Inl \text{ - ` } A \in \text{sets } M$
proof –
have $Inl \text{ - ` } A = Inl \text{ - ` } A \cap \text{space } M$
using *sets.sets-into-space[OF assms]*
by(*auto simp add: space-copair-measure*)
also have $\dots \in \text{sets } M$
by *simp*
finally show *?thesis* .
qed

lemma *measurable-vimage-Inr*:
assumes [*measurable*]: $A \in \text{sets } (M \oplus_M N)$
shows $Inr \text{ - ` } A \in \text{sets } N$
proof –
have $Inr \text{ - ` } A = Inr \text{ - ` } A \cap \text{space } N$
using *sets.sets-into-space[OF assms]*
by(*auto simp add: space-copair-measure*)
also have $\dots \in \text{sets } N$
by *simp*
finally show *?thesis* .
qed

lemma *in-sets-copair-measure-iff*:

$A \in \text{sets } (\text{copair-measure } M N) \iff \text{Inl } -' A \in \text{sets } M \wedge \text{Inr } -' A \in \text{sets } N$
proof *safe*
assume [*measurable*]: $\text{Inl } -' A \in \text{sets } M \text{ Inr } -' A \in \text{sets } N$
have $A = ((\text{Inl } -' \text{Inl } -' A) \cup (\text{Inr } -' \text{Inr } -' A))$
by (*simp add: vimage-def image-def*) (*safe, metis obj-sumE*)
also have $\dots \in \text{sets } (\text{copair-measure } M N)$
by *measurable*
finally show $A \in \text{sets } (\text{copair-measure } M N)$.
qed (*use measurable-vimage-Inl measurable-vimage-Inr in auto*)

lemma *measurable-copair-Inl-Inr*:

assumes [*measurable*]: $(\lambda x. f (\text{Inl } x)) \in M \rightarrow_M L \ (\lambda x. f (\text{Inr } x)) \in N \rightarrow_M L$
shows $f \in M \oplus_M N \rightarrow_M L$
proof (*rule measurableI*)
fix A
assume [*measurable*]: $A \in \text{sets } L$
have $f -' A = \text{Inl } -' ((\lambda x. f (\text{Inl } x)) -' A) \cup \text{Inr } -' ((\lambda x. f (\text{Inr } x)) -' A)$
by (*simp add: image-def vimage-def*) (*safe, metis obj-sumE*)
hence $f -' A \cap \text{space } (M \oplus_M N)$
 $= \text{Inl } -' ((\lambda x. f (\text{Inl } x)) -' A \cap \text{space } M) \cup \text{Inr } -' ((\lambda x. f (\text{Inr } x)) -' A \cap \text{space } N)$
by (*auto simp: space-copair-measure*)
also have $\dots \in \text{sets } (M \oplus_M N)$
by *measurable*
finally show $f -' A \cap \text{space } (M \oplus_M N) \in \text{sets } (M \oplus_M N)$.
next
show $\bigwedge x. x \in \text{space } (M \oplus_M N) \implies f x \in \text{space } L$
using *measurable-space[OF assms(1)] measurable-space[OF assms(2)]*
by (*auto simp add: space-copair-measure*)
qed

corollary *measurable-copair-measure-iff*:

$f \in M \oplus_M N \rightarrow_M L \iff (\lambda x. f (\text{Inl } x)) \in M \rightarrow_M L \wedge (\lambda x. f (\text{Inr } x)) \in N \rightarrow_M L$
by (*auto simp add: measurable-copair-Inl-Inr*)

lemma *measurable-copair-dest1*:

assumes [*measurable*]: $f \in L \rightarrow_M M \oplus_M N$ **and** $f -' (\text{Inl } -' \text{space } M) \cap \text{space } L = \text{space } L$
obtains f' **where** $f' \in L \rightarrow_M M \ \bigwedge x. x \in \text{space } L \implies f x = \text{Inl } (f' x)$
proof –
define f' **where** $f' \equiv (\lambda x. \text{SOME } y. f x = \text{Inl } y)$
have $f': \bigwedge x. x \in \text{space } L \implies f x = \text{Inl } (f' x)$
unfolding f' -*def* **by** (*rule someI-ex*) (*use assms(2) in blast*)
moreover have $f' \in L \rightarrow_M M$
proof (*rule measurableI*)
show $\bigwedge x. x \in \text{space } L \implies f' x \in \text{space } M$
using f' *measurable-space[OF assms(1)]*
by (*auto simp: space-copair-measure*)

```

next
  fix A
  assume A[measurable]:A ∈ sets M
  have [simp]:f' -' A ∩ space L = f -' (Inl ' A) ∩ space L
    using f' sets.sets-into-space[OF A] by auto
  show f' -' A ∩ space L ∈ sets L
    by auto
qed
ultimately show ?thesis
  using that by blast
qed

lemma measurable-copair-dest2:
  assumes [measurable]:f ∈ L →M M ⊕M N and f -' (Inr ' space N) ∩ space
L = space L
  obtains f' where f' ∈ L →M N ∧x. x ∈ space L ⇒ f x = Inr (f' x)
proof -
  define f' where f' ≡ (λx. SOME y. f x = Inr y)
  have f':∧x. x ∈ space L ⇒ f x = Inr (f' x)
    unfolding f'-def by(rule someI-ex) (use assms(2) in blast)
  moreover have f' ∈ L →M N
  proof(rule measurableI)
    show ∧x. x ∈ space L ⇒ f' x ∈ space N
      using f' measurable-space[OF assms(1)]
      by(auto simp: space-copair-measure)
  next
  fix A
  assume A[measurable]:A ∈ sets N
  have [simp]:f' -' A ∩ space L = f -' (Inr ' A) ∩ space L
    using f' sets.sets-into-space[OF A] by auto
  show f' -' A ∩ space L ∈ sets L
    by auto
qed
ultimately show ?thesis
  using that by blast
qed

lemma measurable-copair-dest3:
  assumes [measurable]:f ∈ L →M M ⊕M N
  and f -' (Inl ' space M) ∩ space L ⊂ space L f -' (Inr ' space N) ∩ space L
⊂ space L
  obtains f' f'' where f' ∈ L →M M f'' ∈ L →M N
  ∧x. x ∈ space L ⇒ x ∈ f -' Inl ' space M ⇒ f x = Inl (f' x)
  ∧x. x ∈ space L ⇒ x ∉ f -' Inl ' space M ⇒ f x = Inr (f'' x)
proof -
  have ne:space M ≠ {} space N ≠ {}
    using assms(2,3) measurable-space[OF assms(1)] by(fastforce simp: space-copair-measure)+
  define m where m ≡ SOME y. y ∈ space M
  define n where n ≡ SOME y. y ∈ space N

```

```

have  $m$ [measurable, simp]: $m \in \text{space } M$  and  $n$ [measurable, simp]: $n \in \text{space } N$ 
using ne by(auto simp: n-def m-def some-in-eq)
define  $f'$  where  $f' \equiv (\lambda x. \text{if } x \in f - ' \text{Inl } ' \text{space } M \text{ then } \text{SOME } y. f x = \text{Inl } y$ 
else } m)
have  $\bigwedge x. x \in \text{space } L \implies x \in f - ' \text{Inl } ' \text{space } M \implies f x = \text{Inl } (\text{SOME } y. f x$ 
 $= \text{Inl } y)$ 
unfolding  $f'$ -def by(rule someI-ex) (use assms(2) in blast)
hence  $f'$ : $\bigwedge x. x \in \text{space } L \implies x \in f - ' \text{Inl } ' \text{space } M \implies f x = \text{Inl } (f' x)$ 
by(simp add: f'-def)
hence  $f'$ -space:  $x \in \text{space } L \implies f' x \in \text{space } M$  for  $x$ 
using measurable-space[OF assms(1)]
by(cases x \in f - ' Inl ' space M) (auto simp: space-copair-measure f'-def)
define  $f''$  where  $f'' \equiv (\lambda x. \text{if } x \notin f - ' \text{Inl } ' \text{space } M \text{ then } \text{SOME } y. f x = \text{Inr } y$ 
else } n)
have  $*$ : $\bigwedge x. x \in \text{space } L \implies x \notin f - ' \text{Inl } ' \text{space } M \implies x \in f - ' \text{Inr } ' \text{space } N$ 
using measurable-space[OF assms(1)] by(fastforce simp: space-copair-measure)
have  $\bigwedge x. x \in \text{space } L \implies x \notin f - ' \text{Inl } ' \text{space } M \implies f x = \text{Inr } (\text{SOME } y. f x$ 
 $= \text{Inr } y)$ 
unfolding  $f''$ -def by(rule someI-ex) (use * in blast)
hence  $f''$ : $\bigwedge x. x \in \text{space } L \implies x \notin f - ' \text{Inl } ' \text{space } M \implies f x = \text{Inr } (f'' x)$ 
by(simp add: f''-def)
hence  $f''$ -space: $x \in \text{space } L \implies f'' x \in \text{space } N$  for  $x$ 
using measurable-space[OF assms(1), of x]
by(cases x \notin f - ' Inl ' space M) (auto simp add: space-copair-measure f''-def)
have  $f' \in L \rightarrow_M M$ 
proof -
have  $f' = (\lambda x. \text{if } x \in f - ' \text{Inl } ' \text{space } M \text{ then } f' x \text{ else } m)$ 
by(auto simp add: f'-def)
also have  $\dots \in L \rightarrow_M M$ 
proof(intro measurable-restrict-space-iff[THEN iffD1] measurableI)
fix  $A$ 
assume  $A$ [measurable]: $A \in \text{sets } M$ 
have [measurable]: $f \in \text{restrict-space } L (f - ' \text{Inl } ' \text{space } M) \rightarrow_M M \oplus_M N$ 
by(auto intro!: measurable-restrict-space1)
have [simp]: $f' - ' A \cap \text{space } (\text{restrict-space } L (f - ' \text{Inl } ' \text{space } M))$ 
 $= f - ' (\text{Inl } ' A) \cap \text{space } (\text{restrict-space } L (f - ' \text{Inl } ' \text{space } M))$ 
using  $f'$  sets.sets-into-space[OF A] by(fastforce simp: space-restrict-space)
show  $f' - ' A \cap \text{space } (\text{restrict-space } L (f - ' \text{Inl } ' \text{space } M))$ 
 $\in \text{sets } (\text{restrict-space } L (f - ' \text{Inl } ' \text{space } M))$ 
by simp
next
show  $\bigwedge x. x \in \text{space } (\text{restrict-space } L (f - ' \text{Inl } ' \text{space } M)) \implies f' x \in \text{space } M$ 
by(auto simp: space-restrict-space f'-space)
qed simp-all
finally show ?thesis .
qed
moreover have  $f'' \in L \rightarrow_M N$ 
proof -

```

```

have  $f'' = (\lambda x. \text{if } x \notin f - ' \text{Inl } ' \text{ space } M \text{ then } f'' x \text{ else } n)$ 
  by(auto simp add: f''-def)
also have  $\dots \in L \rightarrow_M N$ 
proof(rule measurable-If-restrict-space-iff [THEN iffD2, OF - conjI [OF measurableI]])
  fix  $A$ 
  assume  $A[\text{measurable}]: A \in \text{sets } N$ 
  have  $f: f \in \text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\} \rightarrow_M M \oplus_M N$ 
    by(auto intro!: measurable-restrict-space1)
  have  $1: f'' - ' A \cap \text{space } (\text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\})$ 
     $= f - ' (\text{Inr } ' A) \cap \text{space } (\text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\})$ 
    using  $f'' \text{ sets.sets-into-space [OF } A]$  by(fastforce simp: space-restrict-space)
  show  $f'' - ' A \cap \text{space } (\text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\})$ 
     $\in \text{sets } (\text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\})$ 
    unfolding 1 using f by simp
  next
  show  $\bigwedge x. x \in \text{space } (\text{restrict-space } L \{x. x \notin f - ' \text{Inl } ' \text{ space } M\}) \implies f'' x \in \text{space } N$ 
    by(auto simp: space-restrict-space f''-space)
  qed simp-all
  finally show ?thesis .
qed
ultimately show ?thesis
  using that f' f'' by blast
qed

```

2.2 Measures

lemma *emeasure-copair-measure*:

```

assumes  $[\text{measurable}]: A \in \text{sets } (M \oplus_M N)$ 
shows  $\text{emeasure } (M \oplus_M N) A = \text{emeasure } M (\text{Inl } - ' A) + \text{emeasure } N (\text{Inr } - ' A)$ 
proof(rule emeasure-measure-of)
  show  $\{\text{Inl } ' A \mid A. A \in \text{sets } M\} \cup \{\text{Inr } ' A \mid A. A \in \text{sets } N\} \subseteq \text{Pow } (\text{space } M <+> \text{space } N)$ 
    using sets.sets-into-space [of - M] sets.sets-into-space [of - N] by fastforce
  show  $A \in \text{sets } (M \oplus_M N)$ 
    by fact
  show countably-additive  $(\text{sets } (M \oplus_M N)) (\lambda a. \text{emeasure } M (\text{Inl } - ' a) + \text{emeasure } N (\text{Inr } - ' a))$ 
  proof(safe intro!: countably-additiveI)
    note  $[\text{measurable}] = \text{measurable-vimage-Inl [of - M N]} \text{ measurable-vimage-Inr [of - M N]}$ 
    fix  $A :: \text{nat} \Rightarrow \text{set}$ 
    assume  $h: \text{range } A \subseteq \text{sets } (M \oplus_M N) \text{ disjoint-family } A$ 
    then have  $[\text{measurable}]: \bigwedge i. A i \in \text{sets } (M \oplus_M N)$ 
      by blast
    have disj: disjoint-family  $(\lambda i. \text{Inl } - ' A i) \text{ disjoint-family } (\lambda i. \text{Inr } - ' A i)$ 
      using h by (auto simp: disjoint-family-on-def)

```

show $(\sum i. \text{emeasure } M (\text{Inl } -' A i) + \text{emeasure } N (\text{Inr } -' A i))$
 $= \text{emeasure } M (\text{Inl } -' \bigcup (\text{range } A)) + \text{emeasure } N (\text{Inr } -' \bigcup (\text{range } A))$ (is ?lhs = ?rhs)
proof –
have ?lhs = $(\sum i. \text{emeasure } M (\text{Inl } -' A i) + (\sum i. \text{emeasure } N (\text{Inr } -' A i)))$
by(simp add: suminf-add)
also have ... = $\text{emeasure } M (\bigcup i. (\text{Inl } -' A i)) + \text{emeasure } N (\bigcup i. (\text{Inr } -' A i))$
proof –
have $(\sum i. \text{emeasure } M (\text{Inl } -' A i)) = \text{emeasure } M (\bigcup i. (\text{Inl } -' A i))$
 $(\sum i. \text{emeasure } N (\text{Inr } -' A i)) = \text{emeasure } N (\bigcup i. (\text{Inr } -' A i))$
by(auto intro!: suminf-emeasure disj)
thus ?thesis
by argo
qed
also have ... = ?rhs
by(simp add: vimage-UN)
finally show ?thesis .
qed
qed
qed(auto simp: positive-def copair-measure-def)

lemma *emeasure-copair-measure-space*:
 $\text{emeasure } (M \oplus_M N) (\text{space } (M \oplus_M N)) = \text{emeasure } M (\text{space } M) + \text{emeasure } N (\text{space } N)$
proof –
have [simp]: $\text{Inl } -' \text{space } (M \oplus_M N) = \text{space } M$ $\text{Inr } -' \text{space } (M \oplus_M N) = \text{space } N$
by(auto simp: space-copair-measure)
show ?thesis
by(simp add: emeasure-copair-measure)
qed

corollary
shows *emeasure-copair-measure-Inl*: $A \in \text{sets } M \implies \text{emeasure } (M \oplus_M N) (\text{Inl } -' A) = \text{emeasure } M A$
and *emeasure-copair-measure-Inr*: $B \in \text{sets } N \implies \text{emeasure } (M \oplus_M N) (\text{Inr } -' B) = \text{emeasure } N B$
proof –
have [simp]: $\text{Inl } -' \text{Inl } -' A = A$ $\text{Inr } -' \text{Inl } -' A = \{\}$ $\text{Inl } -' \text{Inr } -' B = \{\}$ $\text{Inr } -' \text{Inr } -' B = B$
by auto
show $A \in \text{sets } M \implies \text{emeasure } (M \oplus_M N) (\text{Inl } -' A) = \text{emeasure } M A$
 $B \in \text{sets } N \implies \text{emeasure } (M \oplus_M N) (\text{Inr } -' B) = \text{emeasure } N B$
by(simp-all add: emeasure-copair-measure)
qed

lemma *measure-copair-measure*:

assumes $[measurable]: A \in \text{sets } (M \oplus_M N) \text{ emeasure } (M \oplus_M N) A < \infty$
shows $\text{measure } (M \oplus_M N) A = \text{measure } M (\text{Inl } - ' A) + \text{measure } N (\text{Inr } - ' A)$
using *assms(2)* **by**(*auto simp add: emeasure-copair-measure measure-def intro!: enn2real-plus*)

lemma

shows *measure-copair-measure-Inl*: $A \in \text{sets } M \implies \text{measure } (M \oplus_M N) (\text{Inl } - ' A) = \text{measure } M A$
and *measure-copair-measure-Inr*: $B \in \text{sets } N \implies \text{measure } (M \oplus_M N) (\text{Inr } - ' B) = \text{measure } N B$
by(*auto simp: emeasure-copair-measure-Inl measure-def emeasure-copair-measure-Inr*)

2.3 Finiteness

lemma *finite-measure-copair-measure*: $\text{finite-measure } M \implies \text{finite-measure } N \implies \text{finite-measure } (M \oplus_M N)$
by(*auto intro!: finite-measureI simp: emeasure-copair-measure-space finite-measure.finite-emeasure-space*)

2.4 σ -Finiteness

lemma *sigma-finite-measure-copair-measure*:

assumes *sigma-finite-measure M sigma-finite-measure N*
shows *sigma-finite-measure (M \oplus_M N)*

proof –

obtain $A B$ **where** $AB[measurable]: \bigwedge i. A i \in \text{sets } M (\bigcup (\text{range } A)) = \text{space } M \bigwedge i::\text{nat. emeasure } M (A i) \neq \infty$

$\bigwedge i. B i \in \text{sets } N (\bigcup (\text{range } B)) = \text{space } N \bigwedge i::\text{nat. emeasure } N (B i) \neq \infty$

by (*metis range-subsetD sigma-finite-measure.sigma-finite assms*)

then have $*(\bigcup (\text{range } (\lambda i. \text{Inl } - ' (A i) \cup \text{Inr } - ' (B i)))) = \text{space } (M \oplus_M N)$

unfolding *space-copair-measure Plus-def* **by** *fastforce*

have $[simp]: \bigwedge i. \text{Inl } - ' \text{Inl } - ' A i \cup \text{Inl } - ' \text{Inr } - ' B i = A i \bigwedge i. \text{Inr } - ' \text{Inl } - ' A i \cup \text{Inr } - ' \text{Inr } - ' B i = B i$

using *sets.sets-into-space AB(1,4)* **by** *blast+*

show *?thesis*

apply *standard*

using $AB *$ **by**(*auto intro!: exI[where x=range ($\lambda i. \text{Inl } - ' (A i) \cup \text{Inr } - ' (B i)$)] simp: space-copair-measure emeasure-copair-measure*)

qed

2.5 Non-Negative Integral

lemma *nn-integral-copair-measure*:

assumes $f \in \text{borel-measurable } (M \oplus_M N)$

shows $(\int^{+x}. f x \partial(M \oplus_M N)) = (\int^{+x}. f (\text{Inl } x) \partial M) + (\int^{+x}. f (\text{Inr } x) \partial N)$

using *assms*

proof *induction*

case (*cong f g*)

moreover hence $\bigwedge x. x \in \text{space } M \implies f (\text{Inl } x) = g (\text{Inl } x)$

$\bigwedge x. x \in \text{space } N \implies f (\text{Inr } x) = g (\text{Inr } x)$

```

    by(auto simp: space-copair-measure)
  ultimately show ?case
    by(simp cong: nn-integral-cong)
next
  case [measurable]:(set A)
  note [measurable] = measurable-vimage-Inl[of - M N] measurable-vimage-Inr[of
- M N]
  show ?case
    by (simp add: indicator-vimage[symmetric] emeasure-copair-measure)
next
  case (mult u c)
  then show ?case
    by(simp add: measurable-copair-measure-iff nn-integral-cmult distrib-left)
next
  case (add u v)
  then show ?case
    by(simp add: nn-integral-add)
next
  case h[measurable]:(seq U)
  have inc:  $\bigwedge x. \text{incseq } (\lambda i. U i x)$ 
    by (metis h(3) incseq-def le-funE)
  have lim:  $(\lambda i. U i x) \longrightarrow \text{Sup } (\text{range } U) x$  for x
    by (metis SUP-apply LIMSEQ-SUP[OF inc[of x]])
  have  $(\lambda i. (\int^+ x. U i x \partial(M \oplus_M N))) \longrightarrow (\int^+ x. (\text{Sup } (\text{range } U)) x \partial(M$ 
 $\oplus_M N))$ 
    by(intro nn-integral-LIMSEQ[OF - - lim]) (auto simp: h)
  moreover have  $(\lambda i. (\int^+ x. U i x \partial(M \oplus_M N))) \longrightarrow (\int^+ x. \text{Sup } (\text{range}$ 
 $U) (\text{Inl } x) \partial M) + (\int^+ x. \text{Sup } (\text{range } U) (\text{Inr } x) \partial N)$ 
    proof -
      have  $(\lambda i. (\int^+ x. U i x \partial(M \oplus_M N))) = (\lambda i. (\int^+ x. U i (\text{Inl } x) \partial M) + (\int^+$ 
 $x. U i (\text{Inr } x) \partial N))$ 
        by(simp add: h)
      also have ...  $\longrightarrow (\int^+ x. \text{Sup } (\text{range } U) (\text{Inl } x) \partial M) + (\int^+ x. \text{Sup } (\text{range}$ 
 $U) (\text{Inr } x) \partial N)$ 
        proof(rule tendsto-add)
          have inc:  $\bigwedge x. \text{incseq } (\lambda i. U i (\text{Inl } x))$ 
            by (metis h(3) incseq-def le-funE)
          have lim:  $(\lambda i. U i (\text{Inl } x)) \longrightarrow \text{Sup } (\text{range } U) (\text{Inl } x)$  for x
            by (metis SUP-apply LIMSEQ-SUP[OF inc[of x]])
          show  $(\lambda i. (\int^+ x. U i (\text{Inl } x) \partial M)) \longrightarrow (\int^+ x. \text{Sup } (\text{range } U) (\text{Inl } x)$ 
 $\partial M)$ 
            using inc by(intro nn-integral-LIMSEQ[OF - - lim]) (auto simp: incseq-def
intro!: le-funI)
        next
          have inc:  $\bigwedge x. \text{incseq } (\lambda i. U i (\text{Inr } x))$ 
            by (metis h(3) incseq-def le-funE)
          have lim:  $(\lambda i. U i (\text{Inr } x)) \longrightarrow \text{Sup } (\text{range } U) (\text{Inr } x)$  for x
            by (metis SUP-apply LIMSEQ-SUP[OF inc[of x]])
          show  $(\lambda i. (\int^+ x. U i (\text{Inr } x) \partial N)) \longrightarrow (\int^+ x. \text{Sup } (\text{range } U) (\text{Inr } x)$ 

```

```

∂N)
  using inc by(intro nn-integral-LIMSEQ[OF - - lim]) (auto simp: incseq-def
intro!: le-funI)
  qed
  finally show ?thesis .
  qed
  ultimately show ?case
  using LIMSEQ-unique by blast
qed

```

2.6 Integrability

lemma *integrable-copair-measure-iff*:

```

  fixes f :: 'a + 'b ⇒ 'c::{banach, second-countable-topology}
  shows integrable (M ⊕M N) f ⟷ integrable M (λx. f (Inl x)) ∧ integrable N
(λx. f (Inr x))
  by(auto simp add: measurable-copair-measure-iff nn-integral-copair-measure in-
tegrable-iff-bounded)

```

corollary *interable-copair-measureI*:

```

  fixes f :: 'a + 'b ⇒ 'c::{banach, second-countable-topology}
  shows integrable M (λx. f (Inl x)) ⟹ integrable N (λx. f (Inr x)) ⟹ integrable
(M ⊕M N) f
  by(simp add: integrable-copair-measure-iff)

```

2.7 The Lebesgue Integral

lemma *integral-copair-measure*:

```

  fixes f :: 'a + 'b ⇒ 'c::{banach, second-countable-topology}
  assumes integrable (M ⊕M N) f
  shows (∫ x. f x ∂(M ⊕M N)) = (∫ x. f (Inl x) ∂M) + (∫ x. f (Inr x) ∂N)
  using assms

```

proof *induction*

```

  case h[measurable]:(base A c)
  note [measurable] = measurable-vimage-Inl[of - M N] measurable-vimage-Inr[of
- M N]
  have [simp]:integrable (M ⊕M N) (indicat-real A) integrable M (indicat-real
(Inl - ' A))
    integrable N (indicat-real (Inr - ' A))
  using h(2) by(auto simp: emeasure-copair-measure)
  show ?case
  by(cases c = 0)
    (simp-all add: indicator-vimage[symmetric] measure-copair-measure mea-
sure-copair-measure[OF - h(2)] scaleR-left-distrib)
  next
  case (add f g)
  then show ?case
  by(simp add: integrable-copair-measure-iff)
  next
  case ih:(lim f s)

```

have $(\lambda n. (\int x. s \ n \ x \ \partial(M \oplus_M N))) \longrightarrow (\int x. f \ x \ \partial(M \oplus_M N))$
using $ih(1-4)$ **by** $(auto \ intro! : integral-dominated-convergence[where \ w=\lambda x. 2 * norm \ (f \ x)])$
moreover have $(\lambda n. (\int x. s \ n \ x \ \partial(M \oplus_M N))) \longrightarrow (\int x. f \ (Inl \ x) \ \partial M) + (\int x. f \ (Inr \ x) \ \partial N)$
using $ih(1-4)$
by $(auto \ intro! : integral-dominated-convergence[where \ w=\lambda x. 2 * norm \ (f \ (Inl \ x))])$
 $integral-dominated-convergence[where \ w=\lambda x. 2 * norm \ (f \ (Inr \ x))]$ *tendsto-add*
 $simp : ih(5) \ integrable-copair-measure-iff \ measurable-copair-measure-iff \ borel-measurable-integrable \ space-copair-measure \ Inl \ Inr \ I$
ultimately show *?case*
using $LIMSEQ-unique$ **by** *blast*
qed

3 Coproduct Measures

definition $coPiM :: ['i \ set, 'i \Rightarrow 'a \ measure] \Rightarrow ('i \times 'a) \ measure$ **where**
 $coPiM \ I \ Mi \equiv measure-of$
 $(SIGMA \ i:I. \ space \ (Mi \ i))$
 $\{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i - ' A \in sets \ (Mi \ i))\}$
 $(\lambda A. (\sum_{\infty i \in I. \ emeasure \ (Mi \ i) \ (Pair \ i - ' A))$

syntax

$-coPiM :: pttm \Rightarrow 'i \ set \Rightarrow 'a \ measure \Rightarrow ('i \times 'a) \ measure \ (\langle \exists \Pi_M \ - \cdot \ / \ - \rangle \ 10)$

translations

$\Pi_M \ x \in I. \ M \Leftrightarrow CONST \ coPiM \ I \ (\lambda x. \ M)$

3.1 The Measurable Space and Measurability

lemma

shows $space-coPiM : space \ (coPiM \ I \ Mi) = (SIGMA \ i:I. \ space \ (Mi \ i))$
and $sets-coPiM :$
 $sets \ (coPiM \ I \ Mi) = sigma-sets \ (SIGMA \ i:I. \ space \ (Mi \ i)) \ \{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i - ' A \in sets \ (Mi \ i))\}$
and $sets-coPiM-eq : sets \ (coPiM \ I \ Mi) = \{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i - ' A \in sets \ (Mi \ i))\}$
proof $-$
have $1 : \{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i - ' A \in sets \ (Mi \ i))\} \subseteq Pow \ (SIGMA \ i:I. \ space \ (Mi \ i))$
using $sets.sets-into-space$ **by** $auto$
show $space \ (coPiM \ I \ Mi) = (SIGMA \ i:I. \ space \ (Mi \ i))$
and $2 : sets \ (coPiM \ I \ Mi)$
 $= sigma-sets \ (SIGMA \ i:I. \ space \ (Mi \ i)) \ \{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i - ' A \in sets \ (Mi \ i))\}$
by $(auto \ simp : sets-measure-of[OF \ 1] \ space-measure-of[OF \ 1] \ coPiM-def)$
show $sets \ (coPiM \ I \ Mi) = \{A. \ A \subseteq (SIGMA \ i:I. \ space \ (Mi \ i)) \wedge (\forall i \in I. \ Pair \ i$

$- ' A \in \text{sets } (Mi\ i))\}$
proof –
have *sigma-algebra* (*SIGMA* $i:I$. *space* ($Mi\ i$)) $\{A. A \subseteq (\text{SIGMA } i:I. \text{space } (Mi\ i)) \wedge (\forall i \in I. \text{Pair } i - ' A \in \text{sets } (Mi\ i))\}$
proof (*subst Dynkin-system.sigma-algebra-eq-Int-stable*)
show *Dynkin-system* (*SIGMA* $i:I$. *space* ($Mi\ i$)) $\{A. A \subseteq (\text{SIGMA } i:I. \text{space } (Mi\ i)) \wedge (\forall i \in I. \text{Pair } i - ' A \in \text{sets } (Mi\ i))\}$
by *unfold-locales* (*auto simp: Pair-vimage-Sigma sets.Diff vimage-Diff vimage-Union 1*)
qed (*auto intro!: Int-stableI*)
thus *?thesis*
by (*auto simp: 2 intro!: sigma-algebra.sigma-sets-eq*)
qed
qed

lemma *sets-coPiM-cong*:
 $I = J \implies (\bigwedge i. i \in I \implies \text{sets } (Mi\ i) = \text{sets } (Ni\ i)) \implies \text{sets } (\text{coPiM } I\ Mi) = \text{sets } (\text{coPiM } J\ Ni)$
by (*simp cong: sets-eq-imp-space-eq Sigma-cong add: sets-coPiM*)

lemma *measurable-coPiM2*:
assumes [*measurable*]: $\bigwedge i. i \in I \implies f\ i \in Mi\ i \rightarrow_M N$
shows $(\lambda(i,x). f\ i\ x) \in \text{coPiM } I\ Mi \rightarrow_M N$
proof (*rule measurableI*)
fix A
assume [*measurable*]: $A \in \text{sets } N$
have [*simp*]:
 $\bigwedge i. i \in I$
 $\implies \text{Pair } i - ' (\lambda(x,y). f\ x\ y) - ' A \cap \text{Pair } i - ' (\text{SIGMA } i:I. \text{space } (Mi\ i)) = f\ i - ' A \cap \text{space } (Mi\ i)$
by *auto*
show $(\lambda(i,x). f\ i\ x) - ' A \cap \text{space } (\text{coPiM } I\ Mi) \in \text{sets } (\text{coPiM } I\ Mi)$
by (*auto simp: sets-coPiM space-coPiM*)
qed (*auto simp: space-coPiM measurable-space[OF assms]*)

lemma *measurable-Pair-coPiM* [*measurable* (*raw*)]:
assumes $i \in I$
shows $\text{Pair } i \in Mi\ i \rightarrow_M \text{coPiM } I\ Mi$
proof (*rule measurable-sigma-sets*)
show $\{A. A \subseteq (\text{SIGMA } i:I. \text{space } (Mi\ i)) \wedge (\forall i \in I. \text{Pair } i - ' A \in \text{sets } (Mi\ i))\} \subseteq \text{Pow } (\text{SIGMA } i:I. \text{space } (Mi\ i))$
by *blast*
qed (*auto simp: assms sets-coPiM*)

lemma *measurable-Pair-coPiM'*:
assumes $i \in I$ $(\lambda(i,x). f\ i\ x) \in \text{coPiM } I\ Mi \rightarrow_M N$
shows $f\ i \in Mi\ i \rightarrow_M N$
using *measurable-compose* [*OF measurable-Pair-coPiM assms(2)*] *assms(1)* **by** *fast*

lemma *measurable-copair-iff*: $(\lambda(i,x). f i x) \in \text{coPiM } I \text{ Mi} \rightarrow_M N \iff (\forall i \in I. f i \in \text{Mi } i \rightarrow_M N)$

by(*auto intro!*: *measurable-coPiM2 simp: measurable-Pair-coPiM'*)

lemma *measurable-copair-iff'*: $f \in \text{coPiM } I \text{ Mi} \rightarrow_M N \iff (\forall i \in I. (\lambda x. f (i, x)) \in \text{Mi } i \rightarrow_M N)$

using *measurable-copair-iff*[*of curry f*] **by**(*simp add: split-beta' curry-def*)

lemma *coPair-inverse-space-unit*:

$i \in I \implies A \in \text{sets } (\text{coPiM } I \text{ Mi}) \implies \text{Pair } i -' A \cap \text{space } (\text{Mi } i) = \text{Pair } i -' A$

using *sets.sets-into-space* **by**(*fastforce simp: space-coPiM*)

lemma *measurable-Pair-vimage*:

assumes $i \in I \ A \in \text{sets } (\text{coPiM } I \text{ Mi})$

shows $\text{Pair } i -' A \in \text{sets } (\text{Mi } i)$

using *measurable-sets*[*OF measurable-Pair-coPiM*][*OF assms(1)*] *assms(2)*

by (*simp add: coPair-inverse-space-unit* [*OF assms*])

lemma *measurable-Sigma-singleton*[*measurable (raw)*]:

$\bigwedge i \ A. i \in I \implies A \in \text{sets } (\text{Mi } i) \implies \{i\} \times A \in \text{sets } (\text{coPiM } I \text{ Mi})$

using *sets.sets-into-space sets-coPiM* **by** *fastforce*

lemma *sets-coPiM-countable*:

assumes *countable I*

shows $\text{sets } (\text{coPiM } I \text{ Mi}) = \text{sigma-sets } (\text{SIGMA } i:I. \text{space } (\text{Mi } i)) (\bigcup i \in I. (\times) \{i\} -' (\text{sets } (\text{Mi } i)))$

unfolding *sets-coPiM*

proof(*safe intro!*: *sigma-sets-eqI*)

fix *a*

assume $h:a \subseteq (\text{SIGMA } i:I. \text{space } (\text{Mi } i)) \ \forall i \in I. \text{Pair } i -' a \in \text{sets } (\text{Mi } i)$

then have $a = (\bigcup i \in I. \{i\} \times \text{Pair } i -' a)$

by *auto*

moreover have $(\bigcup i \in I. \{i\} \times \text{Pair } i -' a) \in \text{sigma-sets } (\text{SIGMA } i:I. \text{space } (\text{Mi } i)) (\bigcup i \in I. (\times) \{i\} -' (\text{sets } (\text{Mi } i)))$

using *h(2)* **by**(*auto intro!*: *sigma-sets-UNION* [*OF countable-image*] [*OF assms*])

ultimately show $a \in \text{sigma-sets } (\text{SIGMA } i:I. \text{space } (\text{Mi } i)) (\bigcup i \in I. (\times) \{i\} -' (\text{sets } (\text{Mi } i)))$

by *argo*

qed(*use sets.sets-into-space in fastforce*)

lemma *measurable-coPiM1'*:

assumes *countable I*

and [*measurable*]: $a \in N \rightarrow_M \text{count-space } I \ \bigwedge i. i \in a -' (\text{space } N) \implies g i \in N \rightarrow_M \text{Mi } i$

shows $(\lambda x. (a \ x, g (a \ x) \ x)) \in N \rightarrow_M \text{coPiM } I \text{ Mi}$

proof(*safe intro!*: *measurable-sigma-sets* [*OF sets-coPiM-countable*] [*OF assms(1)*])

fix *i B*

assume $iB[\text{measurable}]: i \in I \ B \in \text{sets } (\text{Mi } i)$

```

show  $(\lambda x. (a\ x, g\ (a\ x)\ x)) -' (\{i\} \times B) \cap \text{space } N \in \text{sets } N$ 
proof(cases  $i \in a -' (\text{space } N)$ )
  assume [measurable]: $i \in a -' (\text{space } N)$ 
  have  $(\lambda x. (a\ x, g\ (a\ x)\ x)) -' (\{i\} \times B) \cap \text{space } N = (a -' \{i\} \cap \text{space } N) \cap$ 
 $(g\ i -' B \cap \text{space } N)$ 
  by auto
  also have  $\dots \in \text{sets } N$ 
  by simp
  finally show ?thesis .
next
  assume  $i \notin a -' (\text{space } N)$ 
  then have  $(\lambda x. (a\ x, g\ (a\ x)\ x)) -' (\{i\} \times B) \cap \text{space } N = \{\}$ 
  using measurable-space[OF assms(2)] by blast
  thus ?thesis
  by simp
qed
qed(use measurable-space[OF assms(2)] measurable-space[OF assms(3)] sets.sets-into-space
in fastforce)+
```

lemma *measurable-coPiM1*:

```

assumes countable I
  and  $a \in N \rightarrow_M \text{count-space } I \wedge i. i \in I \implies g\ i \in N \rightarrow_M Mi\ i$ 
shows  $(\lambda x. (a\ x, g\ (a\ x)\ x)) \in N \rightarrow_M \text{coPiM } I\ Mi$ 
using measurable-space[OF assms(2)] by(auto intro!: measurable-coPiM1' assms)
```

lemma *measurable-coPiM1-elements*:

```

assumes countable I and [measurable]: $f \in N \rightarrow_M \text{coPiM } I\ Mi$ 
obtains  $a\ g$ 
where  $a \in N \rightarrow_M \text{count-space } I$ 
   $\wedge i. i \in I \implies \text{space } (Mi\ i) \neq \{\} \implies g\ i \in N \rightarrow_M Mi\ i$ 
   $f = (\lambda x. (a\ x, g\ (a\ x)\ x))$ 
```

proof –

```

define  $a$  where  $a \equiv \text{fst} \circ f$ 
have  $1$  [measurable]: $a \in N \rightarrow_M \text{count-space } I$ 
proof(safe intro!: measurable-count-space-eq-countable[THEN iffD2] assms)
  fix  $i$ 
  assume  $i: i \in I$ 
  have  $a -' \{i\} \cap \text{space } N = f -' (\{i\} \times \text{space } (Mi\ i)) \cap \text{space } N$ 
  using measurable-space[OF assms(2)] by(fastforce simp: a-def space-coPiM)
  also have  $\dots \in \text{sets } N$ 
  using  $i$  by auto
  finally show  $a -' \{i\} \cap \text{space } N \in \text{sets } N$  .
```

next

```

show  $\wedge x. x \in \text{space } N \implies a\ x \in I$ 
  using measurable-space[OF assms(2)] by(fastforce simp: space-coPiM a-def)
```

qed

```

define  $g$  where  $g \equiv (\lambda i\ x. \text{if } a\ x = i \text{ then } \text{snd } (f\ x) \text{ else } (\text{SOME } y. y \in \text{space } (Mi\ i)))$ 
```

```

have  $2: g\ i \in N \rightarrow_M Mi\ i$  if  $i: i \in I$  and  $ne: \text{space } (Mi\ i) \neq \{\}$  for  $i$ 
```

```

unfolding g-def
proof(safe intro!: measurable-If-restrict-space-iff[THEN iffD2] measurable-const
some-in-eq[THEN iffD2] ne)
  show  $(\lambda x. \text{snd } (f x)) \in \text{restrict-space } N \{x. a x = i\} \rightarrow_M Mi i$ 
  proof(safe intro!: measurableI)
    show  $\bigwedge x. x \in \text{space } (\text{restrict-space } N \{x. a x = i\}) \implies \text{snd } (f x) \in \text{space } (Mi i)$ 
    using measurable-space[OF assms(2)] by(fastforce simp: space-restrict-space
a-def space-coPiM)
  next
  fix A
  assume [measurable]:  $A \in \text{sets } (Mi i)$ 
  have  $(\lambda x. \text{snd } (f x)) -' A \cap \text{space } (\text{restrict-space } N \{x. a x = i\}) = f -' (\{i\} \times A) \cap \text{space } N$ 
  using i measurable-space[OF assms(2)] by(fastforce simp: space-restrict-space
a-def space-coPiM)
  also have  $\dots \in \text{sets } N$ 
  using i by simp
  finally show  $(\lambda x. \text{snd } (f x)) -' A \cap \text{space } (\text{restrict-space } N \{x. a x = i\}) \in \text{sets } (\text{restrict-space } N \{x. a x = i\})$ 
  by(auto simp: sets-restrict-space space-restrict-space)
qed
qed(use i ne in auto)
have  $\exists f = (\lambda x. (a x, g (a x) x))$ 
  by(auto simp: a-def g-def)
show ?thesis
  using 1 2 3 that by blast
qed

```

3.2 Measures

lemma *emeasure-coPiM*:

```

assumes  $A \in \text{sets } (coPiM I Mi)$ 
shows  $\text{emeasure } (coPiM I Mi) A = (\sum_{\infty i \in I. \text{emeasure } (Mi i) (Pair i -' A)})$ 
proof(rule emeasure-measure-of)
  show  $\{A. A \subseteq (\text{SIGMA } i:I. \text{space } (Mi i)) \wedge (\forall i \in I. Pair i -' A \in \text{sets } (Mi i))\} \subseteq \text{Pow } (\text{SIGMA } i:I. \text{space } (Mi i))$ 
  by blast
next
  note measurable-Pair-vimage[of - I - Mi, measurable (raw)]
  show countably-additive (sets (coPiM I Mi))  $(\lambda a. \sum_{\infty i \in I. \text{emeasure } (Mi i) (Pair i -' a)})$ 
  unfolding countably-additive-def
proof safe
  fix A :: nat  $\Rightarrow -$ 
  assume A:  $\text{range } A \subseteq \text{sets } (coPiM I Mi)$  disjoint-family A
  then have [measurable]:  $\bigwedge n. A n \in \text{sets } (coPiM I Mi)$ 
  by blast
  show  $(\sum n. \sum_{\infty i \in I. \text{emeasure } (Mi i) (Pair i -' A n)})$ 

```

$$= (\sum_{\infty i \in I}. \text{emeasure } (Mi\ i) (Pair\ i - ' \bigcup (\text{range } A))) \text{ (is } ?lhs = ?rhs)$$
proof –

have $?lhs = (\sum_{\infty n \in UNIV}. \sum_{\infty i \in I}. \text{emeasure } (Mi\ i) (Pair\ i - ' A\ n))$

by(*auto intro!*: *infsum-eq-suminf*[*symmetric*] *nonneg-summable-on-complete*)

also have $\dots = (\sum_{\infty i \in I}. \sum_{\infty n \in UNIV}. \text{emeasure } (Mi\ i) (Pair\ i - ' A\ n))$

by(*rule infsum-swap-ennreal*)

also have $\dots = ?rhs$

proof(*rule infsum-cong*)

fix i

assume $i \in I$

then have $(\sum n. Mi\ i (Pair\ i - ' A\ n)) = Mi\ i (\bigcup n. Pair\ i - ' A\ n)$

using $A(2)$ **by**(*intro suminf-emeasure*) (*auto simp: disjoint-family-on-def*)

also have $\dots = Mi\ i (Pair\ i - ' \bigcup (\text{range } A))$

by (*metis vimage-UN*)

finally show $(\sum_{\infty n}. \text{emeasure } (Mi\ i) (Pair\ i - ' A\ n)) = \text{emeasure } (Mi\ i)$

 $(Pair\ i - ' \bigcup (\text{range } A))$

by(*auto simp: infsum-eq-suminf*[*OF nonneg-summable-on-complete*])

qed

finally show $?thesis$.

qed

qed

next

show $A \in \text{sets } (coPiM\ I\ Mi)$

by *fact*

qed(*auto simp: positive-def coPiM-def*)

corollary *emeasure-coPiM-space*:

 $\text{emeasure } (coPiM\ I\ Mi) (\text{space } (coPiM\ I\ Mi)) = (\sum_{\infty i \in I}. \text{emeasure } (Mi\ i) (\text{space } (Mi\ i)))$

proof –

have [*simp*]: $\bigwedge i. i \in I \implies Pair\ i - ' \text{space } (coPiM\ I\ Mi) = \text{space } (Mi\ i)$

by(*auto simp: space-coPiM*)

show $?thesis$

by(*auto simp: emeasure-coPiM intro!: infsum-cong*)

qed

lemma *emeasure-coPiM-coproj*:

assumes [*measurable*]: $i \in I\ A \in \text{sets } (Mi\ i)$

shows $\text{emeasure } (coPiM\ I\ Mi) (\{i\} \times A) = \text{emeasure } (Mi\ i)\ A$

proof –

have $\text{emeasure } (coPiM\ I\ Mi) (\{i\} \times A) = (\sum_{\infty j \in I}. \text{emeasure } (Mi\ j) (\text{if } j = i \text{ then } A \text{ else } \{\}))$

by(*simp add: emeasure-coPiM*)

also have $\dots = (\sum_{\infty j \in (I - \{i\}) \cup \{i\}}. \text{emeasure } (Mi\ j) (\text{if } j = i \text{ then } A \text{ else } \{\}))$

by(*rule arg-cong*[**where** $f = \text{infsum } -$] (*use assms in auto*))

also have $\dots = (\sum_{\infty j \in I - \{i\}}. \text{emeasure } (Mi\ j) (\text{if } j = i \text{ then } A \text{ else } \{\}))$

 $+ (\sum_{\infty j \in \{i\}}. \text{emeasure } (Mi\ j) (\text{if } j = i \text{ then } A \text{ else } \{\}))$

by(*rule infsum-Un-disjoint*) (*auto intro!: nonneg-summable-on-complete*)

also have ... = $\text{emeasure } (Mi \ i) \ A$
proof –
have $(\sum_{\infty j \in I - \{i\}}. \text{emeasure } (Mi \ j) \ (\text{if } j = i \text{ then } A \text{ else } \{\})) = 0$
by (rule *infsun-0*) *simp*
thus ?thesis **by** *simp*
qed
finally show ?thesis .
qed

lemma *measure-coPiM-coproj*: $i \in I \implies A \in \text{sets } (Mi \ i) \implies \text{measure } (\text{coPiM } I \ Mi) \ (\{i\} \times A) = \text{measure } (Mi \ i) \ A$
by(*simp add: emeasure-coPiM-coproj measure-def*)

lemma *emeasure-coPiM-less-top-summable*:

assumes [*measurable*]: $A \in \text{sets } (\text{coPiM } I \ Mi) \ \text{emeasure } (\text{coPiM } I \ Mi) \ A < \infty$
shows $(\lambda i. \text{measure } (Mi \ i) \ (\text{Pair } i \ -' \ A)) \ \text{summable-on } I$
proof –
have *: $(\sum_{\infty i \in I}. \text{emeasure } (Mi \ i) \ (\text{Pair } i \ -' \ A)) < \text{top}$
using *assms(2)* **by**(*simp add: emeasure-coPiM*)
from *infsun-less-top-dest[OF this]* **have** *ifin*: $\bigwedge i. i \in I \implies \text{emeasure } (Mi \ i) \ (\text{Pair } i \ -' \ A) < \text{top}$
by *simp*
with * **have** $(\sum_{\infty i \in I}. \text{ennreal } (\text{measure } (Mi \ i) \ (\text{Pair } i \ -' \ A))) < \text{top}$
by (*metis (mono-tags, lifting) emeasure-eq-ennreal-measure infsun-cong top.not-eq-extremum*)
thus ?thesis
by(*auto intro!: bounded-infsun-summable*)
qed

lemma *measure-coPiM*:

assumes [*measurable*]: $A \in \text{sets } (\text{coPiM } I \ Mi) \ \text{emeasure } (\text{coPiM } I \ Mi) \ A < \infty$
shows $\text{measure } (\text{coPiM } I \ Mi) \ A = (\sum_{\infty i \in I}. \text{measure } (Mi \ i) \ (\text{Pair } i \ -' \ A))$
proof(*subst ennreal-inj[symmetric]*)
have *: $(\sum_{\infty i \in I}. \text{emeasure } (Mi \ i) \ (\text{Pair } i \ -' \ A)) < \text{top}$
using *assms(2)* **by**(*simp add: emeasure-coPiM*)
from *infsun-less-top-dest[OF this]* **have** *ifin*: $\bigwedge i. i \in I \implies \text{emeasure } (Mi \ i) \ (\text{Pair } i \ -' \ A) < \text{top}$
by *simp*
show $\text{ennreal } (\text{measure } (\text{coPiM } I \ Mi) \ A) = \text{ennreal } (\sum_{\infty i \in I}. \text{measure } (Mi \ i) \ (\text{Pair } i \ -' \ A))$ (**is** ?lhs = ?rhs)
proof –
have ?lhs = $\text{emeasure } (\text{coPiM } I \ Mi) \ A$
using *assms* **by**(*auto intro!: emeasure-eq-ennreal-measure[symmetric]*)
also have ... = $(\sum_{\infty i \in I}. \text{emeasure } (Mi \ i) \ (\text{Pair } i \ -' \ A))$
by(*simp add: emeasure-coPiM*)
also have ... = $(\sum_{\infty i \in I}. \text{ennreal } (\text{measure } (Mi \ i) \ (\text{Pair } i \ -' \ A)))$
using *ifin* **by**(*fastforce intro!: infsun-cong emeasure-eq-ennreal-measure*)
also have ... = ?rhs
by(*simp add: infsun-ennreal-eq[OF emeasure-coPiM-less-top-summable[OF assms]]*)

finally show *?thesis* .
qed
qed(*auto intro!*: *infsum-nonneg*)

3.3 Non-Negative Integral

lemma *nn-integral-coPiM*:
assumes $f \in \text{borel-measurable } (\text{coPiM } I \text{ } Mi)$
shows $(\int^+ x. f \ x \ \partial \text{coPiM } I \text{ } Mi) = (\sum_{\infty} i \in I. (\int^+ x. f \ (i, x) \ \partial Mi \ i))$
using *assms*
proof *induction*
case (*cong f g*)
moreover hence $\bigwedge i \ x. i \in I \implies x \in \text{space } (Mi \ i) \implies f \ (i, x) = g \ (i, x)$
by(*auto simp: space-coPiM*)
ultimately show *?case*
by(*simp cong: nn-integral-cong infsum-cong*)
next
case [*measurable*]:(*set A*)
note [*measurable*] = *measurable-Pair-vimage[OF - this]*
show *?case*
by(*simp add: indicator-vimage[symmetric] emeasure-coPiM cong: infsum-cong*)
next
case (*add u v*)
then show *?case*
by(*simp add: nn-integral-add infsum-add nonneg-summable-on-complete cong: infsum-cong*)
next
case (*mult u c*)
then show *?case*
by(*simp add: nn-integral-cmult infsum-cmult-right-ennreal cong: infsum-cong*)
next
case *ih*[*measurable*]:(*seq U*)
show *?case* (**is** *?lhs = ?rhs*)
proof –
have *?lhs* = $(\int^+ x. (\text{SUP } j. U \ j \ x) \ \partial \text{coPiM } I \text{ } Mi)$
by(*auto intro!: nn-integral-cong simp: SUP-apply[symmetric]*)
also have $\dots = (\text{SUP } j. (\int^+ x. U \ j \ x \ \partial \text{coPiM } I \text{ } Mi))$
by(*auto intro!: nn-integral-monotone-convergence-SUP ih(3)*)
also have $\dots = (\text{SUP } j. (\sum_{\infty} i \in I. (\int^+ x. U \ j \ (i, x) \ \partial Mi \ i)))$
by(*simp add: ih*)
also have $\dots = (\sum_{\infty} i \in I. (\text{SUP } j. (\int^+ x. U \ j \ (i, x) \ \partial Mi \ i)))$
using *ih(3)* **by**(*auto intro!: ennreal-infsum-Sup-eq[symmetric] incseq-nn-integral simp: mono-compose*)
also have $\dots = (\sum_{\infty} i \in I. (\int^+ x. (\text{SUP } j. U \ j \ (i, x) \ \partial Mi \ i)))$
using *ih(3)* **by**(*auto intro!: infsum-cong nn-integral-monotone-convergence-SUP[symmetric] mono-compose*)
also have $\dots = ?rhs$
by(*simp add: SUP-apply[symmetric]*)
finally show *?thesis* .

qed
qed

3.4 Integrability

lemma

fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$
 assumes $\text{integrable } (\text{coPiM } I \text{ } Mi) \text{ } f$
 shows $\text{integrable-coPiM-dest-sum}:(\sum_{\infty} i \in I. (\int^+ x. \text{norm } (f \text{ } (i, x)) \text{ } \partial Mi \text{ } i)) < \infty$
 and $\text{integrable-coPiM-dest-integrable}:\bigwedge i. i \in I \implies \text{integrable } (Mi \text{ } i) (\lambda x. f \text{ } (i, x))$
 and $\text{integrable-coPiM-summable-norm}:(\lambda i. (\int x. \text{norm } (f \text{ } (i, x)) \text{ } \partial Mi \text{ } i)) \text{ summable-on } I$
 and $\text{integrable-coPiM-abs-summable}:\text{Infinite-Sum.abs-summable-on } (\lambda i. (\int x. f \text{ } (i, x) \text{ } \partial Mi \text{ } i)) \text{ } I$
 and $\text{integrable-coPiM-summable}:(\lambda i. (\int x. f \text{ } (i, x) \text{ } \partial Mi \text{ } i)) \text{ summable-on } I$
 proof -
 show $\text{fin}:(\sum_{\infty} i \in I. (\int^+ x. \text{norm } (f \text{ } (i, x)) \text{ } \partial Mi \text{ } i)) < \infty$
 using assms by $(\text{auto simp: integrable-iff-bounded nn-integral-coPiM})$
 thus $\text{integ}:i \in I \implies \text{integrable } (Mi \text{ } i) (\lambda x. f \text{ } (i, x))$ for i
 using assms by $(\text{auto simp: integrable-iff-bounded intro!: infsum-less-top-dest[of - - i]})$
 show $\text{summable}:(\lambda i. (\int x. \text{norm } (f \text{ } (i, x)) \text{ } \partial Mi \text{ } i)) \text{ summable-on } I$
 using $\text{nn-integral-eq-integral[OF integrable-norm[OF integ]]}$ fin
 by $(\text{auto intro!: bounded-infsum-summable cong: infsum-cong})$
 show $\text{Infinite-Sum.abs-summable-on } (\lambda i. (\int x. f \text{ } (i, x) \text{ } \partial Mi \text{ } i)) \text{ } I$
 by $(\text{rule summable-on-comparison-test[OF summable]})$ auto
 thus $(\lambda i. (\int x. f \text{ } (i, x) \text{ } \partial Mi \text{ } i)) \text{ summable-on } I$
 using $\text{abs-summable-summable}$ by fastforce
 qed

3.5 The Lebesgue Integral

lemma integral-coPiM :

fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$
 assumes $\text{integrable } (\text{coPiM } I \text{ } Mi) \text{ } f$
 shows $(\int x. f \text{ } x \text{ } \partial \text{coPiM } I \text{ } Mi) = (\sum_{\infty} i \in I. (\int x. f \text{ } (i, x) \text{ } \partial Mi \text{ } i))$
 using assms
 proof induction
 case $h[\text{measurable}]:(\text{base } A \text{ } c)$
 note $[\text{measurable}] = \text{measurable-Pair-vimage[OF - this(1)]}$
 have $[\text{simp}]: \text{integrable } (\text{coPiM } I \text{ } Mi) (\text{indicat-real } A)$
 $\bigwedge i. i \in I \implies \text{integrable } (Mi \text{ } i) (\text{indicat-real } (\text{Pair } i \text{ } - ' A))$
 using $h(2)$ by $(\text{auto simp: emeasure-coPiM dest: infsum-less-top-dest})$
 show $?case$
 using $h(2)$ $\text{emeasure-coPiM-less-top-summable[OF h]}$
 by $(\text{cases } c = 0)$
 $(\text{auto simp: measure-coPiM indicator-vimage[symmetric] infsum-scaleR-left[symmetric] cong: infsum-cong})$

```

next
case h:(add f g)
show ?case (is ?lhs = ?rhs)
proof -
  have ?lhs = ( $\sum_{\infty i \in I. (\int x. f (i, x) \partial Mi i)$ ) + ( $\sum_{\infty i \in I. (\int x. g (i, x) \partial Mi i)$ )
    using h by simp
  also have ... = ( $\sum_{\infty i \in I. (\int x. f (i, x) \partial Mi i) + (\int x. g (i, x) \partial Mi i)$ )
    by(auto intro!: infsum-add[symmetric] integrable-coPiM-summable h)
  also have ... = ?rhs
    using h
    by(auto intro!: infsum-cong Bochner-Integration.integral-add[symmetric] integrable-coPiM-dest-integrable)
  finally show ?thesis .
qed
next
case ih:(lim f fn)
note [measurable,simp] = ih(1-4)
show ?case (is ?lhs = ?rhs)
proof -
  have ?lhs = lim ( $\lambda n. (\int x. fn n x \partial(\text{coPiM } I \text{ } Mi))$ )
    by(auto intro!: limI[symmetric] integral-dominated-convergence[where w= $\lambda x. 2 * norm (f x)$ ])
  also have ... = lim ( $\lambda n. (\sum_{\infty i \in I. (\int x. fn n (i, x) \partial Mi i)$ )
    by(simp add: ih(5))
  also have ... = lim ( $\lambda n. (\int i. (\int x. fn n (i, x) \partial Mi i) \partial \text{count-space } I)$ )
    by(simp add: integrable-coPiM-abs-summable infsum-eq-integral)
  also have ... = ( $\int i. (\int x. f (i, x) \partial Mi i) \partial \text{count-space } I$ )
  proof(intro limI integral-dominated-convergence[where w= $\lambda i. (\int x. 2 * norm (f (i, x)) \partial Mi i)$ ] AE-I2 )
    show integrable (count-space I) ( $\lambda i. (\int x. 2 * norm (f (i, x)) \partial Mi i)$ )
    by(auto simp: abs-summable-on-integrable-iff[symmetric] integrable-coPiM-summable-norm[OF ih(4)])
  next
  show  $i \in \text{space } (\text{count-space } I) \implies (\lambda n. (\int x. fn n (i, x) \partial Mi i)) \longrightarrow (\int x. f (i, x) \partial Mi i)$  for i
    by(auto intro!: integral-dominated-convergence[where w= $\lambda x. 2 * norm (f (i, x))$ ] integrable-coPiM-dest-integrable
      simp: space-coPiM)
  next
  show  $i \in \text{space } (\text{count-space } I) \implies norm ((\int x. fn n (i, x) \partial Mi i)) \leq (\int x. 2 * norm (f (i, x)) \partial Mi i)$  for n i
    by(rule order.trans[where b= $(\int x. norm (fn n (i, x)) \partial Mi i)$ ]
      (auto simp: space-coPiM
        simp del: Bochner-Integration.integral-mult-right-zero Bochner-Integration.integral-mult-right
        intro!: integral-mono integrable-coPiM-dest-integrable)
    qed simp-all
  also have ... = ?rhs
    by(simp add: infsum-eq-integral integrable-coPiM-abs-summable)
  finally show ?thesis .

```

qed
qed

3.6 Finite Coproduct Measures

lemma *emeasure-coPiM-finite*:

assumes *finite I A ∈ sets (coPiM I Mi)*
shows $emeasure (coPiM I Mi) A = (\sum_{i \in I}. emeasure (Mi i) (Pair i - ' A))$
using *assms* **by** (*simp add: emeasure-coPiM*)

lemma *emeasure-coPiM-finite-space*:

finite I $\implies emeasure (coPiM I Mi) (space (coPiM I Mi)) = (\sum_{i \in I}. emeasure (Mi i) (space (Mi i)))$
by (*simp add: emeasure-coPiM-space*)

lemma *measure-coPiM-finite*:

assumes *finite I A ∈ sets (coPiM I Mi) emeasure (coPiM I Mi) A < ∞*
shows $measure (coPiM I Mi) A = (\sum_{i \in I}. measure (Mi i) (Pair i - ' A))$
using *assms(3)* **by** (*simp add: emeasure-coPiM-finite[OF assms(1,2)] measure-def enn2real-sum assms(1)*)

lemma *nn-integral-coPiM-finite*:

assumes *finite I f ∈ borel-measurable (coPiM I Mi)*
shows $(\int^{+x}. f x \partial(coPiM I Mi)) = (\sum_{i \in I}. (\int^{+x}. f (i, x) \partial(Mi i)))$
by (*simp add: nn-integral-coPiM assms*)

lemma *integrable-coPiM-finite-iff*:

fixes $f :: - \Rightarrow 'c :: \{banach, second-countable-topology\}$
shows *finite I* $\implies integrable (coPiM I Mi) f \iff (\forall i \in I. integrable (Mi i) (\lambda x. f (i, x)))$
using *measurable-copair-iff'[of f I Mi borel]*
by (*auto simp: integrable-iff-bounded nn-integral-coPiM-finite*)

lemma *integral-coPiM-finite*:

fixes $f :: - \Rightarrow 'c :: \{banach, second-countable-topology\}$
assumes *finite I integrable (coPiM I Mi) f*
shows $(\int x. f x \partial(coPiM I Mi)) = (\sum_{i \in I}. (\int x. f (i, x) \partial(Mi i)))$
by (*auto simp: assms integral-coPiM*)

3.7 Countable Infinite Coproduct Measures

lemma *emeasure-coPiM-countable-infinite*:

assumes [*measurable*]: *bij-betw from-n (UNIV :: nat set) I A ∈ sets (coPiM I Mi)*
shows $emeasure (coPiM I Mi) A = (\sum n. emeasure (Mi (from-n n)) (Pair (from-n n) - ' A))$

proof –

have *I:countable I*
using *assms(1) countableI-bij* **by** *blast*

have $[measurable, simp]: Pair (from-n n) - ' A \in sets (Mi (from-n n)) from-n n \in I$ **for** n
using $bij-betwE[OF assms(1)]$ **by** $(auto intro!: measurable-Pair-vimage[where I=I])$
have $emeasure (coPiM I Mi) A = emeasure (coPiM I Mi) (\bigcup n. \{from-n n\} \times Pair (from-n n) - ' A)$
using $sets.sets-into-space[OF assms(2)] assms(1)$
by $(fastforce intro!: arg-cong[where f=emeasure (coPiM I Mi)] simp: space-coPiM bij-betw-def)$
also have $\dots = (\sum n. emeasure (Mi (from-n n)) (Pair (from-n n) - ' A))$
using $injD[OF bij-betw-imp-inj-on[OF assms(1)]]$
by $(subst suminf-emeasure[symmetric])$
 $(auto simp: disjoint-family-on-def emeasure-coPiM-coproj intro!: suminf-cong)$
finally show $?thesis$.
qed

lemmas $emeasure-coPiM-countable-infinite' = emeasure-coPiM-countable-infinite[OF bij-betw-from-nat-into]$
lemmas $emeasure-coPiM-nat = emeasure-coPiM-countable-infinite[OF bij-id, simplified]$

lemma $measure-coPiM-countable-infinite$:

assumes $[measurable, simp]: bij-betw from-n (UNIV :: nat set) I A \in sets (coPiM I Mi)$
and $emeasure (coPiM I Mi) A < \infty$
shows $measure (coPiM I Mi) A = (\sum n. measure (Mi (from-n n)) (Pair (from-n n) - ' A))$ **(is ?lhs = ?rhs)**
and $summable (\lambda n. measure (Mi (from-n n)) (Pair (from-n n) - ' A))$
proof –
have $ennreal ?lhs = emeasure (coPiM I Mi) A$
using $assms(3)$ **by** $(auto intro!: emeasure-eq-ennreal-measure[symmetric])$
also have $\dots = (\sum n. emeasure (Mi (from-n n)) (Pair (from-n n) - ' A))$
by $(simp add: emeasure-coPiM-countable-infinite)$
also have $\dots = (\sum n. ennreal (measure (Mi (from-n n)) (Pair (from-n n) - ' A)))$
using $assms(3)$ $ennreal-suminf-lessD top.not-eq-extremum$
by $(auto intro!: suminf-cong emeasure-eq-ennreal-measure simp: emeasure-coPiM-countable-infinite[OF assms(1)])$
finally have $*:ennreal ?lhs = (\sum n. ennreal (measure (Mi (from-n n)) (Pair (from-n n) - ' A)))$.
thus $**[simp]: summable (\lambda n. measure (Mi (from-n n)) (Pair (from-n n) - ' A))$
by $(auto intro!: summable-suminf-not-top)$
show $?lhs = ?rhs$
proof $(subst ennreal-inj[symmetric])$
have $ennreal ?lhs = (\sum n. ennreal (measure (Mi (from-n n)) (Pair (from-n n) - ' A)))$
by $fact$
also have $\dots = ennreal ?rhs$
using $assms(3)$ **by** $(auto intro!: suminf-ennreal2)$
finally show $ennreal ?lhs = ennreal ?rhs$.

qed(*simp-all add: suminf-nonneg*)
qed

lemmas *measure-coPiM-countable-infinite'* = *measure-coPiM-countable-infinite*[*OF bij-betw-from-nat-into*]

lemmas *measure-coPiM-nat* = *measure-coPiM-countable-infinite*[*OF bij-id,simplified id-apply*]

lemma *nn-integral-coPiM-countable-infinite*:

assumes [*measurable*]:*bij-betw from-n (UNIV :: nat set) I f* ∈ *borel-measurable (coPiM I Mi)*

shows $(\int^+ x. f x \partial(\text{coPiM } I \text{ } Mi)) = (\sum n. (\int^+ x. f (\text{from-n } n, x) \partial(Mi (\text{from-n } n))))$ (**is** - = ?*rhs*)

proof -

have $(\int^+ x. f x \partial(\text{coPiM } I \text{ } Mi)) = (\sum_{\infty} i \in I. (\int^+ x. f (i, x) \partial Mi i))$

by(*simp add: nn-integral-coPiM*)

also have ... = $(\sum_{\infty} i \in \text{from-n } 'UNIV. (\int^+ x. f (i, x) \partial Mi i))$

by(*rule arg-cong[where f=infsum -]*) (*metis assms(1) bij-betw-def*)

also have ... = $(\sum_{\infty} n \in UNIV. (\int^+ x. f (\text{from-n } n, x) \partial(Mi (\text{from-n } n))))$

by(*rule infsum-reindex[simplified comp-def]*) (*use assms(1) bij-betw-imp-inj-on*)

in *blast*)

also have ... = ?*rhs*

by(*auto intro!: infsum-eq-suminf nonneg-summable-on-complete*)

finally show ?*thesis* .

qed

lemmas *nn-integral-coPiM-countable-infinite'* = *nn-integral-coPiM-countable-infinite*[*OF bij-betw-from-nat-into*]

lemmas *nn-integral-coPiM-nat* = *nn-integral-coPiM-countable-infinite*[*OF bij-id,simplified*]

lemma

fixes *f* :: - ⇒ 'b::{*banach, second-countable-topology*}

assumes *bij-betw from-n (UNIV :: nat set) I integrable (coPiM I Mi) f*

shows *integrable-coPiM-countable-infinite-dest-sum*: $(\sum n. (\int^+ x. \text{norm } (f (\text{from-n } n, x)) \partial(Mi (\text{from-n } n)))) < \infty$

and *integrable-coPiM-countable-infinite-dest'*: $\bigwedge n. \text{integrable } (Mi (\text{from-n } n)) (\lambda x. f (\text{from-n } n, x))$

using *ennreal-suminf-lessD assms(1,2) bij-betwE[OF assms(1)]*

by(*auto simp: integrable-iff-bounded nn-integral-coPiM-countable-infinite*)

lemmas *integrable-coPiM-countable-infinite-dest-sum'* = *integrable-coPiM-countable-infinite-dest-sum*[*OF bij-betw-from-nat-into*]

lemmas *integrable-coPiM-countable-infinite-dest''* = *integrable-coPiM-countable-infinite-dest'*[*OF bij-betw-from-nat-into*]

lemmas *integrable-coPiM-nat-dest-sum* = *integrable-coPiM-countable-infinite-dest-sum*[*OF bij-id,simplified id-apply*]

lemmas *integrable-coPiM-nat-dest* = *integrable-coPiM-countable-infinite-dest'*[*OF bij-id,simplified id-apply*]

lemma

fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$
assumes $\text{bij-betw from-n (UNIV :: nat set) I integrable (coPiM I Mi) f}$
shows $\text{integrable-coPiM-countable-infinite-summable-norm: summable } (\lambda n. (\int x. \text{norm } (f \text{ (from-n } n, x)) \partial(\text{Mi (from-n } n))))$
and $\text{integrable-coPiM-countable-infinite-summable-norm': summable } (\lambda n. \text{norm } (\int x. f \text{ (from-n } n, x) \partial(\text{Mi (from-n } n))))$
and $\text{integrable-coPiM-countable-infinite-summable: summable } (\lambda n. (\int x. f \text{ (from-n } n, x) \partial(\text{Mi (from-n } n))))$
proof –
show $*:\text{summable } (\lambda n. (\int x. \text{norm } (f \text{ (from-n } n, x)) \partial(\text{Mi (from-n } n))))$
using $\text{integrable-coPiM-countable-infinite-dest-sum[OF assms]}$
 $\text{nn-integral-eq-integral[OF integrable-norm[OF integrable-coPiM-countable-infinite-dest'[OF assms]]]}$
by $(\text{auto intro!: summable-suminf-not-top})$
show $\text{summable } (\lambda n. \text{norm } (\int x. f \text{ (from-n } n, x) \partial(\text{Mi (from-n } n))))$
by $(\text{rule summable-comparison-test-ev[OF - *] auto})$
thus $\text{summable } (\lambda n. (\int x. f \text{ (from-n } n, x) \partial(\text{Mi (from-n } n))))$
using $\text{summable-norm-cancel by force}$
qed

lemmas $\text{integrable-coPiM-countable-infinite-summable-norm''}$
 $= \text{integrable-coPiM-countable-infinite-summable-norm[OF bij-betw-from-nat-into]}$
lemmas $\text{integrable-coPiM-countable-infinite-summable-norm'''}$
 $= \text{integrable-coPiM-countable-infinite-summable-norm'[OF bij-betw-from-nat-into]}$
lemmas $\text{integrable-coPiM-countable-infinite-summable'}$
 $= \text{integrable-coPiM-countable-infinite-summable[OF bij-betw-from-nat-into]}$
lemmas $\text{integrable-coPiM-nat-summable-norm}$
 $= \text{integrable-coPiM-countable-infinite-summable-norm[OF bij-id,simplified id-apply]}$
lemmas $\text{integrable-coPiM-nat-summable-norm'}$
 $= \text{integrable-coPiM-countable-infinite-summable-norm'[OF bij-id,simplified id-apply]}$
lemmas $\text{integrable-coPiM-nat-summable}$
 $= \text{integrable-coPiM-countable-infinite-summable[OF bij-id,simplified id-apply]}$

lemma
fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$
assumes $\text{countable I infinite I integrable (coPiM I Mi) f}$
shows $\text{integrable-coPiM-countable-infinite-dest: } \bigwedge i. i \in I \Longrightarrow \text{integrable (Mi } i) (\lambda x. f \text{ (} i, x))$
using $\text{integrable-coPiM-countable-infinite-dest'[OF bij-betw-from-nat-into[OF assms(1,2)]]}$
 assms(3)
by $(\text{meson assms(1) countable-all})$

lemma $\text{integrable-coPiM-countable-infiniteI:}$
fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$
assumes $\text{bij-betw from-n (UNIV :: nat set) I } \bigwedge i. i \in I \Longrightarrow (\lambda x. f \text{ (} i, x)) \in \text{borel-measurable (Mi } i)$
and $(\sum n. (\int^+ x. \text{norm } (f \text{ (from-n } n, x)) \partial(\text{Mi (from-n } n)))) < \infty$
shows $\text{integrable (coPiM I Mi) f}$
using $\text{nn-integral-coPiM-countable-infinite[OF assms(1),of - Mi] assms(2,3)}$

by(*auto simp: measurable-copair-iff' integrable-iff-bounded*)

lemmas *integrable-coPiM-countable-infiniteI' = integrable-coPiM-countable-infiniteI*[*OF bij-betw-from-nat-into*]

lemmas *integrable-coPiM-natI = integrable-coPiM-countable-infiniteI*[*OF bij-id, simplified id-apply*]

lemma *integral-coPiM-countable-infinite:*

fixes $f :: - \Rightarrow 'b::\{\text{banach, second-countable-topology}\}$

assumes *bij-betw from-n (UNIV :: nat set) I integrable (coPiM I Mi) f*

shows $(\int x. f x \partial(\text{coPiM } I \text{ } Mi)) = (\sum n. (\int x. f (\text{from-n } n, x) \partial(Mi (\text{from-n } n))))$ (**is** *?lhs = ?rhs*)

proof –

have *?lhs = $(\sum_{\infty} i \in I. (\int x. f (i, x) \partial Mi i))$*

by(*simp add: integral-coPiM assms*)

also have $\dots = (\sum_{\infty} i \in \text{from-n } 'UNIV. (\int x. f (i, x) \partial Mi i))$

by(*rule arg-cong[where f=infsum -] (metis assms(1) bij-betw-def)*)

also have $\dots = (\sum_{\infty} n \in UNIV. (\int x. f (\text{from-n } n, x) \partial(Mi (\text{from-n } n))))$

by(*rule infsum-reindex[simplified comp-def] (use assms(1) bij-betw-imp-inj-on*

in *blast*)

also have $\dots = ?rhs$

by(*auto intro!: infsum-eq-suminf norm-summable-imp-summable-on integrable-coPiM-countable-infinite-sum assms*)

finally show *?thesis .*

qed

lemmas *integral-coPiM-countable-infinite' = integral-coPiM-countable-infinite*[*OF bij-betw-from-nat-into*]

lemmas *integral-coPiM-nat = integral-coPiM-countable-infinite*[*OF bij-id, simplified id-apply*]

3.8 Finiteness

lemma *finite-measure-coPiM:*

assumes *finite I $\wedge i. i \in I \implies$ finite-measure (Mi i)*

shows *finite-measure (coPiM I Mi)*

by(*rule finite-measureI (auto simp: emeasure-coPiM-finite finite-measure.emeasure-finite assms)*)

3.9 σ -Finiteness

lemma *sigma-finite-measure-coPiM:*

assumes *countable I $\wedge i. i \in I \implies$ sigma-finite-measure (Mi i)*

shows *sigma-finite-measure (coPiM I Mi)*

proof

have $\exists A. \text{range } A \subseteq \text{sets } (Mi i) \wedge (\bigcup n. A n) = \text{space } (Mi i) \wedge (\forall n::\text{nat. emeasure } (Mi i) (A n) \neq \infty)$

if $i \in I$ **for** i

using *sigma-finite-measure.sigma-finite*[*OF assms(2)*][*OF that*] **by** *metis*

hence $\exists A. \forall i \in I. \text{range } (A \ i) \subseteq \text{sets } (M \ i) \wedge (\bigcup n. A \ i \ n) = \text{space } (M \ i) \wedge$
 $(\forall n::\text{nat}. \text{emeasure } (M \ i) (A \ i \ n) \neq \infty)$
by *metis*
then obtain A_i
where $A_i[\text{measurable}]: \bigwedge i \ n. i \in I \implies A_i \ i \ n \in \text{sets } (M \ i)$
 $\bigwedge i. i \in I \implies (\bigcup n::\text{nat}. (A_i \ i \ n)) = \text{space } (M \ i)$
 $\bigwedge i \ n. i \in I \implies \text{emeasure } (M \ i) (A_i \ i \ n) \neq \infty$
by (*metis UNIV-I sets-range*)
show $\exists A. \text{countable } A \wedge A \subseteq \text{sets } (\text{coPiM } I \ M) \wedge \bigcup A = \text{space } (\text{coPiM } I \ M)$
 $\wedge (\forall a \in A. \text{emeasure } (\text{coPiM } I \ M) a \neq \infty)$
proof (*intro exI[where x= $\bigcup n. (\bigcup i \in I. \{\{i\} \times A_i \ i \ n\})$] conjI ballI*)
show $\text{countable } (\bigcup n. (\bigcup i \in I. \{\{i\} \times A_i \ i \ n\}))$
using *assms(1) by auto*
next
show $(\bigcup n. \bigcup i \in I. \{\{i\} \times A_i \ i \ n\}) \subseteq \text{sets } (\text{coPiM } I \ M)$
by *auto*
next
show $\bigcup (\bigcup n. \bigcup i \in I. \{\{i\} \times A_i \ i \ n\}) = \text{space } (\text{coPiM } I \ M)$
using *sets.sets-into-space[OF Ai(1)] Ai(2) by(fastforce simp: space-coPiM)*
next
fix a
assume $a \in (\bigcup n. \bigcup i \in I. \{\{i\} \times A_i \ i \ n\})$
then obtain $n \ i$ **where** $a: i \in I \ a = \{i\} \times A_i \ i \ n$
by *blast*
show $\text{emeasure } (\text{coPiM } I \ M) a \neq \infty$
using $a(1) \ Ai(3) \ \text{assms}$ **by** (*auto simp: a(2) emeasure-coPiM-coproj*)
qed
qed
end

4 Additional Properties

theory *Coproduct-Measure-Additional*
imports *Coproduct-Measure*
Standard-Borel-Spaces.StandardBorel
S-Finite-Measure-Monad.Kernels
S-Finite-Measure-Monad.Measure-QuasiBorel-Adjunction
begin

4.1 s-Finiteness

lemma *s-finite-measure-copair-measure*:
assumes *s-finite-measure* M *s-finite-measure* N
shows *s-finite-measure* (*copair-measure* $M \ N$)
proof –
note $[\text{measurable}] = \text{measurable-vimage-Inl}[of \ - \ M \ N] \ \text{measurable-vimage-Inr}[of \ - \ M \ N]$
obtain $M_i \ N_i$ **where** $[\text{measurable-cong}]$:

$\bigwedge i. \text{sets } (M_i i) = \text{sets } M \bigwedge i. \text{finite-measure } (M_i i) \bigwedge A. M A = (\sum i. M_i i A)$
 $\bigwedge i. \text{sets } (N_i i) = \text{sets } N \bigwedge i. \text{finite-measure } (N_i i) \bigwedge A. N A = (\sum i. N_i i A)$
by (*metis* *assms*(1) *assms*(2) *s-finite-measure.finite-measures'*)
thus *?thesis*
by(*auto intro!*: *s-finite-measureI*[**where** $M_i = \lambda i. M_i i \oplus_M N_i i$] *finite-measure-copair-measure*
cong: sets-copair-measure-cong simp: emeasure-copair-measure sum-
inf-add)
qed

lemma *s-finite-measure-coPiM*:
assumes *countable I* $\bigwedge i. i \in I \implies \text{s-finite-measure } (M_i i)$
shows *s-finite-measure (coPiM I Mi)*
proof –
note *measurable-Pair-vimage*[*measurable (raw)*]
consider *finite I | infinite I countable I*
using *assms* **by** *argo*
then show *?thesis*
proof cases
assume *I:finite I*
show *?thesis*
by(*auto intro!*: *s-finite-measure-finite-sumI*[**where** $M_i = \lambda i. \text{distr } (M_i i) (\text{coPiM } I \text{ Mi}) (\text{Pair } i)$
 $\text{and } I = I, OF - \text{s-finite-measure.s-finite-measure-distr}[OF$
 $\text{assms}(2)]$
simp: emeasure-distr emeasure-coPiM-finite I)
next
assume *I:infinite I countable I*
then have [*simp*]: $\bigwedge n. \text{from-nat-into } I \ n \in I$
by (*simp add: from-nat-into infinite-imp-nonempty*)
show *?thesis*
by(*auto intro!*: *s-finite-measure-s-finite-sumI*[**where**
 $M_i = \lambda n. \text{distr } (M_i (\text{from-nat-into } I \ n)) (\text{coPiM } I \text{ Mi}) (\text{Pair } (\text{from-nat-into } I \ n)),$
 $OF - \text{s-finite-measure.s-finite-measure-distr}[OF \ \text{assms}(2)]$
simp: emeasure-distr I emeasure-coPiM-countable-infinite' coPair-inverse-space-unit[**where**
 $I = I$])
qed
qed

4.2 Standardness

lemma *standard-borel-copair-measure*:
assumes *standard-borel M standard-borel N*
shows *standard-borel (M \oplus_M N)*
proof –
obtain *A* **where** $A[\text{measurable}]: A \in \text{sets borel } A \subseteq \{0 < .. < 1 :: \text{real}\}$
 $M \text{ measurable-isomorphic restrict-space borel } A$
by (*meson* *assms*(1) *greaterThanLessThan-borel linorder-not-le not-one-le-zero*
standard-borel.isomorphic-subset-real uncountable-open-interval)

```

then obtain  $f f'$ 
  where  $f$ [measurable]:  $f \in M \rightarrow_M \text{restrict-space borel } A$ 
     $f' \in \text{restrict-space borel } A \rightarrow_M M$ 
     $\bigwedge x. x \in \text{space } M \implies f' (f x) = x \wedge y. y \in A \implies f (f' y) = y$ 
  using measurable-isomorphicD[OF A(3)] unfolding space-restrict-space by
fastforce
obtain  $B$  where  $B$ [measurable]:  $B \in \text{sets borel } B \subseteq \{1 <..<2::\text{real}\}$ 
   $N \text{ measurable-isomorphic restrict-space borel } B$ 
by (metis assms(2) greaterThanLessThan-borel linorder-not-le numeral-le-one-iff
semiring-norm(69) standard-borel.isomorphic-subset-real uncount-
able-open-interval)
then obtain  $g g'$ 
  where  $g$ [measurable]:  $g \in N \rightarrow_M \text{restrict-space borel } B$ 
     $g' \in \text{restrict-space borel } B \rightarrow_M N$ 
     $\bigwedge x. x \in \text{space } N \implies g' (g x) = x$ 
     $\bigwedge y. y \in B \implies g (g' y) = y$ 
  using measurable-isomorphicD[OF B(3)] unfolding space-restrict-space by
fastforce
have  $AB: A \cap B = \{\}$ 
  using  $A B$  by fastforce
have [measurable]:  $f \in M \rightarrow_M \text{restrict-space borel } (A \cup B)$ 
  using  $f(1)$  unfolding measurable-restrict-space2-iff by blast
have [measurable]:  $g \in N \rightarrow_M \text{restrict-space borel } (A \cup B)$ 
  using  $g(1)$  unfolding measurable-restrict-space2-iff by blast

have iso: restrict-space borel  $(A \cup B)$  measurable-isomorphic  $M \oplus_M N$ 
proof (safe intro!: measurable-isomorphic-byWitness)
  show case-sum  $f g \in M \oplus_M N \rightarrow_M \text{restrict-space borel } (A \cup B)$ 
  by (auto intro!: measurable-copair-Inl-Inr)
  show  $(\lambda r. \text{if } r \in A \text{ then } \text{Inl } (f' r) \text{ else if } r \in B \text{ then } \text{Inr } (g' r) \text{ else undefined})$ 
     $\in \text{restrict-space borel } (A \cup B) \rightarrow_M M \oplus_M N$  (is  $?f \in -$ )
proof -
  have 1:
    restrict-space (restrict-space borel  $(A \cup B)$ )  $\{r. r \in A\} = \text{restrict-space}$ 
borel  $A$ 
    restrict-space (restrict-space borel  $(A \cup B)$ )  $\{r. r \notin A\} = \text{restrict-space}$ 
borel  $B$ 
    restrict-space (restrict-space borel  $B$ )  $\{x. x \in B\} = \text{restrict-space borel } B$ 
    restrict-space (restrict-space borel  $B$ )  $\{x. x \notin B\} = \text{count-space } \{\}$ 
  using  $AB$  by (auto simp: restrict-restrict-space
    intro!: arg-cong[where f=restrict-space borel] space-empty)
  have 2:  $\{r \in \text{space } (\text{restrict-space borel } (A \cup B)). r \in A\} = A$ 
     $\{x \in \text{space } (\text{restrict-space } (\text{restrict-space borel } (A \cup B)) \{r. r \notin A\}). x$ 
 $\in B\} = B$ 
     $\{x \in \text{space } (\text{restrict-space borel } B). x \in B\} = B$ 
  using  $AB$  by (auto simp: space-restrict-space)
show ?thesis
  by (intro measurable-If-restrict-space-iff[THEN iffD2] conjI)
    (unfold 1 2, simp-all add: sets-restrict-space-iff)

```

```

qed
show  $\bigwedge x. x \in \text{space } (M \oplus_M N) \implies ?f (\text{case-sum } f \ g \ x) = x$ 
   $\bigwedge r. r \in \text{space } (\text{restrict-space borel } (A \cup B)) \implies \text{case-sum } f \ g \ (?f \ r) = r$ 
  using measurable-space[OF f(1)] measurable-space[OF g(1)] AB
  by (auto simp: space-copair-measure f g)
qed
show ?thesis
  by(auto intro!: standard-borel.measurable-isomorphic-standard[OF - iso]
      standard-borel.standard-borel-restrict-space[OF standard-borel-ne.standard-borel])
qed

corollary
shows standard-borel-ne-copair-measure1: standard-borel-ne M  $\implies$  standard-borel
N  $\implies$  standard-borel-ne (M  $\oplus_M$  N)
  and standard-borel-ne-copair-measure2: standard-borel M  $\implies$  standard-borel-ne
N  $\implies$  standard-borel-ne (M  $\oplus_M$  N)
  and standard-borel-ne-copair-measure: standard-borel-ne M  $\implies$  standard-borel-ne
N  $\implies$  standard-borel-ne (M  $\oplus_M$  N)
  by(auto simp: standard-borel-ne-def standard-borel-ne-axioms-def standard-borel-copair-measure
space-copair-measure)

lemma standard-borel-coPiM:
  assumes countable I  $\bigwedge i. i \in I \implies$  standard-borel (Mi i)
  shows standard-borel (coPiM I Mi)
proof -
  let ?I = {i $\in$ I. space (Mi i)  $\neq$  {}}
  have countable-I: countable ?I
    using assms by auto
  define I' where I'  $\equiv$  to-nat-on ?I ' ?I
  define Mn where Mn  $\equiv$   $\lambda n. Mi$  (from-nat-into ?I n)
  have I':countable I'  $\bigwedge n. n \in I' \implies$  space (Mn n)  $\neq$  {}
     $\bigwedge n. n \in I' \implies$  standard-borel-ne (Mn n)
    using countable-I from-nat-into-to-nat-on[OF countable-I] assms(2)
  by(fastforce simp: I'-def Mn-def standard-borel-ne-def standard-borel-ne-axioms-def
simp del: from-nat-into-to-nat-on)+
  have iso1:coPiM I Mi measurable-isomorphic coPiM I' Mn
  proof(safe intro!: measurable-isomorphic-byWitness[where f= $\lambda(i,x). (to-nat-on$ 
?I i, x)
      and g= $\lambda(n,x). (from-nat-into ?I n, x)$ ])
    show ( $\lambda(i, x). (to-nat-on ?I i, x) \in \text{coPiM } I \text{ } Mi \rightarrow_M \text{coPiM } I' \text{ } Mn$ )
  proof(rule measurable-coPiM2)
    fix i
    assume i:i  $\in$  I
    show Pair (to-nat-on ?I i)  $\in$  Mi i  $\rightarrow_M$  coPiM I' Mn
  proof(cases space (Mi i) = {})
    assume space (Mi i)  $\neq$  {}
    then show ?thesis
      by(intro measurable-compose[OF - measurable-Pair-coPiM[where I=I']]
        (use I'-def i countable-I Mn-def in auto))
  qed
  qed
  qed

```

```

    qed(simp add: measurable-def)
  qed
  show  $(\lambda(n,x). (from\text{-}nat\text{-}into\ ?I\ n, x)) \in coPiM\ I'\ Mn \rightarrow_M coPiM\ I\ Mi$ 
  proof(rule measurable-coPiM2)
    fix n
    assume  $n \in I'$ 
    show Pair (from-nat-into ?I n)  $\in Mn\ n \rightarrow_M coPiM\ I\ Mi$ 
      by (metis (no-types, lifting) Mn-def I'-def  $\langle n \in I' \rangle$  emptyE empty-is-image
          from-nat-into measurable-Pair-coPiM mem-Collect-eq)
    qed
  qed(auto intro!: from-nat-into-to-nat-on to-nat-on-from-nat-into simp: space-coPiM
    I'-def countable-I)
  have  $\exists A. A \in sets\ borel \wedge A \subseteq \{real\ n <.. real\ n + 1\} \wedge Mn\ n\ measurable\text{-}isomorphic\ (restrict\text{-}space\ borel\ A)$ 
    if  $n:n \in I'$  for n
  using standard-borel.isomorphic-subset-real[OF
    standard-borel-ne.standard-borel[OF I'(3)[OF n]], of  $\{real\ n <.. real\ n + 1\}$ ]
    uncountable-half-open-interval-2[of real n real n + 1]
  by fastforce
  then obtain An'
  where An':  $\bigwedge n. n \in I' \implies An'\ n \in sets\ borel$ 
     $\bigwedge n. n \in I' \implies An'\ n \subseteq \{real\ n <.. real\ n + 1\}$ 
     $\bigwedge n. n \in I' \implies Mn\ n\ measurable\text{-}isomorphic\ (restrict\text{-}space\ borel\ (An'\ n))$ 
  by metis
  define An where  $An \equiv \lambda n. if\ n \in I'\ then\ An'\ n\ else\ \{real\ n + 1\}$ 
  have An[measurable]:  $\bigwedge n. An\ n \in sets\ borel$ 
     $\bigwedge n. An\ n \subseteq \{real\ n <.. real\ n + 1\}$ 
     $\bigwedge n. n \in I' \implies Mn\ n\ measurable\text{-}isomorphic\ (restrict\text{-}space\ borel\ (An\ n))$ 
  using An' by(auto simp: An-def)
  hence disj-An: disjoint-family An
  unfolding disjoint-family-on-def
  by safe (metis (no-types, opaque-lifting) greaterThanAtMost-iff less-le nat-less-real-le
    not-less order-trans subset-eq)
  obtain fn gn'
  where fg:  $\bigwedge n. n \in I' \implies fn\ n \in Mn\ n \rightarrow_M restrict\text{-}space\ borel\ (An\ n)$ 
     $\bigwedge n. n \in I' \implies gn'\ n \in restrict\text{-}space\ borel\ (An\ n) \rightarrow_M Mn\ n$ 
     $\bigwedge n\ x. n \in I' \implies x \in space\ (Mn\ n) \implies gn'\ n\ (fn\ n\ x) = x$ 
     $\bigwedge n\ r. n \in I' \implies r \in space\ (restrict\text{-}space\ borel\ (An\ n)) \implies fn\ n\ (gn'\ n\ r) = r$ 
  using measurable-isomorphicD[OF An(3)] by metis
  define gn where  $gn \equiv (\lambda n\ r. if\ r \in An\ n\ then\ gn'\ n\ r\ else\ (SOME\ x. x \in space\ (Mn\ n)))$ 
  have gn-meas[measurable]:  $gn\ n \in borel \rightarrow_M Mn\ n$  if  $n:n \in I'$  for n
  unfolding gn-def by(rule measurable-restrict-space-iff[THEN iffD1, OF - -
    fg(2)[OF n]])
    (auto simp add: I'(2) some-in-eq that)
  have fg':  $\bigwedge n\ x. n \in I' \implies x \in space\ (Mn\ n) \implies gn\ n\ (fn\ n\ x) = x$ 

```

```

       $\bigwedge n r. n \in I' \implies r \in An\ n \implies fn\ n\ (gn\ n\ r) = r$ 
    using fg measurable-space[OF fg(1)] by(auto simp: gn-def)
    have fn[measurable]:fn n ∈ Mn n →M restrict-space borel (⋃ n∈I'. An n) if n:n
    ∈ I' for n
      using measurable-restrict-space2-iff[THEN iffD1,OF fg(1)[OF n]]
      by(auto intro!: measurable-restrict-space2 n)
    let ?f = λ(n,x). fn n x and ?g = λr. (nat ⌈r⌉ - 1, gn (nat ⌈r⌉ - 1) r)
    have iso2:coPiM I' Mn measurable-isomorphic restrict-space borel (⋃ n∈I'. An
    n)
    proof(safe intro!: measurable-isomorphic-byWitness)
      show ?f ∈ coPiM I' Mn →M restrict-space borel (⋃ n∈I'. An n)
        by(auto intro!: measurable-coPiM2)
      next
        show ?g ∈ restrict-space borel (⋃ n∈I'. An n) →M coPiM I' Mn
        proof(safe intro!: measurable-coPiM1)
          have 1:restrict-space borel (⋃ (An ' I')) →M count-space I'
            = restrict-space borel (⋃ (An ' I')) →M restrict-space (count-space
            UNIV) I'
            by (simp add: restrict-count-space)
          show (λx. nat ⌈x⌉ - 1) ∈ restrict-space borel (⋃ (An ' I')) →M count-space
            I'
            unfolding 1
            proof(safe intro!: measurable-restrict-space3)
              fix n r
              assume n:n ∈ I' r ∈ An n
              then have real n < r r ≤ real n + 1
                using An(2) by fastforce+
              thus nat ⌈r⌉ - 1 ∈ I'
                by (metis n(1) add.commute diff-Suc-1 le-SucE nat-ceiling-le-eq not-less
                of-nat-Suc)
              qed simp
            qed(auto simp: measurable-restrict-space1)
          next
            fix n x
            assume (n,x)∈space (coPiM I' Mn)
            then have nx:n ∈ I' x ∈ space (Mn n)
              by(auto simp: space-coPiM)
            have 1:nat ⌈?f (n,x)⌉ = n + 1
              using measurable-space[OF fg(1)[OF nx(1)] nx(2)] An(2)[of n]
              by simp
            (metis add.commute greaterThanAtMost-iff le-SucE nat-ceiling-le-eq not-less
            of-nat-Suc subset-eq)
            show ?g (?f (n,x)) = (n,x)
              unfolding 1 using fg'(1)[OF nx] by simp
          next
            fix y
            assume y ∈ space (restrict-space borel (⋃ (An ' I)))
            then obtain n where n: n ∈ I' y ∈ An n
              by auto

```

```

then have [simp]: nat [y] = n + 1
using An(2)[of n]
by simp (metis add.commute greaterThanAtMost-iff le-SucE nat-ceiling-le-eq
not-less-of-nat-Suc subset-eq)
show ?f (?g y) = y
using fg'(2)[OF n(1)] n(2) by auto
qed
have standard-borel (restrict-space borel (⋃ (An ' I)))
by(auto intro!: standard-borel-ne.standard-borel[THEN standard-borel.standard-borel-restrict-space])
with iso1 iso2 show ?thesis
by (meson measurable-isomorphic-sym standard-borel.measurable-isomorphic-standard)
qed

```

```

lemma standard-borel-ne-coPiM:
assumes countable I  $\wedge i. i \in I \implies$  standard-borel (Mi i)
and  $i \in I$  space (Mi i)  $\neq \{\}$ 
shows standard-borel-ne (coPiM I Mi)
proof -
have space (coPiM I Mi)  $\neq \{\}$ 
using assms(3) assms(4) space-coPiM by fastforce
thus ?thesis
by(auto intro!: standard-borel-coPiM assms simp: standard-borel-ne-def stan-
dard-borel-ne-axioms-def)
qed

```

4.3 Relationships with Quasi-Borel Spaces

Proposition19(3) [1]

```

lemma r-preserve-copair: measure-to-qbs (copair-measure M N) = measure-to-qbs
M  $\oplus_Q$  measure-to-qbs N
proof(safe intro!: qbs-eqI)
fix  $\alpha$ 
assume  $\alpha \in$  qbs-Mx (measure-to-qbs (M  $\oplus_M$  N))
then have a[measurable]:  $\alpha \in$  borel  $\rightarrow_M$  M  $\oplus_M$  N
by(simp add: qbs-Mx-R)
have s[measurable]:  $\alpha - ' Inr ' space N \in$  sets borel  $\alpha - ' Inl ' space M \in$  sets
borel
by(auto intro!: measurable-sets-borel[OF a])
consider  $\alpha - ' Inl ' space M \cap space borel = space borel$ 
|  $\alpha - ' Inr ' (space N) \cap space borel = space borel$ 
|  $\alpha - ' Inl ' space M \cap space borel \subset space borel$ 
|  $\alpha - ' Inr ' (space N) \cap space borel \subset space borel$ 
by blast
then show  $\alpha \in$  qbs-Mx (measure-to-qbs M  $\oplus_Q$  measure-to-qbs N)
proof cases
assume 1:  $\alpha - ' Inl ' space M \cap space borel = space borel$ 
then obtain f' where f'  $\in$  borel  $\rightarrow_M$  M  $\wedge x. x \in space borel \implies \alpha x = Inl$ 
(f' x)
using measurable-copair-dest1[OF a] by blast

```

```

thus ?thesis
  using 1 by(auto simp: copair-qbs-Mx copair-qbs-Mx-def qbs-Mx-R
    intro!: bexI[where x=α - 'Inr ' space N] bexI[where x=f'])
next
  assume 2:α - 'Inr ' space N ∩ space borel = space borel
  then obtain f' where f' ∈ borel →M N ∧x. x ∈ space borel ⇒ α x = Inr
(f' x)
  using measurable-copair-dest2[OF a] by blast
  thus ?thesis
  using 2 by(auto simp: copair-qbs-Mx copair-qbs-Mx-def qbs-Mx-R
    intro!: bexI[where x=α - 'Inr ' space N] bexI[where x=f'])
next
  case 3
  then obtain f' f''
  where f[measurable]:f' ∈ borel →M M
    f'' ∈ borel →M N
    ∧x. x ∈ space borel ⇒ x ∈ α - 'Inl ' space M ⇒ α x =
Inl (f' x)
    ∧x. x ∈ space borel ⇒ x ∉ α - 'Inl ' space M ⇒ α x =
Inr (f'' x)
  using measurable-copair-dest3[OF a] by metis
  moreover have α - 'Inl ' space M ≠ UNIV α - 'Inl ' space M ≠ {}
  using 3 measurable-space[OF a] by(fastforce simp: space-copair-measure)+
  ultimately show ?thesis
  by(auto simp: copair-qbs-Mx copair-qbs-Mx-def qbs-Mx-R simp del: vimage-eq
    intro!: bexI[where x=α - 'Inl ' space M] bexI[where x=f'] bexI[where
x=f''])
  qed
qed(auto simp: qbs-Mx-R copair-qbs-Mx copair-qbs-Mx-def)

lemma r-preserve-coproduct:
  assumes countable I
  shows measure-to-qbs (coPiM I M) = (∏Q i∈I. measure-to-qbs (M i))
proof(safe intro!: qbs-eqI)
  fix α
  assume h:α ∈ qbs-Mx (measure-to-qbs (coPiM I M))
  then obtain a g
  where a ∈ borel →M count-space I
    ∧i. i ∈ I ⇒ space (M i) ≠ {} ⇒ g i ∈ borel →M M i
    α = (λx. (a x, g (a x) x))
  using measurable-coPiM1-elements[OF assms] unfolding qbs-Mx-R by blast
  thus α ∈ qbs-Mx (∏Q i∈I. measure-to-qbs (M i))
  using qbs-Mx-to-X[OF h]
  by(safe intro!: coPiQ-MxI) (auto simp: qbs-Mx-R qbs-space-R space-coPiM)
next
  fix α
  assume α ∈ qbs-Mx (∏Q i∈I. measure-to-qbs (M i))
  then obtain a g where a ∈ borel →M count-space I
    ∧i. i ∈ range a ⇒ g i ∈ borel →M M i α = (λx. (a x, g (a

```

```

x) x))
  unfolding coPiQ-Mx coPiQ-Mx-def qbs-Mx-R by blast
  thus  $\alpha \in$  qbs-Mx (measure-to-qbs (coPiM I M))
  by(auto intro!: measurable-coPiM1' simp: qbs-Mx-R assms)
qed

end

```

References

- [1] C. Heunen, O. Kammar, S. Staton, and H. Yang. A convenient category for higher-order probability theory. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '17*. IEEE Press, 2017.