Ergodic theory in Isabelle

Sebastien Gouezel

Abstract

Ergodic theory is the branch of mathematics that studies the behaviour of measure preserving transformations, in finite or infinite measure. It interacts both with probability theory (mainly through measure theory) and with geometry as a lot of interesting examples are from geometric origin. We implement the first definitions and theorems of ergodic theory, including notably Poincaré recurrence theorem for finite measure preserving systems (together with the notion of conservativity in general), induced maps, Kac's theorem, Birkhoff theorem (arguably the most important theorem in ergodic theory), and variations around it such as conservativity of the corresponding skew product, or Atkinson lemma, and Kingman theorem. Using this material, we formalize completely the proof of the main theorems of [GK15] and [Gou18].

Contents

1 SG Libary complements

theory *SG-Library-Complement*
imports *HOL−Probability.Probability*
begin

In this file are included many statements that were useful to me, but belong rather naturally to existing theories. In a perfect world, some of these statements would get included into these files.

I tried to indicate to which of these classical theories the statements could be added.

lemma *compl-compl-eq-id* [*simp*]:
$UNIV - (UNIV - s) = s$
by *auto*

notation *sym-diff* (**infixl** ‹Δ› *70*)

1.1 Set-Interval.thy

The next two lemmas belong naturally to `Set_Interval.thy`, next to `UN_le_add_shift`. They are not trivially equivalent to the corresponding lemmas with large inequalities, due to the difference when $n = 0$.

lemma *UN-le-eq-Un0-strict*:
  $(\bigcup i<n+1::nat.\ M\ i) = (\bigcup i\in\{1..<n+1\}.\ M\ i) \cup M\ 0$ (**is** $?A = ?B$)
proof
  show $?A \subseteq ?B$
  proof
    fix x **assume** $x \in ?A$
    then obtain i **where** i: $i<n+1\ x \in M\ i$ **by** *auto*
    show $x \in ?B$
    proof(*cases i*)
      case *0* **with** i **show** *?thesis* **by** *simp*
    next
      case (*Suc j*) **with** i **show** *?thesis* **by** *auto*
    qed
  qed
qed (*auto*)

I use repeatedly this one, but I could not find it directly

lemma *union-insert-0*:
  $(\bigcup n::nat.\ A\ n) = A\ 0 \cup (\bigcup n\in\{1..\}.\ A\ n)$
by (*metis UN-insert Un-insert-left sup-bot.left-neutral One-nat-def atLeast-0 atLeast-Suc-greaterThan ivl-disj-un-singleton*(*1*))

Next one could be close to `sum.nat_group`

lemma *sum-arith-progression*:
  $(\sum r<(N::nat).\ (\sum i<a.\ f\ (i*N+r))) = (\sum j<a*N.\ f\ j)$
proof −
  have $*$: $(\sum r<N.\ f\ (i*N+r)) = (\sum\ j \in \{i*N..<i*N + N\}.\ f\ j)$ **for** i
   by (*rule sum.reindex-bij-betw*, *rule bij-betw-byWitness*[**where** $?f' = \lambda r.\ r-i*N$], *auto*)

  have $(\sum r<N.\ (\sum i<a.\ f\ (i*N+r))) = (\sum i<a.\ (\sum r<N.\ f\ (i*N+r)))$
    using *sum.swap* **by** *auto*
  also have ... $= (\sum i<a.\ (\sum\ j \in \{i*N..<i*N + N\}.\ f\ j))$
    using $*$ **by** *auto*
  also have ... $= (\sum j<a*N.\ f\ j)$
    by (*rule sum.nat-group*)
  finally show *?thesis* **by** *simp*
qed

1.2 Miscellanous basic results

lemma *ind-from-1* [*case-names 1 Suc*, *consumes 1*]:
  assumes $n > 0$

assumes *P 1*
and $\bigwedge n.\ n > 0 \Longrightarrow P\ n \Longrightarrow P\ (Suc\ n)$
shows *P n*
proof −
have $(n = 0) \vee P\ n$
proof (*induction n*)
case *0* **then show** *?case* **by** *auto*
next
case (*Suc k*)
consider $Suc\ k = 1 \mid Suc\ k > 1$ **by** *linarith*
then show *?case*
apply (*cases*) **using** *assms Suc.IH* **by** *auto*
qed
then show *?thesis* **using** ‹$n > 0$› **by** *auto*
qed

This lemma is certainly available somewhere, but I couldn't locate it

lemma *tends-to-real-e*:
fixes *u*::*nat* ⇒ *real*
assumes $u \longrightarrow l$ $e>0$
shows $\exists N.\ \forall n>N.\ abs(u\ n - l) < e$
by (*metis assms dist-real-def le-less lim-sequentially*)

lemma *nat-mod-cong*:
assumes $a = b+(c::nat)$
$a\ mod\ n = b\ mod\ n$
shows $c\ mod\ n = 0$
proof −
let $?k = a\ mod\ n$
obtain *a1* **where** $a = a1*n + ?k$ **by** (*metis div-mult-mod-eq*)
moreover obtain *b1* **where** $b = b1*n + ?k$ **using** *assms*(*2*) **by** (*metis div-mult-mod-eq*)
ultimately have $a1 * n + ?k = b1 * n + ?k + c$ **using** *assms*(*1*) **by** *arith*
then have $c = (a1 - b1) * n$ **by** (*simp add*: *diff-mult-distrib*)
then show *?thesis* **by** *simp*
qed

lemma *funpow-add'*: $(f\ \widehat{}\widehat{}\ (m + n))\ x = (f\ \widehat{}\widehat{}\ m)\ ((f\ \widehat{}\widehat{}\ n)\ x)$
by (*simp add*: *funpow-add*)

The next two lemmas are not directly equivalent, since *f* might not be injective.

lemma *abs-Max-sum*:
fixes *A*::*real set*
assumes *finite A* $A \neq \{\}$
shows $abs(Max\ A) \leq (\sum a\in A.\ abs(a))$
by (*simp add*: *assms member-le-sum*)

lemma *abs-Max-sum2*:
fixes *f*::*-* ⇒ *real*

assumes *finite A A* $\neq$ {}
shows *abs*(*Max* (*f*'*A*)) $\leq$ ($\sum$ *a*$\in$*A*. *abs*(*f a*))
using *assms* **by** (*induct rule*: *finite-ne-induct*, *auto*)

1.3 Conditionally-Complete-Lattices.thy

lemma *mono-cInf*:
fixes *f* :: *'a*::*conditionally-complete-lattice* $\Rightarrow$ *'b*::*conditionally-complete-lattice*
assumes *mono f A* $\neq$ {} *bdd-below A*
shows *f*(*Inf A*) $\leq$ *Inf* (*f*'*A*)
using *assms* **by** (*simp add*: *cINF-greatest cInf-lower monoD*)

lemma *mono-bij-cInf*:
fixes *f* :: *'a*::*conditionally-complete-linorder* $\Rightarrow$ *'b*::*conditionally-complete-linorder*
assumes *mono f bij f A* $\neq$ {} *bdd-below A*
shows *f* (*Inf A*) = *Inf* (*f*'*A*)
proof −
have (*inv f*) (*Inf* (*f*'*A*)) $\leq$ *Inf* ((*inv f*)'(*f*'*A*))
apply (*rule cInf-greatest*, *auto simp add*: *assms*(*3*))
using *mono-inv*[*OF assms*(*1*) *assms*(*2*)] *assms* **by** (*simp add*: *mono-def bdd-below-image-mono cInf-lower*)
then have *Inf* (*f*'*A*) $\leq$ *f* (*Inf* ((*inv f*)'(*f*'*A*)))
by (*metis* (*no-types*, *lifting*) *assms*(*1*) *assms*(*2*) *mono-def bij-inv-eq-iff*)
also have ... = *f*(*Inf A*)
using *assms* **by** (*simp add*: *bij-is-inj*)
finally show *?thesis* **using** *mono-cInf*[*OF assms*(*1*) *assms*(*3*) *assms*(*4*)] **by** *auto*
qed

1.4 Topological-spaces.thy

lemma *open-less-abs* [*simp*]:
open {*x*. (*C*::*real*) < *abs x*}
proof −
have ∗: {*x*. *C* < *abs x*} = *abs*−'{*C*<..} **by** *auto*
show *?thesis* **unfolding** ∗ **by** (*auto intro*!: *continuous-intros*)
qed

lemma *closed-le-abs* [*simp*]:
closed {*x*. (*C*::*real*) $\leq$ *abs x*}
proof −
have ∗: {*x*. *C* $\leq$ $|x|$} = *abs*−'{*C*..} **by** *auto*
show *?thesis* **unfolding** ∗ **by** (*auto intro*!: *continuous-intros*)
qed

The next statements come from the same statements for true subsequences

lemma *eventually-weak-subseq*:
fixes *u*::*nat* $\Rightarrow$ *nat*
assumes (λ*n*. *real*(*u n*)) $\longrightarrow$ ∞ *eventually P sequentially*
shows *eventually* (λ*n*. *P* (*u n*)) *sequentially*

proof −
obtain *N* **where** ∗: $\forall n \geq N.\ P\ n$ **using** *assms*(*2*) **unfolding** *eventually-sequentially* **by** *auto*
obtain *M* **where** $\forall m \geq M.\ ereal(u\ m) \geq N$ **using** *assms*(*1*) **by** (*meson Lim-PInfty*)
then have $\bigwedge m.\ m \geq M \Longrightarrow u\ m \geq N$ **by** *auto*
then have $\bigwedge m.\ m \geq M \Longrightarrow P(u\ m)$ **using** ‹$\forall n \geq N.\ P\ n$› **by** *simp*
then show *?thesis* **unfolding** *eventually-sequentially* **by** *auto*
qed

lemma *filterlim-weak-subseq*:
fixes *u*::*nat* ⇒ *nat*
assumes $(\lambda n.\ real(u\ n)) \longrightarrow \infty$
shows *LIM n sequentially. u n*:> *at-top*
unfolding *filterlim-iff* **by** (*metis assms eventually-weak-subseq*)

lemma *limit-along-weak-subseq*:
fixes *u*::*nat* ⇒ *nat* **and** *v*::*nat* ⇒ -
assumes $(\lambda n.\ real(u\ n)) \longrightarrow \infty$ $v \longrightarrow l$
shows $(\lambda\ n.\ v(u\ n)) \longrightarrow l$
using *filterlim-compose*[*of v*, *OF* - *filterlim-weak-subseq*] *assms* **by** *auto*

lemma *frontier-indist-le*:
assumes $x \in frontier\ \{y.\ infdist\ y\ S \leq r\}$
shows $infdist\ x\ S = r$
proof −
have $infdist\ x\ S = r$ **if** *H*: $\forall e>0.\ (\exists y.\ infdist\ y\ S \leq r \wedge dist\ x\ y < e) \wedge (\exists z.\ \neg infdist\ z\ S \leq r \wedge dist\ x\ z < e)$
proof −
have $infdist\ x\ S < r + e$ **if** $e > 0$ **for** *e*
proof −
obtain *y* **where** $infdist\ y\ S \leq r$ $dist\ x\ y < e$
using *H* ‹$e > 0$› **by** *blast*
then show *?thesis*
by (*metis add.commute add-mono-thms-linordered-field*(*3*) *infdist-triangle le-less-trans*)
qed
then have *A*: $infdist\ x\ S \leq r$
by (*meson field-le-epsilon order.order-iff-strict*)
have $r < infdist\ x\ S + e$ **if** $e > 0$ **for** *e*
proof −
obtain *y* **where** $\neg(infdist\ y\ S \leq r)$ $dist\ x\ y < e$
using *H* ‹$e > 0$› **by** *blast*
then have $r < infdist\ y\ S$ **by** *auto*
also have ... $\leq infdist\ x\ S + dist\ y\ x$
by (*rule infdist-triangle*)
finally show *?thesis* **using** ‹$dist\ x\ y < e$›
by (*simp add*: *dist-commute*)
qed
then have *B*: $r \leq infdist\ x\ S$

by (*meson field-le-epsilon order.order-iff-strict*)
show *?thesis* **using** *A B* **by** *auto*
qed
then show *?thesis*
using *assms* **unfolding** *frontier-straddle* **by** *auto*
qed

1.5 Limits

The next lemmas are not very natural, but I needed them several times

lemma *tendsto-shift-1-over-n* [*tendsto-intros*]:
fixes *f*::*nat* ⇒ *real*
assumes (λ*n*. *f n* / *n*) ⟶ *l*
shows (λ*n*. *f* (*n*+*k*) / *n*) ⟶ *l*
proof −
have (*1*+*k*∗(*1*/*n*))∗ (*f*(*n*+*k*)/(*n*+*k*)) = *f*(*n*+*k*)/*n* **if** *n*>*0* **for** *n* **using** *that* **by** (*auto simp add*: *divide-simps*)
with *eventually-mono*[*OF eventually-gt-at-top*[*of 0*::*nat*] *this*]
have *eventually* (λ*n*.(*1*+*k*∗(*1*/*n*))∗ (*f*(*n*+*k*)/(*n*+*k*)) = *f*(*n*+*k*)/*n*) *sequentially*
by *auto*
moreover have (λ*n*. (*1*+*k*∗(*1*/*n*))∗ (*f*(*n*+*k*)/(*n*+*k*))) ⟶ (*1*+*real k*∗*0*) ∗ *l*
by (*intro tendsto-intros LIMSEQ-ignore-initial-segment assms*)
ultimately show *?thesis* **using** *Lim-transform-eventually* **by** *auto*
qed

lemma *tendsto-shift-1-over-n′* [*tendsto-intros*]:
fixes *f*::*nat* ⇒ *real*
assumes (λ*n*. *f n* / *n*) ⟶ *l*
shows (λ*n*. *f* (*n*−*k*) / *n*) ⟶ *l*
proof −
have (*1*−*k*∗(*1*/(*n*+*k*)))∗ (*f n*/ *n*) = *f n*/(*n*+*k*) **if** *n*>*0* **for** *n* **using** *that* **by** (*auto simp add*: *divide-simps*)
with *eventually-mono*[*OF eventually-gt-at-top*[*of 0*::*nat*] *this*]
have *eventually* (λ*n*. (*1*−*k*∗(*1*/(*n*+*k*)))∗ (*f n*/ *n*) = *f n*/(*n*+*k*)) *sequentially*
by *auto*
moreover have (λ*n*. (*1*−*k*∗(*1*/(*n*+*k*)))∗ (*f n*/ *n*)) ⟶ (*1*−*real k*∗*0*) ∗ *l*
by (*intro tendsto-intros assms LIMSEQ-ignore-initial-segment*)
ultimately have (λ*n*. *f n* / (*n*+*k*)) ⟶ *l* **using** *Lim-transform-eventually* **by** *auto*
then have *a*: (λ*n*. *f*(*n*−*k*)/(*n*−*k*+*k*)) ⟶ *l* **using** *seq-offset-neg* **by** *auto*

have *f*(*n*−*k*)/(*n*−*k*+*k*) = *f*(*n*−*k*)/*n* **if** *n*>*k* **for** *n*
using *that* **by** *auto*
with *eventually-mono*[*OF eventually-gt-at-top*[*of k*] *this*]
have *eventually* (λ*n*. *f*(*n*−*k*)/(*n*−*k*+*k*) = *f*(*n*−*k*)/*n*) *sequentially*
by *auto*
with *Lim-transform-eventually*[*OF a this*]
show *?thesis* **by** *auto*
qed

declare *LIMSEQ-realpow-zero* [*tendsto-intros*]

1.6 Topology-Euclidean-Space

A (more usable) variation around `continuous_on_closure_sequentially`. The assumption that the spaces are metric spaces is definitely too strong, but sufficient for most applications.

lemma *continuous-on-closure-sequentially'*:
  fixes *f*::*'a*::*metric-space* ⇒ *'b*::*metric-space*
  assumes *continuous-on* (*closure C*) *f*
    ⋀(*n*::*nat*). *u n* ∈ *C*
    u ⟶ *l*
  shows (λ*n*. *f* (*u n*)) ⟶ *f l*
proof −
  have *l* ∈ *closure C* **unfolding** *closure-sequential* **using** *assms* **by** *auto*
  then show *?thesis*
   using ‹*continuous-on* (*closure C*) *f*› **unfolding** *comp-def continuous-on-closure-sequentially*
    using *assms* **by** *auto*
qed

1.7 Convexity

lemma *convex-on-mean-ineq*:
  fixes *f*::*real* ⇒ *real*
  assumes *convex-on A f x* ∈ *A y* ∈ *A*
  shows *f* ((*x*+*y*)/*2*) ≤ (*f x* + *f y*) / *2*
using *convex-onD*[*OF assms*(*1*), *of 1/2 x y*] **using** *assms* **by** (*auto simp add*: *divide-simps*)

lemma *convex-on-closure*:
  fixes *C* :: *'a*::*real-normed-vector set*
  assumes *convex C*
    convex-on C f
    continuous-on (*closure C*) *f*
  shows *convex-on* (*closure C*) *f*
proof (*rule convex-onI*)
  show *convex* (*closure C*)
   by (*simp add*: ‹*convex C*›)
  fix *x y*::*'a* **and** *t*::*real*
  assume *x* ∈ *closure C y* ∈ *closure C 0* < *t t* < *1*
  obtain *u v*::*nat* ⇒ *'a* **where** ∗: ⋀*n*. *u n* ∈ *C u* ⟶ *x*
    ⋀*n*. *v n* ∈ *C v* ⟶ *y*
   using ‹*x* ∈ *closure C*› ‹*y* ∈ *closure C*› **unfolding** *closure-sequential* **by** *blast*
  define *w* **where** *w* = (λ*n*. (*1*−*t*) $*_R$ (*u n*) + *t* $*_R$ (*v n*))
  have *w n* ∈ *C* **for** *n*
   using ‹*0* < *t*› ‹*t*< *1*› *convexD*[*OF* ‹*convex C*› ∗(*1*)[*of n*] ∗(*3*)[*of n*]] **unfolding** *w-def* **by** *auto*
  have *w* ⟶ ((*1*−*t*) $*_R$ *x* + *t* $*_R$ *y*)

unfolding *w-def* **using** $*(2)$ $*(4)$ **by** (*intro tendsto-intros*)

have $*$: $f(w\ n) \le (1-t) * f(u\ n) + t * f\ (v\ n)$ **for** n
using $*(1)$ $*(3)$ ‹*convex-on C f*› ‹$0<t$› ‹$t<1$› *less-imp-le* **unfolding** *w-def* *convex-on-alt* **by** (*simp add*: *add.commute*)
have *i*: $(\lambda n.\ f\ (w\ n)) \longrightarrow f\ ((1-t) *_R x + t *_R y)$
by (*rule continuous-on-closure-sequentially*′[*OF assms*(3) ‹$\bigwedge n.\ w\ n \in C$› ‹$w \longrightarrow ((1-t) *_R x + t *_R y)$›])
have *ii*: $(\lambda n.\ (1-t) * f(u\ n) + t * f\ (v\ n)) \longrightarrow (1-t) * f\ x + t * f\ y$
apply (*intro tendsto-intros*)
apply (*rule continuous-on-closure-sequentially*′[*OF assms*(3) ‹$\bigwedge n.\ u\ n \in C$› ‹$u \longrightarrow x$›])
apply (*rule continuous-on-closure-sequentially*′[*OF assms*(3) ‹$\bigwedge n.\ v\ n \in C$› ‹$v \longrightarrow y$›])
done
show $f\ ((1 - t) *_R x + t *_R y) \le (1 - t) * f\ x + t * f\ y$
apply (*rule LIMSEQ-le*[*OF i ii*]) **using** $*$ **by** *auto*
qed

lemma *convex-on-norm* [*simp*]:
convex-on UNIV (λ(*x*::′*a*::*real-normed-vector*). *norm x*)
using *convex-on-dist*[*of UNIV 0*::′*a*] **by** *auto*

lemma *continuous-abs-powr* [*continuous-intros*]:
assumes $p > 0$
shows *continuous-on UNIV* (λ(*x*::*real*). $|x|$ *powr p*)
apply (*rule continuous-on-powr*′) **using** *assms* **by** (*auto intro*: *continuous-intros*)

lemma *continuous-mult-sgn* [*continuous-intros*]:
fixes *f*::*real* ⇒ *real*
assumes *continuous-on UNIV f* $f\ 0 = 0$
shows *continuous-on UNIV* ($\lambda x.\ sgn\ x * f\ x$)
proof −
have $*$: *continuous-on* {*0*..} ($\lambda x.\ sgn\ x * f\ x$)
apply (*subst continuous-on-cong*[*of* {*0*..} {*0*..} - *f*], *auto simp add*: *sgn-real-def assms*(2))
by (*rule continuous-on-subset*[*OF assms*(1)], *auto*)
have $**$: *continuous-on* {..*0*} ($\lambda x.\ sgn\ x * f\ x$)
apply (*subst continuous-on-cong*[*of* {..*0*} {..*0*} - $\lambda x.\ -f\ x$], *auto simp add*: *sgn-real-def assms*(2))
by (*rule continuous-on-subset*[*of UNIV*], *auto simp add*: *assms intro*!: *continuous-intros*)
show *?thesis*
using *continuous-on-closed-Un*[*OF* - - $*$ $**$] **apply** (*auto intro*: *continuous-intros*)
using *continuous-on-subset* **by** *fastforce*
qed

lemma *DERIV-abs-powr* [*derivative-intros*]:
assumes $p > (1::real)$

shows *DERIV* (λx. |x| *powr* p) x :> p ∗ *sgn* x ∗ |x| *powr* (p − 1)
proof −
consider x = 0 | x>0 | x < 0 **by** *linarith*
then show *?thesis*
proof (*cases*)
case *1*
have *continuous-on UNIV* (λx. *sgn* x ∗ |x| *powr* (p − 1))
by (*auto simp add*: *assms intro*!:*continuous-intros*)
then have (λh. *sgn* h ∗ |h| *powr* (p−1)) −0→ (λh. *sgn* h ∗ |h| *powr* (p−1)) 0
using *continuous-on-def* **by** *blast*
moreover have |h| *powr* p / h = *sgn* h ∗ |h| *powr* (p−1) **for** h
proof −
have |h| *powr* p / h = *sgn* h ∗ |h| *powr* p / |h|
by (*auto simp add*: *algebra-simps divide-simps sgn-real-def*)
also have ... = *sgn* h ∗ |h| *powr* (p−1)
using *assms* **apply** (*cases* h = 0) **apply** (*auto*)
by (*metis abs-ge-zero powr-diff* [*symmetric*] *powr-one-gt-zero-iff times-divide-eq-right*)
finally show *?thesis* **by** *simp*
qed
ultimately have (λh. |h| *powr* p / h) −0→ 0 **by** *auto*
then show *?thesis* **unfolding** *DERIV-def* **by** (*auto simp add*: ‹x = 0›)
next
case *2*
have ∗: ∀$_F$ y *in nhds* x. |y| *powr* p = y *powr* p
unfolding *eventually-nhds* **apply** (*rule exI*[*of* - {0<..}]) **using** ‹x > 0› **by** *auto*
show *?thesis*
apply (*subst DERIV-cong-ev*[*of* - x - (λx. x *powr* p) - p ∗ x *powr* (p−1)])
using ‹x > 0› **by** (*auto simp add*: ∗ *has-real-derivative-powr*)
next
case *3*
have ∗: ∀$_F$ y *in nhds* x. |y| *powr* p = (−y) *powr* p
unfolding *eventually-nhds* **apply** (*rule exI*[*of* - {..<0}]) **using** ‹x < 0› **by** *auto*
show *?thesis*
apply (*subst DERIV-cong-ev*[*of* - x - (λx. (−x) *powr* p) - p ∗ (− x) *powr* (p − *real* 1) ∗ − 1])
using ‹x < 0› **apply** (*simp*, *simp add*: ∗, *simp*)
apply (*rule DERIV-fun-powr*[*of* λy. −y −1 x p]) **using** ‹x < 0› **by** (*auto simp add*: *derivative-intros*)
qed
qed

lemma *convex-abs-powr*:
assumes p ≥ 1
shows *convex-on UNIV* (λx::*real*. |x| *powr* p)
proof (*cases* p = 1)
case *True*
have *convex-on UNIV* (λx::*real*. *norm* x)

```
    by (rule convex-on-norm)
  moreover have |x| powr p = norm x for x using True by auto
  ultimately show ?thesis by simp
next
  case False
  then have p > 1 using assms by auto
  define g where g = (λx::real. p * sgn x * |x| powr (p − 1))
  have *: DERIV (λx. |x| powr p) x :> g x for x
    unfolding g-def using ‹p>1› by (intro derivative-intros)
  have **: g x ≤ g y if x ≤ y for x y
  proof −
    consider x ≥ 0 ∧ y ≥ 0 | x ≤ 0 ∧ y ≤ 0 | x < 0 ∧ y > 0 using ‹x ≤ y› by
linarith
    then show ?thesis
    proof (cases)
      case 1
       then show ?thesis unfolding g-def sgn-real-def using ‹p>1› ‹x ≤ y› by
(auto simp add: powr-mono2)
    next
      case 2
       then show ?thesis unfolding g-def sgn-real-def using ‹p>1› ‹x ≤ y› by
(auto simp add: powr-mono2)
    next
      case 3
      then have g x ≤ 0 0 ≤ g y unfolding g-def using ‹p > 1› by auto
      then show ?thesis by simp
    qed
  qed
  show ?thesis
    apply (rule convex-on-realI[of - - g]) using * ** by auto
qed

lemma convex-powr:
  assumes p ≥ 1
  shows convex-on {0..} (λx::real. x powr p)
proof −
  have convex-on {0..} (λx::real. |x| powr p)
    using convex-abs-powr[OF ‹p ≥ 1›] convex-on-subset by auto
  moreover have |x| powr p = x powr p if x ∈ {0..} for x using that by auto
  ultimately show ?thesis by (simp add: convex-on-def)
qed

lemma convex-powr':
  assumes p > 0 p ≤ 1
  shows convex-on {0..} (λx::real. − (x powr p))
proof −
  have convex-on {0<..} (λx::real. − (x powr p))
    apply (rule convex-on-realI[of - - λx. −p * x powr (p−1)])
    apply (auto intro!:derivative-intros simp add: has-real-derivative-powr)
```

using ‹p > 0› ‹p ≤ 1› **by** (*auto simp add*: *algebra-simps divide-simps powr-mono2'*)
moreover have *continuous-on* {*0*..} (λx::*real*. − (*x powr p*))
by (*rule continuous-on-minus*, *rule continuous-on-powr'*, *auto simp add*: ‹p > 0› *intro*!: *continuous-intros*)
moreover have {(*0*::*real*)..} = *closure* {*0*<..} *convex* {(*0*::*real*)<..} **by** *auto*
ultimately show *?thesis* **using** *convex-on-closure* **by** *metis*
qed

lemma *convex-fx-plus-fy-ineq*:
fixes *f*::*real* ⇒ *real*
assumes *convex-on* {*0*..} *f*
x ≥ *0* *y* ≥ *0* *f 0* = *0*
shows *f x* + *f y* ≤ *f* (*x*+*y*)
proof −
have ∗: *f a* + *f b* ≤ *f* (*a*+*b*) **if** *a* ≥ *0* *b* ≥ *a* **for** *a b*
proof (*cases a* = *0*)
case *False*
then have *a* > *0* *b* > *0* **using** ‹b ≥ a› ‹a ≥ 0› **by** *auto*
have (*f 0* − *f a*) / (*0* − *a*) ≤ (*f 0* − *f* (*a*+*b*))/ (*0* − (*a*+*b*))
apply (*rule convex-on-slope-le*[*OF* ‹*convex-on* {*0*..} *f*›]) **using** ‹a > 0› ‹b > 0› **by** *auto*
also have ... ≤ (*f b* − *f* (*a*+*b*)) / (*b* − (*a*+*b*))
apply (*rule convex-on-slope-le*[*OF* ‹*convex-on* {*0*..} *f*›]) **using** ‹a > 0› ‹b > 0› **by** *auto*
finally show *?thesis*
using ‹a > 0› ‹b > 0› ‹f 0 = 0› **by** (*auto simp add*: *divide-simps algebra-simps*)
qed (*simp add*: ‹f 0 = 0›)
then show *?thesis*
using ‹x ≥ 0› ‹y ≥ 0› **by** (*metis add.commute le-less not-le*)
qed

lemma *x-plus-y-p-le-xp-plus-yp*:
fixes *p x y*::*real*
assumes *p* > *0* *p* ≤ *1* *x* ≥ *0* *y* ≥ *0*
shows (*x* + *y*) *powr p* ≤ *x powr p* + *y powr p*
using *convex-fx-plus-fy-ineq*[*OF convex-powr'*[*OF* ‹p > 0› ‹p ≤ 1›] ‹x ≥ 0› ‹y ≥ 0›] **by** *auto*

1.8 Nonnegative-extended-real.thy

lemma *x-plus-top-ennreal* [*simp*]:
x + ⊤ = (⊤::*ennreal*)
by *simp*

lemma *ennreal-ge-nat-imp-PInf*:
fixes *x*::*ennreal*
assumes ⋀*N*. *x* ≥ *of-nat N*
shows *x* = ∞
using *assms* **apply** (*cases x*, *auto*) **by** (*meson not-less reals-Archimedean2*)

lemma *ennreal-archimedean*:
  assumes $x \neq (\infty::ennreal)$
  shows $\exists\, n::nat.\ x \leq n$
  using *assms ennreal-ge-nat-imp-PInf linear* **by** *blast*

lemma *e2ennreal-mult*:
  fixes *a b*::*ereal*
  assumes $a \geq 0$
  shows $e2ennreal(a * b) = e2ennreal\ a * e2ennreal\ b$
by (*metis assms e2ennreal-neg eq-onp-same-args ereal-mult-le-0-iff linear times-ennreal.abs-eq*)

lemma *e2ennreal-mult'*:
  fixes *a b*::*ereal*
  assumes $b \geq 0$
  shows $e2ennreal(a * b) = e2ennreal\ a * e2ennreal\ b$
using *e2ennreal-mult*[*OF assms*, *of a*] **by** (*simp add*: *mult.commute*)

lemma *SUP-real-ennreal*:
  assumes $A \neq \{\}$ *bdd-above* (*f'A*)
  shows (*SUP a∈A. ennreal* (*f a*)) = *ennreal*(*SUP a∈A. f a*)
apply (*rule antisym*, *simp add*: *SUP-least assms*(*2*) *cSUP-upper ennreal-leI*)
by (*metis assms*(*1*) *ennreal-SUP ennreal-less-top le-less*)

lemma *e2ennreal-Liminf*:
  $F \neq bot \Longrightarrow e2ennreal\ (Liminf\ F\ f) = Liminf\ F\ (\lambda n.\ e2ennreal\ (f\ n))$
  by (*rule Liminf-compose-continuous-mono*[*symmetric*])
    (*auto simp*: *mono-def e2ennreal-mono continuous-on-e2ennreal*)

lemma *e2ennreal-eq-infty*[*simp*]: $0 \leq x \Longrightarrow e2ennreal\ x = top \longleftrightarrow x = \infty$
  by (*cases x*) (*auto*)

lemma *ennreal-Inf-cmult*:
  assumes $c>(0::real)$
  shows $Inf\ \{ennreal\ c * x\ |x.\ P\ x\} = ennreal\ c * Inf\ \{x.\ P\ x\}$
proof −
  have $(\lambda x::ennreal.\ c * x)\ (Inf\ \{x::ennreal.\ P\ x\}) = Inf\ ((\lambda x::ennreal.\ c * x)`\{x::ennreal.\ P\ x\})$
    apply (*rule mono-bij-Inf*)
    apply (*simp add*: *monoI mult-left-mono*)
    apply (*rule bij-betw-byWitness*[*of - λx.* (*x*::*ennreal*) / *c*], *auto simp add*: *assms*)
   apply (*metis assms ennreal-lessI ennreal-neq-top mult.commute mult-divide-eq-ennreal not-less-zero*)
   apply (*metis assms divide-ennreal-def ennreal-less-zero-iff ennreal-neq-top less-irrefl mult.assoc mult.left-commute mult-divide-eq-ennreal*)
    done
  then show *?thesis* **by** (*simp only*: *setcompr-eq-image*[*symmetric*])
qed

lemma *continuous-on-const-minus-ennreal*:
  fixes *f* :: *'a* :: *topological-space* ⇒ *ennreal*
  shows *continuous-on A f* ⟹ *continuous-on A* (*λx. a* − *f x*)
  including *ennreal.lifting*
proof (*transfer fixing*: *A*; *clarsimp*)
  fix *f* :: *'a* ⇒ *ereal* **and** *a* :: *ereal* **assume** *0* ≤ *a* ∀ *x*. *0* ≤ *f x* **and** *f*: *continuous-on A f*
  then show *continuous-on A* (*λx. max 0* (*a* − *f x*))
  proof *cases*
    assume ∃ *r*. *a* = *ereal r*
    with *f* **show** *?thesis*
     by (*auto simp*: *continuous-on-def minus-ereal-def ereal-Lim-uminus*[*symmetric*]
          intro!: *tendsto-add-ereal-general tendsto-max*)
  next
    assume ∄ *r*. *a* = *ereal r*
    with ‹*0* ≤ *a*› **have** *a* = ∞
      by (*cases a*) *auto*
    then show *?thesis*
      by (*simp add*: *continuous-on-const*)
  qed
qed

lemma *const-minus-Liminf-ennreal*:
  fixes *a* :: *ennreal*
  shows *F* ≠ *bot* ⟹ *a* − *Liminf F f* = *Limsup F* (*λx. a* − *f x*)
by (*intro Limsup-compose-continuous-antimono*[*symmetric*])
  (*auto simp*: *antimono-def ennreal-mono-minus continuous-on-id continuous-on-const-minus-ennreal*)

lemma *tendsto-cmult-ennreal* [*tendsto-intros*]:
  fixes *c l*::*ennreal*
  assumes ¬(*c* = ∞ ∧ *l* = *0*)
    (*f* ⟶ *l*) *F*
  shows ((*λx. c* ∗ *f x*) ⟶ *c* ∗ *l*) *F*
by (*cases c* = *0*, *insert assms*, *auto intro*!: *tendsto-intros*)

1.9 Indicator-Function.thy

There is something weird with `sum_mult_indicator`: it is defined both in Indicator.thy and BochnerIntegration.thy, with a different meaning. I am surprised there is no name collision... Here, I am using the version from BochnerIntegration.

lemma *sum-indicator-eq-card2*:
  assumes *finite I*
  shows (∑ *i*∈*I*. (*indicator* (*P i*) *x*)::*nat*) = *card* {*i*∈*I*. *x* ∈ *P i*}
using *sum-mult-indicator* [*OF assms*, *of λy. 1*::*nat P λy. x*]
unfolding *card-eq-sum* **by** *auto*

lemma *disjoint-family-indicator-le-1*:

assumes *disjoint-family-on A I*
shows ($\sum$ *i*∈ *I*. *indicator* (*A i*) *x*) ≤ (*1*::′*a*:: {*comm-monoid-add*,*zero-less-one*})
proof (*cases finite I*)
case *True*
then have ∗: ($\sum$ *i*∈ *I*. *indicator* (*A i*) *x*) = ((*indicator* ($\bigcup$*i*∈*I*. *A i*) *x*)::′*a*)
by (*simp add*: *indicator-UN-disjoint*[*OF True assms*(*1*), *of x*])
show *?thesis*
unfolding ∗ **unfolding** *indicator-def* **by** (*simp add*: *order-less-imp-le*)
next
case *False*
then show *?thesis* **by** (*simp add*: *order-less-imp-le*)
qed

1.10 sigma-algebra.thy

lemma *algebra-intersection*:
assumes *algebra* Ω *A*
algebra Ω *B*
shows *algebra* Ω (*A* ∩ *B*)
apply (*subst algebra-iff-Un*) **using** *assms* **by** (*auto simp add*: *algebra-iff-Un*)

lemma *sigma-algebra-intersection*:
assumes *sigma-algebra* Ω *A*
sigma-algebra Ω *B*
shows *sigma-algebra* Ω (*A* ∩ *B*)
apply (*subst sigma-algebra-iff*) **using** *assms* **by** (*auto simp add*: *sigma-algebra-iff algebra-intersection*)

lemma *subalgebra-M-M* [*simp*]:
subalgebra M M
by (*simp add*: *subalgebra-def*)

The next one is `disjoint_family_Suc` with inclusions reversed.

lemma *disjoint-family-Suc2*:
assumes *Suc*: $\bigwedge$*n*. *A* (*Suc n*) ⊆ *A n*
shows *disjoint-family* (λ*i*. *A i* − *A* (*Suc i*))
proof −
have *A* (*m*+*n*) ⊆ *A n* **for** *m n*
proof (*induct m*)
case *0* **show** *?case* **by** *simp*
next
case (*Suc m*) **then show** *?case*
by (*metis Suc-eq-plus1 assms add.commute add.left-commute subset-trans*)
qed
then have *A m* ⊆ *A n* **if** *m* > *n* **for** *m n*
by (*metis that add.commute le-add-diff-inverse nat-less-le*)
then show *?thesis*
by (*auto simp add*: *disjoint-family-on-def*)
(*metis insert-absorb insert-subset le-SucE le-antisym not-le-imp-less*)

qed

1.11 Measure-Space.thy

lemma *AE-equal-sum*:
assumes $\bigwedge i.$ *AE x in M. f i x = g i x*
shows *AE x in M.* $(\sum i{\in}I.\ f\ i\ x) = (\sum i{\in}I.\ g\ i\ x)$
proof (*cases*)
assume *finite I*
have $\exists A.\ A \in$ *null-sets M* $\wedge$ ($\forall x\in$ (*space M* $-$ *A*). *f i x = g i x*) **for** *i*
using *assms*(*1*)[*of i*] **by** (*metis* (*mono-tags*, *lifting*) *AE-E3*)
then obtain *A* **where** *A*: $\bigwedge i.\ A\ i \in$ *null-sets M* $\wedge$ ($\forall x\in$ (*space M* $-$*A i*). *f i x = g i x*)
by *metis*
define *B* **where** $B = (\bigcup i{\in}I.\ A\ i)$
have $B \in$ *null-sets M* **using** ‹*finite I*› *A B-def* **by** *blast*
then have *AE x in M.* $x \in$ *space M* $-$ *B* **by** (*simp add*: *AE-not-in*)
moreover
{
fix *x* **assume** $x \in$ *space M* $-$ *B*
then have $\bigwedge i.\ i \in I \Longrightarrow f\ i\ x = g\ i\ x$ **unfolding** *B-def* **using** *A* **by** *auto*
then have $(\sum i{\in}I.\ f\ i\ x) = (\sum i{\in}I.\ g\ i\ x)$ **by** *auto*
}
ultimately show *?thesis* **by** *auto*
qed (*simp*)

lemma *emeasure-pos-unionE*:
assumes $\bigwedge$ (*N*::*nat*). *A N* $\in$ *sets M*
emeasure M $(\bigcup N.\ A\ N) > 0$
shows $\exists N.$ *emeasure M* $(A\ N) > 0$
proof (*rule ccontr*)
assume $\neg(\exists N.$ *emeasure M* $(A\ N) > 0)$
then have $\bigwedge N.\ A\ N \in$ *null-sets M*
using *assms*(*1*) **by** *auto*
then have $(\bigcup N.\ A\ N) \in$ *null-sets M* **by** *auto*
then show *False* **using** *assms*(*2*) **by** *auto*
qed

lemma (**in** *prob-space*) *emeasure-intersection*:
fixes *e*::*nat* $\Rightarrow$ *real*
assumes [*measurable*]: $\bigwedge n.\ U\ n \in$ *sets M*
and [*simp*]: $\bigwedge n.\ 0 \le e\ n$ *summable e*
and *ge*: $\bigwedge n.$ *emeasure M* $(U\ n) \ge 1 - (e\ n)$
shows *emeasure M* $(\bigcap n.\ U\ n) \ge 1 - (\sum n.\ e\ n)$
proof $-$
define *V* **where** $V = (\lambda n.$ *space M* $- (U\ n))$
have [*measurable*]: $V\ n \in$ *sets M* **for** *n*
unfolding *V-def* **by** *auto*
have $*$: *emeasure M* $(V\ n) \le e\ n$ **for** *n*

unfolding *V-def* **using** *ge*[*of n*] **by** (*simp add*: *emeasure-eq-measure prob-compl ennreal-leI*)
have *emeasure M* $(\bigcup n.\ V\ n) \leq (\sum n.\ emeasure\ M\ (V\ n))$
by (*rule emeasure-subadditive-countably*, *auto*)
also have ... $\leq (\sum n.\ ennreal\ (e\ n))$
using $*$ **by** (*intro suminf-le*) *auto*
also have ... $= ennreal\ (\sum n.\ e\ n)$
by (*intro suminf-ennreal-eq*) *auto*
finally have *emeasure M* $(\bigcup n.\ V\ n) \leq$ *suminf e* **by** *simp*
then have $1 -$ *suminf e* $\leq$ *emeasure M* $(space\ M - (\bigcup n.\ V\ n))$
by (*simp add*: *emeasure-eq-measure prob-compl suminf-nonneg*)
also have ... $\leq$ *emeasure M* $(\bigcap n.\ U\ n)$
by (*rule emeasure-mono*) (*auto simp*: *V-def*)
finally show *?thesis* **by** *simp*
qed

lemma *null-sym-diff-transitive*:
assumes $A \Delta B \in$ *null-sets M* $B \Delta C \in$ *null-sets M*
and [*measurable*]: $A \in$ *sets M* $C \in$ *sets M*
shows $A \Delta C \in$ *null-sets M*
proof −
have $A \Delta B \cup B \Delta C \in$ *null-sets M* **using** *assms*(*1*) *assms*(*2*) **by** *auto*
moreover have $A \Delta C \subseteq A \Delta B \cup B \Delta C$ **by** *auto*
ultimately show *?thesis* **by** (*meson null-sets-subset assms*(*3*) *assms*(*4*) *sets.Diff sets.Un*)
qed

lemma *Delta-null-of-null-is-null*:
assumes $B \in$ *sets M* $A \Delta B \in$ *null-sets M* $A \in$ *null-sets M*
shows $B \in$ *null-sets M*
proof −
have $B \subseteq A \cup (A \Delta B)$ **by** *auto*
then show *?thesis* **using** *assms* **by** (*meson null-sets.Un null-sets-subset*)
qed

lemma *Delta-null-same-emeasure*:
assumes $A \Delta B \in$ *null-sets M* **and** [*measurable*]: $A \in$ *sets M* $B \in$ *sets M*
shows *emeasure M A* $=$ *emeasure M B*
proof −
have $A = (A \cap B) \cup (A-B)$ **by** *blast*
moreover have $A-B \in$ *null-sets M* **using** *assms null-sets-subset* **by** *blast*
ultimately have *a*: *emeasure M A* $=$ *emeasure M* $(A \cap B)$ **using** *emeasure-Un-null-set* **by** (*metis assms*(*2*) *assms*(*3*) *sets.Int*)

have $B = (A \cap B) \cup (B-A)$ **by** *blast*
moreover have $B-A \in$ *null-sets M* **using** *assms null-sets-subset* **by** *blast*
ultimately have *emeasure M B* $=$ *emeasure M* $(A \cap B)$ **using** *emeasure-Un-null-set* **by** (*metis assms*(*2*) *assms*(*3*) *sets.Int*)
then show *?thesis* **using** *a* **by** *auto*

qed

lemma *AE-upper-bound-inf-ereal*:
  fixes $F\ G{::}'a \Rightarrow ereal$
  assumes $\bigwedge e.\ (e{::}real) > 0 \Longrightarrow AE\ x\ in\ M.\ F\ x \le G\ x + e$
  shows $AE\ x\ in\ M.\ F\ x \le G\ x$
proof −
  have $AE\ x\ in\ M.\ \forall n{::}nat.\ F\ x \le G\ x + ereal\ (1\ /\ Suc\ n)$
    using *assms* **by** (*auto simp*: *AE-all-countable*)
  then show *?thesis*
  proof (*eventually-elim*)
    fix *x* **assume** *x*: $\forall n{::}nat.\ F\ x \le G\ x + ereal\ (1\ /\ Suc\ n)$
    show $F\ x \le G\ x$
    proof (*intro ereal-le-epsilon2*[*of - G x*] *allI impI*)
      fix $e :: real$ **assume** $0 < e$
      then obtain *n* **where** *n*: $1\ /\ Suc\ n < e$
        by (*blast elim*: *nat-approx-posE*)
      have $F\ x \le G\ x + 1\ /\ Suc\ n$
        using *x* **by** *simp*
      also have $\ldots \le G\ x + e$
        using *n* **by** (*intro add-mono ennreal-leI*) *auto*
      finally show $F\ x \le G\ x + ereal\ e$ **.**
    qed
  qed
qed

Egorov theorem asserts that, if a sequence of functions converges almost everywhere to a limit, then the convergence is uniform on a subset of close to full measure. The first step in the proof is the following lemma, often useful by itself, asserting the same result for predicates: if a property $P_n x$ is eventually true for almost every x, then there exists N such that $P_n x$ is true for all $n \ge N$ and all x in a set of close to full measure.

lemma (**in** *finite-measure*) *Egorov-lemma*:
  assumes [*measurable*]: $\bigwedge n.\ (P\ n) \in measurable\ M\ (count\text{-}space\ UNIV)$
      and $AE\ x\ in\ M.\ eventually\ (\lambda n.\ P\ n\ x)\ sequentially$
        $epsilon > 0$
  shows $\exists U\ N.\ U \in sets\ M \wedge (\forall n \ge N.\ \forall x \in U.\ P\ n\ x) \wedge emeasure\ M\ (space\ M - U) < epsilon$
proof −
  define *K* **where** $K = (\lambda n.\ \{x \in space\ M.\ \exists k{\ge}n.\ \neg(P\ k\ x)\})$
  have [*measurable*]: $K\ n \in sets\ M$ **for** *n*
    unfolding *K-def* **by** *auto*
  have $x \notin (\bigcap n.\ K\ n)$ **if** $eventually\ (\lambda n.\ P\ n\ x)\ sequentially$ **for** *x*
    unfolding *K-def* **using** *that* **unfolding** *K-def eventually-sequentially* **by** *auto*
  then have $AE\ x\ in\ M.\ x \notin (\bigcap n.\ K\ n)$ **using** *assms* **by** *auto*
  then have *Z*: $0 = emeasure\ M\ (\bigcap n.\ K\ n)$
   using *AE-iff-measurable*[*of* $(\bigcap n.\ K\ n)\ M\ \lambda x.\ x \notin (\bigcap n.\ K\ n)$] **unfolding** *K-def* **by** *auto*
  have $*$: $(\lambda n.\ emeasure\ M\ (K\ n)) \longrightarrow 0$

unfolding *Z* **apply** (*rule Lim-emeasure-decseq*) **using** *order-trans* **by** (*auto simp add*: *K-def decseq-def*)
have *eventually* ($\lambda n.$ *emeasure M* (*K n*) < *epsilon*) *sequentially*
by (*rule order-tendstoD*(*2*)[*OF* ∗ ‹*epsilon* > *0*›])
then obtain *N* **where** *N*: $\bigwedge n.\ n \geq N \Longrightarrow$ *emeasure M* (*K n*) < *epsilon*
unfolding *eventually-sequentially* **by** *auto*
define *U* **where** *U* = *space M* − *K N*
have *A* [*measurable*]: *U* ∈ *sets M* **unfolding** *U-def* **by** *auto*
have *space M* − *U* = *K N*
unfolding *U-def K-def* **by** *auto*
then have *B*: *emeasure M* (*space M* − *U*) < *epsilon*
using *N* **by** *auto*
have ∀ *n* ≥ *N*. ∀ *x* ∈ *U*. *P n x*
unfolding *U-def K-def* **by** *auto*
then show *?thesis* **using** *A B* **by** *blast*
qed

The next lemma asserts that, in an uncountable family of disjoint sets, then there is one set with zero measure (and in fact uncountably many). It is often applied to the boundaries of r-neighborhoods of a given set, to show that one could choose r for which this boundary has zero measure (this shows up often in relation with weak convergence).

lemma (**in** *finite-measure*) *uncountable-disjoint-family-then-exists-zero-measure*:
assumes [*measurable*]: $\bigwedge i.\ i \in I \Longrightarrow$ *A i* ∈ *sets M*
and *uncountable I*
disjoint-family-on A I
shows ∃ *i*∈*I*. *measure M* (*A i*) = *0*
proof −
define *f* **where** *f* = (λ(*r*::*real*). {*i* ∈ *I*. *measure M* (*A i*) > *r*})
have ∗: *finite* (*f r*) **if** *r* > *0* **for** *r*
proof −
obtain *N*::*nat* **where** *N*: *measure M* (*space M*)/*r* ≤ *N*
using *real-arch-simple* **by** *blast*
have *finite* (*f r*) ∧ *card* (*f r*) ≤ *N*
proof (*rule finite-if-finite-subsets-card-bdd*)
fix *G* **assume** *G*: *G* ⊆ *f r finite G*
then have *G* ⊆ *I* **unfolding** *f-def* **by** *auto*
have *card G* ∗ *r* = (∑ *i* ∈ *G*. *r*) **by** *auto*
also have ... ≤ (∑ *i* ∈ *G*. *measure M* (*A i*))
apply (*rule sum-mono*) **using** *G* **unfolding** *f-def* **by** *auto*
also have ... = *measure M* (⋃ *i*∈*G*. *A i*)
apply (*rule finite-measure-finite-Union*[*symmetric*])
using ‹*finite G*› ‹*G* ⊆ *I*› ‹*disjoint-family-on A I*› *disjoint-family-on-mono* **by** *auto*
also have ... ≤ *measure M* (*space M*)
by (*simp add*: *bounded-measure*)
finally have *card G* ≤ *measure M* (*space M*)/*r*
using ‹*r* > *0*› **by** (*simp add*: *divide-simps*)
then show *card G* ≤ *N* **using** *N* **by** *auto*

qed
then show *?thesis* **by** *simp*
qed
have *countable* ($\bigcup$ *n*. *f* (((*1*::*real*)/*2*)^*n*))
by (*rule countable-UN*, *auto intro*!: *countable-finite* ∗)
then have *I* − ($\bigcup$ *n*. *f* (((*1*::*real*)/*2*)^*n*)) ≠ {}
using *assms*(*2*) **by** (*metis countable-empty uncountable-minus-countable*)
then obtain *i* **where** *i* ∈ *I i* ∉ ($\bigcup$ *n*. *f* ((*1*/*2*)^*n*)) **by** *auto*
then have *measure M* (*A i*) ≤ (*1* / *2*) ^ *n* **for** *n*
unfolding *f-def* **using** *linorder-not-le* **by** *auto*
moreover have (λ*n*. ((*1*::*real*) / *2*) ^ *n*) ⟶ *0*
by (*intro tendsto-intros*, *auto*)
ultimately have *measure M* (*A i*) ≤ *0*
using *LIMSEQ-le-const* **by** *force*
then have *measure M* (*A i*) = *0*
by (*simp add*: *measure-le-0-iff*)
then show *?thesis* **using** ‹*i* ∈ *I*› **by** *auto*
qed

The next statements are useful measurability statements.

lemma *measurable-Inf* [*measurable*]:
assumes [*measurable*]: $\bigwedge$(*n*::*nat*). *P n* ∈ *measurable M* (*count-space UNIV*)
shows (λ*x*. *Inf* {*n*. *P n x*}) ∈ *measurable M* (*count-space UNIV*) (**is** *?f* ∈ -)
proof −
define *A* **where** *A* = (λ*n*. (*P n*)−‘{*True*} ∩ *space M* − ($\bigcup$ *m*<*n*. (*P m*)−‘{*True*} ∩ *space M*))
have *A-meas* [*measurable*]: *A n* ∈ *sets M* **for** *n* **unfolding** *A-def* **by** *measurable*
define *B* **where** *B* = (λ*n*. *if n* = *0 then* (*space M* − ($\bigcup$ *n*. *A n*)) *else A* (*n*−*1*))
show *?thesis*
proof (*rule measurable-piecewise-restrict2*[*of B*])
show *B n* ∈ *sets M* **for** *n* **unfolding** *B-def* **by** *simp*
show *space M* = ($\bigcup$ *n*. *B n*)
unfolding *B-def* **using** *sets.sets-into-space* [*OF A-meas*] **by** *auto*
have ∗: *?f x* = *n* **if** *x* ∈ *A n* **for** *x n*
apply (*rule cInf-eq-minimum*) **using** *that* **unfolding** *A-def* **by** *auto*
moreover have ∗∗: *?f x* = (*Inf* ({}::*nat set*)) **if** *x* ∈ *space M* − ($\bigcup$ *n*. *A n*) **for** *x*
proof −
have ¬(*P n x*) **for** *n*
apply (*induction n rule*: *nat-less-induct*) **using** *that* **unfolding** *A-def* **by** *auto*
then show *?thesis* **by** *simp*
qed
ultimately have ∃ *c*. ∀ *x* ∈ *B n*. *?f x* = *c* **for** *n*
apply (*cases n* = *0*) **unfolding** *B-def* **by** *auto*
then show ∃ *h* ∈ *measurable M* (*count-space UNIV*). ∀ *x* ∈ *B n*. *?f x* = *h x* **for** *n*
by *fastforce*
qed

qed

lemma *measurable-T-iter* [*measurable*]:
  fixes f::$'a \Rightarrow nat$
  assumes [*measurable*]: $T \in$ *measurable M M*
    $f \in$ *measurable M* (*count-space UNIV*)
  shows (λx. (T^^(f x)) x) $\in$ *measurable M M*
proof −
  have [*measurable*]: (T^^n) $\in$ *measurable M M* **for** n::*nat*
    by (*induction n*, *auto*)
  show *?thesis*
    by (*rule measurable-compose-countable*, *auto*)
qed

lemma *measurable-infdist* [*measurable*]:
  (λx. *infdist x S*) $\in$ *borel-measurable borel*
by (*rule borel-measurable-continuous-onI*, *intro continuous-intros*)

The next lemma shows that, in a sigma finite measure space, sets with large measure can be approximated by sets with large but finite measure.

lemma (**in** *sigma-finite-measure*) *approx-with-finite-emeasure*:
  assumes *W-meas*: $W \in$ *sets M*
    and *W-inf*: *emeasure M W* $> C$
  obtains Z **where** $Z \in$ *sets M* $Z \subseteq W$ *emeasure M Z* $< \infty$ *emeasure M Z* $> C$
proof (*cases emeasure M W* $= \infty$)
  case *True*
  obtain r **where** r: $C =$ *ennreal r* **using** *W-inf* **by** (*cases C*, *auto*)
  obtain Z **where** $Z \in$ *sets M* $Z \subseteq W$ *emeasure M Z* $< \infty$ *emeasure M Z* $> C$
    unfolding r **using** *approx-PInf-emeasure-with-finite*[*OF W-meas True*, *of r*]
by *auto*
  then show *?thesis* **using** *that* **by** *blast*
next
  case *False*
  then have $W \in$ *sets M* $W \subseteq W$ *emeasure M W* $< \infty$ *emeasure M W* $> C$
    using *assms* **apply** *auto* **using** *top.not-eq-extremum* **by** *blast*
  then show *?thesis* **using** *that* **by** *blast*
qed

1.12 Nonnegative-Lebesgue-Integration.thy

The next lemma is a variant of `nn_integral_density`, with the density on the right instead of the left, as seems more common.

lemma *nn-integral-densityR*:
  assumes [*measurable*]: $f \in$ *borel-measurable F* $g \in$ *borel-measurable F*
  shows ($\int^+ x.\ f\ x * g\ x\ \partial F$) = ($\int^+ x.\ f\ x\ \partial$(*density F g*))
proof −
  have ($\int^+ x.\ f\ x * g\ x\ \partial F$) = ($\int^+ x.\ g\ x * f\ x\ \partial F$) **by** (*simp add*: *mult.commute*)
  also have ... = ($\int^+ x.\ f\ x\ \partial$(*density F g*))

by (*rule nn-integral-density*[*symmetric*], *simp-all add*: *assms*)
finally show *?thesis* **by** *simp*
qed

lemma *not-AE-zero-int-ennreal-E*:
fixes $f::'a \Rightarrow ennreal$
assumes $(\int^+ x.\ f\ x\ \partial M) > 0$
and [*measurable*]: $f \in$ *borel-measurable M*
shows $\exists A{\in}sets\ M.\ \exists e{::}real{>}0.\ emeasure\ M\ A > 0 \wedge (\forall x \in A.\ f\ x \geq e)$
proof (*rule not-AE-zero-ennreal-E*, *auto simp add*: *assms*)
assume $*$: *AE x in M*. $f\ x = 0$
have $(\int^+ x.\ f\ x\ \partial M) = (\int^+ x.\ 0\ \partial M)$ **by** (*rule nn-integral-cong-AE*, *simp add*: $*$)
then have $(\int^+ x.\ f\ x\ \partial M) = 0$ **by** *simp*
then show *False* **using** *assms* **by** *simp*
qed

lemma (**in** *finite-measure*) *nn-integral-bounded-eq-bound-then-AE*:
assumes *AE x in M*. $f\ x \leq$ *ennreal c* $(\int^+ x.\ f\ x\ \partial M) = c * emeasure\ M\ (space\ M)$
and [*measurable*]: $f \in$ *borel-measurable M*
shows *AE x in M*. $f\ x = c$
proof (*cases*)
assume *emeasure M* (*space M*) $= 0$
then show *?thesis* **by** (*rule emeasure-0-AE*)
next
assume *emeasure M* (*space M*) $\neq 0$
have *fin*: *AE x in M*. $f\ x \neq top$ **using** *assms* **by** (*auto simp*: *top-unique*)
define *g* **where** $g = (\lambda x.\ c - f\ x)$
have [*measurable*]: $g \in$ *borel-measurable M* **unfolding** *g-def* **by** *auto*
have $(\int^+ x.\ g\ x\ \partial M) = (\int^+ x.\ c\ \partial M) - (\int^+ x.\ f\ x\ \partial M)$
unfolding *g-def* **by** (*rule nn-integral-diff*, *auto simp add*: *assms ennreal-mult-eq-top-iff*)
also have $\ldots = 0$ **using** *assms*(*2*) **by** (*auto simp*: *ennreal-mult-eq-top-iff*)
finally have *AE x in M*. $g\ x = 0$
by (*subst nn-integral-0-iff-AE*[*symmetric*]) *auto*
then have *AE x in M*. $c \leq f\ x$ **unfolding** *g-def* **using** *fin* **by** (*auto simp*: *ennreal-minus-eq-0*)
then show *?thesis* **using** *assms*(*1*) **by** *auto*
qed

lemma *null-sets-density*:
assumes [*measurable*]: $h \in$ *borel-measurable M*
and *AE x in M*. $h\ x \neq 0$
shows *null-sets* (*density M h*) $=$ *null-sets M*
proof −
have $*$: $A \in sets\ M \wedge (AE\ x{\in}A\ in\ M.\ h\ x = 0) \longleftrightarrow A \in null\text{-}sets\ M$ **for** *A*
proof (*auto*)
assume $A \in sets\ M\ AE\ x{\in}A\ in\ M.\ h\ x = 0$

```
    then show A ∈ null-sets M
      unfolding AE-iff-null-sets[OF ‹A ∈ sets M›] using assms(2) by auto
  next
    assume A ∈ null-sets M
    then show AE x∈A in M. h x = 0
      by (metis (mono-tags, lifting) AE-not-in eventually-mono)
  qed
  show ?thesis
    apply (rule set-eqI)
   unfolding null-sets-density-iff[OF ‹h ∈ borel-measurable M›] using * by auto
qed
```

The next proposition asserts that, if a function h is integrable, then its integral on any set with small enough measure is small. The good conceptual proof is by considering the distribution of the function h on $\mathbb{R}$ and looking at its tails. However, there is a less conceptual but more direct proof, based on dominated convergence and a proof by contradiction. This is the proof we give below.

```
proposition integrable-small-integral-on-small-sets:
  fixes h::'a ⇒ real
  assumes [measurable]: integrable M h
      and delta > 0
   shows ∃epsilon>(0::real). ∀ U ∈ sets M. emeasure M U < epsilon ⟶ abs
(∫ x∈U. h x ∂M) < delta
proof (rule ccontr)
  assume H: ¬ (∃epsilon>0. ∀ U∈sets M. emeasure M U < ennreal epsilon ⟶
abs(set-lebesgue-integral M U h) < delta)
  have ∃f. ∀ epsilon∈{0<..}. f epsilon ∈sets M ∧ emeasure M (f epsilon) < ennreal
epsilon
                        ∧ ¬(abs(set-lebesgue-integral M (f epsilon) h) < delta)
    apply (rule bchoice) using H by auto
  then obtain f::real ⇒ 'a set where f:
            ⋀epsilon. epsilon > 0 ⟹ f epsilon ∈sets M
            ⋀epsilon. epsilon > 0 ⟹ emeasure M (f epsilon) < ennreal epsilon
            ⋀epsilon. epsilon > 0 ⟹ ¬(abs(set-lebesgue-integral M (f epsilon) h)
< delta)
    by blast
  define A where A = (λn::nat. f ((1/2)^n))
  have [measurable]: A n ∈ sets M for n
    unfolding A-def using f(1) by auto
  have *: emeasure M (A n) < ennreal ((1/2)^n) for n
    unfolding A-def using f(2) by auto
  have Large: ¬(abs(set-lebesgue-integral M (A n) h) < delta) for n
    unfolding A-def using f(3) by auto

  have S: summable (λn. Sigma-Algebra.measure M (A n))
    apply (rule summable-comparison-test'[of λn. (1/2)^n 0])
    apply (rule summable-geometric, auto)
    apply (subst ennreal-le-iff[symmetric], simp)
```

using *less-imp-le*[*OF* ∗] **by** (*metis* ∗ *emeasure-eq-ennreal-measure top.extremum-strict*)
have *AE x in M*. *eventually* (λ*n*. *x* ∈ *space M* − *A n*) *sequentially*
apply (*rule borel-cantelli-AE1*, *auto simp add*: *S*)
by (*metis* ∗ *top.extremum-strict top.not-eq-extremum*)
moreover have (λ*n*. *indicator* (*A n*) *x* ∗ *h x*) ⟶ *0*
if *eventually* (λ*n*. *x* ∈ *space M* − *A n*) *sequentially* **for** *x*
proof −
have *eventually* (λ*n*. *indicator* (*A n*) *x* ∗ *h x* = *0*) *sequentially*
apply (*rule eventually-mono*[*OF that*]) **unfolding** *indicator-def* **by** *auto*
then show *?thesis*
unfolding *eventually-sequentially* **using** *lim-explicit* **by** *force*
qed
ultimately have *A*: *AE x in M*. ((λ*n*. *indicator* (*A n*) *x* ∗ *h x*) ⟶ *0*)
by *auto*
have *I*: *integrable M* (λ*x*. *abs*(*h x*))
using ‹*integrable M h*› **by** *auto*
have *L*: (λ*n*. *abs* (∫ *x*. *indicator* (*A n*) *x* ∗ *h x* ∂*M*)) ⟶ *abs* (∫ *x*. *0* ∂*M*)
apply (*intro tendsto-intros*)
apply (*rule integral-dominated-convergence*[*OF* - - *I A*])
unfolding *indicator-def* **by** *auto*
have *eventually* (λ*n*. *abs* (∫ *x*. *indicator* (*A n*) *x* ∗ *h x* ∂*M*) < *delta*) *sequentially*
apply (*rule order-tendstoD*[*OF L*]) **using** ‹*delta* > *0*› **by** *auto*
then show *False*
using *Large* **by** (*auto simp*: *set-lebesgue-integral-def*)
qed

We also give the version for nonnegative ennreal valued functions. It follows from the previous one.

proposition *small-nn-integral-on-small-sets*:
fixes *h*::′*a* ⇒ *ennreal*
assumes [*measurable*]: *h* ∈ *borel-measurable M*
and *delta* > (*0*::*real*) (∫ ⁺*x*. *h x* ∂*M*) ≠ ∞
shows ∃ *epsilon*>(*0*::*real*). ∀ *U* ∈ *sets M*. *emeasure M U* < *epsilon* ⟶ (∫ ⁺*x*∈*U*. *h x* ∂*M*) < *delta*
proof −
define *f* **where** *f* = (λ*x*. *enn2real*(*h x*))
have *AE x in M*. *h x* ≠ ∞
using *assms* **by** (*metis nn-integral-PInf-AE*)
then have ∗: *AE x in M*. *ennreal* (*f x*) = *h x*
unfolding *f-def* **using** *ennreal-enn2real-if* **by** *auto*
have ∗∗: (∫ ⁺*x*. *ennreal* (*f x*) ∂*M*) ≠ ∞
using *nn-integral-cong-AE*[*OF* ∗] *assms* **by** *auto*
have [*measurable*]: *f* ∈ *borel-measurable M* **unfolding** *f-def* **by** *auto*
have *integrable M f*
apply (*rule integrableI-nonneg*) **using** *assms* ∗ *f-def* ∗∗ **apply** *auto*
using *top.not-eq-extremum* **by** *blast*
obtain *epsilon*::*real* **where** *H*: *epsilon* > *0* ⋀*U*. *U* ∈ *sets M* ⟹ *emeasure M U* < *epsilon* ⟹ *abs*(∫ *x*∈*U*. *f x* ∂*M*) < *delta*
using *integrable-small-integral-on-small-sets*[*OF* ‹*integrable M f*› ‹*delta* > *0*›]

by *blast*
have ($\int^+ x \in U.\ h\ x\ \partial M$) < *delta* **if** [*measurable*]: $U \in$ *sets M emeasure M U* <
epsilon **for** *U*
proof –
have ($\int^+ x.$ *indicator U x* ∗ *h x* ∂M) = ($\int^+ x.$ *ennreal*(*indicator U x* ∗ *f x*)
∂M)
apply (*rule nn-integral-cong-AE*) **using** ∗ **unfolding** *indicator-def* **by** *auto*
also have ... = *ennreal* ($\int x.$ *indicator U x* ∗ *f x* ∂M)
apply (*rule nn-integral-eq-integral*)
apply (*rule Bochner-Integration.integrable-bound*[*OF* ‹*integrable M f*›])
unfolding *indicator-def f-def* **by** *auto*
also have ... < *ennreal delta*
apply (*rule ennreal-lessI*) **using** *H*(*2*)[*OF that*] **by** (*auto simp*: *set-lebesgue-integral-def*)
finally show *?thesis* **by** (*auto simp add*: *mult.commute*)
qed
then show *?thesis* **using** ‹*epsilon* > *0*› **by** *auto*
qed

1.13 Probability-measure.thy

The next lemmas ensure that, if sets have a probability close to 1, then their intersection also does.

lemma (**in** *prob-space*) *sum-measure-le-measure-inter*:
assumes $A \in$ *sets M* $B \in$ *sets M*
shows *prob A* + *prob B* ≤ *1* + *prob* ($A \cap B$)
proof –
have *prob A* + *prob B* = *prob* ($A \cup B$) + *prob* ($A \cap B$)
by (*simp add*: *assms fmeasurable-eq-sets measure-Un3*)
also have ... ≤ *1* + *prob* ($A \cap B$)
by *auto*
finally show *?thesis* **by** *simp*
qed

lemma (**in** *prob-space*) *sum-measure-le-measure-inter3*:
assumes [*measurable*]: $A \in$ *sets M* $B \in$ *sets M* $C \in$ *sets M*
shows *prob A* + *prob B* + *prob C* ≤ *2* + *prob* ($A \cap B \cap C$)
using *sum-measure-le-measure-inter*[*of B C*] *sum-measure-le-measure-inter*[*of A B*
∩ *C*]
by (*auto simp add*: *inf-assoc*)

lemma (**in** *prob-space*) *sum-measure-le-measure-Inter*:
assumes [*measurable*]: *finite I I* ≠ {} $\bigwedge i.\ i \in I \Longrightarrow A\ i \in$ *sets M*
shows ($\sum i \in I.$ *prob* (*A i*)) ≤ *real*(*card I*) – *1* + *prob* ($\bigcap i \in I.\ A\ i$)
using *assms* **proof** (*induct I rule*: *finite-ne-induct*)
fix *x F* **assume** *H*: *finite F F* ≠ {} $x \notin F$
(($\bigwedge i.\ i \in F \Longrightarrow A\ i \in$ *events*) ⟹ ($\sum i \in F.$ *prob* (*A i*)) ≤ *real* (*card F*)
– *1* + *prob* ($\bigcap$(*A* ‘ *F*)))
and [*measurable*]: ($\bigwedge i.\ i \in$ *insert x F* ⟹ *A i* ∈ *events*)
have ($\bigcap x \in F.\ A\ x$) ∈ *events* **using** ‹*finite F*› ‹*F* ≠ {}› **by** *auto*

have $(\sum i{\in}insert\ x\ F.\ prob\ (A\ i)) = (\sum i{\in}F.\ prob\ (A\ i)) + prob\ (A\ x)$
  using *H(1) H(3)* **by** *auto*
 also have ... $\le$ *real (card F)−1 + prob* $(\bigcap(A\ \text{`}\ F))$ + *prob (A x)*
  using *H(4)* **by** *auto*
 also have ... $\le$ *real (card F) + prob* $((\bigcap(A\ \text{`}\ F)) \cap A\ x)$
  using *sum-measure-le-measure-inter[OF* ‹$(\bigcap x{\in}F.\ A\ x) \in events$›, *of A x]* **by** *auto*
 also have ... = *real (card (insert x F)) − 1 + prob* $(\bigcap(A\ \text{`}\ (insert\ x\ F)))$
  using *H(1) H(2)* **unfolding** *card-insert-disjoint[OF* ‹*finite F*› ‹$x \notin F$›*]* **by** *(simp add: inf-commute)*
 finally show $(\sum i{\in}insert\ x\ F.\ prob\ (A\ i)) \le real\ (card\ (insert\ x\ F)) - 1 + prob$ $(\bigcap(A\ \text{`}\ (insert\ x\ F)))$
  by *simp*
qed *(auto)*

A random variable gives a small mass to small neighborhoods of infinity.

lemma (**in** *prob-space*) *random-variable-small-tails*:
 assumes *alpha* > *0* **and** *[measurable]*: $f \in$ *borel-measurable M*
 shows $\exists(C{::}real).\ prob\ \{x \in space\ M.\ abs(f\ x) \ge C\} < alpha \wedge C \ge K$
proof −
 have ∗: $(\bigcap(n{::}nat).\ \{x{\in}space\ M.\ abs(f\ x) \ge n\}) = \{\}$
  apply *auto*
  by *(metis real-arch-simple add.right-neutral add-mono-thms-linordered-field(4) not-less zero-less-one)*
 have ∗∗: $(\lambda n.\ prob\ \{x \in space\ M.\ abs(f\ x) \ge n\}) \longrightarrow prob\ (\bigcap(n{::}nat).\ \{x \in space\ M.\ abs(f\ x) \ge n\})$
  by *(rule finite-Lim-measure-decseq, auto simp add: decseq-def)*
 have *eventually* $(\lambda n.\ prob\ \{x \in space\ M.\ abs(f\ x) \ge n\} < alpha)$ *sequentially*
  apply *(rule order-tendstoD[OF - ‹alpha > 0›])* **using** ∗∗ **unfolding** ∗ **by** *auto*
 then obtain *N::nat* **where** *N*: $\bigwedge n{::}nat.\ n \ge N \Longrightarrow prob\ \{x \in space\ M.\ abs(f\ x) \ge n\} < alpha$
  unfolding *eventually-sequentially* **by** *blast*
 have $\exists n{::}nat.\ n \ge N \wedge n \ge K$
  by *(meson le-cases of-nat-le-iff order.trans real-arch-simple)*
 then obtain *n::nat* **where** *n*: $n \ge N$ $n \ge K$ **by** *blast*
 show *?thesis*
  apply *(rule exI[of - of-nat n])* **using** *N n* **by** *auto*
qed

1.14 Distribution-functions.thy

There is a locale called `finite_borel_measure` in `distribution-functions.thy`. However, it only deals with real measures, and real weak convergence. I will not need the weak convergence in more general settings, but still it seems more natural to me to do the proofs in the natural settings. Let me introduce the locale `finite_borel_measure'` for this, although it would be better to rename the locale in the library file.

locale *finite-borel-measure'* = *finite-measure M* **for** *M* :: *('a::metric-space) measure*

+
  assumes *M-is-borel* [*simp*, *measurable-cong*]: *sets M* = *sets borel*
begin

lemma *space-eq-univ* [*simp*]: *space M* = *UNIV*
  using *M-is-borel*[*THEN sets-eq-imp-space-eq*] **by** *simp*

lemma *measurable-finite-borel* [*simp*]:
  f ∈ *borel-measurable borel* ⟹ *f* ∈ *borel-measurable M*
  by (*rule borel-measurable-subalgebra*[**where** *N* = *borel*]) *auto*

Any closed set can be slightly enlarged to obtain a set whose boundary has 0 measure.

lemma *approx-closed-set-with-set-zero-measure-boundary*:
  assumes *closed S epsilon* > *0 S* ≠ {}
  shows ∃ *r*. *r* < *epsilon* ∧ *r* > *0* ∧ *measure M* {*x*. *infdist x S* = *r*} = *0* ∧ *measure M* {*x*. *infdist x S* ≤ *r*} < *measure M S* + *epsilon*
proof −
  have [*measurable*]: *S* ∈ *sets M*
    using ‹*closed S*› **by** *auto*
  define *T* **where** *T* = (λ*r*. {*x*. *infdist x S* ≤ *r*})
  have [*measurable*]: *T r* ∈ *sets borel* **for** *r*
    unfolding *T-def* **by** *measurable*
  have ∗: (⋂ *n*. *T* ((*1/2*)^*n*)) = *S*
  unfolding *T-def* **proof** (*auto*)
    fix *x* **assume** ∗: ∀ *n*. *infdist x S* ≤ (*1* / *2*) ^*n*
    have *infdist x S* ≤ *0*
      apply (*rule LIMSEQ-le-const*[*of* λ*n*. (*1/2*)^*n*], *intro tendsto-intros*) **using** ∗ **by** *auto*
    then show *x* ∈ *S*
      using *assms infdist-pos-not-in-closed* **by** *fastforce*
  qed
  have *A*: ((*1*::*real*)/*2*)^*n* ≤ (*1/2*)^*m* **if** *m* ≤ *n* **for** *m n*::*nat*
    using *that* **by** (*simp add*: *power-decreasing*)
  have (λ*n*. *measure M* (*T* ((*1/2*)^*n*))) ⟶ *measure M S*
    unfolding ∗[*symmetric*] **apply** (*rule finite-Lim-measure-decseq*, *auto simp add*: *T-def decseq-def*)
    using *A order.trans* **by** *blast*
  then have *B*: *eventually* (λ*n*. *measure M* (*T* ((*1/2*)^*n*)) < *measure M S* + *epsilon*) *sequentially*
    apply (*rule order-tendstoD*) **using** ‹*epsilon* > *0*› **by** *simp*
  have *C*: *eventually* (λ*n*. (*1/2*)^*n* < *epsilon*) *sequentially*
    by (*rule order-tendstoD*[*OF* - ‹*epsilon* > *0*›], *intro tendsto-intros*, *auto*)
  obtain *n* **where** *n*: (*1/2*)^*n* < *epsilon measure M* (*T* ((*1/2*)^*n*)) < *measure M S* + *epsilon*
    using *eventually-conj*[*OF B C*] **unfolding** *eventually-sequentially* **by** *auto*
  have ∃ *r*∈{*0*<..<(*1/2*)^*n*}. *measure M* {*x*. *infdist x S* = *r*} = *0*
    apply (*rule uncountable-disjoint-family-then-exists-zero-measure*, *auto simp add*: *disjoint-family-on-def*)

using *uncountable-open-interval* **by** *fastforce*
then obtain *r* **where** *r*: *r*∈{*0*<..<(*1*/*2*)^*n*} *measure M* {*x. infdist x S* = *r*} = *0*
by *blast*
then have *r2*: *r* > *0 r* < *epsilon* **using** *n* **by** *auto*
have *measure M* {*x. infdist x S* ≤ *r*} ≤ *measure M* {*x. infdist x S* ≤ (*1*/*2*)^*n*}
apply (*rule finite-measure-mono*) **using** *r* **by** *auto*
then have *measure M* {*x. infdist x S* ≤ *r*} < *measure M S* + *epsilon*
using *n*(*2*) **unfolding** *T-def* **by** *auto*
then show *?thesis*
using *r*(*2*) *r2* **by** *auto*
qed
end

sublocale *finite-borel-measure* ⊆ *finite-borel-measure′*
by (*standard*, *simp add*: *M-is-borel*)

1.15 Weak-convergence.thy

Since weak convergence is not implemented as a topology, the fact that the convergence of a sequence implies the convergence of a subsequence is not automatic. We prove it in the lemma below..

lemma *weak-conv-m-subseq*:
assumes *weak-conv-m M-seq M strict-mono r*
shows *weak-conv-m* (*λn. M-seq* (*r n*)) *M*
using *assms LIMSEQ-subseq-LIMSEQ* **unfolding** *weak-conv-m-def weak-conv-def comp-def* **by** *auto*

context
fixes *μ* :: *nat* ⇒ *real measure*
and *M* :: *real measure*
assumes *μ*: ⋀*n. real-distribution* (*μ n*)
assumes *M*: *real-distribution M*
assumes *μ-to-M*: *weak-conv-m μ M*
begin

The measure of a closed set behaves upper semicontinuously with respect to weak convergence: if $\mu_n \to \mu$, then $\limsup \mu_n(F) \leq \mu(F)$ (and the inequality can be strict, think of the situation where μ is a Dirac mass at 0 and $F = \{0\}$, but μ_n has a density so that $\mu_n(\{0\}) = 0$).

lemma *closed-set-weak-conv-usc*:
assumes *closed S measure M S* < *l*
shows *eventually* (*λn. measure* (*μ n*) *S* < *l*) *sequentially*
proof (*cases S* = {})
case *True*
then show *?thesis*
using ‹*measure M S* < *l*› **by** *auto*
next

case *False*
interpret *real-distribution M* **using** *M* **by** *simp*
define *epsilon* **where** *epsilon = l − measure M S*
have *epsilon > 0* **unfolding** *epsilon-def* **using** *assms(2)* **by** *auto*
obtain *r* **where** *r*: *r > 0 r < epsilon measure M {x. infdist x S = r} = 0 measure M {x. infdist x S ≤ r} < measure M S + epsilon*
using *approx-closed-set-with-set-zero-measure-boundary[OF ‹closed S› ‹epsilon > 0› ‹S ≠ {}›]* **by** *blast*
define *T* **where** *T = {x. infdist x S ≤ r}*
have *[measurable]*: *T ∈ sets borel*
unfolding *T-def* **by** *auto*
have *S ⊆ T*
unfolding *T-def* **using** *‹closed S› ‹r > 0›* **by** *auto*
have *measure M T < l*
using *r(4)* **unfolding** *T-def epsilon-def* **by** *auto*
have *measure M (frontier T) ≤ measure M {x. infdist x S = r}*
apply *(rule finite-measure-mono)* **unfolding** *T-def* **using** *frontier-indist-le* **by** *auto*
then have *measure M (frontier T) = 0*
using *‹measure M {x. infdist x S = r} = 0›* **by** *(auto simp add: measure-le-0-iff)*
then have *(λn. measure (μ n) T) ⟶ measure M T*
using *μ-to-M* **by** *(simp add: μ emeasure-eq-measure real-distribution-axioms weak-conv-imp-continuity-set-conv)*
then have *∗*: *eventually (λn. measure (μ n) T < l) sequentially*
apply *(rule order-tendstoD)* **using** *‹measure M T < l›* **by** *simp*
have *∗∗*: *measure (μ n) S ≤ measure (μ n) T* **for** *n*
apply *(rule finite-measure.finite-measure-mono)*
using *μ* **apply** *(simp add: finite-borel-measure.axioms(1) real-distribution.finite-borel-measure-M)*
using *‹S ⊆ T›* **apply** *simp*
by *(simp add: μ real-distribution.events-eq-borel)*
show *?thesis*
apply *(rule eventually-mono[OF ∗])* **using** *∗∗ le-less-trans* **by** *auto*
qed

In the same way, the measure of an open set behaves lower semicontinuously with respect to weak convergence: if $\mu_n \to \mu$, then $\liminf \mu_n(U) \geq \mu(U)$ (and the inequality can be strict). This follows from the same statement for closed sets by passing to the complement.

lemma *open-set-weak-conv-lsc*:
assumes *open S measure M S > l*
shows *eventually (λn. measure (μ n) S > l) sequentially*
proof −
interpret *real-distribution M*
using *M* **by** *auto*
have *[measurable]*: *S ∈ events* **using** *assms(1)* **by** *auto*
have *eventually (λn. measure (μ n) (UNIV − S) < 1 − l) sequentially*
apply *(rule closed-set-weak-conv-usc)*
using *assms prob-compl[of S]* **by** *auto*
moreover have *measure (μ n) (UNIV − S) = 1 − measure (μ n) S* **for** *n*

proof −
  interpret *mu*: *real-distribution* μ *n*
    using μ **by** *auto*
  have $S \in$ *mu.events* **using** *assms(1)* **by** *auto*
  then show *?thesis* **using** *mu.prob-compl*[*of S*] **by** *auto*
qed
ultimately show *?thesis* **by** *auto*
qed

end

end

theory *ME-Library-Complement*
  imports *HOL−Analysis.Analysis*
begin

1.16 The trivial measurable space

The trivial measurable space is the smallest possible σ-algebra, i.e. only the empty set and everything.

definition *trivial-measure* :: *'a set* ⇒ *'a measure* **where**
  trivial-measure X = *sigma X* {{}, *X*}

lemma *space-trivial-measure* [*simp*]: *space* (*trivial-measure X*) = *X*
  by (*simp add*: *trivial-measure-def*)

lemma *sets-trivial-measure*: *sets* (*trivial-measure X*) = {{}, *X*}
  by (*simp add*: *trivial-measure-def sigma-algebra-trivial sigma-algebra.sigma-sets-eq*)

lemma *measurable-trivial-measure*:
  assumes *f* ∈ *space M* → *X* **and** *f* −‘ *X* ∩ *space M* ∈ *sets M*
  shows *f* ∈ *M* $\rightarrow_M$ *trivial-measure X*
  using *assms* **unfolding** *measurable-def* **by** (*auto simp*: *sets-trivial-measure*)

lemma *measurable-trivial-measure-iff*:
  f ∈ *M* $\rightarrow_M$ *trivial-measure X* ⟷ *f* ∈ *space M* → *X* ∧ *f* −‘ *X* ∩ *space M* ∈ *sets M*
  unfolding *measurable-def* **by** (*auto simp*: *sets-trivial-measure*)

1.17 Pullback algebras

The pullback algebra $f^{-1}(\Sigma)$ of a σ-algebra (Ω, Σ) is the smallest σ-algebra such that f is $f^{-1}(\Sigma) - -\Sigma$-measurable.

definition (**in** *sigma-algebra*) *pullback-algebra* :: (*'b* ⇒ *'a*) ⇒ *'b set* ⇒ *'b set set* **where**
  pullback-algebra f Ω′ = *sigma-sets* Ω′ {*f* −‘ *A* ∩ Ω′ |*A*. *A* ∈ *M*}

lemma *pullback-algebra-minimal*:
  assumes $f \in M \rightarrow_M N$
  shows *sets.pullback-algebra N f* (*space M*) ⊆ *sets M*
proof
  fix *X* **assume** *X* ∈ *sets.pullback-algebra N f* (*space M*)
  thus *X* ∈ *sets M*
    unfolding *sets.pullback-algebra-def*
    by *induction* (*use assms* **in** ‹*auto simp*: *measurable-def*›)
qed

lemma (**in** *sigma-algebra*) *in-pullback-algebra*: *A* ∈ *M* ⟹ *f* −‘ *A* ∩ Ω′ ∈ *pullback-algebra f* Ω′
  unfolding *pullback-algebra-def* **by** (*rule sigma-sets.Basic*) *auto*

end

2 Subadditive and submultiplicative sequences

theory *Fekete*
  imports *HOL−Analysis.Multivariate-Analysis*
begin

A real sequence is subadditive if $u_{n+m} \leq u_n + u_m$. This implies the convergence of u_n/n to $Inf\{u_n/n\} \in [-\infty, +\infty)$, a useful result known as Fekete lemma. We prove it below.

Taking logarithms, the same result applies to submultiplicative sequences. We illustrate it with the definition of the spectral radius as the limit of $\|x^n\|^{1/n}$, the convergence following from Fekete lemma.

2.1 Subadditive sequences

We define subadditive sequences, either from the start or eventually.

definition *subadditive*::(*nat*⇒*real*) ⇒ *bool*
  where *subadditive u* = (∀ *m n*. *u* (*m*+*n*) ≤ *u m* + *u n*)

lemma *subadditiveI*:
  assumes ⋀*m n*. *u* (*m*+*n*) ≤ *u m* + *u n*
  shows *subadditive u*
unfolding *subadditive-def* **using** *assms* **by** *auto*

lemma *subadditiveD*:
  assumes *subadditive u*
  shows *u* (*m*+*n*) ≤ *u m* + *u n*
using *assms* **unfolding** *subadditive-def* **by** *auto*

lemma *subadditive-un-le-nu1*:
  assumes *subadditive u*

```
      n > 0
  shows u n ≤ n ∗ u 1
proof −
  have ∗: n = 0 ∨ (u n ≤ n ∗ u 1) for n
  proof (induction n)
    case 0
    then show ?case by auto
  next
    case (Suc n)
    consider n = 0 | n > 0 by auto
    then show ?case
    proof (cases)
      case 1
      then show ?thesis by auto
    next
      case 2
      then have u (Suc n) ≤ u n + u 1 using subadditiveD[OF assms(1), of n 1]
by auto
      then show ?thesis using Suc.IH 2 by (auto simp add: algebra-simps)
    qed
  qed
  show ?thesis using ∗[of n] ‹n > 0› by auto
qed

definition eventually-subadditive::(nat⇒real) ⇒ nat ⇒ bool
  where eventually-subadditive u N0 = (∀ m>N0. ∀ n>N0. u (m+n) ≤ u m + u
n)

lemma eventually-subadditiveI:
  assumes ⋀m n. m > N0 ⟹ n > N0 ⟹ u (m+n) ≤ u m + u n
  shows eventually-subadditive u N0
unfolding eventually-subadditive-def using assms by auto

lemma subadditive-imp-eventually-subadditive:
  assumes subadditive u
  shows eventually-subadditive u 0
using assms unfolding subadditive-def eventually-subadditive-def by auto
```

The main inequality that will lead to convergence is given in the next lemma: given n, then eventually u_m/m is bounded by u_n/n, up to an arbitrarily small error. This is proved by doing the euclidean division of m by n and using the subadditivity. (the remainder in the euclidean division will give the error term.)

```
lemma eventually-subadditive-ineq:
  assumes eventually-subadditive u N0 e>0 n>N0
  shows ∃ N>N0. ∀ m≥N. u m/m < u n/n + e
proof −
  have ineq-rec: u(a∗n+r) ≤ a ∗ u n + u r if n>N0 r>N0 for a n r
  proof (induct a)
```

case (*Suc a*)
have *a∗n+r>N0* **using** ‹*r>N0*› **by** *simp*
have *u((Suc a)∗n+r) = u(a∗n+r+n)* **by** (*simp add*: *algebra-simps*)
also have ... ≤ *u(a∗n+r)+u n* **using** *assms* ‹*n>N0*› ‹*a∗n+r>N0*› *eventually-subadditive-def* **by** *blast*
also have ... ≤ *a∗u n + u r + u n* **by** (*simp add*: *Suc.hyps*)
also have ... = (*Suc a*) ∗ *u n + u r* **by** (*simp add*: *algebra-simps*)
finally show *?case* **by** *simp*
qed (*simp*)

have *n>0 real n > 0* **using** ‹*n>N0*› **by** *auto*
define *C* **where** *C = Max {abs(u i) |i. i≤2∗n}*
have *ineq-C*: *abs(u i) ≤ C* **if** *i ≤ 2 ∗ n* **for** *i*
unfolding *C-def* **by** (*intro Max-ge*, *auto simp add*: *that*)

have *ineq-all-m*: *u m/m ≤ u n/n + 3∗C/m* **if** *m≥n* **for** *m*
proof −
have *real m>0* **using** ‹*m≥n*› ‹*0 < real n*› **by** *linarith*

obtain *a0 r0* **where** *r0<n m = a0∗n+r0*
using ‹*0 < n*› *mod-div-decomp mod-less-divisor* **by** *blast*
define *a* **where** *a = a0−1*
define *r* **where** *r = r0+n*
have *r<2∗n r≥n* **unfolding** *r-def* **by** (*auto simp add*: ‹*r0<n*›)
have *a0>0* **using** ‹*m = a0∗n + r0*› ‹*n ≤ m*› ‹*r0 < n*› *not-le* **by** *fastforce*
then have *m = a ∗ n + r* **using** *a-def r-def* ‹*m = a0∗n+r0*› *mult-eq-if* **by** *auto*
then have *real-eq*: *−r = real n ∗ a − m* **by** *simp*

have *r>N0* **using** ‹*r≥n*› ‹*n>N0*› **by** *simp*
then have *u m ≤ a ∗ u n + u r* **using** *ineq-rec* ‹*m = a∗n+r*› ‹*n>N0*› **by** *simp*
then have *n ∗ u m ≤ n ∗ (a ∗ u n + u r)* **using** ‹*real n>0*› **by** *simp*
then have *n ∗ u m − m ∗ u n ≤ −r ∗ u n + n ∗ u r*
unfolding *real-eq* **by** (*simp add*: *algebra-simps*)
also have ... ≤ *r ∗ abs(u n) + n ∗ abs(u r)*
apply (*intro add-mono mult-left-mono*) **using** *real-0-le-add-iff* **by** *fastforce+*
also have ... ≤ *(2 ∗ n) ∗ C + n ∗ C*
apply (*intro add-mono mult-mono ineq-C*) **using** *less-imp-le*[*OF* ‹*r < 2 ∗ n*›] **by** *auto*
finally have *n ∗ u m − m ∗ u n ≤ 3∗C∗n* **by** *auto*
then show *u m/m ≤ u n/n + 3∗C/m*
using ‹*0 < real n*› ‹*0 < real m*› **by** (*simp add*: *divide-simps mult.commute*)
qed

obtain *M::nat* **where** *M*: *M ≥ 3 ∗ C / e* **using** *real-nat-ceiling-ge* **by** *auto*
define *N* **where** *N = M + n + N0 + 1*
have *N > 3 ∗ C / e N ≥ n N > N0* **unfolding** *N-def* **using** *M* **by** *auto*
have *u m/m < u n/n + e* **if** *m ≥ N* **for** *m*

```
  proof –
    have 3 ∗ C / m < e
      using that ‹N > 3 ∗ C / e› ‹e > 0› apply (auto simp add: algebra-simps
divide-simps)
      by (meson le-less-trans linorder-not-le mult-less-cancel-left-pos of-nat-less-iff)
    then show ?thesis using ineq-all-m[of m] ‹n ≤ N› ‹N ≤ m› by auto
  qed
  then show ?thesis using ‹N0 < N› by blast
qed
```

From the inequality above, we deduce the convergence of u_n/n to its infimum. As this infimum might be $-\infty$, we formulate this convergence in the extended reals. Then, we specialize it to the real situation, separating the cases where u_n/n is bounded below or not.

```
lemma subadditive-converges-ereal':
  assumes eventually-subadditive u N0
  shows (λm. ereal(u m/m)) ⟶ Inf {ereal(u n/n) | n. n>N0}
proof –
  define v where v = (λm. ereal(u m/m))
  define V where V = {v n | n. n>N0}
  define l where l = Inf V

  have ⋀t. t∈V ⟹ t≥l by (simp add: Inf-lower l-def)
  then have v n ≥ l if n > N0 for n using V-def that by blast
  then have lower: eventually (λn. a < v n) sequentially if a < l for a
    by (meson that dual-order.strict-trans1 eventually-at-top-dense)

  have upper: eventually (λn. a > v n) sequentially if a > l for a
  proof –
    obtain t where t∈V t<a by (metis ‹a>l› Inf-greatest l-def not-le)
     then obtain e::real where e>0 t+e < a by (meson ereal-le-epsilon2 leD
le-less-linear)
    obtain n where n>N0 t = u n/n using V-def v-def ‹t ∈ V› by blast
    then have u n/n + e < a using ‹t+e < a› by simp
    obtain N where ∀m≥N. u m/m < u n/n + e
      using eventually-subadditive-ineq[OF assms] ‹0 < e› ‹N0 < n› by blast
    then have u m/m < a if m ≥ N for m
      using that ‹u n/n + e < a› less-ereal.simps(1) less-trans by blast
    then have v m< a if m ≥ N for m using v-def that by blast
    then show ?thesis using eventually-at-top-linorder by auto
  qed
  show ?thesis
   using lower upper unfolding V-def l-def v-def by (simp add: order-tendsto-iff)
qed

lemma subadditive-converges-ereal:
  assumes subadditive u
  shows (λm. ereal(u m/m)) ⟶ Inf {ereal(u n/n) | n. n>0}
by (rule subadditive-converges-ereal'[OF subadditive-imp-eventually-subadditive[OF
```

assms]])

lemma *subadditive-converges-bounded'*:
  assumes *eventually-subadditive u N0*
    bdd-below {*u n/n* | *n. n>N0*}
  shows (λ*n. u n/n*) ⟶ *Inf* {*u n/n* | *n. n>N0*}
proof−
  have ∗: (λ*n. ereal*(*u n* /*n*)) ⟶ *Inf* {*ereal*(*u n/n*)|*n. n* > *N0*}
    by (*simp add*: *assms*(*1*) *subadditive-converges-ereal'*)
  define *V* **where** *V* = {*u n/n* | *n. n>N0*}
  have *a*: *bdd-below V V*≠{} **by** (*auto simp add*: *V-def assms*(*2*))
  have *Inf* {*ereal*(*t*)| *t. t*∈*V*} = *ereal*(*Inf V*) **by** (*subst ereal-Inf'*[*OF a*], *simp add*: *Setcompr-eq-image*)
  moreover have {*ereal*(*t*)| *t. t*∈*V*} = {*ereal*(*u n/n*)|*n. n* > *N0*} **using** *V-def* **by** *blast*
  ultimately have *Inf* {*ereal*(*u n/n*)|*n. n* > *N0*} = *ereal*(*Inf* {*u n/n* |*n. n* > *N0*}) **using** *V-def* **by** *auto*
  then have (λ*n. ereal*(*u n* /*n*)) ⟶ *ereal*(*Inf* {*u n/n* | *n. n>N0*}) **using** ∗ **by** *auto*
  then show *?thesis* **by** *simp*
qed

lemma *subadditive-converges-bounded*:
  assumes *subadditive u*
    bdd-below {*u n/n* | *n. n>0*}
  shows (λ*n. u n/n*) ⟶ *Inf* {*u n/n* | *n. n>0*}
by (*rule subadditive-converges-bounded'*[*OF subadditive-imp-eventually-subadditive*[*OF assms*(*1*)] *assms*(*2*)]])

We reformulate the previous lemma in a more directly usable form, avoiding the infimum.

lemma *subadditive-converges-bounded''*:
  assumes *subadditive u*
    ⋀*n. n* > *0* ⟹ *u n* ≥ *n* ∗ (*a*::*real*)
  shows ∃ *l*. (λ*n. u n* / *n*) ⟶ *l* ∧ (∀ *n>0*. *u n* ≥ *n* ∗ *l*)
proof −
  have *B*: *bdd-below* {*u n/n* | *n. n>0*}
    apply (*rule bdd-belowI*[*of - a*]) **using** *assms*(*2*)
    apply (*auto simp add*: *divide-simps*)
    apply (*metis mult.commute mult-left-le-imp-le of-nat-0-less-iff*)
    done
  define *l* **where** *l* = *Inf* {*u n/n* | *n. n>0*}
  have ∗: *u n* / *n* ≥ *l* **if** *n* > *0* **for** *n*
    unfolding *l-def* **using** *that* **by** (*auto intro*!: *cInf-lower*[*OF - B*])
  show *?thesis*
    apply (*rule exI*[*of - l*], *auto*)
    using *subadditive-converges-bounded*[*OF assms*(*1*) *B*] **apply** (*simp add*: *l-def*)
    using ∗ **by** (*simp add*: *divide-simps algebra-simps*)
qed

lemma *subadditive-converges-unbounded'*:
  assumes *eventually-subadditive u N0*
    ¬ (*bdd-below* {*u n*/*n* | *n*. *n*>*N0*})
  shows (λ*n*. *ereal*(*u n*/*n*)) ⟶ −∞
proof −
  have ∗: (λ*n*. *ereal*(*u n* /*n*)) ⟶ *Inf* {*ereal*(*u n*/*n*)|*n*. *n* > *N0*}
    by (*simp add*: *assms*(*1*) *subadditive-converges-ereal'*)
  define *V* **where** *V* = {*u n*/*n* | *n*. *n*>*N0*}
  then have ¬ *bdd-below V* **using** *assms* **by** *simp*
  have *Inf* {*ereal*(*t*) | *t*. *t*∈*V*} = −∞
    by (*rule ereal-bot*, *metis* (*mono-tags*, *lifting*) ‹¬ *bdd-below V*› *bdd-below-def*
      leI Inf-lower2 ereal-less-eq(*3*) *le-less mem-Collect-eq*)
  moreover have {*ereal*(*t*)| *t*. *t*∈*V*} = {*ereal*(*u n*/*n*)|*n*. *n* > *N0*} **using** *V-def* **by** *blast*
  ultimately have *Inf* {*ereal*(*u n*/*n*)|*n*. *n* > *N0*} = −∞ **by** *auto*
  then show *?thesis* **using** ∗ **by** *simp*
qed

lemma *subadditive-converges-unbounded*:
  assumes *subadditive u*
    ¬ (*bdd-below* {*u n*/*n* | *n*. *n*>*0*})
  shows (λ*n*. *ereal*(*u n*/*n*)) ⟶ −∞
by (*rule subadditive-converges-unbounded'*[*OF subadditive-imp-eventually-subadditive*[*OF assms*(*1*)] *assms*(*2*)]])

2.2 Superadditive sequences

While most applications involve subadditive sequences, one sometimes encounters superadditive sequences. We reformulate quickly some of the above results in this setting.

definition *superadditive*::(*nat*⇒*real*) ⇒ *bool*
  where *superadditive u* = (∀ *m n*. *u* (*m*+*n*) ≥ *u m* + *u n*)

lemma *subadditive-of-superadditive*:
  assumes *superadditive u*
  shows *subadditive* (λ*n*. −*u n*)
using *assms* **unfolding** *superadditive-def subadditive-def* **by** (*auto simp add*: *algebra-simps*)

lemma *superadditive-un-ge-nu1*:
  assumes *superadditive u*
    n > *0*
  shows *u n* ≥ *n* ∗ *u 1*
using *subadditive-un-le-nu1* [*OF subadditive-of-superadditive*[*OF assms*(*1*)] *assms*(*2*)]
by *auto*

lemma *superadditive-converges-bounded''*:
  assumes *superadditive u*

$\bigwedge n.\ n > 0 \Longrightarrow u\ n \le n * (a::real)$
shows $\exists\, l.\ (\lambda n.\ u\ n\ /\ n) \longrightarrow l \wedge (\forall\, n{>}0.\ u\ n \le n * l)$
proof −
have $\exists\, l.\ (\lambda n.\ -u\ n\ /\ n) \longrightarrow l \wedge (\forall\, n{>}0.\ -u\ n \ge n * l)$
apply (*rule subadditive-converges-bounded''*[*OF subadditive-of-superadditive*[*OF assms*(*1*)], *of* −*a*])
using *assms*(*2*) **by** *auto*
then obtain *l* **where** *l*: $(\lambda n.\ -u\ n\ /\ n) \longrightarrow l\ (\forall\, n{>}0.\ -u\ n \ge n * l)$ **by** *blast*
have $(\lambda n.\ -((-u\ n)/n)) \longrightarrow -l$
by (*intro tendsto-intros l*)
moreover have $\forall\, n{>}0.\ u\ n \le n * (-l)$
using *l*(*2*) **by** (*auto simp add*: *algebra-simps*) (*metis minus-equation-iff neg-le-iff-le*)
ultimately show *?thesis*
by *auto*
qed

2.3 Almost additive sequences

One often encounters sequences which are both subadditive and superadditive, but only up to an additive constant. Adding or subtracting this constant, one can make the sequence genuinely subadditive or superadditive, and thus deduce results about its convergence, as follows. Such sequences appear notably when dealing with quasimorphisms.

lemma *almost-additive-converges*:
fixes *u*::*nat* ⇒ *real*
assumes $\bigwedge m\ n.\ abs(u(m{+}n) - u\ m - u\ n) \le C$
shows *convergent* $(\lambda n.\ u\ n/n)$
$abs(u\ k - k * lim\ (\lambda n.\ u\ n\ /\ n)) \le C$
proof −
have $(abs\ (u\ 0)) \le C$ **using** *assms*[*of 0 0*] **by** *auto*
then have $C \ge 0$ **by** *auto*

define *v* **where** $v = (\lambda n.\ u\ n + C)$
have *subadditive v*
unfolding *subadditive-def v-def* **using** *assms* **by** (*auto simp add*: *algebra-simps abs-diff-le-iff*)
then have *vle*: $v\ n \le n * v\ 1$ **if** $n > 0$ **for** *n*
using *subadditive-un-le-nu1 that* **by** *auto*
define *w* **where** $w = (\lambda n.\ u\ n - C)$
have *superadditive w*
unfolding *superadditive-def w-def* **using** *assms* **by** (*auto simp add*: *algebra-simps abs-diff-le-iff*)
then have *wge*: $w\ n \ge n * w\ 1$ **if** $n > 0$ **for** *n*
using *superadditive-un-ge-nu1 that* **by** *auto*

have *I*: $v\ n \ge w\ n$ **for** *n*
unfolding *v-def w-def* **using** ‹$C \ge 0$› **by** *auto*

then have ∗: *v n* ≥ *n* ∗ *w 1* **if** *n* > *0* **for** *n* **using** *order-trans*[*OF wge*[*OF that*]] **by** *auto*
then obtain *lv* **where** *lv*: (λ*n*. *v n*/*n*) ⟶ *lv* ⋀*n*. *n* > *0* ⟹ *v n* ≥ *n* ∗ *lv*
using *subadditive-converges-bounded''*[*OF* ‹*subadditive v*› ∗] **by** *auto*
have ∗: *w n* ≤ *n* ∗ *v 1* **if** *n* > *0* **for** *n* **using** *order-trans*[*OF - vle*[*OF that*]] *I* **by** *auto*
then obtain *lw* **where** *lw*: (λ*n*. *w n*/*n*) ⟶ *lw* ⋀*n*. *n* > *0* ⟹ *w n* ≤ *n* ∗ *lw*
using *superadditive-converges-bounded''*[*OF* ‹*superadditive w*› ∗] **by** *auto*

have ∗: *v n*/*n* = *w n* /*n* + *2*∗*C*∗(*1*/*n*) **for** *n*
unfolding *v-def w-def* **by** (*auto simp add*: *algebra-simps divide-simps*)
have (λ*n*. *w n* /*n* + *2*∗*C*∗(*1*/*n*)) ⟶ *lw* + *2*∗*C*∗*0*
by (*intro tendsto-add tendsto-mult lim-1-over-n lw*, *auto*)
then have *lw* = *lv*
unfolding ∗[*symmetric*] **using** *lv*(*1*) *LIMSEQ-unique* **by** *auto*

have ∗: *u n*/*n* = *w n* /*n* + *C*∗(*1*/*n*) **for** *n*
unfolding *w-def* **by** (*auto simp add*: *algebra-simps divide-simps*)
have (λ*n*. *u n* /*n*) ⟶ *lw* + *C*∗*0*
unfolding ∗ **by** (*intro tendsto-add tendsto-mult lim-1-over-n lw*, *auto*)
then have *lu*: *convergent* (λ*n*. *u n*/*n*) *lim* (λ*n*. *u n*/*n*) = *lw*
by (*auto simp add*: *convergentI limI*)
then show *convergent* (λ*n*. *u n*/*n*) **by** *simp*

show *abs*(*u k* − *k* ∗ *lim* (λ*n*. *u n* / *n*)) ≤ *C*
proof (*cases k*>*0*)
case *False*
then show *?thesis* **using** *assms*[*of 0 0*] **by** *auto*
next
case *True*
have *u k* − *k* ∗ *lim* (λ*n*. *u n*/*n*) = *v k* − *C* − *k* ∗ *lv* **unfolding** *lu*(*2*) ‹*lw* = *lv*› *v-def* **by** *auto*
also have ... ≥ −*C* **using** *lv*(*2*)[*OF True*] **by** *auto*
finally have *A*: *u k* − *k* ∗ *lim* (λ*n*. *u n*/*n*) ≥ − *C* **by** *simp*
have *u k* − *k* ∗ *lim* (λ*n*. *u n*/*n*) = *w k* + *C* − *k* ∗ *lw* **unfolding** *lu*(*2*) *w-def* **by** *auto*
also have ... ≤ *C* **using** *lw*(*2*)[*OF True*] **by** *auto*
finally show *?thesis* **using** *A* **by** *auto*
qed
qed

2.4 Submultiplicative sequences, application to the spectral radius

In the same way as subadditive sequences, one may define submultiplicative sequences. Essentially, a sequence is submultiplicative if its logarithm is subadditive. A difference is that we allow a submultiplicative sequence to take the value 0, as this shows up in applications. This implies that we have to distinguish in the proofs the situations where the value 0 is taken or not. In the latter situation, we can use directly the results from the subadditive case to deduce convergence. In the former situation, convergence to 0 is obvious as the sequence vanishes eventually.

lemma *submultiplicative-converges*:
  fixes *u*::*nat*⇒*real*
  assumes ⋀*n*. *u n* ≥ *0*
        ⋀*m n*. *u* (*m*+*n*) ≤ *u m* ∗ *u n*
  shows (λ*n*. *root n* (*u n*))⟶ *Inf* {*root n* (*u n*) | *n*. *n*>*0*}
proof −
  define *v* **where** *v* = (λ *n*. *root n* (*u n*))
  define *V* **where** *V* = {*v n* | *n*. *n*>*0*}
  then have *V* ≠ {} **by** *blast*
  have *t* ≥ *0* **if** *t* ∈ *V* **for** *t* **using** *that V-def v-def assms*(*1*) **by** *auto*
  then have *Inf V* ≥ *0* **by** (*simp add*: ‹*V* ≠ {}› *cInf-greatest*)
  have *bdd-below V* **by** (*meson* ‹⋀*t*. *t* ∈ *V* ⟹ *0* ≤ *t*› *bdd-below-def*)

  show *?thesis*
  proof *cases*
    assume ∃ *n*. *u n* = *0*
    then obtain *n* **where** *u n* = *0* **by** *auto*
    then have *u m* = *0* **if** *m* ≥ *n* **for** *m* **by** (*metis that antisym-conv assms*(*1*) *assms*(*2*) *le-Suc-ex mult-zero-left*)
    then have ∗: *v m* = *0* **if** *m* ≥ *n* **for** *m* **using** *v-def that* **by** *simp*
    then have *v* ⟶ *0* **using** *lim-explicit* **by** *force*

    have *v* (*Suc n*) ∈ *V* **using** *V-def* **by** *blast*
    moreover have *v* (*Suc n*) = *0* **using** ∗ **by** *auto*
    ultimately have *Inf V* ≤ *0* **by** (*simp add*: ‹*bdd-below V*› *cInf-lower*)
    then have *Inf V* = *0* **using** ‹*0* ≤ *Inf V*› **by** *auto*
    then show *?thesis* **using** *V-def v-def* ‹*v* ⟶ *0*› **by** *auto*
  next
    assume ¬ (∃ *n*. *u n* = *0*)
    then have *u n* > *0* **for** *n* **by** (*metis assms*(*1*) *less-eq-real-def*)
    define *w* **where** *w n* = *ln* (*u n*) **for** *n*
    have *express-vn*: *v n* = *exp*(*w n*/*n*) **if** *n*>*0* **for** *n*
    proof −
      have (*exp*(*w n*/*n*))^*n* = *exp*(*n*∗(*w n*/*n*)) **by** (*metis exp-of-nat-mult*)
      also have ... = *exp*(*w n*) **by** (*simp add*: ‹*0* < *n*›)
      also have ... = *u n* **by** (*simp add*: ‹⋀*n*. *0* < *u n*› *w-def*)
      finally have *exp*(*w n*/*n*) = *root n* (*u n*) **by** (*metis* ‹*0* < *n*› *exp-ge-zero real-root-power-cancel*)
      then show *?thesis* **unfolding** *v-def* **by** *simp*
    qed

    have *eventually-subadditive w 0*
    proof (*rule eventually-subadditiveI*)
      fix *m n*
      have *w* (*m*+*n*) = *ln* (*u* (*m*+*n*)) **by** (*simp add*: *w-def*)

also have ... ≤ *ln*(*u m* ∗ *u n*)
by (*meson* ‹⋀*n*. *0* < *u n*› *assms*(*2*) *zero-less-mult-iff ln-le-cancel-iff*)
also have ... = *ln*(*u m*) + *ln*(*u n*)
by (*meson* ‹⋀*n*. *0* < *u n*› *ln-mult-pos*)
also have ... = *w m* + *w n* **by** (*simp add*: *w-def*)
finally show *w* (*m*+*n*) ≤ *w m* + *w n*.
qed

define *l* **where** *l* = *Inf V*
then have *v n*≥*l* **if** *n* > *0* **for** *n*
using *V-def that* **by** (*metis* (*mono-tags*, *lifting*) ‹*bdd-below V*› *cInf-lower mem-Collect-eq*)
then have *lower*: *eventually* (λ*n*. *a* < *v n*) *sequentially* **if** *a* < *l* **for** *a*
by (*meson that dual-order.strict-trans1 eventually-at-top-dense*)

have *upper*: *eventually* (λ*n*. *a* > *v n*) *sequentially* **if** *a* > *l* **for** *a*
proof −
obtain *t* **where** *t*∈*V t* < *a* **using** ‹*V* ≠ {}› *cInf-lessD l-def* ‹*a*>*l*› **by** *blast*
then have *t* > *0* **using** *V-def* ‹⋀*n*. *0* < *u n*› *v-def* **by** *auto*
then have *a*/*t* > *1* **using** ‹*t*<*a*› **by** *simp*
define *e* **where** *e* = *ln*(*a*/*t*)/*2*
have *e* > *0 e* < *ln*(*a*/*t*) **unfolding** *e-def* **by** (*simp-all add*: ‹*1* < *a* / *t*› *ln-gt-zero*)
then have *exp*(*e*) < *a*/*t* **by** (*metis* ‹*1* < *a* / *t*› *exp-less-cancel-iff exp-ln less-trans zero-less-one*)

obtain *n* **where** *n*>*0 t* = *v n* **using** *V-def v-def* ‹*t* ∈ *V*› **by** *blast*
with ‹*0* < *t*› **have** *v n* ∗ *exp*(*e*) < *a* **using** ‹*exp*(*e*) < *a*/*t*›
by (*auto simp add*: *field-simps*)

obtain *N* **where** ∗: *N*>*0* ⋀*m*. *m*≥*N* ⟹ *w m*/*m* < *w n*/*n* + *e*
using *eventually-subadditive-ineq*[*OF* ‹*eventually-subadditive w 0*›] ‹*0* < *n*› ‹*e*>*0*› **by** *blast*
have *v m* < *a* **if** *m* ≥ *N* **for** *m*
proof −
have *m*>*0* **using** *that* ‹*N*>*0*› **by** *simp*
have *w m*/*m* < *w n*/*n* + *e* **by** (*simp add*: ‹*N* ≤ *m*› ∗)
then have *exp*(*w m*/*m*) < *exp*(*w n*/*n* + *e*) **by** *simp*
also have ... = *exp*(*w n*/*n*) ∗ *exp*(*e*) **by** (*simp add*: *mult-exp-exp*)
finally have *v m* < *v n* ∗ *exp*(*e*) **using** *express-vn* ‹*m*>*0*› ‹*n*>*0*› **by** *simp*
then show *v m* < *a* **using** ‹*v n* ∗ *exp*(*e*) < *a*› **by** *simp*
qed
then show *?thesis* **using** *eventually-at-top-linorder* **by** *auto*
qed

show *?thesis*
using *lower upper* **unfolding** *v-def l-def V-def* **by** (*simp add*: *order-tendsto-iff*)
qed
qed

An important application of submultiplicativity is to prove the existence of the spectral radius of a matrix, as the limit of $\|A^n\|^{1/n}$.

```
definition spectral-radius::'a::real-normed-algebra-1 ⇒ real
  where spectral-radius x = Inf {root n (norm(x^n))| n. n>0}

lemma spectral-radius-aux:
  fixes x::'a::real-normed-algebra-1
  defines V ≡ {root n (norm(x^n))| n. n>0}
  shows ⋀t. t∈V ⟹ t ≥ spectral-radius x
        ⋀t. t∈V ⟹ t ≥ 0
        bdd-below V
        V ≠ {}
        Inf V ≥ 0
proof –
  show V≠{} using V-def by blast
 show *: t ≥ 0 if t ∈ V for t using that unfolding V-def using real-root-pos-pos-le
by auto
  then show bdd-below V by (meson bdd-below-def)
  then show Inf V ≥ 0 by (simp add: ‹V ≠ {}› * cInf-greatest)
 show ⋀t. t∈V ⟹ t ≥ spectral-radius x by (metis (mono-tags, lifting) ‹bdd-below
V› assms cInf-lower spectral-radius-def)
qed

lemma spectral-radius-nonneg [simp]:
  spectral-radius x ≥ 0
by (simp add: spectral-radius-aux(5) spectral-radius-def)

lemma spectral-radius-upper-bound [simp]:
  (spectral-radius x)^n ≤ norm(x^n)
proof (cases)
  assume ¬(n = 0)
  have root n (norm(x^n)) ≥ spectral-radius x
    using spectral-radius-aux ‹n ≠ 0› by auto
  then show ?thesis
     by (metis ‹n ≠ 0› spectral-radius-nonneg norm-ge-zero not-gr0 power-mono
real-root-pow-pos2)
qed (simp)

lemma spectral-radius-limit:
  (λn. root n (norm(x^n))) ⟶ spectral-radius x
proof –
   have norm(x^(m+n)) ≤ norm(x^m) * norm(x^n) for m n by (simp add:
power-add norm-mult-ineq)
 then show ?thesis unfolding spectral-radius-def using submultiplicative-converges
by auto
qed

end
```

3 Asymptotic densities

```
theory Asymptotic-Density
  imports SG-Library-Complement
begin
```

The upper asymptotic density of a subset A of the integers is $\limsup Card(A \cap [0, n))/n \in [0, 1]$. It measures how big a set of integers is, at some times. In this paragraph, we establish the basic properties of this notion.

There is a corresponding notion of lower asymptotic density, with a liminf instead of a limsup, measuring how big a set is at all times. The corresponding properties are proved exactly in the same way.

3.1 Upper asymptotic densities

As limsups are only defined for sequences taking values in a complete lattice (here the extended reals), we define it in the extended reals and then go back to the reals. This is a little bit artificial, but it is not a real problem as in the applications we will never come back to this definition.

```
definition upper-asymptotic-density::nat set ⇒ real
 where upper-asymptotic-density A = real-of-ereal(limsup (λn. card(A ∩ {..<n})/n))
```

First basic property: the asymptotic density is between 0 and 1.

```
lemma upper-asymptotic-density-in-01:
  ereal(upper-asymptotic-density A) = limsup (λn. card(A ∩ {..<n})/n)
  upper-asymptotic-density A ≤ 1
  upper-asymptotic-density A ≥ 0
proof −
  {
    fix n::nat assume n>0
     have card(A ∩ {..<n}) ≤ n by (metis card-lessThan Int-lower2 card-mono
finite-lessThan)
    then have card(A ∩ {..<n}) / n ≤ ereal 1 using ‹n>0› by auto
  }
  then have eventually (λn. card(A ∩ {..<n}) / n ≤ ereal 1) sequentially
    by (simp add: eventually-at-top-dense)
 then have a: limsup (λn. card(A ∩ {..<n})/n) ≤ 1 by (simp add: Limsup-const
Limsup-bounded)

  have card(A ∩ {..<n}) / n ≥ ereal 0 for n by auto
   then have liminf (λn. card(A ∩ {..<n})/n) ≥ 0 by (simp add: le-Liminf-iff
less-le-trans)
 then have b: limsup (λn. card(A ∩ {..<n})/n) ≥ 0 by (meson Liminf-le-Limsup
order-trans sequentially-bot)

  have abs(limsup (λn. card(A ∩ {..<n})/n)) ≠ ∞ using a b by auto
 then show ereal(upper-asymptotic-density A) = limsup (λn. card(A ∩ {..<n})/n)
```

unfolding *upper-asymptotic-density-def* **by** *auto*
show *upper-asymptotic-density A* ≤ *1 upper-asymptotic-density A* ≥ *0* **unfolding** *upper-asymptotic-density-def*
using *a b* **by** (*auto simp add*: *real-of-ereal-le-1 real-of-ereal-pos*)
qed

The two next propositions give the usable characterization of the asymptotic density, in terms of the eventual cardinality of $A \cap [0, n)$. Note that the inequality is strict for one implication and large for the other.

proposition *upper-asymptotic-densityD*:
fixes *l*::*real*
assumes *upper-asymptotic-density A* < *l*
shows *eventually* (λ*n*. *card*(*A* ∩ {..<*n*}) < *l* ∗ *n*) *sequentially*
proof −
have *limsup* (λ*n*. *card*(*A* ∩ {..<*n*})/*n*) < *l*
using *assms upper-asymptotic-density-in-01*(*1*) *ereal-less-ereal-Ex* **by** *auto*
then have *eventually* (λ*n*. *card*(*A* ∩ {..<*n*})/*n* < *ereal l*) *sequentially*
using *Limsup-lessD* **by** *blast*
then have *eventually* (λ*n*. *card*(*A* ∩ {..<*n*})/*n* < *ereal l* ∧ *n* > *0*) *sequentially*
using *eventually-gt-at-top eventually-conj* **by** *blast*
moreover have *card*(*A* ∩ {..<*n*}) < *l* ∗ *n* **if** *card*(*A* ∩ {..<*n*})/*n* < *ereal l* ∧ *n* > *0* **for** *n*
using *that* **by** (*simp add*: *divide-less-eq*)
ultimately show *eventually* (λ*n*. *card*(*A* ∩ {..<*n*}) < *l* ∗ *n*) *sequentially*
by (*simp add*: *eventually-mono*)
qed

proposition *upper-asymptotic-densityI*:
fixes *l*::*real*
assumes *eventually* (λ*n*. *card*(*A* ∩ {..<*n*}) ≤ *l* ∗ *n*) *sequentially*
shows *upper-asymptotic-density A* ≤ *l*
proof −
have *eventually* (λ*n*. *card*(*A* ∩ {..<*n*}) ≤ *l* ∗ *n* ∧ *n* > *0*) *sequentially*
using *assms eventually-gt-at-top eventually-conj* **by** *blast*
moreover have *card*(*A* ∩ {..<*n*})/*n* ≤ *ereal l* **if** *card*(*A* ∩ {..<*n*}) ≤ *l* ∗ *n* ∧ *n* > *0* **for** *n*
using *that* **by** (*simp add*: *divide-le-eq*)
ultimately have *eventually* (λ*n*. *card*(*A* ∩ {..<*n*})/*n* ≤ *ereal l*) *sequentially*
by (*simp add*: *eventually-mono*)
then have *limsup* (λ*n*. *card*(*A* ∩ {..<*n*})/*n*) ≤ *ereal l*
by (*simp add*: *Limsup-bounded*)
then have *ereal*(*upper-asymptotic-density A*) ≤ *ereal l*
using *upper-asymptotic-density-in-01*(*1*) **by** *auto*
then show *?thesis* **by** (*simp del*: *upper-asymptotic-density-in-01*)
qed

The following trivial lemma is useful to control the asymptotic density of unions.

lemma *lem-ge-sum*:

fixes *l x y*::*real*
assumes *l*>*x*+*y*
shows ∃ *lx ly*. *l* = *lx* + *ly* ∧ *lx* > *x* ∧ *ly* > *y*
proof −
define *lx ly* **where** *lx* = *x* + (*l*−(*x*+*y*))/*2* **and** *ly* = *y* + (*l*−(*x*+*y*))/*2*
have *l* = *lx* + *ly* ∧ *lx* > *x* ∧ *ly* > *y* **unfolding** *lx-def ly-def* **using** *assms* **by** *auto*
then show *?thesis* **by** *auto*
qed

The asymptotic density of a union is bounded by the sum of the asymptotic densities.

lemma *upper-asymptotic-density-union*:
upper-asymptotic-density (*A* ∪ *B*) ≤ *upper-asymptotic-density A* + *upper-asymptotic-density B*
proof −
have *upper-asymptotic-density* (*A* ∪ *B*) ≤ *l* **if** *H*: *l* > *upper-asymptotic-density A* + *upper-asymptotic-density B* **for** *l*
proof −
obtain *lA lB* **where** *l*: *l* = *lA*+*lB* **and** *lA*: *lA* > *upper-asymptotic-density A* **and** *lB*: *lB* > *upper-asymptotic-density B*
using *lem-ge-sum H* **by** *blast*
{
fix *n* **assume** *H*: *card* (*A* ∩ {..<*n*}) < *lA* ∗ *n* ∧ *card* (*B* ∩ {..<*n*}) < *lB* ∗ *n*
have *card*((*A*∪*B*) ∩ {..<*n*}) ≤ *card*(*A* ∩ {..<*n*}) + *card*(*B* ∩ {..<*n*})
by (*simp add*: *card-Un-le inf-sup-distrib2*)
also have ... ≤ *l* ∗ *n* **using** *l H* **by** (*simp add*: *ring-class.ring-distribs*(*2*))
finally have *card* ((*A*∪*B*) ∩ {..<*n*}) ≤ *l* ∗ *n* **by** *simp*
}
moreover have *eventually* (λ*n*. *card* (*A* ∩ {..<*n*}) < *lA* ∗ *n* ∧ *card* (*B* ∩ {..<*n*}) < *lB* ∗ *n*) *sequentially*
using *upper-asymptotic-densityD*[*OF lA*] *upper-asymptotic-densityD*[*OF lB*] *eventually-conj* **by** *blast*
ultimately have *eventually* (λ*n*. *card*((*A*∪*B*) ∩ {..<*n*}) ≤ *l* ∗ *n*) *sequentially*
by (*simp add*: *eventually-mono*)
then show *upper-asymptotic-density* (*A* ∪ *B*) ≤ *l* **using** *upper-asymptotic-densityI* **by** *auto*
qed
then show *?thesis* **by** (*meson dense not-le*)
qed

It follows that the asymptotic density is an increasing function for inclusion.

lemma *upper-asymptotic-density-subset*:
assumes *A* ⊆ *B*
shows *upper-asymptotic-density A* ≤ *upper-asymptotic-density B*
proof −
have *upper-asymptotic-density A* ≤ *l* **if** *l*: *l* > *upper-asymptotic-density B* **for** *l*
proof −
have *card*(*A* ∩ {..<*n*}) ≤ *card*(*B* ∩ {..<*n*}) **for** *n*

using *assms* **by** (*metis Int-lower2 Int-mono card-mono finite-lessThan finite-subset inf.left-idem*)
then have *card*($A \cap \{..<n\}$) $\leq l * n$ **if** *card*($B \cap \{..<n\}$) $< l * n$ **for** n
using *that* **by** (*meson lessThan-def less-imp-le of-nat-le-iff order-trans*)
moreover have *eventually* ($\lambda n.$ *card*($B \cap \{..<n\}$) $< l * n$) *sequentially*
using *upper-asymptotic-densityD l* **by** *simp*
ultimately have *eventually* ($\lambda n.$ *card*($A \cap \{..<n\}$) $\leq l * n$) *sequentially*
by (*simp add*: *eventually-mono*)
then show *?thesis* **using** *upper-asymptotic-densityI* **by** *auto*
qed
then show *?thesis* **by** (*meson dense not-le*)
qed

If a set has a density, then it is also its asymptotic density.

lemma *upper-asymptotic-density-lim*:
assumes ($\lambda n.$ *card*($A \cap \{..<n\}$)$/n$) $\longrightarrow l$
shows *upper-asymptotic-density* $A = l$
proof −
have ($\lambda n.$ *ereal*(*card*($A \cap \{..<n\}$)$/n$)) $\longrightarrow l$ **using** *assms* **by** *auto*
then have *limsup* ($\lambda n.$ *card*($A \cap \{..<n\}$)$/n$) $= l$
using *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *blast*
then show *?thesis* **unfolding** *upper-asymptotic-density-def* **by** *auto*
qed

If two sets are equal up to something small, i.e. a set with zero upper density, then they have the same upper density.

lemma *upper-asymptotic-density-0-diff*:
assumes $A \subseteq B$ *upper-asymptotic-density* $(B-A) = 0$
shows *upper-asymptotic-density* $A =$ *upper-asymptotic-density* B
proof −
have *upper-asymptotic-density* $B \leq$ *upper-asymptotic-density* $A +$ *upper-asymptotic-density* $(B-A)$
using *upper-asymptotic-density-union*[*of* A $B-A$] **by** (*simp add*: *assms*(*1*) *sup.absorb2*)
then have *upper-asymptotic-density* $B \leq$ *upper-asymptotic-density* A
using *assms*(*2*) **by** *simp*
then show *?thesis* **using** *upper-asymptotic-density-subset*[*OF assms*(*1*)] **by** *simp*
qed

lemma *upper-asymptotic-density-0-Delta*:
assumes *upper-asymptotic-density* ($A \Delta B$) $= 0$
shows *upper-asymptotic-density* $A =$ *upper-asymptotic-density* B
proof −
have $A- (A \cap B) \subseteq A \Delta B$ $B- (A \cap B) \subseteq A \Delta B$
using *assms*(*1*) **by** (*auto simp add*: *Diff-Int Un-infinite*)
then have *upper-asymptotic-density* ($A - (A \cap B)$) $= 0$
upper-asymptotic-density ($B - (A \cap B)$) $= 0$
using *upper-asymptotic-density-subset assms*(*1*) *upper-asymptotic-density-in-01*(*3*)
by (*metis inf.absorb-iff2 inf.orderE*)+

then have *upper-asymptotic-density* $(A \cap B)$ = *upper-asymptotic-density A*
upper-asymptotic-density $(A \cap B)$ = *upper-asymptotic-density B*
using *upper-asymptotic-density-0-diff* **by** *auto*
then show *?thesis* **by** *simp*
qed

Finite sets have vanishing upper asymptotic density.

lemma *upper-asymptotic-density-finite*:
assumes *finite A*
shows *upper-asymptotic-density A* = *0*
proof −
have $(\lambda n.\ card(A \cap \{..<n\})/n) \longrightarrow 0$
proof (*rule tendsto-sandwich*[**where** *?f* = $\lambda n.$ *0* **and** *?h* = $\lambda(n::nat).$ *card A* / *n*])
have $card(A \cap \{..<n\})/n \leq card\ A\ /\ n$ **if** $n>0$ **for** *n*
using *that* ‹*finite A*› **by** (*simp add*: *card-mono divide-right-mono*)
then show *eventually* $(\lambda n.\ card(A \cap \{..<n\})/n \leq card\ A\ /\ n)$ *sequentially*
by (*simp add*: *eventually-at-top-dense*)
have $(\lambda n.\ real\ (card\ A) * (1\ /\ real\ n)) \longrightarrow real(card\ A) * 0$
by (*intro tendsto-intros*)
then show $(\lambda n.\ real\ (card\ A)\ /\ real\ n) \longrightarrow 0$ **by** *auto*
qed (*auto*)
then show *upper-asymptotic-density A* = *0* **using** *upper-asymptotic-density-lim* **by** *auto*
qed

In particular, bounded intervals have zero upper density.

lemma *upper-asymptotic-density-bdd-interval* [*simp*]:
upper-asymptotic-density {} = *0*
upper-asymptotic-density $\{..N\}$ = *0*
upper-asymptotic-density $\{..<N\}$ = *0*
upper-asymptotic-density $\{n..N\}$ = *0*
upper-asymptotic-density $\{n..<N\}$ = *0*
upper-asymptotic-density $\{n<..N\}$ = *0*
upper-asymptotic-density $\{n<..<N\}$ = *0*
by (*auto intro*!: *upper-asymptotic-density-finite*)

The density of a finite union is bounded by the sum of the densities.

lemma *upper-asymptotic-density-finite-Union*:
assumes *finite I*
shows *upper-asymptotic-density* $(\bigcup i \in I.\ A\ i) \leq (\sum i \in I.$ *upper-asymptotic-density* $(A\ i))$
using *assms* **apply** (*induction I rule*: *finite-induct*)
using *order-trans*[*OF upper-asymptotic-density-union*] **by** *auto*

It is sometimes useful to compute the asymptotic density by shifting a little bit the set: this only makes a finite difference that vanishes when divided by n.

lemma *upper-asymptotic-density-shift*:
fixes k::*nat* **and** l::*int*
shows *ereal*(*upper-asymptotic-density* A) = *limsup* (λn. *card*($A \cap \{k..nat(n+l)\}$) / n)
proof −
define C **where** $C = k+2*nat(abs(l))+1$
have ∗: (λn. $C*(1/n)$) ⟶ *real* $C * 0$
by (*intro tendsto-intros*)
have *l0*: *limsup* (λn. C/n) = 0
apply (*rule lim-imp-Limsup*, *simp*) **using** ∗ **by** (*simp add*: *zero-ereal-def*)

have *card*($A \cap \{k..nat(n+l)\}$) / $n \leq$ *card* ($A \cap \{..<n\}$)/n + C/n **for** n
proof −
have *card*($A \cap \{k..nat(n+l)\}$) ≤ *card* ($A \cap \{..<n\} \cup \{n..n + nat(abs(l))\}$)
by (*rule card-mono*, *auto*)
also have ... ≤ *card* ($A \cap \{..<n\}$) + *card* $\{n..n + nat(abs(l))\}$
by (*rule card-Un-le*)
also have ... ≤ *card* ($A \cap \{..<n\}$) + *real* C
unfolding *C-def* **by** *auto*
finally have *card*($A \cap \{k..nat(n+l)\}$) / n ≤ (*card* ($A \cap \{..<n\}$) + *real* C) /n
by (*simp add*: *divide-right-mono*)
also have ... = *card* ($A \cap \{..<n\}$)/n + C/n
using *add-divide-distrib* **by** *auto*
finally show *?thesis*
by *auto*
qed
then have *limsup* (λn. *card*($A \cap \{k..nat(n+l)\}$) / n) ≤ *limsup* (λn. *card* ($A \cap \{..<n\}$)/n + *ereal*(C/n))
by (*simp add*: *Limsup-mono*)
also have ... ≤ *limsup* (λn. *card* ($A \cap \{..<n\}$)/n) + *limsup* (λn. C/n)
by (*rule ereal-limsup-add-mono*)
finally have *a*: *limsup* (λn. *card*($A \cap \{k..nat(n+l)\}$) / n) ≤ *limsup* (λn. *card* ($A \cap \{..<n\}$)/n)
using *l0* **by** *simp*

have *card* ($A \cap \{..<n\}$) / n ≤ *card* ($A \cap \{k..nat(n+l)\}$)/n + C/n **for** n
proof −
have *card* ($\{..<k\} \cup \{n-nat(abs(l))..n + nat(abs(l))\}$) ≤ *card* $\{..<k\}$ + *card* $\{n-nat(abs(l))..n + nat(abs(l))\}$
by (*rule card-Un-le*)
also have ... ≤ $k + 2*nat(abs(l)) + 1$ **by** *auto*
finally have ∗: *card* ($\{..<k\} \cup \{n-nat(abs(l))..n + nat(abs(l))\}$) ≤ C **unfolding** *C-def* **by** *blast*

have *card*($A \cap \{..<n\}$) ≤ *card* ($A \cap \{k..nat(n+l)\} \cup (\{..<k\} \cup \{n-nat(abs(l))..n + nat(abs(l))\})$)
by (*rule card-mono*, *auto*)
also have ... ≤ *card* ($A \cap \{k..nat(n+l)\}$) + *card* ($\{..<k\} \cup \{n-nat(abs(l))..n + nat(abs(l))\}$)

by (*rule card-Un-le*)
also have ... ≤ *card* (*A* ∩ {*k*..*nat*(*n*+*l*)}) + *C*
using ∗ **by** *auto*
finally have *card* (*A* ∩ {..<*n*}) / *n* ≤ (*card* (*A* ∩ {*k*..*nat*(*n*+*l*)}) + *real C*)/*n*
by (*simp add*: *divide-right-mono*)
also have ... = *card* (*A* ∩ {*k*..*nat*(*n*+*l*)})/*n* + *C*/*n*
using *add-divide-distrib* **by** *auto*
finally show *?thesis*
by *auto*
qed
then have *limsup* (λ*n*. *card*(*A* ∩ {..<*n*}) / *n*) ≤ *limsup* (λ*n*. *card* (*A* ∩ {*k*..*nat*(*n*+*l*)})/*n* + *ereal*(*C*/*n*))
by (*simp add*: *Limsup-mono*)
also have ... ≤ *limsup* (λ*n*. *card* (*A* ∩ {*k*..*nat*(*n*+*l*)})/*n*) + *limsup* (λ*n*. *C*/*n*)
by (*rule ereal-limsup-add-mono*)
finally have *limsup* (λ*n*. *card*(*A* ∩ {..<*n*}) / *n*) ≤ *limsup* (λ*n*. *card* (*A* ∩ {*k*..*nat*(*n*+*l*)})/*n*)
using *l0* **by** *simp*
then have *limsup* (λ*n*. *card*(*A* ∩ {..<*n*}) / *n*) = *limsup* (λ*n*. *card* (*A* ∩ {*k*..*nat*(*n*+*l*)})/*n*)
using *a* **by** *auto*
then show *?thesis* **using** *upper-asymptotic-density-in-01*(*1*) **by** *auto*
qed

Upper asymptotic density is measurable.

lemma *upper-asymptotic-density-meas* [*measurable*]:
assumes [*measurable*]: ⋀(*n*::*nat*). *Measurable.pred M* (*P n*)
shows (λ*x*. *upper-asymptotic-density* {*n*. *P n x*}) ∈ *borel-measurable M*
unfolding *upper-asymptotic-density-def* **by** *auto*

A finite union of sets with zero upper density still has zero upper density.

lemma *upper-asymptotic-density-zero-union*:
assumes *upper-asymptotic-density A* = *0* *upper-asymptotic-density B* = *0*
shows *upper-asymptotic-density* (*A* ∪ *B*) = *0*
using *upper-asymptotic-density-in-01*(*3*)[*of A* ∪ *B*] *upper-asymptotic-density-union*[*of A B*] **unfolding** *assms* **by** *auto*

lemma *upper-asymptotic-density-zero-finite-Union*:
assumes *finite I* ⋀*i*. *i* ∈ *I* ⟹ *upper-asymptotic-density* (*A i*) = *0*
shows *upper-asymptotic-density* (⋃ *i*∈*I*. *A i*) = *0*
using *assms* **by** (*induction rule*: *finite-induct*, *auto intro*!: *upper-asymptotic-density-zero-union*)

The union of sets with small asymptotic densities can have a large density: think of $A_n = [0, n]$, it has density 0, but the union of the A_n has density 1. However, if one only wants a set which contains each A_n eventually, then one can obtain a "union" that has essentially the same density as each A_n. This is often used as a replacement for the diagonal argument in density arguments: if for each n one can find a set A_n with good properties and a controlled density, then their "union" will have the same properties (eventually) and a controlled density.

proposition *upper-asymptotic-density-incseq-Union*:
assumes $\bigwedge(n::nat).$ *upper-asymptotic-density* $(A\ n) \le l$ *incseq A*
shows $\exists B.$ *upper-asymptotic-density* $B \le l \wedge (\forall n.\ \exists N.\ A\ n \cap \{N..\} \subseteq B)$
proof −
have *A*: $\exists N.\ \forall j \ge N.$ *card* $(A\ k \cap \{..<j\}) < (l + (1/2)\hat{\ }k) * j$ **for** *k*
proof −
have ∗: *upper-asymptotic-density* $(A\ k) < l + (1/2)\hat{\ }k$ **using** *assms*(*1*)[*of k*]
by (*metis add.right-neutral add-mono-thms-linordered-field*(*4*) *less-divide-eq-numeral1* (*1*) *mult-zero-left zero-less-one zero-less-power*)
show *?thesis*
using *upper-asymptotic-densityD*[*OF* ∗] **unfolding** *eventually-sequentially* **by** *auto*
qed
have $\exists N.\ \forall k.\ (\forall j \ge N\ k.$ *card* $(A\ k \cap \{..<j\}) \le (l+(1/2)\hat{\ }k) * j) \wedge N\ (Suc\ k) > N\ k$
proof (*rule dependent-nat-choice*)
fix *x k*::*nat*
obtain *N* **where** *N*: $\forall j{\ge}N.$ *real* (*card* $(A\ (Suc\ k) \cap \{..<j\})) \le (l + (1\ /\ 2)\ \hat{\ }\ Suc\ k) *$ *real j*
using *A*[*of Suc k*] *less-imp-le* **by** *auto*
show $\exists y.\ (\forall j{\ge}y.$ *real* (*card* $(A(Suc\ k) \cap \{..<j\})) \le (l + (1\ /\ 2)\ \hat{\ }\ Suc\ k) *$ *real j*$) \wedge x < y$
apply (*rule exI*[*of - max x N + 1*]) **using** *N* **by** *auto*
next
show $\exists x.\ \forall j{\ge}x.$ *real* (*card* $((A\ 0) \cap \{..<j\})) \le (l + (1\ /\ 2)\ \hat{\ }\ 0) *$ *real j*
using *A*[*of 0*] *less-imp-le* **by** *auto*
qed

Here is the choice of the good waiting function N

then obtain *N* **where** *N*: $\bigwedge k\ j.\ j \ge N\ k \Longrightarrow$ *card* $(A\ k \cap \{..<j\}) \le (l + (1/2)\hat{\ }k) * j\ \bigwedge k.\ N\ (Suc\ k) > N\ k$
by *blast*
then have *strict-mono N* **by** (*simp add*: *strict-monoI-Suc*)
have *Nmono*: $N\ k < N\ l$ **if** $k < l$ **for** *k l*
using *N*(*2*) **by** (*simp add*: *lift-Suc-mono-less that*)

We can now define the global bad set B.

define *B* **where** $B = (\bigcup k.\ A\ k \cap \{N\ k..\})$

We will now show that it also has density at most l.

have *Bcard*: *card* $(B \cap \{..<n\}) \le (l+(1/2)\hat{\ }k) * n$ **if** $N\ k \le n\ n < N\ (Suc\ k)$ **for** *n k*
proof −
have $\{N\ j..<n\} = \{\}$ **if** $j \in \{k<..\}$ **for** *j*
using ‹$n < N\ (Suc\ k)$› *that* **by** (*auto*, *meson* ‹*strict-mono N*› *less-trans not-less-eq strict-mono-less*)
then have ∗: $(\bigcup j{\in}\{k<..\}.\ A\ j \cap \{N\ j..<n\}) = \{\}$ **by** *force*

have $B \cap \{..<n\} = (\bigcup j.\ A\ j \cap \{N\ j..<n\})$

```
      unfolding B-def by auto
    also have ... = (⋃j ∈ {..k}. A j ∩ {N j..<n}) ∪ (⋃j∈{k<..}. A j ∩ {N j..<n})
      unfolding UN-Un [symmetric] by (rule arg-cong [of - - Union]) auto
    also have ... = (⋃j ∈ {..k}. A j ∩ {N j..<n})
      unfolding * by simp
    also have ... ⊆ (⋃j ∈ {..k}. A k ∩ {..<n})
      using ‹incseq A› unfolding incseq-def by (auto intro!: UN-mono)
    also have ... = A k ∩ {..<n}
      by simp
    finally have card (B ∩ {..<n}) ≤ card (A k ∩ {..<n})
      by (rule card-mono[rotated], auto)
    then show ?thesis
      using N(1)[OF ‹n ≥ N k›] by simp
  qed
  have eventually (λn. card (B ∩ {..<n}) ≤ a ∗ n) sequentially if l < a for a::real
  proof −
    have eventually (λk. (l+(1/2)^k) < a) sequentially
     apply (rule order-tendstoD[of - l+0], intro tendsto-intros) using that by auto
    then obtain k where l + (1/2)^k < a
      unfolding eventually-sequentially by auto
    have card (B ∩ {..<n}) ≤ a ∗ n if n ≥ N k + 1for n
    proof −
      have n ≥ N k n ≥ 1 using that by auto
      have {p. n ≥ N p} ⊆ {..n}
        using ‹strict-mono N› dual-order.trans seq-suble by blast
      then have ∗: finite {p. n ≥ N p} {p. n ≥ N p} ≠ {}
        using ‹n ≥ N k› finite-subset by auto
      define m where m = Max {p. n ≥ N p}
      have k ≤ m
        unfolding m-def using Max-ge[OF ∗(1), of k] that by auto
      have N m ≤ n
        unfolding m-def using Max-in[OF ∗] by auto
      have Suc m ∉ {p. n ≥ N p}
        unfolding m-def using ∗ Max-ge Suc-n-not-le-n by blast
      then have n < N (Suc m) by simp
      have card (B ∩ {..<n}) ≤ (l+(1/2)^m) ∗ n
        using Bcard[OF ‹N m ≤ n› ‹n < N (Suc m)›] by simp
      also have ... ≤ (l + (1/2)^k) ∗ n
      apply (rule mult-right-mono) using ‹k ≤ m› by (auto simp add: power-decreasing)
      also have ... ≤ a ∗ n
        using ‹l + (1/2)^k < a› ‹n ≥ 1› by auto
      finally show ?thesis by auto
    qed
    then show ?thesis unfolding eventually-sequentially by auto
  qed
  then have upper-asymptotic-density B ≤ a if a > l for a
    using upper-asymptotic-densityI that by auto
  then have upper-asymptotic-density B ≤ l
    by (meson dense not-le)
```

moreover have ∃ *N*. *A n* ∩ {*N*..} ⊆ *B* **for** *n*
apply (*rule exI*[*of* - *N n*]) **unfolding** *B-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

When the sequence of sets is not increasing, one can only obtain a set whose density is bounded by the sum of the densities.

proposition *upper-asymptotic-density-Union*:
assumes *summable* (*λn*. *upper-asymptotic-density* (*A n*))
shows ∃ *B*. *upper-asymptotic-density B* ≤ (∑ *n*. *upper-asymptotic-density* (*A n*)) ∧ (∀ *n*. ∃ *N*. *A n* ∩ {*N*..} ⊆ *B*)
proof −
define *C* **where** *C* = (*λn*. (⋃ *i*≤*n*. *A i*))
have *C1*: *incseq C*
unfolding *C-def incseq-def* **by** *fastforce*
have *C2*: *upper-asymptotic-density* (*C k*) ≤ (∑ *n*. *upper-asymptotic-density* (*A n*)) **for** *k*
proof −
have *upper-asymptotic-density* (*C k*) ≤ (∑ *i*≤*k*. *upper-asymptotic-density* (*A i*))
unfolding *C-def* **by** (*rule upper-asymptotic-density-finite-Union*, *auto*)
also have ... ≤ (∑ *i*. *upper-asymptotic-density* (*A i*))
apply (*rule sum-le-suminf*[*OF assms*]) **using** *upper-asymptotic-density-in-01* **by** *auto*
finally show *?thesis* **by** *simp*
qed
obtain *B* **where** *B*: *upper-asymptotic-density B* ≤ (∑ *n*. *upper-asymptotic-density* (*A n*))
⋀*n*. ∃ *N*. *C n* ∩ {*N*..} ⊆ *B*
using *upper-asymptotic-density-incseq-Union*[*OF C2 C1*] **by** *blast*
have ∃ *N*. *A n* ∩ {*N*..} ⊆ *B* **for** *n*
using *B*(*2*)[*of n*] **unfolding** *C-def* **by** *auto*
then show *?thesis* **using** *B*(*1*) **by** *blast*
qed

A particular case of the previous proposition, often useful, is when all sets have density zero.

proposition *upper-asymptotic-density-zero-Union*:
assumes ⋀*n*::*nat*. *upper-asymptotic-density* (*A n*) = *0*
shows ∃ *B*. *upper-asymptotic-density B* = *0* ∧ (∀ *n*. ∃ *N*. *A n* ∩ {*N*..} ⊆ *B*)
proof −
have ∃ *B*. *upper-asymptotic-density B* ≤ (∑ *n*. *upper-asymptotic-density* (*A n*)) ∧ (∀ *n*. ∃ *N*. *A n* ∩ {*N*..} ⊆ *B*)
apply (*rule upper-asymptotic-density-Union*) **unfolding** *assms* **by** *auto*
then obtain *B* **where** *upper-asymptotic-density B* ≤ *0* ⋀*n*. ∃ *N*. *A n* ∩ {*N*..} ⊆ *B*
unfolding *assms* **by** *auto*
then show *?thesis*
using *upper-asymptotic-density-in-01*(*3*)[*of B*] **by** *auto*

qed

3.2 Lower asymptotic densities

The lower asymptotic density of a set of natural numbers is defined just as its upper asymptotic density but using a liminf instead of a limsup. Its properties are proved exactly in the same way.

definition *lower-asymptotic-density*::*nat set* ⇒ *real*
 where *lower-asymptotic-density A* = *real-of-ereal*(*liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*))

lemma *lower-asymptotic-density-in-01*:
 ereal(*lower-asymptotic-density A*) = *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*)
 lower-asymptotic-density A ≤ *1*
 lower-asymptotic-density A ≥ *0*
proof −
 {
  fix *n*::*nat* **assume** *n*>*0*
  have *card*(*A* ∩ {..<*n*}) ≤ *n* **by** (*metis card-lessThan Int-lower2 card-mono finite-lessThan*)
  then have *card*(*A* ∩ {..<*n*}) / *n* ≤ *ereal 1* **using** ‹*n*>*0*› **by** *auto*
 }
 then have *eventually* (*λn. card*(*A* ∩ {..<*n*}) / *n* ≤ *ereal 1*) *sequentially*
  by (*simp add*: *eventually-at-top-dense*)
 then have *limsup* (*λn. card*(*A* ∩ {..<*n*})/*n*) ≤ *1* **by** (*simp add*: *Limsup-const Limsup-bounded*)
 then have *a*: *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*) ≤ *1*
  by (*meson Liminf-le-Limsup less-le-trans not-le sequentially-bot*)

 have *card*(*A* ∩ {..<*n*}) / *n* ≥ *ereal 0* **for** *n* **by** *auto*
 then have *b*: *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*) ≥ *0* **by** (*simp add*: *le-Liminf-iff less-le-trans*)

 have *abs*(*liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*)) ≠ ∞ **using** *a b* **by** *auto*
 then show *ereal*(*lower-asymptotic-density A*) = *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*)
  unfolding *lower-asymptotic-density-def* **by** *auto*
 show *lower-asymptotic-density A* ≤ *1* *lower-asymptotic-density A* ≥ *0* **unfolding** *lower-asymptotic-density-def*
  using *a b* **by** (*auto simp add*: *real-of-ereal-le-1 real-of-ereal-pos*)
qed

The lower asymptotic density is bounded by the upper one. When they coincide, $Card(A \cap [0, n))/n$ converges to this common value.

lemma *lower-asymptotic-density-le-upper*:
 lower-asymptotic-density A ≤ *upper-asymptotic-density A*
using *lower-asymptotic-density-in-01*(*1*) *upper-asymptotic-density-in-01*(*1*)
by (*metis* (*mono-tags*, *lifting*) *Liminf-le-Limsup ereal-less-eq*(*3*) *sequentially-bot*)

lemma *lower-asymptotic-density-eq-upper*:

assumes *lower-asymptotic-density A = l upper-asymptotic-density A = l*
shows (*λn. card*(*A* ∩ {..<*n*})/*n*) ⟶ *l*
apply (*rule limsup-le-liminf-real*)
using *upper-asymptotic-density-in-01*(*1*)[*of A*] *lower-asymptotic-density-in-01*(*1*)[*of A*] *assms* **by** *auto*

In particular, when a set has a zero upper density, or a lower density one, then this implies the corresponding convergence of $Card(A \cap [0, n))/n$.

lemma *upper-asymptotic-density-zero-lim*:
assumes *upper-asymptotic-density A = 0*
shows (*λn. card*(*A* ∩ {..<*n*})/*n*) ⟶ *0*
apply (*rule lower-asymptotic-density-eq-upper*)
using *assms lower-asymptotic-density-le-upper*[*of A*] *lower-asymptotic-density-in-01*(*3*)[*of A*] **by** *auto*

lemma *lower-asymptotic-density-one-lim*:
assumes *lower-asymptotic-density A = 1*
shows (*λn. card*(*A* ∩ {..<*n*})/*n*) ⟶ *1*
apply (*rule lower-asymptotic-density-eq-upper*)
using *assms lower-asymptotic-density-le-upper*[*of A*] *upper-asymptotic-density-in-01*(*2*)[*of A*] **by** *auto*

The lower asymptotic density of a set is 1 minus the upper asymptotic density of its complement. Hence, most statements about one of them follow from statements about the other one, although we will rather give direct proofs as they are not more complicated.

lemma *lower-upper-asymptotic-density-complement*:
lower-asymptotic-density A = 1 − upper-asymptotic-density (*UNIV − A*)
proof −
{
fix *n* **assume** *n*>(*0*::*nat*)
have {..<*n*} ∩ *UNIV* − (*UNIV* − ({..<*n*} − (*UNIV* − *A*))) = {..<*n*} ∩ *A*
by *blast*
moreover have {..<*n*} ∩ *UNIV* ∩ (*UNIV* − ({..<*n*} − (*UNIV* − *A*))) = (*UNIV* − *A*) ∩ {..<*n*}
by *blast*
ultimately have *card* (*A* ∩ {..<*n*}) = *n* − *card*((*UNIV*−*A*) ∩ {..<*n*})
by (*metis* (*no-types*) *Int-commute card-Diff-subset-Int card-lessThan finite-Int finite-lessThan inf-top-right*)
then have *card* (*A* ∩ {..<*n*})/*n* = (*real n* − *card*((*UNIV*−*A*) ∩ {..<*n*})) / *n*
by (*metis Int-lower2 card-lessThan card-mono finite-lessThan of-nat-diff*)
then have *card* (*A* ∩ {..<*n*})/*n* = *ereal 1* − *card*((*UNIV*−*A*) ∩ {..<*n*})/*n*
using ‹*n*>*0*› **by** (*simp add*: *diff-divide-distrib*)
}
then have *eventually* (*λn. card* (*A* ∩ {..<*n*})/*n* = *ereal 1* − *card*((*UNIV*−*A*) ∩ {..<*n*})/*n*) *sequentially*
by (*simp add*: *eventually-at-top-dense*)
then have *liminf* (*λn. card* (*A* ∩ {..<*n*})/*n*) = *liminf* (*λn. ereal 1* − *card*((*UNIV*−*A*) ∩ {..<*n*})/*n*)

by (*rule Liminf-eq*)
also have ... = *ereal 1* − *limsup* (*λn. card*((*UNIV*−*A*) ∩ {..<*n*})/*n*)
by (*rule liminf-ereal-cminus*, *simp*)
finally show *?thesis* **unfolding** *lower-asymptotic-density-def*
by (*metis ereal-minus*(*1*) *real-of-ereal.simps*(*1*) *upper-asymptotic-density-in-01*(*1*))
qed

proposition *lower-asymptotic-densityD*:
fixes *l*::*real*
assumes *lower-asymptotic-density A* > *l*
shows *eventually* (*λn. card*(*A* ∩ {..<*n*}) > *l* ∗ *n*) *sequentially*
proof −
have *ereal*(*lower-asymptotic-density A*) > *l* **using** *assms* **by** *auto*
then have *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*) > *l*
using *lower-asymptotic-density-in-01*(*1*) **by** *auto*
then have *eventually* (*λn. card*(*A* ∩ {..<*n*})/*n* > *ereal l*) *sequentially*
using *less-LiminfD* **by** *blast*
then have *eventually* (*λn. card*(*A* ∩ {..<*n*})/*n* > *ereal l* ∧ *n* > *0*) *sequentially*
using *eventually-gt-at-top eventually-conj* **by** *blast*
moreover have *card*(*A* ∩ {..<*n*}) > *l* ∗ *n* **if** *card*(*A* ∩ {..<*n*})/*n* > *ereal l* ∧ *n* > *0* **for** *n*
using *that divide-le-eq ereal-less-eq*(*3*) *less-imp-of-nat-less not-less of-nat-eq-0-iff* **by** *fastforce*
ultimately show *eventually* (*λn. card*(*A* ∩ {..<*n*}) > *l* ∗ *n*) *sequentially*
by (*simp add*: *eventually-mono*)
qed

proposition *lower-asymptotic-densityI*:
fixes *l*::*real*
assumes *eventually* (*λn. card*(*A* ∩ {..<*n*}) ≥ *l* ∗ *n*) *sequentially*
shows *lower-asymptotic-density A* ≥ *l*
proof −
have *eventually* (*λn. card*(*A* ∩ {..<*n*}) ≥ *l* ∗ *n* ∧ *n* > *0*) *sequentially*
using *assms eventually-gt-at-top eventually-conj* **by** *blast*
moreover have *card*(*A* ∩ {..<*n*})/*n* ≥ *ereal l* **if** *card*(*A* ∩ {..<*n*}) ≥ *l* ∗ *n* ∧ *n* > *0* **for** *n*
using *that* **by** (*meson ereal-less-eq*(*3*) *not-less of-nat-0-less-iff pos-divide-less-eq*)
ultimately have *eventually* (*λn. card*(*A* ∩ {..<*n*})/*n* ≥ *ereal l*) *sequentially*
by (*simp add*: *eventually-mono*)
then have *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*) ≥ *ereal l*
by (*simp add*: *Liminf-bounded*)
then have *ereal*(*lower-asymptotic-density A*) ≥ *ereal l*
using *lower-asymptotic-density-in-01*(*1*) **by** *auto*
then show *?thesis* **by** *auto*
qed

One can control the asymptotic density of an intersection in terms of the asymptotic density of each component

lemma *lower-asymptotic-density-intersection*:

lower-asymptotic-density A + *lower-asymptotic-density B* ≤ *lower-asymptotic-density* (*A* ∩ *B*) + *1*
using *upper-asymptotic-density-union*[*of UNIV* − *A UNIV* − *B*]
unfolding *lower-upper-asymptotic-density-complement* **by** (*auto simp add*: *algebra-simps Diff-Int*)

lemma *lower-asymptotic-density-subset*:
assumes *A* ⊆ *B*
shows *lower-asymptotic-density A* ≤ *lower-asymptotic-density B*
using *upper-asymptotic-density-subset*[*of UNIV*−*B UNIV*−*A*] *assms*
unfolding *lower-upper-asymptotic-density-complement* **by** *auto*

lemma *lower-asymptotic-density-lim*:
assumes (*λn. card*(*A* ∩ {..<*n*})/*n*) ⟶ *l*
shows *lower-asymptotic-density A* = *l*
proof −
have (*λn. ereal*(*card*(*A* ∩ {..<*n*})/*n*)) ⟶ *l* **using** *assms* **by** *auto*
then have *liminf* (*λn. card*(*A* ∩ {..<*n*})/*n*) = *l*
using *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *blast*
then show *?thesis* **unfolding** *lower-asymptotic-density-def* **by** *auto*
qed

lemma *lower-asymptotic-density-finite*:
assumes *finite A*
shows *lower-asymptotic-density A* = *0*
using *lower-asymptotic-density-in-01*(*3*) *upper-asymptotic-density-finite*[*OF assms*] *lower-asymptotic-density-le-upper*
by (*metis antisym-conv*)

In particular, bounded intervals have zero lower density.

lemma *lower-asymptotic-density-bdd-interval* [*simp*]:
lower-asymptotic-density {} = *0*
lower-asymptotic-density {..*N*} = *0*
lower-asymptotic-density {..<*N*} = *0*
lower-asymptotic-density {*n*..*N*} = *0*
lower-asymptotic-density {*n*..<*N*} = *0*
lower-asymptotic-density {*n*<..*N*} = *0*
lower-asymptotic-density {*n*<..<*N*} = *0*
by (*auto intro*!: *lower-asymptotic-density-finite*)

Conversely, unbounded intervals have density 1.

lemma *lower-asymptotic-density-infinite-interval* [*simp*]:
lower-asymptotic-density {*N*..} = *1*
lower-asymptotic-density {*N*<..} = *1*
lower-asymptotic-density UNIV = *1*
proof −
have *UNIV* − {*N*..} = {..<*N*} **by** *auto*
then show *lower-asymptotic-density* {*N*..} = *1*
by (*auto simp add*: *lower-upper-asymptotic-density-complement*)

have *UNIV* − {*N*<..} = {..*N*} **by** *auto*
then show *lower-asymptotic-density* {*N*<..} = *1*
by (*auto simp add*: *lower-upper-asymptotic-density-complement*)
show *lower-asymptotic-density UNIV* = *1*
by (*auto simp add*: *lower-upper-asymptotic-density-complement*)
qed

lemma *upper-asymptotic-density-infinite-interval* [*simp*]:
upper-asymptotic-density {*N*..} = *1*
upper-asymptotic-density {*N*<..} = *1*
upper-asymptotic-density UNIV = *1*
by (*metis antisym upper-asymptotic-density-in-01* (*2*) *lower-asymptotic-density-infinite-interval lower-asymptotic-density-le-upper*)+

The intersection of sets with lower density one still has lower density one.

lemma *lower-asymptotic-density-one-intersection*:
assumes *lower-asymptotic-density A* = *1* *lower-asymptotic-density B* = *1*
shows *lower-asymptotic-density* (*A* ∩ *B*) = *1*
using *lower-asymptotic-density-in-01* (*2*)[*of A* ∩ *B*] *lower-asymptotic-density-intersection*[*of A B*] **unfolding** *assms* **by** *auto*

lemma *lower-asymptotic-density-one-finite-Intersection*:
assumes *finite I* ⋀*i*. *i* ∈ *I* ⟹ *lower-asymptotic-density* (*A i*) = *1*
shows *lower-asymptotic-density* (⋂ *i*∈*I*. *A i*) = *1*
using *assms* **by** (*induction rule*: *finite-induct*, *auto intro*!: *lower-asymptotic-density-one-intersection*)

As for the upper asymptotic density, there is a modification of the intersection, akin to the diagonal argument in this context, for which the "intersection" of sets with large lower density still has large lower density.

proposition *lower-asymptotic-density-decseq-Inter*:
assumes ⋀(*n*::*nat*). *lower-asymptotic-density* (*A n*) ≥ *l* *decseq A*
shows ∃ *B*. *lower-asymptotic-density B* ≥ *l* ∧ (∀ *n*. ∃ *N*. *B* ∩ {*N*..} ⊆ *A n*)
proof −
define *C* **where** *C* = (λ*n*. *UNIV* − *A n*)
have ∗: *upper-asymptotic-density* (*C n*) ≤ *1* − *l* **for** *n*
using *assms*(*1*)[*of n*] **unfolding** *C-def lower-upper-asymptotic-density-complement*[*of A n*] **by** *auto*
have ∗∗: *incseq C*
using *assms*(*2*) **unfolding** *C-def incseq-def decseq-def* **by** *auto*
obtain *D* **where** *D*: *upper-asymptotic-density D* ≤ *1* − *l* ⋀*n*. ∃ *N*. *C n* ∩ {*N*..} ⊆ *D*
using *upper-asymptotic-density-incseq-Union*[*OF* ∗ ∗∗] **by** *blast*
define *B* **where** *B* = *UNIV* − *D*
have *lower-asymptotic-density B* ≥ *l*
using *D*(*1*) *lower-upper-asymptotic-density-complement*[*of B*] **by** (*simp add*: *double-diff B-def*)
moreover have ∃ *N*. *B* ∩ {*N*..} ⊆ *A n* **for** *n*
using *D*(*2*)[*of n*] **unfolding** *B-def C-def* **by** *auto*
ultimately show *?thesis* **by** *auto*

qed

In the same way, the modified intersection of sets of density 1 still has density one, and is eventually contained in each of them.

proposition *lower-asymptotic-density-one-Inter*:
  assumes $\bigwedge n::nat.$ *lower-asymptotic-density* $(A\ n) = 1$
  shows $\exists B.$ *lower-asymptotic-density* $B = 1 \wedge (\forall n.\ \exists N.\ B \cap \{N..\} \subseteq A\ n)$
proof −
  define C **where** $C = (\lambda n.\ UNIV - A\ n)$
  have $*$: *upper-asymptotic-density* $(C\ n) = 0$ **for** n
   using *assms*(*1*)[*of n*] **unfolding** *C-def lower-upper-asymptotic-density-complement*[*of A n*] **by** *auto*
  obtain D **where** D: *upper-asymptotic-density* $D = 0$ $\bigwedge n.\ \exists N.\ C\ n \cap \{N..\} \subseteq D$
    using *upper-asymptotic-density-zero-Union*[*OF* $*$] **by** *force*
  define B **where** $B = UNIV - D$
  have *lower-asymptotic-density* $B = 1$
    using *D*(*1*) *lower-upper-asymptotic-density-complement*[*of B*] **by** (*simp add*: *double-diff B-def*)
  moreover have $\exists N.\ B \cap \{N..\} \subseteq A\ n$ **for** n
    using *D*(*2*)[*of n*] **unfolding** *B-def C-def* **by** *auto*
  ultimately show *?thesis* **by** *auto*
qed

Sets with density 1 play an important role in relation to Cesaro convergence of nonnegative bounded sequences: such a sequence converges to 0 in Cesaro average if and only if it converges to 0 along a set of density 1.

The proof is not hard. Since the Cesaro average tends to 0, then given $\epsilon > 0$ the proportion of times where $u_n < \epsilon$ tends to 1, i.e., the set A_ϵ of such good times has density 1. A modified intersection (as constructed in Proposition `lower_asymptotic_density_one_Inter`) of these times has density 1, and u_n tends to 0 along this set.

theorem *cesaro-imp-density-one*:
  assumes $\bigwedge n.\ u\ n \geq (0::real)$ $(\lambda n.\ (\sum i<n.\ u\ i)/n) \longlongrightarrow 0$
  shows $\exists A.$ *lower-asymptotic-density* $A = 1 \wedge (\lambda n.\ u\ n * indicator\ A\ n) \longlongrightarrow 0$
proof −
  define B **where** $B = (\lambda e.\ \{n.\ u\ n \geq e\})$

Be is the set of bad times where $u_n \geq e$. It has density 0 thanks to the assumption of Cesaro convergence to 0.

  have A: *upper-asymptotic-density* $(B\ e) = 0$ **if** $e > 0$ **for** e
  proof −
   have $*$: *card* $(B\ e \cap \{..<n\})\ /\ n \leq (1/e) * ((\sum i\in\{..<n\}.\ u\ i)/n)$ **if** $n \geq 1$ **for** n
    proof −
     have $e *$ *card* $(B\ e \cap \{..<n\}) = (\sum i\in B\ e \cap \{..<n\}.\ e)$ **by** *auto*

also have ... ≤ (∑ *i*∈*B e* ∩ {..<*n*}. *u i*)
apply (*rule sum-mono*) **unfolding** *B-def* **by** *auto*
also have ... ≤ (∑ *i*∈{..<*n*}. *u i*)
apply (*rule sum-mono2*) **using** *assms* **by** *auto*
finally show *?thesis*
using ‹*e* > *0*› ‹*n* ≥ *1*› **by** (*auto simp add*: *divide-simps algebra-simps*)
qed
have (λ*n*. *card* (*B e* ∩ {..<*n*}) / *n*) ⟶ *0*
proof (*rule tendsto-sandwich*[*of* λ-. *0* - - λ*n*. (*1*/*e*) ∗ ((∑ *i*∈{..<*n*}. *u i*)/*n*)])
have (λ*n*. (*1*/*e*) ∗ ((∑ *i*∈{..<*n*}. *u i*)/*n*)) ⟶ (*1*/*e*) ∗ *0*
by (*intro tendsto-intros assms*)
then show (λ*n*. (*1*/*e*) ∗ ((∑ *i*∈{..<*n*}. *u i*)/*n*)) ⟶ *0* **by** *simp*
show $\forall_F$ *n in sequentially*. *real* (*card* (*B e* ∩ {..<*n*})) / *real n* ≤ *1* / *e* ∗ (*sum u* {..<*n*} / *real n*)
using ∗ **unfolding** *eventually-sequentially* **by** *auto*
qed (*auto*)
then show *?thesis*
by (*rule upper-asymptotic-density-lim*)
qed
define *C* **where** *C* = (λ*n*::*nat*. *UNIV* − *B* (((*1*::*real*)/*2*)^*n*))
have *lower-asymptotic-density* (*C n*) = *1* **for** *n*
unfolding *C-def lower-upper-asymptotic-density-complement* **by** (*simp add*: *A double-diff*)
then obtain *A* **where** *A*: *lower-asymptotic-density A* = *1* ⋀*n*. ∃*N*. *A* ∩ {*N*..} ⊆ *C n*
using *lower-asymptotic-density-one-Inter* **by** *blast*
have *E*: *eventually* (λ*n*. *u n* ∗ *indicator A n* < *e*) *sequentially* **if** *e* > *0* **for** *e*
proof −
have *eventually* (λ*n*. ((*1*::*real*)/*2*)^*n* < *e*) *sequentially*
by (*rule order-tendstoD*[*OF* - ‹*e* > *0*›], *intro tendsto-intros*, *auto*)
then obtain *n* **where** *n*: ((*1*::*real*)/*2*)^*n* < *e*
unfolding *eventually-sequentially* **by** *auto*
obtain *N* **where** *N*: *A* ∩ {*N*..} ⊆ *C n*
using *A*(*2*) **by** *blast*
have *u k* ∗ *indicator A k* < *e* **if** *k* ≥ *N* **for** *k*
proof (*cases k* ∈ *A*)
case *True*
then have *k* ∈ *C n* **using** *N that* **by** *auto*
then have *u k* < ((*1*::*real*)/*2*)^*n*
unfolding *C-def B-def* **by** *auto*
then have *u k* < *e*
using *n* **by** *auto*
then show *?thesis*
unfolding *indicator-def* **using** *True* **by** *auto*
next
case *False*
then show *?thesis*
unfolding *indicator-def* **using** ‹*e* > *0*› **by** *auto*
qed

then show *?thesis*
unfolding *eventually-sequentially* **by** *auto*
qed
have $(\lambda n.\ u\ n * indicator\ A\ n) \longrightarrow 0$
apply (*rule order-tendstoI*[*OF - E*])
unfolding *indicator-def* **using** ‹$\bigwedge n.\ u\ n \geq 0$› **by** (*simp add*: *less-le-trans*)
then show *?thesis*
using ‹*lower-asymptotic-density A = 1*› **by** *auto*
qed

The proof of the reverse implication is more direct: in the Cesaro sum, just bound the elements in A by a small ϵ, and the other ones by a uniform bound, to get a bound which is $o(n)$.

theorem *density-one-imp-cesaro*:
assumes $\bigwedge n.\ u\ n \geq (0::real)$ $\bigwedge n.\ u\ n \leq C$
lower-asymptotic-density A = 1
$(\lambda n.\ u\ n * indicator\ A\ n) \longrightarrow 0$
shows $(\lambda n.\ (\sum i<n.\ u\ i)/n) \longrightarrow 0$
proof (*rule order-tendstoI*)
fix *e*::*real* **assume** $e < 0$
have $(\sum i<n.\ u\ i)/n \geq 0$ **for** *n*
using *assms*(*1*) **by** (*simp add*: *sum-nonneg divide-simps*)
then have $(\sum i<n.\ u\ i)/n > e$ **for** *n*
using ‹$e < 0$› *less-le-trans* **by** *auto*
then show *eventually* $(\lambda n.\ (\sum i<n.\ u\ i)/n > e)$ *sequentially*
unfolding *eventually-sequentially* **by** *auto*
next
fix *e*::*real* **assume** $e > 0$
have $C \geq 0$ **using** ‹$u\ 0 \geq 0$› ‹$u\ 0 \leq C$› **by** *auto*
have *eventually* $(\lambda n.\ u\ n * indicator\ A\ n < e/4)$ *sequentially*
using *order-tendstoD*(*2*)[*OF assms*(*4*), *of e/4*] ‹$e>0$› **by** *auto*
then obtain *N* **where** *N*: $\bigwedge k.\ k \geq N \Longrightarrow u\ k * indicator\ A\ k < e/4$
unfolding *eventually-sequentially* **by** *auto*
define *B* **where** $B = UNIV - A$
have $*$: *upper-asymptotic-density B = 0*
using *assms* **unfolding** *B-def lower-upper-asymptotic-density-complement* **by** *auto*
have *eventually* $(\lambda n.\ card(B \cap \{..<n\}) < (e/(4 * (C+1))) * n)$ *sequentially*
apply (*rule upper-asymptotic-densityD*) **using** ‹$e > 0$› ‹$C \geq 0$› $*$ **by** *auto*
then obtain *M* **where** *M*: $\bigwedge n.\ n \geq M \Longrightarrow card(B \cap \{..<n\}) < (e/(4 * (C+1))) * n$
unfolding *eventually-sequentially* **by** *auto*

obtain *P*::*nat* **where** *P*: $P \geq 4 * N * C/e$
using *real-arch-simple* **by** *auto*
define *Q* **where** $Q = N + M + 1 + P$

have $(\sum i<n.\ u\ i)/n < e$ **if** $n \geq Q$ **for** *n*
proof −

```
    have n: n ≥ N n ≥ M n ≥ P n ≥ 1
      using ‹n ≥ Q› unfolding Q-def by auto
    then have n2: n ≥ 4 * N * C/e using P by auto
    have (∑ i<n. u i) ≤ (∑ i∈{..<N} ∪ ({N..<n} ∩ A) ∪ ({N..<n} − A). u i)
      by (rule sum-mono2, auto simp add: assms)
   also have ... = (∑ i∈{..<N}. u i) + (∑ i∈{N..<n} ∩ A. u i) + (∑ i∈{N..<n}
− A. u i)
      by (subst sum.union-disjoint, auto)+
    also have ... = (∑ i∈{..<N}. u i) + (∑ i∈{N..<n} ∩ A. u i * indicator A i)
+ (∑ i∈{N..<n} − A. u i)
      unfolding indicator-def by auto
    also have ... ≤ (∑ i∈{..<N}. u i) + (∑ i∈{N..<n}. u i * indicator A i) +
(∑ i∈ B ∩ {..<n}. u i)
      apply (intro add-mono sum-mono2) unfolding B-def using assms by auto
   also have ... ≤ (∑ i∈{..<N}. C) + (∑ i∈{N..<n}. e/4) + (∑ i∈B ∩ {..<n}.
C)
      apply (intro add-mono sum-mono) using assms less-imp-le[OF N] by auto
    also have ... = N * C + (n−N) * e/4 + card(B ∩ {..<n}) * C
      by auto
    also have ... ≤ n * e/4 + n * e/4 + (e/(4 * (C+1))) * n * C
      apply (intro add-mono)
       using n2 ‹e > 0› mult-right-mono[OF less-imp-le[OF M[OF ‹n ≥ M›]] ‹C
≥ 0›] by (auto simp add: divide-simps)
    also have ... ≤ n * e * 3/4
      using ‹C ≥ 0› ‹e > 0› by (simp add: divide-simps algebra-simps)
    also have ... < n * e
      using ‹n ≥ 1› ‹e > 0› by auto
    finally show ?thesis
      using ‹n ≥ 1› by (simp add: divide-simps algebra-simps)
  qed
  then show eventually (λn. (∑ i<n. u i)/n < e) sequentially
    unfolding eventually-sequentially by auto
qed

end
```

4 Measure preserving or quasi-preserving maps

```
theory Measure-Preserving-Transformations
  imports SG-Library-Complement
begin
```

Ergodic theory in general is the study of the properties of measure preserving or quasi-preserving dynamical systems. In this section, we introduce the basic definitions in this respect.

4.1 The different classes of transformations

```
definition quasi-measure-preserving::'a measure ⇒ 'b measure ⇒ ('a ⇒ 'b) set
```

where *quasi-measure-preserving M N*
  = {*f* ∈ *measurable M N*. ∀ *A* ∈ *sets N*. (*f* −' *A* ∩ *space M* ∈ *null-sets M*) = (*A* ∈ *null-sets N*)}

lemma *quasi-measure-preservingI* [*intro*]:
  assumes *f* ∈ *measurable M N*
    ⋀*A*. *A* ∈ *sets N* ⟹ (*f* −' *A* ∩ *space M* ∈ *null-sets M*) = (*A* ∈ *null-sets N*)
  shows *f* ∈ *quasi-measure-preserving M N*
using *assms* **unfolding** *quasi-measure-preserving-def* **by** *auto*

lemma *quasi-measure-preservingE*:
  assumes *f* ∈ *quasi-measure-preserving M N*
  shows *f* ∈ *measurable M N*
    ⋀*A*. *A* ∈ *sets N* ⟹ (*f* −' *A* ∩ *space M* ∈ *null-sets M*) = (*A* ∈ *null-sets N*)
using *assms* **unfolding** *quasi-measure-preserving-def* **by** *auto*

lemma *id-quasi-measure-preserving*:
  (λ*x*. *x*) ∈ *quasi-measure-preserving M M*
unfolding *quasi-measure-preserving-def* **by** *auto*

lemma *quasi-measure-preserving-composition*:
  assumes *f* ∈ *quasi-measure-preserving M N*
    g ∈ *quasi-measure-preserving N P*
  shows (λ*x*. *g*(*f x*)) ∈ *quasi-measure-preserving M P*
proof (*rule quasi-measure-preservingI*)
  have *f-meas* [*measurable*]: *f* ∈ *measurable M N* **by** (*rule quasi-measure-preservingE*(*1*)[*OF assms*(*1*)])
  have *g-meas* [*measurable*]: *g* ∈ *measurable N P* **by** (*rule quasi-measure-preservingE*(*1*)[*OF assms*(*2*)])
  then show [*measurable*]: (λ*x*. *g* (*f x*)) ∈ *measurable M P* **by** *auto*

  fix *C* **assume** [*measurable*]: *C* ∈ *sets P*
  define *B* **where** *B* = *g*−'*C* ∩ *space N*
  have [*measurable*]: *B* ∈ *sets N* **unfolding** *B-def* **by** *simp*
  have ∗: *B* ∈ *null-sets N* ⟷ *C* ∈ *null-sets P*
    unfolding *B-def* **using** *quasi-measure-preservingE*(*2*)[*OF assms*(*2*)] **by** *simp*

  define *A* **where** *A* = *f*−'*B* ∩ *space M*
  have [*measurable*]: *A* ∈ *sets M* **unfolding** *A-def* **by** *simp*
  have *A* ∈ *null-sets M* ⟷ *B* ∈ *null-sets N*
    unfolding *A-def* **using** *quasi-measure-preservingE*(*2*)[*OF assms*(*1*)] **by** *simp*

  then have *A* ∈ *null-sets M* ⟷ *C* ∈ *null-sets P* **using** ∗ **by** *simp*
  moreover have *A* = (λ*x*. *g* (*f x*)) −' *C* ∩ *space M*
    by (*auto simp add*: *A-def B-def*) (*meson f-meas measurable-space*)
  ultimately show ((λ*x*. *g* (*f x*)) −' *C* ∩ *space M* ∈ *null-sets M*) ⟷ *C* ∈ *null-sets P* **by** *simp*
qed

```
lemma quasi-measure-preserving-comp:
  assumes f ∈ quasi-measure-preserving M N
          g ∈ quasi-measure-preserving N P
  shows g o f ∈ quasi-measure-preserving M P
unfolding comp-def using assms quasi-measure-preserving-composition by blast

lemma quasi-measure-preserving-AE:
  assumes f ∈ quasi-measure-preserving M N
          AE x in N. P x
  shows AE x in M. P (f x)
proof −
  obtain A where ⋀x. x ∈ space N − A ⟹ P x A ∈ null-sets N
    using AE-E3[OF assms(2)] by blast
  define B where B = f−‘A ∩ space M
  have B ∈ null-sets M
    unfolding B-def using quasi-measure-preservingE(2)[OF assms(1)] ‹A ∈ null-sets N› by auto
  moreover have x ∈ space M − B ⟹ P (f x) for x
    using ‹⋀x. x ∈ space N − A ⟹ P x› quasi-measure-preservingE(1)[OF assms(1)]
    unfolding B-def by (metis (no-types, lifting) Diff-iff IntI measurable-space vimage-eq)
  ultimately show ?thesis using AE-not-in AE-space by force
qed

lemma quasi-measure-preserving-AE′:
  assumes f ∈ quasi-measure-preserving M N
          AE x in M. P (f x)
          {x ∈ space N. P x} ∈ sets N
  shows AE x in N. P x
proof −
  have [measurable]: f ∈ measurable M N using quasi-measure-preservingE(1)[OF assms(1)] by simp
  define U where U = {x ∈ space N. ¬(P x)}
  have [measurable]: U ∈ sets N unfolding U-def using assms(3) by auto
  have f−‘U ∩ space M = {x ∈ space M. ¬(P (f x))}
    unfolding U-def using ‹f ∈ measurable M N› by (auto, meson measurable-space)
  also have ... ∈ null-sets M
    apply (subst AE-iff-null[symmetric]) using assms by auto
  finally have U ∈ null-sets N
    using quasi-measure-preservingE(2)[OF assms(1) ‹U ∈ sets N›] by auto
  then show ?thesis unfolding U-def using AE-iff-null by blast
qed
```

The push-forward under a quasi-measure preserving map f of a measure absolutely continuous with respect to M is absolutely continuous with respect to N.

```
lemma quasi-measure-preserving-absolutely-continuous:
  assumes f ∈ quasi-measure-preserving M N
          u ∈ borel-measurable M
  shows absolutely-continuous N (distr (density M u) N f)
proof –
   have [measurable]: f ∈ measurable M N using quasi-measure-preservingE[OF
assms(1)] by auto
  have S ∈ null-sets (distr (density M u) N f) if [measurable]: S ∈ null-sets N for
S
  proof –
    have [measurable]: S ∈ sets N using null-setsD2[OF that] by auto
    have ∗: AE x in N. x ∉ S
      by (metis AE-not-in that)
    have AE x in M. f x ∉ S
      by (rule quasi-measure-preserving-AE[OF - ∗], simp add: assms)
    then have ∗: AE x in M. indicator S (f x) ∗ u x = 0
      by force

     have emeasure (distr (density M u) N f) S = (∫+x. indicator S x ∂(distr
(density M u) N f))
      by auto
    also have ... = (∫+x. indicator S (f x) ∂(density M u))
      by (rule nn-integral-distr, auto)
    also have ... = (∫+x. indicator S (f x) ∗ u x ∂M)
      by (rule nn-integral-densityR[symmetric], auto simp add: assms)
    also have ... = (∫+x. 0 ∂M)
      using ∗ by (rule nn-integral-cong-AE)
    finally have emeasure (distr (density M u) N f) S = 0 by auto
    then show ?thesis by auto
  qed
  then show ?thesis unfolding absolutely-continuous-def by auto
qed

definition measure-preserving::'a measure ⇒ 'b measure ⇒ ('a ⇒ 'b) set
  where measure-preserving M N
          = {f ∈ measurable M N. (∀ A ∈ sets N. emeasure M (f−‘A ∩ space M)
= emeasure N A)}

lemma measure-preservingE:
  assumes f ∈ measure-preserving M N
  shows f ∈ measurable M N
        ⋀A. A ∈ sets N ⟹ emeasure M (f−‘A ∩ space M) = emeasure N A
using assms unfolding measure-preserving-def by auto

lemma measure-preservingI [intro]:
  assumes f ∈ measurable M N
          ⋀A. A ∈ sets N ⟹ emeasure M (f−‘A ∩ space M) = emeasure N A
  shows f ∈ measure-preserving M N
using assms unfolding measure-preserving-def by auto
```

lemma *measure-preserving-distr*:
assumes $f \in$ *measure-preserving M N*
shows *distr M N f* = *N*
proof −
let *?N2* = *distr M N f*
have *sets ?N2* = *sets N* **by** *simp*
moreover have *emeasure ?N2 A* = *emeasure N A* **if** $A \in$ *sets N* **for** *A*
proof −
have *emeasure ?N2 A* = *emeasure M* $(f -` A \cap$ *space M*)
using ‹$A \in$ *sets N*› *assms emeasure-distr measure-preservingE*(*1*)[*OF assms*] **by** *blast*
then show *emeasure ?N2 A* = *emeasure N A*
using ‹$A \in$ *sets N*› *measure-preservingE*(*2*)[*OF assms*] **by** *auto*
qed
ultimately show *?thesis* **by** (*metis measure-eqI*)
qed

lemma *measure-preserving-distr'*:
assumes $f \in$ *measurable M N*
shows $f \in$ *measure-preserving M* (*distr M N f*)
proof (*rule measure-preservingI*)
show $f \in$ *measurable M* (*distr M N f*) **using** *assms*(*1*) **by** *auto*
show *emeasure M* $(f -` A \cap$ *space M*) = *emeasure* (*distr M N f*) *A* **if** $A \in$ *sets* (*distr M N f*) **for** *A*
using *that emeasure-distr*[*OF assms*] **by** *auto*
qed

lemma *measure-preserving-preserves-nn-integral*:
assumes $T \in$ *measure-preserving M N*
$f \in$ *borel-measurable N*
shows $(\int^+ x.\ f\ x\ \partial N) = (\int^+ x.\ f\ (T\ x)\ \partial M)$
proof −
have $(\int^+ x.\ f\ (T\ x)\ \partial M) = (\int^+ y.\ f\ y\ \partial distr\ M\ N\ T)$
using *assms nn-integral-distr*[*of T M N f*, *OF measure-preservingE*(*1*)[*OF assms*(*1*)]] **by** *simp*
also have ... $= (\int^+ y.\ f\ y\ \partial N)$
using *measure-preserving-distr*[*OF assms*(*1*)] **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *measure-preserving-preserves-integral*:
fixes f :: $'a \Rightarrow\ 'b$::{*banach*, *second-countable-topology*}
assumes $T \in$ *measure-preserving M N*
and [*measurable*]: *integrable N f*
shows *integrable M* $(\lambda x.\ f(T\ x))$ $(\int x.\ f\ x\ \partial N) = (\int x.\ f\ (T\ x)\ \partial M)$
proof −
have *a* [*measurable*]: $T \in$ *measurable M N* **by** (*rule measure-preservingE*(*1*)[*OF assms*(*1*)])

have *b* [*measurable*]: *f* ∈ *borel-measurable N* **by** *simp*
have *distr M N T* = *N* **using** *measure-preserving-distr*[*OF assms*(*1*)] **by** *simp*
then have *integrable* (*distr M N T*) *f* **using** *assms*(*2*) **by** *simp*
then show *integrable M* (λ*x*. *f*(*T x*)) **using** *integrable-distr-eq*[*OF a b*] **by** *simp*

have (∫ *x*. *f* (*T x*) ∂*M*) = (∫ *y*. *f y* ∂*distr M N T*) **using** *integral-distr*[*OF a b*] **by** *simp*
then show (∫ *x*. *f x* ∂*N*) = (∫ *x*. *f* (*T x*) ∂*M*) **using** ‹*distr M N T* = *N*› **by** *simp*
qed

lemma *measure-preserving-preserves-integral*′:
fixes *f* :: ′*a* ⇒ ′*b*::{*banach*, *second-countable-topology*}
assumes *T* ∈ *measure-preserving M N*
and [*measurable*]: *integrable M* (λ*x*. *f* (*T x*)) *f* ∈ *borel-measurable N*
shows *integrable N f* (∫ *x*. *f x* ∂*N*) = (∫ *x*. *f* (*T x*) ∂*M*)
proof −
have *a* [*measurable*]: *T* ∈ *measurable M N* **by** (*rule measure-preservingE*(*1*)[*OF assms*(*1*)])
have *integrable M* (λ*x*. *f*(*T x*)) **using** *assms*(*2*) **unfolding** *comp-def* **by** *auto*
then have *integrable* (*distr M N T*) *f*
using *integrable-distr-eq*[*OF a assms*(*3*)] **by** *simp*
then show ∗: *integrable N f* **using** *measure-preserving-distr*[*OF assms*(*1*)] **by** *simp*

then show (∫ *x*. *f x* ∂*N*) = (∫ *x*. *f* (*T x*) ∂*M*)
using *measure-preserving-preserves-integral*[*OF assms*(*1*) ∗] **by** *simp*
qed

lemma *id-measure-preserving*:
(λ*x*. *x*) ∈ *measure-preserving M M*
unfolding *measure-preserving-def* **by** *auto*

lemma *measure-preserving-is-quasi-measure-preserving*:
assumes *f* ∈ *measure-preserving M N*
shows *f* ∈ *quasi-measure-preserving M N*
using *assms* **unfolding** *measure-preserving-def quasi-measure-preserving-def* **apply** *auto*
by (*metis null-setsD1 null-setsI*, *metis measurable-sets null-setsD1 null-setsI*)

lemma *measure-preserving-composition*:
assumes *f* ∈ *measure-preserving M N*
g ∈ *measure-preserving N P*
shows (λ*x*. *g*(*f x*)) ∈ *measure-preserving M P*
proof (*rule measure-preservingI*)
have *f* [*measurable*]: *f* ∈ *measurable M N* **by** (*rule measure-preservingE*(*1*)[*OF assms*(*1*)])
have *g* [*measurable*]: *g* ∈ *measurable N P* **by** (*rule measure-preservingE*(*1*)[*OF assms*(*2*)])

show [*measurable*]: (*λx. g* (*f x*)) ∈ *measurable M P* **by** *auto*

fix *C* **assume** [*measurable*]: *C* ∈ *sets P*
define *B* **where** *B* = *g*−'*C* ∩ *space N*
have [*measurable*]: *B* ∈ *sets N* **unfolding** *B-def* **by** *simp*
have ∗: *emeasure N B* = *emeasure P C*
unfolding *B-def* **using** *measure-preservingE*(*2*)[*OF assms*(*2*)] **by** *simp*

define *A* **where** *A* = *f*−'*B* ∩ *space M*
have [*measurable*]: *A* ∈ *sets M* **unfolding** *A-def* **by** *simp*
have *emeasure M A* = *emeasure N B*
unfolding *A-def* **using** *measure-preservingE*(*2*)[*OF assms*(*1*)] **by** *simp*

then have *emeasure M A* = *emeasure P C* **using** ∗ **by** *simp*
moreover have *A* = (*λx. g*(*f x*))−'*C* ∩ *space M*
by (*auto simp add*: *A-def B-def*) (*meson f measurable-space*)
ultimately show *emeasure M* ((*λx. g*(*f x*))−'*C* ∩ *space M*) = *emeasure P C*
by *simp*
qed

lemma *measure-preserving-comp*:
assumes *f* ∈ *measure-preserving M N*
g ∈ *measure-preserving N P*
shows *g o f* ∈ *measure-preserving M P*
unfolding *o-def* **using** *measure-preserving-composition assms* **by** *blast*

lemma *measure-preserving-total-measure*:
assumes *f* ∈ *measure-preserving M N*
shows *emeasure M* (*space M*) = *emeasure N* (*space N*)
proof −
have *f* ∈ *measurable M N* **by** (*rule measure-preservingE*(*1*)[*OF assms*(*1*)])
then have *f*−'(*space N*) ∩ *space M* = *space M* **by** (*meson Int-absorb1 measurable-space subsetI vimageI*)
then show *emeasure M* (*space M*) = *emeasure N* (*space N*)
by (*metis* (*mono-tags*, *lifting*) *measure-preservingE*(*2*)[*OF assms*(*1*)] *sets.top*)
qed

lemma *measure-preserving-finite-measure*:
assumes *f* ∈ *measure-preserving M N*
shows *finite-measure M* ⟷ *finite-measure N*
using *measure-preserving-total-measure*[*OF assms*]
by (*metis finite-measure.emeasure-finite finite-measureI infinity-ennreal-def*)

lemma *measure-preserving-prob-space*:
assumes *f* ∈ *measure-preserving M N*
shows *prob-space M* ⟷ *prob-space N*
using *measure-preserving-total-measure*[*OF assms*] **by** (*metis prob-space.emeasure-space-1 prob-spaceI*)

locale *qmpt* = *sigma-finite-measure* +
  fixes *T*
  assumes *Tqm*: *T* ∈ *quasi-measure-preserving M M*

locale *mpt* = *qmpt* +
  assumes *Tm*: *T* ∈ *measure-preserving M M*

locale *fmpt* = *mpt* + *finite-measure*

locale *pmpt* = *fmpt* + *prob-space*

lemma *qmpt-I*:
  assumes *sigma-finite-measure M*
          T ∈ *measurable M M*
        ⋀*A*. *A* ∈ *sets M* ⟹ ((*T*−‘*A* ∩ *space M*) ∈ *null-sets M*) ⟷ (*A* ∈ *null-sets M*)
  shows *qmpt M T*
unfolding *qmpt-def qmpt-axioms-def quasi-measure-preserving-def*
by (*auto simp add*: *assms*)

lemma *mpt-I*:
  assumes *sigma-finite-measure M*
          T ∈ *measurable M M*
          ⋀*A*. *A* ∈ *sets M* ⟹ *emeasure M* (*T*−‘*A* ∩ *space M*) = *emeasure M A*
  shows *mpt M T*
proof −
  have ∗: *T* ∈ *measure-preserving M M*
    by (*rule measure-preservingI*[*OF assms*(*2*) *assms*(*3*)])
  then have ∗∗: *T* ∈ *quasi-measure-preserving M M*
    using *measure-preserving-is-quasi-measure-preserving* **by** *auto*
  show *mpt M T*
   unfolding *mpt-def qmpt-def qmpt-axioms-def mpt-axioms-def* **using** ∗ ∗∗ *assms*(*1*)
by *auto*
qed

lemma *fmpt-I*:
  assumes *finite-measure M*
          T ∈ *measurable M M*
          ⋀*A*. *A* ∈ *sets M* ⟹ *emeasure M* (*T*−‘*A* ∩ *space M*) = *emeasure M A*
  shows *fmpt M T*
proof −
  have ∗: *T* ∈ *measure-preserving M M*
    by (*rule measure-preservingI*[*OF assms*(*2*) *assms*(*3*)])
  then have ∗∗: *T* ∈ *quasi-measure-preserving M M*
    using *measure-preserving-is-quasi-measure-preserving* **by** *auto*
  show *fmpt M T*
    unfolding *fmpt-def mpt-def qmpt-def mpt-axioms-def qmpt-axioms-def*
    using ∗ ∗∗ *assms*(*1*) *finite-measure-def* **by** *auto*
qed

lemma *pmpt-I*:
assumes *prob-space M*
T ∈ *measurable M M*
$\bigwedge A.\ A \in sets\ M \Longrightarrow emeasure\ M\ (T-`A \cap space\ M) = emeasure\ M\ A$
shows *pmpt M T*
proof −
have ∗: *T* ∈ *measure-preserving M M*
by (*rule measure-preservingI*[*OF assms*(*2*) *assms*(*3*)])
then have ∗∗: *T* ∈ *quasi-measure-preserving M M*
using *measure-preserving-is-quasi-measure-preserving* **by** *auto*
show *pmpt M T*
unfolding *pmpt-def fmpt-def mpt-def qmpt-def mpt-axioms-def qmpt-axioms-def*
using ∗ ∗∗ *assms*(*1*) *prob-space-imp-sigma-finite prob-space.finite-measure* **by** *auto*
qed

4.2 Examples

lemma *fmpt-null-space*:
assumes *emeasure M* (*space M*) = *0*
T ∈ *measurable M M*
shows *fmpt M T*
apply (*rule fmpt-I*)
apply (*auto simp add*: *assms finite-measureI*)
apply (*metis assms emeasure-eq-0 measurable-sets sets.sets-into-space sets.top*)
done

lemma *fmpt-empty-space*:
assumes *space M* = {}
shows *fmpt M T*
by (*rule fmpt-null-space*, *auto simp add*: *assms measurable-empty-iff*)

Translations are measure-preserving

lemma *mpt-translation*:
fixes *c* :: *'a*::*euclidean-space*
shows *mpt lborel* (*λx. x* + *c*)
proof (*rule mpt-I*, *auto simp add*: *lborel.sigma-finite-measure-axioms*)
fix *A*::*'a set* **assume** [*measurable*]: *A* ∈ *sets borel*
have *emeasure lborel* ((*λx. x* + *c*) −` *A*) = *emeasure lborel* ((((+))*c*)−`*A*) **by** (*meson add.commute*)
also have ... = *emeasure lborel* ((((+))*c*)−`*A* ∩ *space lborel*) **by** *simp*
also have ... = *emeasure* (*distr lborel borel* ((+) *c*)) *A* **by** (*rule emeasure-distr*[*symmetric*], *auto*)
also have ... = *emeasure lborel A* **using** *lborel-distr-plus*[*of c*] **by** *simp*
finally show *emeasure lborel* ((*λx. x* + *c*) −` *A*) = *emeasure lborel A* **by** *simp*
qed

Skew products are fibered maps of the form $(x, y) \mapsto (Tx, U(x, y))$. If the base map and the fiber maps all are measure preserving, so is the skew product.

lemma *pair-measure-null-product*:
  assumes *emeasure M* (*space M*) = *0*
  shows *emeasure* ($M \bigotimes_M N$) (*space* ($M \bigotimes_M N$)) = *0*
proof −
  have ($\int^+ x.$ ($\int^+ y.$ *indicator X* (*x*,*y*) ∂N) ∂M) = *0* **for** *X*
  proof −
    have ($\int^+ x.$ ($\int^+ y.$ *indicator X* (*x*,*y*) ∂N) ∂M) = ($\int^+ x.$ *0* ∂M)
      by (*intro nn-integral-cong-AE emeasure-0-AE*[*OF assms*])
    then show *?thesis* **by** *auto*
  qed
  then have $M \bigotimes_M N$ = *measure-of* (*space M* × *space N*)
      {*a* × *b* | *a b*. *a* ∈ *sets M* ∧ *b* ∈ *sets N*}
      (λ*X*. *0*)
    unfolding *pair-measure-def* **by** *auto*
  then show *?thesis* **by** (*simp add*: *emeasure-sigma*)
qed

lemma *mpt-skew-product*:
  assumes *mpt M T*
        AE x in M. *mpt N* (*U x*)
    and [*measurable*]: (λ(*x*,*y*). *U x y*) ∈ *measurable* ($M \bigotimes_M N$) *N*
  shows *mpt* ($M \bigotimes_M N$) (λ(*x*,*y*). (*T x*, *U x y*))
proof (*cases*)
  assume *H*: *emeasure M* (*space M*) = *0*
  then have ∗: *emeasure* ($M \bigotimes_M N$) (*space* ($M \bigotimes_M N$)) = *0*
    using *pair-measure-null-product* **by** *auto*
  have [*measurable*]: *T* ∈ *measurable M M*
   using *assms*(*1*) **unfolding** *mpt-def qmpt-def qmpt-axioms-def quasi-measure-preserving-def*
by *auto*
  then have [*measurable*]: (λ(*x*, *y*). (*T x*, *U x y*)) ∈ *measurable* ($M \bigotimes_M N$) (M
$\bigotimes_M N$) **by** *auto*
  with *fmpt-null-space*[*OF* ∗] **show** *?thesis* **by** (*simp add*: *fmpt.axioms*(*1*))
next
  assume ¬(*emeasure M* (*space M*) = *0*)
  show *?thesis*
  proof (*rule mpt-I*)
   have *sigma-finite-measure M* **using** *assms*(*1*) **unfolding** *mpt-def qmpt-def* **by**
auto
    then interpret *M*: *sigma-finite-measure M* **.**

    have ∃ *p*. ¬ *almost-everywhere M p*
        by (*metis* (*lifting*) *AE-E* ‹*emeasure M* (*space M*) ≠ *0*› *emeasure-eq-AE*
emeasure-notin-sets)
   then have ∃ *x*. *mpt N* (*U x*) **using** *assms*(*2*) ‹¬(*emeasure M* (*space M*) = *0*)›
     by (*metis* (*full-types*) ‹*AE x in M*. *mpt N* (*U x*)› *eventually-mono*)
    then have *sigma-finite-measure N* **unfolding** *mpt-def qmpt-def* **by** *auto*
    then interpret *N*: *sigma-finite-measure N* **.**
    show *sigma-finite-measure* ($M \bigotimes_M N$)

by (*rule sigma-finite-pair-measure*) *standard+*

have [*measurable*]: $T \in$ *measurable M M*
using *assms*(*1*) **unfolding** *mpt-def qmpt-def qmpt-axioms-def quasi-measure-preserving-def* **by** *auto*
show [*measurable*]: $(\lambda(x, y). (T\ x, U\ x\ y)) \in$ *measurable* $(M \bigotimes_M N)$ $(M \bigotimes_M N)$ **by** *auto*
have $T \in$ *measure-preserving M M* **using** *assms*(*1*) **by** (*simp add*: *mpt.Tm*)

fix *A* **assume** [*measurable*]: $A \in$ *sets* $(M \bigotimes_M N)$
then have [*measurable*]: $(\lambda\ (x,y).\ (indicator\ A\ (x,y))::ennreal) \in$ *borel-measurable* $(M \bigotimes_M N)$ **by** *auto*
then have [*measurable*]: $(\lambda x. \int^+ y.\ indicator\ A\ (x, y)\ \partial N) \in$ *borel-measurable M*
by *simp*

define *B* **where** $B = (\lambda(x, y). (T\ x, U\ x\ y))$ −' $A \cap$ *space* $(M \bigotimes_M N)$
then have [*measurable*]: $B \in$ *sets* $(M \bigotimes_M N)$ **by** *auto*

have $(\int^+ y.\ indicator\ B\ (x,y)\ \partial N) = (\int^+ y.\ indicator\ A\ (T\ x, y)\ \partial N)$ **if** $x \in$ *space M mpt N* $(U\ x)$ **for** *x*
proof −
have $T\ x \in$ *space M* **by** (*meson* ‹$T \in$ *measurable M M*› ‹$x \in$ *space M*› *measurable-space*)
then have *1*: $(\lambda y.\ (indicator\ A\ (T\ x,\ y))::ennreal) \in$ *borel-measurable N* **using** ‹$A \in$ *sets* $(M \bigotimes_M N)$› **by** *auto*
have *2*: $\bigwedge y.\ ((indicator\ B\ (x,\ y))::ennreal) = indicator\ A\ (T\ x,\ U\ x\ y) * indicator\ (space\ M)\ x * indicator\ (space\ N)\ y$
unfolding *B-def* **by** (*simp add*: *indicator-def space-pair-measure*)
have *3*: $U\ x \in$ *measure-preserving N N* **using** *assms*(*2*) *that*(*2*) **by** (*simp add*: *mpt.Tm*)

have $(\int^+ y.\ indicator\ B\ (x,y)\ \partial N) = (\int^+ y.\ indicator\ A\ (T\ x,\ U\ x\ y)\ \partial N)$
using *2* **by** (*intro nn-integral-cong-simp*) (*auto simp add*: *indicator-def* ‹$x \in$ *space M*›)
also have ... $= (\int^+ y.\ indicator\ A\ (T\ x,\ y)\ \partial N)$
by (*rule measure-preserving-preserves-nn-integral*[*OF 3*, *symmetric*], *metis 1*)
finally show *?thesis* **by** *simp*
qed
then have $*$: *AE x in M*. $(\int^+ y.\ indicator\ B\ (x,y)\ \partial N) = (\int^+ y.\ indicator\ A\ (T\ x,\ y)\ \partial N)$
using *assms*(*2*) **by** *auto*

have *emeasure* $(M \bigotimes_M N)$ $B = (\int^+ x.\ (\int^+ y.\ indicator\ B\ (x,y)\ \partial N)\ \partial M)$
using ‹$B \in$ *sets* $(M \bigotimes_M N)$› ‹*sigma-finite-measure N*› *sigma-finite-measure.emeasure-pair-measure* **by** *fastforce*
also have ... $= (\int^+ x.\ (\int^+ y.\ indicator\ A\ (T\ x,\ y)\ \partial N)\ \partial M)$
by (*intro nn-integral-cong-AE* $*$)

also have ... = ($\int^+ x.$ ($\int^+ y.$ *indicator A* (*x*, *y*) ∂N) ∂M)
by (*rule measure-preserving-preserves-nn-integral*[*OF* ‹*T* ∈ *measure-preserving M M*›, *symmetric*]) *auto*
also have ... = *emeasure* ($M \bigotimes_M N$) *A*
by (*simp add*: ‹*sigma-finite-measure N*› *sigma-finite-measure.emeasure-pair-measure*)
finally show *emeasure* ($M \bigotimes_M N$) ((λ(*x*, *y*). (*T x*, *U x y*)) −‘ *A* ∩ *space* ($M \bigotimes_M N$)) = *emeasure* ($M \bigotimes_M N$) *A*
unfolding *B-def* **by** *simp*
qed
qed

lemma *mpt-skew-product-real*:
fixes *f*::′*a* ⇒ ′*b*::*euclidean-space*
assumes *mpt M T* **and** [*measurable*]: *f* ∈ *borel-measurable M*
shows *mpt* ($M \bigotimes_M$ *lborel*) (λ(*x*,*y*). (*T x*, *y* + *f x*))
by (*rule mpt-skew-product*, *auto simp add*: *mpt-translation assms*(*1*))

4.3 Preimages restricted to *spaceM*

context *qmpt* **begin**

One is all the time lead to take the preimages of sets, and restrict them to `space M` where the dynamics is living. We introduce a shortcut for this notion.

definition *vimage-restr* :: (′*a* ⇒ ′*a*) ⇒ ′*a set* ⇒ ′*a set* (**infixr** ‹−−‘› *90*)
where
f −−‘ *A* ≡ *f*−‘ (*A* ∩ *space M*) ∩ *space M*

lemma *vrestr-eq* [*simp*]:
a ∈ *f*−−‘ *A* ⟷ *a* ∈ *space M* ∧ *f a* ∈ *A* ∩ *space M*
unfolding *vimage-restr-def* **by** *auto*

lemma *vrestr-intersec* [*simp*]:
f−−‘ (*A* ∩ *B*) = (*f*−−‘*A*) ∩ (*f*−−‘ *B*)
using *vimage-restr-def* **by** *auto*

lemma *vrestr-union* [*simp*]:
f−−‘ (*A* ∪ *B*) = *f*−−‘*A* ∪ *f*−−‘*B*
using *vimage-restr-def* **by** *auto*

lemma *vrestr-difference* [*simp*]:
f−−‘(*A*−*B*) = *f*−−‘*A* − *f*−−‘*B*
using *vimage-restr-def* **by** *auto*

lemma *vrestr-inclusion*:
A ⊆ *B* ⟹ *f*−−‘*A* ⊆ *f*−−‘*B*
using *vimage-restr-def* **by** *auto*

lemma *vrestr-Union* [*simp*]:

f −−‘ ($\bigcup$ A) = ($\bigcup$ X∈A. f −−‘ X)
using *vimage-restr-def* **by** *auto*

lemma *vrestr-UN* [*simp*]:
f −−‘ ($\bigcup$ x∈A. B x) = ($\bigcup$ x∈A. f −−‘ B x)
using *vimage-restr-def* **by** *auto*

lemma *vrestr-Inter* [*simp*]:
assumes $A \neq \{\}$
shows f −−‘ ($\bigcap$ A) = ($\bigcap$ X∈A. f −−‘ X)
using *vimage-restr-def assms* **by** *auto*

lemma *vrestr-INT* [*simp*]:
assumes $A \neq \{\}$
shows f −−‘ ($\bigcap$ x∈A. B x) = ($\bigcap$ x∈A. f −−‘ B x)
using *vimage-restr-def assms* **by** *auto*

lemma *vrestr-empty* [*simp*]:
f−−‘{} = {}
using *vimage-restr-def* **by** *auto*

lemma *vrestr-sym-diff* [*simp*]:
f−−‘(A Δ B) = (f−−‘A) Δ (f−−‘B)
by *auto*

lemma *vrestr-image*:
assumes $x \in f$−−‘A
shows $x \in$ *space* M f $x \in$ *space* M f $x \in A$
using *assms* **unfolding** *vimage-restr-def* **by** *auto*

lemma *vrestr-intersec-in-space*:
assumes $A \in$ *sets* M $B \in$ *sets* M
shows $A \cap f$−−‘B = $A \cap f$−‘B
unfolding *vimage-restr-def* **using** *assms sets.sets-into-space* **by** *auto*

lemma *vrestr-compose*:
assumes $g \in$ *measurable* M M
shows (λ x. f(g x))−−‘ A = g−−‘ (f−−‘ A)
proof −
define B **where** $B = A \cap$ *space* M
have (λ x. f(g x))−−‘ A = (λ x. f(g x)) −‘ $B \cap$ *space* M
using *B-def vimage-restr-def* **by** *blast*
moreover have (λ x. f(g x)) −‘ $B \cap$ *space* M = g−‘ (f−‘ $B \cap$ *space* M) $\cap$ *space* M
using *measurable-space*[*OF* ‹$g \in$ *measurable* M M›] **by** *auto*
moreover have g−‘ (f−‘ $B \cap$ *space* M) $\cap$ *space* M = g−−‘ (f−−‘ A)
using *B-def vimage-restr-def* **by** *simp*
ultimately show *?thesis* **by** *auto*
qed

lemma *vrestr-comp*:
assumes $g \in$ *measurable M M*
shows $(f\ o\ g)--`\ A = g--`\ (f--`\ A)$
proof −
have $f\ o\ g = (\lambda\ x.\ f(g\ x))$ **by** *auto*
then have $(f\ o\ g)--`\ A = (\lambda\ x.\ f(g\ x))--`\ A$ **by** *auto*
moreover have $(\lambda\ x.\ f(g\ x))--`\ A = g--`\ (f--`\ A)$ **using** *vrestr-compose assms* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed

lemma *vrestr-of-set*:
assumes $g \in$ *measurable M M*
shows $A \in$ *sets M* $\Longrightarrow g--`A = g-`A \cap$ *space M*
by (*simp add*: *vimage-restr-def*)

lemma *vrestr-meas* [*measurable* (*raw*)]:
assumes $g \in$ *measurable M M*
$A \in$ *sets M*
shows $g--`A \in$ *sets M*
using *assms vimage-restr-def* **by** *auto*

lemma *vrestr-same-emeasure-f*:
assumes $f \in$ *measure-preserving M M*
$A \in$ *sets M*
shows *emeasure M* $(f--`A) =$ *emeasure M A*
by (*metis* (*mono-tags*, *lifting*) *assms measure-preserving-def mem-Collect-eq sets.Int-space-eq2 vimage-restr-def*)

lemma *vrestr-same-measure-f*:
assumes $f \in$ *measure-preserving M M*
$A \in$ *sets M*
shows *measure M* $(f--`A) =$ *measure M A*
proof −
have *measure M* $(f--`A) =$ *enn2real* (*emeasure M* $(f--`A)$) **by** (*simp add*: *Sigma-Algebra.measure-def*)
also have ... = *enn2real* (*emeasure M A*) **using** *vrestr-same-emeasure-f*[*OF assms*] **by** *simp*
also have ... = *measure M A* **by** (*simp add*: *Sigma-Algebra.measure-def*)
finally show *measure M* $(f--`\ A) =$ *measure M A* **by** *simp*
qed

4.4 Basic properties of qmpt

lemma *T-meas* [*measurable* (*raw*)]:
$T \in$ *measurable M M*
by (*rule quasi-measure-preservingE*(*1*)[*OF Tqm*])

lemma *Tn-quasi-measure-preserving*:
  $T^{\frown\frown}n \in$ *quasi-measure-preserving M M*
proof (*induction n*)
  case *0*
  show *?case* **using** *id-quasi-measure-preserving* **by** *simp*
 next
  case (*Suc n*)
  then show *?case* **using** *Tqm quasi-measure-preserving-comp* **by** (*metis funpow-Suc-right*)
qed

lemma *Tn-meas* [*measurable* (*raw*)]:
  $T^{\frown\frown}n \in$ *measurable M M*
by (*rule quasi-measure-preservingE*(*1*)[*OF Tn-quasi-measure-preserving*])

lemma *T-vrestr-meas* [*measurable*]:
 assumes $A \in$ *sets M*
 shows $T--`\ A \in$ *sets M*
    $(T^{\frown\frown}n)--`\ A \in$ *sets M*
by (*auto simp add*: *vrestr-meas assms*)

We state the next lemma both with T^0 and with *id* as sometimes the simplifier simplifies T^0 to *id* before applying the first instance of the lemma.

lemma *T-vrestr-0* [*simp*]:
 assumes $A \in$ *sets M*
 shows $(T^{\frown\frown}0)--`A = A$
    $id--`A = A$
using *sets.sets-into-space*[*OF assms*] **by** *auto*

lemma *T-vrestr-composed*:
 assumes $A \in$ *sets M*
 shows $(T^{\frown\frown}n)--`\ (T^{\frown\frown}m)--`\ A = (T^{\frown\frown}(n+m))--`\ A$
    $T--`\ (T^{\frown\frown}m)--`\ A = (T^{\frown\frown}(m+1))--`\ A$
    $(T^{\frown\frown}m)--`\ T--`\ A = (T^{\frown\frown}(m+1))--`\ A$
proof −
 show $(T^{\frown\frown}n)--`\ (T^{\frown\frown}m)--`\ A = (T^{\frown\frown}(n+m))--`\ A$
  by (*simp add*: *Tn-meas funpow-add add.commute vrestr-comp*)
 show $T--`\ (T^{\frown\frown}m)--`\ A = (T^{\frown\frown}(m+1))--`\ A$
  by (*metis Suc-eq-plus1 T-meas funpow-Suc-right vrestr-comp*)
 show $(T^{\frown\frown}m)--`\ T--`\ A = (T^{\frown\frown}(m+1))--`\ A$
  by (*simp add*: *Tn-meas vrestr-comp*)
qed

In the next two lemmas, we give measurability statements that show up all the time for the usual preimage.

lemma *T-intersec-meas* [*measurable*]:
 assumes [*measurable*]: $A \in$ *sets M* $B \in$ *sets M*
 shows $A \cap T-`B \in$ *sets M*
    $A \cap (T^{\frown\frown}n)-`B \in$ *sets M*

T−‘A ∩ B ∈ sets M
(T^^n)−‘A ∩ B ∈ sets M
A ∩ (T ∘ T ^^ n) −‘ B ∈ sets M
(T ∘ T ^^ n) −‘ A ∩ B ∈ sets M
by (*metis T-meas Tn-meas assms(1) assms(2) measurable-comp sets.Int inf-commute vrestr-intersec-in-space vrestr-meas*)+

lemma *T-diff-meas* [*measurable*]:
assumes [*measurable*]: *A ∈ sets M B ∈ sets M*
shows *A − T−‘B ∈ sets M*
A − (T^^n)−‘B ∈ sets M
proof −
have *A − T−‘B = A ∩ space M − (T−‘B ∩ space M)*
using *sets.sets-into-space*[*OF assms(1)*] **by** *auto*
then show *A − T−‘B ∈ sets M* **by** *auto*
have *A − (T^^n)−‘B = A ∩ space M − ((T^^n)−‘B ∩ space M)*
using *sets.sets-into-space*[*OF assms(1)*] **by** *auto*
then show *A − (T^^n)−‘B ∈ sets M* **by** *auto*
qed

lemma *T-spaceM-stable* [*simp*]:
assumes *x ∈ space M*
shows *T x ∈ space M*
(T^^n) x ∈ space M
proof −
show *T x ∈ space M* **by** (*meson measurable-space T-meas measurable-def assms*)
show *(T^^n) x ∈ space M* **by** (*meson measurable-space Tn-meas measurable-def assms*)
qed

lemma *T-quasi-preserves-null*:
assumes *A ∈ sets M*
shows *A ∈ null-sets M ⟷ T−−‘ A ∈ null-sets M*
A ∈ null-sets M ⟷ (T^^n)−−‘ A ∈ null-sets M
using *Tqm Tn-quasi-measure-preserving* **unfolding** *quasi-measure-preserving-def*
by (*auto simp add*: *assms vimage-restr-def*)

lemma *T-quasi-preserves*:
assumes *A ∈ sets M*
shows *emeasure M A = 0 ⟷ emeasure M (T−−‘ A) = 0*
emeasure M A = 0 ⟷ emeasure M ((T^^n)−−‘ A) = 0
using *T-quasi-preserves-null*[*OF assms*] *T-vrestr-meas assms* **by** *blast*+

lemma *T-quasi-preserves-null2*:
assumes *A ∈ null-sets M*
shows *T−−‘ A ∈ null-sets M*
(T^^n)−−‘ A ∈ null-sets M
using *T-quasi-preserves-null*[*OF null-setsD2*[*OF assms*]] *assms* **by** *auto*

lemma *T-composition-borel* [*measurable*]:
  assumes *f* ∈ *borel-measurable M*
  shows (λ*x*. *f*(*T x*)) ∈ *borel-measurable M* (λ*x*. *f*((*T*^^*k*) *x*)) ∈ *borel-measurable M*
using *T-meas Tn-meas assms measurable-compose* **by** *auto*

lemma *T-AE-iterates*:
  assumes *AE x in M*. *P x*
  shows *AE x in M*. ∀ *n*. *P* ((*T*^^*n*) *x*)
proof −
  have *AE x in M*. *P* ((*T*^^*n*) *x*) **for** *n*
    by (*rule quasi-measure-preserving-AE*[*OF Tn-quasi-measure-preserving*[*of n*] *assms*])
  then show *?thesis* **unfolding** *AE-all-countable* **by** *simp*
qed

lemma *qmpt-power*:
  qmpt M (*T*^^*n*)
by (*standard*, *simp add*: *Tn-quasi-measure-preserving*)

lemma *T-Tn-T-compose*:
  T ((*T*^^*n*) *x*) = (*T*^^(*Suc n*)) *x*
  (*T*^^*n*) (*T x*) = (*T*^^(*Suc n*)) *x*
by (*auto simp add*: *funpow-swap1*)

lemma (**in** *qmpt*) *qmpt-density*:
  assumes [*measurable*]: *h* ∈ *borel-measurable M*
    and *AE x in M*. *h x* ≠ *0 AE x in M*. *h x* ≠ ∞
  shows *qmpt* (*density M h*) *T*
proof −
  interpret *A*: *sigma-finite-measure density M h*
    apply (*subst sigma-finite-iff-density-finite*) **using** *assms* **by** *auto*
  show *?thesis*
    apply (*standard*) **apply** (*rule quasi-measure-preservingI*)
    unfolding *null-sets-density*[*OF* ‹*h* ∈ *borel-measurable M*› ‹*AE x in M*. *h x* ≠ *0*›] *sets-density space-density*
    using *quasi-measure-preservingE*(*2*)[*OF Tqm*] **by** *auto*
qed

end

4.5 Basic properties of mpt

context *mpt*
begin

lemma *Tn-measure-preserving*:
  T^^*n* ∈ *measure-preserving M M*
proof (*induction n*)

case (*Suc n*)
then show *?case* **using** *Tm measure-preserving-comp* **by** (*metis funpow-Suc-right*)
qed (*simp add*: *id-measure-preserving*)

lemma *T-integral-preserving*:
fixes f :: $'a \Rightarrow 'b$::{*banach*, *second-countable-topology*}
assumes *integrable M f*
shows *integrable M* $(\lambda x.\ f(T\ x))$ $(\int x.\ f(T\ x)\ \partial M) = (\int x.\ f\ x\ \partial M)$
using *measure-preserving-preserves-integral*[*OF Tm assms*] **by** *auto*

lemma *Tn-integral-preserving*:
fixes f :: $'a \Rightarrow 'b$::{*banach*, *second-countable-topology*}
assumes *integrable M f*
shows *integrable M* $(\lambda x.\ f((T\widehat{\ }\widehat{\ }n)\ x))$ $(\int x.\ f((T\widehat{\ }\widehat{\ }n)\ x)\ \partial M) = (\int x.\ f\ x\ \partial M)$
using *measure-preserving-preserves-integral*[*OF Tn-measure-preserving assms*] **by** *auto*

lemma *T-nn-integral-preserving*:
fixes f :: $'a \Rightarrow$ *ennreal*
assumes $f \in$ *borel-measurable M*
shows $(\int^+ x.\ f(T\ x)\ \partial M) = (\int^+ x.\ f\ x\ \partial M)$
using *measure-preserving-preserves-nn-integral*[*OF Tm assms*] **by** *auto*

lemma *Tn-nn-integral-preserving*:
fixes f :: $'a \Rightarrow$ *ennreal*
assumes $f \in$ *borel-measurable M*
shows $(\int^+ x.\ f((T\widehat{\ }\widehat{\ }n)\ x)\ \partial M) = (\int^+ x.\ f\ x\ \partial M)$
using *measure-preserving-preserves-nn-integral*[*OF Tn-measure-preserving assms*(*1*)]
by *auto*

lemma *mpt-power*:
mpt M $(T\widehat{\ }\widehat{\ }n)$
by (*standard*, *simp-all add*: *Tn-quasi-measure-preserving Tn-measure-preserving*)

lemma *T-vrestr-same-emeasure*:
assumes $A \in$ *sets M*
shows *emeasure M* $(T--`\ A)$ = *emeasure M A*
emeasure M $((T\ \widehat{\ }\widehat{\ }\ n)--`A)$ = *emeasure M A*
by (*auto simp add*: *vrestr-same-emeasure-f Tm Tn-measure-preserving assms*)

lemma *T-vrestr-same-measure*:
assumes $A \in$ *sets M*
shows *measure M* $(T--`\ A)$ = *measure M A*
measure M $((T\ \widehat{\ }\widehat{\ }\ n)--`A)$ = *measure M A*
by (*auto simp add*: *vrestr-same-measure-f Tm Tn-measure-preserving assms*)

lemma (**in** *fmpt*) *fmpt-power*:
fmpt M $(T\widehat{\ }\widehat{\ }n)$
by (*standard*, *simp-all add*: *Tn-quasi-measure-preserving Tn-measure-preserving*)

end

4.6 Birkhoff sums

Birkhoff sums, obtained by summing a function along the orbit of a map, are basic objects to be understood in ergodic theory.

```
context qmpt
begin

definition birkhoff-sum::('a ⇒ 'b::comm-monoid-add) ⇒ nat ⇒ 'a ⇒ 'b
  where birkhoff-sum f n x = (∑ i∈{..<n}. f((T^^i)x))

lemma birkhoff-sum-meas [measurable]:
  fixes f::'a ⇒ 'b::{second-countable-topology, topological-comm-monoid-add}
  assumes f ∈ borel-measurable M
  shows birkhoff-sum f n ∈ borel-measurable M
proof −
  define F where F = (λi x. f((T^^i)x))
  have ⋀i. F i ∈ borel-measurable M using assms F-def by auto
  then have (λx. (∑ i<n. F i x)) ∈ borel-measurable M by measurable
 then have (λx. birkhoff-sum f n x) ∈ borel-measurable M unfolding birkhoff-sum-def
F-def by auto
  then show ?thesis by simp
qed

lemma birkhoff-sum-1 [simp]:
  birkhoff-sum f 0 x = 0
  birkhoff-sum f 1 x = f x
  birkhoff-sum f (Suc 0) x = f x
unfolding birkhoff-sum-def by auto

lemma birkhoff-sum-cocycle:
  birkhoff-sum f (n+m) x = birkhoff-sum f n x + birkhoff-sum f m ((T^^n)x)
proof −
  have (∑ i<m. f ((T ^^ i) ((T ^^ n) x))) = (∑ i<m. f ((T ^^ (i+n)) x)) by
(simp add: funpow-add)
  also have ... = (∑ j∈{n..< m+n}. f ((T ^^j) x))
    using atLeast0LessThan sum.shift-bounds-nat-ivl[where ?g = λj. f((T^^j)x)
and ?k = n and ?m = 0 and ?n = m, symmetric]
          add.commute add.left-neutral by auto
  finally have *: birkhoff-sum f m ((T^^n)x) = (∑ j∈{n..< m+n}. f ((T ^^j)
x)) unfolding birkhoff-sum-def by auto
   have birkhoff-sum f (n+m) x = (∑ i<n. f((T^^i)x)) + (∑ i∈{n..<m+n}.
f((T^^i)x))
   unfolding birkhoff-sum-def by (metis add.commute add.right-neutral atLeast0LessThan
le-add2 sum.atLeastLessThan-concat)
  also have ... = birkhoff-sum f n x + (∑ i∈{n..<m+n}. f((T^^i)x)) unfolding
```

birkhoff-sum-def **by** *simp*
 finally show *?thesis* **using** $*$ **by** *simp*
qed

lemma *birkhoff-sum-mono*:
 fixes *f g*::- $\Rightarrow$ *real*
 assumes $\bigwedge x.\ f\ x \leq g\ x$
 shows *birkhoff-sum f n x* $\leq$ *birkhoff-sum g n x*
unfolding *birkhoff-sum-def* **by** (*simp add*: *assms sum-mono*)

lemma *birkhoff-sum-abs*:
 fixes *f*::- $\Rightarrow$ *'b*::*real-normed-vector*
 shows *norm*(*birkhoff-sum f n x*) $\leq$ *birkhoff-sum* ($\lambda x.$ *norm*(*f x*)) *n x*
unfolding *birkhoff-sum-def* **using** *norm-sum* **by** *auto*

lemma *birkhoff-sum-add*:
 birkhoff-sum ($\lambda x.\ f\ x + g\ x$) *n x* = *birkhoff-sum f n x* + *birkhoff-sum g n x*
unfolding *birkhoff-sum-def* **by** (*simp add*: *sum.distrib*)

lemma *birkhoff-sum-diff*:
 fixes *f g*::- $\Rightarrow$ *real*
 shows *birkhoff-sum* ($\lambda x.\ f\ x - g\ x$) *n x* = *birkhoff-sum f n x* $-$ *birkhoff-sum g n*
x
unfolding *birkhoff-sum-def* **by** (*simp add*: *sum-subtractf*)

lemma *birkhoff-sum-cmult*:
 fixes *f*::- $\Rightarrow$ *real*
 shows *birkhoff-sum* ($\lambda x.\ c * f\ x$) *n x* = $c *$ *birkhoff-sum f n x*
unfolding *birkhoff-sum-def* **by** (*simp add*: *sum-distrib-left*)

lemma *skew-product-real-iterates*:
 fixes *f*::*'a* $\Rightarrow$ *real*
 shows (($\lambda(x,y).\ (T\ x,\ y + f\ x)$)^^*n*) (*x*,*y*) = ((*T*^^*n*) *x*, *y* + *birkhoff-sum f n x*)
apply (*induction n*)
apply (*auto*)
apply (*metis* (*no-types*, *lifting*) *Suc-eq-plus1 birkhoff-sum-cocycle qmpt.birkhoff-sum-1*(*2*)
qmpt-axioms)
done

end

lemma (**in** *mpt*) *birkhoff-sum-integral*:
 fixes *f* :: *'a* $\Rightarrow$ *'b*::{*banach*, *second-countable-topology*}
 assumes [*measurable*]: *integrable M f*
 shows *integrable M* (*birkhoff-sum f n*) ($\int x.$ *birkhoff-sum f n x* ∂M) = $n *_R$ ($\int x.$
f x ∂M)
proof $-$
 have *a*: $\bigwedge k.$ *integrable M* ($\lambda x.\ f((T$^^*k*) *x*))
  using *Tn-integral-preserving*(*1*) *assms* **by** *blast*

then have *integrable M* ($\lambda x.\ \sum k{\in}\{..{<}n\}.\ f((T\widehat{\ }\widehat{\ }k)\ x)$) **by** *simp*
then have *integrable M* ($\lambda x.$ *birkhoff-sum f n x*) **unfolding** *birkhoff-sum-def* **by** *auto*
then show *integrable M* (*birkhoff-sum f n*) **by** *simp*

have *b*: $\bigwedge k.\ (\int x.\ f((T\widehat{\ }\widehat{\ }k)x)\ \partial M) = (\int x.\ f\ x\ \partial M)$
using *Tn-integral-preserving*(*2*) *assms* **by** *blast*
have ($\int x.$ *birkhoff-sum f n x* ∂M) = ($\int x.\ (\sum k{\in}\{..{<}n\}.\ f((T\widehat{\ }\widehat{\ }k)\ x))\ \partial M$)
unfolding *birkhoff-sum-def* **by** *blast*
also have ... = ($\sum k{\in}\{..{<}n\}.\ (\int x.\ f((T\widehat{\ }\widehat{\ }k)\ x)\ \partial M)$)
by (*rule Bochner-Integration.integral-sum*, *simp add*: *a*)
also have ... = ($\sum k{\in}\{..{<}n\}.\ (\int x.\ f\ x\ \partial M)$) **using** *b* **by** *simp*
also have ... = $n *_R (\int x.\ f\ x\ \partial M)$ **by** (*simp add*: *sum-constant-scaleR*)
finally show ($\int x.$ *birkhoff-sum f n x* ∂M) = $n *_R (\int x.\ f\ x\ \partial M)$ **by** *simp*
qed

lemma (**in** *mpt*) *birkhoff-sum-nn-integral*:
fixes *f* :: *'a* ⇒ *ennreal*
assumes [*measurable*]: *f* ∈ *borel-measurable M* **and** *pos*: $\bigwedge x.\ f\ x \geq 0$
shows ($\int^+ x.$ *birkhoff-sum f n x* ∂M) = $n * (\int^+ x.\ f\ x\ \partial M)$
proof −
have [*measurable*]: $\bigwedge k.\ (\lambda x.\ f((T\widehat{\ }\widehat{\ }k)x)) \in$ *borel-measurable M* **by** *simp*
have *posk*: $\bigwedge k\ x.\ f((T\widehat{\ }\widehat{\ }k)x) \geq 0$ **using** *pos* **by** *simp*
have *b*: $\bigwedge k.\ (\int^+ x.\ f((T\widehat{\ }\widehat{\ }k)x)\ \partial M) = (\int^+ x.\ f\ x\ \partial M)$
using *Tn-nn-integral-preserving assms* **by** *blast*
have ($\int^+ x.$ *birkhoff-sum f n x* ∂M) = ($\int^+ x.\ (\sum k{\in}\{..{<}n\}.\ f((T\widehat{\ }\widehat{\ }k)\ x))\ \partial M$)
unfolding *birkhoff-sum-def* **by** *blast*
also have ... = ($\sum k{\in}\{..{<}n\}.\ (\int^+ x.\ f((T\widehat{\ }\widehat{\ }k)\ x)\ \partial M)$)
by (*rule nn-integral-sum*, *auto simp add*: *posk*)
also have ... = ($\sum k{\in}\{..{<}n\}.\ (\int^+ x.\ f\ x\ \partial M)$) **using** *b* **by** *simp*
also have ... = $n * (\int^+ x.\ f\ x\ \partial M)$ **by** *simp*
finally show ($\int^+ x.$ *birkhoff-sum f n x* ∂M) = $n * (\int^+ x.\ f\ x\ \partial M)$ **by** *simp*
qed

4.7 Inverse map

context *qmpt* **begin**

definition
invertible-qmpt ≡ (*bij T* ∧ *inv T* ∈ *measurable M M*)

definition
Tinv ≡ *inv T*

lemma *T-Tinv-of-set*:
assumes *invertible-qmpt*
A ∈ *sets M*
shows *T*−‘(*Tinv*−‘*A* ∩ *space M*) ∩ *space M* = *A*
using *assms sets.sets-into-space* **unfolding** *Tinv-def invertible-qmpt-def*

apply (*auto simp add*: *bij-betw-def*)
using *T-spaceM-stable*(*1*) **by** *blast*

lemma *Tinv-quasi-measure-preserving*:
  assumes *invertible-qmpt*
  shows *Tinv* ∈ *quasi-measure-preserving M M*
proof (*rule quasi-measure-preservingI*, *auto*)
  fix *A* **assume** [*measurable*]: *A* ∈ *sets M Tinv* −' *A* ∩ *space M* ∈ *null-sets M*
  then have *T*−'(*Tinv* −' *A* ∩ *space M*) ∩ *space M* ∈ *null-sets M*
    by (*metis T-quasi-preserves-null2*(*1*) *null-sets.Int-space-eq2 vimage-restr-def*)
  then show *A* ∈ *null-sets M*
    using *T-Tinv-of-set*[*OF assms* ‹*A* ∈ *sets M*›] **by** *auto*
next
  show [*measurable*]: *Tinv* ∈ *measurable M M*
    using *assms* **unfolding** *Tinv-def invertible-qmpt-def* **by** *blast*
  fix *A* **assume** [*measurable*]: *A* ∈ *sets M A* ∈ *null-sets M*
  then have *T*−'(*Tinv* −' *A* ∩ *space M*) ∩ *space M* ∈ *null-sets M*
    using *T-Tinv-of-set*[*OF assms* ‹*A* ∈ *sets M*›] **by** *auto*
  moreover have [*measurable*]: *Tinv*−'*A* ∩ *space M* ∈ *sets M*
    by *auto*
  ultimately show *Tinv* −' *A* ∩ *space M* ∈ *null-sets M*
    using *T-meas T-quasi-preserves-null*(*1*) *vrestr-of-set* **by** *presburger*
qed

lemma *Tinv-qmpt*:
  assumes *invertible-qmpt*
  shows *qmpt M Tinv*
unfolding *qmpt-def qmpt-axioms-def* **using** *Tinv-quasi-measure-preserving*[*OF assms*]
by (*simp add*: *sigma-finite-measure-axioms*)

end

lemma (**in** *mpt*) *Tinv-measure-preserving*:
  assumes *invertible-qmpt*
  shows *Tinv* ∈ *measure-preserving M M*
proof (*rule measure-preservingI*)
  show [*measurable*]: *Tinv* ∈ *measurable M M*
    using *assms* **unfolding** *Tinv-def invertible-qmpt-def* **by** *blast*
  fix *A* **assume** [*measurable*]: *A* ∈ *sets M*
  have *A* = *T*−'(*Tinv* −' *A* ∩ *space M*) ∩ *space M*
    using *T-Tinv-of-set*[*OF assms* ‹*A* ∈ *sets M*›] **by** *auto*
  then show *emeasure M* (*Tinv* −' *A* ∩ *space M*) = *emeasure M A*
    by (*metis T-vrestr-same-emeasure*(*1*) ‹*A* ∈ *sets M*› ‹*Tinv* ∈ *M* $\rightarrow_M$ *M*› *measurable-sets sets.Int-space-eq2 vimage-restr-def*)
qed

lemma (**in** *mpt*) *Tinv-mpt*:
  assumes *invertible-qmpt*
  shows *mpt M Tinv*

unfolding *mpt-def mpt-axioms-def* **using** *Tinv-qmpt*[*OF assms*] *Tinv-measure-preserving*[*OF assms*] **by** *auto*

lemma (**in** *fmpt*) *Tinv-fmpt*:
  assumes *invertible-qmpt*
  shows *fmpt M Tinv*
unfolding *fmpt-def* **using** *Tinv-mpt*[*OF assms*] **by** (*simp add*: *finite-measure-axioms*)

lemma (**in** *pmpt*) *Tinv-fmpt*:
  assumes *invertible-qmpt*
  shows *pmpt M Tinv*
unfolding *pmpt-def* **using** *Tinv-fmpt*[*OF assms*] **by** (*simp add*: *prob-space-axioms*)

4.8 Factors

Factors of a system are quotients of this system, i.e., systems that can be obtained by a projection, forgetting some part of the dynamics. It is sometimes possible to transfer a result from a factor to the original system, making it possible to prove theorems by reduction to a simpler situation.

The dual notion, extension, is equally important and useful. We only mention factors below, as the results for extension readily follow by considering the original system as a factor of its extension.

In this paragraph, we define factors both in the qmpt and mpt categories, and prove their basic properties.

definition (**in** *qmpt*) *qmpt-factor*::(*'a* ⇒ *'b*) ⇒ (*'b measure*) ⇒ (*'b* ⇒ *'b*) ⇒ *bool*
  where *qmpt-factor proj M2 T2* =
    ((*proj* ∈ *quasi-measure-preserving M M2*) ∧ (*AE x in M*. *proj* (*T x*) = *T2* (*proj x*)) ∧ *qmpt M2 T2*)

lemma (**in** *qmpt*) *qmpt-factorE*:
  assumes *qmpt-factor proj M2 T2*
  shows *proj* ∈ *quasi-measure-preserving M M2*
        AE x in M. *proj* (*T x*) = *T2* (*proj x*)
        qmpt M2 T2
using *assms* **unfolding** *qmpt-factor-def* **by** *auto*

lemma (**in** *qmpt*) *qmpt-factor-iterates*:
  assumes *qmpt-factor proj M2 T2*
  shows *AE x in M*. ∀ *n*. *proj* ((*T*^^*n*) *x*) = (*T2*^^*n*) (*proj x*)
proof −
  have *AE x in M*. ∀ *n*. *proj* (*T* ((*T*^^*n*) *x*)) = *T2* (*proj* ((*T*^^*n*) *x*))
    by (*rule T-AE-iterates*[*OF qmpt-factorE*(*2*)[*OF assms*]])
  moreover
  {
    fix *x* **assume** ∀ *n*. *proj* (*T* ((*T*^^*n*) *x*)) = *T2* (*proj* ((*T*^^*n*) *x*))
    then have *H*: *proj* (*T* ((*T*^^*n*) *x*)) = *T2* (*proj* ((*T*^^*n*) *x*)) **for** *n* **by** *auto*
    have *proj* ((*T*^^*n*) *x*) = (*T2*^^*n*) (*proj x*) **for** *n*

```
    apply (induction n) using H by auto
  then have ∀ n. proj ((T^^n) x) = (T2^^n) (proj x) by auto
 }
 ultimately show ?thesis by fast
qed

lemma (in qmpt) qmpt-factorI:
 assumes proj ∈ quasi-measure-preserving M M2
         AE x in M. proj (T x) = T2 (proj x)
         qmpt M2 T2
 shows qmpt-factor proj M2 T2
using assms unfolding qmpt-factor-def by auto
```

When there is a quasi-measure-preserving projection, then the quotient map automatically is quasi-measure-preserving. The same goes for measure-preservation below.

```
lemma (in qmpt) qmpt-factorI':
 assumes proj ∈ quasi-measure-preserving M M2
         AE x in M. proj (T x) = T2 (proj x)
         sigma-finite-measure M2
         T2 ∈ measurable M2 M2
 shows qmpt-factor proj M2 T2
proof −
 have [measurable]: T ∈ measurable M M
                    T2 ∈ measurable M2 M2
                    proj ∈ measurable M M2
   using assms(4) quasi-measure-preservingE(1)[OF assms(1)] by auto

 have ∗: (T2 −‘ A ∩ space M2 ∈ null-sets M2) = (A ∈ null-sets M2) if A ∈ sets
M2 for A
 proof −
   obtain U where U: ⋀x. x ∈ space M − U ⟹ proj (T x) = T2 (proj x) U
∈ null-sets M
     using AE-E3[OF assms(2)] by blast

   then have [measurable]: U ∈ sets M by auto
   have [measurable]: A ∈ sets M2 using that by simp
   have e1: (T−‘(proj−‘A ∩ space M)) ∩ space M = T−‘(proj−‘A) ∩ space M
     using subset-eq by auto
   have e2: T−‘(proj−‘A) ∩ space M − U = proj−‘(T2−‘A) ∩ space M − U
     using U(1) by auto
   have e3: proj−‘(T2−‘A) ∩ space M = proj−‘(T2−‘A ∩ space M2) ∩ space M
     by (auto, meson ‹proj ∈ M →M M2› measurable-space)

   have A ∈ null-sets M2 ⟷ proj−‘A ∩ space M ∈ null-sets M
     using quasi-measure-preservingE(2)[OF assms(1)] by simp
   also have ... ⟷ (T−‘(proj−‘A ∩ space M)) ∩ space M ∈ null-sets M
     by (rule quasi-measure-preservingE(2)[OF Tqm, symmetric], auto)
   also have ... ⟷ T−‘(proj−‘A) ∩ space M ∈ null-sets M
```

using *e1* **by** *simp*
also have ... ⟷ *T*−‘(*proj*−‘*A*) ∩ *space M* − *U* ∈ *null-sets M*
using *emeasure-Diff-null-set*[*OF* ‹*U* ∈ *null-sets M*›] **unfolding** *null-sets-def* **by** *auto*
also have ... ⟷ *proj*−‘(*T2*−‘*A*) ∩ *space M* − *U* ∈ *null-sets M*
using *e2* **by** *simp*
also have ... ⟷ *proj*−‘(*T2*−‘*A*) ∩ *space M* ∈ *null-sets M*
using *emeasure-Diff-null-set*[*OF* ‹*U* ∈ *null-sets M*›] **unfolding** *null-sets-def* **by** *auto*
also have ... ⟷ *proj*−‘(*T2*−‘*A* ∩ *space M2*) ∩ *space M* ∈ *null-sets M*
using *e3* **by** *simp*
also have ... ⟷ *T2*−‘*A* ∩ *space M2* ∈ *null-sets M2*
using *quasi-measure-preservingE*(*2*)[*OF assms*(*1*), *of T2*−‘*A* ∩ *space M2*] **by** *simp*
finally show *T2*−‘*A* ∩ *space M2* ∈ *null-sets M2* ⟷ *A* ∈ *null-sets M2*
by *simp*
qed
show *?thesis*
by (*intro qmpt-factorI qmpt-I*) (*auto simp add*: *assms* ∗)
qed

lemma *qmpt-factor-compose*:
assumes *qmpt M1 T1*
qmpt.qmpt-factor M1 T1 proj1 M2 T2
qmpt.qmpt-factor M2 T2 proj2 M3 T3
shows *qmpt.qmpt-factor M1 T1* (*proj2 o proj1*) *M3 T3*
proof −
have ∗: *proj1* ∈ *quasi-measure-preserving M1 M2* ⟹ *AE x in M2*. *proj2* (*T2 x*) = *T3* (*proj2 x*)
⟹ (*AE x in M1*. *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*)))
proof −
assume *AE y in M2*. *proj2* (*T2 y*) = *T3* (*proj2 y*)
proj1 ∈ *quasi-measure-preserving M1 M2*
then have *AE x in M1*. *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
using *quasi-measure-preserving-AE* **by** *auto*
moreover
{
fix *x* **assume** *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
then have *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
by *auto*
}
ultimately show *AE x in M1*. *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
by *auto*
qed

interpret *I*: *qmpt M1 T1* **using** *assms*(*1*) **by** *simp*

interpret *J*: *qmpt M2 T2* **using** *I.qmpt-factorE*(*3*)[*OF assms*(*2*)] **by** *simp*
show *I.qmpt-factor* (*proj2 o proj1*) *M3 T3*
apply (*rule I.qmpt-factorI*)
using *I.qmpt-factorE*[*OF assms*(*2*)] *J.qmpt-factorE*[*OF assms*(*3*)]
by (*auto simp add*: *quasi-measure-preserving-comp* ∗)
qed

The left shift on natural integers is a very natural dynamical system, that can be used to model many systems as we see below. For invertible systems, one uses rather all the integers.

definition *nat-left-shift*::(*nat* ⇒ *'a*) ⇒ (*nat* ⇒ *'a*)
where *nat-left-shift x* = (λ*i*. *x* (*i*+*1*))

lemma *nat-left-shift-continuous* [*intro*, *continuous-intros*]:
continuous-on UNIV nat-left-shift
by (*rule continuous-on-coordinatewise-then-product*, *auto simp add*: *nat-left-shift-def*)

lemma *nat-left-shift-measurable* [*intro*, *measurable*]:
nat-left-shift ∈ *measurable borel borel*
by (*rule borel-measurable-continuous-onI*, *auto*)

definition *int-left-shift*::(*int* ⇒ *'a*) ⇒ (*int* ⇒ *'a*)
where *int-left-shift x* = (λ*i*. *x* (*i*+*1*))

definition *int-right-shift*::(*int* ⇒ *'a*) ⇒ (*int* ⇒ *'a*)
where *int-right-shift x* = (λ*i*. *x* (*i*−*1*))

lemma *int-shift-continuous* [*intro*, *continuous-intros*]:
continuous-on UNIV int-left-shift
continuous-on UNIV int-right-shift
apply (*rule continuous-on-coordinatewise-then-product*, *auto simp add*: *int-left-shift-def*)
apply (*rule continuous-on-coordinatewise-then-product*, *auto simp add*: *int-right-shift-def*)
done

lemma *int-shift-measurable* [*intro*, *measurable*]:
int-left-shift ∈ *measurable borel borel*
int-right-shift ∈ *measurable borel borel*
by (*rule borel-measurable-continuous-onI*, *auto*)+

lemma *int-shift-bij*:
bij int-left-shift inv int-left-shift = *int-right-shift*
bij int-right-shift inv int-right-shift = *int-left-shift*
proof −
show *bij int-left-shift*
apply (*rule bij-betw-byWitness*[**where** *?f'* = λ*x*. (λ*i*. *x* (*i*−*1*))]) **unfolding** *int-left-shift-def* **by** *auto*
show *inv int-left-shift* = *int-right-shift*
apply (*rule inv-equality*)
unfolding *int-left-shift-def int-right-shift-def* **by** *auto*

show *bij int-right-shift*
apply (*rule bij-betw-byWitness*[**where** *?f′* = λ*x*. (λ*i*. *x* (*i*+*1*))]) **unfolding** *int-right-shift-def* **by** *auto*
show *inv int-right-shift* = *int-left-shift*
apply (*rule inv-equality*)
unfolding *int-left-shift-def int-right-shift-def* **by** *auto*
qed

lemma (**in** *qmpt*) *qmpt-factor-projection*:
fixes *f*::*′a* ⇒ (*′b*::*second-countable-topology*)
assumes [*measurable*]: *f* ∈ *borel-measurable M*
and *sigma-finite-measure* (*distr M borel* (λ*x n*. *f* ((*T* ^^ *n*) *x*)))
shows *qmpt-factor* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) (*distr M borel* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*)))) *nat-left-shift*
proof (*rule qmpt-factorI′*)
have ∗ [*measurable*]: (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) ∈ *borel-measurable M*
using *measurable-coordinatewise-then-product* **by** *measurable*
show (λ*x n*. *f* ((*T* ^^ *n*) *x*)) ∈ *quasi-measure-preserving M* (*distr M borel* (λ*x n*. *f* ((*T* ^^ *n*) *x*)))
by (*rule measure-preserving-is-quasi-measure-preserving*[*OF measure-preserving-distr′*[*OF* ∗]])
have (λ*n*. *f* ((*T* ^^ *n*) (*T x*))) = *nat-left-shift* (λ*n*. *f* ((*T* ^^ *n*) *x*)) **for** *x*
unfolding *nat-left-shift-def* **by** (*auto simp add*: *funpow-swap1*)
then show *AE x in M*. (λ*n*. *f* ((*T* ^^ *n*) (*T x*))) = *nat-left-shift* (λ*n*. *f* ((*T* ^^ *n*) *x*))
by *simp*
qed (*auto simp add*: *assms*(*2*))

Let us now define factors of measure-preserving transformations, in the same way as above.

definition (**in** *mpt*) *mpt-factor*::(*′a* ⇒ *′b*) ⇒ (*′b measure*) ⇒ (*′b* ⇒ *′b*) ⇒ *bool*
where *mpt-factor proj M2 T2* =
((*proj* ∈ *measure-preserving M M2*) ∧ (*AE x in M*. *proj* (*T x*) = *T2* (*proj x*)) ∧ *mpt M2 T2*)

lemma (**in** *mpt*) *mpt-factor-is-qmpt-factor*:
assumes *mpt-factor proj M2 T2*
shows *qmpt-factor proj M2 T2*
using *assms* **unfolding** *mpt-factor-def qmpt-factor-def*
by (*simp add*: *measure-preserving-is-quasi-measure-preserving mpt-def*)

lemma (**in** *mpt*) *mpt-factorE*:
assumes *mpt-factor proj M2 T2*
shows *proj* ∈ *measure-preserving M M2*
AE x in M. *proj* (*T x*) = *T2* (*proj x*)
mpt M2 T2
using *assms* **unfolding** *mpt-factor-def* **by** *auto*

lemma (**in** *mpt*) *mpt-factorI*:

assumes *proj* ∈ *measure-preserving M M2*
AE x in M. proj (T x) = T2 (proj x)
mpt M2 T2
shows *mpt-factor proj M2 T2*
using *assms* **unfolding** *mpt-factor-def* **by** *auto*

When there is a measure-preserving projection commuting with the dynamics, and the dynamics above preserves the measure, then so does the dynamics below.

lemma (**in** *mpt*) *mpt-factorI′*:
assumes *proj* ∈ *measure-preserving M M2*
AE x in M. proj (T x) = T2 (proj x)
sigma-finite-measure M2
T2 ∈ *measurable M2 M2*
shows *mpt-factor proj M2 T2*
proof −
have [*measurable*]: *T* ∈ *measurable M M*
T2 ∈ *measurable M2 M2*
proj ∈ *measurable M M2*
using *assms(4) measure-preservingE(1)[OF assms(1)]* **by** *auto*

have ∗: *emeasure M2 (T2 −‘ A* ∩ *space M2) = emeasure M2 A* **if** *A* ∈ *sets M2*
for *A*
proof −
obtain *U* **where** *U*: ⋀*x. x* ∈ *space M − U* ⟹ *proj (T x) = T2 (proj x) U*
∈ *null-sets M*
using *AE-E3[OF assms(2)]* **by** *blast*

then have [*measurable*]: *U* ∈ *sets M* **by** *auto*
have [*measurable*]: *A* ∈ *sets M2* **using** *that* **by** *simp*
have *e1*: *(T−‘(proj−‘A* ∩ *space M))* ∩ *space M = T−‘(proj−‘A)* ∩ *space M*
using *subset-eq* **by** *auto*
have *e2*: *T−‘(proj−‘A)* ∩ *space M − U = proj−‘(T2−‘A)* ∩ *space M − U*
using *U(1)* **by** *auto*
have *e3*: *proj−‘(T2−‘A)* ∩ *space M = proj−‘(T2−‘A* ∩ *space M2)* ∩ *space M*
by (*auto, meson* ‹*proj* ∈ *M* $\rightarrow_M$ *M2*› *measurable-space*)

have *emeasure M2 A = emeasure M (proj−‘A* ∩ *space M)*
using *measure-preservingE(2)[OF assms(1)]* **by** *simp*
also have ... = *emeasure M (T−‘(proj−‘A* ∩ *space M)* ∩ *space M)*
by (*rule measure-preservingE(2)[OF Tm, symmetric], auto*)
also have ... = *emeasure M (T−‘(proj−‘A)* ∩ *space M)*
using *e1* **by** *simp*
also have ... = *emeasure M (T−‘(proj−‘A)* ∩ *space M − U)*
using *emeasure-Diff-null-set[OF* ‹*U* ∈ *null-sets M*›*]* **by** *auto*
also have ... = *emeasure M (proj−‘(T2−‘A)* ∩ *space M − U)*
using *e2* **by** *simp*
also have ... = *emeasure M (proj−‘(T2−‘A)* ∩ *space M)*
using *emeasure-Diff-null-set[OF* ‹*U* ∈ *null-sets M*›*]* **by** *auto*

also have ... = *emeasure M* (*proj*−'(*T2*−'*A* ∩ *space M2*) ∩ *space M*)
using *e3* **by** *simp*
also have ... = *emeasure M2* (*T2*−'*A* ∩ *space M2*)
using *measure-preservingE*(*2*)[*OF assms*(*1*), *of T2*−'*A* ∩ *space M2*] **by** *simp*
finally show *emeasure M2* (*T2*−'*A* ∩ *space M2*) = *emeasure M2 A*
by *simp*
qed
show *?thesis*
by (*intro mpt-factorI mpt-I*) (*auto simp add*: *assms* ∗)
qed

lemma (**in** *fmpt*) *mpt-factorI''*:
assumes *proj* ∈ *measure-preserving M M2*
AE x in M. *proj* (*T x*) = *T2* (*proj x*)
T2 ∈ *measurable M2 M2*
shows *mpt-factor proj M2 T2*
apply (*rule mpt-factorI'*, *auto simp add*: *assms*)
using *measure-preserving-finite-measure*[*OF assms*(*1*)] *finite-measure-axioms finite-measure-def*
by *blast*

lemma (**in** *fmpt*) *fmpt-factor*:
assumes *mpt-factor proj M2 T2*
shows *fmpt M2 T2*
unfolding *fmpt-def* **using** *mpt-factorE*(*3*)[*OF assms*]
measure-preserving-finite-measure[*OF mpt-factorE*(*1*)[*OF assms*]] *finite-measure-axioms*
by *auto*

lemma (**in** *pmpt*) *pmpt-factor*:
assumes *mpt-factor proj M2 T2*
shows *pmpt M2 T2*
unfolding *pmpt-def* **using** *fmpt-factor*[*OF assms*]
measure-preserving-prob-space[*OF mpt-factorE*(*1*)[*OF assms*]] *prob-space-axioms* **by**
auto

lemma *mpt-factor-compose*:
assumes *mpt M1 T1*
mpt.mpt-factor M1 T1 proj1 M2 T2
mpt.mpt-factor M2 T2 proj2 M3 T3
shows *mpt.mpt-factor M1 T1* (*proj2 o proj1*) *M3 T3*
proof −
have ∗: *proj1* ∈ *measure-preserving M1 M2* ⟹ *AE x in M2*. *proj2* (*T2 x*) =
T3 (*proj2 x*) ⟹
(*AE x in M1*. *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3*
(*proj2* (*proj1 x*)))
proof −
assume *AE y in M2*. *proj2* (*T2 y*) = *T3* (*proj2 y*)
proj1 ∈ *measure-preserving M1 M2*
then have *AE x in M1*. *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
using *quasi-measure-preserving-AE measure-preserving-is-quasi-measure-preserving*

by *blast*
moreover
{
fix *x* **assume** *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
then have *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
by *auto*
}
ultimately show *AE x in M1*. *proj1* (*T1 x*) = *T2* (*proj1 x*) ⟶ *proj2* (*T2* (*proj1 x*)) = *T3* (*proj2* (*proj1 x*))
by *auto*
qed

interpret *I*: *mpt M1 T1* **using** *assms*(*1*) **by** *simp*
interpret *J*: *mpt M2 T2* **using** *I.mpt-factorE*(*3*)[*OF assms*(*2*)] **by** *simp*
show *I.mpt-factor* (*proj2 o proj1*) *M3 T3*
apply (*rule I.mpt-factorI*)
using *I.mpt-factorE*[*OF assms*(*2*)] *J.mpt-factorE*[*OF assms*(*3*)]
by (*auto simp add*: *measure-preserving-comp* ∗)
qed

Left shifts are naturally factors of finite measure preserving transformations.

lemma (**in** *mpt*) *mpt-factor-projection*:
fixes *f*::*'a* ⇒ (*'b*::*second-countable-topology*)
assumes [*measurable*]: *f* ∈ *borel-measurable M*
and *sigma-finite-measure* (*distr M borel* (λ*x n*. *f* ((*T* ^^ *n*) *x*)))
shows *mpt-factor* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) (*distr M borel* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*)))) *nat-left-shift*
proof (*rule mpt-factorI'*)
have ∗ [*measurable*]: (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) ∈ *borel-measurable M*
using *measurable-coordinatewise-then-product* **by** *measurable*
show (λ*x n*. *f* ((*T* ^^ *n*) *x*)) ∈ *measure-preserving M* (*distr M borel* (λ*x n*. *f* ((*T* ^^ *n*) *x*)))
by (*rule measure-preserving-distr'*[*OF* ∗])
have (λ*n*. *f* ((*T* ^^ *n*) (*T x*))) = *nat-left-shift* (λ*n*. *f* ((*T* ^^ *n*) *x*)) **for** *x*
unfolding *nat-left-shift-def* **by** (*auto simp add*: *funpow-swap1*)
then show *AE x in M*. (λ*n*. *f* ((*T* ^^ *n*) (*T x*))) = *nat-left-shift* (λ*n*. *f* ((*T* ^^ *n*) *x*))
by *simp*
qed (*auto simp add*: *assms*(*2*))

lemma (**in** *fmpt*) *fmpt-factor-projection*:
fixes *f*::*'a* ⇒ (*'b*::*second-countable-topology*)
assumes [*measurable*]: *f* ∈ *borel-measurable M*
shows *mpt-factor* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) (*distr M borel* (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*)))) *nat-left-shift*
proof (*rule mpt-factor-projection*, *simp add*: *assms*)
have ∗ [*measurable*]: (λ*x*. (λ*n*. *f* ((*T*^^*n*)*x*))) ∈ *borel-measurable M*
using *measurable-coordinatewise-then-product* **by** *measurable*

have ∗∗: $(\lambda x\ n.\ f\ ((T\ \widehat{\ }\widehat{\ }\ n)\ x)) \in$ *measure-preserving M* (*distr M borel* $(\lambda x\ n.\ f$ $((T\ \widehat{\ }\widehat{\ }\ n)\ x)))$
by (*rule measure-preserving-distr'*[*OF* ∗])
have *a*: *finite-measure* (*distr M borel* $(\lambda x\ n.\ f\ ((T\ \widehat{\ }\widehat{\ }\ n)\ x)))$
using *measure-preserving-finite-measure*[*OF* ∗∗] *finite-measure-axioms* **by** *blast*
then show *sigma-finite-measure* (*distr M borel* $(\lambda x\ n.\ f\ ((T\ \widehat{\ }\widehat{\ }\ n)\ x)))$
by (*simp add*: *finite-measure-def*)
qed

4.9 Natural extension

Many probability preserving dynamical systems are not invertible, while invertibility is often useful in proofs. The notion of natural extension is a solution to this problem: it shows that (essentially) any system has an extension which is invertible.

This extension is constructed by considering the space of orbits indexed by integer numbers, with the left shift acting on it. If one considers the orbits starting from time $-N$ (for some fixed N), then there is a natural measure on this space: such an orbit is parameterized by its starting point at time $-N$, hence one may use the original measure on this point. The invariance of the measure ensures that these measures are compatible with each other. Their projective limit (when N tends to infinity) is thus an invariant measure on the bilateral shift. The shift with this measure is the desired extension of the original system.

There is a difficulty in the above argument: one needs to make sure that the projective limit of a system of compatible measures is well defined. This requires some topological conditions on the measures (they should be inner regular, i.e., the measure of any set should be approximated from below by compact subsets – this is automatic on polish spaces). The existence of projective limits is proved in `Projective_Limits.thy` under the (sufficient) polish condition. We use this theory, so we need the underlying space to be a polish space and the measure to be a Borel measure. This is almost completely satisfactory.

What is not completely satisfactory is that the completion of a Borel measure on a polish space (i.e., we add all subsets of sets of measure 0 into the sigma algebra) does not fit into this setting, while this is an important framework in dynamical systems. It would readily follow once `Projective_Limits.thy` is extended to the more general inner regularity setting (the completion of a Borel measure on a polish space is always inner regular).

locale *polish-pmpt* = *pmpt M*::(*'a*::*polish-space measure*) *T* **for** *M T*
+ **assumes** *M-eq-borel*: *sets M* = *sets borel*
begin

definition *natural-extension-map*
where *natural-extension-map* = (*int-left-shift*::((*int* ⇒ *'a*) ⇒ (*int* ⇒ *'a*)))

definition *natural-extension-measure*::(*int* ⇒ *'a*) *measure*
  where *natural-extension-measure* =
   projective-family.lim UNIV (λ*I*. *distr M* (Π_M *i*∈*I*. *borel*) (λ*x*. (λ*i*∈*I*. (*T*^^(*nat*(*i*− *Min I*))) *x*))) (λ*i*. *borel*)

definition *natural-extension-proj*::(*int* ⇒ *'a*) ⇒ *'a*
  where *natural-extension-proj* = (λ*x*. *x 0*)

theorem *natural-extension*:
  pmpt natural-extension-measure natural-extension-map
  qmpt.invertible-qmpt natural-extension-measure natural-extension-map
  mpt.mpt-factor natural-extension-measure natural-extension-map natural-extension-proj M T
proof −
  define *P*::*int set* ⇒ (*int* ⇒ *'a*) *measure* **where**
    P = (λ*I*. *distr M* (Π_M *i*∈*I*. *borel*) (λ*x*. (λ*i*∈*I*. (*T*^^(*nat*(*i*− *Min I*))) *x*)))
  have [*measurable*]: (*T*^^*n*) ∈ *measurable M borel* **for** *n*
    using *M-eq-borel* **by** *auto*

  interpret *polish-projective UNIV P*
  unfolding *polish-projective-def projective-family-def*
  proof (*auto*)
   show *prob-space* (*P I*) **if** *finite I* **for** *I* **unfolding** *P-def* **by** (*rule prob-space-distr*, *auto*)
    fix *J H*::*int set* **assume** *J* ⊆ *H finite H*
    then have *H* ∩ *J* = *J* **by** *blast*

    have ((λ*f*. *restrict f J*) *o* (λ*x*. (λ*i*∈*H*. (*T*^^(*nat*(*i*− *Min H*))) *x*))) *x*
       = ((λ*x*. (λ*i*∈*J*. (*T*^^(*nat*(*i*− *Min J*))) *x*)) *o* (*T*^^(*nat*(*Min J* − *Min H*)))) *x* **for** *x*
    proof −
     have *nat*(*i*− *Min H*) = *nat*(*i*− *Min J*) + *nat*(*Min J* − *Min H*) **if** *i* ∈ *J* **for** *i*
      proof −
        have *finite J* **using** ‹*J* ⊆ *H*› ‹*finite H*› *finite-subset* **by** *auto*
        then have *Min J* ∈ *J* **using** *Min-in* ‹*i* ∈ *J*› **by** *auto*
        then have *Min J* ∈ *H* **using** ‹*J* ⊆ *H*› **by** *blast*
        then have *Min H* ≤ *Min J* **using** *Min.coboundedI*[*OF* ‹*finite H*›] **by** *auto*
        moreover have *Min J* ≤ *i* **using** *Min.coboundedI*[*OF* ‹*finite J*› ‹*i* ∈ *J*›] **by** *auto*
        ultimately show *?thesis* **by** *auto*
      qed
      then show *?thesis*
        unfolding *comp-def* **by** (*auto simp add*: ‹*H* ∩ *J* = *J*› *funpow-add*)
    qed
    then have ∗: (λ*f*. *restrict f J*) *o* (λ*x*. (λ*i*∈*H*. (*T*^^(*nat*(*i*− *Min H*))) *x*))
       = (λ*x*. (λ*i*∈*J*. (*T*^^(*nat*(*i*− *Min J*))) *x*)) *o* (*T*^^(*nat*(*Min J* − *Min H*)))
      by *auto*

have *distr* (*P H*) (Pi_M *J* (λ-. *borel*)) (λ*f*. *restrict f J*)
= *distr M* (Π_M *i*∈*J*. *borel*) ((λ*f*. *restrict f J*) *o* (λ*x*. (λ*i*∈*H*. (*T*^^(*nat*(*i*− *Min H*))) *x*)))
unfolding *P-def* **by** (*rule distr-distr*, *auto simp add*: ‹*J* ⊆ *H*› *measurable-restrict-subset*)
also have ... = *distr M* (Π_M *i*∈*J*. *borel*) ((λ*x*. (λ*i*∈*J*. (*T*^^(*nat*(*i*− *Min J*))) *x*)) *o* (*T*^^(*nat*(*Min J* − *Min H*))))
using ∗ **by** *auto*
also have ... = *distr* (*distr M M* (*T*^^(*nat*(*Min J* − *Min H*)))) (Π_M *i*∈*J*. *borel*) (λ*x*. (λ*i*∈*J*. (*T*^^(*nat*(*i*− *Min J*))) *x*))
by (*rule distr-distr*[*symmetric*], *auto*)
also have ... = *distr M* (Π_M *i*∈*J*. *borel*) (λ*x*. (λ*i*∈*J*. (*T*^^(*nat*(*i*− *Min J*))) *x*))
using *measure-preserving-distr*[*OF Tn-measure-preserving*] **by** *auto*
also have ... = *P J*
unfolding *P-def* **by** *auto*
finally show *P J* = *distr* (*P H*) (Pi_M *J* (λ-. *borel*)) (λ*f*. *restrict f J*)
by *simp*
qed

have *S*: *sets* (Pi_M *UNIV* (λ-. *borel*)) = *sets* (*borel*::(*int* ⇒ ′*a*) *measure*)
by (*rule sets-PiM-equal-borel*)
have *natural-extension-measure* = *lim*
unfolding *natural-extension-measure-def P-def* **by** *simp*
have *measurable lim lim* = *measurable borel borel*
by (*rule measurable-cong-sets*, *auto simp add*: *S*)
then have [*measurable*]: *int-left-shift* ∈ *measurable lim lim int-right-shift* ∈ *measurable lim lim*
using *int-shift-measurable* **by** *fast*+
have [*simp*]: *space lim* = *UNIV*
unfolding *space-lim space-PiM space-borel* **by** *auto*

show *pmpt natural-extension-measure natural-extension-map*
proof (*rule pmpt-I*)
show *prob-space natural-extension-measure*
unfolding ‹*natural-extension-measure* = *lim*› **by** (*simp add*: *P*.*prob-space-axioms*)
show *natural-extension-map* ∈ *measurable natural-extension-measure natural-extension-measure*
unfolding *natural-extension-map-def* ‹*natural-extension-measure* = *lim*› **by** *simp*

define *E* **where** *E* = {(Π_E *i*∈*UNIV*. *X i*) |*X*::(*int* ⇒ ′*a set*). (∀ *i*. *X i* ∈ *sets borel*) ∧ *finite* {*i*. *X i* ≠ *UNIV*}}
have *lim* = *distr lim lim int-left-shift*
proof (*rule measure-eqI-generator-eq*[*of E UNIV*, **where** *?A* = λ-. *UNIV*])
show *sets lim* = *sigma-sets UNIV E*
unfolding *E-def* **using** *sets-PiM-finite*[*of UNIV*::*int set* λ-. (*borel*::′*a measure*)]
by (*simp add*: *PiE-def*)

moreover have *sets* (*distr lim lim int-left-shift*) = *sets lim* **by** *auto*
ultimately show *sets* (*distr lim lim int-left-shift*) = *sigma-sets UNIV E* **by** *simp*

show *emeasure lim UNIV* $\neq \infty$ **by** (*simp add*: *P.prob-space-axioms*)
have *UNIV* = (Π_E *i*∈(*UNIV*::*int set*). (*UNIV*::′*a set*)) **by** (*simp add*: *PiE-def*)
moreover have ... ∈ *E* **unfolding** *E-def* **by** *auto*
ultimately show *range* (λ(*i*::*nat*). (*UNIV*::(*int* ⇒ ′*a*) *set*)) ⊆ *E*
by *auto*

show *Int-stable E*
proof (*rule Int-stableI*)
fix *U V* **assume** *U* ∈ *E V* ∈ *E*
then obtain *X Y* **where** *H*: *U* = (Π_E *i*∈*UNIV*. *X i*) ⋀*i*. *X i* ∈ *sets borel finite* {*i*. *X i* ≠ *UNIV*}
V = (Π_E *i*∈*UNIV*. *Y i*) ⋀*i*. *Y i* ∈ *sets borel finite* {*i*. *Y i* ≠ *UNIV*}
unfolding *E-def* **by** *blast*
define *Z* **where** *Z* = (λ*i*. *X i* ∩ *Y i*)
have {*i*. *Z i* ≠ *UNIV*} ⊆ {*i*. *X i* ≠ *UNIV*} ∪ {*i*. *Y i* ≠ *UNIV*}
unfolding *Z-def* **by** *auto*
then have *finite* {*i*. *Z i* ≠ *UNIV*}
using *H*(*3*) *H*(*6*) *finite-subset* **by** *auto*
moreover have *U* ∩ *V* = (Π_E *i*∈*UNIV*. *Z i*)
unfolding *Z-def* **using** *H*(*1*) *H*(*4*) **by** *auto*
moreover have ⋀*i*. *Z i* ∈ *sets borel*
unfolding *Z-def* **using** *H*(*2*) *H*(*5*) **by** *auto*
ultimately show *U* ∩ *V* ∈ *E*
unfolding *E-def* **by** *auto*
qed

fix *U* **assume** *U* ∈ *E*
then obtain *X* **where** *H* [*measurable*]: *U* = (Π_E *i*∈*UNIV*. *X i*) ⋀*i*. *X i* ∈ *sets borel finite* {*i*. *X i* ≠ *UNIV*}
unfolding *E-def* **by** *blast*
define *I* **where** *I* = {*i*. *X i* ≠ *UNIV*}
have [*simp*]: *finite I* **unfolding** *I-def* **using** *H*(*3*) **by** *auto*
have [*measurable*]: (Π_E *i*∈*I*. *X i*) ∈ *sets* (Pi_M *I* (λ*i*. *borel*)) **using** *H*(*2*) **by** *simp*
have ∗: *U* = *emb UNIV I* (Π_E *i*∈*I*. *X i*)
unfolding *H*(*1*) *I-def prod-emb-def space-borel* **apply** (*auto simp add*: *PiE-def*)
by (*metis* (*mono-tags*, *lifting*) *PiE UNIV-I mem-Collect-eq restrict-Pi-cancel*)
have *emeasure lim U* = *emeasure lim* (*int-left-shift*−‘*U*)
proof (*cases I* = {})
case *True*
then have *U* = *UNIV* **unfolding** *H*(*1*) *I-def* **by** *auto*
then show *?thesis* **by** *auto*
next

case *False*
have *emeasure lim U* = *emeasure* (*P I*) (Π_E *i*∈*I*. *X i*)
unfolding ∗ **by** (*rule emeasure-lim-emb*, *auto*)
also have ... = *emeasure M* (((λ*x*. (λ*i*∈*I*. (*T*^^(*nat*(*i*− *Min I*))) *x*)))− ‘(Π_E *i*∈*I*. *X i*) ∩ *space M*)
unfolding *P-def* **by** (*rule emeasure-distr*, *auto*)
finally have *A*: *emeasure lim U* = *emeasure M* (((λ*x*. (λ*i*∈*I*. (*T*^^(*nat*(*i*− *Min I*))) *x*)))− ‘(Π_E *i*∈*I*. *X i*) ∩ *space M*)
by *simp*

have *i*: *int-left-shift*− ‘*U* = (Π_E *i*∈*UNIV*. *X* (*i*−*1*))
unfolding *H*(*1*) **apply** (*auto simp add*: *int-left-shift-def PiE-def*)
by (*metis PiE UNIV-I diff-add-cancel*, *metis Pi-mem add.commute add-diff-cancel-left′ iso-tuple-UNIV-I*)
define *Im* **where** *Im* = {*i*. *X* (*i*−*1*) ≠ *UNIV*}
have *Im* = (λ*i*. *i*+*1*) ‘*I*
unfolding *I-def Im-def* **using** *image-iff* **by** (*auto*, *fastforce*)
then have [*simp*]: *finite Im* **by** *auto*
have ∗: *int-left-shift*− ‘*U* = *emb UNIV Im* (Π_E *i*∈*Im*. *X* (*i*−*1*))
unfolding *i Im-def prod-emb-def space-borel* **apply** (*auto simp add*: *PiE-def*)
by (*metis* (*mono-tags*, *lifting*) *PiE UNIV-I mem-Collect-eq restrict-Pi-cancel*)
have *emeasure lim* (*int-left-shift*− ‘*U*) = *emeasure* (*P Im*) (Π_E *i*∈*Im*. *X* (*i*−*1*))
unfolding ∗ **by** (*rule emeasure-lim-emb*, *auto*)
also have ... = *emeasure M* (((λ*x*. (λ*i*∈*Im*. (*T*^^(*nat*(*i*− *Min Im*))) *x*)))− ‘(Π_E *i*∈*Im*. *X* (*i*−*1*)) ∩ *space M*)
unfolding *P-def* **by** (*rule emeasure-distr*, *auto*)
finally have *B*: *emeasure lim* (*int-left-shift*− ‘*U*) = *emeasure M* (((λ*x*. (λ*i*∈*Im*. (*T*^^(*nat*(*i*− *Min Im*))) *x*)))− ‘(Π_E *i*∈*Im*. *X* (*i*−*1*)) ∩ *space M*)
by *simp*

have *Min Im* = *Min I* + *1* **unfolding** ‹*Im* = (λ*i*. *i*+*1*) ‘*I*›
by (*rule mono-Min-commute*[*symmetric*], *auto simp add*: *False monoI*)
have ((λ*x*. (λ*i*∈*Im*. (*T*^^(*nat*(*i*− *Min Im*))) *x*)))− ‘(Π_E *i*∈*Im*. *X* (*i*−*1*)) = ((λ*x*. (λ*i*∈*I*. (*T*^^(*nat*(*i*− *Min I*))) *x*)))− ‘(Π_E *i*∈*I*. *X i*)
unfolding ‹*Min Im* = *Min I* + *1*› **unfolding** ‹*Im* = (λ*i*. *i*+*1*) ‘*I*› **by** (*auto simp add*: *Pi-iff*)
then show *emeasure lim U* = *emeasure lim* (*int-left-shift* −‘ *U*) **using** *A B* **by** *auto*
qed
also have ... = *emeasure lim* (*int-left-shift*− ‘*U* ∩ *space lim*)
unfolding ‹*space lim* = *UNIV*› **by** *auto*
also have ... = *emeasure* (*distr lim lim int-left-shift*) *U*
apply (*rule emeasure-distr*[*symmetric*], *auto*) **using** ∗ **by** *auto*
finally show *emeasure lim U* = *emeasure* (*distr lim lim int-left-shift*) *U*
by *simp*
qed (*auto*)

fix *U* **assume** *U* ∈ *sets natural-extension-measure*
then have [*measurable*]: *U* ∈ *sets lim* **using** ‹*natural-extension-measure* = *lim*› **by** *simp*
have *emeasure natural-extension-measure* (*natural-extension-map* −' *U* ∩ *space natural-extension-measure*)
= *emeasure lim* (*int-left-shift*−'*U* ∩ *space lim*)
unfolding ‹*natural-extension-measure* = *lim*› *natural-extension-map-def* **by** *simp*
also have ... = *emeasure* (*distr lim lim int-left-shift*) *U*
apply (*rule emeasure-distr*[*symmetric*], *auto*) **using** ‹*U* ∈ *P.events*› **by** *auto*
also have ... = *emeasure lim U*
using ‹*lim* = *distr lim lim int-left-shift*› **by** *simp*
also have ... = *emeasure natural-extension-measure U*
using ‹*natural-extension-measure* = *lim*› **by** *simp*
finally show *emeasure natural-extension-measure* (*natural-extension-map* −' *U* ∩ *space natural-extension-measure*)
= *emeasure natural-extension-measure U*
by *simp*
qed
then interpret *I*: *pmpt natural-extension-measure natural-extension-map* **by** *simp*

show *I.invertible-qmpt*
unfolding *I.invertible-qmpt-def* **unfolding** *natural-extension-map-def* ‹*natural-extension-measure* = *lim*›
by (*auto simp add*: *int-shift-bij*)

show *I.mpt-factor natural-extension-proj M T* **unfolding** *I.mpt-factor-def*
proof (*auto*)
show *mpt M T* **by** (*simp add*: *mpt-axioms*)
show *natural-extension-proj* ∈ *measure-preserving natural-extension-measure M*
unfolding ‹*natural-extension-measure* = *lim*›
proof
have ∗: *measurable lim M* = *measurable borel borel*
apply (*rule measurable-cong-sets*) **using** *sets-PiM-equal-borel M-eq-borel* **by** *auto*
show *natural-extension-proj* ∈ *measurable lim M*
unfolding ∗ *natural-extension-proj-def* **by** *auto*

fix *U* **assume** [*measurable*]: *U* ∈ *sets M*
have ∗: (((*λx*. *λi*∈{*0*}. (*T* ^^ *nat* (*i* − *Min* {*0*})) *x*))−' ({*0*} $\rightarrow_E$ *U*) ∩ *space M*) = *U*
using *sets.sets-into-space*[*OF* ‹*U* ∈ *sets M*›] **by** *auto*

have *natural-extension-proj*−'*U* ∩ *space lim* = *emb UNIV* {*0*} (Π_E *i*∈{*0*}. *U*)
unfolding ‹*space lim* = *UNIV*› *natural-extension-proj-def prod-emb-def* **by** (*auto simp add*: *PiE-iff*)

then have *emeasure lim* (*natural-extension-proj* $-$'*U* ∩ *space lim*) = *emeasure lim* (*emb UNIV* {*0*} (Π_E *i*∈{*0*}. *U*))
by *simp*
also have ... = *emeasure* (*P* {*0*}) (Π_E *i*∈{*0*}. *U*)
apply (*rule emeasure-lim-emb*, *auto*) **using** ‹*U* ∈ *sets M*› *M-eq-borel* **by** *auto*
also have ... = *emeasure M* (((λ*x*. λ*i*∈{*0*}. (*T* ^^ *nat* (*i* − *Min* {*0*})) *x*))$-$'
({*0*} $\rightarrow_E$ *U*) ∩ *space M*)
unfolding *P-def* **apply** (*rule emeasure-distr*) **using** ‹*U* ∈ *sets M*› *M-eq-borel* **by** *auto*
also have ... = *emeasure M U*
using ∗ **by** *simp*
finally show *emeasure lim* (*natural-extension-proj* $-$'*U* ∩ *space lim*) = *emeasure M U* **by** *simp*
qed

define *U*::(*int* ⇒ '*a*) *set* **where** *U* = {*x* ∈ *space* (Pi_M {*0*, *1*} (λ*i*. *borel*)). *x 1* = *T* (*x 0*)}
have ∗: ((λ*x*. λ*i*∈{*0*, *1*}. (*T* ^^ *nat* (*i* − *Min* {*0*, *1*})) *x*))$-$' *U* ∩ *space M* = *space M*
unfolding *U-def space-PiM space-borel* **by** *auto*
have [*measurable*]: *T* ∈ *measurable borel borel*
using *M-eq-borel* **by** *auto*
have [*measurable*]: *U* ∈ *sets* (Pi_M {*0*, *1*} (λ*i*. *borel*))
unfolding *U-def* **by** (*rule measurable-equality-set*, *auto*)
have *emeasure natural-extension-measure* (*emb UNIV* {*0*, *1*} *U*) = *emeasure* (*P* {*0*, *1*}) *U*
unfolding ‹*natural-extension-measure* = *lim*› **by** (*rule emeasure-lim-emb*, *auto*)
also have ... = *emeasure M* (((λ*x*. λ*i*∈{*0*, *1*}. (*T* ^^ *nat* (*i* − *Min* {*0*, *1*})) *x*))$-$' *U* ∩ *space M*)
unfolding *P-def* **by** (*rule emeasure-distr*, *auto*)
also have ... = *emeasure M* (*space M*)
using ∗ **by** *simp*
also have ... = *1* **by** (*simp add*: *emeasure-space-1*)
finally have ∗: *emeasure natural-extension-measure* (*emb UNIV* {*0*, *1*} *U*) = *1* **by** *simp*
have *AE x in natural-extension-measure*. *x* ∈ *emb UNIV* {*0*, *1*} *U*
apply (*rule I.AE-prob-1*) **using** ∗ **by** (*simp add*: *I.emeasure-eq-measure*)
moreover
{
fix *x* **assume** *x* ∈ *emb UNIV* {*0*, *1*} *U*
then have *x 1* = *T* (*x 0*) **unfolding** *prod-emb-def U-def* **by** *auto*
then have *natural-extension-proj* (*natural-extension-map x*) = *T* (*natural-extension-proj x*)
unfolding *natural-extension-proj-def natural-extension-map-def int-left-shift-def* **by** *auto*
}
ultimately show *AE x in natural-extension-measure*.

natural-extension-proj (*natural-extension-map x*) = *T* (*natural-extension-proj x*)
by *auto*
qed
qed

end

end

5 Conservativity, recurrence

theory *Recurrence*
imports *Measure-Preserving-Transformations*
begin

A dynamical system is conservative if almost every point comes back close to its starting point. This is always the case if the measure is finite, not when it is infinite (think of the translation on $\mathbb{Z}$). In conservative systems, an important construction is the induced map: the first return map to a set of finite measure. It is measure-preserving and conservative if the original system is. This makes it possible to reduce statements about general conservative systems in infinite measure to statements about systems in finite measure, and as such is extremely useful.

5.1 Definition of conservativity

locale *conservative* = *qmpt* +
assumes *conservative*: $\bigwedge A.\ A \in sets\ M \Longrightarrow emeasure\ M\ A > 0 \Longrightarrow \exists n>0.\ emeasure\ M\ ((T\widehat{\ }\widehat{\ }n)-`A \cap A) > 0$

lemma *conservativeI*:
assumes *qmpt M T*
$\bigwedge A.\ A \in sets\ M \Longrightarrow emeasure\ M\ A > 0 \Longrightarrow \exists n>0.\ emeasure\ M\ ((T\widehat{\ }\widehat{\ }n)-`A \cap A) > 0$
shows *conservative M T*
unfolding *conservative-def conservative-axioms-def* **using** *assms* **by** *auto*

To prove conservativity, it is in fact sufficient to show that the preimages of a set of positive measure intersect it, without any measure control. Indeed, in a non-conservative system, one can construct a set which does not satisfy this property.

lemma *conservativeI2*:
assumes *qmpt M T*
$\bigwedge A.\ A \in sets\ M \Longrightarrow emeasure\ M\ A > 0 \Longrightarrow \exists n>0.\ (T\widehat{\ }\widehat{\ }n)-`A \cap A \neq \{\}$
shows *conservative M T*
unfolding *conservative-def conservative-axioms-def*

proof (*auto simp add*: *assms*)
  interpret *qmpt M T* **using** *assms* **by** *auto*
  fix *A*
  assume *A-meas* [*measurable*]: $A \in$ *sets M* **and** *emeasure M A* > 0
  show $\exists n>0.$ $0 <$ *emeasure M* $((T$ ^^ $n) -` A \cap A)$
  proof (*rule ccontr*)
    assume $\neg$ ($\exists n>0.$ $0 <$ *emeasure M* $((T$ ^^ $n) -` A \cap A))$
    then have *meas-0*: *emeasure M* $((T$ ^^ $n) -` A \cap A) = 0$ **if** $n>0$ **for** n
      by (*metis zero-less-iff-neq-zero that*)
    define C **where** $C = (\bigcup n.\ (T$^^$(Suc\ n))-`A \cap A)$
    have *C-meas* [*measurable*]: $C \in$ *sets M* **unfolding** *C-def* **by** *measurable*
    have *emeasure M C* $= 0$ **unfolding** *C-def*
     by (*intro emeasure-UN-eq-0*[*of M*, *of* $\lambda n.\ (T$^^$(Suc\ n))-`A \cap A$, *OF meas-0*], *auto*)

    define *A2* **where** $A2 = A-C$
    then have *A2-meas* [*measurable*]: $A2 \in$ *sets M* **by** *simp*
    have $\neg(\exists n>0.\ (T$^^$n)-`A2 \cap A2 \neq \{\})$
    proof (*rule ccontr*, *simp*)
      assume $\exists n>0.\ (T$^^$n)-`A2 \cap A2 \neq \{\}$
      then obtain n **where** n: $n > 0$ $(T$^^$n)-`A2 \cap A2 \neq \{\}$ **by** *auto*
      define m **where** $m = n-1$
      have $(T$^^$(m+1))-`A2 \cap A2 \neq \{\}$ **unfolding** *m-def* **using** n **by** *auto*
      then show *False* **using** *C-def A2-def* **by** *auto*
    qed
     then have *emeasure M A2* $= 0$ **using** *assms*(*2*)[*OF A2-meas*] **by** (*meson zero-less-iff-neq-zero*)
    then have *emeasure M* $(C \cup A2) = 0$ **using** ‹*emeasure M C* $= 0$› **by** (*simp add*: *emeasure-Un-null-set null-setsI*)
    moreover have $A \subseteq C \cup A2$ **unfolding** *A2-def* **by** *auto*
   ultimately have *emeasure M A* $= 0$ **by** (*meson A2-meas C-meas emeasure-eq-0 sets.Un*)
    then show *False* **using** ‹*emeasure M A* > 0› **by** *auto*
  qed
qed

There is also a dual formulation, saying that conservativity follows from the fact that a set disjoint from all its preimages has to be null.

lemma *conservativeI3*:
  assumes *qmpt M T*
        $\bigwedge A.\ A \in$ *sets M* $\Longrightarrow (\forall n>0.\ (T$^^$n)-`A \cap A = \{\}) \Longrightarrow A \in$ *null-sets M*
  shows *conservative M T*
proof (*rule conservativeI2*[*OF assms*(*1*)])
  fix *A* **assume** $A \in$ *sets M* $0 <$ *emeasure M A*
  then have $\neg(A \in$ *null-sets M*$)$ **unfolding** *null-sets-def* **by** *auto*
  then show $\exists n>0.\ (T$ ^^ $n) -` A \cap A \neq \{\}$
    using *assms*(*2*)[*OF* ‹$A \in$ *sets M*›] **by** *auto*
qed

The inverse of a conservative map is still conservative

```
lemma (in conservative) conservative-Tinv:
  assumes invertible-qmpt
  shows conservative M Tinv
proof (rule conservativeI2)
  show qmpt M Tinv using Tinv-qmpt[OF assms].
  have bij T using assms unfolding invertible-qmpt-def by auto
  fix A assume [measurable]: A ∈ sets M and emeasure M A > 0
  then obtain n where ∗: n>0 emeasure M ((T^^n)−‘A ∩ A) > 0
    using conservative[OF ‹A ∈ sets M› ‹emeasure M A > 0›] by blast
  have bij (T^^n) using bij-fn[OF ‹bij T›] by auto
  then have bij(inv (T^^n)) using bij-imp-bij-inv by auto
   then have bij (Tinv^^n) unfolding Tinv-def using inv-fn[OF ‹bij T›, of n]
by auto

  have (T^^n)−‘A ∩ A ≠ {} using ∗ by auto
  then have (Tinv^^n)−‘((T^^n)−‘A ∩ A) ≠ {}
    using surj-vimage-empty[OF bij-is-surj[OF ‹bij (Tinv^^n)›]] by meson
  then have ∗∗: (Tinv^^n)−‘((T^^n)−‘A) ∩ (Tinv^^n)−‘ A ≠ {}
    by auto

  have (Tinv^^n)−‘((T^^n)−‘A) = ((T^^n) o (Tinv^^n))−‘A
    by auto
  moreover have (T^^n) o (Tinv^^n) = (λx. x)
    unfolding Tinv-def using ‹bij T› fn-o-inv-fn-is-id by blast
  ultimately have (Tinv^^n)−‘((T^^n)−‘A) = A by auto
  then have (Tinv^^n)−‘ A ∩ A ≠ {} using ∗∗ by auto
  then show ∃ n>0. (Tinv ^^ n) −‘ A ∩ A ≠ {} using ‹n>0› by auto
qed
```

We introduce the locale of a conservative measure preserving map.

```
locale conservative-mpt = mpt + conservative

lemma conservative-mptI:
  assumes mpt M T
    ⋀A. A ∈ sets M ⟹ emeasure M A > 0 ⟹ ∃ n>0. (T^^n)−‘A ∩ A ≠ {}
  shows conservative-mpt M T
unfolding conservative-mpt-def
apply (auto simp add: assms(1), rule conservativeI2)
using assms(1) by (auto simp add: mpt-def assms(2))
```

The fact that finite measure preserving transformations are conservative, albeit easy, is extremely important. This result is known as Poincaré recurrence theorem.

```
sublocale fmpt ⊆ conservative-mpt
proof (rule conservative-mptI)
  show mpt M T by (simp add: mpt-axioms)
  fix A assume A-meas [measurable]: A ∈ sets M and emeasure M A > 0
```

show ∃ *n*>*0*. (*T*^^*n*)−'*A* ∩ *A* ≠ {}
proof (*rule ccontr*)
assume ¬(∃ *n*>*0*. (*T*^^*n*)−'*A* ∩ *A* ≠ {})
then have *disj*: (*T*^^(*Suc n*))−−'*A* ∩ *A* = {} **for** *n* **unfolding** *vimage-restr-def* **using** *zero-less-one* **by** *blast*

define *B* **where** *B* = (λ *n*. (*T*^^*n*)−−'*A*)
then have *B-meas* [*measurable*]: *B n* ∈ *sets M* **for** *n* **by** *simp*
have *same*: *measure M* (*B n*) = *measure M A* **for** *n*
by (*simp add*: *B-def A-meas T-vrestr-same-measure*(*2*))

have *B n* ∩ *B m* = {} **if** *n* > *m* **for** *m n*
proof −
have *B n* ∩ *B m* = (*T*^^*m*)−−' (*B* (*n*−*m*) ∩ *A*)
using *B-def* ‹*m* < *n*› *A-meas vrestr-intersec T-vrestr-composed*(*1*) **by** *auto*
moreover have *B* (*n*−*m*) ∩ *A* = {} **unfolding** *B-def*
by (*metis disj* ‹*m* < *n*› *Suc-diff-Suc*)
ultimately show *?thesis* **by** *simp*
qed
then have *disjoint-family B* **by** (*metis disjoint-family-on-def inf-sup-aci*(*1*) *less-linear*)

have *measure M A* < *e* **if** *e*>*0* **for** *e*::*real*
proof −
obtain *N*::*nat* **where** *N*>*0* (*measure M* (*space M*))/*e*<*N* **using** ‹*0* < *e*›
by (*metis divide-less-0-iff reals-Archimedean2 less-eq-real-def measure-nonneg not-gr0 not-le of-nat-0*)
then have (*measure M* (*space M*))/*N* < *e* **using** ‹*0* < *e*› ‹*N*>*0*›
by (*metis bounded-measure div-0 le-less-trans measure-empty mult.commute pos-divide-less-eq*)
have ∗: *disjoint-family-on B* {..<*N*}
by (*meson UNIV-I* ‹*disjoint-family B*› *disjoint-family-on-mono subsetI*)
then have (∑ *i*∈{..<*N*}. *measure M* (*B i*)) ≤ *measure M* (*space M*)
by (*metis bounded-measure* ‹⋀*n*. *B n* ∈ *sets M*›
image-subset-iff finite-lessThan finite-measure-finite-Union)
also have (∑ *i*∈{..<*N*}. *measure M* (*B i*)) = (∑ *i*∈{..<*N*}. *measure M A*) **using** *same* **by** *simp*
also have ... = *N* ∗ (*measure M A*) **by** *simp*
finally have *N* ∗ (*measure M A*) ≤ *measure M* (*space M*) **by** *simp*
then have *measure M A* ≤ (*measure M* (*space M*))/*N* **using** ‹*N*>*0*› **by** (*simp add*: *mult.commute mult-imp-le-div-pos*)
then show *measure M A* < *e* **using** ‹(*measure M* (*space M*))/*N*<*e*› **by** *simp*
qed
then have *measure M A* ≤ *0* **using** *not-less* **by** *blast*
then have *measure M A* = *0* **by** (*simp add*: *measure-le-0-iff*)
then have *emeasure M A* = *0* **using** *emeasure-eq-measure* **by** *simp*
then show *False* **using** ‹*emeasure M A* > *0*› **by** *simp*
qed
qed

The following fact that powers of conservative maps are also conservative is true, but nontrivial. It is proved as follows: consider a set A with positive measure, take a time n_1 such that $A_1 = T^{-n_1}A \cap A$ has positive measure, then a time n_2 such that $A_2 = T^{-n_2}A_1 \cap A$ has positive measure, and so on. It follows that $T^{-(n_i+n_{i+1}+\cdots+n_j)}A \cap A$ has positive measure for all $i < j$. Then, one can find $i < j$ such that $n_i + \cdots + n_j$ is a multiple of N.

proposition (**in** *conservative*) *conservative-power*:
  conservative M (T^^n)
proof (*unfold-locales*)
  show T ^^ n ∈ *quasi-measure-preserving M M*
    by (*auto simp add*: *Tn-quasi-measure-preserving*)

  fix *A* **assume** [*measurable*]: *A* ∈ *sets M 0* < *emeasure M A*
  define *good-time* **where** *good-time* = (λ*K*. *Inf*{(*i*::*nat*). *i* > *0* ∧ *emeasure M* ((*T*^^*i*)−'*K* ∩ *A*) > *0*})
  define *next-good-set* **where** *next-good-set* = (λ*K*. (*T*^^(*good-time K*))−'*K* ∩ *A*)

  have *good-rec*: ((*good-time K* > *0*) ∧ (*next-good-set K* ⊆ *A*) ∧
        (*next-good-set K* ∈ *sets M*) ∧ (*emeasure M* (*next-good-set K*) > *0*))
      if [*measurable*]: *K* ∈ *sets M* **and** *K* ⊆ *A emeasure M K* > *0* **for** *K*
  proof −
    have *a*: *next-good-set K* ∈ *sets M next-good-set K* ⊆ *A*
      using *next-good-set-def* **by** *simp-all*
    obtain *k* **where** *k* > *0* **and** *posK*: *emeasure M* ((*T*^^*k*)−'*K* ∩ *K*) > *0*
      using *conservative*[*OF* ‹*K* ∈ *sets M*›, *OF* ‹*emeasure M K* > *0*›] **by** *auto*
    have ∗:(*T*^^*k*)−'*K* ∩ *K* ⊆ (*T*^^*k*)−'*K* ∩ *A* **using** ‹*K* ⊆ *A*› **by** *auto*
    have *posKA*: *emeasure M* ((*T*^^*k*)−'*K* ∩ *A*) > *0* **using** *emeasure-mono*[*OF* ∗, *of M*] *posK* **by** *simp*
    let *?S* = {(*i*::*nat*). *i*>*0* ∧ *emeasure M* ((*T*^^*i*)−'*K* ∩ *A*) > *0*}
    have *k* ∈ *?S* **using** ‹*k*>*0*› *posKA* **by** *simp*
    then have *?S* ≠ {} **by** *auto*
    then have *Inf ?S* ∈ *?S* **using** *Inf-nat-def1*[*of ?S*] **by** *simp*
    then have *good-time K* ∈ *?S* **using** *good-time-def* **by** *simp*
    then show (*good-time K* > *0*) ∧ (*next-good-set K* ⊆ *A*) ∧
        (*next-good-set K* ∈ *sets M*) ∧ (*emeasure M* (*next-good-set K*) > *0*)
      using *a next-good-set-def* **by** *auto*
  qed

  define *B* **where** *B* = (λ*i*. (*next-good-set*^^*i*) *A*)
  define *t* **where** *t* = (λ*i*. *good-time* (*B i*))
  have *good-B*: (*B i* ⊆ *A*) ∧ (*B i* ∈ *sets M*) ∧ (*emeasure M* (*B i*) > *0*) **for** *i*
  proof (*induction i*)
    case *0*
    have *B 0* = *A* **using** *B-def* **by** *simp*
    then show *?case* **using** ‹*B 0* = *A*› ‹*A* ∈ *sets M*› ‹*emeasure M A* > *0*› **by** *simp*
  next
    case (*Suc i*)

moreover have $B\ (i{+}1) = \textit{next-good-set}\ (B\ i)$ **using** *B-def* **by** *simp*
ultimately show *?case* **using** *good-rec*[*of B i*] **by** *auto*
qed
have *t-pos*: $\bigwedge i.\ t\ i > 0$ **using** *t-def* **by** (*simp add*: *good-B good-rec*)

define s **where** $s = (\lambda i\ k.\ (\sum n \in \{i..<i{+}k\}.\ t\ n))$
have $B\ (i{+}k) \subseteq (T\,\widehat{}\widehat{}\,(s\ i\ k)) -` A \cap A$ **for** $i\ k$
proof (*induction k*)
case *0*
show *?case* **using** *s-def good-B*[*of i*] **by** *simp*
next
case (*Suc k*)
have $B(i{+}k{+}1) = (T\,\widehat{}\widehat{}\,(t\ (i{+}k))) -` (B\ (i{+}k)) \cap A$ **using** *t-def B-def next-good-set-def* **by** *simp*
moreover have $B(i{+}k) \subseteq (T\,\widehat{}\widehat{}\,(s\ i\ k)) -` A$ **using** *Suc.IH* **by** *simp*
ultimately have $B(i{+}k{+}1) \subseteq (T\,\widehat{}\widehat{}\,(t\ (i{+}k))) -` (T\,\widehat{}\widehat{}\,(s\ i\ k)) -` A \cap A$ **by** *auto*
then have $B(i{+}k{+}1) \subseteq (T\,\widehat{}\widehat{}\,(t(i{+}k) + s\ i\ k)) -` A \cap A$ **by** (*simp add*: *add.commute funpow-add vimage-comp*)
moreover have $t(i{+}k) + s\ i\ k = s\ i\ (k{+}1)$ **using** *s-def* **by** *simp*
ultimately show *?case* **by** *simp*
qed
moreover have $(T\,\widehat{}\widehat{}\,j) -` A \cap A \in \textit{sets}\ M$ **for** j **by** *simp*
ultimately have $*$: $\textit{emeasure}\ M\ ((T\,\widehat{}\widehat{}\,(s\ i\ k)) -` A \cap A) > 0$ **for** $i\ k$
by (*metis inf.orderE inf.strict-boundedE good-B emeasure-mono*)

show $\exists k{>}0.\ 0 < \textit{emeasure}\ M\ (((T\ \widehat{}\widehat{}\ n)\ \widehat{}\widehat{}\ k) -` A \cap A)$
proof (*cases*)
assume $n = 0$
then have $((T\ \widehat{}\widehat{}\ n)\ \widehat{}\widehat{}\ 1) -` A = A$ **by** *simp*
then show *?thesis* **using** ‹$\textit{emeasure}\ M\ A > 0$› **by** *auto*
next
assume $\neg(n = 0)$
then have $n > 0$ **by** *simp*
define u **where** $u = (\lambda i.\ s\ 0\ i\ \textit{mod}\ n)$
have $\textit{range}\ u \subseteq \{..<n\}$ **by** (*simp add*: ‹$0 < n$› *image-subset-iff u-def*)
then have *finite* (*range u*) **using** *finite-nat-iff-bounded* **by** *auto*
then have $\exists\, i\ j.\ (i<j) \wedge (u\ i = u\ j)$ **by** (*metis finite-imageD infinite-UNIV-nat injI less-linear*)
then obtain $i\ k$ **where** $k{>}0\ u\ i = u\ (i{+}k)$ **using** *less-imp-add-positive* **by** *blast*
moreover have $s\ 0\ (i{+}k) = s\ 0\ i + s\ i\ k$ **unfolding** *s-def* **by** (*simp add*: *sum.atLeastLessThan-concat*)
ultimately have $(s\ i\ k)\ \textit{mod}\ n = 0$ **using** *u-def nat-mod-cong* **by** *metis*
then obtain r **where** $s\ i\ k = n * r$ **by** *auto*
moreover have $s\ i\ k > 0$ **unfolding** *s-def*
using ‹$k > 0$› *t-pos sum-strict-mono*[*of* $\{i..<i{+}k\}$, *of* $\lambda x.\ 0$, *of* $\lambda x.\ t\ x$] **by** *simp*
ultimately have $r > 0$ **by** *simp*
moreover have $\textit{emeasure}\ M\ ((T\,\widehat{}\widehat{}\,(n * r)) -` A \cap A) > 0$ **using** $*$ ‹$s\ i\ k = n$

$* \ r\rangle$ **by** *metis*
  ultimately show *?thesis* **by** (*metis funpow-mult*)
 qed
qed

proposition (**in** *conservative-mpt*) *conservative-mpt-power*:
 conservative-mpt M $(T\widehat{\ }\widehat{\ }n)$
using *conservative-power mpt-power* **unfolding** *conservative-mpt-def* **by** *auto*

The standard way to use conservativity is as follows: if a set is almost disjoint from all its preimages, then it is null:

lemma (**in** *conservative*) *ae-disjoint-then-null*:
 assumes $A \in sets\ M$
  $\bigwedge n.\ n > 0 \Longrightarrow A \cap (T\widehat{\ }\widehat{\ }n) -\text{`}A \in null\text{-}sets\ M$
 shows $A \in null\text{-}sets\ M$
by (*metis Int-commute assms*(*1*) *assms*(*2*) *conservative zero-less-iff-neq-zero null-setsD1 null-setsI*)

lemma (**in** *conservative*) *disjoint-then-null*:
 assumes $A \in sets\ M$
  $\bigwedge n.\ n > 0 \Longrightarrow A \cap (T\widehat{\ }\widehat{\ }n) -\text{`}A = \{\}$
 shows $A \in null\text{-}sets\ M$
by (*rule ae-disjoint-then-null*, *auto simp add*: *assms*)

Conservativity is preserved by replacing the measure by an equivalent one.

lemma (**in** *conservative*) *conservative-density*:
 assumes [*measurable*]: $h \in borel\text{-}measurable\ M$
  and $AE\ x\ in\ M.\ h\ x \neq 0$ $AE\ x\ in\ M.\ h\ x \neq \infty$
 shows *conservative* (*density M h*) *T*
proof −
 interpret *A*: *qmpt density M h T*
  by (*rule qmpt-density*[*OF assms*])
 show *?thesis*
  apply (*rule conservativeI3*) **apply** (*simp add*: *A.qmpt-axioms*)
  unfolding *sets-density null-sets-density*[*OF assms*(*1*) *assms*(*2*)]
  by (*metis conservative emeasure-empty not-gr-zero null-setsI*)
qed

context *qmpt* **begin**

We introduce the recurrent subset of A, i.e., the set of points of A that return to A, and the infinitely recurrent subset, i.e., the set of points of A that return infinitely often to A. In conservative systems, both coincide with A almost everywhere.

definition *recurrent-subset*::*'a set* ⇒ *'a set*
 where $recurrent\text{-}subset\ A = (\bigcup n \in \{1..\}.\ A \cap (T\widehat{\ }\widehat{\ }n) -\text{`}A)$

definition *recurrent-subset-infty*::*'a set* ⇒ *'a set*

where *recurrent-subset-infty A* = *A* − (⋃*n*. (*T*^^*n*)−' (*A* − *recurrent-subset A*))

lemma *recurrent-subset-infty-inf-returns*:
  x ∈ *recurrent-subset-infty A* ⟷ (*x* ∈ *A* ∧ *infinite* {*n*. (*T*^^*n*) *x* ∈ *A*})
proof
  assume ∗: *x* ∈ *recurrent-subset-infty A*
  have *infinite* {*n*. (*T*^^*n*) *x* ∈ *A*}
  proof (*rule ccontr*)
    assume ¬(*infinite* {*n*. (*T*^^*n*) *x* ∈ *A*})
    then have *F*: *finite* {*n*. (*T*^^*n*) *x* ∈ *A*} **by** *auto*
    have *0* ∈ {*n*. (*T*^^*n*) *x* ∈ *A*} **using** ∗ *recurrent-subset-infty-def* **by** *auto*
    then have *NE*: {*n*. (*T*^^*n*) *x* ∈ *A*} ≠ {} **by** *blast*
    define *N* **where** *N* = *Max* {*n*. (*T*^^*n*) *x* ∈ *A*}
    have *N* ∈ {*n*. (*T*^^*n*) *x* ∈ *A*} **unfolding** *N-def* **using** *F NE* **using** *Max-in* **by** *auto*
    then have (*T*^^*N*) *x* ∈ *A* **by** *auto*
    moreover have *x* ∉ (*T*^^*N*)−' (*A* − *recurrent-subset A*) **using** ∗ **unfolding** *recurrent-subset-infty-def* **by** *auto*
    ultimately have (*T*^^*N*) *x* ∈ *recurrent-subset A* **by** *auto*
    then have (*T* ^^ *N*) *x* ∈ *A* ∧ (∃*n*. *n* ∈ {*1*..} ∧ (*T* ^^ *n*) ((*T* ^^ *N*) *x*) ∈ *A*)
      unfolding *recurrent-subset-def* **by** *blast*
    then obtain *n* **where** *n*>*0* (*T*^^*n*) ((*T*^^*N*) *x*) ∈ *A*
      by (*metis atLeast-iff gr0I not-one-le-zero*)
    then have *n*+*N* ∈ {*n*. (*T*^^*n*) *x* ∈ *A*} **by** (*simp add*: *funpow-add*)
    then show *False* **unfolding** *N-def* **using** ‹*n*>*0*› *F NE*
      by (*metis Max-ge Nat.add-0-right add.commute nat-add-left-cancel-less not-le*)
  qed
  then show *x* ∈ *A* ∧ *infinite* {*n*. (*T*^^*n*) *x* ∈ *A*} **using** ∗ *recurrent-subset-infty-def* **by** *auto*
next
  assume ∗: (*x* ∈ *A* ∧ *infinite* {*n*. (*T* ^^ *n*) *x* ∈ *A*})
  {
    fix *n*
    obtain *N* **where** *N*>*n* (*T*^^*N*) *x* ∈ *A* **using** ∗
      using *infinite-nat-iff-unbounded* **by** *force*
    define *k* **where** *k* = *N*−*n*
    then have *k*>*0* *N* = *n*+*k* **using** ‹*N*>*n*› **by** *auto*
    then have (*T*^^*k*) ((*T*^^*n*) *x*) ∈ *A*
      by (*metis* ‹(*T* ^^ *N*) *x* ∈ *A*› ‹*N* = *n* + *k*› *add.commute comp-def funpow-add*)
    then have (*T*^^ *n*) *x* ∉ *A* − *recurrent-subset A*
      unfolding *recurrent-subset-def* **using** ‹*k*>*0*› **by** *auto*
  }
  then show *x* ∈ *recurrent-subset-infty A* **unfolding** *recurrent-subset-infty-def* **using** ∗ **by** *auto*
qed

lemma *recurrent-subset-infty-series-infinite*:
  assumes *x* ∈ *recurrent-subset-infty A*

shows ($\sum$ *n. indicator A* ((*T*^^*n*) *x*)) = (∞::*ennreal*)
proof (*rule ennreal-ge-nat-imp-PInf*)
have ∗: ¬ *finite* {*n.* (*T*^^*n*) *x* ∈ *A*} **using** *recurrent-subset-infty-inf-returns assms*
by *auto*
fix *N*::*nat*
obtain *F* **where** *F*: *finite F F* ⊆ {*n.* (*T*^^*n*) *x* ∈ *A*} *card F* = *N*
using *infinite-arbitrarily-large*[*OF* ∗] **by** *blast*
have *N* = ($\sum$ *n* ∈ *F*. *1*::*ennreal*)
using *F*(*3*) **by** *auto*
also have ... = ($\sum$ *n* ∈ *F*. (*indicator A* ((*T*^^*n*) *x*))::*ennreal*)
apply (*rule sum.cong*) **using** *F*(*2*) *indicator-def* **by** *auto*
also have ... ≤ ($\sum$ *n. indicator A* ((*T*^^*n*) *x*))
by (*rule sum-le-suminf*, *auto simp add*: *F*)
finally show *N* ≤ ($\sum$ *n.* (*indicator A* ((*T*^^*n*) *x*))::*ennreal*) **by** *auto*
qed

lemma *recurrent-subset-infty-def'*:
recurrent-subset-infty A = ($\bigcap$ *m.* ($\bigcup$ *n*∈{*m*..}. *A* ∩ (*T*^^*n*)−‘*A*))
proof (*auto*)
fix *x* **assume** *x*: *x* ∈ *recurrent-subset-infty A*
then show *x* ∈ *A* **unfolding** *recurrent-subset-infty-def* **by** *auto*
fix *N*::*nat*
show ∃ *n*∈{*N*..}. (*T*^^*n*) *x* ∈ *A* **using** *recurrent-subset-infty-inf-returns x*
using *infinite-nat-iff-unbounded-le* **by** *auto*
next
fix *x* **assume** *x* ∈ *A* ∀ *N*. ∃ *n*∈{*N*..}. (*T*^^*n*) *x* ∈ *A*
then show *x* ∈ *recurrent-subset-infty A*
unfolding *recurrent-subset-infty-inf-returns* **using** *infinite-nat-iff-unbounded-le*
by *auto*
qed

lemma *recurrent-subset-incl*:
recurrent-subset A ⊆ *A*
recurrent-subset-infty A ⊆ *A*
recurrent-subset-infty A ⊆ *recurrent-subset A*
unfolding *recurrent-subset-def recurrent-subset-infty-def'* **by** (*simp*, *simp*, *fast*)

lemma *recurrent-subset-meas* [*measurable*]:
assumes [*measurable*]: *A* ∈ *sets M*
shows *recurrent-subset A* ∈ *sets M*
recurrent-subset-infty A ∈ *sets M*
unfolding *recurrent-subset-def recurrent-subset-infty-def'* **by** *measurable*

lemma *recurrent-subset-rel-incl*:
assumes *A* ⊆ *B*
shows *recurrent-subset A* ⊆ *recurrent-subset B*
recurrent-subset-infty A ⊆ *recurrent-subset-infty B*
proof −
show *recurrent-subset A* ⊆ *recurrent-subset B*

unfolding *recurrent-subset-def* **using** *assms* **by** *auto*
show *recurrent-subset-infty* $A \subseteq$ *recurrent-subset-infty* B
apply (*auto*, *subst recurrent-subset-infty-inf-returns*)
using *assms recurrent-subset-incl*(*2*) *infinite-nat-iff-unbounded-le recurrent-subset-infty-inf-returns*
by *fastforce*
qed

If a point belongs to the infinitely recurrent subset of A, then when they return to A its iterates also belong to the infinitely recurrent subset.

lemma *recurrent-subset-infty-returns*:
assumes $x \in$ *recurrent-subset-infty* A $(T\widehat{}\widehat{}n)\ x \in A$
shows $(T\widehat{}\widehat{}n)\ x \in$ *recurrent-subset-infty* A
proof (*subst recurrent-subset-infty-inf-returns*, *rule ccontr*)
assume $\neg$ $((T\ \widehat{}\widehat{}\ n)\ x \in A \wedge$ *infinite* $\{k.\ (T\ \widehat{}\widehat{}\ k)\ ((T\ \widehat{}\widehat{}\ n)\ x) \in A\})$
then have *1*: *finite* $\{k.\ (T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)\ x) \in A\}$ **using** *assms*(*2*) **by** *auto*
have $0 \in \{k.\ (T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)\ x) \in A\}$ **using** *assms*(*2*) **by** *auto*
then have *2*: $\{k.\ (T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)\ x) \in A\} \neq \{\}$ **by** *blast*
define M **where** $M = Max\ \{k.\ (T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)\ x) \in A\}$
have *M-prop*: $\bigwedge k.\ k > M \Longrightarrow (T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)\ x) \notin A$
unfolding *M-def* **using** *1 2* **by** *auto*
{
fix N **assume** $*$: $(T\widehat{}\widehat{}N)\ x \in A$
have $N \leq n+M$
proof (*cases*)
assume $N \leq n$
then show *?thesis* **by** *auto*
next
assume $\neg(N \leq n)$
then have $N > n$ **by** *simp*
define k **where** $k = N-n$
have $N = n + k$ **unfolding** *k-def* **using** ‹$N > n$› **by** *auto*
then have $(T\widehat{}\widehat{}k)\ ((T\widehat{}\widehat{}n)x) \in A$ **using** $*$ **by** (*simp add*: *add.commute funpow-add*)
then have $k \leq M$ **using** *M-prop* **using** *not-le* **by** *blast*
then show *?thesis* **unfolding** *k-def* **by** *auto*
qed
}
then have *finite* $\{N.\ (T\widehat{}\widehat{}N)\ x \in A\}$
by (*metis* (*no-types*, *lifting*) *infinite-nat-iff-unbounded mem-Collect-eq not-less*)
moreover have *infinite* $\{N.\ (T\widehat{}\widehat{}N)\ x \in A\}$
using *recurrent-subset-infty-inf-returns assms*(*1*) **by** *auto*
ultimately show *False* **by** *auto*
qed

lemma *recurrent-subset-of-recurrent-subset*:
recurrent-subset-infty(*recurrent-subset-infty* A) = *recurrent-subset-infty* A
proof
show *recurrent-subset-infty* (*recurrent-subset-infty* A) $\subseteq$ *recurrent-subset-infty* A
using *recurrent-subset-incl*(*2*)[*of* A] *recurrent-subset-rel-incl*(*2*) **by** *auto*

show *recurrent-subset-infty A* ⊆ *recurrent-subset-infty* (*recurrent-subset-infty A*)
using *recurrent-subset-infty-returns recurrent-subset-infty-inf-returns*
by (*metis* (*no-types*, *lifting*) *Collect-cong subsetI*)
qed

The Poincare recurrence theorem states that almost every point of A returns (infinitely often) to A, i.e., the recurrent and infinitely recurrent subsets of A coincide almost everywhere with A. This is essentially trivial in conservative systems, as it is a reformulation of the definition of conservativity. (What is not trivial, and has been proved above, is that it is true in finite measure preserving systems, i.e., finite measure preserving systems are automatically conservative.)

theorem (**in** *conservative*) *Poincare-recurrence-thm*:
assumes [*measurable*]: *A* ∈ *sets M*
shows *A* − *recurrent-subset A* ∈ *null-sets M*
A − *recurrent-subset-infty A* ∈ *null-sets M*
A Δ *recurrent-subset A* ∈ *null-sets M*
A Δ *recurrent-subset-infty A* ∈ *null-sets M*
emeasure M (*recurrent-subset A*) = *emeasure M A*
emeasure M (*recurrent-subset-infty A*) = *emeasure M A*
AE x ∈ *A in M*. *x* ∈ *recurrent-subset-infty A*
proof −
define *B* **where** *B* = {*x* ∈ *A*. ∀ *n*∈{*1*..}. (*T*^^*n*) *x* ∈ (*space M* − *A*)}

have *rs*: *recurrent-subset A* = *A* − *B*
by (*auto simp add*: *B-def recurrent-subset-def*)
(*meson Tn-meas assms measurable-space sets.sets-into-space subsetCE*)
then have ∗: *A* − *recurrent-subset A* = *B* **using** *B-def* **by** *blast*
have *B* ∈ *null-sets M*
by (*rule disjoint-then-null*, *auto simp add*: *B-def*)
then show *A* − *recurrent-subset A* ∈ *null-sets M* **using** ∗ **by** *simp*

then have ∗: (⋃ *n*. (*T*^^*n*)−−‘(*A*−*recurrent-subset A*)) ∈ *null-sets M*
using *T-quasi-preserves-null2*(*2*) **by** *blast*
have *recurrent-subset-infty A* = *recurrent-subset-infty A* ∩ *space M* **using** *sets.sets-into-space* **by** *auto*
also have ... = *A* ∩ *space M* − (⋃ *n*. (*T*^^*n*)−‘(*A*−*recurrent-subset A*) ∩ *space M*) **unfolding** *recurrent-subset-infty-def* **by** *blast*
also have ... = *A* − (⋃ *n*. (*T*^^*n*)−−‘(*A*−*recurrent-subset A*)) **unfolding** *vimage-restr-def* **using** *sets.sets-into-space* **by** *auto*
finally have ∗∗: *recurrent-subset-infty A* = *A* − (⋃ *n*. (*T* ^^ *n*) −−‘ (*A* − *recurrent-subset A*)) **.**
then have *A* − *recurrent-subset-infty A* ⊆ (⋃ *n*. (*T*^^*n*)−−‘(*A*−*recurrent-subset A*)) **by** *auto*
with ∗ ∗∗ **show** *A* − *recurrent-subset-infty A* ∈ *null-sets M*
by (*simp add*: *Diff-Diff-Int null-set-Int1*)

have *A* Δ *recurrent-subset A* = *A* − *recurrent-subset A* **using** *recurrent-subset-incl*(*1*)[*of*

A] **by** *blast*
then show *A* Δ *recurrent-subset A* ∈ *null-sets M* **using** ‹*A* − *recurrent-subset A* ∈ *null-sets M*› **by** *auto*
then show *emeasure M* (*recurrent-subset A*) = *emeasure M A*
by (*rule Delta-null-same-emeasure*[*symmetric*], *auto*)

have *A* Δ *recurrent-subset-infty A* = *A* − *recurrent-subset-infty A* **using** *recurrent-subset-incl*(*2*)[*of A*] **by** *blast*
then show *A* Δ *recurrent-subset-infty A* ∈ *null-sets M* **using** ‹*A* − *recurrent-subset-infty A* ∈ *null-sets M*› **by** *auto*
then show *emeasure M* (*recurrent-subset-infty A*) = *emeasure M A*
by (*rule Delta-null-same-emeasure*[*symmetric*], *auto*)

show *AE x*∈*A in M*. *x* ∈ *recurrent-subset-infty A*
unfolding *eventually-ae-filter*
by (*metis* (*no-types*, *lifting*) *DiffI* ‹*A* − *recurrent-subset-infty A* ∈ *null-sets M*› *mem-Collect-eq subsetI*)
qed

A convenient way to use conservativity is given in the following theorem: if T is conservative, then the series $\sum_n f(T^n x)$ is infinite for almost every x with $f x > 0$. When f is an indicator function, this is the fact that, starting from B, one returns infinitely many times to B almost surely. The general case follows by approximating f from below by constants time indicators.

theorem (**in** *conservative*) *recurrence-series-infinite*:
fixes *f*::*'a* ⇒ *ennreal*
assumes [*measurable*]: *f* ∈ *borel-measurable M*
shows *AE x in M*. *f x* > *0* ⟶ (∑ *n*. *f* ((*T*^^*n*) *x*)) = ∞
proof −
have ∗: *AE x in M*. *f x* > *epsilon* ⟶ (∑ *n*. *f* ((*T*^^*n*) *x*)) = ⊤ **if** *epsilon* > *0* **for** *epsilon*
proof −
define *B* **where** *B* = {*x* ∈ *space M*. *f x* > *epsilon*}
have [*measurable*]: *B* ∈ *sets M* **unfolding** *B-def* **by** *auto*
have (∑ *n*. *f* ((*T*^^*n*) *x*)) = ∞ **if** *x* ∈ *recurrent-subset-infty B* **for** *x*
proof −
have ∞ = *epsilon* ∗ ∞ **using** ‹*epsilon* > *0*› *ennreal-mult-top* **by** *auto*
also have ... = *epsilon* ∗ (∑ *n*. *indicator B* ((*T*^^*n*) *x*))
using *recurrent-subset-infty-series-infinite*[*OF that*] **by** *simp*
also have ... = (∑ *n*. *epsilon* ∗ *indicator B* ((*T*^^*n*) *x*))
by *auto*
also have ... ≤ (∑ *n*. *f* ((*T*^^*n*) *x*))
apply (*rule suminf-le*) **unfolding** *indicator-def B-def* **by** *auto*
finally show *?thesis*
by (*simp add*: *dual-order.antisym*)
qed
moreover have *AE x in M*. *f x* > *epsilon* ⟶ *x* ∈ *recurrent-subset-infty B*
using *Poincare-recurrence-thm*(*7*)[*OF* ‹*B* ∈ *sets M*›] **unfolding** *B-def* **by** *auto*

ultimately show *?thesis* **by** *auto*
qed
have $\exists\, u::(nat \Rightarrow ennreal).\ (\forall\, n.\ u\ n > 0) \wedge u \longlongrightarrow 0$
by (*meson approx-from-above-dense-linorder ex-gt-or-lt gr-implies-not-zero*)
then obtain $u::nat \Rightarrow ennreal$ **where** u: $\bigwedge n.\ u\ n > 0\ u \longrightarrow 0$
by *auto*
have $AE\ x\ in\ M.\ (\forall\, n::nat.\ (f\ x > u\ n \longrightarrow (\sum n.\ f\ ((T\widehat{\ }\widehat{\ }n)\ x)) = \top))$
unfolding *AE-all-countable* **using** *u* **by** (*auto intro*!: ∗)
moreover have $f\ x > 0 \longrightarrow (\sum n.\ f\ ((T\widehat{\ }\widehat{\ }n)\ x)) = \infty$ **if** $(\forall\, n::nat.\ (f\ x > u\ n \longrightarrow (\sum n.\ f\ ((T\widehat{\ }\widehat{\ }n)\ x)) = \top))$ **for** *x*
proof (*auto*)
assume $f\ x > 0$
obtain *n* **where** $u\ n < f\ x$
using *order-tendstoD*(*2*)[*OF u*(*2*) ‹$f\ x > 0$›] *eventually-False-sequentially eventually-mono* **by** *blast*
then show $(\sum n.\ f\ ((T\widehat{\ }\widehat{\ }n)\ x)) = \top$ **using** *that* **by** *auto*
qed
ultimately show *?thesis* **by** *auto*
qed

5.2 The first return time

The first return time to a set A under the dynamics T is the smallest integer n such that $T^n(x) \in A$. The first return time is only well defined on the recurrent subset of A, elsewhere we set it to 0 for definiteness. We can partition A according to the value of the return time on it, thus defining the return partition of A.

definition *return-time-function*::*'a set* ⇒ (*'a* ⇒ *nat*)
where *return-time-function A x* = (
if ($x \in$ *recurrent-subset A*) *then* (*Inf* $\{n::nat \in \{1..\}.\ (T\widehat{\ }\widehat{\ }n)\ x \in A\}$)
else 0)

definition *return-partition*::*'a set* ⇒ *nat* ⇒ *'a set*
where *return-partition A k* $= A \cap (T\widehat{\ }\widehat{\ }k) {-}{-}` A - (\bigcup i \in \{0<..<k\}.\ (T\widehat{\ }\widehat{\ }i) {-}{-}` A)$

Basic properties of the return partition.

lemma *return-partition-basics*:
assumes *A-meas* [*measurable*]: $A \in$ *sets M*
shows [*measurable*]: *return-partition A n* ∈ *sets M*
and *disjoint-family* (λ*n*. *return-partition A* (*n*+*1*))
($\bigcup n$. *return-partition A* (*n*+*1*)) = *recurrent-subset A*
proof −
show *return-partition A n* ∈ *sets M* **for** *n* **unfolding** *return-partition-def* **by** *auto*

define *B* **where** $B = (\lambda n.\ A \cap (T\widehat{\ }\widehat{\ }(n+1)) {-}{-}` A)$
have *return-partition A* (*n*+*1*) $= B\ n - (\bigcup i \in \{0..<n\}.\ B\ i)$ **for** *n*
unfolding *return-partition-def B-def* **by** (*auto*) (*auto simp add*: *less-Suc-eq-0-disj*)

then have ∗: $\bigwedge n$. *return-partition A* (*n+1*) = *disjointed B n* **using** *disjointed-def*[*of B*] **by** *simp*
then show *disjoint-family* (λ*n. return-partition A* (*n+1*)) **using** *disjoint-family-disjointed* **by** *simp*

have *A* ∩ (*T*^^*n*)−'*A* = *A* ∩ (*T*^^*n*)−−'*A* **for** *n*
using *sets.sets-into-space*[*OF A-meas*] **by** *auto*
then have *recurrent-subset A* = ($\bigcup$*n*∈ {*1..*}. *A* ∩ (*T*^^*n*)−−'*A*) **unfolding** *recurrent-subset-def* **by** *simp*
also have ... = ($\bigcup$ *n. B n*) **by** (*simp add*: *B-def atLeast-Suc-greaterThan greaterThan-0*)
also have ... = ($\bigcup$ *n. return-partition A* (*n+1*)) **using** ∗ *UN-disjointed-eq*[*of B*] **by** *simp*
finally show ($\bigcup$ *n. return-partition A* (*n+1*)) = *recurrent-subset A* **by** *simp*
qed

Basic properties of the return time, relationship with the return partition.

lemma *return-time0*:
(*return-time-function A*)−'{*0*} = *UNIV* − *recurrent-subset A*
proof (*auto*)
fix *x*
assume ∗: *x* ∈ *recurrent-subset A return-time-function A x* = *0*
define *K* **where** *K* = {*n*::*nat*∈{*1..*}. (*T*^^*n*) *x* ∈ *A*}
have ∗∗: *return-time-function A x* = *Inf K*
using *K-def return-time-function-def* ∗ **by** *simp*
have *K* ≠ {} **using** *K-def recurrent-subset-def* ∗ **by** *auto*
moreover have *0* ∉ *K* **using** *K-def* **by** *auto*
ultimately have *Inf K* >*0*
by (*metis* (*no-types*, *lifting*) *K-def One-nat-def atLeast-iff cInf-lessD mem-Collect-eq neq0-conv not-le zero-less-Suc*)
then have *return-time-function A x* > *0* **using** ∗∗ **by** *simp*
then show *False* **using** ∗ **by** *simp*
qed (*auto simp add*: *return-time-function-def*)

lemma *return-time-n*:
assumes [*measurable*]: *A* ∈ *sets M*
shows (*return-time-function A*)−'{*Suc n*} = *return-partition A* (*Suc n*)
proof (*auto*)
fix *x* **assume** ∗: *return-time-function A x* = *Suc n*
then have *rx*: *x* ∈ *recurrent-subset A* **using** *return-time-function-def* **by** (*auto*, *meson Zero-not-Suc*)
define *K* **where** *K* = {*i*∈{*1..*}. (*T*^^*i*) *x* ∈ *A*}
have *return-time-function A x* = *Inf K* **using** *return-time-function-def rx K-def* **by** *auto*
then have *Inf K* = *Suc n* **using** ∗ **by** *simp*
moreover have *K* ≠ {} **using** *rx recurrent-subset-def K-def* **by** *auto*
ultimately have *Suc n* ∈ *K* **using** *Inf-nat-def1*[*of K*] **by** *simp*
then have (*T*^^(*Suc n*))*x* ∈ *A* **using** *K-def* **by** *auto*
then have *a*: *x* ∈ *A* ∩ (*T*^^(*Suc n*))−−'*A*
using *rx recurrent-subset-incl*[*of A*] *sets.sets-into-space*[*OF assms*] **by** *auto*

have $\bigwedge i.\ i \in \{1..<Suc\ n\} \Longrightarrow i \notin K$ **using** *cInf-lower* ‹*Inf K = Suc n*› **by** *force*
then have $\bigwedge i.\ i \in \{1..<Suc\ n\} \Longrightarrow x \notin (T\widehat{\ }\widehat{\ }i) -- ` A$ **using** *K-def* **by** *auto*
then have $x \notin (\bigcup i \in \{1..<Suc\ n\}.\ (T\widehat{\ }\widehat{\ }i) -- ` A)$ **by** *auto*
then show $x \in$ *return-partition A* (*Suc n*) **using** *a return-partition-def* **by** *simp*
next
fix *x* **assume** ∗: $x \in$ *return-partition A* (*Suc n*)
then have *a*: $x \in$ *space M* **unfolding** *return-partition-def* **using** *vimage-restr-def* **by** *blast*
define *K* **where** $K = \{i::nat \in \{1..\}.\ (T\widehat{\ }\widehat{\ }i)\ x \in A\}$
have *Inf K = Suc n*
apply (*rule cInf-eq-minimum*) **using** ∗ **by** (*auto simp add*: *a assms K-def return-partition-def*)

have $x \in$ *recurrent-subset A* **using** ∗ *return-partition-basics*(*3*)[*OF assms*] **by** *auto*
then show *return-time-function A x = Suc n*
using *return-time-function-def K-def* ‹*Inf K = Suc n*› **by** *auto*
qed

The return time is measurable.

lemma *return-time-function-meas* [*measurable*]:
assumes [*measurable*]: $A \in$ *sets M*
shows *return-time-function A* $\in$ *measurable M* (*count-space UNIV*)
return-time-function A $\in$ *borel-measurable M*
proof −
have (*return-time-function A*)$-`\{n\} \cap$ *space M* $\in$ *sets M* **for** *n*
proof (*cases n = 0*)
case *True*
then show *?thesis* **using** *return-time0 recurrent-subset-meas*[*OF assms*] **by** *auto*
next
case *False*
show *?thesis*
using *return-time-n return-partition-basics*(*1*)[*OF assms*] *not0-implies-Suc*[*OF False*] **by** *auto*
qed
then show *return-time-function A* $\in$ *measurable M* (*count-space UNIV*)
by (*simp add*: *measurable-count-space-eq2-countable assms*)
then show *return-time-function A* $\in$ *borel-measurable M*
using *measurable-cong-sets sets-borel-eq-count-space* **by** *blast*
qed

A close cousin of the return time and the return partition is the first entrance set: we partition the space according to the first positive time where a point enters *A*.

definition *first-entrance-set*::$'a$ *set* ⇒ *nat* ⇒ $'a$ *set*
where *first-entrance-set A n* $= (T\widehat{\ }\widehat{\ }n) -- ` A - (\bigcup i<n.\ (T\widehat{\ }\widehat{\ }i) -- ` A)$

lemma *first-entrance-meas* [*measurable*]:

assumes [*measurable*]: *A* ∈ *sets M*
shows *first-entrance-set A n* ∈ *sets M*
unfolding *first-entrance-set-def* **by** *measurable*

lemma *first-entrance-disjoint*:
disjoint-family (*first-entrance-set A*)
proof −
have *first-entrance-set A* = *disjointed* (λ*i*. (*T*^^*i*)−−‘*A*)
by (*auto simp add*: *disjointed-def first-entrance-set-def*)
then show *?thesis* **by** (*simp add*: *disjoint-family-disjointed*)
qed

There is an important dynamical phenomenon: if a point has first entrance time equal to n, then their preimages either have first entrance time equal to $n+1$ (these are the preimages not in A) or they belong to A and have first return time equal to $n+1$. When T preserves the measure, this gives an inductive control on the measure of the first entrance set, that will be used again and again in the proof of Kac's Formula. We formulate these (simple but extremely useful) facts now.

lemma *first-entrance-rec*:
assumes [*measurable*]: *A* ∈ *sets M*
shows *first-entrance-set A* (*Suc n*) = *T*−−‘(*first-entrance-set A n*) − *A*
proof −
have *A0*: *A* = (*T*^^*0*)−−‘*A* **by** *auto*
have *first-entrance-set A n* = (*T*^^*n*) −−‘ *A* − (⋃ *i*<*n*. (*T*^^*i*)−−‘*A*)
using *first-entrance-set-def* **by** *simp*
then have *T*−−‘(*first-entrance-set A n*) = (*T*^^(*n*+*1*))−−‘*A* − (⋃ *i*<*n*. (*T*^^(*i*+*1*))−−‘*A*)
using *T-vrestr-composed*(*2*) ‹*A* ∈ *sets M*› **by** *simp*
then have ∗: *T*−−‘(*first-entrance-set A n*) − *A* = (*T*^^(*n*+*1*))−−‘*A* − (*A* ∪ (⋃ *i*<*n*. (*T*^^(*i*+*1*))−−‘*A*))
by *blast*
have (⋃ *i*<*n*. (*T*^^(*i*+*1*))−−‘*A*) = (⋃ *j*∈{*1*..<*n*+*1*}. (*T*^^*j*)−−‘*A*)
by (*rule UN-le-add-shift-strict*)
then have *A* ∪ (⋃ *i*<*n*. (*T*^^(*i*+*1*))−−‘*A*) = (⋃ *j*∈{*0*..<*n*+*1*}. (*T*^^*j*)−−‘*A*)
by (*metis A0 Un-commute atLeast0LessThan UN-le-eq-Un0-strict*)
then show *?thesis* **using** ∗ *first-entrance-set-def* **by** *auto*
qed

lemma *return-time-rec*:
assumes *A* ∈ *sets M*
shows (*return-time-function A*)−‘{*Suc n*} = *T*−−‘(*first-entrance-set A n*) ∩ *A*
proof −
have *return-partition A* (*Suc n*) = *T*−−‘(*first-entrance-set A n*) ∩ *A*
unfolding *return-partition-def first-entrance-set-def*
by (*auto simp add*: *T-vrestr-composed*[*OF assms*]) (*auto simp add*: *less-Suc-eq-0-disj*)
then show *?thesis* **using** *return-time-n*[*OF assms*] **by** *simp*
qed

5.3 Local time controls

The local time is the time that an orbit spends in a given set. Local time controls are basic to all the forthcoming developments.

definition *local-time*::*'a set* ⇒ *nat* ⇒ *'a* ⇒ *nat*
where *local-time A n x* = *card* {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*}

lemma *local-time-birkhoff*:
local-time A n x = *birkhoff-sum* (*indicator A*) *n x*
proof (*induction n*)
case *0*
then show *?case* **unfolding** *local-time-def birkhoff-sum-def* **by** *simp*
next
case (*Suc n*)
have *local-time A* (*n+1*) *x* = *local-time A n x* + *indicator A* ((*T*^^*n*) *x*)
proof (*cases*)
assume ∗: (*T*^^*n*) *x* ∈ *A*
then have {*i*∈{..<*Suc n*}. (*T*^^*i*) *x* ∈ *A*} = {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*} ∪ {*n*}
by *auto*
then have *card* {*i*∈{..<*Suc n*}. (*T*^^*i*) *x* ∈ *A*} = *card* {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*} + *card* {*n*}
using *card-Un-disjoint* **by** *auto*
then have *local-time A* (*n+1*) *x* = *local-time A n x* + *1* **using** *local-time-def* **by** *simp*
moreover have *indicator A* ((*T*^^*n*)*x*) = (*1*::*nat*) **using** ∗ *indicator-def* **by** *auto*
ultimately show *?thesis* **by** *simp*
next
assume ∗: ¬((*T*^^*n*) *x* ∈ *A*)
then have {*i*∈{..<*Suc n*}. (*T*^^*i*) *x* ∈ *A*} = {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*} **using** *less-Suc-eq* **by** *force*
then have *card* {*i*∈{..<*Suc n*}. (*T*^^*i*) *x* ∈ *A*} = *card* {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*}
by *auto*
then have *local-time A* (*n+1*) *x* = *local-time A n x* **using** *local-time-def* **by** *simp*
moreover have *indicator A* ((*T*^^*n*)*x*) = (*0*::*nat*) **using** ∗ *indicator-def* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed
then have *local-time A* (*n+1*) *x* = *birkhoff-sum* (*indicator A*) *n x* + *indicator A* ((*T*^^*n*) *x*)
using *Suc.IH* **by** *auto*
moreover have *birkhoff-sum* (*indicator A*) (*n+1*) *x* = *birkhoff-sum* (*indicator A*) *n x* + *indicator A* ((*T*^^*n*) *x*)
by (*metis birkhoff-sum-cocycle*[**where** *?n* = *n* **and** *?m* = *1*] *birkhoff-sum-1*(*2*))
ultimately have *local-time A* (*n+1*) *x* = *birkhoff-sum* (*indicator A*) (*n+1*) *x* **by** *metis*
then show *?case* **by** (*metis Suc-eq-plus1*)

qed

lemma *local-time-meas* [*measurable*]:
  assumes [*measurable*]: *A* ∈ *sets M*
  shows *local-time A n* ∈ *borel-measurable M*
unfolding *local-time-birkhoff* **by** *auto*

lemma *local-time-cocycle*:
  local-time A n x + *local-time A m* ((*T*^^*n*)*x*) = *local-time A* (*n*+*m*) *x*
by (*metis local-time-birkhoff birkhoff-sum-cocycle*)

lemma *local-time-incseq*:
  incseq (λ*n*. *local-time A n x*)
using *local-time-cocycle incseq-def* **by** (*metis le-iff-add*)

lemma *local-time-Suc*:
  local-time A (*n*+*1*) *x* = *local-time A n x* + *indicator A* ((*T*^^*n*)*x*)
by (*metis local-time-birkhoff birkhoff-sum-cocycle birkhoff-sum-1*(*2*))

The local time is bounded by n: at most, one returns to A all the time!

lemma *local-time-bound*:
  local-time A n x ≤ *n*
proof −
  have *card* {*i*∈{..<*n*}. (*T*^^*i*) *x* ∈ *A*} ≤ *card* {..<*n*} **by** (*rule card-mono*, *auto*)
  then show *?thesis* **unfolding** *local-time-def* **by** *auto*
qed

The fact that local times are unbounded will be the main technical tool in the proof of recurrence results or Kac formula below. In this direction, we prove more and more general results in the lemmas below.

We show that, in $T^{-n}(A)$, the number of visits to A tends to infinity in measure, when A has finite measure. In other words, the points in $T^{-n}(A)$ with local time $< k$ have a measure tending to 0 with k. The argument, by induction on k, goes as follows.

Consider the last return to A before time n, say at time $n-i$. It lands in the set S_i with retun time i. We get $T^{-n}A \subseteq \bigcup_{n<N} T^{-(n-i)}S_i \cup R$, where the union is disjoint and R is a set of measure $\mu(T^{-n}A) - \sum_{n<N} \mu(T^{-(n-i)}S_i) = \mu(A) - \sum_{n<N} \mu(S_i)$, which tends to 0 with N and that we may therefore discard. A point with local time $< k$ at time n in $T^{-n}A$ is then a point with local time $< k-1$ at time $n-i$ in $T^{-(n-i)}S_i \subseteq T^{-(n-i)}A$. Hence, we may conclude by the induction assumption that this has small measure.

lemma (**in** *conservative-mpt*) *local-time-unbounded1*:
  assumes *A-meas* [*measurable*]: *A* ∈ *sets M*
    and *fin*: *emeasure M A* < ∞
  shows (λ*n*. *emeasure M* {*x* ∈ (*T*^^*n*)−−‘*A*. *local-time A n x* < *k*}) ⟶ *0*
proof (*induction k*)
  case *0*

```
  have {x ∈ (T^^n)−−‘A. local-time A n x < 0} = {} for n by simp
  then show ?case by simp
next
  case (Suc k)
  define K where K = (λp n. {x ∈ (T^^n)−−‘A. local-time A n x < p})
  have K-meas [measurable]: K p n ∈ sets M for n p
    unfolding K-def by measurable

  show ?case
  proof (rule tendsto-zero-ennreal)
    fix e :: real assume 0 < e
    define e2 where e2 = e/3
    have e2 > 0 using e2-def ‹e>0› by simp
     have (∑n. emeasure M (return-partition A (n+1))) = emeasure M ((⋃n.
return-partition A (n + 1)))
      apply (rule suminf-emeasure) using return-partition-basics[OF A-meas] by
auto
    also have ... = emeasure M (recurrent-subset A)
      using return-partition-basics(3)[OF A-meas] by simp
    also have ... = emeasure M A
     by (metis A-meas double-diff emeasure-Diff-null-set order-refl Poincare-recurrence-thm(1)[OF
A-meas] recurrent-subset-incl(1))
    finally have (∑n. emeasure M (return-partition A (n+1))) = emeasure M A
by simp
    moreover have summable (λn. emeasure M (return-partition A (n+1)))
      by simp
     ultimately have (λN. (∑n<N. emeasure M (return-partition A (n+1))))
⟶ emeasure M A
      unfolding sums-def[symmetric] sums-iff by simp
     then have (λN. (∑n<N. emeasure M (return-partition A (n+1))) + e2)
⟶ emeasure M A + e2
      by (intro tendsto-add) auto
    moreover have emeasure M A < emeasure M A + e2
      using ‹emeasure M A < ∞› ‹0 < e2› by auto
     ultimately have eventually (λN. (∑n<N. emeasure M (return-partition A
(n+1))) + e2 > emeasure M A) sequentially
      by (simp add: order-tendsto-iff)
   then obtain N where N>0 and largeM: (∑n<N. emeasure M (return-partition
A (n+1))) + e2 > emeasure M A
     by (metis (no-types, lifting) add.commute add-Suc-right eventually-at-top-linorder
le-add2 zero-less-Suc)

     have upper: emeasure M (K (Suc k) n) ≤ e2 + (∑i<N. emeasure M (K k
(n−i−1))) if n>N for n
    proof −
      define B where B = (λi. (T^^(n−i−1))−−‘(return-partition A (i+1)))
     have B-meas [measurable]: B i ∈ sets M for i unfolding B-def by measurable
      have disj-B: disjoint-family-on B {..<N}
      proof −
```

have *B i* ∩ *B j* = {} **if** *i*∈{..<*N*} *j*∈{..<*N*} *i* < *j* **for** *i j*
proof −
have *n* > *i n*>*j* **using** ‹*n*>*N*› *that* **by** *auto*
let *?k* = *j*−*i*
have *x* ∉ *B i* **if** *x* ∈ *B j* **for** *x*
proof −
have (*T*^^(*n*−*j*−*1*)) *x* ∈ *return-partition A* (*j*+*1*) **using** *B-def that* **by** *auto*
moreover have *?k*>*0* **using** ‹*i* < *j*› **by** *simp*
moreover have *?k* < *j*+*1* **by** *simp*
ultimately have (*T*^^(*n*−*j*−*1*)) *x* ∉ (*T*^^*?k*)−−‘*A* **using** *return-partition-def* **by** *auto*
then have *x* ∉ (*T*^^(*n*−*j*−*1*))−−‘ (*T*^^*?k*)−−‘*A* **by** *auto*
then have *x* ∉ (*T*^^(*n*−*j*−*1* + *?k*))−−‘*A* **using** *T-vrestr-composed*[*OF A-meas*] **by** *simp*
then have *x* ∉ (*T*^^(*n*−*i*−*1*))−−‘*A* **using** ‹*i*<*j*› ‹*n*>*j*› **by** *auto*
then have *x* ∉ (*T*^^(*n*−*i*−*1*))−−‘ (*return-partition A* (*i*+*1*)) **using** *return-partition-def* **by** *auto*
then show *x* ∉ *B i* **using** *B-def* **by** *auto*
qed
then show *B i* ∩ *B j* = {} **by** *auto*
qed
then have ⋀*i j*. *i*∈{..<*N*} ⟹ *j*∈{..<*N*} ⟹ *i* ≠ *j* ⟹ *B i* ∩ *B j* = {}
by (*metis Int-commute linorder-neqE-nat*)
then show *?thesis* **unfolding** *disjoint-family-on-def* **by** *auto*
qed

have *incl-B*: *B i* ⊆ (*T*^^*n*)−−‘*A* **if** *i* ∈ {..<*N*} **for** *i*
proof −
have *n* > *i* **using** ‹*n*>*N*› *that* **by** *auto*
have *B i* ⊆ (*T*^^(*n*−*i*−*1*))−−‘ (*T*^^(*i*+*1*))−−‘ *A*
using *B-def return-partition-def* **by** *auto*
then show *B i* ⊆ (*T*^^*n*)−−‘*A*
using *T-vrestr-composed*(*1*)[*OF A-meas*, *of n*−*i*−*1*, *of i*+*1*] ‹*n*>*i*› **by** *auto*
qed

define *R* **where** *R* = (*T*^^*n*)−−‘*A* − (⋃*i* ∈ {..<*N*}. *B i*)
have [*measurable*]: *R* ∈ *sets M* **unfolding** *R-def* **by** *measurable*
have *dec-n*: (*T*^^*n*)−−‘*A* = *R* ∪ (⋃*i* ∈ {..<*N*}. *B i*) **using** *R-def incl-B* **by** *blast*
have *small-R*: *emeasure M R* < *e2*
proof −
have *R* ∩ (⋃*i* ∈ {..<*N*}. *B i*) = {} **using** *R-def* **by** *blast*
then have *emeasure M* ((*T*^^*n*)−−‘*A*) = *emeasure M R* + *emeasure M* (⋃*i* ∈ {..<*N*}. *B i*)
using *plus-emeasure*[*of R*, *of M*, *of* ⋃*i* ∈ {..<*N*}. *B i*] *dec-n* **by** *auto*
moreover have *emeasure M* (⋃*i* ∈ {..<*N*}. *B i*) = (∑*i* ∈ {..<*N*}. *emeasure M* (*B i*))
by (*intro disj-B sum-emeasure*[*symmetric*], *auto*)

ultimately have *emeasure M* $((T\widehat{\ }\widehat{\ }n){-}{-}`A)$ = *emeasure M R* + ($\sum i \in \{..<N\}$. *emeasure M* (*B i*))
by *simp*
moreover have *emeasure M* $((T\widehat{\ }\widehat{\ }n){-}{-}`A)$ = *emeasure M A*
using *T-vrestr-same-emeasure*(*2*)[*OF A-meas*] **by** *simp*
moreover have $\bigwedge i$. *emeasure M* (*B i*) = *emeasure M* (*return-partition A* (*i+1*))
using *T-vrestr-same-emeasure*(*2*) *B-def return-partition-basics*(*1*)[*OF A-meas*] **by** *simp*
ultimately have *a*: *emeasure M A* = *emeasure M R* + ($\sum i \in \{..<N\}$. *emeasure M* (*return-partition A* (*i+1*)))
by *simp*
moreover have *b*: ($\sum i \in \{..<N\}$. *emeasure M* (*return-partition A* (*i+1*))) $\neq \infty$ **using** *fin*
by (*simp add*: *a less-top*)
ultimately show *?thesis*
using *largeM fin b* **by** *simp*
qed

have *K* (*Suc k*) *n* $\subseteq$ *R* $\cup$ ($\bigcup i{<}N$. *K k* (*n−i−1*))
proof
fix *x* **assume** *a*: *x* $\in$ *K* (*Suc k*) *n*
show *x* $\in$ *R* $\cup$ ($\bigcup i{<}N$. *K k* (*n−i−1*))
proof (*cases*)
assume $\neg$(*x* $\in$ *R*)
have *x* $\in$ $(T\widehat{\ }\widehat{\ }n){-}{-}`A$ **using** *a K-def* **by** *simp*
then have *x*$\in$ ($\bigcup i \in \{..<N\}$. *B i*) **using** *dec-n* ‹$\neg$(*x* $\in$ *R*)› **by** *simp*
then obtain *i* **where** *i*$\in$$\{..<N\}$ *x* $\in$ *B i* **by** *auto*
then have *n*>*i* **using** ‹*n*>*N*› **by** *auto*
then have $(T\widehat{\ }\widehat{\ }(n{-}i{-}1))$ *x* $\in$ *return-partition A* (*i+1*) **using** *B-def* ‹*x* $\in$ *B i*› **by** *auto*
then have *i*: $(T\widehat{\ }\widehat{\ }(n{-}i{-}1))$ *x* $\in$ *A* **using** *return-partition-def* **by** *auto*
then have *indicator A* $((T\widehat{\ }\widehat{\ }(n{-}i{-}1))$ *x*) = (*1*::*nat*) **by** *auto*
then have *local-time A* (*n−i*) *x* = *local-time A* (*n−i−1*) *x* + *1*
by (*metis Suc-diff-Suc Suc-eq-plus1 diff-diff-add local-time-Suc*[*of A*, *of n−i−1*] ‹*n*>*i*›)
then have *local-time A* (*n−i*) *x* > *local-time A* (*n−i−1*) *x* **by** *simp*
moreover have *local-time A n x* $\geq$ *local-time A* (*n−i*) *x* **using** *local-time-incseq*
by (*metis* ‹*i* < *n*› *le-add-diff-inverse2 less-or-eq-imp-le local-time-cocycle le-iff-add*)
ultimately have *local-time A n x* > *local-time A* (*n−i−1*) *x* **by** *simp*
moreover have *local-time A n x* < *Suc k* **using** *a K-def* **by** *simp*
ultimately have *∗*: *local-time A* (*n−i−1*) *x* < *k* **by** *simp*

have *x* $\in$ *space M* **using** ‹*x* $\in$ $(T\widehat{\ }\widehat{\ }n){-}{-}`A$› **by** *auto*
then have *x* $\in$ $(T\widehat{\ }\widehat{\ }(n{-}i{-}1)){-}{-}`A$ **using** *i A-meas vimage-restr-def* **by** (*metis IntI sets.Int-space-eq2 vimageI*)
then have *x* $\in$ *K k* (*n−i−1*) **using** *∗ K-def* **by** *blast*

then show *?thesis* **using** ‹$i \in \{..<N\}$› **by** *auto*
qed (*simp*)
qed
then have $emeasure\ M\ (K\ (Suc\ k)\ n) \leq emeasure\ M\ (R \cup (\bigcup i<N.\ K\ k\ (n-i-1)))$
by (*intro emeasure-mono*, *auto*)
also have ... $\leq emeasure\ M\ R + emeasure\ M\ (\bigcup i<N.\ K\ k\ (n-i-1))$
by (*rule emeasure-subadditive*, *auto*)
also have ... $\leq emeasure\ M\ R + (\sum i<N.\ emeasure\ M\ (K\ k\ (n-i-1)))$
by (*metis add-left-mono image-subset-iff emeasure-subadditive-finite*[**where** $?A = \lambda i.\ K\ k\ (n-i-1)$ **and** $?I = \{..<N\}$, *OF finite-lessThan*[*of N*]] *K-meas*)
also have ... $\leq e2 + (\sum i<N.\ emeasure\ M\ (K\ k\ (n-i-1)))$
using *small-R* **by** (*auto intro*!: *add-right-mono*)
finally show $emeasure\ M\ (K\ (Suc\ k)\ n) \leq e2 + (\sum i<N.\ emeasure\ M\ (K\ k\ (n-i-1)))$ **.**
qed

have $(\lambda n.\ (\sum i\in\{..<N\}.\ emeasure\ M\ (K\ k\ (n-i-1)))) \longlongrightarrow (\sum i\in\{..<N\}.\ 0)$
apply (*intro tendsto-intros seq-offset-neg*) **using** *Suc.IH K-def* **by** *simp*
then have *eventually* $(\lambda n.\ (\sum i\in\{..<N\}.\ emeasure\ M\ (K\ k\ (n-i-1))) < e2)$ *sequentially*
using ‹$e2 > 0$› **by** (*simp add*: *order-tendsto-iff*)
then obtain *N2* **where** *N2bound*: $\bigwedge n.\ n > N2 \Longrightarrow (\sum i\in\{..<N\}.\ emeasure\ M\ (K\ k\ (n-i-1))) < e2$
by (*meson eventually-at-top-dense*)
define *N3* **where** $N3 = max\ N\ N2$
have $emeasure\ M\ (K\ (Suc\ k)\ n) < e$ **if** $n > N3$ **for** *n*
proof −
have $n>N2$ $n > N$ **using** *N3-def that* **by** *auto*
then have $emeasure\ M\ (K\ (Suc\ k)\ n) \leq ennreal\ e2 + (\sum i\in\{..<N\}.\ emeasure\ M\ (K\ k\ (n-i-1)))$
using *upper* **by** *simp*
also have ... $\leq ennreal\ e2 + ennreal\ e2$
using *N2bound*[*OF* ‹$n > N2$›] *less-imp-le* **by** *auto*
also have ... $< e$ **using** *e2-def* ‹$e > 0$›
by (*auto simp add*: *ennreal-plus*[*symmetric*] *simp del*: *ennreal-plus intro*!: *ennreal-lessI*)
ultimately show $emeasure\ M\ (K\ (Suc\ k)\ n) < e$ **using** *le-less-trans* **by** *blast*
qed
then show $\forall_F\ x$ *in sequentially*. $emeasure\ M\ \{xa \in (T\ \hat{}\hat{}\ x)\ --`\ A.\ local\text{-}time\ A\ x\ xa < Suc\ k\} < ennreal\ e$
unfolding *K-def* **by** (*auto simp*: *eventually-at-top-dense intro*!: *exI*[*of - N3*])
qed
qed

We deduce that local times to a set B also tend to infinity on $T^{-n}A$ if B is related to A, i.e., if points in A have some iterate in B. This is clearly a necessary condition for the lemmas to hold: otherwise, points of A that never visit B have a local time equal to B equal to 0, and so do all their preimages.

The lemmas are readily reduced to the previous one on the local time to A, since if one visits A then one visits B in finite time by assumption (uniformly bounded in the first lemma, uniformly bounded on a set of large measure in the second lemma).

```
lemma (in conservative-mpt) local-time-unbounded2:
  assumes A-meas [measurable]: A ∈ sets M
      and fin: emeasure M A < ∞
      and incl: A ⊆ (T^^i)−−‘B
  shows (λn. emeasure M {x ∈ (T^^n)−−‘A. local-time B n x < k}) ⟶ 0
proof −
  have emeasure M {x ∈ (T^^n)−−‘A. local-time B n x < k} ≤ emeasure M {x
∈ (T^^n)−−‘A. local-time A n x < k + i}
    if n > i for n
  proof −
    have local-time A n x ≤ local-time B n x + i for x
    proof −
      have local-time B n x ≥ local-time A (n−i) x
      proof −
        define KA where KA = {t ∈ {0..<n−i}. (T^^t) x ∈ A}
        define KB where KB = {t ∈ {0..<n}. (T^^t) x ∈ B}
        then have KB ⊆ {0..<n} by auto
        then have finite KB using finite-lessThan[of n] finite-subset by auto
        let ?g = λt. t + i
        have ⋀t. t ∈ KA ⟹ ?g t ∈ KB
        proof −
          fix t assume t ∈ KA
          then have (T^^t) x ∈ A using KA-def by simp
          then have (T^^i) ((T^^t) x) ∈ B using incl by auto
          then have (T^^(t+i)) x ∈ B by (simp add: funpow-add add.commute)
          moreover have t+i < n using ‹t ∈ KA› KA-def ‹n > i› by auto
          ultimately show ?g t ∈ KB unfolding KB-def by simp
        qed
        then have ?g‘KA ⊆ KB by auto
        moreover have inj-on ?g KA by simp
        ultimately have card KB ≥ card KA
          using card-inj-on-le[where ?f = ?g and ?A = KA and ?B = KB] ‹finite
KB› by simp
        then show ?thesis using KA-def KB-def local-time-def by simp
      qed
      moreover have i ≥ local-time A i ((T^^(n−i))x) using local-time-bound by
auto
      ultimately show local-time B n x + i ≥ local-time A n x
        using local-time-cocycle[where ?n = n−i and ?m = i and ?x = x and
?A = A] ‹n>i› by auto
    qed
    then have local-time B n x < k ⟹ local-time A n x < k + i for x
```

by (*meson add-le-cancel-right le-trans not-less*)
then show *?thesis*
by (*intro emeasure-mono*, *auto*)
qed
then have *eventually* (λ*n*. *emeasure M* {*x* ∈ (*T*^^*n*)−−‘*A*. *local-time B n x* < *k*}
≤ *emeasure M* {*x* ∈ (*T*^^*n*)−−‘*A*. *local-time A n x* < *k* + *i*}) *sequentially*
using *eventually-at-top-dense* **by** *blast*
from *tendsto-sandwich*[*OF - this tendsto-const local-time-unbounded1* [*OF A-meas fin*, *of k*+*i*]]
show *?thesis* **by** *auto*
qed

lemma (**in** *conservative-mpt*) *local-time-unbounded3*:
assumes *A-meas*[*measurable*]: *A* ∈ *sets M*
and *B-meas*[*measurable*]: *B* ∈ *sets M*
and *fin*: *emeasure M A* < ∞
and *incl*: *A* − (⋃*i*. (*T*^^*i*)−−‘*B*) ∈ *null-sets M*
shows (λ*n*. *emeasure M* {*x* ∈ (*T*^^*n*)−−‘*A*. *local-time B n x* < *k*}) ⟶ *0*
proof −
define *R* **where** *R* = *A* − (⋃*i*. (*T*^^*i*)−−‘*B*)
have *R-meas*[*measurable*]: *R* ∈ *sets M*
by (*simp add*: *A-meas B-meas T-vrestr-meas*(*2*)[*OF B-meas*] *R-def countable-Un-Int*(*1*) *sets.Diff*)
have *emeasure M R* = *0* **using** *incl R-def* **by** *auto*
define *A2* **where** *A2* = *A* − *R*
have *A2-meas* [*measurable*]: *A2* ∈ *sets M* **unfolding** *A2-def* **by** *auto*
have *meq*: *emeasure M A2* = *emeasure M A* **using** ‹*emeasure M R* = *0*›
unfolding *A2-def* **by** (*subst emeasure-Diff*) (*auto simp*: *R-def*)
then have *A2-fin*: *emeasure M A2* < ∞ **using** *fin* **by** *auto*
define *K* **where** *K* = (λ*N*. *A2* ∩ (⋃*i*<*N*. (*T*^^*i*)−−‘*B*))
have *K-meas* [*measurable*]: *K N* ∈ *sets M* **for** *N* **unfolding** *K-def* **by** *auto*
have *K-incl*: ⋀*N*. *K N* ⊆ *A* **using** *K-def A2-def* **by** *blast*
have (⋃*N*. *K N*) = *A2* **using** *A2-def R-def K-def* **by** *blast*
moreover have *incseq K* **unfolding** *K-def incseq-def* **by** *fastforce*
ultimately have (λ*N*. *emeasure M* (*K N*)) ⟶ *emeasure M A2* **by** (*auto intro*: *Lim-emeasure-incseq*)
then have *conv*: (λ*N*. *emeasure M* (*K N*)) ⟶ *emeasure M A* **using** *meq* **by** *simp*

define *Bad* **where** *Bad* = (λ*U n*. {*x* ∈ (*T*^^*n*)−−‘*U*. *local-time B n x* < *k*})
define *Bad0* **where** *Bad0* = (λ*n*. {*x* ∈ *space M*. *local-time B n x* < *k*})
have *Bad0-meas* [*measurable*]: *Bad0 n* ∈ *sets M* **for** *n* **unfolding** *Bad0-def* **by** *auto*
have *Bad-inter*: ⋀*U n*. *Bad U n* = (*T*^^*n*)−−‘*U* ∩ *Bad0 n* **unfolding** *Bad-def Bad0-def* **by** *auto*
have *Bad-meas* [*measurable*]: ⋀*U n*. *U* ∈ *sets M* ⟹ *Bad U n* ∈ *sets M* **unfolding** *Bad-def* **by** *auto*

show *?thesis*
proof (*rule tendsto-zero-ennreal*)
fix *e*::*real*
assume *e* > *0*
define *e2* **where** *e2* = *e*/*3*
then have *e2* > *0* **using** ‹*e*>*0*› **by** *simp*
then have *ennreal e2* > *0* **by** *simp*
have (λ*N*. *emeasure M* (*K N*) + *e2*) ⟶ *emeasure M A* + *e2*
using *conv* **by** (*intro tendsto-add*) *auto*
moreover have *emeasure M A* < *emeasure M A* + *e2* **using** *fin* ‹*e2* > *0*› **by** *simp*
ultimately have *eventually* (λ*N*. *emeasure M* (*K N*) + *e2* > *emeasure M A*) *sequentially*
by (*simp add*: *order-tendsto-iff*)
then obtain *N* **where** *N*>*0* **and** *largeK*: *emeasure M* (*K N*) + *e2* > *emeasure M A*
by (*metis* (*no-types*, *lifting*) *add.commute add-Suc-right eventually-at-top-linorder le-add2 zero-less-Suc*)
define *S* **where** *S* = *A* − (*K N*)
have *S-meas* [*measurable*]: *S* ∈ *sets M* **using** *A-meas K-meas S-def* **by** *simp*
have *emeasure M A* = *emeasure M* (*K N*) + *emeasure M S*
by (*metis Diff-disjoint Diff-partition plus-emeasure*[*OF K-meas*[*of N*], *OF S-meas*] *S-def K-incl*[*of N*])
then have *S-small*: *emeasure M S* < *e2* **using** *largeK fin* **by** *simp*
have *A-incl*: *A* ⊆ *S* ∪ (⋃ *i*<*N*. *A2* ∩ (*T*^^*i*)−−‘*B*) **using** *S-def K-def* **by** *auto*

define *L* **where** *L* = (λ*i*. *A2* ∩ (*T*^^*i*)−−‘*B*)
have *L-meas* [*measurable*]: *L i* ∈ *sets M* **for** *i* **unfolding** *L-def* **by** *auto*
have ⋀*i*. *L i* ⊆ *A2* **using** *L-def* **by** *simp*
then have *L-fin*: *emeasure M* (*L i*) < ∞ **for** *i*
using *emeasure-mono*[*of L i A2 M*] *A2-meas A2-fin* **by** *simp*
have ⋀*i*. *L i* ⊆ (*T*^^*i*)−−‘*B* **using** *L-def* **by** *auto*
then have *a*: ⋀*i*. (λ*n*. *emeasure M* (*Bad* (*L i*) *n*)) ⟶ *0* **unfolding** *Bad-def*
using *local-time-unbounded2*[*OF L-meas*, *OF L-fin*] **by** *blast*
have (λ*n*. (∑ *i*<*N*. *emeasure M* (*Bad* (*L i*) *n*))) ⟶ *0* **using** *tendsto-sum*[*OF a*] **by** *auto*
then have *eventually* (λ*n*. (∑ *i*<*N*. *emeasure M* (*Bad* (*L i*) *n*)) < *e2*) *sequentially*
using ‹*ennreal e2* > *0*› *order-tendsto-iff* **by** *metis*
then obtain *N2* **where** ∗: ⋀*n*. *n* > *N2* ⟹ (∑ *i*<*N*. *emeasure M* (*Bad* (*L i*) *n*)) < *e2*
by (*auto simp add*: *eventually-at-top-dense*)

have *emeasure M* (*Bad A n*) < *e* **if** *n* > *N2* **for** *n*
proof −
have *emeasure M* (*Bad S n*) ≤ *emeasure M* ((*T*^^*n*)−−‘*S*)
apply (*rule emeasure-mono*) **unfolding** *Bad-def* **by** *auto*
also have ... = *emeasure M S* **using** *T-vrestr-same-emeasure*(*2*) **by** *simp*

also have ... ≤ *e2* **using** *S-small* **by** *simp*
finally have *SBad-small*: *emeasure M (Bad S n) ≤ e2* **by** *simp*

have $(T^{\frown\frown}n)--`A \subseteq (T^{\frown\frown}n)--`S \cup (\bigcup i<N. (T^{\frown\frown}n)--`(L\ i))$
using *A-incl* **unfolding** *L-def* **by** *fastforce*
then have *I*: *Bad A n ⊆ Bad S n ∪ (⋃ i<N. Bad (L i) n)* **using** *Bad-inter* **by** *force*
have *emeasure M (Bad A n) ≤ emeasure M (Bad S n ∪ (⋃ i<N. Bad (L i) n))*
by (*rule emeasure-mono*[*OF I*], *measurable*)
also have ... ≤ *emeasure M (Bad S n) + emeasure M (⋃ i<N. Bad (L i) n)*
by (*intro emeasure-subadditive countable-Un-Int*(*1*), *auto*)
also have ... ≤ *emeasure M (Bad S n) + (∑ i<N. emeasure M (Bad (L i) n))*
by (*simp add*: *add-left-mono image-subset-iff Bad-meas*[*OF L-meas*] *emeasure-subadditive-finite*[*OF finite-lessThan*[*of N*], **where** *?A = λi. Bad (L i) n*])
also have ... ≤ *ennreal e2 + ennreal e2*
using *SBad-small less-imp-le*[*OF* ∗[*OF* ‹*n > N2*›]] **by** (*rule add-mono*)
also have ... < *e* **using** *e2-def* ‹*e>0*› **by** (*simp del*: *ennreal-plus add*: *ennreal-plus*[*symmetric*] *ennreal-lessI*)
finally show *emeasure M (Bad A n) < e* **by** *simp*
qed
then show $\forall_F$ *x in sequentially. emeasure M {xa ∈ (T* ^^ *x) −−' A. local-time B x xa < k} < e*
unfolding *eventually-at-top-dense Bad-def* **by** *auto*
qed
qed

5.4 The induced map

The map induced by T on a set A is obtained by iterating T until one lands again in A. (Outside of A, we take the identity for definiteness.) It has very nice properties: if T is conservative, then the induced map T_A also is. If T is measure preserving, then so is T_A. (In particular, even if T preserves an infinite measure, T_A is a probability preserving map if A has measure 1 – this makes it possible to prove some statements in infinite measure by using results in finite measure systems). If T is invertible, then so is T_A. We prove all these properties in this paragraph.

definition *induced-map*::*'a set ⇒ ('a ⇒ 'a)*
where *induced-map A = (λ x. (T^^(return-time-function A x)) x)*

The set A is stabilized by the induced map.

lemma *induced-map-stabilizes-A*:
x ∈ A ⟷ induced-map A x ∈ A
proof
assume *x ∈ A*
show *induced-map A x ∈ A*

proof (*cases* $x \in$ *recurrent-subset* A)
case *False*
then have *induced-map* $A\ x = x$ **using** *induced-map-def return-time-function-def* **by** *simp*
then show *?thesis* **using** ‹$x \in A$› **by** *simp*
next
case *True*
define K **where** $K = \{n \in \{1..\}.\ (T\widehat{\ }\widehat{\ }n)\ x \in A\}$
have $K \neq \{\}$ **using** *True recurrent-subset-def K-def*
by *blast*
moreover have *return-time-function* $A\ x =$ *Inf* K
using *return-time-function-def K-def True* **by** *simp*
ultimately have *return-time-function* $A\ x \in K$ **using** *Inf-nat-def1* **by** *simp*
then show *?thesis*
unfolding *induced-map-def K-def* **by** *blast*
qed
next
have *induced-map* $A\ x = x$ **if** $x \notin A$
using *that*
by (*auto simp*: *induced-map-def return-time-function-def recurrent-subset-def*)
then show *induced-map* $A\ x \in A \Longrightarrow x \in A$
by *fastforce*
qed

lemma *induced-map-iterates-stabilize-A*:
assumes $x \in A$
shows ((*induced-map* A)$\widehat{\ }\widehat{\ }n$) $x \in A$
proof (*induction* n)
case *0*
show *?case* **using** ‹$x \in A$› **by** *auto*
next
case (*Suc* n)
have ((*induced-map* A)$\widehat{\ }\widehat{\ }$(*Suc* n)) x = (*induced-map* A) (((*induced-map* A)$\widehat{\ }\widehat{\ }n$) x) **by** *auto*
then show *?case* **using** *Suc.IH induced-map-stabilizes-A* **by** *auto*
qed

lemma *induced-map-meas* [*measurable*]:
assumes [*measurable*]: $A \in$ *sets* M
shows *induced-map* $A \in$ *measurable* $M\ M$
unfolding *induced-map-def* **by** *auto*

The iterates of the induced map are given by a power of the original map, where the power is the Birkhoff sum (for the induced map) of the first return time. This is obvious, but useful.

lemma *induced-map-iterates*:
((*induced-map* A)$\widehat{\ }\widehat{\ }n$) x = ($T\widehat{\ }\widehat{\ }$($\sum i < n.$ *return-time-function* A ((*induced-map* $A\ \widehat{\ }\widehat{\ }i$) x))) x
proof (*induction* n)

case *0*
show *?case* **by** *auto*
next
case (*Suc n*)
have ((*induced-map A*)^^(*n*+*1*)) *x* = *induced-map A* (((*induced-map A*)^^*n*) *x*)
by (*simp add*: *funpow-add*)
also have ... = (*T*^^(*return-time-function A* (((*induced-map A*)^^*n*) *x*))) (((*induced-map A*)^^*n*) *x*)
using *induced-map-def* **by** *auto*
also have ... = (*T*^^(*return-time-function A* (((*induced-map A*)^^*n*) *x*))) ((*T*^^(∑ *i* < *n*. *return-time-function A* ((*induced-map A* ^^*i*) *x*))) *x*)
using *Suc.IH* **by** *auto*
also have ... = (*T*^^(*return-time-function A* (((*induced-map A*)^^*n*) *x*) + (∑ *i* < *n*. *return-time-function A* ((*induced-map A* ^^*i*) *x*)))) *x*
by (*simp add*: *funpow-add*)
also have ... = (*T*^^(∑ *i* < *Suc n*. *return-time-function A* ((*induced-map A* ^^*i*) *x*))) *x* **by** (*simp add*: *add.commute*)
finally show *?case* **by** *simp*
qed

lemma *induced-map-stabilizes-recurrent-infty*:
assumes *x* ∈ *recurrent-subset-infty A*
shows ((*induced-map A*)^^*n*) *x* ∈ *recurrent-subset-infty A*
proof −
have *x* ∈ *A* **using** *assms*(*1*) *recurrent-subset-incl*(*2*) **by** *auto*

define *R* **where** *R* = (∑ *i* < *n*. *return-time-function A* ((*induced-map A* ^^*i*) *x*))
have ∗: ((*induced-map A*)^^*n*) *x* = (*T*^^*R*) *x* **unfolding** *R-def* **by** (*rule induced-map-iterates*)
moreover have ((*induced-map A*)^^*n*) *x* ∈ *A*
by (*rule induced-map-iterates-stabilize-A*, *simp add*: ‹*x* ∈ *A*›)
ultimately have (*T*^^*R*) *x* ∈ *A* **by** *simp*
then show *?thesis* **using** *recurrent-subset-infty-returns*[*OF assms*] ∗ **by** *auto*
qed

If $x \in A$, then its successive returns to A are exactly given by the iterations of the induced map.

lemma *induced-map-returns*:
assumes *x* ∈ *A*
shows ((*T*^^*n*) *x* ∈ *A*) ⟷ (∃ *N*≤*n*. *n* = (∑ *i*<*N*. *return-time-function A* ((*induced-map A* ^^ *i*) *x*)))
proof
assume (*T*^^*n*) *x* ∈ *A*
have ⋀*y*. *y* ∈ *A* ⟹ (*T*^^*n*)*y* ∈ *A* ⟹ ∃ *N*≤*n*. *n* = (∑ *i*<*N*. *return-time-function A* (((*induced-map A*)^^*i*) *y*)) **for** *n*
proof (*induction n rule*: *nat-less-induct*)
case (*1 n*)
show ∃ *N*≤*n*. *n* = (∑ *i*<*N*. *return-time-function A* (((*induced-map A*)^^*i*) *y*))

proof (*cases*)
assume $n = 0$
then show *?thesis* **by** *auto*
next
assume $\neg(n = 0)$
then have $n > 0$ **by** *simp*
then have *y-rec*: $y \in$ *recurrent-subset A* **using** ‹$y \in A$› ‹$(T\widehat{\ }\widehat{\ }n)\ y \in A$› *recurrent-subset-def* **by** *auto*
then have $*$: *return-time-function A y* > 0 **by** (*metis DiffE insert-iff neq0-conv vimage-eq return-time0*)
define *m* **where** $m =$ *return-time-function A y*
have $m > 0$ **using** $*$ *m-def* **by** *simp*
define *K* **where** $K = \{t \in \{1..\}.\ (T\ \widehat{\ }\widehat{\ }\ t)\ y \in A\}$
have $n \in K$ **unfolding** *K-def* **using** ‹$n > 0$› ‹$(T\widehat{\ }\widehat{\ }n)y \in A$› **by** *simp*
then have $n \geq$ *Inf K* **by** (*simp add*: *cInf-lower*)
moreover have $m =$ *Inf K* **unfolding** *m-def K-def return-time-function-def* **using** *y-rec* **by** *simp*
ultimately have $n \geq m$ **by** *simp*
define *z* **where** $z =$ *induced-map A y*
have $z \in A$ **using** ‹$y \in A$› *induced-map-stabilizes-A z-def* **by** *simp*
have $z = (T\widehat{\ }\widehat{\ }m)\ y$ **using** *induced-map-def y-rec z-def m-def* **by** *auto*
then have $(T\widehat{\ }\widehat{\ }(n-m))\ z = (T\widehat{\ }\widehat{\ }n)\ y$ **using** ‹$n \geq m$› *funpow-add*[*of n−m m T*, *symmetric*]
by (*metis comp-apply le-add-diff-inverse2*)
then have $(T\widehat{\ }\widehat{\ }(n-m))\ z \in A$ **using** ‹$(T\widehat{\ }\widehat{\ }n)\ y \in A$› **by** *simp*
moreover have $n-m < n$ **using** ‹$m > 0$› ‹$n > 0$› **by** *simp*
ultimately obtain *N0* **where** $N0 \leq n-m$ $n-m = (\sum i<N0.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *z*))
using ‹$z \in A$› *1.IH* **by** *blast*
then have $n-m = (\sum i<N0.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) (*induced-map A y*)))
using *z-def* **by** *auto*
moreover have $\bigwedge i.$ ((*induced-map A*)$\widehat{\ }\widehat{\ }i$) (*induced-map A y*) = ((*induced-map A*)$\widehat{\ }\widehat{\ }(i+1)$) *y*
by (*metis Suc-eq-plus1 comp-apply funpow-Suc-right*)
ultimately have $n-m = (\sum i<N0.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }(i+1)$) *y*))
by *simp*
then have $n-m = (\sum i \in \{1..<N0+1\}.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *y*))
using *sum.shift-bounds-nat-ivl*[*of* $\lambda i.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *y*), *of 0*, *of 1*, *of N0*, *symmetric*]
atLeast0LessThan **by** *auto*
moreover have $m = (\sum i\in\{0..<1\}.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *y*)) **using** *m-def* **by** *simp*
ultimately have $n = (\sum i\in\{0..<1\}.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *y*))
$+ (\sum i \in \{1..<N0+1\}.$ *return-time-function A* (((*induced-map A*)$\widehat{\ }\widehat{\ }i$) *y*)) **using** ‹$n \geq m$› **by** *simp*

then have *n* = (∑ *i*∈{*0..<N0+1*}. *return-time-function A* (((*induced-map A*)^^*i*) *y*))
using *le-add2 sum.atLeastLessThan-concat* **by** *blast*
moreover have *N0 + 1* ≤ *n* **using** ‹*N0* ≤ *n−m*› ‹*n* − *m* < *n*› **by** *linarith*
ultimately show *?thesis* **by** (*metis atLeast0LessThan*)
qed
qed
then show ∃ *N*≤*n*. *n* = (∑ *i*<*N*. *return-time-function A* ((*induced-map A* ^^ *i*) *x*))
using ‹*x* ∈ *A*› ‹(*T*^^*n*) *x* ∈ *A*› **by** *simp*
next
assume ∃ *N*≤*n*. *n* = (∑ *i*<*N*. *return-time-function A* ((*induced-map A* ^^ *i*) *x*))
then obtain *N* **where** *n* = (∑ *i*<*N*. *return-time-function A* ((*induced-map A* ^^ *i*) *x*)) **by** *blast*
then have (*T*^^*n*) *x* = ((*induced-map A*)^^*N*) *x* **using** *induced-map-iterates*[*of N*, *of A*, *of x*] **by** *simp*
then show (*T*^^*n*) *x* ∈ *A* **using** ‹*x* ∈ *A*› *induced-map-iterates-stabilize-A* **by** *auto*
qed

If a map is conservative, then the induced map is still conservative. Note that this statement is not true if one replaces the word "conservative" with "qmpt": inducion only works well in conservative settings.

For instance, the right translation on $\mathbb{Z}$ is qmpt, but the induced map on $\mathbb{N}$ (again the right translation) is not, since the measure of $\{0\}$ is nonzero, while its preimage, the empty set, has zero measure.

To prove conservativity, given a subset B of A, there exists some time n such that $T^{-n}B \cap B$ has positive measure. But this time n corresponds to some returns to A for the induced map, so $T^{-n}B \cap B$ is included in $\bigcup_m T_A^{-m}B \cap B$, hence one of these sets must have positive measure.

The fact that the map is qmpt is then deduced from the conservativity.

proposition (**in** *conservative*) *induced-map-conservative*:
assumes *A-meas*: *A* ∈ *sets M*
shows *conservative* (*restrict-space M A*) (*induced-map A*)
proof
have *sigma-finite-measure M* **by** *unfold-locales*
then have *sigma-finite-measure* (*restrict-space M A*)
using *sigma-finite-measure-restrict-space assms* **by** *auto*
then show ∃ *Aa*. *countable Aa* ∧ *Aa* ⊆ *sets* (*restrict-space M A*) ∧ ⋃ *Aa* = *space* (*restrict-space M A*)
∧ (∀ *a*∈*Aa*. *emeasure* (*restrict-space M A*) *a* ≠ ∞) **using** *sigma-finite-measure-def* **by** *auto*

have *imp*: ⋀*B*. (*B* ∈ *sets M* ∧ *B* ⊆ *A* ∧ *emeasure M B* > *0*) ⟹ (∃ *N*>*0*. *emeasure M* (((*induced-map A*)^^*N*)−‘*B* ∩ *B*) > *0*)
proof −
fix *B*

assume *assm*: $B \in$ *sets* $M \wedge B \subseteq A \wedge$ *emeasure* $M\ B > 0$
then have $B \subseteq A$ **by** *simp*
have *inc*: $(\bigcup n\in\{1..\}.\ (T\widehat{\ }\widehat{\ }n)-`B \cap B) \subseteq (\bigcup N\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }N)-`B \cap B)$
proof
fix x **assume** $x \in (\bigcup n\in\{1..\}.\ (T\widehat{\ }\widehat{\ }n)-`B \cap B)$
then obtain n **where** $n\in\{1..\}$ **and** $*$: $x \in (T\widehat{\ }\widehat{\ }n)-`B \cap B$ **by** *auto*
then have $n > 0$ **by** *auto*
have $x \in A\ (T\widehat{\ }\widehat{\ }n)\ x \in A$ **using** $*$ ‹$B \subseteq A$› **by** *auto*
then obtain N **where** $**$: $n = (\sum i<N.\ \textit{return-time-function}\ A\ ((\textit{induced-map}\ A\ \widehat{\ }\widehat{\ }\ i)\ x))$
using *induced-map-returns* **by** *auto*
then have $((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }N)\ x = (T\widehat{\ }\widehat{\ }n)\ x$ **using** *induced-map-iterates*[*of N*, *of A*, *of x*] **by** *simp*
then have $((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }N)\ x \in B$ **using** $*$ **by** *simp*
then have $x \in ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }N)-`B \cap B$ **using** $*$ **by** *simp*
moreover have $N > 0$ **using** $**$ ‹$n > 0$›
by (*metis leD lessThan-iff less-nat-zero-code neq0-conv sum.neutral-const sum-mono*)
ultimately show $x \in (\bigcup N\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }N)-`B \cap B)$ **by** *auto*
qed
have *B-meas* [*measurable*]: $B \in$ *sets* M **and** *B-pos*: *emeasure* $M\ B > 0$ **using** *assm* **by** *auto*
obtain n **where** $n > 0$ **and** *pos*: *emeasure* $M\ ((T\widehat{\ }\widehat{\ }n)-`B \cap B) > 0$
using *conservative*[*OF B-meas*, *OF B-pos*] **by** *auto*
then have $n \in \{1..\}$ **by** *auto*

have *itB-meas*: $\bigwedge i.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B \in$ *sets* M
using *B-meas measurable-compose-n*[*OF induced-map-meas*[*OF A-meas*]] **by** (*metis Int-assoc measurable-sets sets.Int sets.Int-space-eq1*)
then have $(\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) \in$ *sets* M **by** *measurable*
moreover have $(T\widehat{\ }\widehat{\ }n)-`B \cap B \subseteq (\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B)$ **using** *inc* ‹$n \in \{1..\}$› **by** *force*
ultimately have *emeasure* $M\ (\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) > 0$
by (*metis* (*no-types*, *lifting*) *emeasure-eq-0 zero-less-iff-neq-zero pos*)
then have *emeasure* $M\ (\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) \neq 0$ **by** *simp*
have $\exists i\in\{1..\}.$ *emeasure* $M\ (((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) \neq 0$
proof (*rule ccontr*)
assume $\neg(\exists i\in\{1..\}.$ *emeasure* $M\ (((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) \neq 0)$
then have *a*: $\bigwedge i.\ i \in \{1..\} \Longrightarrow ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B \in$ *null-sets* M
using *itB-meas* **by** *auto*
have $(\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) \in$ *null-sets* M
by (*rule null-sets-UN'*, *simp-all add*: *a*)
then show *False* **using** ‹*emeasure* $M\ (\bigcup i\in\{1..\}.\ ((\textit{induced-map}\ A)\widehat{\ }\widehat{\ }i)-`B \cap B) > 0$› **by** *auto*
qed

then show $\exists N>0.$ *emeasure M* $(((induced\text{-}map\ A)^{\frown\frown}N)-`\ B \cap B) > 0$
by (*simp add*: *Bex-def less-eq-Suc-le zero-less-iff-neq-zero*)
qed

define *K* **where** $K = \{B.\ B \in sets\ M \wedge B \subseteq A\}$
have *K-stable*: $(induced\text{-}map\ A)-`B \in K$ **if** $B \in K$ **for** *B*
proof −
have *B-meas*: $B \in sets\ M$ **and** $B \subseteq A$ **using** *that* **unfolding** *K-def* **by** *auto*
then have *a*: $(induced\text{-}map\ A)-`B \subseteq A$ **using** *induced-map-stabilizes-A* **by** *auto*
then have $(induced\text{-}map\ A)-`B = (induced\text{-}map\ A)-`B \cap space\ M$ **using** *assms sets.sets-into-space* **by** *auto*
then have $(induced\text{-}map\ A)-`B \in sets\ M$ **using** *induced-map-meas*[*OF assms*] *B-meas* **by** (*metis vrestr-meas vrestr-of-set*)
then show $(induced\text{-}map\ A)-`B \in K$ **unfolding** *K-def* **using** *a* **by** *auto*
qed

define *K0* **where** $K0 = K \cap (null\text{-}sets\ M)$
have *K0-stable*: $(induced\text{-}map\ A)-`B \in K0$ **if** $B \in K0$ **for** *B*
proof −
have $B \in K$ **using** *that* **unfolding** *K0-def* **by** *simp*
then have *a*: $(induced\text{-}map\ A)-`B \subseteq A$ **and** *b*: $(induced\text{-}map\ A)-`B \in sets\ M$
using *K-stable* **unfolding** *K-def* **by** *auto*
have *B-meas* [*measurable*]: $B \in sets\ M$ **using** ‹$B \in K$› **unfolding** *K-def* **by** *simp*
have *B0*: $B \in null\text{-}sets\ M$ **using** ‹$B \in K0$› **unfolding** *K0-def* **by** *simp*

have $(induced\text{-}map\ A)-`B \subseteq (\bigcup n.\ (T^{\frown\frown}n)-`B)$ **unfolding** *induced-map-def* **by** *auto*
then have $(induced\text{-}map\ A)-`B \subseteq (\bigcup n.\ (T^{\frown\frown}n)-`B \cap space\ M)$
using *b sets.sets-into-space* **by** *simp blast*
then have *inc*: $(induced\text{-}map\ A)-`B \subseteq (\bigcup n.\ (T^{\frown\frown}n)--`B)$ **unfolding** *vimage-restr-def*
using *sets.sets-into-space* [*OF B-meas*] **by** *simp*

have $(T^{\frown\frown}n)--`B \in null\text{-}sets\ M$ **for** *n* **using** *B0 T-quasi-preserves-null*(*2*)[*OF B-meas*] **by** *simp*
then have $(\bigcup n.\ (T^{\frown\frown}n)--`B) \in null\text{-}sets\ M$ **using** *null-sets-UN* **by** *auto*
then have $(induced\text{-}map\ A)-`B \in null\text{-}sets\ M$ **using** *null-sets-subset*[*OF - b inc*] **by** *auto*
then show $(induced\text{-}map\ A)-`B \in K0$ **unfolding** *K0-def K-def* **by** (*simp add*: *a b*)
qed

have ∗: $D \in null\text{-}sets\ M \longleftrightarrow D \in null\text{-}sets\ (restrict\text{-}space\ M\ A)$ **if** $D \in K$ **for** *D*
using *that* **unfolding** *K-def* **apply** *auto*
apply (*metis assms emeasure-restrict-space null-setsD1 null-setsI sets.Int-space-eq2 sets-restrict-space-iff*)
by (*metis assms emeasure-restrict-space null-setsD1 null-setsI sets.Int-space-eq2*)

show *induced-map A* ∈ *quasi-measure-preserving* (*restrict-space M A*) (*restrict-space M A*)
  unfolding *quasi-measure-preserving-def*
 proof (*auto*)
  have *induced-map A* ∈ *A* → *A* **using** *induced-map-stabilizes-A* **by** *auto*
  then show *a*: *induced-map A* ∈ *measurable* (*restrict-space M A*) (*restrict-space M A*)
   using *measurable-restrict-space3*[**where** *?A* = *A* **and** *?B* = *A* **and** *?M* = *M* **and** *?N* = *M*] *induced-map-meas*[*OF A-meas*] **by** *auto*

  fix *B* **assume** *H*: *B* ∈ *sets* (*restrict-space M A*)
   induced-map A −‘*B* ∩ *space* (*restrict-space M A*) ∈ *null-sets* (*restrict-space M A*)
 then have *B* ∈ *K* **unfolding** *K-def* **by** (*metis assms mem-Collect-eq sets.Int-space-eq2 sets-restrict-space-iff*)
  then have *B-meas* [*measurable*]: *B* ∈ *sets M* **and** *B-incl*: *B* ⊆ *A* **unfolding** *K-def* **by** *auto*
  have *induced-map A* −‘*B* ∈ *K* **using** *K-stable* ‹*B* ∈ *K*› **by** *auto*
  then have *B2-meas*: *induced-map A* −‘*B* ∈ *sets M* **and** *B2-incl*: *induced-map A* −‘*B* ⊆ *A*
   unfolding *K-def* **by** *auto*
  have *induced-map A* −‘ *B* = *induced-map A* −‘*B* ∩ *space* (*restrict-space M A*)
   using *B2-incl* **by** (*simp add*: *Int-absorb2 assms space-restrict-space*)
  then have *induced-map A* −‘ *B* ∈ *null-sets* (*restrict-space M A*) **using** *H*(*2*) **by** *simp*
  then have *induced-map A* −‘ *B* ∈ *K0* **unfolding** *K0-def* **using** ‹*induced-map A* −‘*B* ∈ *K*› ∗ **by** *auto*
  {
   fix *n*
   have ∗: ((*induced-map A*)^^(*n*+*1*))−‘*B* ∈ *K0*
   proof (*induction n*)
    case (*Suc n*)
   have ((*induced-map A*)^^(*Suc n*+*1*))−‘*B* = (*induced-map A*)−‘(((*induced-map A*)^^(*n*+*1*))−‘ *B*)
     by (*metis Suc-eq-plus1 funpow-Suc-right vimage-comp*)
    then show *?case* **by** (*metis Suc.IH K0-stable*)
   qed (*auto simp add*: ‹*induced-map A* −‘ *B* ∈ *K0*›)
   have ∗∗: ((*induced-map A*)^^(*n*+*1*))−‘ *B* ∈ *sets M* **using** ∗ *K0-def K-def* **by** *auto*
   have ((*induced-map A*)^^(*n*+*1*))−‘ *B* ∩ *B* ∈ *null-sets M*
    apply (*rule null-sets-subset*[*of* ((*induced-map A*)^^(*n*+*1*))−‘*B*])
    using ∗ **unfolding** *K0-def* **apply** *simp*
    using ∗∗ **by** *auto*
  }
  then have ((*induced-map A*)^^*n*)−‘ *B* ∩ *B* ∈ *null-sets M* **if** *n*>*0* **for** *n*
   using *that* **by** (*metis Suc-eq-plus1 neq0-conv not0-implies-Suc*)
  then have *B* ∈ *null-sets M* **using** *imp B-incl B-meas zero-less-iff-neq-zero inf.strict-order-iff*

by (*metis null-setsD1 null-setsI*)
then show *B* ∈ *null-sets* (*restrict-space M A*) **using** ∗ ‹*B* ∈ *K*› **by** *auto*
next
fix *B* **assume** *H*: *B* ∈ *sets* (*restrict-space M A*)
B ∈ *null-sets* (*restrict-space M A*)
then have *B* ∈ *K* **unfolding** *K-def* **by** (*metis assms mem-Collect-eq sets.Int-space-eq2 sets-restrict-space-iff*)
then have *B-meas* [*measurable*]: *B* ∈ *sets M* **and** *B-incl*: *B* ⊆ *A* **unfolding** *K-def* **by** *auto*
have *B* ∈ *null-sets M* **using** ∗ *H*(*2*) ‹*B* ∈ *K*› **by** *simp*
then have *B* ∈ *K0* **unfolding** *K0-def* **using** ‹*B* ∈ *K*› **by** *simp*
then have *inK*: (*induced-map A*)−‘*B* ∈ *K0* **using** *K0-stable* **by** *auto*
then have *inA*: (*induced-map A*)−‘*B* ⊆ *A* **unfolding** *K0-def K-def* **by** *auto*
then have (*induced-map A*)−‘*B* = (*induced-map A*)−‘*B* ∩ *space* (*restrict-space M A*)
by (*simp add*: *Int-absorb2 assms space-restrict-space2*)
then show *induced-map A* −‘ *B* ∩ *space* (*restrict-space M A*) ∈ *null-sets* (*restrict-space M A*)
using ∗ *inK* **unfolding** *K0-def* **by** *auto*
qed

fix *B*
assume *B-measA*: *B* ∈ *sets* (*restrict-space M A*) **and** *B-posA*: *0* < *emeasure* (*restrict-space M A*) *B*
then have *B-meas*: *B* ∈ *sets M* **by** (*metis assms sets.Int-space-eq2 sets-restrict-space-iff*)
have *B-incl*: *B* ⊆ *A* **by** (*metis B-measA assms sets.Int-space-eq2 sets-restrict-space-iff*)
then have *B-pos*: *0* < *emeasure M B* **using** *B-posA* **by** (*simp add*: *assms emeasure-restrict-space*)
obtain *N* **where** *N*>*0 emeasure M* (((*induced-map A*)^^*N*)−‘*B* ∩ *B*) > *0* **using** *imp B-meas B-incl B-pos* **by** *auto*
then have *emeasure* (*restrict-space M A*) ((*induced-map A* ^^ *N*) −‘ *B* ∩ *B*) > *0*
using *assms emeasure-restrict-space* **by** (*metis B-incl Int-lower2 sets.Int-space-eq2 subset-trans*)
then show ∃ *n*>*0*. *0* < *emeasure* (*restrict-space M A*) ((*induced-map A* ^^ *n*) −‘ *B* ∩ *B*)
using ‹*N* > *0*› **by** *auto*
qed

Now, we want to prove that, if a map is conservative and measure preserving, then the induced map is also measure preserving. We first prove it for subsets W of A of finite measure, the general case will readily follow.

The argument uses the fact that the preimage of the set of points with first entrance time n is the union of the set of points with first entrance time $n+1$, and the points of A with first return $n+1$. Following the preimage of W under this process, we will get the intersection of $T_A^{-1}W$ with the different elements of the return partition, and the points in $T^{-n}W$ whose first $n-1$ iterates do not meet A (and the measures of these sets add up to $\mu(W)$).

To conclude, it suffices to show that the measure of points in $T^{-n}W$ whose first $n-1$ iterates do not meet A tends to 0. This follows from our local times estimates above.

lemma (**in** *conservative-mpt*) *induced-map-measure-preserving-aux*:
  assumes *A-meas* [*measurable*]: $A \in$ *sets M*
    and *W-meas* [*measurable*]: $W \in$ *sets M*
    and *incl*: $W \subseteq A$
    and *fin*: *emeasure M W* $< \infty$
  shows *emeasure M* ((*induced-map A*)−−'*W*) = *emeasure M W*
proof −
  have $W \subseteq$ *space M* **using** *W-meas*
    using *sets.sets-into-space* **by** *blast*
  define *BW* **where** *BW* = (λ*n.* (*first-entrance-set A n*) ∩ (T^^*n*)−−'*W*)
  define *DW* **where** *DW* = (λ*n.* (*return-time-function A*)−' {*n*} ∩ (*induced-map A*)−−'*W*)

  have $\bigwedge$*n. DW n* = (*return-time-function A*)−' {*n*} ∩ *space M* ∩ (*induced-map A*)−−'*W*
    using *DW-def* **by** *auto*
  then have *DW-meas* [*measurable*]: $\bigwedge$*n. DW n* ∈ *sets M* **by** *auto*
  have *disj-DW*: *disjoint-family* (λ*n. DW n*) **using** *DW-def disjoint-family-on-def* **by** *blast*
  then have *disj-DW2*: *disjoint-family* (λ*n. DW* (*n+1*)) **by** (*simp add*: *disjoint-family-on-def*)

  have ($\bigcup$*n. DW n*) = *DW 0* ∪ ($\bigcup$*n. DW* (*n+1*)) **by** (*auto*) (*metis not0-implies-Suc*)
  moreover have (*DW 0*) ∩ ($\bigcup$*n. DW* (*n+1*)) = {}
  by (*auto*) (*metis IntI Suc-neq-Zero UNIV-I empty-iff disj-DW disjoint-family-on-def*)
  ultimately have ∗: *emeasure M* ($\bigcup$*n. DW n*) = *emeasure M* (*DW 0*) + *emeasure M* ($\bigcup$*n. DW* (*n+1*))
    by (*simp add*: *countable-Un-Int*(*1*) *plus-emeasure*)

  have *DW 0* = (*return-time-function A*)−' {*0*} ∩ *W*
    unfolding *DW-def induced-map-def return-time-function-def*
    apply (*auto simp add*: *return-time0*[*of A*]) **using** *sets.sets-into-space*[*OF W-meas*] **by** *auto*
  also have ... = *W* − *recurrent-subset A* **using** *return-time0* **by** *blast*
  also have ... ⊆ *A* − *recurrent-subset A* **using** *incl* **by** *blast*
  finally have *DW 0* ∈ *null-sets M* **by** (*metis A-meas DW-meas null-sets-subset Poincare-recurrence-thm*(*1*))
  then have *emeasure M* (*DW 0*) = *0* **by** *auto*
  have (*induced-map A*)−−'*W* = ($\bigcup$*n. DW n*) **using** *DW-def* **by** *blast*
  then have *emeasure M* ((*induced-map A*)−−'*W*) = *emeasure M* ($\bigcup$*n. DW n*) **by** *simp*
  also have ... = *emeasure M* ($\bigcup$*n. DW* (*n+1*)) **using** ∗ ‹*emeasure M* (*DW 0*) = *0*› **by** *simp*
  also have ... = ($\sum$*n. emeasure M* (*DW* (*n+1*)))
    apply (*rule suminf-emeasure*[*symmetric*]) **using** *disj-DW2* **by** *auto*
  finally have *m*: *emeasure M* ((*induced-map A*)−−'*W*) = ($\sum$*n. emeasure M*

(*DW* (*n*+*1*))) **by** *simp*
moreover have *summable* (λ*n*. *emeasure M* (*DW* (*n*+*1*))) **by** *simp*
ultimately have *lim*: (λ*N*. (∑ *n*∈{..<*N*}. *emeasure M* (*DW* (*n*+*1*)))) ⟶ *emeasure M* ((*induced-map A*)−−‘*W*)
by (*simp add*: *summable-LIMSEQ*)

have *BW-meas* [*measurable*]: ⋀*n*. *BW n* ∈ *sets M* **unfolding** *BW-def* **by** *simp*
have ∗: ⋀*n*. *T*−−‘(*BW n*) − *A* = *BW* (*n*+*1*)
proof −
fix *n*
have *T*−−‘(*BW n*) = *T*−−‘(*first-entrance-set A n*) ∩ (*T*^^(*n*+*1*))−−‘*W*
unfolding *BW-def* **by** (*simp add*: *assms*(*2*) *T-vrestr-composed*(*2*))
then have *T*−−‘(*BW n*) − *A* = (*T*−−‘(*first-entrance-set A n*) − *A*) ∩ (*T*^^(*n*+*1*))−−‘*W*
by *blast*
then have *T*−−‘(*BW n*) − *A* = *first-entrance-set A* (*n*+*1*) ∩ (*T*^^(*n*+*1*))−−‘*W*
using *first-entrance-rec*[*OF A-meas*] **by** *simp*
then show *T*−−‘(*BW n*) − *A* = *BW* (*n*+*1*) **using** *BW-def* **by** *simp*
qed

have ∗∗: *DW* (*n*+*1*) = *T*−−‘(*BW n*) ∩ *A* **for** *n*
proof −
have *T*−−‘(*BW n*) = *T*−−‘(*first-entrance-set A n*) ∩ (*T*^^(*n*+*1*))−−‘*W*
unfolding *BW-def* **by** (*simp add*: *assms*(*2*) *T-vrestr-composed*(*2*))
then have *T*−−‘(*BW n*) ∩ *A* = (*T*−−‘(*first-entrance-set A n*) ∩ *A*) ∩ (*T*^^(*n*+*1*))−−‘*W*
by *blast*
then have ∗: *T*−−‘(*BW n*) ∩ *A* = (*return-time-function A*)−‘{*n*+*1*} ∩ (*T*^^(*n*+*1*))−−‘*W*
using *return-time-rec*[*OF A-meas*] **by** *simp*

have *DW* (*n*+*1*) = (*return-time-function A*)−‘{*n*+*1*} ∩ (*induced-map A*)−‘*W*
using *DW-def* ‹*W* ⊆ *space M*› *return-time-rec* **by** *auto*
also have ... = (*return-time-function A*)−‘{*n*+*1*} ∩ (*T*^^(*n*+*1*))−‘*W*
by (*auto simp add*: *induced-map-def*)
also have ... = (*return-time-function A*)−‘{*n*+*1*} ∩ (*T*^^(*n*+*1*))−−‘*W*
using ‹*W* ⊆ *space M*› *return-time-rec* **by** *auto*
finally show *DW* (*n*+*1*) = *T*−−‘(*BW n*) ∩ *A* **using** ∗ **by** *simp*
qed

have *emeasure M W* = (∑ *n*∈{..<*N*}. *emeasure M* (*DW* (*n*+*1*))) + *emeasure M* (*BW N*) **for** *N*
proof (*induction N*)
case *0*
have *BW 0* = *W* **unfolding** *BW-def first-entrance-set-def* **using** *incl* **by** *auto*
then show *?case* **by** *simp*
next
case (*Suc N*)
have *T*−−‘(*BW N*) = *BW* (*N*+*1*) ∪ *DW* (*N*+*1*) **using** ∗ ∗∗ **by** *blast*

moreover have *BW* (*N*+*1*) ∩ *DW* (*N*+*1*) = {} **using** ∗ ∗∗ **by** *blast*
ultimately have *emeasure M* (*T*−−‘(*BW N*)) = *emeasure M* (*BW* (*N*+*1*)) + *emeasure M* (*DW* (*N*+*1*))
using *DW-meas BW-meas plus-emeasure*[*of BW* (*N*+*1*)] **by** *simp*
then have *emeasure M* (*BW N*) = *emeasure M* (*BW* (*N*+*1*)) + *emeasure M* (*DW* (*N*+*1*))
using *T-vrestr-same-emeasure*(*1*) *BW-meas* **by** *auto*
then have ($\sum$ *n*∈{..<*N*}. *emeasure M* (*DW* (*n*+*1*))) + *emeasure M* (*BW N*) = ($\sum$ *n*∈{..<*N*+*1*}. *emeasure M* (*DW* (*n*+*1*))) + *emeasure M* (*BW* (*N*+*1*))
by (*simp add*: *add.commute add.left-commute*)
then show *?case* **using** *Suc.IH* **by** *simp*
qed
moreover
have (λ*N*. *emeasure M* (*BW N*)) ⟶ *0*
proof (*rule tendsto-sandwich*[*of* λ-. *0*- - λ*N*. *emeasure M* {*x* ∈ (*T*^^*N*)−−‘*W*. *local-time A N x* < *1*}])
have *emeasure M* (*BW N*) ≤ *emeasure M* {*x* ∈ (*T*^^*N*)−−‘*W*. *local-time A N x* < *1*} **for** *N*
apply (*rule emeasure-mono*) **unfolding** *BW-def local-time-def first-entrance-set-def* **by** *auto*
then show ∀$_F$ *n in sequentially*. *emeasure M* (*BW n*) ≤ *emeasure M* {*x* ∈ (*T* ^^ *n*) −−‘ *W*. *local-time A n x* < *1*}
by *auto*
have *i*: *W* ⊆ (*T*^^*0*)−−‘*A* **using** *incl* **by** *auto*
show (λ*N*. *emeasure M* {*x* ∈ (*T* ^^ *N*) −−‘ *W*. *local-time A N x* < *1*}) ⟶ *0*
apply (*rule local-time-unbounded2*[*OF* - - *i*]) **using** *fin* **by** *auto*
qed (*auto*)
then have (λ*N*. ($\sum$ *n*∈{..<*N*}. *emeasure M* (*DW* (*n*+*1*))) + *emeasure M* (*BW N*)) ⟶ *emeasure M* (*induced-map A* −−‘ *W*) + *0*
using *lim* **by** (*intro tendsto-add*) *auto*
ultimately show *?thesis*
by (*auto intro*: *LIMSEQ-unique LIMSEQ-const-iff*)
qed

lemma (**in** *conservative-mpt*) *induced-map-measure-preserving*:
assumes *A-meas* [*measurable*]: *A* ∈ *sets M*
and *W-meas* [*measurable*]: *W* ∈ *sets M*
shows *emeasure M* ((*induced-map A*)−−‘*W*) = *emeasure M W*
proof −
define *WA* **where** *WA* = *W* ∩ *A*
have *WA-meas* [*measurable*]: *WA* ∈ *sets M WA* ⊆ *A* **using** *WA-def* **by** *auto*
have *WAi-meas* [*measurable*]: (*induced-map A*)−−‘*WA* ∈ *sets M* **by** *simp*
have *a*: *emeasure M WA* = *emeasure M* ((*induced-map A*)−−‘*WA*)
proof (*cases*)
assume *emeasure M WA* < ∞
then show *?thesis* **using** *induced-map-measure-preserving-aux*[*OF A-meas*, *OF* ‹*WA* ∈ *sets M*›, *OF* ‹*WA* ⊆ *A*›] **by** *simp*

next
assume ¬(*emeasure M WA* < ∞)
then have *emeasure M WA* = ∞ **by** (*simp add*: *less-top*[*symmetric*])
{
fix *C*::*real*
obtain *Z* **where** *Z* ∈ *sets M Z* ⊆ *WA emeasure M Z* < ∞ *emeasure M Z* > *C*
by (*blast intro*: ‹*emeasure M WA* = ∞› *WA-meas approx-PInf-emeasure-with-finite*)
have *Z* ⊆ *A* **using** ‹*Z* ⊆ *WA*› *WA-def* **by** *simp*
have *C* < *emeasure M Z* **using** ‹*emeasure M Z* > *C*› **by** *simp*
also have ... = *emeasure M* ((*induced-map A*)−−‘*Z*)
using *induced-map-measure-preserving-aux*[*OF A-meas*, *OF* ‹*Z* ∈ *sets M*›, *OF* ‹*Z* ⊆ *A*›] ‹*emeasure M Z* < ∞› **by** *simp*
also have ... ≤ *emeasure M* ((*induced-map A*)−−‘*WA*)
apply(*rule emeasure-mono*) **using** ‹*Z* ⊆ *WA*› *vrestr-inclusion* **by** *auto*
finally have *emeasure M* ((*induced-map A*)−−‘*WA*) > *C* **by** *simp*
}
then have *emeasure M* ((*induced-map A*)−−‘*WA*) = ∞
by (*cases emeasure M* ((*induced-map A*)−−‘*WA*)) *auto*
then show *?thesis* **using** ‹*emeasure M WA* = ∞› **by** *simp*
qed
define *WB* **where** *WB* = *W* − *WA*
have *WB-meas* [*measurable*]: *WB* ∈ *sets M* **unfolding** *WB-def* **by** *simp*
have *WBi-meas* [*measurable*]: (*induced-map A*)−−‘*WB* ∈ *sets M* **by** *simp*
have *WB* ∩ *A* = {} **unfolding** *WB-def WA-def* **by** *auto*
moreover have *id*: ⋀*x*. *x* ∉ *A* ⟹ (*induced-map A x*) = *x* **unfolding** *induced-map-def return-time-function-def*
apply (*auto*) **using** *recurrent-subset-incl* **by** *auto*
ultimately have (*induced-map A*)−−‘*WB* = *WB*
using *induced-map-stabilizes-A sets.sets-into-space*[*OF WB-meas*] **apply** *auto*
by (*metis disjoint-iff-not-equal*) *fastforce+*
then have *b*: *emeasure M* ((*induced-map A*)−−‘*WB*) = *emeasure M WB* **by** *simp*

have *W* = *WA* ∪ *WB WA* ∩ *WB* = {} **using** *WA-def WB-def* **by** *auto*
have ∗: *emeasure M W* = *emeasure M WA* + *emeasure M WB*
by (*subst* ‹*W* = *WA* ∪ *WB*›, *rule plus-emeasure*[*symmetric*], *auto simp add*: ‹*WA* ∩ *WB* = {}›)

have *W-AUB*: (*induced-map A*)−−‘*W* = (*induced-map A*)−−‘*WA* ∪ (*induced-map A*)−−‘*WB*
using ‹*W* = *WA* ∪ *WB*› **by** *auto*
have *W-AIB*: (*induced-map A*)−−‘*WA* ∩ (*induced-map A*)−−‘*WB* = {}
by (*metis* ‹*WA* ∩ *WB* = {}› *vrestr-empty vrestr-intersec*)
have *emeasure M* ((*induced-map A*)−−‘*W*) = *emeasure M* ((*induced-map A*)−−‘*WA*) + *emeasure M* ((*induced-map A*)−−‘*WB*)
unfolding *W-AUB* **by** (*rule plus-emeasure*[*symmetric*]) (*auto simp add*: *W-AIB*)

then show *?thesis* **using** *a b* ∗ **by** *simp*

qed

We can now express the fact that induced maps preserve the measure.

theorem (**in** *conservative-mpt*) *induced-map-conservative-mpt*:
  assumes $A \in$ *sets M*
  shows *conservative-mpt* (*restrict-space M A*) (*induced-map A*)
unfolding *conservative-mpt-def*
proof
 show ∗: *conservative* (*restrict-space M A*) (*induced-map A*) **using** *induced-map-conservative*[*OF assms*] **by** *auto*
 show *mpt* (*restrict-space M A*) (*induced-map A*) **unfolding** *mpt-def mpt-axioms-def*
  proof
   show *qmpt* (*restrict-space M A*) (*induced-map A*) **using** ∗ *conservative-def* **by** *auto*
   then have *meas*: (*induced-map A*) ∈ *measurable* (*restrict-space M A*) (*restrict-space M A*)
     unfolding *qmpt-def qmpt-axioms-def quasi-measure-preserving-def* **by** *auto*
   moreover have ⋀*B*. *B* ∈ *sets* (*restrict-space M A*) ⟹
      emeasure (*restrict-space M A*) ((*induced-map A*) −‘*B* ∩ *space* (*restrict-space M A*)) = *emeasure* (*restrict-space M A*) *B*
   proof −
    have *s*: *space* (*restrict-space M A*) = *A* **using** *assms space-restrict-space2* **by** *auto*
     have *i*: ⋀*D*. *D* ∈ *sets M* ∧ *D* ⊆ *A* ⟹ *emeasure* (*restrict-space M A*) *D* = *emeasure M D*
      using *assms* **by** (*simp add*: *emeasure-restrict-space*)
    have *j*: ⋀*D*. *D* ∈ *sets* (*restrict-space M A*) ⟷ (*D* ∈ *sets M* ∧ *D* ⊆ *A*) **using** *assms*
      by (*metis sets.Int-space-eq2 sets-restrict-space-iff*)
    fix *B*
    assume *a*: *B* ∈ *sets* (*restrict-space M A*)
    then have *B-meas*: *B* ∈ *sets M* **using** *j* **by** *auto*
    then have *first*: *emeasure* (*restrict-space M A*) *B* = *emeasure M B* **using** *i j a* **by** *auto*
     have *incl*: (*induced-map A*) −‘*B* ⊆ *A* **using** *j a induced-map-stabilizes-A assms* **by** *auto*
   then have *eq*: (*induced-map A*) −‘*B* ∩ *space* (*restrict-space M A*) = (*induced-map A*) −−‘*B*
      unfolding *vimage-restr-def s* **using** *assms sets.sets-into-space*
      by (*metis a inf.orderE j meas measurable-sets s*)
     then have *emeasure M B* = *emeasure M* ((*induced-map A*) −‘*B* ∩ *space* (*restrict-space M A*))
      using *induced-map-measure-preserving a j assms* **by** *auto*
    also have ... = *emeasure* (*restrict-space M A*) ((*induced-map A*) −‘*B* ∩ *space* (*restrict-space M A*))
      using *incl eq B-meas induced-map-meas*[*OF assms*] *assms i j*
      by (*metis emeasure-restrict-space inf.orderE s space-restrict-space*)
    finally show *emeasure* (*restrict-space M A*) ((*induced-map A*) −‘*B* ∩ *space* (*restrict-space M A*))

= *emeasure* (*restrict-space M A*) *B*
using *first* **by** *auto*
qed
ultimately show *induced-map A* ∈ *measure-preserving* (*restrict-space M A*) (*restrict-space M A*)
unfolding *measure-preserving-def* **by** *auto*
qed
qed

theorem (**in** *fmpt*) *induced-map-fmpt*:
assumes *A* ∈ *sets M*
shows *fmpt* (*restrict-space M A*) (*induced-map A*)
unfolding *fmpt-def*
proof −
have *conservative-mpt* (*restrict-space M A*) (*induced-map A*) **using** *induced-map-conservative-mpt*[*OF assms*] **by** *simp*
then have *mpt* (*restrict-space M A*) (*induced-map A*) **using** *conservative-mpt-def* **by** *auto*
moreover have *finite-measure* (*restrict-space M A*) **by** (*simp add*: *assms finite-measureI finite-measure-restrict-space*)
ultimately show *mpt* (*restrict-space M A*) (*induced-map A*) ∧ *finite-measure* (*restrict-space M A*) **by** *simp*
qed

It will be useful to reformulate the fact that the recurrent subset has full measure in terms of the induced measure, to simplify the use of the induced map later on.

lemma (**in** *conservative*) *induced-map-recurrent-typical*:
assumes *A-meas* [*measurable*]: *A* ∈ *sets M*
shows *AE z in* (*restrict-space M A*). *z* ∈ *recurrent-subset A*
AE z in (*restrict-space M A*). *z* ∈ *recurrent-subset-infty A*
proof −
have *recurrent-subset A* ∈ *sets M* **using** *recurrent-subset-meas*[*OF A-meas*] **by** *auto*
then have *rsA*: *recurrent-subset A* ∈ *sets* (*restrict-space M A*)
using *recurrent-subset-incl*(*1*)[*of A*]
by (*metis* (*no-types*, *lifting*) *A-meas sets-restrict-space-iff space-restrict-space space-restrict-space2*)

have *emeasure* (*restrict-space M A*) (*space* (*restrict-space M A*) − *recurrent-subset A*) = *emeasure* (*restrict-space M A*) (*A* − *recurrent-subset A*)
by (*metis* (*no-types*, *lifting*) *A-meas space-restrict-space2*)
also have ... = *emeasure M* (*A* − *recurrent-subset A*)
by (*simp add*: *emeasure-restrict-space*)
also have ... = *0* **using** *Poincare-recurrence-thm*[*OF A-meas*] **by** *auto*
finally have *space* (*restrict-space M A*) − *recurrent-subset A* ∈ *null-sets* (*restrict-space M A*)
using *rsA* **by** *blast*
then show *AE z in* (*restrict-space M A*). *z* ∈ *recurrent-subset A*

by (*metis* (*no-types*, *lifting*) *DiffI eventually-ae-filter mem-Collect-eq subsetI*)

have *recurrent-subset-infty A* ∈ *sets M* **using** *recurrent-subset-meas*[*OF A-meas*]
by *auto*
then have *rsiA*: *recurrent-subset-infty A* ∈ *sets* (*restrict-space M A*)
using *recurrent-subset-incl*(*2*)[*of A*]
by (*metis* (*no-types*, *lifting*) *A-meas sets-restrict-space-iff space-restrict-space space-restrict-space2*)

have *emeasure* (*restrict-space M A*) (*space* (*restrict-space M A*) − *recurrent-subset-infty A*) = *emeasure* (*restrict-space M A*) (*A* − *recurrent-subset-infty A*)
by (*metis* (*no-types*, *lifting*) *A-meas space-restrict-space2*)
also have ... = *emeasure M* (*A* − *recurrent-subset-infty A*)
apply (*rule emeasure-restrict-space*) **using** *A-meas* **by** *auto*
also have ... = *0* **using** *Poincare-recurrence-thm*[*OF A-meas*] **by** *auto*
finally have *space* (*restrict-space M A*) − *recurrent-subset-infty A* ∈ *null-sets* (*restrict-space M A*)
using *rsiA* **by** *blast*
then show *AE z in* (*restrict-space M A*). *z* ∈ *recurrent-subset-infty A*
by (*metis* (*no-types*, *lifting*) *DiffI eventually-ae-filter mem-Collect-eq subsetI*)
qed

5.5 Kac's theorem, and variants

Kac's theorem states that, for conservative maps, the integral of the return time to a subset A is equal to the measure of the space if the dynamics is ergodic, or of the space seen by A in the general case.

This result generalizes to any induced function, not just the return time, that we define now.

definition *induced-function*::′*a set* ⇒ (′*a* ⇒ ′*b*::*comm-monoid-add*) ⇒ (′*a* ⇒ ′*b*)
where *induced-function A f* = (λ*x*. (∑ *i*∈{..< *return-time-function A x*}. *f*((*T*^^*i*) *x*)))

By definition, the induced function is supported on the recurrent subset of A.

lemma *induced-function-support*:
fixes *f*::′*a* ⇒ *ennreal*
shows *induced-function A f y* = *induced-function A f y* ∗ *indicator* ((*return-time-function A*)−‘{*1*..}) *y*
by (*auto simp add*: *induced-function-def indicator-def not-less-eq-eq*)

Basic measurability statements.

lemma *induced-function-meas-ennreal* [*measurable*]:
fixes *f*::′*a* ⇒ *ennreal*
assumes [*measurable*]: *f* ∈ *borel-measurable M A* ∈ *sets M*
shows *induced-function A f* ∈ *borel-measurable M*
unfolding *induced-function-def* **by** *simp*

lemma *induced-function-meas-real* [*measurable*]:
  fixes *f*::′*a* ⇒ *real*
  assumes [*measurable*]: *f* ∈ *borel-measurable M A* ∈ *sets M*
  shows *induced-function A f* ∈ *borel-measurable M*
unfolding *induced-function-def* **by** *simp*

The Birkhoff sums of the induced function for the induced map form a subsequence of the original Birkhoff sums for the original map, corresponding to the return times to A.

lemma (**in** *conservative*) *induced-function-birkhoff-sum*:
  fixes *f*::′*a* ⇒ *real*
  assumes *A* ∈ *sets M*
  shows *birkhoff-sum f* (*qmpt.birkhoff-sum* (*induced-map A*) (*return-time-function A*) *n x*) *x*
        = *qmpt.birkhoff-sum* (*induced-map A*) (*induced-function A f*) *n x*
proof −
 interpret *A*: *conservative restrict-space M A induced-map A* **by** (*rule induced-map-conservative*[*OF assms*])
  define *TA* **where** *TA* = *induced-map A*
  define *phiA* **where** *phiA* = *return-time-function A*
  define *R* **where** *R* = (λ*n*. *A.birkhoff-sum phiA n x*)
  show *?thesis*
  proof (*induction n*)
    case *0*
    show *?case* **using** *birkhoff-sum-1*(*1*) *A.birkhoff-sum-1*(*1*) **by** *auto*
  next
    case (*Suc n*)
    have (*T*^^(*R n*)) *x* = (*TA*^^*n*) *x* **unfolding** *TA-def R-def A.birkhoff-sum-def phiA-def* **by** (*rule induced-map-iterates*[*symmetric*])
    have *R*(*n*+*1*) = *R n* + *phiA* ((*TA*^^*n*) *x*)
      unfolding *R-def* **using** *A.birkhoff-sum-cocycle*[**where** *?n* = *n* **and** *?m* = *1* **and** *?f* = *phiA*] *A.birkhoff-sum-1*(*2*) *TA-def* **by** *auto*
    then have *birkhoff-sum f* (*R* (*n*+*1*)) *x* = *birkhoff-sum f* (*R n*) *x* + *birkhoff-sum f* (*phiA* ((*TA*^^*n*) *x*)) ((*T*^^(*R n*)) *x*)
      using *birkhoff-sum-cocycle*[**where** *?n* = *R n* **and** *?f* = *f*] **by** *auto*
    also have ... = *birkhoff-sum f* (*R n*) *x* + *birkhoff-sum f* (*phiA* ((*TA*^^*n*) *x*)) ((*TA*^^*n*) *x*)
      using ‹(*T*^^(*R n*)) *x* = (*TA*^^*n*) *x*› **by** *simp*
    also have ... = *birkhoff-sum f* (*R n*) *x* + (*induced-function A f*) ((*TA*^^*n*) *x*)
      unfolding *induced-function-def birkhoff-sum-def phiA-def* **by** *simp*
    also have ... = *A.birkhoff-sum* (*induced-function A f*) *n x* + (*induced-function A f*) ((*TA*^^*n*) *x*) **using** *Suc.IH R-def phiA-def* **by** *auto*
    also have ... = *A.birkhoff-sum* (*induced-function A f*) (*n*+*1*) *x*
      using *A.birkhoff-sum-cocycle*[**where** *?n* = *n* **and** *?m* = *1* **and** *?f* = *induced-function A f* **and** *?x* = *x*, *symmetric*]
      A.birkhoff-sum-1(*2*)[**where** *?f* = *induced-function A f* **and** *?x* = (*TA*^^*n*) *x*]
      unfolding *TA-def* **by** *auto*
    finally show *?case* **unfolding** *R-def phiA-def* **by** *simp*

qed
qed

The next lemma is very simple (just a change of variables to reorder the indices in the double sum). However, the proof I give is very tedious: infinite sums on proper subsets are not handled well, hence I use integrals on products of discrete spaces instead, and go back and forth between the two notions – maybe there are better suited tools in the library, but I could not locate them...

This is the main combinatorial tool used in the proof of Kac's Formula.

lemma *kac-series-aux*:
fixes *d*:: *nat* ⇒ *nat* ⇒ *ennreal*
shows $(\sum n.\ (\sum i{\leq}n.\ d\ i\ n)) = (\sum n.\ d\ 0\ n) + (\sum n.\ (\sum i.\ d\ (i{+}1)\ (n{+}1{+}i)))$
(**is** - = *?R*)
proof −
define *g* **where** $g = (\lambda(i,n).\ (i{+}(1{::}nat),\ n{+}1{+}i))$
define *U* **where** $U = \{(i,n).\ (i{>}(0{::}nat)) \wedge (n{\geq}i)\}$
have *bij*: *bij-betw g UNIV U*
by (*rule bij-betw-byWitness*[**where** $?f' = \lambda(i,\ n).\ (i{-}1,\ n{-}i)$], *auto simp add*: *g-def U-def*)

define *e* **where** $e = (\lambda\ (i,j).\ d\ i\ j)$
have *pos*: $\bigwedge x.\ e\ x \geq 0$ **using** *e-def* **by** *auto*
have $(\sum n.\ (\sum i.\ d\ (i{+}1)\ (n{+}1{+}i))) = (\sum n.\ (\sum i.\ e(i{+}1,\ n{+}1{+}i)))$ **using** *e-def* **by** *simp*
also have ... $= \int^{+}n.\ \int^{+}i.\ e\ (i{+}1,\ n{+}1{+}i)\ \partial count\text{-}space\ UNIV\ \partial count\text{-}space\ UNIV$
using *pos nn-integral-count-space-nat suminf-0-le* **by** *auto*
also have ... $= (\int^{+}x.\ e\ (g\ x)\ \partial count\text{-}space\ UNIV)$
unfolding *g-def* **using** *nn-integral-snd-count-space*[*of* $\lambda(i,n).\ e(i{+}1,\ n{+}1{+}i)$]
by (*auto simp add*: *prod.case-distrib*)
also have ... $= (\int^{+}y \in U.\ e\ y\ \partial count\text{-}space\ UNIV)$
using *nn-integral-count-compose-bij*[*OF bij*] **by** *simp*
finally have ∗: $(\sum n.\ (\sum i.\ d\ (i{+}1)\ (n{+}1{+}i))) = (\int^{+}y \in U.\ e\ y\ \partial count\text{-}space\ UNIV)$
by *simp*

define *V* **where** $V = \{((i{::}nat),(n{::}nat)).\ i = 0\}$
have *i*: $e\ (i,\ n) * indicator\ \{0\}\ i = e\ (i,\ n) * indicator\ V\ (i,\ n)$ **for** *i n*
by (*auto simp add*: *indicator-def V-def*)
have $d\ 0\ n = (\int^{+}i \in \{0\}.\ e\ (i,\ n)\ \partial count\text{-}space\ UNIV)$ **for** *n*
proof −
have $(\int^{+}i \in \{0\}.\ e\ (i,\ n)\ \partial count\text{-}space\ UNIV) = (\int^{+}i.\ e\ (i,\ n)\ \partial count\text{-}space\ \{0\})$
using *nn-integral-count-space-indicator*[*of* - $\lambda i.\ e(i,\ n)$] **by** *simp*
also have ... $= e\ (0,\ n)$
using *nn-integral-count-space-finite*[**where** $?f = \lambda i.\ e\ (i,\ n)$] **by** *simp*
finally show *?thesis* **using** *e-def* **by** *simp*

qed
then have ($\sum$ *n. d 0 n*) = ($\sum$ *n.* ($\int^+$*i. e (i, n) ∗ indicator {0} i ∂count-space UNIV*))
by *simp*
also have ... = ($\int^+$*n.* ($\int^+$*i. e (i, n) ∗ indicator {0} i ∂count-space UNIV*) *∂count-space UNIV*)
by (*simp add*: *nn-integral-count-space-nat*)
also have ... = ($\int^+$*(i,n). e (i, n) ∗ indicator {0} i ∂count-space UNIV*)
using *nn-integral-snd-count-space*[*of* λ *(i,n). e(i,n) ∗ indicator {0} i*] **by** *auto*
also have ... = ($\int^+$*(i,n). e (i, n) ∗ indicator V (i,n) ∂count-space UNIV*)
by (*metis i*)
finally have ($\sum$ *n. d 0 n*) = ($\int^+$*y* ∈ *V. e y ∂count-space UNIV*)
by (*simp add*: *split-def*)

then have *?R* = ($\int^+$*y* ∈ *V. e y ∂count-space UNIV*) + ($\int^+$*y* ∈ *U. e y ∂count-space UNIV*)
using ∗ **by** *simp*
also have ... = ($\int^+$*y* ∈ *V* ∪ *U. e y ∂count-space UNIV*)
by (*rule nn-integral-disjoint-pair-countspace*[*symmetric*], *auto simp add*: *U-def V-def*)
also have ... = ($\int^+$*(i, n). e (i, n) ∗ indicator {..n} i ∂count-space UNIV*)
by (*rule nn-integral-cong*, *auto simp add*: *indicator-def of-bool-def V-def U-def pos*, *meson*)
also have ... = ($\int^+$*n.* ($\int^+$*i. e (i, n) ∗ indicator {..n} i ∂count-space UNIV*)*∂count-space UNIV*)
using *nn-integral-snd-count-space*[*of* λ*(i,n). e(i,n) ∗ indicator {..n} i*] **by** *auto*
also have ... = ($\sum$ *n.* ($\sum$ *i. e (i, n) ∗ indicator {..n} i*))
using *pos nn-integral-count-space-nat suminf-0-le* **by** *auto*
moreover have ($\sum$ *i. e (i, n) ∗ indicator {..n} i*) = ($\sum$ *i≤n. e (i, n)*) **for** *n*
proof −
have *finite {..n}* **by** *simp*
moreover have ⋀*i. i* ∉ *{..n}* ⟹ *e (i,n) ∗ indicator {..n} i = 0* **using** *indicator-def* **by** *simp*
then have ($\sum$ *i. e (i,n) ∗ indicator {..n} i*) = ($\sum$ *i* ∈ *{..n} . e (i, n) ∗ indicator {..n} i*)
by (*meson calculation suminf-finite*)
moreover have ⋀*i. i* ∈ *{..n}* ⟹ *e (i, n) ∗ indicator {..n} i = e (i, n)* **using** *indicator-def* **by** *auto*
ultimately show ($\sum$ *i. e (i, n) ∗ indicator {..n} i*) = ($\sum$ *i≤n. e (i, n)*) **by** *simp*
qed
ultimately show *?thesis* **using** *e-def* **by** *simp*
qed

end

context *conservative-mpt* **begin**

We prove Kac's Formula (in the general form for induced functions) first for functions taking values in ennreal (to avoid all summabilities issues). The result for real functions will follow by domination. First, we assume additionally that f is bounded and has a support of finite measure, the general case will follow readily by truncation.

The proof is again an instance of the fact that the preimage of the set of points with first entrance time n is the union of the set of points with first entrance time $n+1$, and the points of A with first return $n+1$. Keeping track of the integral of f on the different parts that appear in this argument, we will see that the integral of the induced function on the set of points with return time at most n is equal to the integral of the function, up to an error controlled by the measure of points in $T^{-n}(supp(f))$ with local time 0. Local time controls ensure that this contribution vanishes asymptotically.

lemma *induced-function-nn-integral-aux*:
  fixes *f*::*'a* ⇒ *ennreal*
  assumes *A-meas* [*measurable*]: *A* ∈ *sets M*
    and *f-meas* [*measurable*]: *f* ∈ *borel-measurable M*
    and *f-bound*: ⋀*x*. *f x* ≤ *ennreal C 0* ≤ *C*
    and *f-supp*: *emeasure M* {*x* ∈ *space M*. *f x* > *0*} < ∞
  shows (∫ ⁺*y*. *induced-function A f y* ∂*M*) = (∫ ⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *f x* ∂*M*)
proof −
  define *B* **where** *B* = (λ*n*. *first-entrance-set A n*)
  have *B-meas* [*measurable*]: ⋀*n*. *B n* ∈ *sets M* **by** (*simp add*: *B-def*)
  then have *B2* [*measurable*]: (⋃*n*. *B* (*n+1*)) ∈ *sets M* **by** *measurable*

  have ∗: *B* = *disjointed* (λ*i*. (*T*^^*i*)−−‘*A*)
    by (*auto simp add*: *B-def disjointed-def first-entrance-set-def*)
  then have *disjoint-family B* **by** (*simp add*: *disjoint-family-disjointed*)

  have (⋃*n*. (*T*^^*n*)−−‘*A*) = (⋃*n*. *disjointed* (λ*i*. (*T*^^*i*)−−‘*A*) *n*) **by** (*simp add*: *UN-disjointed-eq*)
  then have (⋃*n*. (*T*^^*n*)−−‘*A*) = (⋃*n*. *B n*) **using** ∗ **by** *simp*
  then have (⋃*n*. (*T*^^*n*)−−‘*A*) = *B 0* ∪ (⋃*n*. *B* (*n+1*)) **by** (*auto*) (*metis not0-implies-Suc*)
  then have (∫ ⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *f x* ∂*M*) = (∫ ⁺ *x* ∈ (*B 0* ∪ (⋃*n*. *B* (*n+1*))). *f x* ∂*M*) **by** *simp*
  also have ... = (∫ ⁺ *x* ∈ *B 0*. *f x* ∂*M*) + (∫ ⁺ *x* ∈ (⋃*n*. *B* (*n+1*)). *f x* ∂*M*)
  proof (*rule nn-integral-disjoint-pair*)
    show *B 0* ∩ (⋃*n*. *B* (*n+1*)) = {}
      by (*auto*) (*metis IntI Suc-neq-Zero UNIV-I empty-iff* ‹*disjoint-family B*› *disjoint-family-on-def*)
  qed *auto*
  finally have (∫ ⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *f x* ∂*M*) = (∫ ⁺ *x* ∈ *B 0*. *f x* ∂*M*) + (∫ ⁺ *x* ∈ (⋃*n*. *B* (*n+1*)). *f x* ∂*M*)
    by *simp*
  moreover have (∫ ⁺ *x* ∈ (⋃*n*. *B* (*n+1*)). *f x* ∂*M*) = (∑*n*. (∫ ⁺ *x* ∈ *B* (*n+1*). *f x* ∂*M*))

apply (*rule nn-integral-disjoint-family*) **using** ‹*disjoint-family B*› **by** (*auto simp add*: *disjoint-family-on-def*)
ultimately have *Bdec*: $(\int^+ x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n){-}{-}`A).\ f\ x\ \partial M) = (\int^+ x \in B\ 0.\ f\ x\ \partial M) + (\sum n.\ \int^+ x \in B\ (n+1).\ f\ x\ \partial M)$ **by** *simp*

define *D* **where** $D = (\lambda n.\ (\textit{return-time-function}\ A){-}`\ \{n+1\})$
then have *disjoint-family D* **by** (*auto simp add*: *disjoint-family-on-def*)
have $*$: $\bigwedge n.\ D\ n = T{-}{-}`(B\ n) \cap A$
using *D-def B-def return-time-rec*[*OF assms*(*1*)] **by** *simp*
then have [*measurable*]: $\bigwedge n.\ D\ n \in \textit{sets}\ M$ **by** *simp*
have $**$: $\bigwedge n.\ B\ (n+1) = T{-}{-}`(B\ n) - A$ **using** *first-entrance-rec*[*OF assms*(*1*)] *B-def* **by** *simp*

have *pos0*: $\bigwedge i\ x.\ f((T\widehat{\ }\widehat{\ }i)x) \geq 0$ **using** *assms*(*3*) **by** *simp*
have *pos*: $\bigwedge i\ C\ x.\ f((T\widehat{\ }\widehat{\ }i)x) * \textit{indicator}\ (C)\ x \geq 0$ **using** *assms*(*3*) **by** *simp*
have *mes0* [*measurable*]: $\bigwedge i.\ (\lambda x.\ f((T\widehat{\ }\widehat{\ }i)x)) \in \textit{borel-measurable}\ M$ **by** *simp*
then have [*measurable*]: $\bigwedge i\ C.\ C \in \textit{sets}\ M \Longrightarrow (\lambda x.\ f((T\widehat{\ }\widehat{\ }i)x) * \textit{indicator}\ C\ x) \in \textit{borel-measurable}\ M$ **by** *simp*

have $\bigwedge y.$ *induced-function A f y* = *induced-function A f y* $*$ *indicator* ((*return-time-function A*)$-$‘$\{1..\}$) *y*
by (*rule induced-function-support*)
moreover have (*return-time-function A*)$-$‘$\{(1{::}nat)..\} = (\bigcup n.\ D\ n)$
by (*auto simp add*: *D-def Suc-le-D*)
ultimately have $\bigwedge y.$ *induced-function A f y* = *induced-function A f y* $*$ *indicator* $(\bigcup n.\ D\ n)\ y$ **by** *simp*
then have $(\int^+ y.$ *induced-function A f y* $\partial M) = (\int^+ y \in (\bigcup n.\ D\ n).$ *induced-function A f y* $\partial M)$ **by** *simp*
also have ... $= (\sum n.\ (\int^+ y \in D\ n.$ *induced-function A f y* $\partial M))$
apply (*rule nn-integral-disjoint-family*)
unfolding *induced-function-def* **by** (*auto simp add*: *pos0 sum-nonneg* ‹*disjoint-family D*›)
finally have *a*: $(\int^+ y.$ *induced-function A f y* $\partial M) = (\sum n.\ (\int^+ y \in D\ n.$ *induced-function A f y* $\partial M))$
by *simp*

define *d* **where** $d = (\lambda i\ n.\ (\int^+ y \in D\ n.\ f((T\widehat{\ }\widehat{\ }i)y)\ \partial M))$

have $(\int^+ y \in D\ n.$ *induced-function A f y* $\partial M) = (\sum i{\in}\{..n\}.\ d\ i\ n)$ **for** *n*
proof $-$
have *induced-function A f y* $*$ *indicator* $(D\ n)\ y = (\sum i{\in}\{..{<}n+1\}.\ f((T\widehat{\ }\widehat{\ }i)y) * \textit{indicator}\ (D\ n)\ y)$ **for** *y*
by (*auto simp add*: *induced-function-def D-def indicator-def*)
then have $(\int^+ y \in D\ n.$ *induced-function A f y* $\partial M) = (\sum i{\in}\{..{<}n+1\}.\ (\int^+ y \in D\ n.\ f((T\widehat{\ }\widehat{\ }i)y)\ \partial M))$
using *pos nn-integral-sum*[*of* $\{..{<}n+1\}$, *of* $\lambda i\ y.\ f((T\widehat{\ }\widehat{\ }i)y) * \textit{indicator}\ (D\ n)\ y$] **by** *simp*
also have ... $= (\sum i{\in}\{..n\}.\ (\int^+ y \in D\ n.\ f((T\widehat{\ }\widehat{\ }i)y)\ \partial M))$
using *lessThan-Suc-atMost* **by** *auto*

finally show *?thesis* **using** *d-def* **by** *simp*
qed
then have *induced-dec*: $(\int^+ y.\ \textit{induced-function A f y}\ \partial M) = (\sum n.\ (\sum i{\in}\{..n\}.\ d\ i\ n))$
using *a* **by** *simp*

have $(\bigcup n{\in}\{1..\}.\ (\textit{return-time-function}\ A){-}`\{n\}) = \textit{UNIV} - (\textit{return-time-function}\ A){-}`\{0\}$ **by** *auto*
then have $(\bigcup n{\in}\{1..\}.\ (\textit{return-time-function}\ A){-}`\{n\}) = \textit{recurrent-subset}\ A$
using *return-time0* **by** *auto*
moreover have $(\bigcup n.\ (\textit{return-time-function}\ A){-}`\{n{+}1\}) = (\bigcup n{\in}\{1..\}.\ (\textit{return-time-function}\ A){-}`\{n\})$
by (*auto simp add*: *Suc-le-D*)
ultimately have $(\bigcup n.\ D\ n) = \textit{recurrent-subset}\ A$ **using** *D-def* **by** *simp*
moreover have $(\int^+ x \in A.\ f\ x\ \partial M) = (\int^+ x \in \textit{recurrent-subset}\ A.\ f\ x\ \partial M)$
by (*rule nn-integral-null-delta*, *auto simp add*: *Diff-mono Un-absorb2 recurrent-subset-incl*(*1*)[*of A*] *Poincare-recurrence-thm*(*1*)[*OF assms*(*1*)])
moreover have $B\ 0 = A$ **using** *B-def first-entrance-set-def* **by** *simp*
ultimately have $(\int^+ x \in B\ 0.\ f\ x\ \partial M) = (\int^+ x \in (\bigcup n.\ D\ n).\ f\ x\ \partial M)$ **by** *simp*
also have ... $= (\sum n.\ (\int^+ x \in D\ n.\ f\ x\ \partial M))$
by (*rule nn-integral-disjoint-family*, *auto simp add*: ‹*disjoint-family D*›)
finally have *B0dec*: $(\int^+ x \in B\ 0.\ f\ x\ \partial M) = (\sum n.\ d\ 0\ n)$ **using** *d-def* **by** *simp*

have $*$: $(\int^+ x \in B\ n.\ f\ x\ \partial M) = (\sum i{<}k.\ (\int^+ x \in D(n{+}i).\ f((T\widehat{\ }\widehat{\ }(i{+}1))x)\ \partial M)) + (\int^+ x \in B(n{+}k).\ f((T\widehat{\ }\widehat{\ }k)x)\ \partial M)$ **for** *n k*
proof (*induction k*)
case *0*
show *?case* **by** *simp*
next
case (*Suc k*)
have $T{-}{-}`(B(n{+}k)) = B(n{+}k{+}1) \cup D(n{+}k)$ **using** $*$ $**$ **by** *blast*

have $(\int^+ x \in B(n{+}k).\ f((T\widehat{\ }\widehat{\ }k)x)\ \partial M) = (\int^+ x.\ (\lambda x.\ f((T\widehat{\ }\widehat{\ }k)x) * \textit{indicator}\ (B\ (n{+}k))\ x)(T\ x)\ \partial M)$
by (*rule measure-preserving-preserves-nn-integral*[*OF Tm*], *auto simp add*: *pos*)
also have ... $= (\int^+ x.\ f((T\widehat{\ }\widehat{\ }(k{+}1))x) * \textit{indicator}\ (T{-}{-}`(B\ (n{+}k)))\ x\ \partial M)$
proof (*rule nn-integral-cong-AE*)
have $(T\widehat{\ }\widehat{\ }k)(T\ x) = (T\widehat{\ }\widehat{\ }(k{+}1))x$ **for** *x*
using *comp-eq-dest-lhs* **by** (*metis Suc-eq-plus1 funpow.simps*(*2*) *funpow-swap1*)
moreover have $\textit{AE}\ x\ \textit{in}\ M.\ f((T\widehat{\ }\widehat{\ }k)(T\ x)) * \textit{indicator}\ (B\ (n{+}k))\ (T\ x) = f((T\widehat{\ }\widehat{\ }k)(T\ x)) * \textit{indicator}\ (T{-}{-}`(B\ (n{+}k)))\ x$
by (*simp add*: *indicator-def* ‹$\bigwedge n.\ B\ n \in \textit{sets}\ M$›)
ultimately show $\textit{AE}\ x\ \textit{in}\ M.\ f((T\widehat{\ }\widehat{\ }k)(T\ x)) * \textit{indicator}\ (B\ (n{+}k))\ (T\ x) = f((T\widehat{\ }\widehat{\ }(k{+}1))x) * \textit{indicator}\ (T{-}{-}`(B\ (n{+}k)))\ x$
by *simp*
qed
also have ... $= (\int^+ x \in B(n{+}k{+}1) \cup D(n{+}k).\ f((T\widehat{\ }\widehat{\ }(k{+}1))x)\ \partial M)$
using ‹$T{-}{-}`(B(n{+}k)) = B(n{+}k{+}1) \cup D(n{+}k)$› **by** *simp*

also have ... = (∫ ⁺x ∈ B(n+k+1). f((T^^(k+1))x) ∂M) + (∫ ⁺x ∈ D(n+k). f((T^^(k+1))x) ∂M)
proof (*rule nn-integral-disjoint-pair*[*OF mes0*[*of k+1*]])
show B(n+k+1) ∩ D(n+k) = {} **using** ∗ ∗∗ **by** *blast*
qed (*auto*)
finally have (∫ ⁺x ∈ B(n+k). f((T^^k)x) ∂M) = (∫ ⁺x ∈ B(n+k+1). f((T^^(k+1))x) ∂M) + (∫ ⁺x ∈ D(n+k). f((T^^(k+1))x) ∂M)
by *simp*
then show *?case* **by** (*simp add*: *Suc.IH add.commute add.left-commute*)
qed

have *a*: (λk. (∫ ⁺x ∈ B(n+k). f((T^^k)x) ∂M)) ⟶ *0* **for** *n*
proof −
define *W* **where** *W* = {*x* ∈ *space M*. *f x* > *0*} ∩ (*T*^^*n*)−−‘*A*
have *emeasure M W* ≤ *emeasure M* {*x* ∈ *space M*. *f x* > *0*}
by (*intro emeasure-mono*, *auto simp add*: *W-def*)
then have *W-fin*: *emeasure M W* < ∞ **using** *f-supp* **by** *auto*
have *W-meas* [*measurable*]: *W* ∈ *sets M* **unfolding** *W-def* **by** *simp*
have *W-incl*: *W* ⊆ (*T*^^*n*)−−‘*A* **unfolding** *W-def* **by** *simp*

define *V* **where** *V* = (λ*k*. {*x* ∈ (*T*^^*k*)−−‘*W*. *local-time A k x* = *0*})
have *V-meas* [*measurable*]: *V k* ∈ *sets M* **for** *k* **unfolding** *V-def* **by** *simp*
have *a*: (∫ ⁺x ∈ B(n+k). f((T^^k)x) ∂M) ≤ *C* ∗ *emeasure M* (*V k*) **for** *k*
proof −
have *f*((*T*^^*k*)*x*) ∗ *indicator* (*B*(*n+k*)) *x* = *f*((*T*^^*k*)*x*) ∗ *indicator* (*B*(*n+k*) ∩ (*T*^^*k*)−−‘*W*) *x* **for** *x*
proof (*cases*)
assume *f*((*T*^^*k*)*x*) ∗ *indicator* (*B*(*n+k*)) *x* = *0*
then show *?thesis* **by** (*simp add*: *indicator-def*)
next
assume ¬(*f*((*T*^^*k*)*x*) ∗ *indicator* (*B*(*n+k*)) *x* = *0*)
then have *H*: *f*((*T*^^*k*)*x*) ∗ *indicator* (*B*(*n+k*)) *x* ≠ *0* **by** *simp*
then have *inB*: *x* ∈ *B*(*n+k*) **using** *H* **using** *indicator-simps*(*2*) **by** *fastforce*
then have *s*: *x* ∈ *space M* **using** *B-meas*[*of n+k*] *sets.sets-into-space* **by** *blast*
then have *a*: (*T*^^*k*)*x* ∈ *space M* **by** (*metis measurable-space Tn-meas*[*of k*])

have *f*((*T*^^*k*)*x*) > *0* **using** *H* **by** (*simp add*: *le-neq-trans*)
then have ∗: (*T*^^*k*)*x* ∈ {*y* ∈ *space M*. *f y* > *0*} **using** *a* **by** *simp*

have (*T*^^(*n+k*))*x* ∈ *A* **using** *inB B-def first-entrance-set-def* **by** *auto*
then have (*T*^^*n*)((*T*^^*k*)*x*) ∈ *A* **by** (*simp add*: *funpow-add*)
then have (*T*^^*k*)*x* ∈ (*T*^^*n*)−−‘*A* **using** *a* **by** *auto*
then have (*T*^^*k*)*x* ∈ *W* **using** ∗ *W-def* **by** *simp*
then have *x* ∈ (*T*^^*k*)−−‘*W* **using** *s a* **by** *simp*
then have *x* ∈ (*B*(*n+k*) ∩ (*T*^^*k*)−−‘*W*) **using** *inB* **by** *simp*
then show *?thesis* **by** *auto*
qed

then have $*$: ($\int^+ x \in B(n+k).\ f((T^{\frown\frown}k)x)\ \partial M$) = ($\int^+ x \in B(n+k) \cap (T^{\frown\frown}k)--`W.\ f((T^{\frown\frown}k)x)\ \partial M$)
by *simp*
have $B(n+k) \subseteq \{x \in space\ M.\ local\text{-}time\ A\ k\ x = 0\}$
unfolding *local-time-def B-def first-entrance-set-def* **by** *auto*
then have $B(n+k) \cap (T^{\frown\frown}k)--`W \subseteq V\ k$ **unfolding** *V-def* **by** *blast*
then have $f((T^{\frown\frown}k)x) * indicator\ (B(n+k) \cap (T^{\frown\frown}k)--`W)\ x \leq ennreal\ C * indicator\ (V\ k)\ x$ **for** x
using *f-bound* **by** (*auto split*: *split-indicator*)
then have ($\int^+ x \in B(n+k) \cap (T^{\frown\frown}k)--`W.\ f((T^{\frown\frown}k)x)\ \partial M$) $\leq$ ($\int^+ x.\ ennreal\ C * indicator\ (V\ k)\ x\ \partial M$)
by (*simp add*: *nn-integral-mono*)
also have ... = *ennreal C* $*$ *emeasure M* (*V k*) **by** (*simp add*: ‹$0 \leq C$› *nn-integral-cmult*)
finally show ($\int^+ x \in B(n+k).\ f((T^{\frown\frown}k)x)\ \partial M$) $\leq C *$ *emeasure M* (*V k*) **using** $*$ **by** *simp*
qed

have ($\lambda k.$ *emeasure M* (*V k*)) $\longrightarrow 0$ **unfolding** *V-def*
using *local-time-unbounded2*[*OF W-meas*, *OF W-fin*, *OF W-incl*, *of 1*] **by** *auto*
from *ennreal-tendsto-cmult*[*OF - this*, *of C*]
have *t0*: ($\lambda k.\ C *$ *emeasure M* (*V k*)) $\longrightarrow 0$
by *simp*
from *a* **show** ($\lambda k.$ ($\int^+ x \in B(n+k).\ f((T^{\frown\frown}k)x)\ \partial M$)) $\longrightarrow 0$
by (*intro tendsto-sandwich*[*OF - - tendsto-const t0*]) *auto*
qed
have *b*: ($\lambda k.$ ($\sum i<k.(\int^+ x \in D(n+i).\ f((T^{\frown\frown}(i+1))x)\ \partial M)$)) $\longrightarrow$ ($\sum i.\ d\ (i+1)\ (n+i)$) **for** n
proof −
define e **where** $e = (\lambda i.\ d\ (i+1)\ (n+i))$
then have ($\lambda k.$ ($\sum i<k.\ e\ i$)) $\longrightarrow$ ($\sum i.\ e\ i$)
by (*intro summable-LIMSEQ*) *simp*
then show ($\lambda k.$ ($\sum i<k.(\int^+ x \in D(n+i).\ f((T^{\frown\frown}(i+1))x)\ \partial M)$)) $\longrightarrow$ ($\sum i.\ d\ (i+1)\ (n+i)$)
using *e-def d-def* **by** *simp*
qed

have ($\lambda k.$ ($\sum i<k.$ ($\int^+ x \in D(n+i).\ f((T^{\frown\frown}(i+1))x)\ \partial M$)) + ($\int^+ x \in B(n+k).\ f((T^{\frown\frown}k)x)\ \partial M$))
$\longrightarrow$ ($\sum i.\ d\ (i+1)\ (n+i)$) **for** n
using *tendsto-add*[*OF b a*] **by** *simp*
moreover have ($\lambda k.$ ($\sum i<k.$ ($\int^+ x \in D(n+i).\ f((T^{\frown\frown}(i+1))x)\ \partial M$)) + ($\int^+ x \in B(n+k).\ f((T^{\frown\frown}k)x)\ \partial M$))
$\longrightarrow$ ($\int^+ x \in B\ n.\ f\ x\ \partial M$) **for** n **using** $*$ **by** *simp*
ultimately have ($\int^+ x \in B\ n.\ f\ x\ \partial M$) = ($\sum i.\ d\ (i+1)\ (n+i)$) **for** n **using** *LIMSEQ-unique* **by** *blast*
then have ($\sum n.$ ($\int^+ x \in B\ (n+1).\ f\ x\ \partial M$)) = ($\sum n.$ ($\sum i.\ d\ (i+1)\ (n+1+i)$)) **by** *simp*

then have ($\int^+ x \in (\bigcup n.\ (T^{\frown\frown}n){-}{-}`A).\ f\ x\ \partial M) = (\sum n.\ d\ 0\ n) + (\sum n.\ (\sum i.\ d\ (i{+}1)\ (n{+}1{+}i)))$
using *Bdec B0dec* **by** *simp*
then show *?thesis* **using** *induced-dec kac-series-aux* **by** *simp*
qed

We remove the boundedness assumption on f and the finiteness assumption on its support by truncation (both in space and on the values of f).

theorem *induced-function-nn-integral*:
fixes $f{::}'a \Rightarrow ennreal$
assumes *A-meas* [*measurable*]: $A \in sets\ M$
and *f-meas* [*measurable*]: $f \in borel\text{-}measurable\ M$
shows $(\int^+ y.\ induced\text{-}function\ A\ f\ y\ \partial M) = (\int^+ x \in (\bigcup n.\ (T^{\frown\frown}n){-}{-}`A).\ f\ x\ \partial M)$
proof –
obtain $Y{::}nat \Rightarrow\ 'a\ set$ **where** *Y-meas*: $\bigwedge n.\ Y\ n \in sets\ M$ **and** *Y-fin*: $\bigwedge n.\ emeasure\ M\ (Y\ n) \neq \infty$
and *Y-full*: $(\bigcup n.\ Y\ n) = space\ M$ **and** *Y-inc*: *incseq Y*
by (*meson range-subsetD sigma-finite-incseq*)
define F **where** $F = (\lambda(n{::}nat)\ x.\ min\ (f\ x)\ n * indicator\ (Y\ n)\ x)$
have *mes* [*measurable*]: $\bigwedge n.\ (F\ n) \in borel\text{-}measurable\ M$ **unfolding** *F-def* **using** *assms(2) Y-meas* **by** *measurable*
then have *mes-rel* [*measurable*]: $(\lambda x.\ F\ n\ x * indicator\ (\bigcup n.\ (T^{\frown\frown}n){-}{-}`A)\ x) \in borel\text{-}measurable\ M$ **for** n
by *measurable*
have *bound*: $\bigwedge n\ x.\ F\ n\ x \leq ennreal\ n$ **by** (*simp add*: *F-def indicator-def ennreal-of-nat-eq-real-of-nat*)
have $\bigwedge n.\ \{x \in space\ M.\ F\ n\ x > 0\} \subseteq Y\ n$ **unfolding** *F-def* **using** *not-le* **by** *fastforce*
then have *le*: $emeasure\ M\ \{x \in space\ M.\ F\ n\ x > 0\} \leq emeasure\ M\ (Y\ n)$ **for** n **by** (*metis emeasure-mono Y-meas*)
have *fin*: $emeasure\ M\ \{x \in space\ M.\ F\ n\ x > 0\} < \infty$ **for** n
using *Y-fin*[*of n*] *le*[*of n*] **by** (*simp add*: *less-top*)
have $*$: $(\int^+ y.\ induced\text{-}function\ A\ (F\ n)\ y\ \partial M) = (\int^+ x \in (\bigcup n.\ (T^{\frown\frown}n){-}{-}`A).\ (F\ n)\ x\ \partial M)$ **for** n
by (*rule induced-function-nn-integral-aux*[*OF A-meas mes bound - fin*]) *simp*

have *inc-Fx*: $\bigwedge x.\ incseq\ (\lambda n.\ F\ n\ x)$ **unfolding** *F-def incseq-def*
proof (*auto simp add*: *incseq-def*)
fix $x{::}'a$ **and** $m\ n{::}nat$
assume $m \leq n$
then have $min\ (f\ x)\ m \leq min\ (f\ x)\ n$ **using** *linear* **by** *fastforce*
moreover have $(indicator\ (Y\ m)\ x{::}ennreal) \leq (indicator\ (Y\ n)\ x{::}ennreal)$ **using** *Y-inc*
apply (*auto simp add*: *incseq-def*) **using** ‹$m \leq n$› **by** *blast*
ultimately show $min\ (f\ x)\ m * (indicator\ (Y\ m)\ x{::}ennreal) \leq min\ (f\ x)\ n * (indicator\ (Y\ n)\ x{::}ennreal)$
by (*auto split*: *split-indicator*)
qed

then have *incseq* (λ*n*. *F n x* ∗ *indicator* (⋃*n*. (*T*^^*n*)−−‘*A*) *x*) **for** *x*
  by (*auto simp add*: *indicator-def incseq-def*)
 then have *inc-rel*: *incseq* (λ*n x*. *F n x* ∗ *indicator* (⋃*n*. (*T*^^*n*)−−‘*A*) *x*) **by** (*auto simp add*: *incseq-def le-fun-def*)
 then have *a*: (*SUP n*. (∫⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *F n x* ∂*M*))
   = (∫⁺ *x*. (*SUP n*. *F n x* ∗ *indicator* (⋃*n*. (*T*^^*n*)−−‘*A*) *x*) ∂*M*)
  using *nn-integral-monotone-convergence-SUP*[*OF inc-rel*, *OF mes-rel*] **by** *simp*

 have *SUP-Fx*: (*SUP n*. *F n x*) = *f x* **if** *x* ∈ *space M* **for** *x*
 proof −
  obtain *N* **where** *x* ∈ *Y N* **using** *Y-full* ‹*x* ∈ *space M*› **by** *auto*
  have (*SUP n*. *F n x*) = (*SUP n*. *inf* (*f x*) (*of-nat n*))
  proof (*rule SUP-eq*)
   show ∃*j*∈*UNIV*. *F i x* ≤ *inf* (*f x*) (*of-nat j*) **for** *i*
    by (*auto simp*: *F-def intro*!: *exI*[*of - i*] *split*: *split-indicator*)
   show ∃*i*∈*UNIV*. *inf* (*f x*) (*of-nat j*) ≤ *F i x* **for** *j*
    using ‹*x* ∈ *Y N*› ‹*incseq Y*›[*THEN monoD*, *of N max N j*]
    by (*intro bexI*[*of - max N j*])
     (*auto simp*: *F-def subset-eq not-le inf-min intro*: *min.coboundedI2 less-imp-le split*: *split-indicator split-max*)
  qed
  then show *?thesis*
   by (*simp add*: *inf-SUP*[*symmetric*] *ennreal-SUP-of-nat-eq-top*)
 qed
 then have (*SUP n*. *F n x* ∗ *indicator* (⋃*n*. (*T*^^*n*)−−‘*A*) *x*) = *f x* ∗ *indicator* (⋃*n*. (*T*^^*n*)−−‘*A*) *x*
  if *x* ∈ *space M* **for** *x*
  by (*auto simp add*: *indicator-def SUP-Fx that*)
 then have ∗∗: (*SUP n*. (∫⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *F n x* ∂*M*)) = (∫⁺ *x* ∈ (⋃*n*. (*T*^^*n*)−−‘*A*). *f x* ∂*M*)
  by (*simp add*: *a cong*: *nn-integral-cong*)

 have *incseq* (λ*n*. *induced-function A* (*F n*) *x*) **for** *x*
  unfolding *induced-function-def*
  using *incseq-sumI2*[*of* {..<*return-time-function A x*}, *of* λ*i n*. *F n* ((*T*^^*i*)*x*)] *inc-Fx* **by** *simp*
 then have *incseq* (λ*n*. *induced-function A* (*F n*)) **by** (*auto simp add*: *incseq-def le-fun-def*)
 then have *b*: (*SUP n*. (∫⁺ *x*. *induced-function A* (*F n*) *x* ∂*M*)) = (∫⁺ *x*. (*SUP n*. *induced-function A* (*F n*) *x*) ∂*M*)
  by (*rule nn-integral-monotone-convergence-SUP*[*symmetric*]) (*measurable*)

 have (*SUP n*. *induced-function A* (*F n*) *x*) = *induced-function A f x* **if** [*simp*]: *x* ∈ *space M* **for** *x*
 proof −
  have (*SUP n*. (∑ *i* ∈{..<*return-time-function A x*}. *F n* ((*T*^^*i*)*x*)))
   = (∑ *i* ∈ {..<*return-time-function A x*}. *f* ((*T*^^*i*)*x*))
   using *ennreal-SUP-sum*[*OF inc-Fx*, **where** *?I* = {..<*return-time-function A x*}] *SUP-Fx* **by** *simp*

```
    then show (SUP n. induced-function A (F n) x) = induced-function A f x
      by (auto simp add: induced-function-def)
  qed
   then have (SUP n. (∫⁺ x. induced-function A (F n) x ∂M)) = (∫⁺ x. in-
duced-function A f x ∂M)
    by (simp add: b cong: nn-integral-cong)
  then show ?thesis using * ** by simp
qed
```

Taking the constant function equal to 1 in the previous statement, we obtain the usual Kac Formula.

```
theorem kac-formula-nonergodic:
  assumes A-meas [measurable]: A ∈ sets M
  shows (∫⁺y. return-time-function A y ∂M) = emeasure M (⋃n. (T^^n)−−‘A)
proof −
  define f where f = (λ(x::'a). 1::ennreal)
  have ⋀x. induced-function A f x = return-time-function A x
    unfolding induced-function-def f-def by (simp add:)
  then have (∫⁺y. return-time-function A y ∂M) = (∫⁺y. induced-function A f
y ∂M) by auto
  also have ... = (∫⁺ x ∈ (⋃n. (T^^n)−−‘A). f x ∂M)
    by (rule induced-function-nn-integral) (auto simp add: f-def)
  also have ... = emeasure M (⋃n. (T^^n)−−‘A) using f-def by auto
  finally show ?thesis by simp
qed

lemma (in fmpt) return-time-integrable:
  assumes A-meas [measurable]: A ∈ sets M
  shows integrable M (return-time-function A)
  by (rule integrableI-nonneg)
    (auto simp add: kac-formula-nonergodic[OF assms] ennreal-of-nat-eq-real-of-nat[symmetric]
less-top[symmetric])
```

Now, we want to prove the same result but for real-valued integrable function. We first prove the statement for nonnegative functions by reducing to the nonnegative extended reals, and then for general functions by difference.

```
lemma induced-function-integral-aux:
  fixes f::'a ⇒ real
  assumes A-meas [measurable]: A ∈ sets M
      and f-int [measurable]: integrable M f
      and f-pos: ⋀x. f x ≥ 0
  shows integrable M (induced-function A f)
        (∫ y. induced-function A f y ∂M) = (∫ x ∈ (⋃n. (T^^n)−−‘A). f x ∂M)
proof −
  show integrable M (induced-function A f)
  proof (rule integrableI-nonneg)
    show AE x in M. induced-function A f x ≥ 0 unfolding induced-function-def
by (simp add: f-pos sum-nonneg)
```

have $(\int^+ x.\ ennreal\ (induced\text{-}function\ A\ f\ x)\ \partial M) = (\int^+\ x.\ induced\text{-}function\ A\ (\lambda x.\ ennreal(f\ x))\ x\ \partial M)$
unfolding *induced-function-def* **by** (*auto simp*: *f-pos*)
also have ... $= (\int^+\ x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A).\ f\ x\ \partial M)$
by (*rule induced-function-nn-integral*, *auto simp add*: *assms*)
also have ... $\leq (\int^+\ x.\ f\ x\ \partial M)$
using *nn-set-integral-set-mono*[**where** $?A = (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A)$ **and** $?B = UNIV$ **and** $?f = \lambda x.\ ennreal(f\ x)$]
by *auto*
also have ... $< \infty$ **using** *assms* **by** (*auto simp*: *less-top*)
finally show $(\int^+\ x.\ induced\text{-}function\ A\ f\ x\ \partial M) < \infty$ **by** *simp*
qed (*simp*)

have $(\int^+\ x.\ (f\ x * indicator\ (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A)\ x)\ \partial M) = (\int^+\ x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A).\ f\ x\ \partial M)$
by (*auto split*: *split-indicator intro*!: *nn-integral-cong*)
also have ... $= (\int^+\ x.\ induced\text{-}function\ A\ (\lambda x.\ ennreal(f\ x))\ x\ \partial M)$
by (*rule induced-function-nn-integral*[*symmetric*], *auto simp add*: *assms*)
also have ... $= (\int^+ x.\ ennreal\ (induced\text{-}function\ A\ f\ x)\ \partial M)$ **unfolding** *induced-function-def* **by** (*auto simp*: *f-pos*)
finally have $*$: $(\int^+\ x.\ (f\ x * indicator\ (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A)\ x)\ \partial M) = (\int^+ x.\ ennreal\ (induced\text{-}function\ A\ f\ x)\ \partial M)$
by *simp*

have $(\int\ x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A).\ f\ x\ \partial M) = (\int\ x.\ f\ x * indicator\ (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A)\ x\ \partial M)$
by (*simp add*: *mult.commute set-lebesgue-integral-def*)
also have ... $= enn2real\ (\int^+\ x.\ (f\ x * indicator\ (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A)\ x)\ \partial M)$
by (*rule integral-eq-nn-integral*, *auto simp add*: *f-pos*)
also have ... $= enn2real\ (\int^+ x.\ ennreal\ (induced\text{-}function\ A\ f\ x)\ \partial M)$ **using** $*$ **by** *simp*
also have ... $= (\int\ x.\ induced\text{-}function\ A\ f\ x\ \partial M)$
apply (*rule integral-eq-nn-integral*[*symmetric*])
unfolding *induced-function-def* **by** (*auto simp add*: *f-pos sum-nonneg*)
finally show $(\int\ x.\ induced\text{-}function\ A\ f\ x\ \partial M) = (\int\ x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A).\ f\ x\ \partial M)$
by *simp*
qed

Here is the general version of Kac's Formula (for a general induced function, starting from a real-valued integrable function).

theorem *induced-function-integral-nonergodic*:
fixes $f{::}'a \Rightarrow real$
assumes [*measurable*]: $A \in sets\ M$ *integrable M f*
shows *integrable M* (*induced-function A f*)
$(\int y.\ induced\text{-}function\ A\ f\ y\ \partial M) = (\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A).\ f\ x\ \partial M)$
proof −
have *U-meas* [*measurable*]: $(\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--`A) \in sets\ M$ **by** *measurable*
define g **where** $g = (\lambda x.\ max\ (f\ x)\ 0)$

have *g-int* [*measurable*]: *integrable M g* **unfolding** *g-def* **using** *assms* **by** *auto*
then have *g-int2*: *integrable M* (*induced-function A g*)
using *induced-function-integral-aux*(*1*) *g-def* **by** *auto*
define *h* **where** $h = (\lambda x.\ max\ (-f\ x)\ 0)$
have *h-int* [*measurable*]: *integrable M h* **unfolding** *h-def* **using** *assms* **by** *auto*
then have *h-int2*: *integrable M* (*induced-function A h*)
using *induced-function-integral-aux*(*1*) *h-def* **by** *auto*
have *D1*: $f = (\lambda x.\ g\ x - h\ x)$ **unfolding** *g-def h-def* **by** *auto*
have *D2*: *induced-function A f* = (λx. *induced-function A g x* − *induced-function A h x*)
unfolding *induced-function-def* **using** *D1* **by** (*simp add*: *sum-subtractf*)
then show *integrable M* (*induced-function A f*) **using** *g-int2 h-int2* **by** *auto*

have ($\int$ *x*. *induced-function A f x* ∂M) = ($\int$ *x*. *induced-function A g x* − *induced-function A h x* ∂M)
using *D2* **by** *simp*
also have ... = ($\int$ *x*. *induced-function A g x* ∂M) − ($\int$ *x*. *induced-function A h x* ∂M)
using *g-int2 h-int2* **by** *auto*
also have ... = ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ g\ x\ \partial M$) − ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ h\ x\ \partial M$)
using *induced-function-integral-aux*(*2*) *g-def h-def g-int h-int* **by** *auto*
also have ... = ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ (g\ x - h\ x)\ \partial M$)
apply (*rule set-integral-diff*(*2*)[*symmetric*])
unfolding *set-integrable-def*
using *g-int h-int integrable-mult-indicator*[*OF U-meas*] **by** *blast*+
also have ... = ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ f\ x\ \partial M$)
using *D1* **by** *simp*
finally show ($\int$ *x*. *induced-function A f x* ∂M) = ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ f\ x\ \partial M$) **by** *simp*
qed

We can reformulate the previous statement in terms of induced measure.

lemma *induced-function-integral-restr-nonergodic*:
fixes $f{::}'a \Rightarrow real$
assumes [*measurable*]: $A \in$ *sets M integrable M f*
shows *integrable* (*restrict-space M A*) (*induced-function A f*)
($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int x \in (\bigcup n.\ (T\widehat{\ }\widehat{\ }n)--\text{`}A).\ f\ x\ \partial M$)
proof −
have [*measurable*]: *integrable M* (*induced-function A f*) **by** (*rule induced-function-integral-nonergodic*(*1*)[*OF assms*])
then show *integrable* (*restrict-space M A*) (*induced-function A f*)
by (*metis assms*(*1*) *integrable-mult-indicator integrable-restrict-space sets.Int-space-eq2*)
have ($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int y \in A$. *induced-function A f y* ∂M)
by (*simp add*: *integral-restrict-space set-lebesgue-integral-def*)
also have ... = ($\int$ *y*. *induced-function A f y* ∂M)
unfolding *real-scaleR-def set-lebesgue-integral-def*

proof (*rule Bochner-Integration.integral-cong* [*OF refl*])
have *induced-function A f y* = *0* **if** $y \notin A$ **for** *y* **unfolding** *induced-function-def*
using *that return-time0*[*of A*] *recurrent-subset-incl*(*1*)[*of A*] *return-time-function-def*
by *auto*
then show $\bigwedge x$. *indicator A x* ∗ *induced-function A f x* = *induced-function A f x*
unfolding *indicator-def* **by** *auto*
qed
also have ... = ($\int$ *x* ∈ ($\bigcup$*n*. (*T*^^*n*)−−'*A*). *f x* ∂*M*)
by (*rule induced-function-integral-nonergodic*(*2*)[*OF assms*])
finally show ($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int$ *x* ∈ ($\bigcup$*n*. (*T*^^*n*)−−'*A*). *f x* ∂*M*)
by *simp*
qed

end

end

6 The invariant sigma-algebra, Birkhoff theorem

theory *Invariants*
imports *Recurrence HOL*−*Probability.Conditional-Expectation*
begin

6.1 The sigma-algebra of invariant subsets

The invariant sigma-algebra of a qmpt is made of those sets that are invariant under the dynamics. When the transformation is ergodic, it is made of sets of zero or full measure. In general, the Birkhoff theorem is expressed in terms of the conditional expectation of an integrable function with respect to the invariant sigma-algebra.

context *qmpt* **begin**

We define the invariant sigma-algebra, as the sigma algebra of sets which are invariant under the dynamics, i.e., they coincide with their preimage under T.

definition *Invariants* **where** *Invariants* = *sigma* (*space M*) {*A* ∈ *sets M*. *T*−'*A* ∩ *space M* = *A*}

For this definition to make sense, we need to check that it really defines a sigma-algebra: otherwise, the `sigma` operation would make garbage out of it. This is the content of the next lemma.

lemma *Invariants-sets*: *sets Invariants* = {*A* ∈ *sets M*. *T*−'*A* ∩ *space M* = *A*}
proof −
have *sigma-algebra* (*space M*) {*A* ∈ *sets M*. *T*−'*A* ∩ *space M* = *A*}
proof −

define *I* **where** $I = \{A.\ T- `A \cap space\ M = A\}$
have *i*: $\bigwedge A.\ A \in I \Longrightarrow A \subseteq space\ M$ **unfolding** *I-def* **by** *auto*
have *algebra* (*space M*) *I*
proof (*subst algebra-iff-Un*)
have *a*: $I \subseteq Pow\ (space\ M)$ **using** *i* **by** *auto*
have *b*: $\{\} \in I$ **unfolding** *I-def* **by** *auto*
{
fix *A* **assume** $*$: $A \in I$
then have $T- `(space\ M - A) = T- `(space\ M) - T- `A$ **by** *auto*
then have $T- `(space\ M - A) \cap space\ M = T- `(space\ M) \cap (space\ M) - T- `A \cap (space\ M)$ **by** *auto*
also have ... $= space\ M - A$ **using** $*$ *I-def* **by** (*simp add*: *inf-absorb2 subsetI*)
finally have $space\ M - A \in I$ **unfolding** *I-def* **by** *simp*
}
then have *c*: $(\forall a \in I.\ space\ M - a \in I)$ **by** *simp*
have *d*: $(\forall a \in I.\ \forall b \in I.\ a \cup b \in I)$ **unfolding** *I-def* **by** *auto*
show $I \subseteq Pow\ (space\ M) \wedge \{\} \in I \wedge (\forall a \in I.\ space\ M - a \in I) \wedge (\forall a \in I.\ \forall b \in I.\ a \cup b \in I)$
using *a b c d* **by** *blast*
qed
moreover have $(\forall F.\ range\ F \subseteq I \longrightarrow (\bigcup i::nat.\ F\ i) \in I)$ **unfolding** *I-def* **by** *auto*
ultimately have *sigma-algebra* (*space M*) *I* **using** *sigma-algebra-iff* **by** *auto*
moreover have *sigma-algebra* (*space M*) (*sets M*) **using** *measure-space measure-space-def* **by** *auto*
ultimately have *sigma-algebra* (*space M*) $(I \cap (sets\ M))$ **using** *sigma-algebra-intersection* **by** *auto*
moreover have $I \cap sets\ M = \{A \in sets\ M.\ T- `A \cap space\ M = A\}$ **unfolding** *I-def* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed
then show *?thesis* **unfolding** *Invariants-def* **using** *sigma-algebra.sets-measure-of-eq* **by** *blast*
qed

By definition, the invariant subalgebra is a subalgebra of the original algebra. This is expressed in the following lemmas.

lemma *Invariants-is-subalg*: *subalgebra M Invariants*
unfolding *subalgebra-def*
using *Invariants-sets Invariants-def* **by** (*simp add*: *space-measure-of-conv*)

lemma *Invariants-in-sets*:
assumes $A \in sets\ Invariants$
shows $A \in sets\ M$
using *Invariants-sets assms* **by** *blast*

lemma *Invariants-measurable-func*:
assumes $f \in measurable\ Invariants\ N$

shows $f \in$ *measurable M N*
using *Invariants-is-subalg measurable-from-subalg assms* **by** *auto*

We give several trivial characterizations of invariant sets or functions.

lemma *Invariants-vrestr*:
assumes $A \in$ *sets Invariants*
shows $T--`A = A$
using *assms Invariants-sets Invariants-in-sets*[*OF assms*] **by** *auto*

lemma *Invariants-points*:
assumes $A \in$ *sets Invariants* $x \in A$
shows $T\ x \in A$
using *assms Invariants-sets* **by** *auto*

lemma *Invariants-func-is-invariant*:
fixes f::- $\Rightarrow$ $'b$::*t2-space*
assumes $f \in$ *borel-measurable Invariants* $x \in$ *space M*
shows $f\ (T\ x) = f\ x$
proof −
have $\{f\ x\} \in$ *sets borel* **by** *simp*
then have $f-`(\{f\ x\}) \cap$ *space M* $\in$ *Invariants* **using** *assms*(*1*)
by (*metis* (*no-types*, *lifting*) *Invariants-def measurable-sets space-measure-of-conv*)
moreover have $x \in f-`(\{f\ x\}) \cap$ *space M* **using** *assms*(*2*) **by** *blast*
ultimately have $T\ x \in f-`(\{f\ x\}) \cap$ *space M* **by** (*rule Invariants-points*)
then show *?thesis* **by** *simp*
qed

lemma *Invariants-func-is-invariant-n*:
fixes f::- $\Rightarrow$ $'b$::*t2-space*
assumes $f \in$ *borel-measurable Invariants* $x \in$ *space M*
shows $f\ ((T\widehat{\ }\widehat{\ }n)\ x) = f\ x$
by (*induction n*, *auto simp add*: *assms Invariants-func-is-invariant*)

lemma *Invariants-func-charac*:
assumes [*measurable*]: $f \in$ *measurable M N*
and $\bigwedge x.\ x \in$ *space M* $\Longrightarrow f(T\ x) = f\ x$
shows $f \in$ *measurable Invariants N*
proof (*rule measurableI*)
fix A **assume** $A \in$ *sets N*
have *space Invariants = space M* **using** *Invariants-is-subalg subalgebra-def* **by** *force*
show $f -`\ A \cap$ *space Invariants* $\in$ *sets Invariants*
apply (*subst Invariants-sets*)
apply (*auto simp add*: *assms* ‹$A \in$ *sets N*› ‹*space Invariants = space M*›)
using ‹$A \in$ *sets N*› *assms*(*1*) *measurable-sets* **by** *blast*
next
fix x **assume** $x \in$ *space Invariants*
have *space Invariants = space M* **using** *Invariants-is-subalg subalgebra-def* **by** *force*

then show *f x* ∈ *space N* **using** *assms*(*1*) ‹*x* ∈ *space Invariants*› **by** (*metis measurable-space*)
qed

lemma *birkhoff-sum-of-invariants*:
fixes *f*:: - ⇒ *real*
assumes *f* ∈ *borel-measurable Invariants x* ∈ *space M*
shows *birkhoff-sum f n x* = *n* ∗ *f x*
unfolding *birkhoff-sum-def* **using** *Invariants-func-is-invariant-n*[*OF assms*] **by** *auto*

There are two possible definitions of the invariant sigma-algebra, competing in the literature: one could also use the sets such that $T^{-1}A$ coincides with A up to a measure 0 set. It turns out that this is equivalent to being invariant (in our sense) up to a measure 0 set. Therefore, for all interesting purposes, the two definitions would give the same results.

For the proof, we start from an almost invariant set, and build a genuinely invariant set that coincides with it by adding or throwing away null parts.

proposition *Invariants-quasi-Invariants-sets*:
assumes [*measurable*]: *A* ∈ *sets M*
shows (∃ *B* ∈ *sets Invariants*. *A* Δ *B* ∈ *null-sets M*) ⟷ (*T*−−‘*A* Δ *A* ∈ *null-sets M*)
proof
assume ∃ *B* ∈ *sets Invariants*. *A* Δ *B* ∈ *null-sets M*
then obtain *B* **where** *B* ∈ *sets Invariants A* Δ *B* ∈ *null-sets M* **by** *auto*
then have [*measurable*]: *B* ∈ *sets M* **using** *Invariants-in-sets* **by** *simp*

have *B* = *T*−−‘ *B* **using** *Invariants-vrestr* ‹*B* ∈ *sets Invariants*› **by** *simp*
then have *T*−−‘*A* Δ *B* = *T*−−‘(*A* Δ *B*) **by** *simp*
moreover have *T*−−‘(*A* Δ *B*) ∈ *null-sets M*
by (*rule T-quasi-preserves-null2*(*1*)[*OF* ‹*A* Δ *B* ∈ *null-sets M*›])
ultimately have *T*−−‘*A* Δ *B* ∈ *null-sets M* **by** *simp*
then show *T*−−‘*A* Δ *A* ∈ *null-sets M*
by (*rule null-sym-diff-transitive*) (*auto simp add*: ‹*A* Δ *B* ∈ *null-sets M*› *Un-commute*)
next
assume *H*: *T* −−‘ *A* Δ *A* ∈ *null-sets M*
have [*measurable*]: ⋀*n*. (*T*^^*n*)−−‘*A* ∈ *sets M* **by** *simp*
{
fix *K* **assume** [*measurable*]: *K* ∈ *sets M* **and** *T*−−‘*K* Δ *K* ∈ *null-sets M*
fix *n*::*nat*
have (*T*^^*n*)−−‘*K* Δ *K* ∈ *null-sets M*
proof (*induction n*)
case *0*
have (*T*^^*0*)−−‘ *K* = *K* **using** *T-vrestr-0* **by** *simp*
then show *?case* **using** *Diff-cancel sup.idem* **by** (*metis null-sets.empty-sets*)
next
case (*Suc n*)

have *T−−‘((T^^n)−−‘K Δ K) ∈ null-sets M*
using *Suc.IH T-quasi-preserves-null(1)[of ((T^^n)−−‘K Δ K)]* **by** *simp*
then have ∗: *(T^^(Suc n))−−‘K Δ T−−‘K ∈ null-sets M* **using** *T-vrestr-composed(2)[OF ‹K ∈ sets M›]* **by** *simp*
then show *?case*
by (*rule null-sym-diff-transitive*, *simp add*: ‹*T−−‘K Δ K ∈ null-sets M*› ‹*K ∈ sets M*›, *measurable*)
qed
} **note** ∗ = *this*

define *C* **where** *C* = (⋂*n*. (*T^^n*)−−‘*A*)
have [*measurable*]: *C ∈ sets M* **unfolding** *C-def* **by** *simp*
have *C Δ A* ⊆ (⋃*n*. (*T^^n*)−−‘*A Δ A*) **unfolding** *C-def* **by** *auto*
moreover have (⋃*n*. (*T^^n*)−−‘*A Δ A*) ∈ *null-sets M*
using ∗ *null-sets-UN assms* ‹*T −−‘ A Δ A ∈ null-sets M*› **by** *auto*
ultimately have *CA*: *C Δ A ∈ null-sets M* **by** (*meson* ‹*C ∈ sets M*› *assms sets.Diff sets.Un null-sets-subset*)
then have *T−−‘(C Δ A) ∈ null-sets M* **by** (*rule T-quasi-preserves-null2(1)*)
then have *T−−‘C Δ T−−‘A ∈ null-sets M* **by** *simp*
then have *T−−‘C Δ A ∈ null-sets M*
by (*rule null-sym-diff-transitive*, *auto simp add*: *H*)
then have *TCC*: *T−−‘C Δ C ∈ null-sets M*
apply (*rule null-sym-diff-transitive*) **using** *CA* **by** (*auto simp add*: *Un-commute*)

have *C* ⊆ (⋂*n*∈{*1..*}. (*T^^n*)−−‘*A*) **unfolding** *C-def* **by** *auto*
moreover have *T−−‘C* = (⋂*n*∈{*1..*}. (*T^^n*)−−‘*A*)
using *T-vrestr-composed(2)[OF assms]* **by** (*simp add*: *C-def atLeast-Suc-greaterThan greaterThan-0*)
ultimately have *C ⊆ T−−‘C* **by** *blast*
then have (*T^^0*)−−‘*C* ⊆ (*T^^1*)−−‘*C* **using** *T-vrestr-0* **by** *auto*
moreover have (*T^^1*)−−‘*C* ⊆ (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) **by** *auto*
ultimately have (*T^^0*)−−‘*C* ⊆ (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) **by** *auto*
then have (*T^^0*)−−‘*C* ∪ (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) = (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) **by** *auto*
moreover have (⋃*n*. (*T^^n*)−−‘*C*) = (*T^^0*)−−‘*C* ∪ (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) **by** (*rule union-insert-0*)
ultimately have *C2*: (⋃*n*. (*T^^n*)−−‘*C*) = (⋃*n*∈{*1..*}. (*T^^n*)−−‘*C*) **by** *simp*

define *B* **where** *B* = (⋃*n*. (*T^^n*)−−‘*C*)
have [*measurable*]: *B ∈ sets M* **unfolding** *B-def* **by** *simp*
have *B Δ C* ⊆ (⋃*n*. (*T^^n*)−−‘*C Δ C*) **unfolding** *B-def* **by** *auto*
moreover have (⋃*n*. (*T^^n*)−−‘*C Δ C*) ∈ *null-sets M*
using ∗ *null-sets-UN assms TCC* **by** *auto*
ultimately have *B Δ C ∈ null-sets M* **by** (*meson* ‹*B ∈ sets M*› ‹*C ∈ sets M*› *assms sets.Diff sets.Un null-sets-subset*)
then have *B Δ A ∈ null-sets M*
by (*rule null-sym-diff-transitive*, *auto simp add*: *CA*)
then have *a*: *A Δ B ∈ null-sets M* **by** (*simp add*: *Un-commute*)

have $T--`B = (\bigcup n\in\{1..\}.\ (T\widehat{\ }\widehat{\ }n)--`C)$
using *T-vrestr-composed(2)[OF ‹C ∈ sets M›]* **by** (*simp add*: *B-def atLeast-Suc-greaterThan greaterThan-0*)
then have $T--`B = B$ **unfolding** *B-def* **using** *C2* **by** *simp*
then have $B \in$ *sets Invariants* **using** *Invariants-sets vimage-restr-def* **by** *auto*

then show $\exists B \in$ *sets Invariants*. $A\ \Delta\ B \in$ *null-sets M* **using** *a* **by** *blast*
qed

In a conservative setting, it is enough to be included in its image or its preimage to be almost invariant: otherwise, since the difference set has disjoint preimages, and is therefore null by conservativity.

lemma (**in** *conservative*) *preimage-included-then-almost-invariant*:
assumes [*measurable*]: $A \in$ *sets M* **and** $T--`A \subseteq A$
shows $A\ \Delta\ (T--`A) \in$ *null-sets M*
proof –
define *B* **where** $B = A - T--`A$
then have [*measurable*]: $B \in$ *sets M* **by** *simp*
have $(T\widehat{\ }\widehat{\ }(Suc\ n))--`A \subseteq (T\widehat{\ }\widehat{\ }n)--`A$ **for** *n* **using** *T-vrestr-composed(3)[OF assms(1)] vrestr-inclusion[OF assms(2)]* **by** *auto*
then have *disjoint-family* $(\lambda n.\ (T\widehat{\ }\widehat{\ }n)--`A - (T\widehat{\ }\widehat{\ }(Suc\ n))--`A)$ **by** (*rule disjoint-family-Suc2*[**where** $?A = \lambda n.\ (T\widehat{\ }\widehat{\ }n)--`A$])
moreover have $(T\widehat{\ }\widehat{\ }n)--`A - (T\widehat{\ }\widehat{\ }(Suc\ n))--`A = (T\widehat{\ }\widehat{\ }n)--`B$ **for** *n* **unfolding** *B-def Suc-eq-plus1* **using** *T-vrestr-composed(3)[OF assms(1)]* **by** *auto*
ultimately have *disjoint-family* $(\lambda n.\ (T\widehat{\ }\widehat{\ }n)--`\ B)$ **by** *simp*
then have $\bigwedge n.\ n \neq 0 \Longrightarrow ((T\widehat{\ }\widehat{\ }n)--`B) \cap B = \{\}$ **unfolding** *disjoint-family-on-def* **by** (*metis UNIV-I T-vrestr-0(1)[OF ‹B ∈ sets M›]*)
then have $\bigwedge n.\ n > 0 \Longrightarrow (T\widehat{\ }\widehat{\ }n)-`B \cap B = \{\}$ **unfolding** *vimage-restr-def* **by** (*simp add*: *Int-assoc*)
then have $B \in$ *null-sets M* **using** *disjoint-then-null[OF ‹B ∈ sets M›] Int-commute* **by** *auto*
then show *?thesis* **unfolding** *B-def* **using** *assms(2)* **by** (*simp add*: *Diff-mono Un-absorb2*)
qed

lemma (**in** *conservative*) *preimage-includes-then-almost-invariant*:
assumes [*measurable*]: $A \in$ *sets M* **and** $A \subseteq T--`A$
shows $A\ \Delta\ (T--`A) \in$ *null-sets M*
proof –
define *B* **where** $B = T--`A - A$
then have [*measurable*]: $B \in$ *sets M* **by** *simp*
have $\bigwedge n.\ (T\widehat{\ }\widehat{\ }(Suc\ n))--`A \supseteq (T\widehat{\ }\widehat{\ }n)--`A$ **using** *T-vrestr-composed(3)[OF assms(1)] vrestr-inclusion[OF assms(2)]* **by** *auto*
then have *disjoint-family* $(\lambda n.\ (T\widehat{\ }\widehat{\ }(Suc\ n))--`A - (T\widehat{\ }\widehat{\ }n)--`A)$ **by** (*rule disjoint-family-Suc*[**where** $?A = \lambda n.\ (T\widehat{\ }\widehat{\ }n)--`A$])
moreover have $\bigwedge n.\ (T\widehat{\ }\widehat{\ }(Suc\ n))--`A - (T\widehat{\ }\widehat{\ }n)--`A = (T\widehat{\ }\widehat{\ }n)--`B$ **unfolding** *B-def Suc-eq-plus1* **using** *T-vrestr-composed(3)[OF assms(1)]* **by** *auto*
ultimately have *disjoint-family* $(\lambda n.\ (T\widehat{\ }\widehat{\ }n)--`\ B)$ **by** *simp*

then have $\bigwedge n.\ n \neq 0 \Longrightarrow ((T\widehat{\ }\widehat{\ }n)--`B) \cap B = \{\}$ **unfolding** *disjoint-family-on-def* **by** (*metis UNIV-I T-vrestr-0*(*1*)[*OF* ‹*B* ∈ *sets M*›])
then have $\bigwedge n.\ n > 0 \Longrightarrow (T\widehat{\ }\widehat{\ }n)-`B \cap B = \{\}$ **unfolding** *vimage-restr-def* **by** (*simp add*: *Int-assoc*)
then have *B* ∈ *null-sets M* **using** *disjoint-then-null*[*OF* ‹*B* ∈ *sets M*›] *Int-commute* **by** *auto*
then show *?thesis* **unfolding** *B-def* **using** *assms*(*2*) **by** (*simp add*: *Diff-mono Un-absorb1*)
qed

The above properties for sets are also true for functions: if f and $f \circ T$ coincide almost everywhere, i.e., f is almost invariant, then f coincides almost everywhere with a true invariant function.

The idea of the proof is straightforward: throw away the orbits on which f is not really invariant (say this is the complement of the good set), and replace it by 0 there. However, this does not work directly: the good set is not invariant, some points may have a non-constant value of f on their orbit but reach the good set eventually. One can however define g to be equal to the eventual value of f along the orbit, if the orbit reaches the good set, and 0 elsewhere.

proposition *Invariants-quasi-Invariants-functions*:
fixes *f*::- ⇒ *'b*::{*second-countable-topology*, *t2-space*}
assumes *f-meas* [*measurable*]: *f* ∈ *borel-measurable M*
shows (∃ *g* ∈ *borel-measurable Invariants*. *AE x in M*. *f x* = *g x*) ⟷ (*AE x in M*. *f*(*T x*) = *f x*)
proof
assume ∃ *g*∈*borel-measurable Invariants*. *AE x in M*. *f x* = *g x*
then obtain *g* **where** *g*:*g*∈*borel-measurable Invariants AE x in M*. *f x* = *g x* **by** *blast*
then have [*measurable*]: *g* ∈ *borel-measurable M* **using** *Invariants-measurable-func* **by** *auto*
define *A* **where** *A* = {*x* ∈ *space M*. *f x* = *g x*}
have [*measurable*]: *A* ∈ *sets M* **unfolding** *A-def* **by** *simp*
define *B* **where** *B* = *space M* − *A*
have [*measurable*]: *B* ∈ *sets M* **unfolding** *B-def* **by** *simp*
moreover have *AE x in M*. *x* ∉ *B* **unfolding** *B-def A-def* **using** *g*(*2*) **by** *auto*
ultimately have *B* ∈ *null-sets M* **using** *AE-iff-null-sets* **by** *blast*
then have *T*−−`*B* ∈ *null-sets M* **by** (*rule T-quasi-preserves-null2*(*1*))
then have *B* ∪ *T*−−`*B* ∈ *null-sets M* **using** ‹*B* ∈ *null-sets M*› **by** *auto*
then have *AE x in M*. *x* ∉ (*B* ∪ *T*−−`*B*) **using** *AE-iff-null-sets null-setsD2* **by** *blast*
then have *i*: *AE x in M*. *x* ∈ *space M* − (*B* ∪ *T*−−`*B*) **by** *auto*
{
fix *x* **assume** ∗: *x* ∈ *space M* − (*B* ∪ *T*−−`*B*)
then have *x* ∈ *A* **unfolding** *B-def* **by** *blast*
then have *f x* = *g x* **unfolding** *A-def* **by** *blast*
have *T x* ∈ *A* **using** ∗ *B-def* **by** *auto*
then have *f*(*T x*) = *g*(*T x*) **unfolding** *A-def* **by** *blast*

moreover have *g*(*T x*) = *g x*
apply (*rule Invariants-func-is-invariant*) **using** ∗ **by** (*auto simp add*: *assms* ‹*g*∈*borel-measurable Invariants*›)
ultimately have *f*(*T x*) = *f x* **using** ‹*f x* = *g x*› **by** *simp*
}
then show *AE x in M*. *f*(*T x*) = *f x* **using** *i* **by** *auto*
next
assume ∗: *AE x in M*. *f* (*T x*) = *f x*

`good_set` is the set of points for which f is constant on their orbit. Here, we define $g = f$. If a point ever enters `good_set`, then we take g to be the value of f there. Otherwise, g takes an arbitrary value, say y_0.

define *good-set* **where** *good-set* = {*x* ∈ *space M*. ∀ *n*. *f*((*T*^^(*Suc n*)) *x*) = *f*((*T*^^*n*) *x*)}
define *good-time* **where** *good-time* = (λ*x*. *Inf* {*n*. (*T*^^*n*) *x* ∈ *good-set*})
have *AE x in M*. *x* ∈ *good-set* **using** *T-AE-iterates*[*OF* ∗] **by** (*simp add*: *good-set-def*)
have [*measurable*]: *good-set* ∈ *sets M* **unfolding** *good-set-def* **by** *auto*
obtain *y0*::′*b* **where** *True* **by** *auto*
define *g* **where** *g* = (λ*x*. *if* (∃ *n*. (*T*^^*n*) *x* ∈ *good-set*) *then f*((*T*^^(*good-time x*)) *x*) *else y0*)
have [*measurable*]: *good-time* ∈ *measurable M* (*count-space UNIV*) **unfolding** *good-time-def* **by** *measurable*
have [*measurable*]: *g* ∈ *borel-measurable M* **unfolding** *g-def* **by** *measurable*

have *f x* = *g x* **if** *x* ∈ *good-set* **for** *x*
proof −
have *a*: *0* ∈ {*n*. (*T*^^*n*) *x* ∈ *good-set*} **using** *that* **by** *simp*
have *good-time x* = *0*
unfolding *good-time-def* **apply** (*intro cInf-eq-non-empty*) **using** *a* **by** *blast*+
moreover have {*n*. (*T*^^*n*) *x* ∈ *good-set*} ≠ {} **using** *a* **by** *blast*
ultimately show *f x* = *g x* **unfolding** *g-def* **by** *auto*
qed
then have *AE x in M*. *f x* = *g x* **using** ‹*AE x in M*. *x* ∈ *good-set*› **by** *auto*

have ∗: *f*((*T*^^(*Suc 0*)) *x*) = *f*((*T*^^*0*) *x*) **if** *x* ∈ *good-set* **for** *x*
using *that* **unfolding** *good-set-def* **by** *blast*
have *good-1*: *T x* ∈ *good-set* ∧ *f*(*T x*) = *f x* **if** *x* ∈ *good-set* **for** *x*
using ∗[*OF that*] *that* **unfolding** *good-set-def* **apply** (*auto*)
unfolding *T-Tn-T-compose* **by** *blast*
then have *good-k*: ⋀*x*. *x* ∈ *good-set* ⟹ (*T*^^*k*) *x* ∈ *good-set* ∧ *f*((*T*^^*k*) *x*) = *f x* **for** *k*
by (*induction k*, *auto*)

have *g*(*T x*) = *g x* **if** *x* ∈ *space M* **for** *x*
proof (*cases*)
assume ∗: ∃ *n*. (*T*^^*n*) (*T x*) ∈ *good-set*
define *n* **where** *n* = *Inf* {*n*. (*T*^^*n*) (*T x*) ∈ *good-set*}
have (*T*^^*n*)(*T x*) ∈ *good-set* **using** ∗ *Inf-nat-def1* **by** (*metis empty-iff mem-Collect-eq*

n-def)
then have *a*: $(T^{\frown\frown}(n+1))$ *x* ∈ *good-set* **by** (*metis Suc-eq-plus1 comp-eq-dest-lhs funpow.simps*(*2*) *funpow-swap1*)
then have ∗∗: ∃ *m*. $(T^{\frown\frown}m)$ *x* ∈ *good-set* **by** *blast*
define *m* **where** *m* = *Inf* {*m*. $(T^{\frown\frown}m)$ *x* ∈ *good-set*}
then have $(T^{\frown\frown}m)$ *x* ∈ *good-set* **using** ∗∗ *Inf-nat-def1* **by** (*metis empty-iff mem-Collect-eq*)
have *n+1* ∈ {*m*. $(T^{\frown\frown}m)$ *x* ∈ *good-set*} **using** *a* **by** *simp*
then have *m* ≤ *n+1* **using** *m-def* **by** (*simp add*: *Inf-nat-def Least-le*)
then obtain *k* **where** *n+1* = *m* + *k* **using** *le-iff-add* **by** *blast*
have *g x* = *f*($(T^{\frown\frown}m)$ *x*) **unfolding** *g-def good-time-def* **using** ∗∗ *m-def* **by** *simp*
also have ... = *f*($(T^{\frown\frown}k)$ ($(T^{\frown\frown}m)$ *x*)) **using** ‹$(T^{\frown\frown}m)$ *x* ∈ *good-set*› *good-k* **by** *simp*
also have ... = *f*($(T^{\frown\frown}(n+1))$*x*) **using** ‹*n+1* = *m* + *k*›[*symmetric*] *funpow-add* **by** (*metis add.commute comp-apply*)
also have ... = *f*($(T^{\frown\frown}n)$ (*T x*)) **using** *funpow-Suc-right* **by** (*metis Suc-eq-plus1 comp-apply*)
also have ... = *g*(*T x*) **unfolding** *g-def good-time-def* **using** ∗ *n-def* **by** *simp*
finally show *g*(*T x*) = *g x* **by** *simp*
next
assume ∗: ¬(∃ *n*. $(T^{\frown\frown}n)$ (*T x*) ∈ *good-set*)
then have *g*(*T x*) = *y0* **unfolding** *g-def* **by** *simp*
have ∗∗: ¬(∃ *n*. $(T^{\frown\frown}(Suc\ n))$ *x* ∈ *good-set*) **using** *funpow-Suc-right* ∗ **by** (*metis comp-apply*)
have *T x* ∉ *good-set* **using** *good-k* ∗ **by** *blast*
then have *x* ∉ *good-set* **using** *good-1* **by** *auto*
then have ¬(∃ *n*. $(T^{\frown\frown}n)$ *x* ∈ *good-set*) **using** ∗∗ **using** *good-1* **by** *fastforce*
then have *g x* = *y0* **unfolding** *g-def* **by** *simp*
then show *g*(*T x*) = *g x* **using** ‹*g*(*T x*) = *y0*› **by** *simp*
qed
then have *g* ∈ *borel-measurable Invariants* **by** (*rule Invariants-func-charac*[*OF* ‹*g* ∈ *borel-measurable M*›])
then show ∃ *g*∈*borel-measurable Invariants*. *AE x in M*. *f x* = *g x* **using** ‹*AE x in M*. *f x* = *g x*› **by** *blast*
qed

In a conservative setting, it suffices to have an almost everywhere inequality to get an almost everywhere equality, as the set where there is strict inequality has 0 measure as its iterates are disjoint, by conservativity.

proposition (**in** *conservative*) *AE-decreasing-then-invariant*:
fixes *f*::- ⇒ *'b*::{*linorder-topology*, *second-countable-topology*}
assumes *AE x in M*. *f*(*T x*) ≤ *f x*
and [*measurable*]: *f* ∈ *borel-measurable M*
shows *AE x in M*. *f*(*T x*) = *f x*
proof −
obtain *D*::*'b set* **where** *D*: *countable D* (∀ *x y*. *x* < *y* ⟶ (∃ *d* ∈ *D*. *x* ≤ *d* ∧ *d* < *y*))
using *countable-separating-set-linorder2* **by** *blast*

define *A* **where** *A* = {*x* ∈ *space M*. *f*(*T x*) ≤ *f x*}
then have [*measurable*]: *A* ∈ *sets M* **by** *simp*
define *B* **where** *B* = {*x* ∈ *space M*. ∀ *n*. *f*((*T*^^(*n*+*1*)) *x*) ≤ *f*((*T*^^*n*)*x*)}
then have [*measurable*]: *B* ∈ *sets M* **by** *simp*

have *space M* − *A* ∈ *null-sets M* **unfolding** *A-def* **using** *assms* **by** (*simp add*: *assms*(*1*) *AE-iff-null-sets*)
then have (⋃ *n*. (*T*^^*n*)−−‘(*space M* − *A*)) ∈ *null-sets M* **by** (*metis null-sets-UN T-quasi-preserves-null2*(*2*))
moreover have *space M* − *B* = (⋃ *n*. (*T*^^*n*)−−‘(*space M* − *A*))
unfolding *B-def A-def* **by** *auto*
ultimately have *space M* − *B* ∈ *null-sets M* **by** *simp*

have ∗: *B* = (⋂ *n*. (*T*^^*n*)−−‘*A*)
unfolding *B-def A-def* **by** *auto*
then have *T*−−‘*B* = (⋂ *n*. *T*−−‘ (*T*^^*n*)−−‘*A*) **by** *auto*
also have ... = (⋂ *n*. (*T*^^(*n*+*1*))−−‘*A*) **using** *T-vrestr-composed*(*2*)[*OF* ‹*A* ∈ *sets M*›] **by** *simp*
also have ... ⊇ (⋂ *n*. (*T*^^*n*)−−‘*A*) **by** *blast*
finally have *B1*: *B* ⊆ *T*−−‘*B* **using** ∗ **by** *simp*
have *B* ⊆ *A* **using** ∗ *T-vrestr-0*[*OF* ‹*A* ∈ *sets M*›] **by** *blast*
then have *B2*: ⋀*x*. *x* ∈ *B* ⟹ *f*(*T x*) ≤ *f x* **unfolding** *A-def* **by** *auto*

define *C* **where** *C* = (λ*t*. {*x* ∈ *B*. *f x* ≤ *t*})
{
fix *t*
have *C t* = *B* ∩ *f*−‘{..*t*} ∩ *space M* **unfolding** *C-def* **using** *sets.sets-into-space*[*OF* ‹*B* ∈ *sets M*›] **by** *auto*
then have [*measurable*]: *C t* ∈ *sets M* **using** *assms*(*2*) **by** *simp*
have *C t* ⊆ *T*−−‘(*C t*) **using** *B1* **unfolding** *C-def vimage-restr-def* **apply** *auto* **using** *B2 order-trans* **by** *blast*
then have *T*−−‘(*C t*) − *C t* ∈ *null-sets M* **by** (*metis Diff-mono Un-absorb1 preimage-includes-then-almost-invariant*[*OF* ‹*C t* ∈ *sets M*›])
}
then have (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) ∈ *null-sets M* **using** ‹*countable D*› **by** (*simp add*: *null-sets-UN′*)
then have (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) ∈ *null-sets M* **using** ‹*space M* − *B* ∈ *null-sets M*› **by** *auto*
then have *AE x in M*. *x* ∉ (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) **using** *AE-not-in* **by** *blast*
moreover
{
fix *x* **assume** *x*: *x* ∈ *space M x* ∉ (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*)
then have *x* ∈ *B* **by** *simp*
then have *T x* ∈ *B* **using** *B1* **by** *auto*
have *f*(*T x*) = *f x*
proof (*rule ccontr*)

assume $f(T\ x) \neq f\ x$
then have $f(T\ x) < f\ x$ **using** *B2*[*OF* ‹$x \in B$›] **by** *simp*
then obtain *d* **where** *d*: $d \in D\ f(T\ x) \leq d \wedge d < f\ x$ **using** *D* **by** *auto*
then have $T\ x \in C\ d$ **using** ‹$T\ x \in B$› **unfolding** *C-def* **by** *simp*
then have $x \in T{-}{-}`(C\ d)$ **using** ‹$x \in$ *space M*› **by** *simp*
then have $x \in C\ d$ **using** *x* ‹$d \in D$› **by** *simp*
then have $f\ x \leq d$ **unfolding** *C-def* **by** *simp*
then show *False* **using** *d* **by** *auto*
qed
}
ultimately show *?thesis* **by** *auto*
qed

proposition (**in** *conservative*) *AE-increasing-then-invariant*:
fixes *f*::- ⇒ *'b*::{*linorder-topology*, *second-countable-topology*}
assumes *AE x in M*. $f(T\ x) \geq f\ x$
and [*measurable*]: $f \in$ *borel-measurable M*
shows *AE x in M*. $f(T\ x) = f\ x$
proof −
obtain *D*::*'b set* **where** *D*: *countable D* $(\forall x\ y.\ x < y \longrightarrow (\exists d \in D.\ x < d \wedge d \leq y))$
using *countable-separating-set-linorder1* **by** *blast*

define *A* **where** $A = \{x \in$ *space M*. $f(T\ x) \geq f\ x\}$
then have [*measurable*]: $A \in$ *sets M* **by** *simp*
define *B* **where** $B = \{x \in$ *space M*. $\forall n.\ f((T\widehat{\ }\widehat{\ }(n+1))\ x) \geq f((T\widehat{\ }\widehat{\ }n)x)\}$
then have [*measurable*]: $B \in$ *sets M* **by** *simp*

have *space M* − $A \in$ *null-sets M* **unfolding** *A-def* **using** *assms* **by** (*simp add*: *assms*(*1*) *AE-iff-null-sets*)
then have $(\bigcup n.\ (T\widehat{\ }\widehat{\ }n){-}{-}`($*space M* − $A)) \in$ *null-sets M* **by** (*metis null-sets-UN T-quasi-preserves-null2*(*2*))
moreover have *space M* − $B = (\bigcup n.\ (T\widehat{\ }\widehat{\ }n){-}{-}`($*space M* − $A))$
unfolding *B-def A-def* **by** *auto*
ultimately have *space M* − $B \in$ *null-sets M* **by** *simp*

have ∗: $B = (\bigcap n.\ (T\widehat{\ }\widehat{\ }n){-}{-}`A)$
unfolding *B-def A-def* **by** *auto*
then have $T{-}{-}`B = (\bigcap n.\ T{-}{-}`\ (T\widehat{\ }\widehat{\ }n){-}{-}`A)$ **by** *auto*
also have ... $= (\bigcap n.\ (T\widehat{\ }\widehat{\ }(n+1)){-}{-}`A)$ **using** *T-vrestr-composed*(*2*)[*OF* ‹$A \in$ *sets M*›] **by** *simp*
also have ... $\supseteq (\bigcap n.\ (T\widehat{\ }\widehat{\ }n){-}{-}`A)$ **by** *blast*
finally have *B1*: $B \subseteq T{-}{-}`B$ **using** ∗ **by** *simp*
have $B \subseteq A$ **using** ∗ *T-vrestr-0*[*OF* ‹$A \in$ *sets M*›] **by** *blast*
then have *B2*: $\bigwedge x.\ x \in B \Longrightarrow f(T\ x) \geq f\ x$ **unfolding** *A-def* **by** *auto*

define *C* **where** $C = (\lambda t.\ \{x \in B.\ f\ x \geq t\})$
{
fix *t*

have *C t* = *B* ∩ *f* −‘{*t*..} ∩ *space M* **unfolding** *C-def* **using** *sets.sets-into-space*[*OF* ‹*B* ∈ *sets M*›] **by** *auto*
then have [*measurable*]: *C t* ∈ *sets M* **using** *assms*(*2*) **by** *simp*
have *C t* ⊆ *T*−−‘(*C t*) **using** *B1* **unfolding** *C-def vimage-restr-def* **apply** *auto* **using** *B2 order-trans* **by** *blast*
then have *T*−−‘(*C t*) − *C t* ∈ *null-sets M* **by** (*metis Diff-mono Un-absorb1 preimage-includes-then-almost-invariant*[*OF* ‹*C t* ∈ *sets M*›])
}
then have (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) ∈ *null-sets M* **using** ‹*countable D*› **by** (*simp add*: *null-sets-UN'*)
then have (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) ∈ *null-sets M* **using** ‹*space M* − *B* ∈ *null-sets M*› **by** *auto*
then have *AE x in M*. *x* ∉ (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*) **using** *AE-not-in* **by** *blast*
moreover
{
fix *x* **assume** *x*: *x* ∈ *space M x* ∉ (*space M* − *B*) ∪ (⋃ *d*∈*D*. *T*−−‘(*C d*) − *C d*)
then have *x* ∈ *B* **by** *simp*
then have *T x* ∈ *B* **using** *B1* **by** *auto*
have *f*(*T x*) = *f x*
proof (*rule ccontr*)
assume *f*(*T x*) ≠ *f x*
then have *f*(*T x*) > *f x* **using** *B2*[*OF* ‹*x* ∈ *B*›] **by** *simp*
then obtain *d* **where** *d*: *d* ∈ *D f*(*T x*) ≥ *d* ∧ *d* > *f x* **using** *D* **by** *auto*
then have *T x* ∈ *C d* **using** ‹*T x* ∈ *B*› **unfolding** *C-def* **by** *simp*
then have *x* ∈ *T*−−‘(*C d*) **using** ‹*x* ∈ *space M*› **by** *simp*
then have *x* ∈ *C d* **using** *x* ‹*d* ∈ *D*› **by** *simp*
then have *f x* ≥ *d* **unfolding** *C-def* **by** *simp*
then show *False* **using** *d* **by** *auto*
qed
}
ultimately show *?thesis* **by** *auto*
qed

For an invertible map, the invariants of T and T^{-1} are the same.

lemma *Invariants-Tinv*:
assumes *invertible-qmpt*
shows *qmpt.Invariants M Tinv* = *Invariants*
proof −
interpret *I*: *qmpt M Tinv* **using** *Tinv-qmpt*[*OF assms*] **by** *auto*
have (*T* −‘ *A* ∩ *space M* = *A*) ⟷ (*Tinv* −‘ *A* ∩ *space M* = *A*) **if** *A* ∈ *sets M* **for** *A*
proof
assume *T* −‘ *A* ∩ *space M* = *A*
then show *Tinv* −‘ *A* ∩ *space M* = *A*
using *assms that* **unfolding** *Tinv-def invertible-qmpt-def*
apply *auto*
apply (*metis IntE UNIV-I bij-def imageE inv-f-f vimageE*)

apply (*metis I.T-spaceM-stable*(*1*) *Int-iff Tinv-def bij-inv-eq-iff vimageI*)
done
next
assume *Tinv* −' *A* ∩ *space M* = *A*
then show *T* −' *A* ∩ *space M* = *A*
using *assms that* **unfolding** *Tinv-def invertible-qmpt-def*
apply *auto*
apply (*metis IntE bij-def inv-f-f vimageE*)
apply (*metis T-Tinv-of-set T-meas Tinv-def assms qmpt.vrestr-of-set qmpt-axioms vrestr-image*(*3*))
done
qed
then have {*A* ∈ *sets M*. *Tinv* −' *A* ∩ *space M* = *A*} = {*A* ∈ *sets M*. *T* −' *A* ∩ *space M* = *A*}
by *blast*
then show *?thesis* **unfolding** *Invariants-def I.Invariants-def* **by** *auto*
qed

end

sublocale *fmpt* ⊆ *finite-measure-subalgebra M Invariants*
unfolding *finite-measure-subalgebra-def finite-measure-subalgebra-axioms-def*
using *Invariants-is-subalg* **by** (*simp add*: *finite-measureI*)

context *fmpt*
begin

The conditional expectation with respect to the invariant sigma-algebra is the same for f or $f \circ T$, essentially by definition.

lemma *Invariants-of-foTn*:
fixes *f*::'*a* ⇒ *real*
assumes [*measurable*]: *integrable M f*
shows *AE x in M*. *real-cond-exp M Invariants* (*f o* (*T*^^*n*)) *x* = *real-cond-exp M Invariants f x*
proof (*rule real-cond-exp-charact*)
fix *A* **assume** [*measurable*]: *A* ∈ *sets Invariants*
then have [*measurable*]: *A* ∈ *sets M* **using** *Invariants-in-sets* **by** *blast*
then have *ind-meas* [*measurable*]: ((*indicator A*)::('*a* ⇒ *real*)) ∈ *borel-measurable Invariants* **by** *auto*

have *set-lebesgue-integral M A* (*f* ∘ (*T*^^*n*)) = (∫ *x*. *indicator A x* ∗ *f*((*T*^^*n*) *x*) ∂*M*)
by (*auto simp*: *comp-def set-lebesgue-integral-def*)
also have ... = (∫ *x*. *indicator A* ((*T*^^*n*) *x*) ∗ *f* ((*T*^^*n*) *x*) ∂*M*)
by (*rule Bochner-Integration.integral-cong*, *auto simp add*: *Invariants-func-is-invariant-n*[*OF ind-meas*])
also have ... = (∫ *x*. *indicator A x* ∗ *f x* ∂*M*)
apply (*rule Tn-integral-preserving*(*2*)) **using** *integrable-mult-indicator*[*OF* ‹*A* ∈ *sets M*› *assms*] **by** *auto*

also have ... = ($\int x$. *indicator A x* ∗ *real-cond-exp M Invariants f x* ∂*M*)
apply (*rule real-cond-exp-intg*(*2*)[*symmetric*]) **using** *integrable-mult-indicator*[*OF* ‹*A* ∈ *sets M*› *assms*] **by** *auto*
also have ... = *set-lebesgue-integral M A* (*real-cond-exp M Invariants f*)
by (*auto simp*: *set-lebesgue-integral-def*)
finally show *set-lebesgue-integral M A* (*f* ∘ (*T*^^*n*)) = *set-lebesgue-integral M A* (*real-cond-exp M Invariants f*)
by *simp*
qed (*auto simp add*: *assms real-cond-exp-int Tn-integral-preserving*(*1*)[*OF assms*] *comp-def*)

lemma *Invariants-of-foT*:
fixes *f*::′*a* ⇒ *real*
assumes [*measurable*]: *integrable M f*
shows *AE x in M*. *real-cond-exp M Invariants f x* = *real-cond-exp M Invariants* (*f o T*) *x*
using *Invariants-of-foTn*[*OF assms*, **where** *?n* = *1*] **by** *auto*

lemma *birkhoff-sum-Invariants*:
fixes *f*::′*a* ⇒ *real*
assumes [*measurable*]: *integrable M f*
shows *AE x in M*. *real-cond-exp M Invariants* (*birkhoff-sum f n*) *x* = *n* ∗ *real-cond-exp M Invariants f x*
proof −
define *F* **where** *F* = (λ*i*. *f o* (*T*^^*i*))
have [*measurable*]: ⋀*i*. *F i* ∈ *borel-measurable M* **unfolding** *F-def* **by** *auto*
have ∗: *integrable M* (*F i*) **for** *i* **unfolding** *F-def*
by (*subst comp-def*, *rule Tn-integral-preserving*(*1*)[*OF assms*, *of i*])

have *AE x in M*. *n* ∗ *real-cond-exp M Invariants f x* = (∑ *i*∈{..<*n*}. *real-cond-exp M Invariants f x*) **by** *auto*
moreover have *AE x in M*. (∑ *i*∈{..<*n*}. *real-cond-exp M Invariants f x*) = (∑ *i*∈{..<*n*}. *real-cond-exp M Invariants* (*F i*) *x*)
apply (*rule AE-symmetric*[*OF AE-equal-sum*]) **unfolding** *F-def* **using** *Invariants-of-foTn*[*OF assms*] **by** *simp*
moreover have *AE x in M*. (∑ *i*∈{..<*n*}. *real-cond-exp M Invariants* (*F i*) *x*) = *real-cond-exp M Invariants* (λ*x*. ∑ *i*∈{..<*n*}. *F i x*) *x*
by (*rule AE-symmetric*[*OF real-cond-exp-sum* [*OF* ∗]])
moreover have *AE x in M*. *real-cond-exp M Invariants* (λ*x*. ∑ *i*∈{..<*n*}. *F i x*) *x* = *real-cond-exp M Invariants* (*birkhoff-sum f n*) *x*
apply (*rule real-cond-exp-cong*) **unfolding** *F-def* **using** *birkhoff-sum-def*[*symmetric*] **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

end

6.2 Birkhoff theorem

6.2.1 Almost everywhere version of Birkhoff theorem

This paragraph is devoted to the proof of Birkhoff theorem, arguably the most fundamental result of ergodic theory. This theorem asserts that Birkhoff averages of an integrable function f converge almost surely, to the conditional expectation of f with respect to the invariant sigma algebra.

This result implies for instance the strong law of large numbers (in probability theory).

There are numerous proofs of this statement, but none is really easy. We follow the very efficient argument given in Katok-Hasselblatt. To help the reader, here is the same proof informally. The first part of the proof is formalized in `birkhoff_lemma1`, the second one in `birkhoff_lemma`, and the conclusion in `birkhoff_theorem`.

Start with an integrable function g. let $G_n(x) = \max_{k \leq n} S_k g(x)$. Then $\limsup S_n g/n \leq 0$ outside of A, the set where G_n tends to infinity. Moreover, $G_{n+1} - G_n \circ T$ is bounded by g, and tends to g on A. It follows from the dominated convergence theorem that $\int_A G_{n+1} - G_n \circ T \to \int_A g$. As $\int_A G_{n+1} - G_n \circ T = \int_A G_{n+1} - G_n \geq 0$, we obtain $\int_A g \geq 0$.

Apply now this result to the function $g = f - E(f|I) - \epsilon$, where $\epsilon > 0$ is fixed. Then $\int_A g = -\epsilon\mu(A)$, then have $\mu(A) = 0$. Thus, almost surely, $\limsup S_n g/n \leq 0$, i.e., $\limsup S_n f/n \leq E(f|I) + \epsilon$. Letting ϵ tend to 0 gives $\limsup S_n f/n \leq E(f|I)$.

Applying the same result to $-f$ gives $S_n f/n \to E(f|I)$.

context *fmpt*
begin

lemma *birkhoff-aux1*:
  fixes *f*::*'a* ⇒ *real*
  assumes [*measurable*]: *integrable M f*
  defines *A* ≡ {*x* ∈ *space M*. *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x*)) = ∞}
    shows *A* ∈ *sets Invariants* (∫ *x*. *f x* ∗ *indicator A x* ∂*M*) ≥ *0*
proof −
  let *?bsf* = *birkhoff-sum f*
  have [*measurable*]: *A* ∈ *sets M* **unfolding** *A-def* **by** *simp*
  have *Ainv*: *x* ∈ *A* ⟷ *T x* ∈ *A* **if** *x* ∈ *space M* **for** *x*
  proof −
    have *ereal*(*?bsf* (*1* + *n*) *x*) = *ereal*(*f x*) + *ereal*(*?bsf n* (*T x*)) **for** *n*
      unfolding *birkhoff-sum-cocycle birkhoff-sum-1* **by** *simp*
    moreover have *limsup* (λ*n*. *ereal*(*f x*) + *ereal*(*?bsf n* (*T x*)))
                = *ereal*(*f x*) + *limsup*(λ*n*. *ereal*(*?bsf n* (*T x*)))
      by (*rule ereal-limsup-lim-add*, *auto*)
    moreover have *limsup* (λ*n*. *ereal*(*?bsf* (*n*+*1*) *x*)) = *limsup* (λ*n*. *ereal*(*?bsf n* *x*)) **using** *limsup-shift* **by** *simp*
    ultimately have *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x*)) = *ereal*(*f x*) + *limsup*

(λn. *ereal*(*?bsf n* (*T x*))) **by** *simp*
  then have *limsup* (λn. *ereal*(*?bsf n x*)) = ∞ ⟷ *limsup* (λn. *ereal*(*?bsf n* (*T x*))) = ∞ **by** *simp*
  then show *x* ∈ *A* ⟷ *T x* ∈ *A* **using** ‹*x* ∈ *space M*› *A-def* **by** *simp*
 qed
 then show *A* ∈ *sets Invariants* **using** *assms*(*2*) *Invariants-sets* **by** *auto*

 define *F* **where** *F* = (λ*n x*. *MAX k* ∈{*0..n*}. *?bsf k x*)
 have [*measurable*]: ⋀*n*. *F n* ∈ *borel-measurable M* **unfolding** *F-def* **by** *measurable*
 have *intFn*: *integrable M* (*F n*) **for** *n*
  unfolding *F-def* **by** (*rule integrable-MAX*, *auto simp add*: *birkhoff-sum-integral*(*1*)[*OF assms*(*1*)])

 have *Frec*: *F* (*n+1*) *x* − *F n* (*T x*) = *max* (−*F n* (*T x*)) (*f x*) **for** *n x*
 proof −
  have {*0..n+1*} = {*0*} ∪ {*1..n+1*} **by** *auto*
  then have (λ*k*. *?bsf k x*) ‘ {*0..n+1*} = (λ*k*. *?bsf k x*) ‘ {*0*} ∪ (λ*k*. *?bsf k x*) ‘ {*1..n+1*} **by** *blast*
  then have ∗: (λ*k*. *?bsf k x*) ‘ {*0..n+1*} = {*0*} ∪ (λ*k*. *?bsf k x*) ‘ {*1..n+1*} **using** *birkhoff-sum-1*(*1*) **by** *simp*
  have *b*: *F* (*n+1*) *x* = *max* (*Max* {*0*}) (*MAX k* ∈{*1..n+1*}. *?bsf k x*)
   by (*subst F-def*, *subst* ∗, *rule Max.union*, *auto*)

  have (λ*k*. *?bsf k x*) ‘ {*1..n+1*} = (λ*k*. *?bsf* (*1+k*) *x*) ‘ {*0..n*} **using** *Suc-le-D* **by** *fastforce*
  also have ... = (λ*k*. *f x* + *?bsf k* (*T x*)) ‘ {*0..n*}
   by (*subst birkhoff-sum-cocycle*, *subst birkhoff-sum-1*(*2*), *auto*)
  finally have *c*: *F* (*n+1*) *x* = *max 0* (*MAX k* ∈{*0..n*}. *?bsf k* (*T x*) + *f x*) **using** *b* **by** (*simp add*: *add-ac*)

  have {*f x* + *birkhoff-sum f k* (*T x*) |*k*. *k* ∈{*0..n*}} = (+) (*f x*) ‘ {*birkhoff-sum f k* (*T x*) |*k*. *k* ∈{*0..n*}} **by** *blast*
  have (*MAX k* ∈{*0..n*}. *?bsf k* (*T x*) + *f x*) = (*MAX k* ∈{*0..n*}. *?bsf k* (*T x*)) + *f x*
   by (*rule Max-add-commute*) *auto*
  also have ... = *F n* (*T x*) + *f x* **unfolding** *F-def* **by** *simp*
  finally have (*MAX k* ∈{*0..n*}. *?bsf k* (*T x*) + *f x*) = *f x* + *F n* (*T x*) **by** *simp*
  then have *F* (*n+1*) *x* = *max 0* (*f x* + *F n* (*T x*)) **using** *c* **by** *simp*
  then show *F* (*n+1*) *x* − *F n* (*T x*) = *max* (−*F n* (*T x*)) (*f x*) **by** *auto*
 qed

 have *a*: *abs*((*F* (*n+1*) *x* − *F n* (*T x*)) ∗ *indicator A x*) ≤ *abs*(*f x*) **for** *n x*
 proof −
  have *F* (*n+1*) *x* −*F n* (*T x*) ≥ *f x* **using** *Frec* **by** *simp*
  then have ∗: *F* (*n+1*) *x* −*F n* (*T x*) ≥ − *abs*(*f x*) **by** *simp*

  have *F n* (*T x*) ≥ *birkhoff-sum f 0* (*T x*)
   unfolding *F-def* **apply** (*rule Max-ge*, *simp*) **using** *atLeastAtMost-iff* **by** *blast*

then have $F\ n\ (T\ x) \geq 0$ **using** *birkhoff-sum-1(1)* **by** *simp*
then have $-F\ n\ (T\ x) \leq abs\ (f\ x)$ **by** *simp*
moreover have $f\ x \leq abs(f\ x)$ **by** *simp*
ultimately have $F\ (n+1)\ x - F\ n\ (T\ x) \leq abs(f\ x)$ **using** *Frec* **by** *simp*
then have $abs(F\ (n+1)\ x - F\ n\ (T\ x)) \leq abs(f\ x)$ **using** $*$ **by** *simp*
then show $abs((F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x) \leq abs(f\ x)$ **unfolding** *indicator-def* **by** *auto*
qed
have *b*: $(\lambda n.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x) \longrightarrow f\ x * indicator\ A\ x$ **for** *x*
proof (*rule tendsto-eventually*, *cases*)
assume $x \in A$
then have $T\ x \in A$ **using** *Ainv A-def* **by** *auto*
then have $limsup\ (\lambda n.\ ereal(birkhoff\text{-}sum\ f\ n\ (T\ x))) > ereal(-f\ x)$ **unfolding** *A-def* **by** *simp*
then obtain *N* **where** $ereal(?bsf\ N\ (T\ x)) > ereal(-f\ x)$ **using** *Limsup-obtain* **by** *blast*
then have $*$: $?bsf\ N\ (T\ x) > -f\ x$ **by** *simp*
{
fix *n* **assume** $n \geq N$
then have $?bsf\ N\ (T\ x) \in (\lambda k.\ ?bsf\ k\ (T\ x))\ `\ \{0..n\}$ **by** *auto*
then have $F\ n\ (T\ x) \geq ?bsf\ N\ (T\ x)$ **unfolding** *F-def* **by** *simp*
then have $F\ n\ (T\ x) \geq -f\ x$ **using** $*$ **by** *simp*
then have $max\ (-F\ n\ (T\ x))\ (f\ x) = f\ x$ **by** *simp*
then have $F\ (n+1)\ x - F\ n\ (T\ x) = f\ x$ **using** *Frec* **by** *simp*
then have $(F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x = f\ x * indicator\ A\ x$ **by** *simp*
}
then show $eventually\ (\lambda n.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x = f\ x * indicator\ A\ x)\ sequentially$
using *eventually-sequentially* **by** *blast*
next
assume $\neg(x \in A)$
then have $indicator\ A\ x = (0::real)$ **by** *simp*
then show $eventually\ (\lambda n.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x = f\ x * indicator\ A\ x)\ sequentially$ **by** *auto*
qed
have *lim*: $(\lambda n.\ (\int x.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x\ \partial M)) \longrightarrow (\int x.\ f\ x * indicator\ A\ x\ \partial M)$
proof (*rule integral-dominated-convergence*[**where** $?w = (\lambda x.\ abs(f\ x))$])
show $integrable\ M\ (\lambda x.\ |f\ x|)$ **using** *assms(1)* **by** *auto*
show $AE\ x\ in\ M.\ (\lambda n.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x) \longrightarrow f\ x * indicator\ A\ x$ **using** *b* **by** *auto*
show $\bigwedge n.\ AE\ x\ in\ M.\ norm\ ((F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x) \leq |f\ x|$ **using** *a* **by** *auto*
qed (*simp-all*)

have $(\int x.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x\ \partial M) \geq 0$ **for** *n*
proof $-$

have $(\int x.\ F\ n\ (T\ x) * indicator\ A\ x\ \partial M) = (\int x.\ (\lambda x.\ F\ n\ x * indicator\ A\ x)\ (T\ x)\ \partial M)$
by (*rule Bochner-Integration.integral-cong*, *auto simp add*: *Ainv indicator-def*)
also have ... $= (\int x.\ F\ n\ x * indicator\ A\ x\ \partial M)$
by (*rule T-integral-preserving*, *auto simp add*: *intFn integrable-real-mult-indicator*)
finally have *i*: $(\int x.\ F\ n\ (T\ x) * indicator\ A\ x\ \partial M) = (\int x.\ F\ n\ x * indicator\ A\ x\ \partial M)$ **by** *simp*

have $(\int x.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x\ \partial M) = (\int x.\ F\ (n+1)\ x * indicator\ A\ x - F\ n\ (T\ x) * indicator\ A\ x\ \partial M)$
by (*simp add*: *mult.commute right-diff-distrib*)
also have ... $= (\int x.\ F\ (n+1)\ x * indicator\ A\ x\ \partial M) - (\int x.\ F\ n\ (T\ x) * indicator\ A\ x\ \partial M)$
by (*rule Bochner-Integration.integral-diff*, *auto simp add*: *intFn integrable-real-mult-indicator T-meas T-integral-preserving*(*1*))
also have ... $= (\int x.\ F\ (n+1)\ x * indicator\ A\ x\ \partial M) - (\int x.\ F\ n\ x * indicator\ A\ x\ \partial M)$
using *i* **by** *simp*
also have ... $= (\int x.\ F\ (n+1)\ x * indicator\ A\ x - F\ n\ x * indicator\ A\ x\ \partial M)$
by (*rule Bochner-Integration.integral-diff*[*symmetric*], *auto simp add*: *intFn integrable-real-mult-indicator*)
also have ... $= (\int x.\ (F\ (n+1)\ x - F\ n\ x) * indicator\ A\ x\ \partial M)$
by (*simp add*: *mult.commute right-diff-distrib*)
finally have $*$: $(\int x.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x\ \partial M) = (\int x.\ (F\ (n+1)\ x - F\ n\ x) * indicator\ A\ x\ \partial M)$
by *simp*

have $F\ n\ x \le F\ (n+1)\ x$ **for** *x* **unfolding** *F-def* **by** (*rule Max-mono*, *auto*)
then have $(F\ (n+1)\ x - F\ n\ x) * indicator\ A\ x \ge 0$ **for** *x* **by** *simp*
then have $integral^L\ M\ (\lambda x.\ 0) \le integral^L\ M\ (\lambda x.\ (F\ (n+1)\ x - F\ n\ x) * indicator\ A\ x)$
by (*auto simp add*: *intFn integrable-real-mult-indicator intro*: *integral-mono*)
then have $(\int x.\ (F\ (n+1)\ x - F\ n\ x) * indicator\ A\ x\ \partial M) \ge 0$ **by** *simp*
then show $(\int x.\ (F\ (n+1)\ x - F\ n\ (T\ x)) * indicator\ A\ x\ \partial M) \ge 0$ **using** $*$ **by** *simp*
qed
then show $(\int x.\ f\ x * indicator\ A\ x\ \partial M) \ge 0$ **using** *lim* **by** (*simp add*: *LIM-SEQ-le-const*)
qed

lemma *birkhoff-aux2*:
fixes $f{::}'a \Rightarrow real$
assumes [*measurable*]: *integrable M f*
shows $AE\ x\ in\ M.\ limsup\ (\lambda n.\ ereal(birkhoff\text{-}sum\ f\ n\ x\ /\ n)) \le real\text{-}cond\text{-}exp\ M\ Invariants\ f\ x$
proof −
{
fix ε **assume** $\varepsilon > (0{::}real)$
define *g* **where** $g = (\lambda x.\ f\ x - real\text{-}cond\text{-}exp\ M\ Invariants\ f\ x - \varepsilon)$

```
then have intg: integrable M g using assms real-cond-exp-int(1) assms by auto
define A where A = {x ∈ space M. limsup (λn. ereal(birkhoff-sum g n x)) = ∞}
have Ag: A ∈ sets Invariants (∫ x. g x ∗ indicator A x ∂M) ≥ 0
  unfolding A-def by (rule birkhoff-aux1[where ?f = g, OF intg])+
then have [measurable]: A ∈ sets M by (simp add: Invariants-in-sets)

have eq: (∫ x. indicator A x ∗ real-cond-exp M Invariants f x ∂M) = (∫ x. indicator A x ∗ f x ∂M)
proof (rule real-cond-exp-intg[where ?f = λx. (indicator A x)::real and ?g = f])
  have (λx. indicator A x ∗ f x) = (λx. f x ∗ indicator A x) by auto
  then show integrable M (λx. indicator A x ∗ f x)
    using integrable-real-mult-indicator[OF ‹A ∈ sets M› assms] by simp
  show indicator A ∈ borel-measurable Invariants using ‹A ∈ sets Invariants› by measurable
qed (simp)

have 0 ≤ (∫ x. g x ∗ indicator A x ∂M) using Ag by simp
also have ... = (∫ x. f x ∗ indicator A x − real-cond-exp M Invariants f x ∗ indicator A x − ε ∗ indicator A x ∂M)
  unfolding g-def by (simp add: left-diff-distrib)
also have ... = (∫ x. f x ∗ indicator A x ∂M) − (∫ x. real-cond-exp M Invariants f x ∗ indicator A x ∂M) − (∫ x. ε ∗ indicator A x ∂M)
  using assms real-cond-exp-int(1)[OF assms] integrable-real-mult-indicator[OF ‹A ∈ sets M›]
  by (auto simp: simp del: integrable-mult-left-iff)
also have ... = − (∫ x. ε ∗ indicator A x ∂M)
  by (auto simp add: eq mult.commute)
also have ... = − ε ∗ measure M A by auto
finally have 0 ≤ − ε ∗ measure M A by simp
then have measure M A = 0 using ‹ε > 0› by (simp add: measure-le-0-iff mult-le-0-iff)
then have A ∈ null-sets M by (simp add: emeasure-eq-measure null-setsI)
then have AE x in M. x ∈ space M − A by (metis (no-types, lifting) AE-cong Diff-iff AE-not-in)
moreover
{
  fix x assume x ∈ space M − A
  then have limsup (λn. ereal(birkhoff-sum g n x)) < ∞ unfolding A-def by auto
  then obtain C where C: ⋀n. birkhoff-sum g n x ≤ C using limsup-finite-then-bounded by presburger
  {
    fix n::nat assume n > 0
    have birkhoff-sum g n x = birkhoff-sum f n x − birkhoff-sum (real-cond-exp M Invariants f) n x − birkhoff-sum (λx. ε) n x
      unfolding g-def using birkhoff-sum-add birkhoff-sum-diff by auto
```

moreover have *birkhoff-sum* (*real-cond-exp M Invariants f*) *n x* = *n* ∗ *real-cond-exp M Invariants f x*
using *birkhoff-sum-of-invariants* **using** ‹*x* ∈ *space M* − *A*› **by** *auto*
moreover have *birkhoff-sum* (λ*x*. ε) *n x* = *n* ∗ ε **unfolding** *birkhoff-sum-def* **by** *auto*
ultimately have *birkhoff-sum g n x* = *birkhoff-sum f n x* − *n* ∗ *real-cond-exp M Invariants f x* − *n* ∗ ε
by *simp*
then have *birkhoff-sum f n x* = *birkhoff-sum g n x* + *n* ∗ *real-cond-exp M Invariants f x* + *n* ∗ ε
by *simp*
then have *birkhoff-sum f n x* / *n* = *birkhoff-sum g n x* / *n* + *real-cond-exp M Invariants f x* + ε
using ‹*n* > *0*› **by** (*simp add*: *field-simps*)
then have *birkhoff-sum f n x* / *n* ≤ *C*/*n* + *real-cond-exp M Invariants f x* + ε
using *C*[*of n*] ‹*n* > *0*› **by** (*simp add*: *divide-right-mono*)
then have *ereal*(*birkhoff-sum f n x* / *n*) ≤ *ereal*(*C*/*n* + *real-cond-exp M Invariants f x* + ε)
by *simp*
}
then have *eventually* (λ*n*. *ereal*(*birkhoff-sum f n x* / *n*) ≤ *ereal*(*C*/*n* + *real-cond-exp M Invariants f x* + ε)) *sequentially*
by (*simp add*: *eventually-at-top-dense*)
then have *b*: *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x* / *n*)) ≤ *limsup* (λ*n*. *ereal*(*C*/*n* + *real-cond-exp M Invariants f x* + ε))
by (*simp add*: *Limsup-mono*)

have (λ*n*. *ereal*(*C*∗(*1*/*real n*) + *real-cond-exp M Invariants f x* + ε)) ⟶ *ereal*(*C* ∗ *0* + *real-cond-exp M Invariants f x* + ε)
by (*intro tendsto-intros*)
then have *limsup* (λ*n*. *ereal*(*C*/*real n* + *real-cond-exp M Invariants f x* + ε)) = *real-cond-exp M Invariants f x* + ε
using *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *force*
then have *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x* / *n*)) ≤ *real-cond-exp M Invariants f x* + ε
using *b* **by** *simp*
}
ultimately have *AE x in M*. *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x* / *n*)) ≤ *real-cond-exp M Invariants f x* + ε
by *auto*
then have *AE x in M*. *limsup* (λ*n*. *ereal*(*birkhoff-sum f n x* / *n*)) ≤ *ereal*(*real-cond-exp M Invariants f x*) + ε
by *auto*
}
then show *?thesis*
by (*rule AE-upper-bound-inf-ereal*)
qed

theorem *birkhoff-theorem-AE-nonergodic*:
fixes *f*::*'a ⇒ real*
assumes *integrable M f*
shows *AE x in M*. (*λn. birkhoff-sum f n x / n*) ———→ *real-cond-exp M Invariants f x*
proof −
{
fix *x* **assume** *i*: *limsup* (*λn. ereal*(*birkhoff-sum f n x /n*)) ≤ *real-cond-exp M Invariants f x*
and *ii*: *limsup* (*λn. ereal*(*birkhoff-sum* (*λx.* −*f x*) *n x / n*)) ≤ *real-cond-exp M Invariants* (*λx.* −*f x*) *x*
and *iii*: *real-cond-exp M Invariants* (*λx.* −*f x*) *x* = − *real-cond-exp M Invariants f x*
have ⋀*n. birkhoff-sum* (*λx.* −*f x*) *n x* = − *birkhoff-sum f n x*
using *birkhoff-sum-cmult*[**where** *?c* = −*1* **and** *?f* = *f*] **by** *auto*
then have ⋀*n. ereal*(*birkhoff-sum* (*λx.* −*f x*) *n x / n*) = − *ereal*(*birkhoff-sum f n x / n*) **by** *auto*
moreover have *limsup* (*λn.* − *ereal*(*birkhoff-sum f n x / n*)) = − *liminf* (*λn. ereal*(*birkhoff-sum f n x /n*))
by (*rule ereal-Limsup-uminus*)
ultimately have −*liminf* (*λn. ereal*(*birkhoff-sum f n x /n*)) = *limsup* (*λn. ereal*(*birkhoff-sum* (*λx.* −*f x*) *n x / n*))
by *simp*
then have −*liminf* (*λn. ereal*(*birkhoff-sum f n x /n*)) ≤ − *real-cond-exp M Invariants f x*
using *ii iii* **by** *simp*
then have *liminf* (*λn. ereal*(*birkhoff-sum f n x /n*)) ≥ *real-cond-exp M Invariants f x*
by (*simp add*: *ereal-uminus-le-reorder*)
then have (*λn. birkhoff-sum f n x /n*) ———→ *real-cond-exp M Invariants f x*
using *i* **by** (*simp add*: *limsup-le-liminf-real*)
} **note** ∗ = *this*
moreover have *AE x in M*. *limsup* (*λn. ereal*(*birkhoff-sum f n x /n*)) ≤ *real-cond-exp M Invariants f x*
using *birkhoff-aux2 assms* **by** *simp*
moreover have *AE x in M*. *limsup* (*λn. ereal*(*birkhoff-sum* (*λx.* −*f x*) *n x / n*)) ≤ *real-cond-exp M Invariants* (*λx.* −*f x*) *x*
using *birkhoff-aux2 assms* **by** *simp*
moreover have *AE x in M*. *real-cond-exp M Invariants* (*λx.* −*f x*) *x* = − *real-cond-exp M Invariants f x*
using *real-cond-exp-cmult*[**where** *?c* = −*1*] *assms* **by** *force*
ultimately show *?thesis* **by** *auto*
qed

If a function f is integrable, then $E(f \circ T - f|I) = E(f \circ T|I) - E(f|I) = 0$. Hence, $S_n(f \circ T - f)/n$ converges almost everywhere to 0, i.e., $f(T^n x)/n \to 0$. It is remarkable (and sometimes useful) that this holds under the weaker condition that $f \circ T - f$ is integrable (but not necessarily f), where this naive argument fails.

The reason is that the Birkhoff sum of $f \circ T - f$ is $f \circ T^n - f$. If n is such that x and $T^n(x)$ belong to a set where f is bounded, it follows that this Birkhoff sum is also bounded. Along such a sequence of times, $S_n(f \circ T - f)/n$ tends to 0. By Poincare recurrence theorem, there are such times for almost every points. As it also converges to $E(f \circ T - f|I)$, it follows that this function is almost everywhere 0. Then $f(T^n x)/n = S_n(f \circ T^n - f)/n - f/n$ tends almost surely to $E(f \circ T - f|I) = 0$.

lemma *limit-foTn-over-n*:
  fixes *f*::*'a* ⇒ *real*
  assumes [*measurable*]: *f* ∈ *borel-measurable M*
    and *integrable M* (*λx. f*(*T x*) − *f x*)
  shows *AE x in M. real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x* = *0*
    AE x in M. (*λn. f*((*T*^^*n*) *x*) / *n*) ⟶ *0*
proof −
  define *E*::*nat* ⇒ *'a set* **where** *E k* = {*x* ∈ *space M.* |*f x*| ≤ *k*} **for** *k*
  have [*measurable*]: *E k* ∈ *sets M* **for** *k* **unfolding** *E-def* **by** *auto*
  have ∗: (⋃*k. E k*) = *space M* **unfolding** *E-def* **by** (*auto simp add*: *real-arch-simple*)
  define *F*::*nat* ⇒ *'a set* **where** *F k* = *recurrent-subset-infty* (*E k*) **for** *k*
  have [*measurable*]: *F k* ∈ *sets M* **for** *k* **unfolding** *F-def* **by** *auto*
  have ∗∗: *E k* − *F k* ∈ *null-sets M* **for** *k* **unfolding** *F-def* **using** *Poincare-recurrence-thm* **by** *auto*
  have *space M* − (⋃*k. F k*) ∈ *null-sets M*
    apply (*rule null-sets-subset*[*of* (⋃*k. E k* − *F k*)]) **unfolding** ∗[*symmetric*] **using** ∗∗ **by** *auto*
  with *AE-not-in*[*OF this*] **have** *AE x in M. x* ∈ (⋃*k. F k*) **by** *auto*
  moreover have *AE x in M.* (*λn. birkhoff-sum* (*λx. f*(*T x*) − *f x*) *n x* / *n*)
    ⟶ *real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x*
   by (*rule birkhoff-theorem-AE-nonergodic*[*OF assms*(*2*)])
  moreover have *real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x* = *0* ∧ (*λn. f*((*T*^^*n*) *x*) / *n*) ⟶ *0*
    if *H*: (*λn. birkhoff-sum* (*λx. f*(*T x*) − *f x*) *n x* / *n*) ⟶ *real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x*
      x ∈ (⋃*k. F k*) **for** *x*
  proof −
   have *f*((*T*^^*n*) *x*) = *birkhoff-sum* (*λx. f*(*T x*) − *f x*) *n x* + *f x* **for** *n*
    unfolding *birkhoff-sum-def* **by** (*induction n*, *auto*)
   then have *f*((*T*^^*n*) *x*) / *n* = *birkhoff-sum* (*λx. f*(*T x*) − *f x*) *n x* / *n* + *f x* ∗ (*1*/*n*) **for** *n*
    by (*auto simp add*: *divide-simps*)
   moreover have (*λn. birkhoff-sum* (*λx. f*(*T x*) − *f x*) *n x* / *n* + *f x* ∗ (*1*/*n*)) ⟶ *real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x* + *f x* ∗ *0*
    by (*intro tendsto-intros H*(*1*))
   ultimately have *lim*: (*λn. f*((*T*^^*n*) *x*) / *n*) ⟶ *real-cond-exp M Invariants* (*λx. f*(*T x*) − *f x*) *x*
    by *auto*

   obtain *k* **where** *x* ∈ *F k* **using** *H*(*2*) **by** *auto*
   then have *infinite* {*n.* (*T*^^*n*) *x* ∈ *E k*}

unfolding *F-def recurrent-subset-infty-inf-returns* **by** *auto*
with *infinite-enumerate*[*OF this*] **obtain** $r :: nat \Rightarrow nat$
where r: *strict-mono r* $\bigwedge n.\ r\ n \in \{n.\ (T^{\frown\frown}n)\ x \in E\ k\}$
by *auto*
have A: $(\lambda n.\ k * (1/r\ n)) \longrightarrow real\ k * 0$
apply (*intro tendsto-intros*)
using *LIMSEQ-subseq-LIMSEQ*[*OF lim-1-over-n* ‹*strict-mono r*›] **unfolding** *comp-def* **by** *auto*
have B: $|f((T^{\frown\frown}(r\ n))\ x)\ /\ r\ n| \le k\ /\ (r\ n)$ **for** n
using $r(2)$ **unfolding** *E-def* **by** (*auto simp add*: *divide-simps*)
have $(\lambda n.\ f((T^{\frown\frown}(r\ n))\ x)\ /\ r\ n) \longrightarrow 0$
apply (*rule tendsto-rabs-zero-cancel*, *rule tendsto-sandwich*[*of* $\lambda n.\ 0$ - - $\lambda n.\ k * (1/r\ n)$])
using $A\ B$ **by** *auto*
moreover have $(\lambda n.\ f((T^{\frown\frown}(r\ n))\ x)\ /\ r\ n) \longrightarrow$ *real-cond-exp M Invariants* $(\lambda x.\ f(T\ x) - f\ x)\ x$
using *LIMSEQ-subseq-LIMSEQ*[*OF lim* ‹*strict-mono r*›] **unfolding** *comp-def* **by** *auto*
ultimately have $*$: *real-cond-exp M Invariants* $(\lambda x.\ f(T\ x) - f\ x)\ x = 0$
using *LIMSEQ-unique* **by** *auto*
then have $(\lambda n.\ f((T^{\frown\frown}n)\ x)\ /\ n) \longrightarrow 0$ **using** *lim* **by** *auto*
then show *?thesis* **using** $*$ **by** *auto*
qed
ultimately show *AE x in M*. *real-cond-exp M Invariants* $(\lambda x.\ f(T\ x) - f\ x)\ x = 0$
AE x in M. $(\lambda n.\ f((T^{\frown\frown}n)\ x)\ /\ n) \longrightarrow 0$
by *auto*
qed

We specialize the previous statement to the case where f itself is integrable.

lemma *limit-foTn-over-n′*:
fixes $f::'a \Rightarrow real$
assumes [*measurable*]: *integrable M f*
shows *AE x in M*. $(\lambda n.\ f((T^{\frown\frown}n)\ x)\ /\ n) \longrightarrow 0$
by (*rule limit-foTn-over-n*, *simp*, *rule Bochner-Integration.integrable-diff*)
(*auto intro*: *assms T-integral-preserving*(*1*))

It is often useful to show that a function is cohomologous to a nicer function, i.e., to prove that a given f can be written as $f = g + u - u \circ T$ where g is nicer than f. We show below that any integrable function is cohomologous to a function which is arbitrarily close to $E(f|I)$. This is an improved version of Lemma 2.1 in [Benoist-Quint, Annals of maths, 2011]. Note that the function g to which f is cohomologous is very nice (and, in particular, integrable), but the transfer function is only measurable in this argument. The fact that the control on conditional expectation is nevertheless preserved throughout the argument follows from Lemma `limit_foTn_over_n` above.

We start with the lemma (and the proof) of [BQ2011]. It shows that, if a function has a conditional expectation with respect to invariants which is positive, then it is cohomologous to a nonnegative function. The argument is the clever remark that $g = \max(0, \inf_n S_n f)$ and $u = \min(0, \inf_n S_n f)$ work (where these expressions are well defined as $S_n f$ tends to infinity thanks to our assumption).

lemma *cohomologous-approx-cond-exp-aux*:
fixes *f*::*'a ⇒ real*
assumes [*measurable*]: *integrable M f*
and *AE x in M. real-cond-exp M Invariants f x > 0*
shows ∃ *u g. u* ∈ *borel-measurable M* ∧ (*integrable M g*) ∧ (*AE x in M. g x* ≥ *0* ∧ *g x* ≤ *max 0* (*f x*)) ∧ (∀ *x. f x* = *g x* + *u x* − *u* (*T x*))
proof −
define *h*::*'a ⇒ real* **where** *h* = (λ*x.* (*INF n*∈{*1*..}. *birkhoff-sum f n x*))
define *u* **where** *u* = (λ*x. min* (*h x*) *0*)
define *g* **where** *g* = (λ*x. f x* − *u x* + *u* (*T x*))
have [*measurable*]: *h* ∈ *borel-measurable M u* ∈ *borel-measurable M g* ∈ *borel-measurable M*
unfolding *g-def h-def u-def* **by** *auto*
have *f x* = *g x* + *u x* − *u* (*T x*) **for** *x* **unfolding** *g-def* **by** *auto*
{
fix *x* **assume** *H*: *real-cond-exp M Invariants f x > 0*
(λ*n. birkhoff-sum f n x* / *n*) ⟶ *real-cond-exp M Invariants f x*
have *eventually* (λ*n. ereal*(*birkhoff-sum f n x* / *n*) ∗ *ereal n* = *ereal*(*birkhoff-sum f n x*)) *sequentially*
unfolding *eventually-sequentially* **by** (*rule exI*[*of* - *1*], *auto*)
moreover have (λ*n. ereal*(*birkhoff-sum f n x* / *n*) ∗ *ereal n*) ⟶ *ereal*(*real-cond-exp M Invariants f x*) ∗ ∞
apply (*intro tendsto-intros*) **using** *H* **by** *auto*
ultimately have (λ*n. ereal*(*birkhoff-sum f n x*)) ⟶ *ereal*(*real-cond-exp M Invariants f x*) ∗ ∞
by (*blast intro*: *Lim-transform-eventually*)
then have (λ*n. ereal*(*birkhoff-sum f n x*)) ⟶ ∞
using *H* **by** *auto*
then have *B*: ∃ *C.* ∀ *n. C* ≤ *birkhoff-sum f n x*
by (*intro liminf-finite-then-bounded-below*, *simp add*: *liminf-PInfty*)

have *h x* ≤ *f x*
unfolding *h-def* **apply** (*rule cInf-lower*) **using** *B* **by** *force+*

have {*birkhoff-sum f n* (*T x*) |*n. n* ∈ {*1*..}} = {*birkhoff-sum f* (*1*+*n*) (*x*) − *f x* |*n. n* ∈ {*1*..}}
unfolding *birkhoff-sum-cocycle* **by** *auto*
also have ... = {*birkhoff-sum f n x* − *f x* |*n. n* ∈ {*2*..}}
by (*metis* (*no-types*, *opaque-lifting*) *Suc-1 Suc-eq-plus1-left Suc-le-D Suc-le-mono atLeast-iff*)
finally have ∗: {*birkhoff-sum f n* (*T x*) |*n. n* ∈ {*1*..}} = (λ*t. t* − (*f x*)) ' {*birkhoff-sum f n x* |*n. n* ∈ {*2*..}}
by *auto*

have *h*(*T x*) = *Inf* {*birkhoff-sum f n* (*T x*) |*n. n* ∈ {*1*..}}

unfolding *h-def* **by** (*metis Setcompr-eq-image*)
also have ... = $(\prod t \in \{birkhoff\text{-}sum\ f\ n\ x\ |n.\ n \in \{2..\}\}.\ t - f\ x)$
by (*simp only*: ∗)
also have ... = $(\lambda t.\ t - (f\ x))$ (*Inf* {*birkhoff-sum f n x* |*n. n* ∈ {*2..*}})
using *B* **by** (*auto intro*!: *monoI bijI mono-bij-cInf* [*symmetric*])
finally have *I*: *Inf* {*birkhoff-sum f n x* |*n. n* ∈ {*2..*}} = *f x* + *h* (*T x*) **by** *auto*
have *max 0* (*h x*) + *u x* = *h x*
unfolding *u-def* **by** *auto*
also have ... = *Inf* {*birkhoff-sum f n x* |*n. n* ∈ {*1..*}}
unfolding *h-def* **by** (*metis Setcompr-eq-image*)
also have ... = *Inf* ({*birkhoff-sum f n x* |*n. n* ∈ {*1*}} ∪ {*birkhoff-sum f n x* |*n. n* ∈ {*2..*}})
by (*auto intro*!: *arg-cong*[*of - - Inf*], *metis One-nat-def Suc-1 antisym birkhoff-sum-1*(*2*) *not-less-eq-eq*, *force*)
also have *Inf* ({*birkhoff-sum f n x* |*n. n* ∈ {*1*}} ∪ {*birkhoff-sum f n x* |*n. n* ∈ {*2..*}})
= *min* (*Inf* {*birkhoff-sum f n x* |*n. n* ∈ {*1*}}) (*Inf* {*birkhoff-sum f n x* |*n. n* ∈ {*2..*}})
unfolding *inf-min*[*symmetric*] **apply** (*intro cInf-union-distrib*) **using** *B* **by** *auto*
also have ... = *min* (*f x*) (*f x* + *h* (*T x*)) **using** *I* **by** *auto*
also have ... = *f x* + *u* (*T x*) **unfolding** *u-def* **by** *auto*
finally have *max 0* (*h x*) = *f x* + *u* (*T x*) − *u x* **by** *auto*
then have *g x* = *max 0* (*h x*) **unfolding** *g-def* **by** *auto*
then have *g x* ≥ *0* ∧ *g x* ≤ *max 0* (*f x*) **using** ‹*h x* ≤ *f x*› **by** *auto*
}
then have ∗: *AE x in M*. *g x* ≥ *0* ∧ *g x* ≤ *max 0* (*f x*)
using *assms*(*2*) *birkhoff-theorem-AE-nonergodic*[*OF assms*(*1*)] **by** *auto*
moreover have *integrable M g*
apply (*rule Bochner-Integration.integrable-bound*[*of - f*]) **using** ∗ **by** (*auto simp add*: *assms*)
ultimately have *u* ∈ *borel-measurable M* ∧ *integrable M g* ∧ (*AE x in M*. *0* ≤ *g x* ∧ *g x* ≤ *max 0* (*f x*)) ∧ (∀ *x*. *f x* = *g x* + *u x* − *u* (*T x*))
using ‹⋀*x*. *f x* = *g x* + *u x* − *u* (*T x*)› ‹*u* ∈ *borel-measurable M*› **by** *auto*
then show *?thesis* **by** *blast*
qed

To deduce the stronger version that f is cohomologous to an arbitrarily good approximation of $E(f|I)$, we apply the previous lemma twice, to control successively the negative and the positive side. The sign control in the conclusion of the previous lemma implies that the second step does not spoil the first one.

lemma *cohomologous-approx-cond-exp*:
fixes *f*::′*a* ⇒ *real* **and** *B*::′*a* ⇒ *real*
assumes [*measurable*]: *integrable M f B* ∈ *borel-measurable M*
and *AE x in M*. *B x* > *0*
shows ∃ *g u*. *u* ∈ *borel-measurable M*
∧ *integrable M g*
∧ (∀ *x*. *f x* = *g x* + *u x* − *u* (*T x*))

∧ (*AE x in M. abs*(*g x* − *real-cond-exp M Invariants f x*) ≤ *B x*)
proof −
define *C* **where** *C* = (λ*x. min* (*B x*) *1*)
have [*measurable*]: *integrable M C*
apply (*rule Bochner-Integration.integrable-bound*[*of* - λ-. (*1*::*real*)], *auto*)
unfolding *C-def* **using** *assms*(*3*) **by** *auto*
have *C x* ≤ *B x* **for** *x* **unfolding** *C-def* **by** *auto*
have *AE x in M. C x* > *0* **unfolding** *C-def* **using** *assms*(*3*) **by** *auto*
have *AECI*: *AE x in M. real-cond-exp M Invariants C x* > *0*
by (*intro real-cond-exp-gr-c* ‹*integrable M C*› ‹*AE x in M. C x* > *0*›)

define *f1* **where** *f1* = (λ*x. f x* − *real-cond-exp M Invariants f x*)
have *integrable M f1*
unfolding *f1-def* **by** (*intro Bochner-Integration.integrable-diff* ‹*integrable M f*› *real-cond-exp-int*(*1*))
have *AE x in M. real-cond-exp M Invariants f1 x* = *real-cond-exp M Invariants f x* − *real-cond-exp M Invariants* (*real-cond-exp M Invariants f*) *x*
unfolding *f1-def* **apply** (*rule real-cond-exp-diff*) **by** (*intro Bochner-Integration.integrable-diff* ‹*integrable M f*› ‹*integrable M C*› *real-cond-exp-int*(*1*))+
moreover have *AE x in M. real-cond-exp M Invariants* (*real-cond-exp M Invariants f*) *x* = *real-cond-exp M Invariants f x*
by (*intro real-cond-exp-nested-subalg subalg* ‹*integrable M f*›, *auto*)
ultimately have *AEf1*: *AE x in M. real-cond-exp M Invariants f1 x* = *0* **by** *auto*

have *A* [*measurable*]: *integrable M* (λ*x. f1 x* + *C x*)
by (*intro Bochner-Integration.integrable-add* ‹*integrable M f1*› ‹*integrable M C*›)
have *AE x in M. real-cond-exp M Invariants* (λ*x. f1 x* + *C x*) *x* = *real-cond-exp M Invariants f1 x* + *real-cond-exp M Invariants C x*
by (*intro real-cond-exp-add* ‹*integrable M f1*› ‹*integrable M C*›)
then have *B*: *AE x in M. real-cond-exp M Invariants* (λ*x. f1 x* + *C x*) *x* > *0*
using *AECI AEf1* **by** *auto*

obtain *u2 g2* **where** *H2*: *u2* ∈ *borel-measurable M integrable M g2 AE x in M. g2 x* ≥ *0* ∧ *g2 x* ≤ *max 0* (*f1 x* + *C x*) ⋀*x. f1 x* + *C x* = *g2 x* + *u2 x* − *u2* (*T x*)
using *cohomologous-approx-cond-exp-aux*[*OF A B*] **by** *blast*

define *f2* **where** *f2* = (λ*x.* (*g2 x* − *C x*))
have ∗: *u2*(*T x*) − *u2 x* = *f2 x* −*f1 x* **for** *x* **unfolding** *f2-def* **using** *H2*(*4*)[*of x*] **by** *auto*
have *AE x in M. f2 x* ≥ − *C x* **using** *H2*(*3*) **unfolding** *f2-def* **by** *auto*
have *integrable M f2*
unfolding *f2-def* **by** (*intro Bochner-Integration.integrable-diff* ‹*integrable M g2*› ‹*integrable M C*›)
have *AE x in M. real-cond-exp M Invariants* (λ*x. u2*(*T x*) − *u2 x*) *x* = *0*
proof (*rule limit-foTn-over-n*)
show *integrable M* (λ*x. u2*(*T x*) − *u2 x*)
unfolding ∗ **by** (*intro Bochner-Integration.integrable-diff* ‹*integrable M f1*›

‹*integrable M f2*›)
qed (*simp add*: ‹*u2* ∈ *borel-measurable M*›)
then have *AE x in M. real-cond-exp M Invariants* (λ*x. f2 x* − *f1 x*) *x* = *0*
unfolding ∗ **by** *simp*
moreover have *AE x in M. real-cond-exp M Invariants* (λ*x. f2 x* − *f1 x*) *x* = *real-cond-exp M Invariants f2 x* − *real-cond-exp M Invariants f1 x*
by (*intro real-cond-exp-diff* ‹*integrable M f2*› ‹*integrable M f1*›)
ultimately have *AEf2*: *AE x in M. real-cond-exp M Invariants f2 x* = *0*
using *AEf1* **by** *auto*

have *A* [*measurable*]: *integrable M* (λ*x. C x* − *f2 x*)
by (*intro Bochner-Integration.integrable-diff* ‹*integrable M f2*› ‹*integrable M C*›)
have *AE x in M. real-cond-exp M Invariants* (λ*x. C x* − *f2 x*) *x* = *real-cond-exp M Invariants C x* − *real-cond-exp M Invariants f2 x*
by (*intro real-cond-exp-diff* ‹*integrable M f2*› ‹*integrable M C*›)
then have *B*: *AE x in M. real-cond-exp M Invariants* (λ*x. C x* − *f2 x*) *x* > *0*
using *AECI AEf2* **by** *auto*

obtain *u3 g3* **where** *H3*: *u3* ∈ *borel-measurable M integrable M g3 AE x in M. g3 x* ≥ *0* ∧ *g3 x* ≤ *max 0* (*C x* − *f2 x*) ⋀*x. C x* − *f2 x* = *g3 x* + *u3 x* − *u3* (*T x*)
using *cohomologous-approx-cond-exp-aux*[*OF A B*] **by** *blast*

define *f3* **where** *f3* = (λ*x. C x* − *g3 x*)
have *AE x in M. f3 x* ≥ *min* (*C x*) (*f2 x*) **unfolding** *f3-def* **using** *H3*(*3*) **by** *auto*
then have *AE x in M. f3 x* ≥ −*C x* **using** ‹*AE x in M. f2 x* ≥ − *C x*› ‹*AE x in M. C x* > *0*› **by** *auto*
moreover have *AE x in M. f3 x* ≤ *C x* **unfolding** *f3-def* **using** *H3*(*3*) **by** *auto*
ultimately have *AE x in M. abs*(*f3 x*) ≤ *C x* **by** *auto*
then have ∗: *AE x in M. abs*(*f3 x*) ≤ *B x* **using** *order-trans*[*OF* - ‹⋀*x. C x* ≤ *B x*›] **by** *auto*

define *g* **where** *g* = (λ*x. f3 x* + *real-cond-exp M Invariants f x*)
define *u* **where** *u* = (λ*x. u2 x* − *u3 x*)
have *AE x in M. abs* (*g x* − *real-cond-exp M Invariants f x*) ≤ *B x*
unfolding *g-def* **using** ∗ **by** *auto*
moreover have *f x* = *g x* + *u x* − *u*(*T x*) **for** *x*
using *H3*(*4*)[*of x*] *H2*(*4*)[*of x*] **unfolding** *u-def g-def f3-def f2-def f1-def* **by** *auto*
moreover have *u* ∈ *borel-measurable M*
unfolding *u-def* **using** ‹*u2* ∈ *borel-measurable M*› ‹*u3* ∈ *borel-measurable M*› **by** *auto*
moreover have *integrable M g*
unfolding *g-def f3-def* **by** (*intro Bochner-Integration.integrable-add Bochner-Integration.integrable-diff* ‹*integrable M C*› ‹*integrable M g3*› ‹*integrable M f*› *real-cond-exp-int*(*1*))
ultimately show *?thesis* **by** *auto*
qed

6.2.2 L^1 version of Birkhoff theorem

The L^1 convergence in Birkhoff theorem follows from the almost everywhere convergence and general considerations on L^1 convergence (Scheffe's lemma) explained in `AE_and_int_bound_implies_L1_conv2`. This argument works neatly for nonnegative functions, the general case reduces to this one by taking the positive and negative parts of a given function.

One could also prove it by truncation: for bounded functions, the L^1 convergence follows from the boundedness and almost sure convergence. The general case follows by density, but it is a little bit tedious to write as one need to make sure that the conditional expectation of the truncation converges to the conditional expectation of the original function. This is true in L^1 as the conditional expectation is a contraction in L^1, it follows almost everywhere after taking a subsequence. All in all, the argument based on Scheffe's lemma seems more economical.

lemma *birkhoff-lemma-L1*:
  fixes f::'*a* ⇒ *real*
  assumes $\bigwedge x.\ f\ x \geq 0$
    and [*measurable*]: *integrable M f*
  shows ($\lambda n.\ \int^{+} x.$ *norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂M) ⟶ *0*
proof (*rule Scheffe-lemma2*)
  show *i*: *integrable M* (*real-cond-exp M Invariants f*) **using** *assms* **by** (*simp add*: *real-cond-exp-int*(*1*))
  show *AE x in M*. ($\lambda n.$ *birkhoff-sum f n x* / *real n*) ⟶ *real-cond-exp M Invariants f x*
    using *birkhoff-theorem-AE-nonergodic assms* **by** *simp*
  fix *n*
  have [*measurable*]: ($\lambda x.$ *ennreal* |*birkhoff-sum f n x*|) ∈ *borel-measurable M* **by** *measurable*
  show [*measurable*]: ($\lambda x.$ *birkhoff-sum f n x* / *real n*) ∈ *borel-measurable M* **by** *measurable*

  have *AE x in M*. *real-cond-exp M Invariants f x* ≥ *0* **using** *assms*(*1*) *real-cond-exp-pos* **by** *simp*
  then have ∗: *AE x in M*. *norm* (*real-cond-exp M Invariants f x*) = *real-cond-exp M Invariants f x* **by** *auto*
  have ∗∗: ($\int x.$ *norm* (*real-cond-exp M Invariants f x*) ∂M) = ($\int x.$ *real-cond-exp M Invariants f x* ∂M)
    apply (*rule integral-cong-AE*) **using** ∗ **by** *auto*

  have ($\int^{+} x.$ *ennreal* (*norm* (*real-cond-exp M Invariants f x*)) ∂M) = ($\int x.$ *norm* (*real-cond-exp M Invariants f x*) ∂M)
    by (*rule nn-integral-eq-integral*) (*auto simp add*: *i*)
  also have ... = ($\int x.$ *real-cond-exp M Invariants f x* ∂M)
    using ∗∗ **by** *simp*
  also have ... = ($\int x.\ f\ x\ \partial M$)

using *real-cond-exp-int*(*2*) *assms*(*2*) **by** *auto*
also have ... = (∫ *x*. *norm*(*f x*) ∂*M*) **using** *assms* **by** *auto*
also have ... = (∫ ⁺*x*. *norm*(*f x*) ∂*M*)
by (*rule nn-integral-eq-integral*[*symmetric*], *auto simp add*: *assms*(*2*))
finally have *eq*: (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*) = (∫ ⁺ *x*. *norm*(*f x*) ∂*M*) **by** *simp*

{
fix *x*
have *norm*(*birkhoff-sum f n x*) ≤ *birkhoff-sum* (λ*x*. *norm*(*f x*)) *n x*
using *birkhoff-sum-abs* **by** *fastforce*
then have *norm*(*birkhoff-sum f n x*) ≤ *birkhoff-sum* (λ*x*. *ennreal*(*norm*(*f x*))) *n x*
unfolding *birkhoff-sum-def* **by** *auto*
}
then have (∫ ⁺*x*. *norm*(*birkhoff-sum f n x*) ∂*M*) ≤ (∫ ⁺*x*. *birkhoff-sum* (λ*x*. *ennreal*(*norm*(*f x*))) *n x* ∂*M*)
by (*simp add*: *nn-integral-mono*)
also have ... = *n* ∗ (∫ ⁺*x*. *norm*(*f x*) ∂*M*)
by (*rule birkhoff-sum-nn-integral*, *auto*)
also have ... = *n* ∗ (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*)
using *eq* **by** *simp*
finally have ∗: (∫ ⁺*x*. *norm*(*birkhoff-sum f n x*) ∂*M*) ≤ *n* ∗ (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*)
by *simp*

show (∫ ⁺ *x*. *ennreal* (*norm* (*birkhoff-sum f n x* / *real n*)) ∂*M*) ≤ (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*)
proof (*cases*)
assume *n* = *0*
then show *?thesis* **by** *auto*
next
assume ¬(*n* = *0*)
then have *n* > *0* **by** *simp*
then have *1*/*ennreal*(*real n*) ≥ *0* **by** *simp*
have (∫ ⁺ *x*. *ennreal* (*norm* (*birkhoff-sum f n x* / *real n*)) ∂*M*) = (∫ ⁺ *x*. *ennreal* (*norm* (*birkhoff-sum f n x*)) / *ennreal*(*real n*) ∂*M*)
using ‹*n* > *0*› **by** (*auto simp*: *divide-ennreal*)
also have ... = (∫ ⁺ *x*. (*1*/*ennreal*(*real n*)) ∗ *ennreal* (*norm* (*birkhoff-sum f n x*)) ∂*M*)
by (*simp add*: ‹*0* < *n*› *divide-ennreal-def mult.commute*)
also have ... = (*1*/*ennreal*(*real n*) ∗ (∫ ⁺ *x*. *ennreal* (*norm* (*birkhoff-sum f n x*)) ∂*M*))
by (*subst nn-integral-cmult*) *auto*
also have ... ≤ (*1*/*ennreal*(*real n*)) ∗ (*ennreal*(*real n*) ∗ (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*))
using ∗ **by** (*intro mult-mono*) (*auto simp*: *ennreal-of-nat-eq-real-of-nat*)
also have ... = (∫ ⁺ *x*. *norm* (*real-cond-exp M Invariants f x*) ∂*M*)
using ‹*n* > *0*›

by (*auto simp del*: *ennreal-1 simp add*: *ennreal-1*[*symmetric*] *divide-ennreal ennreal-mult*[*symmetric*] *mult.assoc*[*symmetric*])
simp
finally show *?thesis* **by** *simp*
qed
qed

theorem *birkhoff-theorem-L1-nonergodic*:
fixes $f::'a \Rightarrow real$
assumes [*measurable*]: *integrable M f*
shows $(\lambda n. \int^{+}x.\ norm(birkhoff\text{-}sum\ f\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ f\ x)\ \partial M) \longrightarrow 0$
proof −
define *g* **where** $g = (\lambda x.\ max\ (f\ x)\ 0)$
have *g-int* [*measurable*]: *integrable M g* **unfolding** *g-def* **using** *assms* **by** *auto*
define *h* **where** $h = (\lambda x.\ max\ (-f\ x)\ 0)$
have *h-int* [*measurable*]: *integrable M h* **unfolding** *h-def* **using** *assms* **by** *auto*
have $f = (\lambda x.\ g\ x - h\ x)$ **unfolding** *g-def h-def* **by** *auto*
{
fix *n*::*nat* **assume** $n > 0$
have $\bigwedge x.\ birkhoff\text{-}sum\ f\ n\ x = birkhoff\text{-}sum\ g\ n\ x - birkhoff\text{-}sum\ h\ n\ x$ **using** *birkhoff-sum-diff* ‹$f = (\lambda x.\ g\ x - h\ x)$› **by** *auto*
then have $\bigwedge x.\ birkhoff\text{-}sum\ f\ n\ x\ /\ n = birkhoff\text{-}sum\ g\ n\ x\ /\ n - birkhoff\text{-}sum\ h\ n\ x\ /\ n$ **using** ‹$n > 0$› **by** (*simp add*: *diff-divide-distrib*)
moreover have *AE x in M*. *real-cond-exp M Invariants g x* − *real-cond-exp M Invariants h x* = *real-cond-exp M Invariants f x*
using *AE-symmetric*[*OF real-cond-exp-diff*] *g-int h-int* ‹$f = (\lambda x.\ g\ x - h\ x)$› **by** *auto*
ultimately have *AE x in M*. *birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x* =
(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) − (*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*)
by *auto*
then have ∗: *AE x in M*. *norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ≤
norm(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) + *norm*(*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*)
by *auto*
have $(\int^{+}x.\ norm(birkhoff\text{-}sum\ f\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ f\ x)\ \partial M) \leq$
$(\int^{+}x.\ ennreal(norm(birkhoff\text{-}sum\ g\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ g\ x)) + norm(birkhoff\text{-}sum\ h\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ h\ x)\ \partial M)$
apply (*rule nn-integral-mono-AE*) **using** ∗ **by** (*simp add*: *ennreal-plus*[*symmetric*] *del*: *ennreal-plus*)
also have ... $= (\int^{+}x.\ norm(birkhoff\text{-}sum\ g\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ g\ x)\ \partial M) + (\int^{+}x.\ norm(birkhoff\text{-}sum\ h\ n\ x\ /\ n - real\text{-}cond\text{-}exp\ M\ Invariants\ h\ x)\ \partial M)$
apply (*rule nn-integral-add*) **apply** *auto* **using** *real-cond-exp-F-meas borel-measurable-cond-exp2* **by** *measurable*

finally have ($\int^+$ *x. norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂M) ≤
($\int^+$ *x. norm*(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) ∂M) + ($\int^+$ *x. norm*(*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*) ∂M)
by *simp*
}
then have ∗: *eventually* (λn. ($\int^+$ *x. norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂M) ≤
($\int^+$ *x. norm*(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) ∂M) + ($\int^+$ *x. norm*(*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*) ∂M))
sequentially
using *eventually-at-top-dense* **by** *auto*
have ∗∗: *eventually* (λn. ($\int^+$ *x. norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂M) ≥ *0*) *sequentially*
by *simp*

have (λn. ($\int^+$ *x. norm*(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) ∂M)) ⟶ *0*
apply (*rule birkhoff-lemma-L1*, *auto simp add*: *g-int*) **unfolding** *g-def* **by** *auto*
moreover have (λn. ($\int^+$ *x. norm*(*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*) ∂M)) ⟶ *0*
apply (*rule birkhoff-lemma-L1*, *auto simp add*: *h-int*) **unfolding** *h-def* **by** *auto*
ultimately have (λn. ($\int^+$ *x. norm*(*birkhoff-sum g n x* / *n* − *real-cond-exp M Invariants g x*) ∂M) + ($\int^+$ *x. norm*(*birkhoff-sum h n x* / *n* − *real-cond-exp M Invariants h x*) ∂M)) ⟶ *0*
using *tendsto-add*[*of - 0 - - 0*] **by** *auto*
then show *?thesis*
using *tendsto-sandwich*[*OF* ∗∗ ∗] **by** *auto*
qed

6.2.3 Conservativity of skew products

The behaviour of skew-products of the form $(x, y) \mapsto (Tx, y + fx)$ is directly related to Birkhoff theorem, as the iterates involve the Birkhoff sums in the fiber. Birkhoff theorem implies that such a skew product is conservative when the function f has vanishing conditional expectation.

To prove the theorem, assume by contradiction that a set A with positive measure does not intersect its preimages. Replacing A with a smaller set C, we can assume that C is bounded in the y-direction, by a constant N, and also that all its nonempty vertical fibers, above the projection Cx, have a measure bounded from below. Then, by Birkhoff theorem, for any $r > 0$, most of the first n preimages of C are contained in the set $\{|y| \leq rn + N\}$, of measure $O(rn)$. Hence, they can not be disjoint if $r < \mu(C)$. To make this argument rigorous, one should only consider the preimages whose x-component belongs to a set Dx where the Birkhoff sums are small. This condition has a positive measure if $\mu(Cx) + \mu(Dx) > \mu(M)$, which one can ensure by taking Dx large enough.

theorem (**in** *fmpt*) *skew-product-conservative*:
fixes *f*::*'a* ⇒ *real*
assumes [*measurable*]: *integrable M f*
and *AE x in M. real-cond-exp M Invariants f x = 0*
shows *conservative-mpt* (*M* $\bigotimes_M$ *lborel*) (λ(*x*,*y*). (*T x*, *y* + *f x*))
proof (*rule conservative-mptI*)
let *?TS* = (λ(*x*,*y*). (*T x*, *y* + *f x*))
let *?MS* = *M* $\bigotimes_M$ (*lborel*::*real measure*)

have *f-meas* [*measurable*]: *f* ∈ *borel-measurable M* **by** *auto*
have *mpt M T* **by** (*simp add*: *mpt-axioms*)
with *mpt-skew-product-real*[*OF this f-meas*] **show** *mpt ?MS ?TS* **by** *simp*
then interpret *TS*: *mpt ?MS ?TS* **by** *auto*

fix *A*::(*'a* × *real*) *set*
assume *A1* [*measurable*]: *A* ∈ *sets ?MS* **and** *A2*: *emeasure ?MS A* > *0*
have *A* = (⋃ *N*::*nat*. *A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *N*}) **by** (*auto simp add*: *real-arch-simple*)
then have ∗: *emeasure ?MS* (⋃ *N*::*nat*. *A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *N*}) > *0*
using *A2* **by** *simp*

have *space ?MS* = *space M* × *space* (*lborel*::*real measure*) **using** *space-pair-measure* **by** *auto*
then have *A-inc*: *A* ⊆ *space M* × *space* (*lborel*::*real measure*) **using** *sets.sets-into-space*[*OF A1*] **by** *auto*

{
fix *N*::*nat*
have {(*x*, *y*). *abs*(*y*) ≤ *real N* ∧ *x* ∈ *space M*} = *space M* × {−(*real N*)..(*real N*)} **by** *auto*
then have {(*x*, *y*). |*y*| ≤ *real N* ∧ *x* ∈ *space M*} ∈ *sets ?MS* **by** *auto*
then have *A* ∩ {(*x*, *y*). |*y*| ≤ *real N* ∧ *x* ∈ *space M*} ∈ *sets ?MS* **using** *A1* **by** *auto*
moreover have *A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *real N*} = *A* ∩ {(*x*, *y*). |*y*| ≤ *real N* ∧ *x* ∈ *space M*}
using *A-inc* **by** *blast*
ultimately have *A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *real N*} ∈ *sets ?MS* **by** *auto*
}
then have [*measurable*]: ⋀*N*::*nat*. *A* ∩ {(*x*, *y*). |*y*| ≤ *real N*} ∈ *sets* (*M* $\bigotimes_M$ *borel*) **by** *auto*

have ∃ *N*::*nat*. *emeasure ?MS* (*A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *N*}) > *0*
apply (*rule emeasure-pos-unionE*) **using** ∗ **by** *auto*
then obtain *N*::*nat* **where** *N*: *emeasure ?MS* (*A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *N*}) > *0*
by *auto*

define *B* **where** *B* = *A* ∩ {(*x*,*y*). *abs*(*y*) ≤ *N*}
have *B-meas* [*measurable*]: *B* ∈ *sets* (*M* $\bigotimes_M$ *lborel*) **unfolding** *B-def* **by** *auto*
have *0* < *emeasure* (*M* $\bigotimes_M$ *lborel*) *B* **unfolding** *B-def* **using** *N* **by** *auto*
also have ... = (∫⁺*x*. *emeasure lborel* (*Pair x* −‘ *B*) ∂*M*)

apply (*rule sigma-finite-measure.emeasure-pair-measure-alt*)
using *B-meas* **by** (*auto simp add*: *lborel.sigma-finite-measure-axioms*)
finally have ∗: $(\int^{+} x.$ *emeasure lborel* (*Pair x* −' *B*) ∂*M*) > *0* **by** *simp*

have ∃ *Cx*∈*sets M*. ∃ *e*::*real*>*0*. *emeasure M Cx* > *0* ∧ (∀ *x* ∈ *Cx*. *emeasure lborel* (*Pair x* −' *B*) ≥ *e*)
by (*rule not-AE-zero-int-ennreal-E*, *auto simp add*: ∗)
then obtain *Cx e* **where** [*measurable*]: *Cx* ∈ *sets M* **and** *Cxe*: *e*>(*0*::*real*) *emeasure M Cx* > *0* ⋀*x*. *x* ∈ *Cx* ⟹ *emeasure lborel* (*Pair x* −' *B*) ≥ *e*
by *blast*
define *C* **where** *C* = *B* ∩ (*Cx* × (*UNIV*::*real set*))
have *C-meas* [*measurable*]: *C* ∈ *sets* (*M* $\bigotimes_M$ *lborel*) **unfolding** *C-def* **using** *B-meas* **by** *auto*
have *Cx-fibers*: ⋀*x*. *x* ∈ *Cx* ⟹ *emeasure lborel* (*Pair x* −' *C*) ≥ *e* **using** *Cxe*(*3*) *C-def* **by** *auto*

define *c* **where** *c* = (*measure M Cx*)/*2*
have *c* > *0* **unfolding** *c-def* **using** *Cxe*(*2*) **by** (*simp add*: *emeasure-eq-measure*)

We will apply Birkhoff theorem to show that most preimages of C at time n are contained in a cylinder of height roughly rn, for some suitably small r. How small r should be to get a contradiction can be determined at the end of the proof. It turns out that the good condition is the following one – this is by no means obvious now.

define *r* **where** *r* = (*if measure M* (*space M*) = *0 then 1 else e* ∗ *c* / (*4* ∗ *measure M* (*space M*)))
have *r* > *0* **using** ‹*e* > *0*› ‹*c* > *0*› **unfolding** *r-def*
apply *auto* **using** *measure-le-0-iff* **by** *fastforce*
have *pos*: *e*∗*c*−*2*∗*r*∗*measure M* (*space M*) > *0* **using** ‹*e* > *0*› ‹*c* > *0*› **unfolding** *r-def* **by** *auto*

define *Bgood* **where** *Bgood* = {*x* ∈ *space M*. (λ*n*. *birkhoff-sum f n x* / *n*) ⟶ *0*}
have [*measurable*]: *Bgood* ∈ *sets M* **unfolding** *Bgood-def* **by** *auto*
have ∗: *AE x in M*. *x* ∈ *Bgood* **unfolding** *Bgood-def* **using** *birkhoff-theorem-AE-nonergodic*[*OF assms*(*1*)] *assms*(*2*) **by** *auto*
then have *emeasure M Bgood* = *emeasure M* (*space M*)
by (*intro emeasure-eq-AE*) *auto*

{
fix *x* **assume** *x* ∈ *Bgood*
then have *x* ∈ *space M* **unfolding** *Bgood-def* **by** *auto*
have (λ*n*. *birkhoff-sum f n x* / *n*) ⟶ *0* **using** ‹*x* ∈ *Bgood*› **unfolding** *Bgood-def* **by** *auto*
moreover have *0* ∈ {−*r*<..<*r*} *open* {−*r*<..<*r*} **using** ‹*r*>*0*› **by** *auto*
ultimately have *eventually* (λ*n*. *birkhoff-sum f n x* / *n* ∈ {−*r*<..<*r*}) *sequentially*
using *topological-tendstoD* **by** *blast*
then obtain *n0* **where** *n0*: *n0*>*0* ⋀*n*. *n* ≥ *n0* ⟹ *birkhoff-sum f n x* / *n* ∈

$\{-r<..<r\}$
using *eventually-sequentially* **by** (*metis* (*mono-tags*, *lifting*) *le0 le-simps*(*3*) *neq0-conv*)
{
fix *n* **assume** $n \geq n0$
then have *n>0* **using** ‹*n0>0*› **by** *auto*
with *n0*(*2*)[*OF* ‹$n \geq n0$›] **have** *abs*(*birkhoff-sum f n x / n*) $\leq$ *r* **by** *auto*
then have *abs*(*birkhoff-sum f n x*) $\leq r * n$ **using** ‹*n>0*› **by** (*simp add*: *divide-le-eq*)
}
then have $x \in (\bigcup n0.\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\})$ **using** ‹$x \in space\ M$› **by** *blast*
}
then have $AE\ x\ in\ M.\ x \notin space\ M - (\bigcup n0.\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\})$
using $*$ **by** *auto*
then have *eqM*: $emeasure\ M\ (\bigcup n0.\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\}) = emeasure\ M\ (space\ M)$
by (*intro emeasure-eq-AE*) *auto*

have $(\lambda n0.\ emeasure\ M\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\} + c)$
$\longrightarrow emeasure\ M\ (\bigcup n0.\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\}) + c$
by (*intro tendsto-intros Lim-emeasure-incseq*) (*auto simp add*: *incseq-def*)
moreover have $emeasure\ M\ (\bigcup n0.\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\}) + c > emeasure\ M\ (space\ M)$
using *eqM* ‹$c > 0$› *emeasure-eq-measure* **by** *auto*
ultimately have $eventually\ (\lambda n0.\ emeasure\ M\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\} + c > emeasure\ M\ (space\ M))\ sequentially$
unfolding *order-tendsto-iff* **by** *auto*
then obtain *n0* **where** *n0*: $emeasure\ M\ \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\} + c > emeasure\ M\ (space\ M)$
using *eventually-sequentially* **by** *auto*

define *Dx* **where** $Dx = \{x \in space\ M.\ \forall n\in\{n0..\}.\ abs(birkhoff\text{-}sum\ f\ n\ x) \leq r * n\}$
have *Dx-meas* [*measurable*]: $Dx \in sets\ M$ **unfolding** *Dx-def* **by** *auto*
have $emeasure\ M\ Dx + c \geq emeasure\ M\ (space\ M)$ **using** *n0 Dx-def* **by** *auto*

obtain *n1*::*nat* **where** *n1*: $n1 > max\ n0\ ((measure\ M\ (space\ M) * 2 * N + e*c*n0 - e*c) / (e*c-2*r*measure\ M\ (space\ M)))$
by (*metis mult.commute mult.left-neutral numeral-One reals-Archimedean3 zero-less-numeral*)
then have $n1 > n0$ **by** *auto*
have *n1-ineq*: $n1 * (e*c-2*r*measure\ M\ (space\ M)) > (measure\ M\ (space\ M) * 2 * N + e*c*n0 - e*c)$
using *n1 pos* **by** (*simp add*: *pos-divide-less-eq*)

define *D* **where** $D = (\lambda n.\ Dx \times \{-r*n1-N..r*n1+N\} \cap (?TS\widehat{\ }\widehat{\ }n)-`C)$
have *Dn-meas* [*measurable*]: $D\ n \in sets\ (M \bigotimes_M lborel)$ **for** *n*
unfolding *D-def* **apply** (*rule TS.T-intersec-meas*(*2*)) **using** *C-meas* **by** *auto*

have *emeasure ?MS* $(D\ n) \geq e * c$ **if** $n \in \{n0..n1\}$ **for** *n*
proof −
have $n \geq n0\ n \leq n1$ **using** *that* **by** *auto*
{
fix *x* **assume** [*simp*]: $x \in space\ M$

define *F* **where** $F = \{y \in \{-r*n1-N..r*n1+N\}.\ y + birkhoff\text{-}sum\ f\ n\ x \in Pair\ ((T\widehat{\ }\widehat{\ }n)x) -`C\}$
have [*measurable*]: $F \in sets\ lborel$ **unfolding** *F-def* **by** *measurable*
{
fix *y*::*real*
have $(?TS\widehat{\ }\widehat{\ }n)(x,y) = ((T\widehat{\ }\widehat{\ }n)x,\ y + birkhoff\text{-}sum\ f\ n\ x)$
using *skew-product-real-iterates* **by** *simp*
then have $(indicator\ C\ ((?TS\widehat{\ }\widehat{\ }n)\ (x,y))::ennreal) = indicator\ Cx\ ((T\widehat{\ }\widehat{\ }n)x) * indicator\ (Pair\ ((T\widehat{\ }\widehat{\ }n)x) -`C)\ (y + birkhoff\text{-}sum\ f\ n\ x)$
using *C-def* **by** (*simp add*: *indicator-def*)
moreover have $(indicator\ (D\ n)\ (x,\ y)::ennreal) = indicator\ Dx\ x * indicator\ \{-r*n1-N..r*n1+N\}\ y * indicator\ C\ ((?TS\widehat{\ }\widehat{\ }n)\ (x,y))$
unfolding *D-def* **by** (*simp add*: *indicator-def*)
ultimately have $(indicator\ (D\ n)\ (x,\ y)::ennreal) = indicator\ Dx\ x * indicator\ \{-r*n1-N..r*n1+N\}\ y$
$* indicator\ Cx\ ((T\widehat{\ }\widehat{\ }n)x) * indicator\ (Pair\ ((T\widehat{\ }\widehat{\ }n)x) -`C)\ (y + birkhoff\text{-}sum\ f\ n\ x)$
by (*simp add*: *mult.assoc*)
then have $(indicator\ (D\ n)\ (x,\ y)::ennreal) = indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x * indicator\ F\ y$
unfolding *F-def* **by** (*simp add*: *indicator-def*)
}
then have $(\int^+ y.\ indicator\ (D\ n)\ (x,\ y)\ \partial lborel) = (\int^+ y.\ indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x * indicator\ F\ y\ \partial lborel)$
by *auto*
also have ... $= indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x * (\int^+ y.\ indicator\ F\ y\ \partial lborel)$
by (*rule nn-integral-cmult*, *auto*)
also have ... $= indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x * emeasure\ lborel\ F$ **using** ‹$F \in sets\ lborel$› **by** *auto*
finally have *A*: $(\int^+ y.\ indicator\ (D\ n)\ (x,\ y)\ \partial lborel) = indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x * emeasure\ lborel\ F$
by *simp*

have $(\int^+ y.\ indicator\ (D\ n)\ (x,\ y)\ \partial lborel) \geq ennreal\ e * indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x$
proof (*cases*)
assume $indicator\ (Dx \cap (T\widehat{\ }\widehat{\ }n)-`Cx)\ x = (0::ennreal)$
then show *?thesis* **by** *auto*
next

assume ¬(*indicator* (*Dx* ∩ (*T*^^*n*)−'*Cx*) *x* = (*0*::*ennreal*))
then have *x* ∈ *Dx* ∩ (*T*^^*n*)−'*Cx* **by** (*simp add*: *indicator-eq-0-iff*)
then have *x* ∈ *Dx* (*T*^^*n*) *x* ∈ *Cx* **by** *auto*
then have *abs*(*birkhoff-sum f n x*) ≤ *r* ∗ *n* **using** ‹*n* ∈ {*n0*..*n1*}› *Dx-def* **by** *auto*
then have ∗: *abs*(*birkhoff-sum f n x*) ≤ *r* ∗ *n1* **using** ‹*n* ≤ *n1*› ‹*r*>*0*›
by (*meson of-nat-le-iff order-trans mult-le-cancel-left-pos*)

have *F-expr*: *F* = {−*r*∗*n1*−*N*..*r*∗*n1*+*N*} ∩ (+)(*birkhoff-sum f n x*) −' (*Pair* ((*T*^^*n*)*x*) −'*C*)
unfolding *F-def* **by** (*auto simp add*: *add.commute*)
have (*Pair* ((*T*^^*n*)*x*) −'*C*) ⊆ {*real-of-int* (− *int N*)..*real N*} **unfolding** *C-def B-def* **by** *auto*
then have ((+)(*birkhoff-sum f n x*)) −' (*Pair* ((*T*^^*n*)*x*) −'*C*) ⊆ {−*N*−*birkhoff-sum f n x*..*N*−*birkhoff-sum f n x*}
by *auto*
also have ... ⊆ {−*r* ∗ *n1* − *N* .. *r* ∗ *n1* + *N*} **using** ∗ **by** *auto*
finally have *F* = ((+)(*birkhoff-sum f n x*)) −' (*Pair* ((*T*^^*n*)*x*) −'*C*) **unfolding** *F-expr* **by** *auto*

then have *emeasure lborel F* = *emeasure lborel* ((+)(*birkhoff-sum f n x*) −' (*Pair* ((*T*^^*n*)*x*) −'*C*)) **by** *auto*
also have ... = *emeasure lborel* (((+)(*birkhoff-sum f n x*) −' (*Pair* ((*T*^^*n*)*x*) −'*C*)) ∩ *space lborel*) **by** *simp*
also have ... = *emeasure* (*distr lborel borel* ((+) (*birkhoff-sum f n x*))) (*Pair* ((*T*^^*n*)*x*) −'*C*)
apply (*rule emeasure-distr*[*symmetric*]) **using** *C-meas* **by** *auto*
also have ... = *emeasure lborel* (*Pair* ((*T*^^*n*)*x*) −'*C*) **using** *lborel-distr-plus*[*of birkhoff-sum f n x*] **by** *simp*
also have ... ≥ *e* **using** *Cx-fibers* ‹(*T*^^*n*) *x* ∈ *Cx*› **by** *auto*
finally have *emeasure lborel F* ≥ *e* **by** *auto*
then show *?thesis* **using** *A* **by** (*simp add*: *indicator-def*)
qed
}
moreover have *emeasure ?MS* (*D n*) = (∫⁺*x*. (∫⁺*y*. *indicator* (*D n*) (*x*, *y*) ∂*lborel*) ∂*M*)
using *Dn-meas lborel.emeasure-pair-measure* **by** *blast*
ultimately have *emeasure ?MS* (*D n*) ≥ (∫⁺*x*. *ennreal e* ∗ *indicator* (*Dx* ∩ (*T* ^^ *n*) −' *Cx*) *x* ∂*M*)
by (*simp add*: *nn-integral-mono*)
also have (∫⁺*x*. *ennreal e* ∗ *indicator* (*Dx* ∩ (*T* ^^ *n*) −' *Cx*) *x* ∂*M*) = *e* ∗ (∫⁺*x*. *indicator* (*Dx* ∩ (*T* ^^ *n*) −' *Cx*) *x* ∂*M*)
apply (*rule nn-integral-cmult*) **using** ‹*e*>*0*› **by** *auto*
also have ... = *ennreal e* ∗ *emeasure M* (*Dx* ∩ (*T* ^^ *n*) −' *Cx*) **by** *simp*
finally have ∗: *emeasure ?MS* (*D n*) ≥ *ennreal e* ∗ *emeasure M* (*Dx* ∩ (*T* ^^ *n*) −' *Cx*) **by** *auto*

have *c* + *emeasure M* (*space M*) ≤ *emeasure M Dx* + *emeasure M Cx*
using ‹*emeasure M Dx* + *c* ≥ *emeasure M* (*space M*)› **unfolding** *c-def*

by (*auto simp*: *emeasure-eq-measure ennreal-plus*[*symmetric*] *simp del*: *ennreal-plus*)
also have ... = *emeasure M Dx* + *emeasure M* ((*T*^^*n*)−−'*Cx*)
by (*simp add*: *T-vrestr-same-emeasure*(*2*))
also have ... = *emeasure M* (*Dx* ∪ ((*T*^^*n*)−−'*Cx*)) + *emeasure M* (*Dx* ∩ ((*T*^^*n*)−−'*Cx*))
by (*rule emeasure-Un-Int*, *auto*)
also have ... ≤ *emeasure M* (*space M*) + *emeasure M* (*Dx* ∩ ((*T*^^*n*)−'*Cx*))
proof −
have *emeasure M* (*Dx* ∪ ((*T*^^*n*)−−'*Cx*)) ≤ *emeasure M* (*space M*)
by (*rule emeasure-mono*, *auto simp add*: *sets.sets-into-space*)
moreover have *Dx* ∩ ((*T*^^*n*)−−'*Cx*) = *Dx* ∩ ((*T*^^*n*)−'*Cx*)
by (*simp add*: *vrestr-intersec-in-space*)
ultimately show *?thesis* **by** (*metis add.commute add-left-mono*)
qed
finally have *emeasure M* (*Dx* ∩ ((*T*^^*n*)−'*Cx*)) ≥ *c* **by** (*simp add*: *emeasure-eq-measure*)
then have *ennreal e* ∗ *emeasure M* (*Dx* ∩ ((*T*^^*n*)−'*Cx*)) ≥ *ennreal e* ∗ *c* **using** ‹*e* > *0*›
using *mult-left-mono* **by** *fastforce*
with ∗ **show** *emeasure ?MS* (*D n*) ≥ *e* ∗ *c*
using ‹*0*<*c*› ‹*0*<*e*› **by** (*auto simp*: *ennreal-mult*[*symmetric*])
qed

have ¬(*disjoint-family-on D* {*n0*..*n1*})
proof
assume *D*: *disjoint-family-on D* {*n0*..*n1*}
have *emeasure lborel* {−*r*∗*n1*−*N*..*r*∗*n1*+*N*} = (*r* ∗ *real n1* + *real N*) − (− *r* ∗ *real n1* − *real N*)
apply (*rule emeasure-lborel-Icc*) **using** ‹*r*>*0*› **by** *auto*
then have ∗: *emeasure lborel* {−*r*∗*n1*−*N*..*r*∗*n1*+*N*} = *ennreal*(*2* ∗ *r* ∗ *real n1* + *2* ∗ *real N*)
by (*auto simp*: *ac-simps*)

have *ennreal*(*e* ∗ *c*) ∗ (*real n1* − *real n0* + *1*) = *ennreal*(*e*∗*c*) ∗ *card* {*n0*..*n1*}
using ‹*n1* > *n0*› **by** (*auto simp*: *ennreal-of-nat-eq-real-of-nat Suc-diff-le ac-simps of-nat-diff*)
also have ... = (∑ *n*∈{*n0*..*n1*}. *ennreal*(*e*∗*c*))
by (*simp add*: *ac-simps*)
also have ... ≤ (∑ *n*∈{*n0*..*n1*}. *emeasure ?MS* (*D n*))
using ‹⋀*n*. *n* ∈ {*n0*..*n1*} ⟹ *emeasure ?MS* (*D n*) ≥ *e* ∗ *c*› **by** (*meson sum-mono*)
also have ... = *emeasure ?MS* (⋃ *n*∈{*n0*..*n1*}. *D n*)
apply (*rule sum-emeasure*) **using** *Dn-meas* **by** (*auto simp add*: *D*)
also have ... ≤ *emeasure ?MS* (*space M* × {−*r*∗*n1*−*N*..*r*∗*n1*+*N*})
apply (*rule emeasure-mono*) **unfolding** *D-def* **using** *sets.sets-into-space*[*OF Dx-meas*] **by** *auto*
also have ... = *emeasure M* (*space M*) ∗ *emeasure lborel* {−*r*∗*n1*−*N*..*r*∗*n1*+*N*}
by (*rule sigma-finite-measure.emeasure-pair-measure-Times*, *auto simp add*:

lborel.sigma-finite-measure-axioms)
also have ... = *emeasure M* (*space M*) ∗ *ennreal*(*2* ∗ *r* ∗ *real n1* + *2* ∗ *real N*)
using ∗ **by** *auto*
finally have *ennreal*(*e* ∗ *c*) ∗ (*real n1*− *real n0*+*1*) ≤ *emeasure M* (*space M*) ∗ *ennreal*(*2* ∗ *r* ∗ *real n1* + *2* ∗ *real N*) **by** *simp*
then have *e*∗*c* ∗ (*real n1*− *real n0* + *1*) ≤ *measure M* (*space M*) ∗ (*2* ∗ *r* ∗ *real n1* + *2* ∗ *real N*)
using ‹*0*<*r*› ‹*0*<*e*› ‹*0*<*c*› ‹*n0* < *n1*› *emeasure-eq-measure* **by** (*auto simp*: *ennreal-mult′*[*symmetric*] *simp del*: *ennreal-plus*)
then have *0* ≤ *measure M* (*space M*) ∗ (*2* ∗ *r* ∗ *real n1* + *2* ∗ *real N*) − *e*∗*c* ∗ (*real n1*− *real n0* + *1*) **by** *auto*
also have ... = (*measure M* (*space M*) ∗ *2* ∗ *N* + *e*∗*c*∗*n0* − *e*∗*c*) − *n1* ∗ (*e*∗*c*−*2*∗*r*∗*measure M* (*space M*))
by *algebra*
finally have *n1* ∗ (*e*∗*c*−*2*∗*r*∗*measure M* (*space M*)) ≤ *measure M* (*space M*) ∗ *2* ∗ *N* + *e*∗*c*∗*n0* − *e*∗*c*
by *linarith*
then show *False* **using** *n1-ineq* **by** *auto*
qed
then obtain *n m* **where** *nm*: *n*<*m D m* ∩ *D n* ≠ {} **unfolding** *disjoint-family-on-def* **by** (*metis inf-sup-aci*(*1*) *linorder-cases*)
define *k* **where** *k* = *m*−*n*
then have *k*>*0 D* (*n*+*k*) ∩ *D n* ≠ {} **using** *nm* **by** *auto*
then have ((*?TS*^^(*n*+*k*))−‘*A*) ∩ ((*?TS*^^*n*)−‘*A*) ≠ {} **unfolding** *D-def C-def B-def* **by** *auto*
moreover have ((*?TS*^^(*n*+*k*))−‘*A*) ∩ ((*?TS*^^*n*)−‘*A*) = (*?TS*^^*n*)−‘(((*?TS*^^*k*)−‘*A*) ∩ *A*)
using *funpow-add* **by** (*simp add*: *add.commute funpow-add set.compositionality*)
ultimately have ((*?TS*^^*k*)−‘*A*) ∩ *A* ≠ {} **by** *auto*
then show ∃ *k*>*0*. ((*?TS*^^*k*)−‘*A*) ∩ *A* ≠ {} **using** ‹*k*>*0*› **by** *auto*
qed

6.2.4 Oscillations around the limit in Birkhoff theorem

In this paragraph, we prove that, in Birkhoff theorem with vanishing limit, the Birkhoff sums are infinitely many times arbitrarily close to 0, both on the positive and the negative side.

In the ergodic case, this statement implies for instance that if the Birkhoff sums of an integrable function tend to infinity almost everywhere, then the integral of the function can not vanish, it has to be strictly positive (while Birkhoff theorem per se does not exclude the convergence to infinity, at a rate slower than linear). This converts a qualitative information (convergence to infinity at an unknown rate) to a quantitative information (linear convergence to infinity). This result (sometimes known as Atkinson's Lemma) has been reinvented many times, for instance by Kesten and by Guivarch. It plays an important role in the study of random products of matrices.

This is essentially a consequence of the conservativity of the corresponding skew-product, proved in `skew_product_conservative`. Indeed, this implies that, starting from a small set $X \times [-e/2, e/2]$, the skew-product comes back infinitely often to itself, which implies that the Birkhoff sums at these times are bounded by e.

To show that the Birkhoff sums come back to $[0, e]$ is a little bit more tricky. Argue by contradiction, and induce on $A \times [0, e/2]$ where A is the set of points where the Birkhoff sums don't come back to $[0, e]$. Then the second coordinate decreases strictly when one iterates the skew product, which is not compatible with conservativity.

lemma *birkhoff-sum-small-asymp-lemma*:
  assumes [*measurable*]: *integrable M f*
    and *AE x in M. real-cond-exp M Invariants f x = 0 e>(0::real)*
  shows *AE x in M. infinite {n. birkhoff-sum f n x ∈ {0..e}}*
proof −
  have [*measurable*]: *f ∈ borel-measurable M* **by** *auto*
  have [*measurable*]: *⋀N. {x ∈ space M. ∃N. ∀n∈{N..}. birkhoff-sum f n x ∉ {0..e}} ∈ sets M* **by** *auto*

  {
    fix *N* **assume** *N>(0::nat)*
    define *Ax* **where** *Ax = {x ∈ space M. ∀n∈{N..}. birkhoff-sum f n x ∉ {0..e}}*
    then have [*measurable*]: *Ax ∈ sets M* **by** *auto*
    define *A* **where** *A = Ax × {0..e/2}*
    then have *A-meas* [*measurable*]: *A ∈ sets (M ⨂_M lborel)* **by** *auto*

    define *TN* **where** *TN = T^^N*
    interpret *TN*: *fmpt M TN*
      unfolding *TN-def* **using** *fmpt-power* **by** *auto*
    define *fN* **where** *fN = birkhoff-sum f N*
    have *TN.birkhoff-sum fN n x = birkhoff-sum f (n∗N) x* **for** *n x*
    proof (*induction n*)
      case *0*
      then show *?case* **by** *auto*
    next
      case (*Suc n*)
      have *TN.birkhoff-sum fN (Suc n) x = TN.birkhoff-sum fN n x + fN ((TN^^n) x)*
        using *TN.birkhoff-sum-cocycle[of fN n 1]* **by** *auto*
      also have ... = *birkhoff-sum f (n∗N) x + birkhoff-sum f N ((TN^^n) x)*
        using *Suc.IH fN-def* **by** *auto*
      also have ... = *birkhoff-sum f (n∗N+N) x* **unfolding** *TN-def*
      by (*subst funpow-mult, subst mult.commute[of N n], rule birkhoff-sum-cocycle[of f n∗N N x, symmetric]*)
      finally show *?case* **by** (*simp add: add.commute*)
    qed
    then have *not0e*: *⋀x n. x ∈ Ax ⟹ n ≠ 0 ⟹ TN.birkhoff-sum fN n x ∉ {0..e}* **unfolding** *Ax-def* **by** *auto*

```
  let ?TS = (λ(x,y). (T x, y + f x))
  let ?MS = M ⨂M (lborel::real measure)
  interpret TS: conservative-mpt ?MS ?TS
    by (rule skew-product-conservative, auto simp add: assms)

  let ?TSN = (λ(x,y). (TN x, y + fN x))
 have *:?TSN = ?TS^^N unfolding TN-def fN-def using skew-product-real-iterates
by auto
  interpret TSN: conservative-mpt ?MS ?TSN
    using * TS.conservative-mpt-power by auto

 define MA TA where MA = restrict-space ?MS A and TA = TSN.induced-map
A
  interpret TA: conservative-mpt MA TA unfolding MA-def TA-def
    by (rule TSN.induced-map-conservative-mpt, measurable)
 have *: ⋀ x y. snd (TA (x,y)) = snd (x,y) + TN.birkhoff-sum fN (TSN.return-time-function
A (x,y)) x
   unfolding TA-def TSN.induced-map-def using TN.skew-product-real-iterates
Pair-def by auto
  have [measurable]: snd ∈ borel-measurable ?MS by auto
  then have [measurable]: snd ∈ borel-measurable MA unfolding MA-def using
measurable-restrict-space1 by blast

  have AE z in MA. z ∈ TSN.recurrent-subset A
   unfolding MA-def using TSN.induced-map-recurrent-typical(1)[OF A-meas].
  moreover
  {
    fix z assume z: z ∈ TSN.recurrent-subset A
    define x y where x = fst z and y = snd z
    then have z = (x,y) by simp
    have z ∈ A using z TSN.recurrent-subset-incl(1) by auto
    then have x ∈ Ax y ∈ {0..e/2} unfolding A-def x-def y-def by auto
    define y2 where y2 = y + TN.birkhoff-sum fN (TSN.return-time-function
A (x,y)) x
    have y2 = snd (TA z) unfolding y2-def using * ‹z = (x, y)› by force
   moreover have TA z ∈ A unfolding TA-def using ‹z ∈ A› TSN.induced-map-stabilizes-A
by auto
    ultimately have y2 ∈ {0..e/2} unfolding A-def by auto

    have TSN.return-time-function A (x,y) ≠ 0
     using ‹z = (x,y)› ‹z ∈ TSN.recurrent-subset A› TSN.return-time0[of A] by
fast
   then have TN.birkhoff-sum fN (TSN.return-time-function A (x,y)) x ∉ {0..e}
      using not0e[OF ‹x ∈ Ax›] by auto
     moreover have TN.birkhoff-sum fN (TSN.return-time-function A (x,y)) x
∈ {−e..e}
      using ‹y ∈ {0..e/2}› ‹y2 ∈ {0..e/2}› y2-def by auto
    ultimately have TN.birkhoff-sum fN (TSN.return-time-function A (x,y)) x
∈ {−e..<0}
```

by *auto*
then have *y2* < *y* **using** *y2-def* **by** *auto*
then have *snd*(*TA z*) < *snd z* **unfolding** *y-def* **using** ‹*y2* = *snd* (*TA z*)›
by *auto*
}
ultimately have *a*: *AE z in MA*. *snd*(*TA z*) < *snd z* **by** *auto*
then have *AE z in MA*. *snd*(*TA z*) ≤ *snd z* **by** *auto*
then have *AE z in MA*. *snd*(*TA z*) = *snd z* **using** *TA.AE-decreasing-then-invariant*[*of snd*] **by** *auto*
then have *AE z in MA*. *False* **using** *a* **by** *auto*
then have *space MA* ∈ *null-sets MA* **by** (*simp add*: *AE-iff-null-sets*)
then have *emeasure MA A* = *0* **by** (*metis A-meas MA-def null-setsD1 space-restrict-space2*)
then have *emeasure ?MS A* = *0* **unfolding** *MA-def*
by (*metis A-meas emeasure-restrict-space sets.sets-into-space sets.top space-restrict-space space-restrict-space2*)
moreover have *emeasure ?MS A* = *emeasure M Ax* ∗ *emeasure lborel* {*0*..*e*/*2*}
unfolding *A-def* **by** (*intro lborel.emeasure-pair-measure-Times*) *auto*
ultimately have *emeasure M* {*x* ∈ *space M*. ∀ *n*∈{*N*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}} = *0* **using** ‹*e*>*0*› *Ax-def* **by** *simp*
then have {*x* ∈ *space M*. ∀ *n*∈{*N*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}} ∈ *null-sets M* **by** *auto*
}
then have (⋃ *N*∈{*0*<..}. {*x* ∈ *space M*. ∀ *n*∈{*N*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}}) ∈ *null-sets M* **by** (*auto simp*: *greaterThan-0*)
moreover have {*x* ∈ *space M*. ¬(*infinite* {*n*. *birkhoff-sum f n x* ∈ {*0*..*e*}})} ⊆ (⋃ *N*∈{*0*<..}. {*x* ∈ *space M*. ∀ *n*∈{*N*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}})
proof
fix *x* **assume** *H*: *x* ∈ {*x* ∈ *space M*. ¬(*infinite* {*n*. *birkhoff-sum f n x* ∈ {*0*..*e*}})}
then have *x* ∈ *space M* **by** *auto*
have ∗: *finite* {*n*. *birkhoff-sum f n x* ∈ {*0*..*e*}} **using** *H* **by** *auto*
then obtain *N* **where** ⋀*n*. *n* ≥ *N* ⟹ *n* ∉ {*n*. *birkhoff-sum f n x* ∈ {*0*..*e*}}
by (*metis finite-nat-set-iff-bounded not-less*)
then have *x* ∈ {*x* ∈ *space M*. ∀ *n*∈{*N*+*1*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}}
using ‹*x* ∈ *space M*› **by** *auto*
moreover have *N*+*1*>*0* **by** *auto*
ultimately show *x* ∈ (⋃ *N*∈{*0*<..}. {*x* ∈ *space M*. ∀ *n*∈{*N*..}. *birkhoff-sum f n x* ∉ {*0*..*e*}}) **by** *auto*
qed
ultimately show *?thesis* **unfolding** *eventually-ae-filter* **by** *auto*
qed

theorem *birkhoff-sum-small-asymp-pos-nonergodic*:
assumes [*measurable*]: *integrable M f* **and** *e* > (*0*::*real*)
shows *AE x in M*. *infinite* {*n*. *birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* .. *n* ∗ *real-cond-exp M Invariants f x* + *e*}}
proof −
define *g* **where** *g* = (λ*x*. *f x* − *real-cond-exp M Invariants f x*)
have *g-meas* [*measurable*]: *integrable M g* **unfolding** *g-def* **using** *real-cond-exp-int*(*1*)[*OF assms*(*1*)] *assms*(*1*) **by** *auto*

have *AE x in M. real-cond-exp M Invariants* (*real-cond-exp M Invariants f*) *x* = *real-cond-exp M Invariants f x*
by (*rule real-cond-exp-F-meas*, *auto simp add*: *real-cond-exp-int*(*1*)[*OF assms*(*1*)])
then have ∗: *AE x in M. real-cond-exp M Invariants g x* = *0*
unfolding *g-def* **using** *real-cond-exp-diff* [*OF assms*(*1*) *real-cond-exp-int*(*1*)[*OF assms*(*1*)]] **by** *auto*
have *AE x in M. infinite* {*n. birkhoff-sum g n x* ∈ {*0..e*}}
by (*rule birkhoff-sum-small-asymp-lemma*, *auto simp add*: ‹*e*>*0*› ∗ *g-meas*)
moreover
{
fix *x* **assume** *x* ∈ *space M infinite* {*n. birkhoff-sum g n x* ∈ {*0..e*}}
{
fix *n* **assume** *H*: *birkhoff-sum g n x* ∈ {*0..e*}
have *birkhoff-sum g n x* = *birkhoff-sum f n x* − *birkhoff-sum* (*real-cond-exp M Invariants f*) *n x*
unfolding *g-def* **using** *birkhoff-sum-diff* **by** *auto*
also have ... = *birkhoff-sum f n x* − *n* ∗ *real-cond-exp M Invariants f x*
using *birkhoff-sum-of-invariants* ‹*x* ∈ *space M*› **by** *auto*
finally have *birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* .. *n* ∗ *real-cond-exp M Invariants f x* + *e*} **using** *H* **by** *simp*
}
then have {*n. birkhoff-sum g n x* ∈ {*0..e*}} ⊆ {*n. birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* .. *n* ∗ *real-cond-exp M Invariants f x* + *e*}}
by *auto*
then have *infinite* {*n. birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* .. *n* ∗ *real-cond-exp M Invariants f x* + *e*}}
using ‹*infinite* {*n. birkhoff-sum g n x* ∈ {*0..e*}}› *finite-subset* **by** *blast*
}
ultimately show *?thesis* **by** *auto*
qed

theorem *birkhoff-sum-small-asymp-neg-nonergodic*:
assumes [*measurable*]: *integrable M f* **and** *e* > (*0*::*real*)
shows *AE x in M. infinite* {*n. birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* − *e* .. *n* ∗ *real-cond-exp M Invariants f x*}}
proof −
define *g* **where** *g* = (λ*x. real-cond-exp M Invariants f x* − *f x*)
have *g-meas* [*measurable*]: *integrable M g* **unfolding** *g-def* **using** *real-cond-exp-int*(*1*)[*OF assms*(*1*)] *assms*(*1*) **by** *auto*
have *AE x in M. real-cond-exp M Invariants* (*real-cond-exp M Invariants f*) *x* = *real-cond-exp M Invariants f x*
by (*rule real-cond-exp-F-meas*, *auto simp add*: *real-cond-exp-int*(*1*)[*OF assms*(*1*)])
then have ∗: *AE x in M. real-cond-exp M Invariants g x* = *0*
unfolding *g-def* **using** *real-cond-exp-diff* [*OF real-cond-exp-int*(*1*)[*OF assms*(*1*)] *assms*(*1*)] **by** *auto*
have *AE x in M. infinite* {*n. birkhoff-sum g n x* ∈ {*0..e*}}
by (*rule birkhoff-sum-small-asymp-lemma*, *auto simp add*: ‹*e*>*0*› ∗ *g-meas*)
moreover
{

fix *x* **assume** $x \in$ *space M infinite* {*n. birkhoff-sum g n x* $\in$ {*0..e*}}
{
fix *n* **assume** *H*: *birkhoff-sum g n x* $\in$ {*0..e*}
have *birkhoff-sum g n x* = *birkhoff-sum* (*real-cond-exp M Invariants f*) *n x* − *birkhoff-sum f n x*
unfolding *g-def* **using** *birkhoff-sum-diff* **by** *auto*
also have ... = *n* ∗ *real-cond-exp M Invariants f x* − *birkhoff-sum f n x*
using *birkhoff-sum-of-invariants* ‹*x* ∈ *space M*› **by** *auto*
finally have *birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* − *e* .. *n* ∗ *real-cond-exp M Invariants f x*} **using** *H* **by** *simp*
}
then have {*n. birkhoff-sum g n x* ∈ {*0..e*}} ⊆ {*n. birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* − *e* .. *n* ∗ *real-cond-exp M Invariants f x*}}
by *auto*
then have *infinite* {*n. birkhoff-sum f n x* ∈ {*n* ∗ *real-cond-exp M Invariants f x* − *e* .. *n* ∗ *real-cond-exp M Invariants f x*}}
using ‹*infinite* {*n. birkhoff-sum g n x* ∈ {*0..e*}}› *finite-subset* **by** *blast*
}
ultimately show *?thesis* **by** *auto*
qed

6.2.5 Conditional expectation for the induced map

Thanks to Birkhoff theorem, one can relate conditional expectations with respect to the invariant sigma algebra, for a map and for a corresponding induced map, as follows.

proposition *Invariants-cond-exp-induced-map*:
fixes *f*::*'a* ⇒ *real*
assumes [*measurable*]: *A* ∈ *sets M integrable M f*
defines *MA* ≡ *restrict-space M A* **and** *TA* ≡ *induced-map A* **and** *fA* ≡ *induced-function A f*
shows *AE x in MA. real-cond-exp MA* (*qmpt.Invariants MA TA*) *fA x*
= *real-cond-exp M Invariants f x* ∗ *real-cond-exp MA* (*qmpt.Invariants MA TA*) (*return-time-function A*) *x*
proof −
interpret *A*: *fmpt MA TA* **unfolding** *MA-def TA-def* **by** (*rule induced-map-fmpt*[*OF assms*(*1*)])

have *integrable M fA* **unfolding** *fA-def* **using** *induced-function-integral-nonergodic*(*1*) *assms* **by** *auto*
then have *integrable MA fA* **unfolding** *MA-def*
by (*metis assms*(*1*) *integrable-mult-indicator integrable-restrict-space sets.Int-space-eq2*)
then have *a*: *AE x in MA.* (λ*n. A.birkhoff-sum fA n x* / *n*) ⟶ *real-cond-exp MA A.Invariants fA x*
using *A.birkhoff-theorem-AE-nonergodic* **by** *auto*

have *AE x in M.* (λ*n. birkhoff-sum f n x* / *n*) ⟶ *real-cond-exp M Invariants f x*
using *birkhoff-theorem-AE-nonergodic assms*(*2*) **by** *auto*

then have *b*: *AE x in MA*. (*λn. birkhoff-sum f n x / n*) ⟶ *real-cond-exp M Invariants f x*
unfolding *MA-def* **by** (*metis* (*mono-tags*, *lifting*) *AE-restrict-space-iff assms*(*1*) *eventually-mono sets.Int-space-eq2*)

define *phiA* **where** *phiA* = (*λx. return-time-function A x*)
have *integrable M phiA* **unfolding** *phiA-def* **using** *return-time-integrable* **by** *auto*
then have *integrable MA phiA* **unfolding** *MA-def*
by (*metis assms*(*1*) *integrable-mult-indicator integrable-restrict-space sets.Int-space-eq2*)
then have *c*: *AE x in MA*. (*λn. A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n*) ⟶ *real-cond-exp MA A.Invariants phiA x*
using *A.birkhoff-theorem-AE-nonergodic* **by** *auto*

have *A*−*recurrent-subset A* ∈ *null-sets M* **using** *Poincare-recurrence-thm*(*1*)[*OF assms*(*1*)] **by** *auto*
then have *A* − *recurrent-subset A* ∈ *null-sets MA* **unfolding** *MA-def*
by (*metis Diff-subset assms*(*1*) *emeasure-restrict-space null-setsD1 null-setsD2 null-setsI sets.Int-space-eq2 sets-restrict-space-iff*)
then have *AE x in MA*. *x* ∈ *recurrent-subset A*
by (*simp add*: *AE-iff-null MA-def null-setsD2 set-diff-eq space-restrict-space2*)
moreover have ⋀*x*. *x* ∈ *recurrent-subset A* ⟹ *phiA x* > *0* **unfolding** *phiA-def* **using** *return-time0* **by** *fastforce*
ultimately have ∗: *AE x in MA*. *phiA x* > *0* **by** *auto*
have *d*: *AE x in MA*. *real-cond-exp MA A.Invariants phiA x* > *0*
by (*rule A.real-cond-exp-gr-c*, *auto simp add*: ∗ ‹*integrable MA phiA*›)

{
fix *x*
assume *A*: (*λn. A.birkhoff-sum fA n x / n*) ⟶ *real-cond-exp MA A.Invariants fA x*
and *B*: (*λn. birkhoff-sum f n x / n*) ⟶ *real-cond-exp M Invariants f x*
and *C*: (*λn. A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n*) ⟶ *real-cond-exp MA A.Invariants phiA x*
and *D*: *real-cond-exp MA A.Invariants phiA x* > *0*
define *R* **where** *R* = (*λn. A.birkhoff-sum phiA n x*)

have *D2*: *ereal*(*real-cond-exp MA A.Invariants phiA x*) > *0* **using** *D* **by** *simp*
have ⋀*n. real*(*R n*)/ *n* = *A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n* **unfolding** *R-def A.birkhoff-sum-def* **by** *auto*
moreover have (*λn. A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n*) ⟶ *real-cond-exp MA A.Invariants phiA x* **using** *C* **by** *auto*
ultimately have *Rnn*: (*λn. real*(*R n*)/*n*) ⟶ *real-cond-exp MA A.Invariants phiA x* **by** *presburger*

have ⋀*n. ereal*(*real*(*R n*))/ *n* = *ereal*(*A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n*) **unfolding** *R-def A.birkhoff-sum-def* **by** *auto*
moreover have (*λn. ereal*(*A.birkhoff-sum* (*λx. real*(*phiA x*)) *n x / n*)) ⟶ *real-cond-exp MA A.Invariants phiA x* **using** *C* **by** *auto*

ultimately have *i*: (*λn. ereal*(*real*(*R n*))/*n*) ⟶ *real-cond-exp MA A.Invariants phiA x* **by** *auto*
have *ii*: (*λn. real n*) ⟶ ∞ **by** (*rule id-nat-ereal-tendsto-PInf*)
have *iii*: (*λn. ereal*(*real*(*R n*))/*n* ∗ *real n*) ⟶ ∞ **using** *tendsto-mult-ereal-PInf*[*OF i D2 ii*] **by** *simp*
have ⋀*n. n* > *0* ⟹ *ereal*(*real*(*R n*))/*n* ∗ *real n* = *R n* **by** *auto*
then have *eventually* (*λn. ereal*(*real*(*R n*))/*n* ∗ *real n* = *R n*) *sequentially*
using *eventually-at-top-dense* **by** *blast*
then have (*λn. real*(*R n*)) ⟶ ∞ **using** *iii* **by** (*simp add*: *filterlim-cong*)
then have (*λn. birkhoff-sum f* (*R n*) *x* / (*R n*)) ⟶ *real-cond-exp M Invariants f x* **using** *limit-along-weak-subseq B* **by** *auto*
then have *l*: (*λn.* (*birkhoff-sum f* (*R n*) *x* / (*R n*)) ∗ ((*R n*)/*n*)) ⟶ *real-cond-exp M Invariants f x* ∗ *real-cond-exp MA A.Invariants phiA x*
by (*rule tendsto-mult*, *simp add*: *Rnn*)
obtain *N* **where** *N*: ⋀*n. n* > *N* ⟹ *R n* > *0* **using** ‹(*λn. real*(*R n*)) ⟶ ∞›
by (*metis* (*full-types*) *eventually-at-top-dense filterlim-iff filterlim-weak-subseq*)
then have ⋀*n. n* > *N* ⟹ (*birkhoff-sum f* (*R n*) *x* / (*R n*)) ∗ ((*R n*)/*n*) = *birkhoff-sum f* (*R n*) *x* / *n*
by *auto*
then have *eventually* (*λn.* (*birkhoff-sum f* (*R n*) *x* / (*R n*)) ∗ ((*R n*)/*n*) = *birkhoff-sum f* (*R n*) *x* / *n*) *sequentially*
by *simp*
with *tendsto-cong*[*OF this*] **have** *main-limit*: (*λn. birkhoff-sum f* (*R n*) *x* / *n*) ⟶ *real-cond-exp M Invariants f x* ∗ *real-cond-exp MA A.Invariants phiA x*
using *l* **by** *auto*
have ⋀*n. birkhoff-sum f* (*R n*) *x* = *A.birkhoff-sum fA n x*
unfolding *R-def fA-def phiA-def TA-def* **using** *induced-function-birkhoff-sum*[*OF assms*(*1*)] **by** *simp*
then have ⋀*n. birkhoff-sum f* (*R n*) *x* /*n* = *A.birkhoff-sum fA n x* / *n* **by** *simp*
then have (*λn. A.birkhoff-sum fA n x* / *n*) ⟶ *real-cond-exp M Invariants f x* ∗ *real-cond-exp MA A.Invariants phiA x*
using *main-limit* **by** *presburger*
then have *real-cond-exp MA A.Invariants fA x* = *real-cond-exp M Invariants f x* ∗ *real-cond-exp MA A.Invariants phiA x*
using *A LIMSEQ-unique* **by** *auto*
}
then show *?thesis* **using** *a b c d* **unfolding** *phiA-def* **by** *auto*
qed

corollary *Invariants-cond-exp-induced-map-0*:
fixes *f*::′*a* ⇒ *real*
assumes [*measurable*]: *A* ∈ *sets M integrable M f* **and** *AE x in M. real-cond-exp M Invariants f x* = *0*
defines *MA* ≡ *restrict-space M A* **and** *TA* ≡ *induced-map A* **and** *fA* ≡ *induced-function A f*
shows *AE x in MA. real-cond-exp MA* (*qmpt.Invariants MA TA*) *fA x* = *0*
proof −

have *AE x in MA. real-cond-exp M Invariants f x = 0* **unfolding** *MA-def*
apply (*subst AE-restrict-space-iff*) **using** *assms*(*3*) **by** *auto*
then show *?thesis* **unfolding** *MA-def TA-def fA-def* **using** *Invariants-cond-exp-induced-map*[*OF assms*(*1*) *assms*(*2*)]
by *auto*
qed

end
end

7 Ergodicity

theory *Ergodicity*
imports *Invariants*
begin

A transformation is *ergodic* if any invariant set has zero measure or full measure. Ergodic transformations are, in a sense, extremal among measure preserving transformations. Hence, any transformation can be seen as an average of ergodic ones. This can be made precise by the notion of ergodic decomposition, only valid on standard measure spaces.

Many statements get nicer in the ergodic case, hence we will reformulate many of the previous results in this setting.

7.1 Ergodicity locales

locale *ergodic-qmpt* = *qmpt* +
assumes *ergodic*: $\bigwedge A.\ A \in$ *sets Invariants* $\Longrightarrow$ ($A \in$ *null-sets M* $\vee$ *space M* $- A \in$ *null-sets M*)

locale *ergodic-mpt* = *mpt* + *ergodic-qmpt*

locale *ergodic-fmpt* = *ergodic-qmpt* + *fmpt*

locale *ergodic-pmpt* = *ergodic-qmpt* + *pmpt*

locale *ergodic-conservative* = *ergodic-qmpt* + *conservative*

locale *ergodic-conservative-mpt* = *ergodic-qmpt* + *conservative-mpt*

sublocale *ergodic-fmpt* ⊆ *ergodic-mpt*
by *unfold-locales*

sublocale *ergodic-pmpt* ⊆ *ergodic-fmpt*
by *unfold-locales*

sublocale *ergodic-fmpt* ⊆ *ergodic-conservative-mpt*
by *unfold-locales*

sublocale *ergodic-conservative-mpt* ⊆ *ergodic-conservative*
by *unfold-locales*

7.2 Behavior of sets in ergodic transformations

The main property of an ergodic transformation, essentially equivalent to the definition, is that a set which is almost invariant under the dynamics is null or conull.

lemma (**in** *ergodic-qmpt*) *AE-equal-preimage-then-null-or-conull*:
assumes [*measurable*]: $A \in$ *sets M* **and** $A \ \Delta\ T{-}{-}`A \in$ *null-sets M*
shows $A \in$ *null-sets M* ∨ *space M* − $A \in$ *null-sets M*
proof −
obtain *B* **where** *B*: $B \in$ *sets Invariants* $A \ \Delta\ B \in$ *null-sets M*
by (*metis Un-commute Invariants-quasi-Invariants-sets*[*OF assms*(*1*)] *assms*(*2*))
have [*measurable*]: $B \in$ *sets M* **using** *B*(*1*) **using** *Invariants-in-sets* **by** *blast*
have ∗: $B \in$ *null-sets M* ∨ *space M* − $B \in$ *null-sets M* **using** *ergodic B*(*1*) **by** *blast*
show *?thesis*
proof (*cases*)
assume $B \in$ *null-sets M*
then have $A \in$ *null-sets M* **by** (*metis Un-commute B*(*2*) *Delta-null-of-null-is-null*[*OF assms*(*1*), **where** *?A* = *B*])
then show *?thesis* **by** *simp*
next
assume ¬($B \in$ *null-sets M*)
then have *i*: *space M* − $B \in$ *null-sets M* **using** ∗ **by** *simp*
have (*space M* − *B*) Δ (*space M* − *A*) = *A* Δ *B*
using *sets.sets-into-space*[*OF* ‹$A \in$ *sets M*›] *sets.sets-into-space*[*OF* ‹$B \in$ *sets M*›] **by** *blast*
then have (*space M* − *B*) Δ (*space M* − *A*) ∈ *null-sets M* **using** *B*(*2*) **by** *auto*
then have *space M* − $A \in$ *null-sets M*
using *Delta-null-of-null-is-null*[**where** *?A* = *space M* − *B* **and** *?B* = *space M* − *A*] *i* **by** *auto*
then show *?thesis* **by** *simp*
qed
qed

The inverse of an ergodic transformation is also ergodic.

lemma (**in** *ergodic-qmpt*) *ergodic-Tinv*:
assumes *invertible-qmpt*
shows *ergodic-qmpt M Tinv*
unfolding *ergodic-qmpt-def ergodic-qmpt-axioms-def*
proof
show *qmpt M Tinv* **using** *Tinv-qmpt*[*OF assms*] **by** *simp*
show ∀ *A*. $A \in$ *sets* (*qmpt.Invariants M Tinv*) ⟶ $A \in$ *null-sets M* ∨ *space M* − $A \in$ *null-sets M*

proof (*intro allI impI*)
fix *A* **assume** $A \in$ *sets* (*qmpt.Invariants M Tinv*)
then have $A \in$ *sets Invariants* **using** *Invariants-Tinv*[*OF assms*] **by** *simp*
then show $A \in$ *null-sets M* $\vee$ *space M* $-$ $A \in$ *null-sets M* **using** *ergodic* **by** *auto*
qed
qed

In the conservative case, instead of the almost invariance of a set, it suffices to assume that the preimage is contained in the set, or contains the set, to deduce that it is null or conull.

lemma (**in** *ergodic-conservative*) *preimage-included-then-null-or-conull*:
assumes $A \in$ *sets M T*−−'*A* $\subseteq$ *A*
shows $A \in$ *null-sets M* $\vee$ *space M* $-$ $A \in$ *null-sets M*
proof −
have *A* Δ *T*−−'*A* $\in$ *null-sets M* **using** *preimage-included-then-almost-invariant*[*OF assms*] **by** *auto*
then show *?thesis* **using** *AE-equal-preimage-then-null-or-conull assms*(*1*) **by** *auto*
qed

lemma (**in** *ergodic-conservative*) *preimage-includes-then-null-or-conull*:
assumes $A \in$ *sets M T*−−'*A* $\supseteq$ *A*
shows $A \in$ *null-sets M* $\vee$ *space M* $-$ $A \in$ *null-sets M*
proof −
have *A* Δ *T*−−'*A* $\in$ *null-sets M* **using** *preimage-includes-then-almost-invariant*[*OF assms*] **by** *auto*
then show *?thesis* **using** *AE-equal-preimage-then-null-or-conull assms*(*1*) **by** *auto*
qed

lemma (**in** *ergodic-conservative*) *preimages-conull*:
assumes [*measurable*]: $A \in$ *sets M* **and** *emeasure M A* > 0
shows *space M* $-$ ($\bigcup$ *n*. (*T*^^*n*)−−'*A*) $\in$ *null-sets M*
space M Δ ($\bigcup$ *n*. (*T*^^*n*)−−'*A*) $\in$ *null-sets M*
proof −
define *B* **where** *B* = ($\bigcup$ *n*. (*T*^^*n*)−−'*A*)
then have [*measurable*]: $B \in$ *sets M* **by** *auto*
have *T*−−'*B* = ($\bigcup$ *n*. (*T*^^(*n*+*1*))−−'*A*) **unfolding** *B-def* **using** *T-vrestr-composed*(*2*) **by** *auto*
then have *T*−−'*B* $\subseteq$ *B* **using** *B-def* **by** *blast*
then have ∗: $B \in$ *null-sets M* $\vee$ *space M* $-$ $B \in$ *null-sets M*
using *preimage-included-then-null-or-conull* **by** *auto*
have $A \subseteq B$ **unfolding** *B-def* **using** *T-vrestr-0 assms*(*1*) **by** *blast*
then have *emeasure M B* > 0 **using** *assms*(*2*)
by (*metis* ‹$B \in$ *sets M*› *emeasure-eq-0 zero-less-iff-neq-zero*)
then have $B \notin$ *null-sets M* **by** *auto*
then have *i*: *space M* $-$ $B \in$ *null-sets M* **using** ∗ **by** *auto*
then show *space M* $-$ ($\bigcup$ *n*. (*T*^^*n*)−−'*A*) $\in$ *null-sets M* **using** *B-def* **by** *auto*

have *B* ⊆ *space M* **using** *sets.sets-into-space*[*OF* ‹*B* ∈ *sets M*›] **by** *auto*
then have *space M* Δ *B* ∈ *null-sets M* **using** *i* **by** (*simp add*: *Diff-mono sup.absorb1*)
then show *space M* Δ (⋃ *n*. (*T*^^*n*)−−‘*A*) ∈ *null-sets M* **using** *B-def* **by** *auto*
qed

7.3 Behavior of functions in ergodic transformations

In the same way that invariant sets are null or conull, invariant functions are almost everywhere constant in an ergodic transformation. For real functions, one can consider the set where $\{f x \geq d\}$, it has measure 0 or 1 depending on d. Then f is almost surely equal to the maximal d such that this set has measure 1. For functions taking values in a general space, the argument is essentially the same, replacing intervals by a basis of the topology.

lemma (**in** *ergodic-qmpt*) *Invariant-func-is-AE-constant*:
fixes *f*::- ⇒ ′*b*::{*second-countable-topology*, *t1-space*}
assumes *f* ∈ *borel-measurable Invariants*
shows ∃ *y*. *AE x in M*. *f x* = *y*
proof (*cases*)
assume *space M* ∈ *null-sets M*
obtain *y*::′*b* **where** *True* **by** *auto*
have *AE x in M*. *f x* = *y* **using** ‹*space M* ∈ *null-sets M*› *AE-I*′ **by** *blast*
then show *?thesis* **by** *auto*
next
assume ∗: ¬(*space M* ∈ *null-sets M*)
obtain *B*::′*b set set* **where** *B*: *countable B topological-basis B* **using** *ex-countable-basis* **by** *auto*
define *C* **where** *C* = {*b* ∈ *B*. *space M* − *f*−‘*b* ∈ *null-sets M*}
then have *countable C* **using** ‹*countable B*› **by** *auto*
define *Y* **where** *Y* = ⋂ *C*
have *space M* − *f*−‘*Y* = (⋃ *b*∈ *C*. *space M* − *f*−‘*b*) **unfolding** *Y-def* **by** *auto*
moreover have ⋀*b*. *b* ∈ *C* ⟹ *space M* − *f*−‘*b* ∈ *null-sets M* **unfolding** *C-def* **by** *blast*
ultimately have *i*: *space M* − *f*−‘*Y* ∈ *null-sets M* **using** ‹*countable C*› **by** (*metis null-sets-UN*′)
then have *f*−‘*Y* ≠ {} **using** ∗ **by** *auto*
then have *Y* ≠ {} **by** *auto*
then obtain *y* **where** *y* ∈ *Y* **by** *auto*
define *D* **where** *D* = {*b* ∈ *B*. *y*∉*b* ∧ *f*−‘*b* ∩ *space M* ∈ *null-sets M*}
have *countable D* **using** ‹*countable B*› *D-def* **by** *auto*
{
fix *z* **assume** *z* ≠ *y*
obtain *U* **where** *U*: *open U z* ∈ *U y* ∉ *U*
using *t1-space*[*OF* ‹*z* ≠ *y*›] **by** *blast*
obtain *V* **where** *V* ∈ *B V* ⊆ *U z* ∈ *V* **by** (*rule topological-basisE*[*OF* ‹*topological-basis B*› ‹*open U*› ‹*z* ∈ *U*›])
then have *y* ∉ *V* **using** *U* **by** *blast*

then have *V* ∉ *C* **using** ‹*y* ∈ *Y*› *Y-def* **by** *auto*
then have *space M* − *f*− ‘*V* ∩ *space M* ∉ *null-sets M* **unfolding** *C-def* **using** ‹*V* ∈ *B*›
by (*metis* (*no-types*, *lifting*) *Diff-Int2 inf.idem mem-Collect-eq*)
moreover have *f*− ‘*V* ∩ *space M* ∈ *sets Invariants*
using *measurable-sets*[*OF assms borel-open*[*OF topological-basis-open*[*OF B*(*2*) ‹*V* ∈ *B*›]]] *subalgebra-def Invariants-is-subalg* **by** *metis*
ultimately have *f*− ‘*V* ∩ *space M* ∈ *null-sets M* **using** *ergodic* **by** *auto*
then have *V* ∈ *D* **unfolding** *D-def* **using** ‹*V* ∈ *B*› ‹*y* ∉ *V*› **by** *auto*
then have ∃ *b* ∈ *D*. *z* ∈ *b* **using** ‹*z* ∈ *V*› **by** *auto*
}
then have ∗: ⋃ *D* = *UNIV* − {*y*}
apply *auto* **unfolding** *D-def* **by** *auto*
have *space M* − *f*− ‘{*y*} = *f*− ‘(*UNIV* −{*y*}) ∩ *space M* **by** *blast*
also have ... = (⋃ *b*∈*D*. *f*− ‘*b* ∩ *space M*) **using** ∗ **by** *auto*
also have ... ∈ *null-sets M* **using** *D-def* ‹*countable D*›
by (*metis* (*no-types*, *lifting*) *mem-Collect-eq null-sets-UN'*)
finally have *space M* − *f*− ‘{*y*} ∈ *null-sets M* **by** *blast*
with *AE-not-in*[*OF this*] **have** *AE x in M*. *x* ∈ *f*− ‘{*y*} **by** *auto*
then show *?thesis* **by** *auto*
qed

The same goes for functions which are only almost invariant, as they coindice almost everywhere with genuine invariant functions.

lemma (**in** *ergodic-qmpt*) *AE-Invariant-func-is-AE-constant*:
fixes *f*::- ⇒ *'b*::{*second-countable-topology*, *t2-space*}
assumes *f* ∈ *borel-measurable M AE x in M*. *f*(*T x*) = *f x*
shows ∃ *y*. *AE x in M*. *f x* = *y*
proof −
obtain *g* **where** *g*: *g* ∈ *borel-measurable Invariants AE x in M*. *f x* = *g x*
using *Invariants-quasi-Invariants-functions*[*OF assms*(*1*)] *assms*(*2*) **by** *auto*
then obtain *y* **where** *y*: *AE x in M*. *g x* = *y* **using** *Invariant-func-is-AE-constant* **by** *auto*
have *AE x in M*. *f x* = *y* **using** *g*(*2*) *y* **by** *auto*
then show *?thesis* **by** *auto*
qed

In conservative systems, it suffices to have an inequality between f and $f \circ T$, since such a function is almost invariant.

lemma (**in** *ergodic-conservative*) *AE-decreasing-func-is-AE-constant*:
fixes *f*::- ⇒ *'b*::{*linorder-topology*, *second-countable-topology*}
assumes *AE x in M*. *f*(*T x*) ≤ *f x*
and [*measurable*]: *f* ∈ *borel-measurable M*
shows ∃ *y*. *AE x in M*. *f x* = *y*
proof −
have *AE x in M*. *f*(*T x*) = *f x* **using** *AE-decreasing-then-invariant*[*OF assms*] **by** *auto*
then show *?thesis* **using** *AE-Invariant-func-is-AE-constant*[*OF assms*(*2*)] **by** *auto*

qed

lemma (**in** *ergodic-conservative*) *AE-increasing-func-is-AE-constant*:
fixes *f*::- ⇒ *'b*::{*linorder-topology*, *second-countable-topology*}
assumes *AE x in M*. *f*(*T x*) ≥ *f x*
and [*measurable*]: *f* ∈ *borel-measurable M*
shows ∃ *y*. *AE x in M*. *f x* = *y*
proof −
have *AE x in M*. *f*(*T x*) = *f x* **using** *AE-increasing-then-invariant*[*OF assms*] **by** *auto*
then show *?thesis* **using** *AE-Invariant-func-is-AE-constant*[*OF assms*(*2*)] **by** *auto*
qed

When the function takes values in a Banach space, the value of the invariant (hence constant) function can be recovered by integrating the function.

lemma (**in** *ergodic-fmpt*) *Invariant-func-integral*:
fixes *f*::- ⇒ *'b*::{*banach*, *second-countable-topology*}
assumes [*measurable*]: *f* ∈ *borel-measurable Invariants*
shows *integrable M f*
AE x in M. *f x* = ($\int$ *x*. *f x* ∂*M*)$/_R$ (*measure M* (*space M*))
proof −
have [*measurable*]: *f* ∈ *borel-measurable M* **using** *assms Invariants-measurable-func* **by** *blast*
obtain *y* **where** *y*: *AE x in M*. *f x* = *y* **using** *Invariant-func-is-AE-constant*[*OF assms*] **by** *auto*
moreover have *integrable M* (λ*x*. *y*) **by** *auto*
ultimately show *integrable M f* **by** (*subst integrable-cong-AE*[**where** *?g* = λ*x*. *y*], *auto*)

have ($\int$ *x*. *f x* ∂*M*) = ($\int$ *x*. *y* ∂*M*) **by** (*subst integral-cong-AE*[**where** *?g* = λ*x*. *y*], *auto simp add*: *y*)
also have ... = *measure M* (*space M*) $*_R$ *y* **by** *auto*
finally have ∗: ($\int$ *x*. *f x* ∂*M*) = *measure M* (*space M*) $*_R$ *y* **by** *simp*
show *AE x in M*. *f x* = ($\int$ *x*. *f x* ∂*M*)$/_R$ (*measure M* (*space M*))
proof (*cases*)
assume *emeasure M* (*space M*) = *0*
then have *space M* ∈ *null-sets M* **by** *auto*
then show *?thesis* **using** *AE-I'* **by** *blast*
next
assume ¬(*emeasure M* (*space M*) = *0*)
then have *measure M* (*space M*) > *0* **using** *emeasure-eq-measure measure-le-0-iff* **by** *fastforce*
then have *y* = ($\int$ *x*. *f x* ∂*M*)$/_R$ (*measure M* (*space M*)) **using** ∗ **by** *auto*
then show *?thesis* **using** *y* **by** *auto*
qed
qed

As the conditional expectation of a function and the original function have the same integral, it follows that the conditional expectation of a function with respect to the invariant sigma algebra is given by the average of the function.

lemma (**in** *ergodic-fmpt*) *Invariants-cond-exp-is-integral-fmpt*:
  fixes *f*::- ⇒ *real*
  assumes *integrable M f*
  shows *AE x in M. real-cond-exp M Invariants f x* = (∫ *x. f x* ∂*M*) / *measure M* (*space M*)
proof −
  have *AE x in M. real-cond-exp M Invariants f x* = (∫ *x. real-cond-exp M Invariants f x* ∂*M*)$/_R$ (*measure M* (*space M*))
    by (*rule Invariant-func-integral*(*2*), *simp add*: *borel-measurable-cond-exp*)
  moreover have (∫ *x. real-cond-exp M Invariants f x* ∂*M*) = (∫ *x. f x* ∂*M*)
    by (*simp add*: *assms real-cond-exp-int*(*2*))
  ultimately show *?thesis* **by** (*simp add*: *divide-real-def mult.commute*)
qed

lemma (**in** *ergodic-pmpt*) *Invariants-cond-exp-is-integral*:
  fixes *f*::- ⇒ *real*
  assumes *integrable M f*
  shows *AE x in M. real-cond-exp M Invariants f x* = (∫ *x. f x* ∂*M*)
by (*metis div-by-1 prob-space Invariants-cond-exp-is-integral-fmpt*[*OF assms*])

7.4 Kac formula

We reformulate the different versions of Kac formula. They simplify as, for any set A with positive measure, the union $\bigcup T^{-n}A$ (which appears in all these statements) almost coincides with the whole space.

lemma (**in** *ergodic-conservative-mpt*) *local-time-unbounded*:
  assumes [*measurable*]: *A* ∈ *sets M B* ∈ *sets M*
    and *emeasure M A* < ∞ *emeasure M B* > *0*
  shows (λ*n. emeasure M* {*x* ∈ (*T*^^*n*)−−‘*A. local-time B n x* < *k*}) ⟶ *0*
proof (*rule local-time-unbounded3*)
  have *A* − (⋃ *i.* (*T* ^^ *i*) −−‘ *B*) ∈ *sets M* **by** *auto*
  moreover have *A* − (⋃ *i.* (*T* ^^ *i*) −−‘ *B*) ⊆ *space M* − (⋃ *i.* (*T* ^^ *i*) −−‘ *B*) **using** *sets.sets-into-space*[*OF assms*(*1*)] **by** *blast*
  ultimately show *A* − (⋃ *i.* (*T* ^^ *i*) −−‘ *B*) ∈ *null-sets M* **by** (*metis null-sets-subset preimages-conull*(*1*)[*OF assms*(*2*) *assms*(*4*)])
  show *emeasure M A* < ∞ **using** *assms*(*3*) **by** *simp*
qed (*simp-all add*: *assms*)

theorem (**in** *ergodic-conservative-mpt*) *kac-formula*:
  assumes [*measurable*]: *A* ∈ *sets M* **and** *emeasure M A* > *0*
  shows (∫⁺*y. return-time-function A y* ∂*M*) = *emeasure M* (*space M*)
proof −
  have *a* [*measurable*]: (⋃ *n.* (*T*^^*n*)−−‘*A*) ∈ *sets M* **by** *auto*
  then have *space M* = (⋃ *n.* (*T*^^*n*)−−‘*A*) ∪ (*space M* − (⋃ *n.* (*T*^^*n*)−−‘*A*))
using *sets.sets-into-space* **by** *blast*

then have *emeasure M* (*space M*) = *emeasure M* ($\bigcup$*n*. (*T*^^*n*)−−'*A*)
by (*metis a preimages-conull*(*1*)[*OF assms*] *emeasure-Un-null-set*)
moreover have ($\int^+$*y*. *return-time-function A y* ∂*M*) = *emeasure M* ($\bigcup$*n*. (*T*^^*n*)−−'*A*)
using *kac-formula-nonergodic*[*OF assms*(*1*)] **by** *simp*
ultimately show *?thesis* **by** *simp*
qed

lemma (**in** *ergodic-conservative-mpt*) *induced-function-integral*:
fixes *f*::'*a* ⇒ *real*
assumes [*measurable*]: *A* ∈ *sets M integrable M f* **and** *emeasure M A* > *0*
shows *integrable M* (*induced-function A f*)
($\int$ *y*. *induced-function A f y* ∂*M*) = ($\int$ *x*. *f x* ∂*M*)
proof −
show *integrable M* (*induced-function A f*)
using *induced-function-integral-nonergodic*(*1*)[*OF assms*(*1*) *assms*(*2*)] **by** *auto*
have ($\int$ *y*. *induced-function A f y* ∂*M*) = ($\int$ *x* ∈ ($\bigcup$*n*. (*T*^^*n*)−−'*A*). *f x* ∂*M*)
using *induced-function-integral-nonergodic*(*2*)[*OF assms*(*1*) *assms*(*2*)] **by** *auto*
also have ... = ($\int$ *x* ∈ *space M*. *f x* ∂*M*)
using *set-integral-null-delta*[*OF assms*(*2*), **where** *?A* = *space M* **and** *?B* = ($\bigcup$*n*. (*T*^^*n*)−−'*A*)]
preimages-conull(*2*)[*OF assms*(*1*) *assms*(*3*)] **by** *auto*
also have ... = ($\int$ *x*. *f x* ∂*M*) **using** *set-integral-space*[*OF assms*(*2*)] **by** *auto*
finally show ($\int$ *y*. *induced-function A f y* ∂*M*) = ($\int$ *x*. *f x* ∂*M*) **by** *simp*
qed

lemma (**in** *ergodic-conservative-mpt*) *induced-function-integral-restr*:
fixes *f*::'*a* ⇒ *real*
assumes [*measurable*]: *A* ∈ *sets M integrable M f* **and** *emeasure M A* > *0*
shows *integrable* (*restrict-space M A*) (*induced-function A f*)
($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int$ *x*. *f x* ∂*M*)
proof −
show *integrable* (*restrict-space M A*) (*induced-function A f*)
using *induced-function-integral-restr-nonergodic*(*1*)[*OF assms*(*1*) *assms*(*2*)] **by** *auto*
have ($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int$ *x* ∈ ($\bigcup$*n*. (*T*^^*n*)−−'*A*). *f x* ∂*M*)
using *induced-function-integral-restr-nonergodic*(*2*)[*OF assms*(*1*) *assms*(*2*)] **by** *auto*
also have ... = ($\int$ *x* ∈ *space M*. *f x* ∂*M*)
using *set-integral-null-delta*[*OF assms*(*2*), **where** *?A* = *space M* **and** *?B* = ($\bigcup$*n*. (*T*^^*n*)−−'*A*)]
preimages-conull(*2*)[*OF assms*(*1*) *assms*(*3*)] **by** *auto*
also have ... = ($\int$ *x*. *f x* ∂*M*) **using** *set-integral-space*[*OF assms*(*2*)] **by** *auto*
finally show ($\int$ *y*. *induced-function A f y* ∂(*restrict-space M A*)) = ($\int$ *x*. *f x* ∂*M*) **by** *simp*
qed

7.5 Birkhoff theorem

The general versions of Birkhoff theorem are formulated in terms of conditional expectations. In ergodic probability measure preserving transformations (the most common setting), they reduce to simpler versions that we state now, as the conditional expectations are simply the averages of the functions.

theorem (**in** *ergodic-pmpt*) *birkhoff-theorem-AE*:
  fixes *f*::*'a* ⇒ *real*
  assumes *integrable M f*
  shows *AE x in M*. (*λn. birkhoff-sum f n x* / *n*) ⟶ (∫ *x*. *f x* ∂*M*)
proof −
  have *AE x in M*. (*λn. birkhoff-sum f n x* / *n*) ⟶ *real-cond-exp M Invariants f x*
    using *birkhoff-theorem-AE-nonergodic*[*OF assms*] **by** *simp*
  moreover have *AE x in M*. *real-cond-exp M Invariants f x* = (∫ *x*. *f x* ∂*M*)
    using *Invariants-cond-exp-is-integral*[*OF assms*] **by** *auto*
  ultimately show *?thesis* **by** *auto*
qed

theorem (**in** *ergodic-pmpt*) *birkhoff-theorem-L1*:
  fixes *f*::*'a* ⇒ *real*
  assumes [*measurable*]: *integrable M f*
  shows (*λn*. ∫⁺*x*. *norm*(*birkhoff-sum f n x* / *n* − (∫ *x*. *f x* ∂*M*)) ∂*M*) ⟶ *0*
proof −
  {
    fix *n*::*nat*
    have *AE x in M*. *real-cond-exp M Invariants f x* = (∫ *x*. *f x* ∂*M*)
      using *Invariants-cond-exp-is-integral*[*OF assms*] **by** *auto*
    then have ∗: *AE x in M*. *norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*)
          = *norm*(*birkhoff-sum f n x* / *n* − (∫ *x*. *f x* ∂*M*))
      by *auto*
    have (∫⁺*x*. *norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂*M*)
          = (∫⁺*x*. *norm*(*birkhoff-sum f n x* / *n* − (∫ *x*. *f x* ∂*M*)) ∂*M*)
      apply (*rule nn-integral-cong-AE*) **using** ∗ **by** *auto*
  }
  moreover have (*λn*. ∫⁺*x*. *norm*(*birkhoff-sum f n x* / *n* − *real-cond-exp M Invariants f x*) ∂*M*) ⟶ *0*
    using *birkhoff-theorem-L1-nonergodic*[*OF assms*] **by** *auto*
  ultimately show *?thesis* **by** *simp*
qed

theorem (**in** *ergodic-pmpt*) *birkhoff-sum-small-asymp-pos*:
  fixes *f*::*'a* ⇒ *real*
  assumes [*measurable*]: *integrable M f* **and** *e>0*
  shows *AE x in M*. *infinite* {*n*. *birkhoff-sum f n x* ∈ {*n* ∗ (∫ *x*. *f x* ∂*M*) .. *n* ∗

$(\int x.\ f\ x\ \partial M) + e\}\}$
proof −
have *AE x in M. infinite* $\{n.$ *birkhoff-sum f n x* $\in \{n *$ *real-cond-exp M Invariants f x* .. $n *$ *real-cond-exp M Invariants f x* $+ e\}\}$
using *birkhoff-sum-small-asymp-pos-nonergodic*[*OF assms*] **by** *simp*
moreover have *AE x in M. real-cond-exp M Invariants f x* $= (\int x.\ f\ x\ \partial M)$
using *Invariants-cond-exp-is-integral*[*OF assms*(*1*)] **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

theorem (**in** *ergodic-pmpt*) *birkhoff-sum-small-asymp-neg*:
fixes $f::'a \Rightarrow real$
assumes [*measurable*]: *integrable M f* **and** $e>0$
shows *AE x in M. infinite* $\{n.$ *birkhoff-sum f n x* $\in \{n * (\int x.\ f\ x\ \partial M) - e\ ..\ n * (\int x.\ f\ x\ \partial M)\}\}$
proof −
have *AE x in M. infinite* $\{n.$ *birkhoff-sum f n x* $\in \{n *$ *real-cond-exp M Invariants f x* $- e$ .. $n *$ *real-cond-exp M Invariants f x*$\}\}$
using *birkhoff-sum-small-asymp-neg-nonergodic*[*OF assms*] **by** *simp*
moreover have *AE x in M. real-cond-exp M Invariants f x* $= (\int x.\ f\ x\ \partial M)$
using *Invariants-cond-exp-is-integral*[*OF assms*(*1*)] **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemma (**in** *ergodic-pmpt*) *birkhoff-positive-average*:
fixes $f::'a \Rightarrow real$
assumes [*measurable*]: *integrable M f* **and** *AE x in M.* $(\lambda n.$ *birkhoff-sum f n x*$) \longrightarrow \infty$
shows $(\int x.\ f\ x\ \partial M) > 0$
proof (*rule ccontr*)
assume $\neg((\int x.\ f\ x\ \partial M) > 0)$
then have $*$: $(\int x.\ f\ x\ \partial M) \leq 0$ **by** *simp*

have *AE x in M.* $(\lambda n.$ *birkhoff-sum f n x*$) \longrightarrow \infty \wedge$ *infinite* $\{n.$ *birkhoff-sum f n x* $\in \{n * (\int x.\ f\ x\ \partial M) - 1\ ..\ n* (\int x.\ f\ x\ \partial M)\}\}$
using *assms*(*2*) *birkhoff-sum-small-asymp-neg*[*OF assms*(*1*)] **by** *auto*
then obtain *x* **where** *x*: $(\lambda n.$ *birkhoff-sum f n x*$) \longrightarrow \infty$ *infinite* $\{n.$ *birkhoff-sum f n x* $\in \{n * (\int x.\ f\ x\ \partial M) - 1\ ..\ n* (\int x.\ f\ x\ \partial M)\}\}$
using *AE-False eventually-elim2* **by** *blast*
{
fix *n* **assume** *birkhoff-sum f n x* $\in \{n * (\int x.\ f\ x\ \partial M) - 1\ ..\ n * (\int x.\ f\ x\ \partial M)\}$
then have *birkhoff-sum f n x* $\leq n * (\int x.\ f\ x\ \partial M)$ **by** *simp*
also have ... ≤ 0 **using** $*$ **by** (*simp add*: *mult-nonneg-nonpos*)
finally have *birkhoff-sum f n x* ≤ 0 **by** *simp*
}
then have $\{n.$ *birkhoff-sum f n x* $\in \{n * (\int x.\ f\ x\ \partial M) - 1\ ..\ n* (\int x.\ f\ x\ \partial M)\}\} \subseteq \{n.$ *birkhoff-sum f n x* $\leq 0\}$ **by** *auto*
then have *inf*: *infinite* $\{n.$ *birkhoff-sum f n x* $\leq 0\}$ **using** *x*(*2*) *finite-subset* **by** *blast*

have $0 < (\infty::ereal)$ **by** *auto*
then have *eventually* $(\lambda n.\ birkhoff\text{-}sum\ f\ n\ x > (0::ereal))$ *sequentially* **using** $x(1)$ *order-tendsto-iff* **by** *metis*
then obtain N **where** $\bigwedge n.\ n \geq N \Longrightarrow birkhoff\text{-}sum\ f\ n\ x > (0::ereal)$ **by** (*meson eventually-at-top-linorder*)
then have $\bigwedge n.\ n \geq N \Longrightarrow birkhoff\text{-}sum\ f\ n\ x > 0$ **by** *auto*
then have $\{n.\ birkhoff\text{-}sum\ f\ n\ x \leq 0\} \subseteq \{..<N\}$ **by** (*metis* (*mono-tags*, *lifting*) *lessThan-iff linorder-not-less mem-Collect-eq subsetI*)
then have *finite* $\{n.\ birkhoff\text{-}sum\ f\ n\ x \leq 0\}$ **using** *finite-nat-iff-bounded* **by** *blast*

then show *False* **using** *inf* **by** *simp*
qed

lemma (**in** *ergodic-pmpt*) *birkhoff-negative-average*:
fixes $f::'a \Rightarrow real$
assumes [*measurable*]: *integrable M f* **and** $AE\ x\ in\ M.\ (\lambda n.\ birkhoff\text{-}sum\ f\ n\ x) \longrightarrow -\infty$
shows $(\int x.\ f\ x\ \partial M) < 0$
proof (*rule ccontr*)
assume $\neg((\int x.\ f\ x\ \partial M) < 0)$
then have $*$: $(\int x.\ f\ x\ \partial M) \geq 0$ **by** *simp*

have $AE\ x\ in\ M.\ (\lambda n.\ birkhoff\text{-}sum\ f\ n\ x) \longrightarrow -\infty \wedge infinite\ \{n.\ birkhoff\text{-}sum\ f\ n\ x \in \{n * (\int x.\ f\ x\ \partial M)\ ..\ n* (\int x.\ f\ x\ \partial M) + 1\}\}$
using *assms*(2) *birkhoff-sum-small-asymp-pos*[*OF assms*(1)] **by** *auto*
then obtain x **where** x: $(\lambda n.\ birkhoff\text{-}sum\ f\ n\ x) \longrightarrow -\infty$ *infinite* $\{n.\ birkhoff\text{-}sum\ f\ n\ x \in \{n * (\int x.\ f\ x\ \partial M)\ ..\ n* (\int x.\ f\ x\ \partial M) + 1\}\}$
using *AE-False eventually-elim2* **by** *blast*
{
fix n **assume** $birkhoff\text{-}sum\ f\ n\ x \in \{n * (\int x.\ f\ x\ \partial M)\ ..\ n * (\int x.\ f\ x\ \partial M) + 1\}$
then have $birkhoff\text{-}sum\ f\ n\ x \geq n * (\int x.\ f\ x\ \partial M)$ **by** *simp*
moreover have $n * (\int x.\ f\ x\ \partial M) \geq 0$ **using** $*$ **by** *simp*
ultimately have $birkhoff\text{-}sum\ f\ n\ x \geq 0$ **by** *simp*
}
then have $\{n.\ birkhoff\text{-}sum\ f\ n\ x \in \{n * (\int x.\ f\ x\ \partial M)\ ..\ n* (\int x.\ f\ x\ \partial M) + 1\}\} \subseteq \{n.\ birkhoff\text{-}sum\ f\ n\ x \geq 0\}$ **by** *auto*
then have *inf*: *infinite* $\{n.\ birkhoff\text{-}sum\ f\ n\ x \geq 0\}$ **using** $x(2)$ *finite-subset* **by** *blast*

have $0 > (-\infty::ereal)$ **by** *auto*
then have *eventually* $(\lambda n.\ birkhoff\text{-}sum\ f\ n\ x < (0::ereal))$ *sequentially* **using** $x(1)$ *order-tendsto-iff* **by** *metis*
then obtain N **where** $\bigwedge n.\ n \geq N \Longrightarrow birkhoff\text{-}sum\ f\ n\ x < (0::ereal)$ **by** (*meson eventually-at-top-linorder*)
then have $\bigwedge n.\ n \geq N \Longrightarrow birkhoff\text{-}sum\ f\ n\ x < 0$ **by** *auto*
then have $\{n.\ birkhoff\text{-}sum\ f\ n\ x \geq 0\} \subseteq \{..<N\}$ **by** (*metis* (*mono-tags*, *lifting*)

lessThan-iff linorder-not-less mem-Collect-eq subsetI)
  then have *finite* {*n. birkhoff-sum f n x* ≥ *0*} **using** *finite-nat-iff-bounded* **by** *blast*

  then show *False* **using** *inf* **by** *simp*
qed

lemma (**in** *ergodic-pmpt*) *birkhoff-nonzero-average*:
  fixes *f*::*'a* ⇒ *real*
  assumes [*measurable*]: *integrable M f* **and** *AE x in M*. (λ*n. abs*(*birkhoff-sum f n x*)) ⟶ ∞
  shows (∫ *x. f x* ∂*M*) ≠ *0*
proof (*rule ccontr*)
  assume ¬((∫ *x. f x* ∂*M*) ≠ *0*)
  then have ∗: (∫ *x. f x* ∂*M*) = *0* **by** *simp*

  have *AE x in M*. (λ*n. abs*(*birkhoff-sum f n x*)) ⟶ ∞ ∧ *infinite* {*n. birkhoff-sum f n x* ∈ {*0* .. *1*}}
    using *assms*(*2*) *birkhoff-sum-small-asymp-pos*[*OF assms*(*1*)] ∗ **by** *auto*
  then obtain *x* **where** *x*: (λ*n. abs*(*birkhoff-sum f n x*)) ⟶ ∞ *infinite* {*n. birkhoff-sum f n x* ∈ {*0* .. *1*}}
    using *AE-False eventually-elim2* **by** *blast*

  have *1* < (∞::*ereal*) **by** *auto*
  then have *eventually* (λ*n. abs*(*birkhoff-sum f n x*) > (*1*::*ereal*)) *sequentially* **using** *x*(*1*) *order-tendsto-iff* **by** *metis*
  then obtain *N* **where** ⋀*n. n* ≥ *N* ⟹ *abs*(*birkhoff-sum f n x*) > (*1*::*ereal*) **by** (*meson eventually-at-top-linorder*)
  then have ∗: ⋀*n. n* ≥ *N* ⟹ *abs*(*birkhoff-sum f n x*) > *1* **by** *auto*
  have {*n. birkhoff-sum f n x* ∈ {*0*..*1*}} ⊆ {..<*N*} **by** (*auto*, *metis* (*full-types*) ∗ *abs-of-nonneg not-less*)
  then have *finite* {*n. birkhoff-sum f n x* ∈ {*0*..*1*}} **using** *finite-nat-iff-bounded* **by** *blast*

  then show *False* **using** *x*(*2*) **by** *simp*
qed

end

8 The shift operator on an infinite product measure

theory *Shift-Operator*
  imports *Ergodicity ME-Library-Complement*
begin

Let *P* be an an infinite product of i.i.d. instances of the distribution *M*.

Then the shift operator is the map

$$T(x_0, x_1, x_2, \ldots) = T(x_1, x_2, \ldots) \ .$$

In this section, we define this operator and show that it is ergodic using Kolmogorov's 0–1 law.

locale *shift-operator-ergodic* = *prob-space* +
  fixes T :: (*nat* ⇒ *'a*) ⇒ (*nat* ⇒ *'a*) **and** P :: (*nat* ⇒ *'a*) *measure*
  defines $T \equiv (\lambda f.\ f \circ Suc)$
  defines $P \equiv$ *PiM* (*UNIV* :: *nat set*) (λ-. *M*)
begin

sublocale *P*: *product-prob-space* λ-. *M UNIV*
  by *unfold-locales*

sublocale *P*: *prob-space P*
  by (*simp add*: *prob-space-PiM prob-space-axioms P-def*)

lemma *measurable-T* [*measurable*]: $T \in P \rightarrow_M P$
  unfolding *P-def T-def o-def*
  by (*rule measurable-abs-UNIV*[*OF measurable-compose*[*OF measurable-component-singleton*]]) *auto*

The n-th tail algebra $\mathcal{T}_n$ is, in some sense, the algebra in which we forget all information about all x_i with $i < n$. We simply change the product algebra of P by replacing the algebra for each $i < n$ with the trivial algebra that contains only the empty set and the entire space.

definition *tail-algebra* :: *nat* ⇒ (*nat* ⇒ *'a*) *measure*
  where *tail-algebra n* = *PiM UNIV* (λ*i*. *if i* < *n then trivial-measure* (*space M*) *else M*)

lemma *tail-algebra-0* [*simp*]: *tail-algebra 0* = *P*
  by (*simp add*: *tail-algebra-def P-def*)

lemma *space-tail-algebra* [*simp*]: *space* (*tail-algebra n*) = *PiE UNIV* (λ-. *space M*)
  by (*simp add*: *tail-algebra-def space-PiM PiE-def Pi-def*)

lemma *measurable-P-component* [*measurable*]: *P.random-variable M* (λ*f*. *f i*)
  unfolding *P-def* **by** *measurable*

lemma *P-component* [*simp*]: *distr P M* (λ*f*. *f i*) = *M*
  unfolding *P-def* **by** (*subst P.PiM-component*) *auto*

lemma *indep-vars*: *P.indep-vars* (λ-. *M*) (λ*i f*. *f i*) *UNIV*
  by (*subst P.indep-vars-iff-distr-eq-PiM*)
    (*simp-all add*: *restrict-def distr-id2 P.PiM-component P-def*)

The shift operator takes us from $\mathcal{T}_n$ to $\mathcal{T}_{n+1}$ (it forgets the information about one more variable):

lemma *measurable-T-tail*: $T \in$ *tail-algebra* (*Suc n*) $\to_M$ *tail-algebra n*
unfolding *T-def tail-algebra-def o-def*
by (*rule measurable-abs-UNIV*[*OF measurable-compose*[*OF measurable-component-singleton*]]) *simp-all*

lemma *measurable-funpow-T*: T ^^ $n \in$ *tail-algebra* $(m + n) \to_M$ *tail-algebra m*
proof (*induction n*)
case (*Suc n*)
have (T ^^ n) $\circ$ $T \in$ *tail-algebra* (m + *Suc n*) $\to_M$ *tail-algebra m*
by (*rule measurable-comp*[*OF - Suc*]) (*simp-all add*: *measurable-T-tail*)
thus *?case* **by** (*simp add*: *o-def funpow-swap1*)
qed *auto*

lemma *measurable-funpow-T'*: T ^^ $n \in$ *tail-algebra n* $\to_M$ P
using *measurable-funpow-T*[*of n 0*] **by** *simp*

The shift operator is clearly measure-preserving:

lemma *measure-preserving*: $T \in$ *measure-preserving P P*
proof
fix A :: (*nat* $\Rightarrow$ *'a*) *set* **assume** $A \in P$.*events*
hence *emeasure P* (T −‘ $A \cap$ *space P*) = *emeasure* (*distr P P T*) *A*
by (*subst emeasure-distr*) *simp-all*
also have *distr P P T* = *P* **unfolding** *P-def T-def o-def*
using *distr-PiM-reindex*[*of UNIV* λ-. *M Suc UNIV*] **by** (*simp add*: *prob-space-axioms restrict-def*)
finally show *emeasure P* (T −‘ $A \cap$ *space P*) = *emeasure P A* **.**
qed *auto*

sublocale *fmpt P T*
by *unfold-locales*
(*use measure-preserving* **in** ‹*blast intro*: *measure-preserving-is-quasi-measure-preserving*›)+

lemma *indep-sets-vimage-algebra*:
P.indep-sets (λ*i*. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*)) *UNIV*
using *indep-vars* **unfolding** *P.indep-vars-def sets-vimage-algebra* **by** *blast*

We can now show that the tail algebra $\mathcal{T}_n$ is a subalgebra of the algebra generated by the algebras induced by all the variables x_i with $i \geq n$:

lemma *tail-algebra-subset*:
sets (*tail-algebra n*) $\subseteq$
sigma-sets (*space P*) ($\bigcup$*i*∈{*n*..}. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*))
proof −
have *sets* (*tail-algebra n*) = *sigma-sets* (*space P*)
(*prod-algebra UNIV* (λ*i*. *if i* < *n then trivial-measure* (*space M*) *else M*))
by (*simp add*: *tail-algebra-def sets-PiM PiE-def Pi-def P-def space-PiM*)

also have ... $\subseteq$ *sigma-sets* (*space P*) ($\bigcup$*i*∈{*n*..}. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*))

proof (*intro sigma-sets-mono subsetI*)
fix C **assume** $C \in$ *prod-algebra UNIV* ($\lambda i.$ *if* $i < n$ *then trivial-measure* (*space* M) *else* M)
then obtain C'
where C': $C = Pi_E$ *UNIV* C'
$C' \in (\Pi\ i \in UNIV.$ *sets* (*if* $i < n$ *then trivial-measure* (*space* M) *else* M))
by (*elim prod-algebraE-all*)
have C'-*1*: $C'\ i \in \{\{\},$ *space* $M\}$ **if** $i < n$ **for** i
using $C'(2)$ *that* **by** (*auto simp*: *Pi-def sets-trivial-measure split*: *if-splits*)
have C'-*2*: $C'\ i \in$ *sets* M **if** $i \geq n$ **for** i
proof −
from *that* **have** $\neg(i < n)$
by *auto*
with $C'(2)$ **show** *?thesis*
by (*force simp*: *Pi-def sets-trivial-measure split*: *if-splits*)
qed
have $C'\ i \in$ *events* **for** i
using C'-*1*[*of i*] C'-*2*[*of i*] **by** (*cases* $i \geq n$) *auto*
hence $C \in$ *sets* P
unfolding *P-def* $C'(1)$ **by** (*intro sets-PiM-I-countable*) *auto*
hence $C \subseteq$ *space* P
using *sets.sets-into-space* **by** *blast*

show $C \in$ *sigma-sets* (*space* P) ($\bigcup i \in \{n..\}$. *sets* (*vimage-algebra* (*space* P) ($\lambda f.\ f\ i$) M))
proof (*cases* $C = \{\}$)
case *False*
have $C = (\bigcap i \in \{n..\}. (\lambda f.\ f\ i)$ −' $C'\ i) \cap$ *space* P
proof (*intro equalityI subsetI*, *goal-cases*)
case (*1 f*)
hence $f \in$ *space* P
using *1* ‹$C \subseteq$ *space* P› **by** *blast*
thus *?case*
using C' *1* **by** (*auto simp*: *Pi-def sets-trivial-measure split*: *if-splits*)
next
case (*2 f*)
hence f: $f\ i \in C'\ i$ **if** $i \geq n$ **for** i
using *that* **by** *auto*
have $f\ i \in C'\ i$ **for** i
proof (*cases* $i \geq n$)
case *True*
thus *?thesis* **using** C'-*2*[*of i*] *f*[*of i*] **by** *auto*
next
case *False*
thus *?thesis* **using** C'-*1*[*of i*] $C'(1)$ ‹$C \neq \{\}$› *2*
by (*auto simp*: *P-def space-PiM*)
qed
thus $f \in C$

using C' **by** *auto*
qed

also have $(\bigcap i{\in}\{n..\}.\ (\lambda f.\ f\ i) -` C'\ i) \cap space\ P =$
$(\bigcap i{\in}\{n..\}.\ (\lambda f.\ f\ i) -` C'\ i \cap space\ P)$
by *blast*

also have $\ldots \in sigma\text{-}sets\ (space\ P)\ (\bigcup i{\in}\{n..\}.\ sets\ (vimage\text{-}algebra\ (space\ P)\ (\lambda f.\ f\ i)\ M))$
(**is** - $\in$ *?rhs*)
proof (*intro sigma-sets-INTER*, *goal-cases*)
fix *i* **show** $(\lambda f.\ f\ i) -` C'\ i \cap space\ P \in$ *?rhs*
proof ($cases\ i \geq n$)
case *False*
hence $C'\ i = \{\} \vee C'\ i = space\ M$
using C'-*1*[*of i*] **by** *auto*
thus *?thesis*
proof
assume [*simp*]: $C'\ i = space\ M$
have $space\ P \subseteq (\lambda f.\ f\ i) -` C'\ i$
by (*auto simp*: *P-def space-PiM*)
hence $(\lambda f.\ f\ i) -` C'\ i \cap space\ P = space\ P$
by *blast*
thus *?thesis* **using** *sigma-sets-top*
by *metis*
qed (*auto intro*: *sigma-sets.Empty*)
next
case *i*: *True*
have $(\lambda f.\ f\ i) -` C'\ i \cap space\ P \in sets\ (vimage\text{-}algebra\ (space\ P)\ (\lambda f.\ f\ i)\ M)$
using C'-*2*[*OF i*] **by** (*blast intro*: *in-vimage-algebra*)
thus *?thesis* **using** *i* **by** *blast*
qed
next
have $C \subseteq space\ P$ **if** $C \in sets\ (vimage\text{-}algebra\ (space\ P)\ (\lambda f.\ f\ i)\ M)$ **for** *i* *C*
using *sets.sets-into-space*[*OF that*] **by** *simp*
thus $(\bigcup i{\in}\{n..\}.\ sets\ (vimage\text{-}algebra\ (space\ P)\ (\lambda f.\ f\ i)\ M)) \subseteq Pow\ (space\ P)$
by *auto*
qed *auto*

finally show *?thesis* **.**
qed (*auto simp*: *sigma-sets.Empty*)
qed

finally show *?thesis* **.**
qed

It now follows that the T-invariant events are a subset of the tail algebra induced by the variables:

lemma *Invariants-subset-tail-algebra*:
  sets Invariants ⊆ *P.tail-events* (λ*i*. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*))
proof
  fix *A* **assume** *A*: *A* ∈ *sets Invariants*
  have *A'*: *A* ∈ *P.events*
    using *A* **unfolding** *Invariants-sets* **by** *simp-all*
  show *A* ∈ *P.tail-events* (λ*i*. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*))
      unfolding *P.tail-events-def*
  proof *safe*
    fix *n* :: *nat*
    have *vimage-restr T A* = *A*
      using *A* **by** (*simp add*: *Invariants-vrestr*)
    hence *A* = *vimage-restr* (*T* ^^ *n*) *A*
      using *A'* **by** (*induction n*) (*simp-all add*: *vrestr-comp*)
    also have *vimage-restr* (*T* ^^ *n*) *A* = (*T* ^^ *n*) −' (*A* ∩ *space P*) ∩ *space P*
      unfolding *vimage-restr-def* **..**
    also have *A* ∩ *space P* = *A*
      using *A'* **by** *simp*
    also have *space P* = *space* (*tail-algebra n*)
      by (*simp add*: *P-def space-PiM*)
    also have (*T* ^^ *n*) −' *A* ∩ *space* (*tail-algebra n*) ∈ *sets* (*tail-algebra n*)
      by (*rule measurable-sets*[*OF measurable-funpow-T' A'*])
    also have *sets* (*tail-algebra n*) ⊆
          sigma-sets (*space P*) (⋃*i*∈{*n*..}. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f*
i) *M*))
      by (*rule tail-algebra-subset*)
    finally show *A* ∈ *sigma-sets* (*space P*)
          (⋃*i*∈{*n*..}. *sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*)) **.**
  qed
qed

A simple invocation of Kolmogorov's 0–1 law now proves that *T* is indeed ergodic:

sublocale *ergodic-fmpt P T*
proof
  fix *A* **assume** *A*: *A* ∈ *sets Invariants*
  have *A'*: *A* ∈ *P.events*
    using *A* **unfolding** *Invariants-sets* **by** *simp-all*
  have *sigma-algebra* (*space P*) (*sets* (*vimage-algebra* (*space P*) (λ*f*. *f i*) *M*)) **for** *i*
    by (*metis sets.sigma-algebra-axioms space-vimage-algebra*)
  hence *P.prob A* = *0* ∨ *P.prob A* = *1*
    using *indep-sets-vimage-algebra*
    by (*rule P.kolmogorov-0-1-law*) (*use A Invariants-subset-tail-algebra* **in** *blast*)
  thus *A* ∈ *null-sets P* ∨ *space P* − *A* ∈ *null-sets P*
   by (*rule disj-forward*) (*use A'*(*1*) *P.prob-compl*[*of A*] **in** ‹*auto simp*: *P.emeasure-eq-measure*›)
qed

end

end

9 Subcocycles, subadditive ergodic theory

theory *Kingman*
  imports *Ergodicity Fekete*
begin

Subadditive ergodic theory is the branch of ergodic theory devoted to the study of subadditive cocycles (named subcocycles in what follows), i.e., functions such that $u(n+m,x) \leq u(n,x) + u(m,T^n x)$ for all x and m,n.

For instance, Birkhoff sums are examples of such subadditive cocycles (in fact, they are additive), but more interesting examples are genuinely subadditive. The main result of the theory is Kingman's theorem, asserting the almost sure convergence of u_n/n (this is a generalization of Birkhoff theorem). If the asymptotic average $\lim \int u_n/n$ (which exists by subadditivity and Fekete lemma) is not $-\infty$, then the convergence takes also place in L^1. We prove all this below.

context *mpt*
begin

9.1 Definition and basic properties

definition *subcocycle*::(*nat* ⇒ *'a* ⇒ *real*) ⇒ *bool*
  where *subcocycle u* = ((∀ *n*. *integrable M* (*u n*)) ∧ (∀ *n m x*. *u* (*n*+*m*) *x* ≤ *u n x* + *u m* ((*T*^^*n*) *x*)))

lemma *subcocycle-ineq*:
  assumes *subcocycle u*
  shows *u* (*n*+*m*) *x* ≤ *u n x* + *u m* ((*T*^^*n*) *x*)
using *assms* **unfolding** *subcocycle-def* **by** *blast*

lemma *subcocycle-0-nonneg*:
  assumes *subcocycle u*
  shows *u 0 x* ≥ *0*
proof −
  have *u* (*0*+*0*) *x* ≤ *u 0 x* + *u 0* ((*T*^^*0*) *x*)
    using *assms* **unfolding** *subcocycle-def* **by** *blast*
  then show *?thesis* **by** *auto*
qed

lemma *subcocycle-integrable*:
  assumes *subcocycle u*
  shows *integrable M* (*u n*)
      u n ∈ *borel-measurable M*
using *assms* **unfolding** *subcocycle-def* **by** *auto*

lemma *subcocycle-birkhoff*:
assumes *integrable M f*
shows *subcocycle* (*birkhoff-sum f*)
unfolding *subcocycle-def* **by** (*auto simp add*: *assms birkhoff-sum-integral*(*1*) *birkhoff-sum-cocycle*)

The set of subcocycles is stable under addition, multiplication by positive numbers, and max.

lemma *subcocycle-add*:
assumes *subcocycle u subcocycle v*
shows *subcocycle* ($\lambda n\ x.\ u\ n\ x + v\ n\ x$)
unfolding *subcocycle-def*
proof (*auto*)
fix *n*
show *integrable M* ($\lambda x.\ u\ n\ x + v\ n\ x$) **using** *assms* **unfolding** *subcocycle-def* **by** *simp*
next
fix *n m x*
have $u\ (n+m)\ x \leq u\ n\ x + u\ m\ ((T\ \widehat{}\widehat{}\ n)\ x)$ **using** *assms*(*1*) *subcocycle-def* **by** *simp*
moreover have $v\ (n+m)\ x \leq v\ n\ x + v\ m\ ((T\ \widehat{}\widehat{}\ n)\ x)$ **using** *assms*(*2*) *subcocycle-def* **by** *simp*
ultimately show $u\ (n + m)\ x + v\ (n + m)\ x \leq u\ n\ x + v\ n\ x + (u\ m\ ((T\ \widehat{}\widehat{}\ n)\ x) + v\ m\ ((T\ \widehat{}\widehat{}\ n)\ x))$
by *simp*
qed

lemma *subcocycle-cmult*:
assumes *subcocycle u* $c \geq 0$
shows *subcocycle* ($\lambda n\ x.\ c * u\ n\ x$)
using *assms* **unfolding** *subcocycle-def* **by** (*auto*, *metis distrib-left mult-left-mono*)

lemma *subcocycle-max*:
assumes *subcocycle u subcocycle v*
shows *subcocycle* ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$)
unfolding *subcocycle-def* **proof** (*auto*)
fix *n*
show *integrable M* ($\lambda x.\ max\ (u\ n\ x)\ (v\ n\ x)$) **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
next
fix *n m x*
have $u\ (n+m)\ x \leq u\ n\ x + u\ m\ ((T\widehat{}\widehat{}n)\ x)$ **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
then show $u\ (n + m)\ x \leq max\ (u\ n\ x)\ (v\ n\ x) + max\ (u\ m\ ((T\ \widehat{}\widehat{}\ n)\ x))\ (v\ m\ ((T\ \widehat{}\widehat{}\ n)\ x))$
by *simp*
next
fix *n m x*
have $v\ (n+m)\ x \leq v\ n\ x + v\ m\ ((T\widehat{}\widehat{}n)\ x)$ **using** *assms*(*2*) **unfolding** *subco-*

cycle-def **by** *auto*
then show $v\ (n + m)\ x \leq max\ (u\ n\ x)\ (v\ n\ x) + max\ (u\ m\ ((T \widehat{\ }\widehat{\ }\ n)\ x))\ (v\ m\ ((T \widehat{\ }\widehat{\ }\ n)\ x))$
by *simp*
qed

Applying inductively the subcocycle equation, it follows that a subcocycle is bounded by the Birkhoff sum of the subcocycle at time 1.

lemma *subcocycle-bounded-by-birkhoff1*:
assumes *subcocycle u* $n > 0$
shows $u\ n\ x \leq$ *birkhoff-sum* $(u\ 1)\ n\ x$
using ‹$n > 0$› **proof** (*induction rule*: *ind-from-1*)
case *1*
show *?case* **by** *auto*
next
case (*Suc p*)
have $u\ (Suc\ p)\ x \leq u\ p\ x + u\ 1\ ((T\widehat{\ }\widehat{\ }p)x)$ **using** *assms*(*1*) *subcocycle-def* **by** (*metis Suc-eq-plus1*)
then show *?case* **using** *Suc birkhoff-sum-cocycle*[**where** $?n = p$ **and** $?m = 1$] ‹ $p>0$ › **by** (*simp add*: *birkhoff-sum-def*)
qed

It is often important to bound a cocycle $u_n(x)$ by the Birkhoff sums of u_N/N. Compared to the trivial upper bound for u_1, there are additional boundary errors that make the estimate more cumbersome (but these terms only come from a N-neighborhood of 0 and n, so they are negligible if N is fixed and n tends to infinity.

lemma *subcocycle-bounded-by-birkhoffN*:
assumes *subcocycle u* $n > 2*N$ $N>0$
shows $u\ n\ x \leq$ *birkhoff-sum* $(\lambda x.\ u\ N\ x\ /\ real\ N)\ (n - 2 * N)\ x$
$+ (\sum i<N.\ |u\ 1\ ((T \widehat{\ }\widehat{\ }\ i)\ x)|)$
$+ 2 * (\sum i<2*N.\ |u\ 1\ ((T \widehat{\ }\widehat{\ }\ (n - (2 * N - i)))\ x)|)$
proof −
have *Iar*: $u\ (a*N+r)\ x \leq u\ r\ x + (\sum i<a.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N + r))x))$ **for** *r a*
proof (*induction a*)
case *0*
then show *?case* **by** *auto*
next
case (*Suc a*)
have $u\ ((a+1)*N+r)\ x = u((a*N+r) + N)\ x$
by (*simp add*: *semiring-normalization-rules*(*2*) *semiring-normalization-rules*(*23*))
also have ... $\leq u(a*N+r)\ x + u\ N\ ((T\widehat{\ }\widehat{\ }(a*N+r))x)$
using *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
also have ... $\leq u\ r\ x + (\sum i<a.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N + r))x)) + u\ N\ ((T\widehat{\ }\widehat{\ }(a*N+r))x)$
using *Suc.IH* **by** *auto*
also have ... $= u\ r\ x + (\sum i<a+1.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N + r))x))$
by *auto*
finally show *?case* **by** *auto*

qed

have *Ia*: $u\ (a*N)\ x \leq (\sum i<a.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N))x))$ **if** $a>0$ **for** a
using *that* **proof** (*induction rule*: *ind-from-1*)
case *1*
show *?case* **by** *auto*
next
case (*Suc a*)
have $u\ ((a+1)*N)\ x = u((a*N) + N)\ x$
by (*simp add*: *semiring-normalization-rules*(*2*) *semiring-normalization-rules*(*23*))
also have ... $\leq u(a*N)\ x + u\ N\ ((T\widehat{\ }\widehat{\ }(a*N))x)$
using *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
also have ... $\leq (\sum i<a.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N))x)) + u\ N\ ((T\widehat{\ }\widehat{\ }(a*N))x)$
using *Suc* **by** *auto*
also have ... $= (\sum i<a+1.\ u\ N\ ((T\widehat{\ }\widehat{\ }(i * N))x))$
by *auto*
finally show *?case* **by** *auto*
qed

define *E1* **where** $E1 = (\sum i<N.\ abs(u\ 1\ ((T\widehat{\ }\widehat{\ }i)x)))$
define *E2* **where** $E2 = (\sum i<2*N.\ abs(u\ 1\ ((T\widehat{\ }\widehat{\ }(n-(2*N-i)))\ x)))$
have $E2 \geq 0$ **unfolding** *E2-def* **by** *auto*

obtain *a0 s0* **where** *0*: $s0 < N\ n = a0 * N + s0$
using ‹$0 < N$› *mod-div-decomp mod-less-divisor* **by** *blast*
then have $a0 \geq 1$ **using** ‹$n > 2 * N$› ‹$N>0$›
by (*metis Nat.add-0-right add.commute add-lessD1 add-mult-distrib comm-monoid-mult-class.mult-1 eq-imp-le*
less-imp-add-positive less-imp-le-nat less-one linorder-neqE-nat mult.left-neutral mult-not-zero not-add-less1 one-add-one)
define *a s* **where** $a = a0-1$ **and** $s = s0+N$
then have *as*: $n = a * N + s$ **unfolding** *a-def s-def* **using** ‹$a0 \geq 1$› *0* **by** (*simp add*: *mult-eq-if*)
have *s*: $s \geq N\ s < 2*N$ **using** *0* **unfolding** *s-def* **by** *auto*
have *a*: $a*N > n - 2*N\ a*N \leq n - N$ **using** *as s* ‹$n > 2*N$› **by** *auto*
then have $(a*N - (n-2*N)) \leq N$ **using** ‹$n > 2*N$› **by** *auto*
have $a*N \geq n - 2*N$ **using** *a* **by** *simp*

{
fix *r*::*nat* **assume** $r < N$
have $a*N+r > n - 2*N$ **using** ‹$n>2*N$› *as s* **by** *auto*

define *tr* **where** $tr = n-(a*N+r)$
have $tr > 0$ **unfolding** *tr-def* **using** *as s* ‹$r<N$› **by** *auto*
then have $*$: $n = (a*N+r) + tr$ **unfolding** *tr-def* **by** *auto*

have *birkhoff-sum* (*u 1*) *tr* $((T\widehat{\ }\widehat{\ }(a*N+r))x) = (\sum i<tr.\ u\ 1\ ((T\widehat{\ }\widehat{\ }(a*N+r+i))x))$
unfolding *birkhoff-sum-def* **by** (*simp add*: *add.commute funpow-add*)
also have ... $= (\sum i\in\{a*N+r..<a*N+r+tr\}.\ u\ 1\ ((T\widehat{\ }\widehat{\ }i)\ x))$

```
      by (rule sum.reindex-bij-betw, rule bij-betw-byWitness[where ?f' = λi. i −
(a ∗ N + r)], auto)
    also have ... ≤ (∑ i∈{a∗N+r..<a∗N+r+tr}. abs(u 1 ((T^^i) x)))
      by (simp add: sum-mono)
    also have ... ≤ (∑ i∈{n−2∗N..<n}. abs(u 1 ((T^^i) x)))
      apply (rule sum-mono2) using as s ‹r<N› tr-def by auto
    also have ... = E2 unfolding E2-def
      apply (rule sum.reindex-bij-betw[symmetric], rule bij-betw-byWitness[where
?f' = λi. i − (n−2∗N)])
      using ‹n > 2∗N› by auto
    finally have A: birkhoff-sum (u 1) tr ((T^^(a∗N+r))x) ≤ E2 by simp

    have u n x ≤ u (a∗N+r) x + u tr ((T^^(a∗N+r))x)
      using assms(1) ∗ unfolding subcocycle-def by auto
    also have ... ≤ u (a∗N+r) x + birkhoff-sum (u 1) tr ((T^^(a∗N+r))x)
      using subcocycle-bounded-by-birkhoff1[OF assms(1)] ‹tr > 0› by auto
    finally have B: u n x ≤ u (a∗N+r) x + E2
      using A by auto

   have u (a∗N+r) x ≤ (∑ i<N. abs(u 1 ((T^^i)x))) + (∑ i<a. u N ((T^^(i∗N+r))x))
    proof (cases r = 0)
      case True
      then have a>0 using ‹a∗N+r > n − 2∗N› not-less by fastforce
       have u(a∗N+r) x ≤ (∑ i<a. u N ((T^^(i∗N+r))x)) using Ia[OF ‹a>0›]
True by auto
      moreover have 0 ≤ (∑ i<N. abs(u 1 ((T^^i)x))) by auto
      ultimately show ?thesis by linarith
    next
      case False
       then have I: u (a∗N+r) x ≤ u r x + (∑ i<a. u N ((T^^(i ∗ N + r))x))
using Iar by auto
      have u r x ≤ (∑ i<r. u 1 ((T^^i)x))
      using subcocycle-bounded-by-birkhoff1[OF assms(1)] False unfolding birkhoff-sum-def
by auto
      also have ... ≤ (∑ i<r. abs(u 1 ((T^^i)x)))
        by (simp add: sum-mono)
      also have ... ≤ (∑ i<N. abs(u 1 ((T^^i)x)))
        apply (rule sum-mono2) using ‹r<N› by auto
      finally show ?thesis using I by auto
    qed
    then have u n x ≤ E1 + (∑ i<a. u N ((T^^(i∗N+r))x)) + E2
      unfolding E1-def using B by auto
  } note ∗ = this

  have I: u N ((T^^j) x) ≤ E2 if j∈{n−2∗N..<a∗N} for j
  proof −
    have u N ((T^^j) x) ≤ (∑ i<N. u 1 ((T^^i) ((T^^j)x)))
     using subcocycle-bounded-by-birkhoff1[OF assms(1) ‹N>0›] unfolding birkhoff-sum-def
by auto
```

also have ... = ($\sum$ *i*<*N*. *u 1* ((*T*^^(*i*+*j*))*x*)) **by** (*simp add*: *funpow-add*)
also have ... ≤ ($\sum$ *i*<*N*. *abs*(*u 1* ((*T*^^(*i*+*j*))*x*))) **by** (*rule sum-mono*, *auto*)
also have ... = ($\sum$ *k*∈{*j*..<*N*+*j*}. *abs*(*u 1* ((*T*^^*k*)*x*)))
by (*rule sum.reindex-bij-betw*, *rule bij-betw-byWitness*[**where** *?f'* = λ*k*. *k*−*j*], *auto*)
also have ... ≤ ($\sum$ *i*∈{*n*−*2*∗*N*..<*n*}. *abs*(*u 1* ((*T*^^*i*) *x*)))
apply (*rule sum-mono2*) **using** ‹*j*∈{*n*−*2*∗*N*..<*a*∗*N*}› ‹*a*∗*N* ≤ *n* − *N*› **by** *auto*
also have ... = *E2* **unfolding** *E2-def*
apply (*rule sum.reindex-bij-betw*[*symmetric*], *rule bij-betw-byWitness*[**where** *?f'* = λ*i*. *i* − (*n*−*2*∗*N*)])
using ‹*n* > *2*∗*N*› **by** *auto*
finally show *?thesis* **by** *auto*
qed
have ($\sum$ *j*<*a*∗*N*. *u N* ((*T*^^*j*) *x*)) − ($\sum$ *j*<*n*−*2*∗*N*. *u N* ((*T*^^*j*) *x*)) = ($\sum$ *j*∈{*n*−*2*∗*N*..<*a*∗*N*}. *u N* ((*T*^^*j*) *x*))
using *sum.atLeastLessThan-concat*[*OF* - ‹*a*∗*N* ≥ *n* − *2*∗*N*›, *of 0* λ*j*. *u N* ((*T*^^*j*) *x*), *symmetric*] *atLeast0LessThan* **by** *simp*
also have ... ≤ ($\sum$ *j*∈{*n*−*2*∗*N*..<*a*∗*N*}. *E2*) **by** (*rule sum-mono*[*OF I*])
also have ... = (*a*∗*N* − (*n*−*2*∗*N*)) ∗ *E2* **by** *simp*
also have ... ≤ *N* ∗ *E2* **using** ‹(*a*∗*N* − (*n*−*2*∗*N*)) ≤ *N*› ‹*E2* ≥ *0*› **by** (*simp add*: *mult-right-mono*)
finally have *J*: ($\sum$ *j*<*a*∗*N*. *u N* ((*T*^^*j*) *x*)) ≤ ($\sum$ *j*<*n*−*2*∗*N*. *u N* ((*T*^^*j*) *x*)) + *N* ∗ *E2* **by** *auto*

have *N* ∗ *u n x* = ($\sum$ *r*<*N*. *u n x*) **by** *auto*
also have ... ≤ ($\sum$ *r*<*N*. *E1* + *E2* + ($\sum$ *i*<*a*. *u N* ((*T*^^(*i*∗*N*+*r*))*x*)))
apply (*rule sum-mono*) **using** ∗ **by** *fastforce*
also have ... = ($\sum$ *r*<*N*. *E1* + *E2*) + ($\sum$ *r*<*N*. ($\sum$ *i*<*a*. *u N* ((*T*^^(*i*∗*N*+*r*))*x*)))
by (*rule sum.distrib*)
also have ... = *N*∗ (*E1* + *E2*) + ($\sum$ *j*<*a*∗*N*. *u N* ((*T*^^*j*) *x*))
using *sum-arith-progression* **by** *auto*
also have ... ≤ *N* ∗(*E1*+*E2*) + ($\sum$ *j*<*n*−*2*∗*N*. *u N* ((*T*^^*j*) *x*)) + *N*∗*E2*
using *J* **by** *auto*
also have ... = *N* ∗ (*E1*+*E2*) + *N* ∗ ($\sum$ *j*<*n*−*2*∗*N*. *u N* ((*T*^^*j*) *x*) / *N*) + *N* ∗ *E2*
using ‹*N*>*0*› **by** (*simp add*: *sum-distrib-left*)
also have ... = *N*∗(*E1* + *E2* + ($\sum$ *j*<*n*−*2*∗*N*. *u N* ((*T*^^*j*) *x*) / *N*) + *E2*)
by (*simp add*: *distrib-left*)
finally have *u n x* ≤ *E1* + *2*∗*E2* + *birkhoff-sum* (λ*x*. *u N x* / *N*) (*n*−*2*∗*N*) *x*
unfolding *birkhoff-sum-def* **using** ‹*N*>*0*› **by** *auto*
then show *?thesis* **unfolding** *E1-def E2-def* **by** *auto*
qed

Many natural cocycles are only defined almost everywhere, and then the subadditivity property only makes sense almost everywhere. We will now show that such an a.e.-subcocycle coincides almost everywhere with a genuine subcocycle in the above sense. Then, all the results for subcocycles will apply to such a.e.-subcocycles. (Usually, in ergodic theory, subcocycles only satisfy the subadditivity property almost everywhere, but we have requested it everywhere for simplicity in the proofs.)

The subcocycle will be defined in a recursive way. This means that is can not be defined in a proof (since complicated function definitions are not available inside proofs). Since it is defined in terms of u, then u has to be available at the top level, which is most conveniently done using a context.

```
context
  fixes u::nat ⇒ 'a ⇒ real
  assumes H: ⋀m n. AE x in M. u (n+m) x ≤ u n x + u m ((T^^n) x)
          ⋀n. integrable M (u n)
begin

private fun v :: nat ⇒ 'a ⇒ real where v n x = (
  if n = 0 then max (u 0 x) 0
  else if n = 1 then u 1 x
  else min (u n x) (Min ((λk. v k x + v (n−k) ((T^^k) x))‘{0<..<n})))

private lemma v0 [simp]:
  ‹v 0 x = max (u 0 x) 0›
  by simp

private lemma v1 [simp]:
  ‹v (Suc 0) x = u 1 x›
  by simp

private lemma v2 [simp]:
  ‹v n x = min (u n x) (Min ((λk. v k x + v (n−k) ((T^^k) x))‘{0<..<n}))› if
‹n ≥ 2›
  using that by (subst v.simps) (simp del: v.simps)

declare v.simps [simp del]

private lemma integrable-v:
  integrable M (v n) for n
proof (induction n rule: nat-less-induct)
  case (1 n)
  consider n = 0 | n = 1 | n>1 by linarith
  then show ?case
  proof (cases)
    assume n = 0
    have v 0 x = max (u 0 x) 0 for x by simp
    then show ?thesis using integrable-max[OF H(2)[of 0]] ‹n = 0› by auto
  next
    assume n = 1
    have v 1 x = u 1 x for x by simp
    then show ?thesis using H(2)[of 1] ‹n = 1› by auto
  next
    assume n > 1
```

```
    hence v n x = min (u n x) (MIN k ∈ {0<..<n}. v k x + v (n−k) ((T^^k) x))
for x
      by simp
    moreover have integrable M (λx. min (u n x) (MIN k ∈ {0<..<n}. v k x +
v (n−k) ((T^^k) x)))
      apply (rule integrable-min)
      apply (simp add: H(2))
      apply (rule integrable-MIN, simp)
      using ‹n >1› apply auto[1]
      apply (rule Bochner-Integration.integrable-add)
      using 1.IH apply auto[1]
      apply (rule Tn-integral-preserving(1))
      using 1.IH by (metis ‹1 < n› diff-less greaterThanLessThan-iff max-0-1(2)
max-less-iff-conj)
    ultimately show ?case by auto
  qed
qed

private lemma u-eq-v-AE:
  AE x in M. v n x = u n x for n
proof (induction n rule: nat-less-induct)
  case (1 n)
  consider n = 0 | n = 1 | n>1 by linarith
  then show ?case
  proof (cases)
    assume n = 0
    have AE x in M. u 0 x ≤ u 0 x + u 0 x using H(1)[of 0 0] by auto
    then have AE x in M. u 0 x ≥ 0 by auto
    moreover have v 0 x = max (u 0 x) 0 for x by simp
    ultimately show ?thesis using ‹n = 0› by auto
  next
    assume n = 1
    have v 1 x = u 1 x for x by simp
    then show ?thesis using ‹n = 1› by simp
  next
    assume n > 1
    {
      fix k assume k<n
      then have AE x in M. v k x = u k x using 1.IH by simp
       with T-AE-iterates[OF this] have AE x in M. ∀ s. v k ((T^^s) x) = u k
((T^^s) x) by simp
    } note * = this
    have AE x in M. ∀ k ∈ {..<n}. ∀ s. v k ((T^^s) x) = u k ((T^^s) x)
      apply (rule AE-finite-allI) using * by simp-all
    moreover have AE x in M. ∀ i j. u (i+j) x ≤ u i x + u j ((T^^i) x)
      apply (subst AE-all-countable, intro allI)+ using H(1) by simp
    moreover
    {
      fix x assume ∀ k ∈ {..<n}. ∀ s. v k ((T^^s) x) = u k ((T^^s) x)
```

```
              ∀ i j. u (i+j) x ≤ u i x + u j ((T^^i) x)
      then have Hx: ⋀k s. k < n ⟹ v k ((T^^s) x) = u k ((T^^s) x)
              ⋀i j. u (i+j) x ≤ u i x + u j ((T^^i) x)
        by auto
      {
        fix k assume k ∈ {0<..<n}
        then have K: k<n n−k<n by auto
          have u n x ≤ u k x + u (n−k) ((T^^k) x) using Hx(2) K by (metis
le-add-diff-inverse less-imp-le-nat)
        also have ... = v k x + v (n−k) ((T^^k)x) using Hx(1)[OF ‹k <n›, of 0]
Hx(1)[OF ‹n−k <n›, of k] by auto
        finally have u n x ≤ v k x + v (n−k) ((T^^k)x) by simp
      }
      then have ∗: ⋀z. z ∈ (λk. v k x + v (n−k) ((T^^k) x))`{0<..<n} ⟹ u n
x ≤ z by blast
      have u n x ≤ Min ((λk. v k x + v (n−k) ((T^^k) x))`{0<..<n})
        apply (rule Min.boundedI) using ‹n>1› ∗ by auto
      moreover have v n x = min (u n x) (Min ((λk. v k x + v (n−k) ((T^^k)
x))`{0<..<n}))
        using ‹1<n› by auto
      ultimately have v n x = u n x by auto
    }
    ultimately show ?thesis by auto
  qed
qed

private lemma subcocycle-v:
  v (n+m) x ≤ v n x + v m ((T^^n) x)
proof −
  consider n = 0 | m = 0 | n>0 ∧ m >0 by auto
  then show ?thesis
  proof (cases)
    case 1
    then have v n x ≥ 0 by simp
    then show ?thesis using ‹n = 0› by auto
  next
    case 2
    then have v m x ≥ 0 by simp
    then show ?thesis using ‹m = 0› by auto
  next
    case 3
    then have n+m > 1 by simp
    then have v (n+m) x = min (u(n+m) x) (Min ((λk. v k x + v ((n+m)−k)
((T^^k) x))`{0<..<n+m})) by simp
    also have ... ≤ Min ((λk. v k x + v ((n+m)−k) ((T^^k) x))`{0<..<n+m})
by simp
    also have ... ≤ v n x + v ((n+m)−n) ((T^^n) x)
      apply (rule Min-le, simp)
      by (metis (lifting) ‹0 < n ∧ 0 < m› add.commute greaterThanLessThan-iff
```

image-iff less-add-same-cancel2)
    finally show *?thesis* **by** *simp*
  qed
qed

lemma *subcocycle-AE-in-context*:
  $\exists\, w.\ subcocycle\ w \wedge (AE\ x\ in\ M.\ \forall\, n.\ w\ n\ x = u\ n\ x)$
proof −
  have *subcocycle v* **using** *subcocycle-v integrable-v* **unfolding** *subcocycle-def* **by** *auto*
  moreover have $AE\ x\ in\ M.\ \forall\, n.\ v\ n\ x = u\ n\ x$
    by (*subst AE-all-countable*, *intro allI*, *rule u-eq-v-AE*)
  ultimately show *?thesis* **by** *blast*
qed

end

lemma *subcocycle-AE*:
  fixes $u{::}nat \Rightarrow {'a} \Rightarrow real$
  assumes $\bigwedge m\ n.\ AE\ x\ in\ M.\ u\ (n{+}m)\ x \leq u\ n\ x + u\ m\ ((T\widehat{\ }\widehat{\ }n)\ x)$
          $\bigwedge n.\ integrable\ M\ (u\ n)$
  shows $\exists\, w.\ subcocycle\ w \wedge (AE\ x\ in\ M.\ \forall\, n.\ w\ n\ x = u\ n\ x)$
using *subcocycle-AE-in-context assms* **by** *blast*

9.2 The asymptotic average

In this subsection, we define the asymptotic average of a subcocycle u, i.e., the limit of $\int u_n(x)/n$ (the convergence follows from subadditivity of $\int u_n$) and study its basic properties, especially in terms of operations on subcocycles. In general, it can be $-\infty$, so we define it in the extended reals.

definition $subcocycle\text{-}avg\text{-}ereal{::}(nat \Rightarrow {'a} \Rightarrow real) \Rightarrow ereal$ **where**
  $subcocycle\text{-}avg\text{-}ereal\ u = Inf\ \{ereal((\int x.\ u\ n\ x\ \partial M)\ /\ n)\ |n.\ n > 0\}$

lemma *subcocycle-avg-finite*:
  $subcocycle\text{-}avg\text{-}ereal\ u < \infty$
unfolding *subcocycle-avg-ereal-def* **using** *Inf-less-iff less-ereal.simps*(*4*) **by** *blast*

lemma *subcocycle-avg-subadditive*:
  assumes *subcocycle u*
  shows $subadditive\ (\lambda n.\ (\int x.\ u\ n\ x\ \partial M))$
unfolding *subadditive-def* **proof** (*intro allI*)
  have *int-u* [*measurable*]: $\bigwedge n.\ integrable\ M\ (u\ n)$ **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
  fix *m n*
  have $(\int x.\ u\ (n{+}m)\ x\ \partial M) \leq (\int x.\ u\ n\ x + u\ m\ ((T\widehat{\ }\widehat{\ }n)\ x)\ \partial M)$
    apply (*rule integral-mono*)
    using *int-u* **apply** (*auto simp add*: *Tn-integral-preserving*(*1*))
    using *assms* **unfolding** *subcocycle-def* **by** *auto*

also have ... ≤ (∫ *x*. *u n x* ∂*M*) + (∫ *x*. *u m* ((*T*^^*n*) *x*) ∂*M*)
using *int-u* **by** (*auto simp add*: *Tn-integral-preserving*(*1*))
also have ... = (∫ *x*. *u n x* ∂*M*) + (∫ *x*. *u m x* ∂*M*)
using *int-u* **by** (*auto simp add*: *Tn-integral-preserving*(*2*))
finally show (∫ *x*. *u* (*n*+*m*) *x* ∂*M*) ≤ (∫ *x*. *u n x* ∂*M*) + (∫ *x*. *u m x* ∂*M*) **by** *simp*
qed

lemma *subcocycle-int-tendsto-avg-ereal*:
assumes *subcocycle u*
shows (λ*n*. (∫ *x*. *u n x* / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal u*
unfolding *subcocycle-avg-ereal-def*
using *subadditive-converges-ereal*[*OF subcocycle-avg-subadditive*[*OF assms*]] **by** *auto*

The average behaves well under addition, scalar multiplication and max, trivially.

lemma *subcocycle-avg-ereal-add*:
assumes *subcocycle u subcocycle v*
shows *subcocycle-avg-ereal* (λ*n x*. *u n x* + *v n x*) = *subcocycle-avg-ereal u* + *subcocycle-avg-ereal v*
proof −
have *int* [*simp*]: ⋀*n*. *integrable M* (*u n*) ⋀*n*. *integrable M* (*v n*) **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
{
fix *n*
have (∫ *x*. *u n x* / *n* ∂*M*) + (∫ *x*. *v n x* / *n* ∂*M*) = (∫ *x*. *u n x* / *n* + *v n x* / *n* ∂*M*)
by (*rule Bochner-Integration.integral-add*[*symmetric*], *auto*)
also have ... = (∫ *x*. (*u n x* + *v n x*) / *n* ∂*M*)
by (*rule Bochner-Integration.integral-cong*, *auto simp add*: *add-divide-distrib*)
finally have *ereal* (∫ *x*. *u n x* / *n* ∂*M*) + (∫ *x*. *v n x* / *n* ∂*M*) = (∫ *x*. (*u n x* + *v n x*) / *n* ∂*M*)
by *auto*
}
moreover have (λ*n*. *ereal* (∫ *x*. *u n x* / *n* ∂*M*) + (∫ *x*. *v n x* / *n* ∂*M*))
⟶ *subcocycle-avg-ereal u* + *subcocycle-avg-ereal v*
apply (*intro tendsto-intros subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)] *subcocycle-int-tendsto-avg-ereal*[*OF assms*(*2*)])
using *subcocycle-avg-finite* **by** *auto*
ultimately have (λ*n*. (∫ *x*. (*u n x* + *v n x*) / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal u* + *subcocycle-avg-ereal v*
by *auto*
moreover have (λ*n*. (∫ *x*. (*u n x* + *v n x*) / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal* (λ*n x*. *u n x* + *v n x*)
by (*rule subcocycle-int-tendsto-avg-ereal*[*OF subcocycle-add*[*OF assms*]])
ultimately show *?thesis* **using** *LIMSEQ-unique* **by** *blast*
qed

lemma *subcocycle-avg-ereal-cmult*:

assumes *subcocycle u c* $\geq$ (*0*::*real*)
shows *subcocycle-avg-ereal* (λ*n x. c* $*$ *u n x*) = *c* $*$ *subcocycle-avg-ereal u*
proof (*cases c* = *0*)
case *True*
have $*$: *ereal* ($\int$ *x.* (*c* $*$ *u n x*) / *n* ∂*M*) = *0* **if** *n*>*0* **for** *n*
by (*subst True*, *auto*)
have (λ*n. ereal* ($\int$ *x.* (*c* $*$ *u n x*) / *n* ∂*M*)) $\longrightarrow$ *0*
by (*subst lim-explicit*, *metis* $*$ *less-le-trans zero-less-one*)
moreover have (λ*n. ereal* ($\int$ *x.* (*c* $*$ *u n x*) / *n* ∂*M*)) $\longrightarrow$ *subcocycle-avg-ereal* (λ*n x. c* $*$ *u n x*)
using *subcocycle-int-tendsto-avg-ereal*[*OF subcocycle-cmult*[*OF assms*]] **by** *auto*
ultimately have *subcocycle-avg-ereal* (λ*n x. c* $*$ *u n x*) = *0*
using *LIMSEQ-unique* **by** *blast*
then show *?thesis* **using** *True* **by** *auto*
next
case *False*
have *int*: $\bigwedge$*n. integrable M* (*u n*) **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
have *ereal* ($\int$ *x. c* $*$ *u n x* / *n* ∂*M*) = *c* $*$ *ereal* ($\int$ *x. u n x* / *n* ∂*M*) **for** *n* **by** *auto*
then have (λ*n. c* $*$ *ereal* ($\int$ *x. u n x* / *n* ∂*M*)) $\longrightarrow$ *subcocycle-avg-ereal* (λ*n x. c* $*$ *u n x*)
using *subcocycle-int-tendsto-avg-ereal*[*OF subcocycle-cmult*[*OF assms*]] **by** *auto*
moreover have (λ*n. c* $*$ *ereal* ($\int$ *x. u n x* / *n* ∂*M*)) $\longrightarrow$ *c* $*$ *subcocycle-avg-ereal u*
apply (*rule tendsto-mult-ereal*) **using** *False subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)] **by** *auto*
ultimately show *?thesis* **using** *LIMSEQ-unique* **by** *blast*
qed

lemma *subcocycle-avg-ereal-max*:
assumes *subcocycle u subcocycle v*
shows *subcocycle-avg-ereal* (λ*n x. max* (*u n x*) (*v n x*)) $\geq$ *max* (*subcocycle-avg-ereal u*) (*subcocycle-avg-ereal v*)
proof (*auto*)
have *int*: *integrable M* (*u n*) *integrable M* (*v n*) **for** *n* **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
have *int2*: *integrable M* (λ*x. max* (*u n x*) (*v n x*)) **for** *n* **using** *integrable-max int* **by** *auto*

have ($\int$ *x. u n x* / *n* ∂*M*) $\leq$ ($\int$ *x. max* (*u n x*) (*v n x*) / *n* ∂*M*) **for** *n*
apply (*rule integral-mono*) **using** *int int2* **by** (*auto simp add*: *divide-simps*)
then show *subcocycle-avg-ereal u* $\leq$ *subcocycle-avg-ereal* (λ*n x. max* (*u n x*) (*v n x*))
using *LIMSEQ-le*[*OF subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)]
subcocycle-int-tendsto-avg-ereal[*OF subcocycle-max*[*OF assms*]]] **by** *auto*

have ($\int$ *x. v n x* / *n* ∂*M*) $\leq$ ($\int$ *x. max* (*u n x*) (*v n x*) / *n* ∂*M*) **for** *n*
apply (*rule integral-mono*) **using** *int int2* **by** (*auto simp add*: *divide-simps*)
then show *subcocycle-avg-ereal v* $\leq$ *subcocycle-avg-ereal* (λ*n x. max* (*u n x*) (*v*

n x))
  using *LIMSEQ-le*[*OF subcocycle-int-tendsto-avg-ereal*[*OF assms*(*2*)]
    subcocycle-int-tendsto-avg-ereal[*OF subcocycle-max*[*OF assms*]]] **by** *auto*
qed

For a Birkhoff sum, the average at each time is the same, equal to the average of the function, so the asymptotic average is also equal to this common value.

lemma *subcocycle-avg-ereal-birkhoff*:
 assumes *integrable M u*
 shows *subcocycle-avg-ereal* (*birkhoff-sum u*) = ($\int$ *x. u x* ∂*M*)
proof −
 have ∗: *ereal* ($\int$ *x.* (*birkhoff-sum u n x*) / *n* ∂*M*) = ($\int$ *x. u x* ∂*M*) **if** *n>0* **for** *n*
  using *birkhoff-sum-integral*(*2*)[*OF assms*] *that* **by** *auto*
 have (λ*n. ereal* ($\int$ *x.* (*birkhoff-sum u n x*) / *n* ∂*M*)) ⟶ ($\int$ *x. u x* ∂*M*)
  by (*subst lim-explicit*, *metis* ∗ *less-le-trans zero-less-one*)
 moreover have (λ*n. ereal* ($\int$ *x.* (*birkhoff-sum u n x*) / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal* (*birkhoff-sum u*)
  using *subcocycle-int-tendsto-avg-ereal*[*OF subcocycle-birkhoff*[*OF assms*]] **by** *auto*
 ultimately show *?thesis* **using** *LIMSEQ-unique* **by** *blast*
qed

In nice situations, where one can avoid the use of ereal, the following definition is more convenient. The kind of statements we are after is as follows: if the ereal average is finite, then something holds, likely involving the real average.

In particular, we show in this setting what we have proved above under this new assumption: convergence (in real numbers) of the average to the asymptotic average, as well as good behavior under sum, scalar multiplication by positive numbers, max, formula for Birkhoff sums.

definition *subcocycle-avg*::(*nat* ⇒ *'a* ⇒ *real*) ⇒ *real* **where**
 subcocycle-avg u = *real-of-ereal*(*subcocycle-avg-ereal u*)

lemma *subcocycle-avg-real-ereal*:
 assumes *subcocycle-avg-ereal u* > − ∞
 shows *subcocycle-avg-ereal u* = *ereal*(*subcocycle-avg u*)
unfolding *subcocycle-avg-def* **using** *assms subcocycle-avg-finite*[*of u*] *ereal-real* **by** *auto*

lemma *subcocycle-int-tendsto-avg*:
 assumes *subcocycle u subcocycle-avg-ereal u* > − ∞
 shows (λ*n.* ($\int$ *x. u n x* / *n* ∂*M*)) ⟶ *subcocycle-avg u*
using *subcocycle-avg-real-ereal*[*OF assms*(*2*)] *subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)] **by** *auto*

lemma *subcocycle-avg-add*:
 assumes *subcocycle u subcocycle v subcocycle-avg-ereal u* > − ∞ *subcocycle-avg-ereal v* > − ∞

shows *subcocycle-avg-ereal* ($\lambda n\ x.\ u\ n\ x + v\ n\ x$) > $-\infty$
subcocycle-avg ($\lambda n\ x.\ u\ n\ x + v\ n\ x$) = *subcocycle-avg u* + *subcocycle-avg v*
using *assms subcocycle-avg-finite real-of-ereal-add*
unfolding *subcocycle-avg-def subcocycle-avg-ereal-add*[*OF assms*(*1*) *assms*(*2*)] **by** *auto*

lemma *subcocycle-avg-cmult*:
assumes *subcocycle u c* $\geq$ (*0*::*real*) *subcocycle-avg-ereal u* > $-\infty$
shows *subcocycle-avg-ereal* ($\lambda n\ x.\ c * u\ n\ x$) > $-\infty$
subcocycle-avg ($\lambda n\ x.\ c * u\ n\ x$) = c * *subcocycle-avg u*
using *assms subcocycle-avg-finite* **unfolding** *subcocycle-avg-def subcocycle-avg-ereal-cmult*[*OF assms*(*1*) *assms*(*2*)] **by** *auto*

lemma *subcocycle-avg-max*:
assumes *subcocycle u subcocycle v subcocycle-avg-ereal u* > $-\infty$ *subcocycle-avg-ereal v* > $-\infty$
shows *subcocycle-avg-ereal* ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$) > $-\infty$
subcocycle-avg ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$) $\geq$ *max* (*subcocycle-avg u*) (*subcocycle-avg v*)
proof −
show ∗: *subcocycle-avg-ereal* ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$) > $-\infty$
using *assms*(*3*) *subcocycle-avg-ereal-max*[*OF assms*(*1*) *assms*(*2*)] **by** *auto*
have *ereal* (*subcocycle-avg* ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$)) $\geq$ *max* (*ereal*(*subcocycle-avg u*)) (*ereal*(*subcocycle-avg v*))
using *subcocycle-avg-real-ereal*[*OF assms*(*3*)] *subcocycle-avg-real-ereal*[*OF assms*(*4*)]
subcocycle-avg-real-ereal[*OF* ∗] *subcocycle-avg-ereal-max*[*OF assms*(*1*) *assms*(*2*)]
by *auto*
then show *subcocycle-avg* ($\lambda n\ x.\ max\ (u\ n\ x)\ (v\ n\ x)$) $\geq$ *max* (*subcocycle-avg u*) (*subcocycle-avg v*)
by *auto*
qed

lemma *subcocycle-avg-birkhoff*:
assumes *integrable M u*
shows *subcocycle-avg-ereal* (*birkhoff-sum u*) > $-\infty$
subcocycle-avg (*birkhoff-sum u*) = ($\int x.\ u\ x\ \partial M$)
unfolding *subcocycle-avg-def subcocycle-avg-ereal-birkhoff*[*OF assms*(*1*)] **by** *auto*

end

9.3 Almost sure convergence of subcocycles

In this paragraph, we prove Kingman's theorem, i.e., the almost sure convergence of subcocycles. Their limit is almost surely invariant. There is no really easy proof. The one we use below is arguably the simplest known one, due to Steele (1989). The idea is to show that the limsup of the subcocycle is bounded by the liminf (which is almost surely constant along trajectories), by using subadditivity along time intervals where the liminf is almost reached, of length at most N. For some points, the liminf takes a large time $> N$ to be reached. We neglect those times, introducing an additional error that gets smaller with N, thanks to Birkhoff ergodic theorem applied to the set of bad points. The error is most easily managed if the subcocycle is assumed to be nonpositive, which one can assume in a first step. The general case is reduced to this one by replacing u_n with $u_n - S_n u_1 \leq 0$, and using Birkhoff theorem to control $S_n u_1$.

context *fmpt* **begin**

First, as explained above, we prove the theorem for nonpositive subcocycles.

lemma *kingman-theorem-AE-aux1*:
  assumes *subcocycle u*
    $\bigwedge$*x. u 1 x ≤ 0*
  shows ∃(*g*::'*a*⇒*ereal*). (*g*∈*borel-measurable Invariants* ∧ (∀ *x. g x* < ∞) ∧ (*AE x in M*. (λ*n. u n x* / *n*) ⟶ *g x*))
proof −
  define *l* **where** *l* = (λ*x. liminf* (λ*n. u n x* / *n*))
  have *u-meas* [*measurable*]: $\bigwedge$*n. u n* ∈ *borel-measurable M* **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
  have *l-meas* [*measurable*]: *l* ∈ *borel-measurable M* **unfolding** *l-def* **by** *auto*

  {
    fix *x* **assume** ∗: (λ*n. birkhoff-sum* (*u 1*) *n x* / *n*) ⟶ *real-cond-exp M Invariants* (*u 1*) *x*
    then have (λ*n. birkhoff-sum* (*u 1*) *n x* / *n*) ⟶ *ereal*(*real-cond-exp M Invariants* (*u 1*) *x*)
      by *auto*
    then have *a*: *liminf* (λ*n. birkhoff-sum* (*u 1*) *n x* / *n*) = *ereal*(*real-cond-exp M Invariants* (*u 1*) *x*)
      using *lim-imp-Liminf* **by** *force*

    have *ereal*(*u n x* / *n*) ≤ *ereal*(*birkhoff-sum* (*u 1*) *n x* / *n*) **if** *n*>*0* **for** *n*
      using *subcocycle-bounded-by-birkhoff1*[*OF assms*(*1*) *that*, *of x*] *that* **by** (*simp add*: *divide-right-mono*)
    with *eventually-mono*[*OF eventually-gt-at-top*[*of 0*] *this*]
    have *eventually* (λ*n. ereal*(*u n x* / *n*) ≤ *ereal*(*birkhoff-sum* (*u 1*) *n x* / *n*)) *sequentially* **by** *auto*
    then have *liminf* (λ*n. u n x* / *n*) ≤ *liminf* (λ*n. birkhoff-sum* (*u 1*) *n x* / *n*)
      by (*simp add*: *Liminf-mono*)
    then have *l x* < ∞ **unfolding** *l-def* **using** *a* **by** *auto*
  }
  then have *AE x in M*. *l x* < ∞
    using *birkhoff-theorem-AE-nonergodic*[*of u 1*] *subcocycle-def assms*(*1*) **by** *auto*

  have *l-dec*: *l x* ≤ *l* (*T x*) **for** *x*
  proof −
    have *l x* = *liminf* (λ*n. ereal* ((*u* (*n*+*1*) *x*)/(*n*+*1*)))
      unfolding *l-def* **by** (*rule liminf-shift*[*of* λ*n. ereal* (*u n x* / *real n*), *symmetric*])

also have ... ≤ *liminf* (*λn. ereal*((*u 1 x*)/(*n+1*)) + *ereal*((*u n* (*T x*))/(*n+1*)))
proof (*rule Liminf-mono*[*OF eventuallyI*])
fix *n*
have *u* (*1+n*) *x* ≤ *u 1 x* + *u n* ((*T*^^*1*) *x*) **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *blast*
then have *u* (*n+1*) *x* ≤ *u 1 x* + *u n* (*T x*) **by** *auto*
then have (*u* (*n+1*) *x*)/(*n+1*) ≤ (*u 1 x*)/(*n+1*) + (*u n* (*T x*))/(*n+1*)
by (*metis add-divide-distrib divide-right-mono of-nat-0-le-iff*)
then show *ereal* ((*u* (*n+1*) *x*)/(*n+1*)) ≤ *ereal*((*u 1 x*)/(*n+1*)) + *ereal*((*u n* (*T x*))/(*n+1*)) **by** *auto*
qed
also have ... = *0* + *liminf*(*λn. ereal*((*u n* (*T x*))/(*n+1*)))
proof (*rule ereal-liminf-lim-add*[*of λn. ereal*((*u 1 x*)/*real*(*n+1*)) *0 λn. ereal*((*u n* (*T x*))/(*n+1*))])
have (*λn. ereal*((*u 1 x*)∗(*1*/*real*(*n+1*)))) ⟶ *ereal*((*u 1 x*) ∗ *0*)
by (*intro tendsto-intros LIMSEQ-ignore-initial-segment*)
then show (*λn. ereal*((*u 1 x*)/*real*(*n+1*))) ⟶ *0* **by** (*simp add*: *zero-ereal-def*)
qed (*simp*)
also have ... = *1* ∗ *liminf*(*λn. ereal*((*u n* (*T x*))/(*n+1*))) **by** *simp*
also have ... = *liminf*(*λn.* (*n+1*)/*n* ∗ *ereal*((*u n* (*T x*))/(*n+1*)))
proof (*rule ereal-liminf-lim-mult*[*symmetric*])
have *real* (*n+1*) / *real n* = *1* + *1*/*real n* **if** *n*>*0* **for** *n* **by** (*simp add*: *divide-simps mult.commute that*)
with *eventually-mono*[*OF eventually-gt-at-top*[*of 0*::*nat*] *this*]
have *eventually* (*λn. real* (*n+1*) / *real n* = *1* + *1*/*real n*) *sequentially* **by** *simp*
moreover have (*λn. 1* + *1*/*real n*) ⟶ *1* + *0*
by (*intro tendsto-intros*)
ultimately have (*λn. real* (*n+1*) / *real n*) ⟶ *1* **using** *Lim-transform-eventually* **by** (*simp add*: *filterlim-cong*)
then show (*λn. ereal*(*real* (*n+1*) / *real n*)) ⟶ *1* **by** (*simp add*: *one-ereal-def*)
qed (*auto*)
also have ... = *l* (*T x*) **unfolding** *l-def* **by** *auto*
finally show *l x* ≤ *l* (*T x*) **by** *simp*
qed
have *AE x in M. l* (*T x*) = *l x*
apply (*rule AE-increasing-then-invariant*) **using** *l-dec* **by** *auto*
then obtain *g0* **where** *g0*: *g0* ∈ *borel-measurable Invariants AE x in M. l x* = *g0 x*
using *Invariants-quasi-Invariants-functions*[*OF l-meas*] **by** *auto*
define *g* **where** *g* = (*λx. if g0 x* = ∞ *then 0 else g0 x*)
have *g*: *g* ∈ *borel-measurable Invariants AE x in M. g x* = *l x*
unfolding *g-def* **using** *g0*(*1*) ‹*AE x in M. l x* = *g0 x*› ‹*AE x in M. l x* < ∞› **by** *auto*
have [*measurable*]: *g* ∈ *borel-measurable M* **using** *g*(*1*) *Invariants-measurable-func* **by** *blast*
have ⋀*x. g x* < ∞ **unfolding** *g-def* **by** *auto*

define *A* **where** $A = \{x \in space\ M.\ l\ x < \infty \wedge (\forall n.\ l\ ((T\widehat{\ }\widehat{\ }n)\ x) = g\ ((T\widehat{\ }\widehat{\ }n)\ x))\}$
have *A-meas* [*measurable*]: $A \in sets\ M$ **unfolding** *A-def* **by** *auto*
have *AE x in M. x* ∈ *A* **unfolding** *A-def* **using** *T-AE-iterates*[*OF g*(*2*)] ‹*AE x in M. l x* < ∞› **by** *auto*
then have *space M* − *A* ∈ *null-sets M* **by** (*simp add*: *AE-iff-null set-diff-eq*)

have *l-inv*: $l((T\widehat{\ }\widehat{\ }n)\ x) = l\ x$ **if** $x \in A$ **for** *x n*
proof −
have $l((T\widehat{\ }\widehat{\ }n)\ x) = g((T\widehat{\ }\widehat{\ }n)\ x)$ **using** ‹ $x \in A$ › **unfolding** *A-def* **by** *blast*
also have ... = *g x* **using** *g*(*1*) *A-def Invariants-func-is-invariant-n that* **by** *blast*
also have ... = $g((T\widehat{\ }\widehat{\ }0)\ x)$ **by** *auto*
also have ... = $l((T\widehat{\ }\widehat{\ }0)\ x)$ **using** ‹ $x \in A$ › **unfolding** *A-def* **by** (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq*)
finally show *?thesis* **by** *auto*
qed

define *F* **where** $F = (\lambda\ K\ e\ x.\ real\text{-}of\text{-}ereal(max\ (l\ x)\ (-ereal\ K)) + e)$
have *F-meas* [*measurable*]: *F K e* ∈ *borel-measurable M* **for** *K e* **unfolding** *F-def* **by** *auto*
define *B* **where** $B = (\lambda N\ K\ e.\ \{x \in A.\ \exists n \in \{1..N\}.\ u\ n\ x - n * F\ K\ e\ x < 0\})$
have *B-meas* [*measurable*]: *B N K e* ∈ *sets M* **for** *N K e* **unfolding** *B-def* **by** (*measurable*)
define *I* **where** $I = (\lambda N\ K\ e\ x.\ (indicator\ (-\ B\ N\ K\ e)\ x)::real)$
have *I-meas* [*measurable*]: *I N K e* ∈ *borel-measurable M* **for** *N K e* **unfolding** *I-def* **by** *auto*
have *I-int*: *integrable M* (*I N K e*) **for** *N K e*
unfolding *I-def* **apply** (*subst Bochner-Integration.integrable-cong*[**where** *?g* = *indicator* (*space M* − *B N K e*)::- ⇒ *real*], *auto*)
by (*auto split*: *split-indicator simp*: *less-top*[*symmetric*])

have *main*: *AE x in M*. $limsup\ (\lambda n.\ u\ n\ x\ /\ n) \leq F\ K\ e\ x + abs(F\ K\ e\ x) * ereal(real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ N\ K\ e)\ x)$
if $N>(1::nat)$ $K>(0::real)$ $e>(0::real)$ **for** *N K e*
proof −
let *?B* = *B N K e* **and** *?I* = *I N K e* **and** *?F* = *F K e*

define *t* **where** $t = (\lambda x.\ if\ x \in ?B\ then\ Min\ \{n \in \{1..N\}.\ u\ n\ x - n * ?F\ x < 0\}\ else\ 1)$
have [*measurable*]: *t* ∈ *measurable M* (*count-space UNIV*) **unfolding** *t-def* **by** *measurable*
have *t1*: $t\ x \in \{1..N\}$ **for** *x*
proof (*cases x* ∈ *?B*)
case *False*
then have *t x* = *1* **by** (*simp add*: *t-def*)
then show *?thesis* **using** ‹*N*>*1*›**by** *auto*
next
case *True*

let *?A* = {*n*∈{*1*..*N*}. *u n x* − *n* ∗ *?F x* < *0*}
have *t x* = *Min ?A* **using** *True* **by** (*simp add*: *t-def*)
moreover have *Min ?A* ∈ *?A* **apply** (*rule Min-in*, *simp*) **using** *True B-def* **by** *blast*
ultimately show *?thesis* **by** *auto*
qed

have *bound1*: *u* (*t x*) *x* ≤ *t x* ∗ *?F x* + *birkhoff-sum ?I* (*t x*) *x* ∗ *abs*(*?F x*) **for** *x*
proof (*cases x* ∈ *?B*)
case *True*
let *?A* = {*n*∈{*1*..*N*}. *u n x* − *n* ∗ *F K e x* < *0*}
have *t x* = *Min ?A* **using** *True* **by** (*simp add*: *t-def*)
moreover have *Min ?A* ∈ *?A* **apply** (*rule Min-in*, *simp*) **using** *True B-def* **by** *blast*
ultimately have *u* (*t x*) *x* ≤ (*t x*) ∗ *?F x* **by** *auto*
moreover have *0* ≤ *birkhoff-sum ?I* (*t x*) *x* ∗ *abs*(*?F x*) **unfolding** *birkhoff-sum-def I-def* **by** (*simp add*: *sum-nonneg*)
ultimately show *?thesis* **by** *auto*
next
case *False*
then have *0* ≤ *?F x* + *?I x* ∗ *abs*(*?F x*) **unfolding** *I-def* **by** *auto*
then have *u 1 x* ≤ *?F x* + *?I x* ∗ *abs*(*?F x*) **using** *assms*(*2*)[*of x*] **by** *auto*
moreover have *t x* = *1* **unfolding** *t-def* **using** *False* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

define *TB* **where** *TB* = (λ*x*. (*T*^^(*t x*)) *x*)
have [*measurable*]: *TB* ∈ *measurable M M* **unfolding** *TB-def* **by** *auto*
define *S* **where** *S* = (λ*n x*. (∑ *i*<*n*. *t*((*TB*^^*i*) *x*)))
have [*measurable*]: *S n* ∈ *measurable M* (*count-space UNIV*) **for** *n* **unfolding** *S-def* **by** *measurable*
have *TB-pow*: (*TB*^^*n*) *x* = (*T*^^(*S n x*)) *x* **for** *n x*
unfolding *S-def TB-def*
by (*induction n*, *auto*, *metis* (*mono-tags*, *lifting*) *add.commute funpow-add o-apply*)

have *uS*: *u* (*S n x*) *x* ≤ (*S n x*) ∗ *?F x* + *birkhoff-sum ?I* (*S n x*) *x* ∗ *abs*(*?F x*) **if** *x* ∈ *A n*>*0* **for** *x n*
using ‹*n* > *0*› **proof** (*induction rule*: *ind-from-1*)
case *1*
show *?case* **unfolding** *S-def* **using** *bound1* **by** *auto*
next
case (*Suc n*)
have ∗: *?F*((*TB*^^*n*) *x*) = *?F x* **apply** (*subst TB-pow*) **unfolding** *F-def* **using** *l-inv*[*OF* ‹*x*∈*A*›] **by** *auto*
have ∗∗: *S n x* + *t* ((*TB*^^*n*) *x*) = *S* (*Suc n*) *x* **unfolding** *S-def* **by** *auto*
have *u* (*S* (*Suc n*) *x*) *x* = *u* (*S n x* + *t*((*TB*^^*n*) *x*)) *x* **unfolding** *S-def* **by** *auto*

also have ... ≤ *u* (*S n x*) *x* + *u* (*t*((*TB*^^*n*) *x*)) ((*T*^^(*S n x*)) *x*)
using *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
also have ... ≤ *u* (*S n x*) *x* + *u* (*t*((*TB*^^*n*) *x*)) ((*TB*^^*n*) *x*)
using *TB-pow* **by** *auto*
also have ... ≤ (*S n x*) ∗ *?F x* + *birkhoff-sum ?I* (*S n x*) *x* ∗ *abs*(*?F x*) + *t* ((*TB*^^*n*) *x*) ∗ *?F* ((*TB*^^*n*) *x*) + *birkhoff-sum ?I* (*t* ((*TB*^^*n*) *x*)) ((*TB*^^*n*) *x*) ∗ *abs*(*?F* ((*TB*^^*n*) *x*))
using *Suc bound1*[*of* ((*TB*^^*n*) *x*)] **by** *auto*
also have ... = (*S n x*) ∗ *?F x* + *birkhoff-sum ?I* (*S n x*) *x* ∗ *abs*(*?F x*) + *t* ((*TB*^^*n*) *x*) ∗ *?F x* + *birkhoff-sum ?I* (*t* ((*TB*^^*n*) *x*)) ((*T*^^(*S n x*)) *x*) ∗ *abs*(*?F x*)
using ∗ *TB-pow* **by** *auto*
also have ... = (*real*(*S n x*) + *t* ((*TB*^^*n*) *x*)) ∗ *?F x* + (*birkhoff-sum ?I* (*S n x*) *x* + *birkhoff-sum ?I* (*t* ((*TB*^^*n*) *x*)) ((*T*^^(*S n x*)) *x*)) ∗ *abs*(*?F x*)
by (*simp add*: *mult.commute ring-class.ring-distribs*(*1*))
also have ... = (*S n x* + *t* ((*TB*^^*n*) *x*)) ∗ *?F x* + (*birkhoff-sum ?I* (*S n x*) *x* + *birkhoff-sum ?I* (*t* ((*TB*^^*n*) *x*)) ((*T*^^(*S n x*)) *x*)) ∗ *abs*(*?F x*)
by *simp*
also have ... = (*S* (*Suc n*) *x*) ∗ *?F x* + *birkhoff-sum ?I* (*S* (*Suc n*) *x*) *x* ∗ *abs*(*?F x*)
by (*subst birkhoff-sum-cocycle*[*symmetric*], *subst* ∗∗, *subst* ∗∗, *simp*)
finally show *?case* **by** *simp*
qed

have *un*: *u n x* ≤ *n* ∗ *?F x* + *N* ∗ *abs*(*?F x*) + *birkhoff-sum ?I n x* ∗ *abs*(*?F x*) **if** *x* ∈ *A n*>*N* **for** *x n*
proof −
let *?A* = {*i*. *S i x* > *n*}
let *?iA* = *Inf ?A*
have *n* < (∑ *i*<*n* + *1*. *1*) **by** *auto*
also have ... ≤ *S* (*n*+*1*) *x* **unfolding** *S-def* **apply** (*rule sum-mono*) **using** *t1* **by** *auto*
finally have *?A* ≠ {} **by** *blast*
then have *?iA* ∈ *?A* **by** (*meson Inf-nat-def1*)
moreover have *0* ∉ *?A* **unfolding** *S-def* **by** *auto*
ultimately have *?iA* ≠ *0* **by** *fastforce*
define *j* **where** *j* = *?iA* − *1*
then have *j* < *?iA* **using** ‹*?iA* ≠ *0*› **by** *auto*
then have *j* ∉ *?A* **by** (*meson bdd-below-def cInf-lower le0 not-less*)
then have *S j x* ≤ *n* **by** *auto*
define *k* **where** *k* = *n* − *S j x*
have *n* = *S j x* + *k* **unfolding** *k-def* **using** ‹*S j x* ≤ *n*› **by** *auto*
have *n* < *S* (*j*+*1*) *x* **unfolding** *j-def* **using** ‹*?iA* ≠ *0*› ‹*?iA* ∈ *?A*› **by** *auto*
also have ... = *S j x* + *t*((*TB*^^*j*) *x*) **unfolding** *S-def* **by** *auto*
also have ... ≤ *S j x* + *N* **using** *t1* **by** *auto*
finally have *k* ≤ *N* **unfolding** *k-def* **using** ‹*n* > *N*› **by** *auto*
then have *S j x* > *0* **unfolding** *k-def* **using** ‹*n* > *N*› **by** *auto*

then have *j* > *0* **unfolding** *S-def* **using** *not-gr0* **by** *fastforce*

have *birkhoff-sum ?I* (*S j x*) *x* ≤ *birkhoff-sum ?I n x*
unfolding *birkhoff-sum-def I-def* **using** ‹*S j x* ≤ *n*›
by (*metis finite-Collect-less-nat indicator-pos-le lessThan-def lessThan-subset-iff sum-mono2*)

have *u n x* ≤ *u* (*S j x*) *x*
proof (*cases k* = *0*)
case *True*
show *?thesis* **using** *True* **unfolding** *k-def* **using** ‹*S j x* ≤ *n*› **by** *auto*
next
case *False*
then have *k* > *0* **by** *simp*
have *u k* ((*T*^^(*S j x*)) *x*) ≤ *birkhoff-sum* (*u 1*) *k* ((*T* ^^ *S j x*) *x*)
using *subcocycle-bounded-by-birkhoff1*[*OF assms*(*1*) ‹*k*>*0*›, *of* (*T*^^(*S j x*)) *x*] **by** *simp*
also have ... ≤ *0* **unfolding** *birkhoff-sum-def* **using** *sum-mono assms*(*2*) **by** (*simp add*: *sum-nonpos*)
also have *u n x* ≤ *u* (*S j x*) *x* + *u k* ((*T*^^(*S j x*)) *x*)
apply (*subst* ‹*n* = *S j x* + *k*›) **using** *assms*(*1*) *subcocycle-def* **by** *auto*
ultimately show *?thesis* **by** *auto*
qed
also have ... ≤ (*S j x*) ∗ *?F x* + *birkhoff-sum ?I* (*S j x*) *x* ∗ *abs*(*?F x*)
using *uS*[*OF* ‹*x* ∈ *A*› ‹*j*>*0*›] **by** *simp*
also have ... ≤ (*S j x*) ∗ *?F x* + *birkhoff-sum ?I n x* ∗ *abs*(*?F x*)
using ‹*birkhoff-sum ?I* (*S j x*) *x* ≤ *birkhoff-sum ?I n x*› **by** (*simp add*: *mult-right-mono*)
also have ... = *n* ∗ *?F x* − *k* ∗ *?F x* + *birkhoff-sum ?I n x* ∗ *abs*(*?F x*)
by (*metis* ‹*n* = *S j x* + *k*› *add-diff-cancel-right′ le-add2 left-diff-distrib′ of-nat-diff*)
also have ... ≤ *n* ∗ *?F x* + *k* ∗ *abs*(*?F x*) + *birkhoff-sum ?I n x* ∗ *abs*(*?F x*)
by (*auto*, *metis abs-ge-minus-self abs-mult abs-of-nat*)
also have ... ≤ *n* ∗ *?F x* + *N* ∗ *abs*(*?F x*) + *birkhoff-sum ?I n x* ∗ *abs*(*?F x*)
using ‹*k* ≤ *N*› **by** (*simp add*: *mult-right-mono*)
finally show *?thesis* **by** *simp*
qed

have *limsup* (λ*n*. *u n x* / *n*) ≤ *?F x* + *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x* / *n*)) **if** *x* ∈ *A* **for** *x*
proof −
have (λ*n*. *ereal*(*?F x* + *N* ∗ *abs*(*?F x*) ∗ (*1* / *n*))) ⟶ *ereal*(*?F x* + *N* ∗ *abs* (*?F x*) ∗ *0*)
by (*intro tendsto-intros*)
then have ∗: *limsup* (λ*n*. *?F x* + *N* ∗ *abs*(*?F x*)/*n*) = *?F x*
using *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *force*

{
fix *n* **assume** *n* > *N*

have *u n x / real n* ≤ *?F x* + *N* ∗ *abs*(*?F x*) / *n* + *abs*(*?F x*) ∗ *birkhoff-sum ?I n x / n*
using *un*[*OF* ‹*x* ∈ *A*› ‹*n* > *N*›] **using** ‹*n*>*N*› **by** (*auto simp add*: *divide-simps mult.commute*)
then have *ereal*(*u n x*/*n*) ≤ *ereal*(*?F x* + *N* ∗ *abs*(*?F x*) / *n*) + *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*)
by *auto*
}
then have *eventually* (λ*n*. *ereal*(*u n x* / *n*) ≤ *ereal*(*?F x* + *N* ∗ *abs*(*?F x*) / *n*) + *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*)) *sequentially*
using *eventually-mono*[*OF eventually-gt-at-top*[*of N*]] **by** *auto*
with *Limsup-mono*[*OF this*]
have *limsup* (λ*n*. *u n x* / *n*) ≤ *limsup* (λ*n*. *ereal*(*?F x* + *N* ∗ *abs*(*?F x*) / *n*) + *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*))
by *auto*
also have ... ≤ *limsup* (λ*n*. *?F x* + *N* ∗ *abs*(*?F x*) / *n*) + *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*))
by (*rule ereal-limsup-add-mono*)
also have ... = *?F x* + *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*))
using ∗ **by** *auto*
finally show *?thesis* **by** *auto*
qed
then have ∗: *AE x in M*. *limsup* (λ*n*. *u n x* / *n*) ≤ *?F x* + *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*))
using ‹*AE x in M*. *x* ∈ *A*› **by** *auto*

{
fix *x* **assume** *H*: (λ*n*. *birkhoff-sum ?I n x / n*) ⟶ *real-cond-exp M Invariants ?I x*
have (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*)) ⟶ *abs*(*?F x*) ∗ *ereal*(*real-cond-exp M Invariants ?I x*)
by (*rule tendsto-mult-ereal*, *auto simp add*: *H*)
then have *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*)) = *abs*(*?F x*) ∗ *ereal*(*real-cond-exp M Invariants ?I x*)
using *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *blast*
}
moreover have *AE x in M*. (λ*n*. *birkhoff-sum ?I n x / n*) ⟶ *real-cond-exp M Invariants ?I x*
by (*rule birkhoff-theorem-AE-nonergodic*[*OF I-int*])
ultimately have *AE x in M*. *limsup* (λ*n*. *abs*(*?F x*) ∗ *ereal*(*birkhoff-sum ?I n x / n*)) = *abs*(*?F x*) ∗ *ereal*(*real-cond-exp M Invariants ?I x*)
by *auto*
then show *?thesis* **using** ∗ **by** *auto*
qed

have *bound2*: *AE x in M*. *limsup* (λ*n*. *u n x* / *n*) ≤ *F K e x* **if** *K* > *0 e* > *0* **for** *K e*
proof −

define C **where** $C = (\lambda N.\ A - B\ N\ K\ e)$
have *C-meas* [*measurable*]: $\bigwedge N.\ C\ N \in sets\ M$ **unfolding** *C-def* **by** *auto*
{
fix x **assume** $x \in A$
have $F\ K\ e\ x > l\ x$ **using** ‹$x \in A$› ‹$e > 0$› **unfolding** *F-def A-def*
by (*cases l x*, *auto*, *metis add.commute ereal-max less-add-same-cancel2 max-less-iff-conj real-of-ereal.simps*(*1*))
then have $\exists n>0.\ ereal(u\ n\ x\ /\ n) < F\ K\ e\ x$ **unfolding** *l-def* **using** *liminf-upper-bound* **by** *fastforce*
then obtain n **where** $n>0\ ereal(u\ n\ x\ /\ n) < F\ K\ e\ x$ **by** *auto*
then have $u\ n\ x - n * F\ K\ e\ x < 0$ **by** (*simp add*: *divide-less-eq mult.commute*)
then have $x \notin C\ n$ **unfolding** *C-def B-def* **using** ‹$x \in A$› ‹$n>0$› **by** *auto*
then have $x \notin (\bigcap n.\ C\ n)$ **by** *auto*
}
then have $(\bigcap n.\ C\ n) = \{\}$ **unfolding** *C-def* **by** *auto*
then have $*$: $0 = measure\ M\ (\bigcap n.\ C\ n)$ **by** *auto*
have $(\lambda n.\ measure\ M\ (C\ n)) \longrightarrow 0$
apply (*subst* $*$, *rule finite-Lim-measure-decseq*, *auto*) **unfolding** *C-def B-def decseq-def* **by** *auto*
moreover have $measure\ M\ (C\ n) = (\int x.\ norm(real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ n\ K\ e)\ x)\ \partial M)$ **for** n
proof −
have $*$: $AE\ x\ in\ M.\ 0 \le real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ n\ K\ e)\ x$
apply (*rule real-cond-exp-pos*, *auto*) **unfolding** *I-def* **by** *auto*

have $measure\ M\ (C\ n) = (\int x.\ indicator\ (C\ n)\ x\ \partial M)$
by *auto*
also have ... $= (\int x.\ I\ n\ K\ e\ x\ \partial M)$
apply (*rule integral-cong-AE*, *auto*)
unfolding *C-def I-def indicator-def* **using** ‹$AE\ x\ in\ M.\ x \in A$› **by** *auto*
also have ... $= (\int x.\ real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ n\ K\ e)\ x\ \partial M)$
by (*rule real-cond-exp-int*(*2*)[*symmetric*, *OF I-int*])
also have ... $= (\int x.\ norm(real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ n\ K\ e)\ x)\ \partial M)$
apply (*rule integral-cong-AE*, *auto*) **using** $*$ **by** *auto*
finally show *?thesis* **by** *auto*
qed
ultimately have $*$: $(\lambda n.\ (\int x.\ norm(real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ n\ K\ e)\ x)\ \partial M)) \longrightarrow 0$ **by** *simp*

have $\exists r.\ strict\text{-}mono\ r \wedge (AE\ x\ in\ M.\ (\lambda n.\ real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ (r\ n)\ K\ e)\ x) \longrightarrow 0)$
apply (*rule tendsto-L1-AE-subseq*) **using** $*$ *real-cond-exp-int*[*OF I-int*] **by** *auto*
then obtain r **where** $strict\text{-}mono\ r\ AE\ x\ in\ M.\ (\lambda n.\ real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ (r\ n)\ K\ e)\ x) \longrightarrow 0$
by *auto*
moreover have $AE\ x\ in\ M.\ \forall N \in \{1<..\}.\ limsup\ (\lambda n.\ u\ n\ x\ /\ n) \le F\ K\ e\ x + abs(F\ K\ e\ x) * ereal(real\text{-}cond\text{-}exp\ M\ Invariants\ (I\ N\ K\ e)\ x)$
apply (*rule AE-ball-countable′*) **using** *main*[*OF* - ‹$K>0$› ‹$e>0$›] **by** *auto*

moreover
{
fix *x* **assume** *H*: (λn. *real-cond-exp M Invariants* (*I* (*r n*) *K e*) *x*) $\longrightarrow$ *0*
$\bigwedge N$. $N > 1 \Longrightarrow$ *limsup* (λn. *u n x* / *n*) $\leq$ *F K e x* + *abs*(*F K e x*) $*$ *ereal*(*real-cond-exp M Invariants* (*I N K e*) *x*)
have *1*: *eventually* (λN. *limsup* (λn. *u n x* / *n*) $\leq$ *F K e x* + *abs*(*F K e x*) $*$ *ereal*(*real-cond-exp M Invariants* (*I* (*r N*) *K e*) *x*)) *sequentially*
apply (*rule eventually-mono*[*OF eventually-gt-at-top*[*of 1*] *H*(*2*)])
using ‹*strict-mono r*› *less-le-trans seq-suble* **by** *blast*
have *2*: (λN. *F K e x* + (*abs*(*F K e x*) $*$ *ereal*(*real-cond-exp M Invariants* (*I* (*r N*) *K e*) *x*))) $\longrightarrow$ *ereal*(*F K e x*) + (*abs*(*F K e x*) $*$ *ereal 0*)
by (*intro tendsto-intros*) (*auto simp add*: *H*(*1*))
have *limsup* (λn. *u n x* / *n*) $\leq$ *F K e x*
apply (*rule LIMSEQ-le-const*) **using** *1 2* **by** (*auto simp add*: *eventually-at-top-linorder*)
}
ultimately show *AE x in M*. *limsup* (λn. *u n x* / *n*) $\leq$ *F K e x* **by** *auto*
qed
have *AE x in M*. *limsup* (λn. *u n x* / *n*) $\leq$ *real-of-ereal*(*max* (*l x*) ($-$*ereal K*)) **if** *K*>(*0*::*nat*) **for** *K*
apply (*rule AE-upper-bound-inf-ereal*) **using** *bound2* ‹*K*>*0*› **unfolding** *F-def* **by** *auto*
then have *AE x in M*. $\forall$ *K*$\in${(*0*::*nat*)<..}. *limsup* (λn. *u n x* / *n*) $\leq$ *real-of-ereal*(*max* (*l x*) ($-$*ereal K*))
by (*rule AE-ball-countable′*, *auto*)
moreover have (λn. *u n x* / *n*) $\longrightarrow$ *l x*
if *H*: $\forall$ *K*$\in${(*0*::*nat*)<..}. *limsup* (λn. *u n x* / *n*) $\leq$ *real-of-ereal*(*max* (*l x*) ($-$*ereal K*)) **for** *x*
proof −
have *limsup* (λn. *u n x* / *n*) $\leq$ *l x*
proof (*cases l x* = ∞)
case *False*
then have (λK. *real-of-ereal*(*max* (*l x*) ($-$*ereal K*))) $\longrightarrow$ *l x*
using *ereal-truncation-real-bottom* **by** *auto*
moreover have *eventually* (λK. *limsup* (λn. *u n x* / *n*) $\leq$ *real-of-ereal*(*max* (*l x*) ($-$*ereal K*))) *sequentially*
using *H* **by** (*metis* (*no-types*, *lifting*) *eventually-at-top-linorder eventually-gt-at-top greaterThan-iff*)
ultimately show *limsup* (λn. *u n x* / *n*) $\leq$ *l x* **using** *Lim-bounded2 eventually-sequentially* **by** *auto*
qed (*simp*)
then have *limsup* (λn. *ereal* (*u n x* / *real n*)) = *l x*
using *Liminf-le-Limsup l-def eq-iff sequentially-bot* **by** *blast*
then show (λn. *u n x* / *n*) $\longrightarrow$ *l x*
by (*simp add*: *l-def tendsto-iff-Liminf-eq-Limsup*)
qed
ultimately have *AE x in M*. (λn. *u n x* / *n*) $\longrightarrow$ *l x* **by** *auto*
then have *AE x in M*. (λn. *u n x* / *n*) $\longrightarrow$ *g x* **using** *g*(*2*) **by** *auto*
then show $\exists$ (*g*::′*a*$\Rightarrow$*ereal*). (*g*$\in$*borel-measurable Invariants* $\wedge$ ($\forall$ *x*. *g x* < ∞) $\wedge$

(*AE x in M*. (*λn. u n x / n*) ⟶ *g x*))
  using *g*(*1*) ‹⋀*x. g x* < ∞› **by** *auto*
qed

We deduce it for general subcocycles, by reducing to nonpositive subcocycles by subtracting the Birkhoff sum of u_1 (for which the convergence follows from Birkhoff theorem).

theorem *kingman-theorem-AE-aux2*:
  assumes *subcocycle u*
  shows ∃(*g*::*'a*⇒*ereal*). (*g*∈*borel-measurable Invariants* ∧ (∀ *x. g x* < ∞) ∧ (*AE x in M*. (*λn. u n x / n*) ⟶ *g x*))
proof −
  define *v* **where** *v* = (*λn x. u n x* + *birkhoff-sum* (*λx.* − *u 1 x*) *n x*)
  have *subcocycle v* **unfolding** *v-def*
    apply (*rule subcocycle-add*[*OF assms*], *rule subcocycle-birkhoff*)
    using *assms* **unfolding** *subcocycle-def* **by** *auto*
  have ∃(*gv*::*'a*⇒*ereal*). (*gv*∈*borel-measurable Invariants* ∧ (∀ *x. gv x* < ∞) ∧ (*AE x in M*. (*λn. v n x / n*) ⟶ *gv x*))
    apply (*rule kingman-theorem-AE-aux1*[*OF* ‹*subcocycle v*›]) **unfolding** *v-def* **by** *auto*
  then obtain *gv* **where** *gv*: *gv* ∈ *borel-measurable Invariants AE x in M*. (*λn. v n x / n*) ⟶ (*gv x*::*ereal*) ⋀*x. gv x* < ∞
    by *blast*
  define *g* **where** *g* = (*λx. gv x* + *ereal*(*real-cond-exp M Invariants* (*u 1*) *x*))
  have *g-meas*: *g* ∈ *borel-measurable Invariants* **unfolding** *g-def* **using** *gv*(*1*) **by** *auto*
  have *g-fin*: ⋀*x. g x* < ∞ **unfolding** *g-def* **using** *gv*(*3*) **by** *auto*

  have *AE x in M*. (*λn. birkhoff-sum* (*u 1*) *n x / n*) ⟶ *real-cond-exp M Invariants* (*u 1*) *x*
    apply (*rule birkhoff-theorem-AE-nonergodic*) **using** *assms* **unfolding** *subcocycle-def* **by** *auto*
  moreover
  {
    fix *x* **assume** *H*: (*λn. v n x / n*) ⟶ (*gv x*)
          (*λn. birkhoff-sum* (*u 1*) *n x / n*) ⟶ *real-cond-exp M Invariants* (*u 1*) *x*
    then have (*λn. ereal*(*birkhoff-sum* (*u 1*) *n x / n*)) ⟶ *ereal*(*real-cond-exp M Invariants* (*u 1*) *x*)
      by *auto*
    {
      fix *n*
      have *u n x* = *v n x* + *birkhoff-sum* (*u 1*) *n x*
        unfolding *v-def birkhoff-sum-def* **apply** *auto* **by** (*simp add*: *sum-negf*)
      then have *u n x / n* = *v n x / n* + *birkhoff-sum* (*u 1*) *n x / n* **by** (*simp add*: *add-divide-distrib*)
      then have *ereal*(*u n x / n*) = *ereal*(*v n x / n*) + *ereal*(*birkhoff-sum* (*u 1*) *n x / n*)
        by *auto*

} **note** ∗ = *this*
have (λn. *ereal*(u n x / n)) ⟶ g x **unfolding** ∗ *g-def*
apply (*intro tendsto-intros*) **using** H **by** *auto*
}
ultimately have *AE x in M*. (λn. *ereal*(u n x / n)) ⟶ g x **using** *gv*(2) **by** *auto*
then show *?thesis* **using** *g-meas g-fin* **by** *blast*
qed

For applications, it is convenient to have a limit which is really measurable with respect to the invariant sigma algebra and does not come from a hard to use abstract existence statement. Hence we introduce the following definition for the would-be limit – Kingman's theorem shows that it is indeed a limit.

We introduce the definition for any function, not only subcocycles, but it will only be usable for subcocycles. We introduce an if clause in the definition so that the limit is always measurable, even when u is not a subcocycle and there is no convergence.

definition *subcocycle-lim-ereal*::(*nat* ⇒ ′*a* ⇒ *real*) ⇒ (′*a* ⇒ *ereal*)
where *subcocycle-lim-ereal u* = (
if (∃ (g::′a⇒ereal). (g∈*borel-measurable Invariants* ∧
(∀ x. g x < ∞) ∧ (*AE x in M*. (λn. u n x / n) ⟶ g x)))
then (*SOME* (g::′a⇒ereal). g∈*borel-measurable Invariants* ∧
(∀ x. g x < ∞) ∧ (*AE x in M*. (λn. u n x / n) ⟶ g x))
else (λ-. *0*))

definition *subcocycle-lim*::(*nat* ⇒ ′*a* ⇒ *real*) ⇒ (′*a* ⇒ *real*)
where *subcocycle-lim u* = (λx. *real-of-ereal*(*subcocycle-lim-ereal u x*))

lemma *subcocycle-lim-meas-Inv* [*measurable*]:
subcocycle-lim-ereal u ∈ *borel-measurable Invariants*
subcocycle-lim u ∈ *borel-measurable Invariants*
proof −
show *subcocycle-lim-ereal u* ∈ *borel-measurable Invariants*
proof (*cases* ∃ (g::′a⇒ereal). (g∈*borel-measurable Invariants* ∧ (∀ x. g x < ∞) ∧ (*AE x in M*. (λn. u n x / n) ⟶ g x)))
case *True*
then have *subcocycle-lim-ereal u* = (*SOME* (g::′a⇒ereal). g∈*borel-measurable Invariants* ∧
(∀ x. g x < ∞) ∧ (*AE x in M*. (λn. u n x / n) ⟶ g x))
unfolding *subcocycle-lim-ereal-def* **by** *auto*
then show *?thesis* **using** *someI-ex*[*OF True*] **by** *auto*
next
case *False*
then have *subcocycle-lim-ereal u* = (λ-. *0*) **unfolding** *subcocycle-lim-ereal-def* **by** *auto*
then show *?thesis* **by** *auto*
qed

then show *subcocycle-lim u* ∈ *borel-measurable Invariants* **unfolding** *subcocycle-lim-def* **by** *auto*
qed

lemma *subcocycle-lim-meas* [*measurable*]:
subcocycle-lim-ereal u ∈ *borel-measurable M*
subcocycle-lim u ∈ *borel-measurable M*
using *subcocycle-lim-meas-Inv Invariants-measurable-func* **apply** *blast*
using *subcocycle-lim-meas-Inv Invariants-measurable-func* **by** *blast*

lemma *subcocycle-lim-ereal-not-PInf*:
subcocycle-lim-ereal u x < ∞
proof (*cases* ∃(*g*::'*a*⇒*ereal*). (*g*∈*borel-measurable Invariants* ∧ (∀ *x*. *g x* < ∞) ∧ (*AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *g x*)))
case *True*
then have *subcocycle-lim-ereal u* = (*SOME* (*g*::'*a*⇒*ereal*). *g*∈*borel-measurable Invariants* ∧
(∀ *x*. *g x* < ∞) ∧ (*AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *g x*))
unfolding *subcocycle-lim-ereal-def* **by** *auto*
then show *?thesis* **using** *someI-ex*[*OF True*] **by** *auto*
next
case *False*
then have *subcocycle-lim-ereal u* = (λ-. *0*) **unfolding** *subcocycle-lim-ereal-def* **by** *auto*
then show *?thesis* **by** *auto*
qed

We reformulate the subadditive ergodic theorem of Kingman with this definition. From this point on, the technical definition of `subcocycle_lim_ereal` will never be used, only the following property will be relevant.

theorem *kingman-theorem-AE-nonergodic-ereal*:
assumes *subcocycle u*
shows *AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *subcocycle-lim-ereal u x*
proof −
have ∗: ∃(*g*::'*a*⇒*ereal*). (*g*∈*borel-measurable Invariants* ∧ (∀ *x*. *g x* < ∞) ∧ (*AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *g x*))
using *kingman-theorem-AE-aux2*[*OF assms*] **by** *auto*
then have *subcocycle-lim-ereal u* = (*SOME* (*g*::'*a*⇒*ereal*). *g*∈*borel-measurable Invariants* ∧
(∀ *x*. *g x* < ∞) ∧ (*AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *g x*))
unfolding *subcocycle-lim-ereal-def* **by** *auto*
then show *?thesis* **using** *someI-ex*[*OF* ∗] **by** *auto*
qed

The subcocycle limit behaves well under addition, multiplication by a positive scalar, max, and is simply the conditional expectation with respect to invariants for Birkhoff sums, thanks to Birkhoff theorem.

lemma *subcocycle-lim-ereal-add*:
assumes *subcocycle u subcocycle v*

shows *AE x in M. subcocycle-lim-ereal* (*λn x. u n x* + *v n x*) *x* = *subcocycle-lim-ereal u x* + *subcocycle-lim-ereal v x*
proof −
have *AE x in M.* (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim-ereal* (*λn x. u n x* + *v n x*) *x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF subcocycle-add*[*OF assms*]])
moreover have *AE x in M.* (*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)])
moreover have *AE x in M.* (*λn. v n x / n*) ⟶ *subcocycle-lim-ereal v x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*2*)])
moreover
{
fix *x* **assume** *H*: (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim-ereal* (*λn x. u n x* + *v n x*) *x*
(*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
(*λn. v n x / n*) ⟶ *subcocycle-lim-ereal v x*
have ∗: (*u n x* + *v n x*)/*n* = *ereal* (*u n x / n*) + (*v n x / n*) **for** *n*
by (*simp add*: *add-divide-distrib*)
have (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim-ereal u x* + *subcocycle-lim-ereal v x*
unfolding ∗ **apply** (*intro tendsto-intros H*(*2*) *H*(*3*)) **using** *subcocycle-lim-ereal-not-PInf* **by** *auto*
then have *subcocycle-lim-ereal* (*λn x. u n x* + *v n x*) *x* = *subcocycle-lim-ereal u x* + *subcocycle-lim-ereal v x*
using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)
}
ultimately show *?thesis* **by** *auto*
qed

lemma *subcocycle-lim-ereal-cmult*:
assumes *subcocycle u c*≥(*0*::*real*)
shows *AE x in M. subcocycle-lim-ereal* (*λn x. c* ∗ *u n x*) *x* = *c* ∗ *subcocycle-lim-ereal u x*
proof −
have *AE x in M.* (*λn.* (*c* ∗ *u n x*)/*n*) ⟶ *subcocycle-lim-ereal* (*λn x. c* ∗ *u n x*) *x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF subcocycle-cmult*[*OF assms*]])
moreover have *AE x in M.* (*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)])
moreover
{
fix *x* **assume** *H*: (*λn.* (*c* ∗ *u n x*)/*n*) ⟶ *subcocycle-lim-ereal* (*λn x. c* ∗ *u n x*) *x*
(*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
have (*λn. c* ∗ *ereal* (*u n x / n*)) ⟶ *c* ∗ *subcocycle-lim-ereal u x*
by (*rule tendsto-cmult-ereal*[*OF - H*(*2*)], *auto*)
then have *subcocycle-lim-ereal* (*λn x. c* ∗ *u n x*) *x* = *c* ∗ *subcocycle-lim-ereal u x*
using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)

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

lemma *subcocycle-lim-ereal-max*:
assumes *subcocycle u subcocycle v*
shows *AE x in M. subcocycle-lim-ereal* (*λn x. max* (*u n x*) (*v n x*)) *x*
= *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
proof −
have *AE x in M.* (*λn. max* (*u n x*) (*v n x*) / *n*) ⟶ *subcocycle-lim-ereal* (*λn x. max* (*u n x*) (*v n x*)) *x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF subcocycle-max*[*OF assms*]])
moreover have *AE x in M.* (*λn. u n x* / *n*) ⟶ *subcocycle-lim-ereal u x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)])
moreover have *AE x in M.* (*λn. v n x* / *n*) ⟶ *subcocycle-lim-ereal v x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*2*)])
moreover
{
fix *x* **assume** *H*: (*λn. max* (*u n x*) (*v n x*) / *n*) ⟶ *subcocycle-lim-ereal* (*λn x. max* (*u n x*) (*v n x*)) *x*
(*λn. u n x* / *n*) ⟶ *subcocycle-lim-ereal u x*
(*λn. v n x* / *n*) ⟶ *subcocycle-lim-ereal v x*
have (*λn. max* (*ereal*(*u n x* / *n*)) (*ereal*(*v n x* / *n*)))
⟶ *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
apply (*rule tendsto-max*) **using** *H* **by** *auto*
moreover have *max* (*ereal*(*u n x* / *n*)) (*ereal*(*v n x* / *n*)) = *max* (*u n x*) (*v n x*) / *n* **for** *n*
by (*simp del*: *ereal-max add*:*ereal-max*[*symmetric*] *max-divide-distrib-right*)
ultimately have (*λn. max* (*u n x*) (*v n x*) / *n*)
⟶ *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
by *auto*
then have *subcocycle-lim-ereal* (*λn x. max* (*u n x*) (*v n x*)) *x*
= *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)
}
ultimately show *?thesis* **by** *auto*
qed

lemma *subcocycle-lim-ereal-birkhoff*:
assumes *integrable M u*
shows *AE x in M. subcocycle-lim-ereal* (*birkhoff-sum u*) *x* = *ereal*(*real-cond-exp M Invariants u x*)
proof −
have *AE x in M.* (*λn. birkhoff-sum u n x* / *n*) ⟶ *real-cond-exp M Invariants u x*
by (*rule birkhoff-theorem-AE-nonergodic*[*OF assms*])
moreover have *AE x in M.* (*λn. birkhoff-sum u n x* / *n*) ⟶ *subcocycle-lim-ereal* (*birkhoff-sum u*) *x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF subcocycle-birkhoff*[*OF assms*]])

moreover
{
fix *x* **assume** *H*: (*λn. birkhoff-sum u n x / n*) ⟶ *real-cond-exp M Invariants u x*
(*λn. birkhoff-sum u n x / n*) ⟶ *subcocycle-lim-ereal* (*birkhoff-sum u*) *x*
have (*λn. birkhoff-sum u n x / n*) ⟶ *ereal*(*real-cond-exp M Invariants u x*)
using *H*(*1*) **by** *auto*
then have *subcocycle-lim-ereal* (*birkhoff-sum u*) *x* = *ereal*(*real-cond-exp M Invariants u x*)
using *H*(*2*) **by** (*simp add*: *LIMSEQ-unique*)
}
ultimately show *?thesis* **by** *auto*
qed

9.4 L^1 and a.e. convergence of subcocycles with finite asymptotic average

In this subsection, we show that the almost sure convergence in Kingman theorem also takes place in L^1 if the limit is integrable, i.e., if the asymptotic average of the subcocycle is $> -\infty$. To deduce it from the almost sure convergence, we only need to show that there is no loss of mass, i.e., that the integral of the limit is not strictly larger than the limit of the integrals (thanks to Scheffe criterion). This follows from comparison to Birkhoff sums, for which we know that the average of the limit is the same as the average of the function.

First, we show that the subcocycle limit is bounded by the limit of the Birkhoff sums of u_N, i.e., its conditional expectation. This follows from the fact that u_n is bounded by the Birkhoff sum of u_N (up to negligible boundary terms).

lemma *subcocycle-lim-ereal-atmost-uN-invariants*:
assumes *subcocycle u N*>(*0*::*nat*)
shows *AE x in M*. *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*λx. u N x / N*) *x*
proof −
have *AE x in M*. (*λn. u 1* ((*T*^^*n*) *x*) / *n*) ⟶ *0*
apply (*rule limit-foTn-over-n'*) **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
moreover have *AE x in M*. (*λn. birkhoff-sum* (*λx. u N x/N*) *n x / n*) ⟶ *real-cond-exp M Invariants* (*λx. u N x / N*) *x*
apply (*rule birkhoff-theorem-AE-nonergodic*) **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
moreover have *AE x in M*. (*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
by (*rule kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)])
moreover
{

fix x **assume** H: $(\lambda n.\ u\ 1\ ((T\widehat{\ }\widehat{\ }n)\ x)\ /\ n) \longrightarrow 0$
$(\lambda n.\ birkhoff\text{-}sum\ (\lambda x.\ u\ N\ x/N)\ n\ x\ /\ n) \longrightarrow real\text{-}cond\text{-}exp\ M$ $Invariants\ (\lambda x.\ u\ N\ x\ /\ N)\ x$
$(\lambda n.\ u\ n\ x\ /\ n) \longrightarrow subcocycle\text{-}lim\text{-}ereal\ u\ x$

let $?f = \lambda n.\ birkhoff\text{-}sum\ (\lambda x.\ u\ N\ x\ /\ real\ N)\ (n - 2 * N)\ x\ /\ n$
$+ (\sum i<N.\ (1/n) * |u\ 1\ ((T\widehat{\ }\widehat{\ }i)\ x)|)$
$+ 2 * (\sum i<2*N.\ |u\ 1\ ((T\widehat{\ }\widehat{\ }(n - (2 * N - i)))\ x)|\ /\ n)$
{
fix n **assume** $n \geq 2*N+1$
then have $n > 2 * N$ **by** *simp*
have $u\ n\ x\ /\ n \leq (birkhoff\text{-}sum\ (\lambda x.\ u\ N\ x\ /\ real\ N)\ (n - 2 * N)\ x$
$+ (\sum i<N.\ |u\ 1\ ((T\widehat{\ }\widehat{\ }i)\ x)|)$
$+ 2 * (\sum i<2*N.\ |u\ 1\ ((T\widehat{\ }\widehat{\ }(n - (2 * N - i)))\ x)|))\ /\ n$
using *subcocycle-bounded-by-birkhoffN*[*OF assms*(*1*) ‹$n>2*N$› ‹$N>0$›, *of* x] ‹$n>2*N$› **by** (*simp add*: *divide-right-mono*)
also have ... = $?f\ n$
apply (*subst add-divide-distrib*)+ **by** (*auto simp add*: *sum-divide-distrib*[*symmetric*])
finally have $u\ n\ x\ /\ n \leq ?f\ n$ **by** *simp*
then have $u\ n\ x\ /\ n \leq ereal(?f\ n)$ **by** *simp*
}

have $(\lambda n.\ ?f\ n) \longrightarrow real\text{-}cond\text{-}exp\ M\ Invariants\ (\lambda x.\ u\ N\ x\ /\ N)\ x + (\sum i<N.\ 0 * |u\ 1\ ((T\widehat{\ }\widehat{\ }i)\ x)|) + 2 * (\sum i<2*N.\ 0)$
apply (*intro tendsto-intros*) **using** $H(2)$ *tendsto-norm*[*OF* $H(1)$] **by** *auto*
then have $(\lambda n.\ ereal(?f\ n)) \longrightarrow real\text{-}cond\text{-}exp\ M\ Invariants\ (\lambda x.\ u\ N\ x\ /\ N)\ x$
by *auto*
with *lim-mono*[*OF* ‹$\bigwedge n.\ n \geq 2*N+1 \Longrightarrow u\ n\ x\ /\ n \leq ereal(?f\ n)$› $H(3)$ *this*]
have $subcocycle\text{-}lim\text{-}ereal\ u\ x \leq real\text{-}cond\text{-}exp\ M\ Invariants\ (\lambda x.\ u\ N\ x\ /\ N)\ x$ **by** *simp*
}
ultimately show *?thesis* **by** *auto*
qed

To apply Scheffe criterion, we need to deal with nonnegative functions, or equivalently with nonpositive functions after a change of sign. Hence, as in the proof of the almost sure version of Kingman theorem above, we first give the proof assuming that the subcocycle is nonpositive, and deduce the general statement by adding a suitable Birkhoff sum.

lemma *kingman-theorem-L1-aux*:
assumes *subcocycle u* $subcocycle\text{-}avg\text{-}ereal\ u > -\infty$ $\bigwedge x.\ u\ 1\ x \leq 0$
shows $AE\ x\ in\ M.\ (\lambda n.\ u\ n\ x\ /\ n) \longrightarrow subcocycle\text{-}lim\ u\ x$
integrable M (*subcocycle-lim u*)
$(\lambda n.\ (\int^+ x.\ abs(u\ n\ x\ /\ n - subcocycle\text{-}lim\ u\ x)\ \partial M)) \longrightarrow 0$
proof –
have *int-u* [*measurable*]: $\bigwedge n.\ integrable\ M\ (u\ n)$ **using** *assms*(*1*) *subcocycle-def* **by** *auto*
then have *int-F* [*measurable*]: $\bigwedge n.\ integrable\ M\ (\lambda x.\ -\ u\ n\ x/\ n)$ **by** *auto*

have *F-pos*: $-$ *u n x* / *n* $\geq$ *0* **for** *x n*
proof (*cases n* $>$ *0*)
case *True*
have *u n x* $\leq$ ($\sum$*i*$<$*n*. *u 1* ((*T* ^^ *i*) *x*))
using *subcocycle-bounded-by-birkhoff1*[*OF assms*(*1*) ‹*n*$>$*0*›] **unfolding** *birkhoff-sum-def* **by** *simp*
also have ... $\leq$ *0* **using** *sum-mono*[*OF assms*(*3*)] **by** *auto*
finally have *u n x* $\leq$ *0* **by** *simp*
then have $-$*u n x* $\geq$ *0* **by** *simp*
with *divide-nonneg-nonneg*[*OF this*] **show** $-$ *u n x* / *n* $\geq$ *0* **using** ‹*n*$>$*0*› **by** *auto*
qed (*auto*)

{
fix *x* **assume** $*$: (λ*n*. *u n x* / *n*) $\longrightarrow$ *subcocycle-lim-ereal u x*
have *H*: (λ*n*. $-$ *u n x* / *n*) $\longrightarrow$ $-$ *subcocycle-lim-ereal u x*
using *tendsto-cmult-ereal*[*OF* - $*$, *of* $-$*1*] **by** *auto*
have *liminf* (λ*n*. $-$*u n x* / *n*) = $-$ *subcocycle-lim-ereal u x*
(λ*n*. $-$ *u n x* / *n*) $\longrightarrow$ $-$ *subcocycle-lim-ereal u x*
$-$ *subcocycle-lim-ereal u x* $\geq$ *0*
using *H* **apply** (*simp add*: *tendsto-iff-Liminf-eq-Limsup*, *simp*)
apply (*rule LIMSEQ-le-const*[*OF H*]) **using** *F-pos* **by** *auto*
}
then have *AE-1*: *AE x in M*. *liminf* (λ*n*. $-$*u n x* / *n*) = $-$ *subcocycle-lim-ereal u x*
AE x in M. (λ*n*. $-$ *u n x* / *n*) $\longrightarrow$ $-$ *subcocycle-lim-ereal u x*
AE x in M. $-$ *subcocycle-lim-ereal u x* $\geq$ *0*
using *kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)] **by** *auto*

have ($\int^+$ *x*. $-$*subcocycle-lim-ereal u x* ∂M) = ($\int^+$ *x*. *liminf* (λ*n*. $-$*u n x* / *n*) ∂M)
apply (*rule nn-integral-cong-AE*) **using** *AE-1*(*1*) **by** *auto*
also have ... $\leq$ *liminf* (λ*n*. $\int^+$ *x*. $-$*u n x* / *n* ∂M)
apply (*subst e2ennreal-Liminf*)
apply (*simp-all add*: *e2ennreal-ereal*)
using *F-pos* **by** (*intro nn-integral-liminf*) (*simp add*: *int-F*)
also have ... = $-$ *subcocycle-avg-ereal u*
proof $-$
have (λ*n*. ($\int$ *x*. *u n x* / *n* ∂M)) $\longrightarrow$ *subcocycle-avg-ereal u*
by (*rule subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)])
with *tendsto-cmult-ereal*[*OF* - *this*, *of* $-$*1*]
have (λ*n*. ($\int$ *x*. $-$ *u n x* / *n* ∂M)) $\longrightarrow$ $-$ *subcocycle-avg-ereal u* **by** *simp*
then have $-$ *subcocycle-avg-ereal u* = *liminf* (λ*n*. ($\int$ *x*. $-$ *u n x* / *n* ∂M))
by (*simp add*: *tendsto-iff-Liminf-eq-Limsup*)
moreover have ($\int^+$ *x*. *ennreal* ($-$*u n x* / *n*) ∂M) = *ennreal*($\int$ *x*. $-$ *u n x* / *n* ∂M) **for** *n*
apply (*rule nn-integral-eq-integral*[*OF int-F*]) **using** *F-pos* **by** *auto*
ultimately show *?thesis*

by (*auto simp*: *e2ennreal-Liminf e2ennreal-ereal*)
qed
finally have ($\int^+ x.\ -$*subcocycle-lim-ereal u x* ∂M) $\leq -$ *subcocycle-avg-ereal u* **by** *simp*
also have ... $< \infty$ **using** *assms*(*2*)
by (*cases subcocycle-avg-ereal u*) (*auto simp*: *e2ennreal-ereal e2ennreal-neg*)
finally have $*$: ($\int^+ x.\ -$*subcocycle-lim-ereal u x* ∂M) $< \infty$ **.**
have *AE x in M*. *e2ennreal* ($-$ *subcocycle-lim-ereal u x*) $\neq \infty$
apply (*rule nn-integral-PInf-AE*) **using** $*$ **by** *auto*
then have $**$: *AE x in M*. $-$ *subcocycle-lim-ereal u x* $\neq \infty$
using *AE-1*(*3*) **by** *eventually-elim simp*

{
fix *x* **assume** *H*: $-$ *subcocycle-lim-ereal u x* $\neq \infty$
($\lambda n.$ *u n x* / *n*) $\longrightarrow$ *subcocycle-lim-ereal u x*
$-$ *subcocycle-lim-ereal u x* $\geq$ *0*
then have *1*: *abs*(*subcocycle-lim-ereal u x*) $\neq \infty$ **by** *auto*
then have *2*: ($\lambda n.$ *u n x* / *n*) $\longrightarrow$ *subcocycle-lim u x* **using** *H*(*2*) **unfolding** *subcocycle-lim-def* **by** *auto*
then have *3*: ($\lambda n.$ $-$ (*u n x* / *n*)) $\longrightarrow$ $-$ *subcocycle-lim u x* **using** *tendsto-mult*[*OF* - *2*, *of* λ-. $-$*1*, *of* $-$*1*] **by** *auto*
have *4*: $-$*subcocycle-lim u x* $\geq$ *0* **using** *H*(*3*) **unfolding** *subcocycle-lim-def* **by** *auto*

have *abs*(*subcocycle-lim-ereal u x*) $\neq \infty$
($\lambda n.$ *u n x* / *n*) $\longrightarrow$ *subcocycle-lim u x*
($\lambda n.$ $-$ (*u n x* / *n*)) $\longrightarrow$ $-$ *subcocycle-lim u x*
$-$*subcocycle-lim u x* $\geq$ *0*
using *1 2 3 4* **by** *auto*
}
then have *AE-2*: *AE x in M*. *abs*(*subcocycle-lim-ereal u x*) $\neq \infty$
AE x in M. ($\lambda n.$ *u n x* / *n*) $\longrightarrow$ *subcocycle-lim u x*
AE x in M. ($\lambda n.$ $-$ (*u n x* / *n*)) $\longrightarrow$ $-$ *subcocycle-lim u x*
AE x in M. $-$*subcocycle-lim u x* $\geq$ *0*
using *kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)] $**$ *AE-1*(*3*) **by** *auto*
then show *AE x in M*. ($\lambda n.$ *u n x* / *n*) $\longrightarrow$ *subcocycle-lim u x* **by** *simp*

have ($\int^+ x.$ *abs*(*subcocycle-lim u x*) ∂M) = ($\int^+ x.\ -$*subcocycle-lim-ereal u x* ∂M)
apply (*rule nn-integral-cong-AE*)
using *AE-2* **unfolding** *subcocycle-lim-def abs-real-of-ereal*
apply *eventually-elim*
by (*auto simp*: *e2ennreal-ereal*)
then have *A*: ($\int^+ x.$ *abs*(*subcocycle-lim u x*) ∂M) $< \infty$ **using** $*$ **by** *auto*
show *int-Gr*: *integrable M* (*subcocycle-lim u*)
apply (*rule integrableI-bounded*) **using** *A* **by** *auto*

have *B*: ($\lambda n.$ ($\int^+ x.$ *norm*(($-$ *u n x* /*n*) $-$ ($-$*subcocycle-lim u x*)) ∂M)) $\longrightarrow$ *0*
proof (*rule Scheffe-lemma1*, *auto simp add*: *int-Gr int-u AE-2*(*2*) *AE-2*(*3*))

{
fix *n* **assume** *n*>(*0*::*nat*)
have ∗: *AE x in M*. *subcocycle-lim u x* ≤ *real-cond-exp M Invariants* (λ*x*. *u n x* / *n*) *x*
using *subcocycle-lim-ereal-atmost-uN-invariants*[*OF assms*(*1*) ‹*n*>*0*›] *AE-2*(*1*)
unfolding *subcocycle-lim-def* **by** *auto*
have (∫ *x*. *subcocycle-lim u x* ∂*M*) ≤ (∫ *x*. *real-cond-exp M Invariants* (λ*x*. *u n x* / *n*) *x* ∂*M*)
apply (*rule integral-mono-AE*[*OF int-Gr* - ∗], *rule real-cond-exp-int*(*1*)) **using** *int-u* **by** *auto*
also have ... = (∫ *x*. *u n x* / *n* ∂*M*) **apply** (*rule real-cond-exp-int*(*2*)) **using** *int-u* **by** *auto*
finally have *A*: (∫ *x*. *subcocycle-lim u x* ∂*M*) ≤ (∫ *x*. *u n x* / *n* ∂*M*) **by** *auto*

have (∫ ⁺*x*. *abs*(*u n x*) / *n* ∂*M*) = (∫ ⁺*x*. − *u n x* / *n* ∂*M*)
apply (*rule nn-integral-cong*) **using** *F-pos abs-of-nonneg* **by** (*intro arg-cong*[**where** *f* = *ennreal*]) *fastforce*
also have ... = (∫ *x*. − *u n x* / *n* ∂*M*)
apply (*rule nn-integral-eq-integral*) **using** *F-pos int-F* **by** *auto*
also have ... ≤ (∫ *x*. − *subcocycle-lim u x* ∂*M*) **using** *A* **by** (*auto intro*!: *ennreal-leI*)
also have ... = (∫ ⁺*x*. − *subcocycle-lim u x* ∂*M*)
apply (*rule nn-integral-eq-integral*[*symmetric*]) **using** *int-Gr AE-2*(*4*) **by** *auto*
also have ... = (∫ ⁺*x*. *abs*(*subcocycle-lim u x*) ∂*M*)
apply (*rule nn-integral-cong-AE*) **using** *AE-2*(*4*) **by** *auto*
finally have (∫ ⁺*x*. *abs*(*u n x*) / *n* ∂*M*) ≤ (∫ ⁺*x*. *abs*(*subcocycle-lim u x*) ∂*M*) **by** *simp*
}
with *eventually-mono*[*OF eventually-gt-at-top*[*of 0*] *this*]
have *eventually* (λ*n*. (∫ ⁺*x*. *abs*(*u n x*) / *n* ∂*M*) ≤ (∫ ⁺*x*. *abs*(*subcocycle-lim u x*) ∂*M*)) *sequentially*
by *fastforce*
then show *limsup* (λ*n*. ∫ ⁺ *x*. *abs*(*u n x*) / *n* ∂*M*) ≤ ∫ ⁺ *x*. *abs*(*subcocycle-lim u x*) ∂*M*
using *Limsup-bounded* **by** *fastforce*
qed
moreover have *norm*((− *u n x* /*n*) − (−*subcocycle-lim u x*)) = *abs*(*u n x* / *n* − *subcocycle-lim u x*)
for *n x* **by** *auto*
ultimately show (λ*n*. ∫ ⁺ *x*. *ennreal* |*u n x* / *real n* − *subcocycle-lim u x*| ∂*M*) ⟶ *0*
by *auto*
qed

We can then remove the nonpositivity assumption, by subtracting the Birkhoff sums of u_1 to a general subcocycle u.

theorem *kingman-theorem-nonergodic*:
assumes *subcocycle u subcocycle-avg-ereal u* > −∞

shows *AE x in M*. (*λn. u n x / n*) ⟶ *subcocycle-lim u x*
integrable M (*subcocycle-lim u*)
(*λn.* (∫⁺*x. abs*(*u n x / n* − *subcocycle-lim u x*) ∂*M*)) ⟶ *0*
proof −
have [*measurable*]: *u n* ∈ *borel-measurable M* **for** *n* **using** *assms*(*1*) **unfolding** *subcocycle-def* **by** *auto*
have *int-u* [*measurable*]: *integrable M* (*u 1*) **using** *assms*(*1*) *subcocycle-def* **by** *auto*
define *v* **where** *v* = (*λn x. u n x* + *birkhoff-sum* (*λx.* − *u 1 x*) *n x*)
have [*measurable*]: *v n* ∈ *borel-measurable M* **for** *n* **unfolding** *v-def* **by** *auto*
define *w* **where** *w* = *birkhoff-sum* (*u 1*)
have [*measurable*]: *w n* ∈ *borel-measurable M* **for** *n* **unfolding** *w-def* **by** *auto*
have *subcocycle v* **unfolding** *v-def*
apply (*rule subcocycle-add*[*OF assms*(*1*)], *rule subcocycle-birkhoff*)
using *assms* **unfolding** *subcocycle-def* **by** *auto*
have *subcocycle w* **unfolding** *w-def* **by** (*rule subcocycle-birkhoff*[*OF int-u*])
have *uvw*: *u n x* = *v n x* + *w n x* **for** *n x*
unfolding *v-def w-def birkhoff-sum-def* **by** (*auto simp add*: *sum-negf*)
then have *subcocycle-avg-ereal* (*λn x. u n x*) = *subcocycle-avg-ereal v* + *subcocycle-avg-ereal w*
using *subcocycle-avg-ereal-add*[*OF* ‹*subcocycle v*› ‹*subcocycle w*›] **by** *auto*
then have *subcocycle-avg-ereal u* = *subcocycle-avg-ereal v* + *subcocycle-avg-ereal w*
by *auto*
then have *subcocycle-avg-ereal v* > −∞
unfolding *w-def* **using** *subcocycle-avg-ereal-birkhoff*[*OF int-u*] *assms*(*2*) **by** *auto*
have *subcocycle-avg-ereal w* > − ∞
unfolding *w-def* **using** *subcocycle-avg-birkhoff*[*OF int-u*] **by** *auto*

have ⋀*x. v 1 x* ≤ *0* **unfolding** *v-def* **by** *auto*
have *v*: *AE x in M*. (*λn. v n x / n*) ⟶ *subcocycle-lim v x*
integrable M (*subcocycle-lim v*)
(*λn.* (∫⁺*x. abs*(*v n x / n* − *subcocycle-lim v x*) ∂*M*)) ⟶ *0*
using *kingman-theorem-L1-aux*[*OF* ‹*subcocycle v*› ‹*subcocycle-avg-ereal v* > −∞› ‹⋀*x. v 1 x* ≤ *0*›] **by** *auto*
have *w*: *AE x in M*. (*λn. w n x / n*) ⟶ *subcocycle-lim w x*
integrable M (*subcocycle-lim w*)
(*λn.* (∫⁺*x. abs*(*w n x / n* − *subcocycle-lim w x*) ∂*M*)) ⟶ *0*
proof −
show *AE x in M*. (*λn. w n x / n*) ⟶ *subcocycle-lim w x*
unfolding *w-def subcocycle-lim-def* **using** *subcocycle-lim-ereal-birkhoff*[*OF int-u*]
birkhoff-theorem-AE-nonergodic[*OF int-u*] **by** *auto*
show *integrable M* (*subcocycle-lim w*)
apply (*subst integrable-cong-AE*[**where** *?g* = *λx. real-cond-exp M Invariants* (*u 1*) *x*])
unfolding *w-def subcocycle-lim-def*
using *subcocycle-lim-ereal-birkhoff*[*OF int-u*] *real-cond-exp-int*(*1*)[*OF int-u*]

by *auto*
  have ($\int^+ x$. *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂*M*)
    = ($\int^+ x$. *abs*(*birkhoff-sum* (*u 1*) *n x* / *n* − *real-cond-exp M Invariants* (*u 1*) *x*) ∂*M*) **for** *n*
   apply (*rule nn-integral-cong-AE*)
    unfolding *w-def subcocycle-lim-def* **using** *subcocycle-lim-ereal-birkhoff*[*OF int-u*] **by** *auto*
  then show (λ*n*. ($\int^+ x$. *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂*M*)) ⟶ *0*
   using *birkhoff-theorem-L1-nonergodic*[*OF int-u*] **by** *auto*
 qed

 {
  fix *x* **assume** *H*: (λ*n*. *v n x* / *n*) ⟶ *subcocycle-lim v x*
        (λ*n*. *w n x* / *n*) ⟶ *subcocycle-lim w x*
        (λ*n*. *u n x* / *n*) ⟶ *subcocycle-lim-ereal u x*
  then have (λ*n*. *v n x* / *n* + *w n x* / *n*) ⟶ *subcocycle-lim v x* + *subcocycle-lim w x*
   using *tendsto-add*[*OF H*(*1*) *H*(*2*)] **by** *simp*
  then have ∗: (λ*n*. *ereal*(*u n x* / *n*)) ⟶ *ereal*(*subcocycle-lim v x* + *subcocycle-lim w x*)
   unfolding *uvw* **by** (*simp add*: *add-divide-distrib*)
  then have *subcocycle-lim-ereal u x* = *ereal*(*subcocycle-lim v x* + *subcocycle-lim w x*)
   using *H*(*3*) *LIMSEQ-unique* **by** *blast*
  then have ∗∗: *subcocycle-lim u x* = *subcocycle-lim v x* + *subcocycle-lim w x*
   using *subcocycle-lim-def* **by** *auto*
  have *u n x* / *n* − *subcocycle-lim u x* = *v n x* / *n* − *subcocycle-lim v x* + *w n x* / *n* − *subcocycle-lim w x* **for** *n*
   apply (*subst* ∗∗, *subst uvw*) **using** *add-divide-distrib add.commute* **by** *auto*
  then have (λ*n*. *u n x* / *n*) ⟶ *subcocycle-lim u x*
    ∧ *subcocycle-lim u x* = *subcocycle-lim v x* + *subcocycle-lim w x*
    ∧ (∀*n*. *u n x* / *n* − *subcocycle-lim u x* = *v n x* / *n* − *subcocycle-lim v x* + *w n x* / *n* − *subcocycle-lim w x*)
   using ∗ ∗∗ **by** *auto*
 }
 then have *AE*: *AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *subcocycle-lim u x*
    AE x in M. *subcocycle-lim u x* = *subcocycle-lim v x* + *subcocycle-lim w x*
    AE x in M. ∀*n*. *u n x* / *n* − *subcocycle-lim u x* = *v n x* / *n* − *subcocycle-lim v x* + *w n x* / *n* − *subcocycle-lim w x*
  using *v*(*1*) *w*(*1*) *kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)] **by** *auto*
 then show *AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *subcocycle-lim u x* **by** *simp*
 show *integrable M* (*subcocycle-lim u*)
  apply (*subst integrable-cong-AE*[**where** *?g* = λ*x*. *subcocycle-lim v x* + *subcocycle-lim w x*])
  by (*auto simp add*: *AE*(*2*) *v*(*2*) *w*(*2*))

 show (λ*n*. ($\int^+ x$. *abs*(*u n x* / *n* − *subcocycle-lim u x*) ∂*M*)) ⟶ *0*
 proof (*rule tendsto-sandwich*[**where** *?f* = λ-. *0*

and *?h* = $\lambda n.$ ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M)], *auto*)
{
fix *n*
have ($\int^+ x.$ *abs*(*u n x* / *n* − *subcocycle-lim u x*) ∂M)
= ($\int^+ x.$ *abs*((*v n x* / *n* − *subcocycle-lim v x*) + (*w n x* / *n* − *subcocycle-lim w x*)) ∂M)
apply (*rule nn-integral-cong-AE*) **using** *AE*(*3*) **by** *auto*
also have ... ≤ ($\int^+ x.$ *ennreal*(*abs*(*v n x* / *n* − *subcocycle-lim v x*)) + *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M)
by (*rule nn-integral-mono*, *auto simp add*: *ennreal-plus*[*symmetric*] *simp del*: *ennreal-plus*)
also have ... = ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M)
by (*rule nn-integral-add*, *auto*, *measurable*)
finally have ($\int^+ x.$ *abs*(*u n x* / *n* − *subcocycle-lim u x*) ∂M)
≤ ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M)
using *tendsto-sandwich* **by** *simp*
}
then show *eventually* ($\lambda n.$ ($\int^+ x.$ *abs*(*u n x* / *n* − *subcocycle-lim u x*) ∂M)
≤ ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M)) *sequentially*
by *auto*

have ($\lambda n.$ ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M))
$\longrightarrow$ *0* + *0*
by (*rule tendsto-add*[*OF v*(*3*) *w*(*3*)])
then show ($\lambda n.$ ($\int^+ x.$ *abs*(*v n x* / *n* − *subcocycle-lim v x*) ∂M) + ($\int^+ x.$ *abs*(*w n x* / *n* − *subcocycle-lim w x*) ∂M))
$\longrightarrow$ *0*
by *simp*
qed
qed

From the almost sure convergence, we can prove the basic properties of the (real) subcocycle limit: relationship to the asymptotic average, behavior under sum, multiplication, max, behavior for Birkhoff sums.

lemma *subcocycle-lim-avg*:
assumes *subcocycle u subcocycle-avg-ereal u* > $-\infty$
shows ($\int x.$ *subcocycle-lim u x* ∂M) = *subcocycle-avg u*
proof −
have *H*: ($\lambda n.$ ($\int^+ x.$ *norm*(*u n x* / *n* − *subcocycle-lim u x*) ∂M)) $\longrightarrow$ *0*
integrable M (*subcocycle-lim u*)
using *kingman-theorem-nonergodic*[*OF assms*] **by** *auto*
have ($\lambda n.$ ($\int x.$ *u n x* / *n* ∂M)) $\longrightarrow$ ($\int x.$ *subcocycle-lim u x* ∂M)
apply (*rule tendsto-L1-int*[*OF - H*(*2*) *H*(*1*)]) **using** *subcocycle-integrable*[*OF assms*(*1*)] **by** *auto*

then have (*λn.* (∫ *x. u n x / n ∂M*)) ⟶ *ereal* (∫ *x. subcocycle-lim u x ∂M*)
by *auto*
moreover have (*λn.* (∫ *x. u n x / n ∂M*)) ⟶ *ereal* (*subcocycle-avg u*)
using *subcocycle-int-tendsto-avg*[*OF assms*] **by** *auto*
ultimately show *?thesis* **using** *LIMSEQ-unique* **by** *blast*
qed

lemma *subcocycle-lim-real-ereal*:
assumes *subcocycle u subcocycle-avg-ereal u* > −∞
shows *AE x in M. subcocycle-lim-ereal u x* = *ereal*(*subcocycle-lim u x*)
proof −
{
fix *x* **assume** *H*: (*λn. u n x / n*) ⟶ *subcocycle-lim-ereal u x*
(*λn. u n x / n*) ⟶ *subcocycle-lim u x*
then have (*λn. u n x / n*) ⟶ *ereal*(*subcocycle-lim u x*) **by** *auto*
then have *subcocycle-lim-ereal u x* = *ereal*(*subcocycle-lim u x*)
using *H*(*1*) *LIMSEQ-unique* **by** *blast*
}
then show *?thesis*
using *kingman-theorem-AE-nonergodic-ereal*[*OF assms*(*1*)] *kingman-theorem-nonergodic*(*1*)[*OF assms*] **by** *auto*
qed

lemma *subcocycle-lim-add*:
assumes *subcocycle u subcocycle v subcocycle-avg-ereal u* > −∞ *subcocycle-avg-ereal v* > −∞
shows *subcocycle-avg-ereal* (*λn x. u n x* + *v n x*) > − ∞
AE x in M. subcocycle-lim (*λn x. u n x* + *v n x*) *x* = *subcocycle-lim u x* + *subcocycle-lim v x*
proof −
show ∗: *subcocycle-avg-ereal* (*λn x. u n x* + *v n x*) > − ∞
using *subcocycle-avg-add*[*OF assms*(*1*) *assms*(*2*)] *assms*(*3*) *assms*(*4*) **by** *auto*
have *AE x in M.* (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim* (*λn x. u n x* + *v n x*) *x*
by (*rule kingman-theorem-nonergodic*(*1*)[*OF subcocycle-add*[*OF assms*(*1*) *assms*(*2*)] ∗])
moreover have *AE x in M.* (*λn. u n x / n*) ⟶ *subcocycle-lim u x*
by (*rule kingman-theorem-nonergodic*[*OF assms*(*1*) *assms*(*3*)])
moreover have *AE x in M.* (*λn. v n x / n*) ⟶ *subcocycle-lim v x*
by (*rule kingman-theorem-nonergodic*[*OF assms*(*2*) *assms*(*4*)])
moreover
{
fix *x* **assume** *H*: (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim* (*λn x. u n x* + *v n x*) *x*
(*λn. u n x / n*) ⟶ *subcocycle-lim u x*
(*λn. v n x / n*) ⟶ *subcocycle-lim v x*
have ∗: (*u n x* + *v n x*)/*n* = (*u n x / n*) + (*v n x / n*) **for** *n*
by (*simp add*: *add-divide-distrib*)
have (*λn.* (*u n x* + *v n x*)/*n*) ⟶ *subcocycle-lim u x* + *subcocycle-lim v x*

unfolding ∗ **by** (*intro tendsto-intros H*)
then have *subcocycle-lim* (λ*n x. u n x* + *v n x*) *x* = *subcocycle-lim u x* + *subcocycle-lim v x*
using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)
}
ultimately show *AE x in M*. *subcocycle-lim* (λ*n x. u n x* + *v n x*) *x*
= *subcocycle-lim u x* + *subcocycle-lim v x*
by *auto*
qed

lemma *subcocycle-lim-cmult*:
assumes *subcocycle u subcocycle-avg-ereal u* > −∞ *c*≥(*0*::*real*)
shows *subcocycle-avg-ereal* (λ*n x. c* ∗ *u n x*) > − ∞
AE x in M. *subcocycle-lim* (λ*n x. c* ∗ *u n x*) *x* = *c* ∗ *subcocycle-lim u x*
proof −
show ∗: *subcocycle-avg-ereal* (λ*n x. c* ∗ *u n x*) > − ∞
using *subcocycle-avg-cmult*[*OF assms*(*1*) *assms*(*3*)] *assms*(*2*) *assms*(*3*) **by** *auto*

have *AE x in M*. (λ*n*. (*c* ∗ *u n x*)/*n*) ⟶ *subcocycle-lim* (λ*n x. c* ∗ *u n x*) *x*
by (*rule kingman-theorem-nonergodic*(*1*)[*OF subcocycle-cmult*[*OF assms*(*1*) *assms*(*3*)] ∗])
moreover have *AE x in M*. (λ*n. u n x* / *n*) ⟶ *subcocycle-lim u x*
by (*rule kingman-theorem-nonergodic*(*1*)[*OF assms*(*1*) *assms*(*2*)])
moreover
{
fix *x* **assume** *H*: (λ*n*. (*c* ∗ *u n x*)/*n*) ⟶ *subcocycle-lim* (λ*n x. c* ∗ *u n x*) *x*
(λ*n. u n x* / *n*) ⟶ *subcocycle-lim u x*
have (λ*n. c* ∗ (*u n x* / *n*)) ⟶ *c* ∗ *subcocycle-lim u x*
by (*rule tendsto-mult*[*OF - H*(*2*)], *auto*)
then have *subcocycle-lim* (λ*n x. c* ∗ *u n x*) *x* = *c* ∗ *subcocycle-lim u x*
using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)
}
ultimately show *AE x in M*. *subcocycle-lim* (λ*n x. c* ∗ *u n x*) *x* = *c* ∗ *subcocycle-lim u x* **by** *auto*
qed

lemma *subcocycle-lim-max*:
assumes *subcocycle u subcocycle v subcocycle-avg-ereal u* > −∞ *subcocycle-avg-ereal v* > −∞
shows *subcocycle-avg-ereal* (λ*n x. max* (*u n x*) (*v n x*)) > − ∞
AE x in M. *subcocycle-lim* (λ*n x. max* (*u n x*) (*v n x*)) *x* = *max* (*subcocycle-lim u x*) (*subcocycle-lim v x*)
proof −
show ∗: *subcocycle-avg-ereal* (λ*n x. max* (*u n x*) (*v n x*)) > − ∞
using *subcocycle-avg-max*(*1*)[*OF assms*(*1*) *assms*(*2*)] *assms*(*3*) *assms*(*4*) **by** *auto*
have *AE x in M*. (λ*n. max* (*u n x*) (*v n x*) / *n*) ⟶ *subcocycle-lim* (λ*n x. max* (*u n x*) (*v n x*)) *x*
by (*rule kingman-theorem-nonergodic*[*OF subcocycle-max*[*OF assms*(*1*) *assms*(*2*)]

∗])
  moreover have *AE x in M*. (*λn*. *u n x* / *n*) ⟶ *subcocycle-lim u x*
    by (*rule kingman-theorem-nonergodic*[*OF assms*(*1*) *assms*(*3*)])
  moreover have *AE x in M*. (*λn*. *v n x* / *n*) ⟶ *subcocycle-lim v x*
    by (*rule kingman-theorem-nonergodic*[*OF assms*(*2*) *assms*(*4*)])
  moreover
  {
    fix *x* **assume** *H*: (*λn*. *max* (*u n x*) (*v n x*) / *n*) ⟶ *subcocycle-lim* (*λn x*. *max* (*u n x*) (*v n x*)) *x*
                  (*λn*. *u n x* / *n*) ⟶ *subcocycle-lim u x*
                  (*λn*. *v n x* / *n*) ⟶ *subcocycle-lim v x*
     have (*λn*. *max* (*u n x* / *n*) (*v n x* / *n*)) ⟶ *max* (*subcocycle-lim u x*) (*subcocycle-lim v x*)
      apply (*rule tendsto-max*) **using** *H* **by** *auto*
    moreover have *max* (*u n x* / *n*) (*v n x* / *n*) = *max* (*u n x*) (*v n x*) / *n* **for** *n*
      by (*simp add*: *max-divide-distrib-right*)
    ultimately have (*λn*. *max* (*u n x*) (*v n x*) / *n*) ⟶ *max* (*subcocycle-lim u x*) (*subcocycle-lim v x*)
      by *auto*
    then have *subcocycle-lim* (*λn x*. *max* (*u n x*) (*v n x*)) *x* = *max* (*subcocycle-lim u x*) (*subcocycle-lim v x*)
      using *H*(*1*) **by** (*simp add*: *LIMSEQ-unique*)
  }
  ultimately show *AE x in M*. *subcocycle-lim* (*λn x*. *max* (*u n x*) (*v n x*)) *x*
                  = *max* (*subcocycle-lim u x*) (*subcocycle-lim v x*) **by** *auto*
qed

lemma *subcocycle-lim-birkhoff*:
  assumes *integrable M u*
  shows *subcocycle-avg-ereal* (*birkhoff-sum u*) > −∞
        AE x in M. *subcocycle-lim* (*birkhoff-sum u*) *x* = *real-cond-exp M Invariants u x*
proof −
  show ∗: *subcocycle-avg-ereal* (*birkhoff-sum u*) > −∞
    using *subcocycle-avg-birkhoff*[*OF assms*] **by** *auto*
  have *AE x in M*. (*λn*. *birkhoff-sum u n x* / *n*) ⟶ *real-cond-exp M Invariants u x*
    by (*rule birkhoff-theorem-AE-nonergodic*[*OF assms*])
  moreover have *AE x in M*. (*λn*. *birkhoff-sum u n x* / *n*) ⟶ *subcocycle-lim* (*birkhoff-sum u*) *x*
    by (*rule kingman-theorem-nonergodic*(*1*)[*OF subcocycle-birkhoff*[*OF assms*] ∗])
  moreover
  {
   fix *x* **assume** *H*: (*λn*. *birkhoff-sum u n x* / *n*) ⟶ *real-cond-exp M Invariants u x*
                  (*λn*. *birkhoff-sum u n x* / *n*) ⟶ *subcocycle-lim* (*birkhoff-sum u*) *x*
    then have *subcocycle-lim* (*birkhoff-sum u*) *x* = *real-cond-exp M Invariants u x*
      using *H*(*2*) **by** (*simp add*: *LIMSEQ-unique*)

}
ultimately show *AE x in M. subcocycle-lim (birkhoff-sum u) x = real-cond-exp M Invariants u x* **by** *auto*
qed

9.5 Conditional expectations of subcocycles

In this subsection, we show that the conditional expectations of a subcocycle (with respect to the invariant subalgebra) also converge, with the same limit as the cocycle.

Note that the conditional expectation of a subcocycle u is still a subcocycle, with the same average at each step so with the same asymptotic average. Kingman theorem can be applied to it, and what we have to show is that the limit of this subcocycle is the same as the limit of the original subcocycle.

When the asymptotic average is $> -\infty$, both limits have the same integral, and moreover the domination of the subcocycle by the Birkhoff sums of u_n for fixed n (which converge to the conditional expectation of u_n) implies that one limit is smaller than the other. Hence, they coincide almost everywhere.

The case when the asymptotic average is $-\infty$ is deduced from the previous one by truncation.

First, we prove the result when the asymptotic average with finite.

theorem *kingman-theorem-nonergodic-invariant*:
assumes *subcocycle u subcocycle-avg-ereal u* $> -\infty$
shows *AE x in M.* (λn. *real-cond-exp M Invariants* (*u n*) *x* / *n*) $\longrightarrow$ *subcocycle-lim u x*
(λn. ($\int^+ x$. *abs*(*real-cond-exp M Invariants* (*u n*) *x* / *n* − *subcocycle-lim u x*) ∂M)) $\longrightarrow$ *0*
proof −
have *int* [*simp*]: *integrable M* (*u n*) **for** *n* **using** *subcocycle-integrable*[*OF assms*(*1*)] **by** *auto*
then have *int2*: *integrable M* (*real-cond-exp M Invariants* (*u n*)) **for** *n* **using** *real-cond-exp-int* **by** *auto*
{
fix *n m*
have *u* (*n+m*) *x* ≤ *u n x* + *u m* ((*T*^^*n*) *x*) **for** *x*
using *subcocycle-ineq*[*OF assms*(*1*)] **by** *auto*
have *AE x in M. real-cond-exp M Invariants* (*u* (*n+m*)) *x* ≤ *real-cond-exp M Invariants* (λx. *u n x* + *u m* ((*T*^^*n*) *x*)) *x*
apply (*rule real-cond-exp-mono*)
using *subcocycle-ineq*[*OF assms*(*1*)] **apply** *auto*
by (*rule Bochner-Integration.integrable-add*, *auto simp add*: *Tn-integral-preserving*)
moreover have *AE x in M. real-cond-exp M Invariants* (λx. *u n x* + *u m* ((*T*^^*n*) *x*)) *x*
= *real-cond-exp M Invariants* (*u n*) *x* + *real-cond-exp M Invariants* (λx. *u m* ((*T*^^*n*) *x*)) *x*
by (*rule real-cond-exp-add*, *auto simp add*: *Tn-integral-preserving*)

moreover have *AE x in M. real-cond-exp M Invariants* (*u m* ∘ ((*T*^^*n*))) *x* = *real-cond-exp M Invariants* (*u m*) *x*
by (*rule Invariants-of-foTn*, *simp*)
moreover have *AE x in M. real-cond-exp M Invariants* (*u m*) *x* = *real-cond-exp M Invariants* (*u m*) ((*T*^^*n*) *x*)
using *Invariants-func-is-invariant-n*[*symmetric*, *of real-cond-exp M Invariants* (*u m*)] **by** *auto*
ultimately have *AE x in M. real-cond-exp M Invariants* (*u* (*n*+*m*)) *x*
≤ *real-cond-exp M Invariants* (*u n*) *x* + *real-cond-exp M Invariants* (*u m*) ((*T*^^*n*) *x*)
unfolding *o-def* **by** *auto*
}
with *subcocycle-AE*[*OF this int2*]
obtain *w* **where** *w*: *subcocycle w AE x in M.* ∀ *n. w n x* = *real-cond-exp M Invariants* (*u n*) *x*
by *blast*
have [*measurable*]: *integrable M* (*w n*) **for** *n* **using** *subcocycle-integrable*[*OF w*(*1*)] **by** *simp*

{
fix *n*::*nat*
have (∫ *x. w n x* / *n* ∂*M*) = (∫ *x. real-cond-exp M Invariants* (*u n*) *x* / *n* ∂*M*)
using *w*(*2*) **by** (*intro integral-cong-AE*) (*auto simp*: *eventually-mono*)
also have ... = (∫ *x. real-cond-exp M Invariants* (*u n*) *x* ∂*M*) / *n*
by (*rule integral-divide-zero*)
also have ... = (∫ *x. u n x* ∂*M*) / *n*
by (*simp add*: *divide-simps real-cond-exp-int*(*2*)[*OF int*[*of n*]])
also have ... = (∫ *x. u n x* / *n* ∂*M*)
by (*rule integral-divide-zero*[*symmetric*])
finally have *ereal* (∫ *x. w n x* / *n* ∂*M*) = *ereal* (∫ *x. u n x* / *n* ∂*M*) **by** *simp*
} note ∗ = *this*
have (λ*n.* (∫ *x. u n x* / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal w*
apply (*rule Lim-transform-eventually*[*OF subcocycle-int-tendsto-avg-ereal*[*OF w*(*1*)]])
using ∗ **by** *auto*
then have *subcocycle-avg-ereal u* = *subcocycle-avg-ereal w*
using *subcocycle-int-tendsto-avg-ereal*[*OF assms*(*1*)] *LIMSEQ-unique* **by** *auto*
then have *subcocycle-avg-ereal w* > −∞ **using** *assms*(*2*) **by** *simp*
have *subcocycle-avg u* = *subcocycle-avg w*
using ‹*subcocycle-avg-ereal u* = *subcocycle-avg-ereal w*› **unfolding** *subcocycle-avg-def* **by** *simp*

have *AE x in M. N* > *0* ⟶ *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (λ*x. u N x* / *N*) *x* **for** *N*
by (*cases N* = *0*, *auto simp add*: *subcocycle-lim-ereal-atmost-uN-invariants*[*OF assms*(*1*)])
then have *AE x in M.* ∀ *N. N* > *0* ⟶ *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (λ*x. u N x* / *N*) *x*
by (*simp add*: *AE-all-countable*)

moreover have *AE x in M. subcocycle-lim-ereal u x = ereal(subcocycle-lim u x)*
by (*rule subcocycle-lim-real-ereal*[*OF assms*])
moreover have *AE x in M.* ($\lambda N.$ *u N x / N*) $\longrightarrow$ *subcocycle-lim u x*
using *kingman-theorem-nonergodic*[*OF assms*] **by** *simp*
moreover have *AE x in M.* ($\lambda N.$ *w N x / N*) $\longrightarrow$ *subcocycle-lim w x*
using *kingman-theorem-nonergodic*[*OF w*(*1*) ‹*subcocycle-avg-ereal w* $> -\infty$›]
by *simp*
moreover have *AE x in M.* $\forall$ *n. w n x = real-cond-exp M Invariants* (*u n*) *x*
using *w*(*2*) **by** *simp*
moreover have *AE x in M.* $\forall$ *n. real-cond-exp M Invariants* (*u n*) *x / n =*
real-cond-exp M Invariants ($\lambda x.$ *u n x / n*) *x*
apply (*subst AE-all-countable*, *intro allI*) **using** *AE-symmetric*[*OF real-cond-exp-cdiv*[*OF*
int]] **by** *auto*
moreover
{
fix *x* **assume** *x*: $\forall$ *N. N* $>$ *0* $\longrightarrow$ *subcocycle-lim-ereal u x* $\leq$ *real-cond-exp M*
Invariants ($\lambda x.$ *u N x / N*) *x*
subcocycle-lim-ereal u x = ereal(*subcocycle-lim u x*)
($\lambda N.$ *u N x / N*) $\longrightarrow$ *subcocycle-lim u x*
($\lambda N.$ *w N x / N*) $\longrightarrow$ *subcocycle-lim w x*
$\forall$ *n. w n x = real-cond-exp M Invariants* (*u n*) *x*
$\forall$ *n. real-cond-exp M Invariants* (*u n*) *x / n = real-cond-exp M*
Invariants ($\lambda x.$ *u n x / n*) *x*
{
fix *N*::*nat* **assume** *N*$\geq$*1*
have *subcocycle-lim u x* $\leq$ *real-cond-exp M Invariants* ($\lambda x.$ *u N x / N*) *x*
using *x*(*1*) *x*(*2*) ‹*N*$\geq$*1*› **by** *auto*
also have ... = *real-cond-exp M Invariants* (*u N*) *x / N*
using *x*(*6*) **by** *simp*
also have ... = *w N x / N*
using *x*(*5*) **by** *simp*
finally have *subcocycle-lim u x* $\leq$ *w N x / N*
by *simp*
} note $*$ = *this*
have *subcocycle-lim u x* $\leq$ *subcocycle-lim w x*
apply (*rule LIMSEQ-le-const*[*OF x*(*4*)]) **using** $*$ **by** *auto*
}
ultimately have $*$: *AE x in M. subcocycle-lim u x* $\leq$ *subcocycle-lim w x*
by *auto*
have $**$: ($\int$ *x. subcocycle-lim u x* ∂M) = ($\int$ *x. subcocycle-lim w x* ∂M)
using *subcocycle-lim-avg*[*OF assms*] *subcocycle-lim-avg*[*OF w*(*1*) ‹*subcocycle-avg-ereal*
w $> -\infty$›]
‹*subcocycle-avg u = subcocycle-avg w*› **by** *simp*
have *AE-eq*: *AE x in M. subcocycle-lim u x = subcocycle-lim w x*
by (*rule integral-ineq-eq-0-then-AE*[*OF* $*$ *kingman-theorem-nonergodic*(*2*)[*OF*
assms]
kingman-theorem-nonergodic(*2*)[*OF w*(*1*) ‹*subcocycle-avg-ereal w* $> -\infty$›]
$**$])
moreover have *AE x in M.* ($\lambda n.$ *w n x / n*) $\longrightarrow$ *subcocycle-lim w x*

by (*rule kingman-theorem-nonergodic*(*1*)[*OF w*(*1*) ‹*subcocycle-avg-ereal w* > −∞›])
moreover have *AE x in M*. ∀ *n*. *w n x* = *real-cond-exp M Invariants* (*u n*) *x*
using *w*(*2*) **by** *auto*
moreover
{
fix *x* **assume** *H*: *subcocycle-lim u x* = *subcocycle-lim w x*
(λ*n*. *w n x* / *n*) ⟶ *subcocycle-lim w x*
∀ *n*. *w n x* = *real-cond-exp M Invariants* (*u n*) *x*
then have (λ*n*. *real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim u x*
by *auto*
}
ultimately show *AE x in M*. (λ*n*. *real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim u x*
by *auto*

{
fix *n*::*nat*
have *AE x in M*. *subcocycle-lim u x* = *subcocycle-lim w x*
using *AE-eq* **by** *simp*
moreover have *AE x in M*. *w n x* = *real-cond-exp M Invariants* (*u n*) *x*
using *w*(*2*) **by** *auto*
moreover
{
fix *x* **assume** *H*: *subcocycle-lim u x* = *subcocycle-lim w x*
w n x = *real-cond-exp M Invariants* (*u n*) *x*
then have *ennreal* |*real-cond-exp M Invariants* (*u n*) *x* / *real n* − *subcocycle-lim u x*|
= *ennreal* |*w n x* / *real n* − *subcocycle-lim w x*|
by *auto*
}
ultimately have *AE x in M*. *ennreal* |*real-cond-exp M Invariants* (*u n*) *x* / *real n* − *subcocycle-lim u x*|
= *ennreal* |*w n x* / *real n* − *subcocycle-lim w x*|
by *auto*
then have (∫⁺ *x*. *ennreal* |*real-cond-exp M Invariants* (*u n*) *x* / *real n* − *subcocycle-lim u x*| ∂*M*)
= (∫⁺ *x*. *ennreal* |*w n x* / *real n* − *subcocycle-lim w x*| ∂*M*)
by (*rule nn-integral-cong-AE*)
}
moreover have (λ*n*. (∫⁺ *x*. |*w n x* / *real n* − *subcocycle-lim w x*| ∂*M*)) ⟶ *0*
by (*rule kingman-theorem-nonergodic*(*3*)[*OF w*(*1*) ‹*subcocycle-avg-ereal w* > −∞›])
ultimately show (λ*n*. (∫⁺ *x*. |*real-cond-exp M Invariants* (*u n*) *x* / *real n* − *subcocycle-lim u x*| ∂*M*)) ⟶ *0*
by *auto*
qed

Then, we extend it by truncation to the general case, i.e., to the asymptotic limit in extended reals.

theorem *kingman-theorem-AE-nonergodic-invariant-ereal*:
assumes *subcocycle u*
shows *AE x in M*. (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim-ereal u x*
proof −
have [*simp*]: *subcocycle u* **using** *assms* **by** *simp*
have *int* [*simp*]: *integrable M* (*u n*) **for** *n* **using** *subcocycle-integrable*[*OF assms*(*1*)] **by** *auto*

have *limsup-ineq-K*: *AE x in M*.
limsup (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *max* (*subcocycle-lim-ereal u x*) (−*real K*) **for** *K*::*nat*
proof −
define *v* **where** *v* = (*λ* (*n*::*nat*) (*x*::*'a*). (−*n* ∗ *real K*))
have [*simp*]: *subcocycle v*
unfolding *v-def subcocycle-def* **by** (*auto simp add*: *algebra-simps*)
have *ereal* (∫ *x*. *v n x* / *n* ∂*M*) = *ereal*(− *real K* ∗ *measure M* (*space M*)) **if** *n*≥*1* **for** *n*
unfolding *v-def* **using** *that* **by** *simp*
then have (*λn. ereal* (∫ *x*. *v n x* / *n* ∂*M*)) ⟶ *ereal*(− *real K* ∗ *measure M* (*space M*))
using *lim-explicit* **by** *force*
moreover have (*λn. ereal* (∫ *x*. *v n x* / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal v*
using *subcocycle-int-tendsto-avg-ereal*[*OF* ‹*subcocycle v*›] **by** *auto*
ultimately have *subcocycle-avg-ereal v* = − *real K* ∗ *measure M* (*space M*)
using *LIMSEQ-unique* **by** *blast*
then have *subcocycle-avg-ereal v* > −∞
by *auto*

{
fix *x* **assume** *H*: (*λn. v n x* / *n*) ⟶ *subcocycle-lim-ereal v x*
have *ereal*(*v n x* / *n*) = −*real K* **if** *n*≥*1* **for** *n*
unfolding *v-def* **using** *that* **by** *auto*
then have (*λn. ereal*(*v n x* / *n*)) ⟶ − *real K*
using *lim-explicit* **by** *force*
then have *subcocycle-lim-ereal v x* = −*real K*
using *H LIMSEQ-unique* **by** *blast*
}
then have *AE x in M*. *subcocycle-lim-ereal v x* = −*real K*
using *kingman-theorem-AE-nonergodic-ereal*[*OF* ‹*subcocycle v*›] **by** *auto*

define *w* **where** *w* = (*λn x*. *max* (*u n x*) (*v n x*))
have [*simp*]: *subcocycle w*
unfolding *w-def* **by** (*rule subcocycle-max*, *auto*)
have *subcocycle-avg-ereal w* ≥ *subcocycle-avg-ereal v*
unfolding *w-def* **using** *subcocycle-avg-ereal-max* **by** *auto*
then have *subcocycle-avg-ereal w* > −∞

using ‹*subcocycle-avg-ereal v* > −∞› **by** *auto*

have ∗: *AE x in M. real-cond-exp M Invariants* (*u n*) *x* ≤ *real-cond-exp M Invariants* (*w n*) *x* **for** *n*
apply (*rule real-cond-exp-mono*)
using *subcocycle-integrable*[*OF assms*, *of n*] *subcocycle-integrable*[*OF* ‹*subcocycle w*›, *of n*] **apply** *auto*
unfolding *w-def* **by** *auto*
have *AE x in M*. ∀ *n. real-cond-exp M Invariants* (*u n*) *x* ≤ *real-cond-exp M Invariants* (*w n*) *x*
apply (*subst AE-all-countable*) **using** ∗ **by** *auto*
moreover have *AE x in M*. (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*) ⟶ *subcocycle-lim w x*
apply (*rule kingman-theorem-nonergodic-invariant*(*1*))
using ‹*subcocycle-avg-ereal w* > −∞› **by** *auto*
moreover have *AE x in M*. *subcocycle-lim-ereal w x* = *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
unfolding *w-def* **using** *subcocycle-lim-ereal-max* **by** *auto*
moreover
{
fix *x* **assume** *H*: (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*) ⟶ *subcocycle-lim w x*
subcocycle-lim-ereal w x = *max* (*subcocycle-lim-ereal u x*) (*subcocycle-lim-ereal v x*)
subcocycle-lim-ereal v x = − *real K*
∀ *n. real-cond-exp M Invariants* (*u n*) *x* ≤ *real-cond-exp M Invariants* (*w n*) *x*
have *subcocycle-lim-ereal w x* > −∞
using *H*(*2*) *H*(*3*)
by *auto* (*metis MInfty-neq-ereal*(*1*) *ereal-infty-less-eq2*(*2*) *max.cobounded2*)
then have *subcocycle-lim-ereal w x* = *ereal*(*subcocycle-lim w x*)
unfolding *subcocycle-lim-def* **using** *subcocycle-lim-ereal-not-PInf*[*of w x*] *ereal-real* **by** *force*
moreover have (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*) ⟶ *ereal*(*subcocycle-lim w x*) **using** *H*(*1*) **by** *auto*
ultimately have (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*) ⟶ *subcocycle-lim-ereal w x* **by** *auto*
then have ∗: *limsup* (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*) = *subcocycle-lim-ereal w x*
using *tendsto-iff-Liminf-eq-Limsup trivial-limit-at-top-linorder* **by** *blast*

have *ereal*(*real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *real-cond-exp M Invariants* (*w n*) *x* / *n* **for** *n*
using *H*(*4*) **by** (*auto simp add*: *divide-simps*)
then have *eventually* (λ*n. ereal*(*real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *real-cond-exp M Invariants* (*w n*) *x* / *n*) *sequentially*
by *auto*
then have *limsup* (λ*n. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *limsup* (λ*n. real-cond-exp M Invariants* (*w n*) *x* / *n*)

using *Limsup-mono*[*of - - sequentially*] **by** *force*
then have *limsup* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *max* (*subcocycle-lim-ereal u x*) (−*real K*)
using ∗ *H*(*2*) *H*(*3*) **by** *auto*
}
ultimately show *?thesis* **using** ‹*AE x in M. subcocycle-lim-ereal v x* = −*real K*› **by** *auto*
qed
have *AE x in M.* ∀ *K*::*nat.*
limsup (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *max* (*subcocycle-lim-ereal u x*) (−*real K*)
apply (*subst AE-all-countable*) **using** *limsup-ineq-K* **by** *auto*

moreover have *AE x in M. liminf* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≥ *subcocycle-lim-ereal u x*
proof −
have *AE x in M. N* > *0* ⟶ *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*λx. u N x* / *N*) *x* **for** *N*
by (*cases N* = *0*, *auto simp add*: *subcocycle-lim-ereal-atmost-uN-invariants*[*OF assms*(*1*)])
then have *AE x in M.* ∀ *N. N* > *0* ⟶ *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*λx. u N x* / *N*) *x*
by (*simp add*: *AE-all-countable*)
moreover have *AE x in M.* ∀ *n. real-cond-exp M Invariants* (*λx. u n x* / *n*) *x* = *real-cond-exp M Invariants* (*u n*) *x* / *n*
apply (*subst AE-all-countable*, *intro allI*) **using** *real-cond-exp-cdiv* **by** *auto*
moreover
{
fix *x* **assume** *x*: ∀ *N. N* > *0* ⟶ *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*λx. u N x* / *N*) *x*
∀ *n. real-cond-exp M Invariants* (*λx. u n x* / *n*) *x* = *real-cond-exp M Invariants* (*u n*) *x* / *n*
then have ∗: *subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*u n*) *x* / *n* **if** *n* ≥ *1* **for** *n*
using *that* **by** *auto*
have *subcocycle-lim-ereal u x* ≤ *liminf* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*)
apply (*subst liminf-bounded-iff*) **using** ∗ *less-le-trans* **by** *blast*
}
ultimately show *?thesis* **by** *auto*
qed

moreover
{
fix *x* **assume** *H*: ∀ *K*::*nat. limsup* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*)
≤ *max* (*subcocycle-lim-ereal u x*) (−*real K*)
liminf (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≥ *subcocycle-lim-ereal u x*
have (*λK*::*nat. max* (*subcocycle-lim-ereal u x*) (−*real K*)) ⟶ *subcocy-*

cle-lim-ereal u x
    by (*rule ereal-truncation-bottom*)
  with *LIMSEQ-le-const*[*OF this*]
    have ∗: *limsup* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ≤ *subcocycle-lim-ereal u x*
    using *H*(*1*) **by** *auto*
  have (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim-ereal u x*
    apply (*subst tendsto-iff-Liminf-eq-Limsup*[*OF trivial-limit-at-top-linorder*])
      using *H*(*2*) ∗ *Liminf-le-Limsup*[*OF trivial-limit-at-top-linorder*, *of* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*)]
    by *auto*
 }
 ultimately show *?thesis* **by** *auto*
qed

end

9.6 Subcocycles in the ergodic case

In this subsection, we describe how all the previous results simplify in the ergodic case. Indeed, subcocycle limits are almost surely constant, given by the asymptotic average.

context *ergodic-pmpt* **begin**

lemma *subcocycle-ergodic-lim-avg*:
 assumes *subcocycle u*
 shows *AE x in M. subcocycle-lim-ereal u x* = *subcocycle-avg-ereal u*
    AE x in M. subcocycle-lim u x = *subcocycle-avg u*
proof −
 have *I*: *integrable M* (*u N*) **for** *N* **using** *subcocycle-integrable*[*OF assms*]**by** *auto*
 obtain *c*::*ereal* **where** *c*: *AE x in M. subcocycle-lim-ereal u x* = *c*
  using *Invariant-func-is-AE-constant*[*OF subcocycle-lim-meas-Inv*(*1*)] **by** *blast*
 have *c* = *subcocycle-avg-ereal u*
 proof (*cases subcocycle-avg-ereal u* = − ∞)
  case *True*
  {
   fix *N* **assume** *N* > (*0*::*nat*)
   have *AE x in M. real-cond-exp M Invariants* (*λx. u N x* / *N*) *x* = (∫ *x. u N x* / *N* ∂*M*)
    apply (*rule Invariants-cond-exp-is-integral*) **using** *I* **by** *auto*
    moreover have *AE x in M. subcocycle-lim-ereal u x* ≤ *real-cond-exp M Invariants* (*λx. u N x* / *N*) *x*
    using *subcocycle-lim-ereal-atmost-uN-invariants*[*OF assms* ‹*N*>*0*›] **by** *simp*
   ultimately have *AE x in M. c* ≤ (∫ *x. u N x* / *N* ∂*M*)
    using *c* **by** *force*
   then have *c* ≤ (∫ *x. u N x* / *N* ∂*M*) **by** *auto*

}
then have $\forall N \geq 1.\ c \leq (\int x.\ u\ N\ x\ /\ N\ \partial M)$ **by** *auto*
with *Lim-bounded2*[*OF subcocycle-int-tendsto-avg-ereal*[*OF assms*] *this*]
have $c \leq$ *subcocycle-avg-ereal u* **by** *simp*
then show *?thesis* **using** *True* **by** *auto*
next
case *False*
then have *fin*: *subcocycle-avg-ereal u* $> -\infty$ **by** *simp*
obtain *cr::real* **where** *cr*: *AE x in M. subcocycle-lim u x = cr*
using *Invariant-func-is-AE-constant*[*OF subcocycle-lim-meas-Inv*(*2*)] **by** *blast*
have *AE x in M. c = ereal cr* **using** *c cr subcocycle-lim-real-ereal*[*OF assms fin*] **by** *force*
then have *c = ereal cr* **by** *auto*
have *subcocycle-avg u* = $(\int x.$ *subcocycle-lim u x* $\partial M)$
using *subcocycle-lim-avg*[*OF assms fin*] **by** *auto*
also have ... = $(\int x.\ cr\ \partial M)$
apply (*rule integral-cong-AE*) **using** *cr* **by** *auto*
also have ... = *cr*
by (*simp add*: *prob-space.prob-space prob-space-axioms*)
finally have *ereal*(*subcocycle-avg u*) = *ereal cr* **by** *simp*
then show *?thesis* **using** ‹ *c = ereal cr* › *subcocycle-avg-real-ereal*[*OF fin*] **by** *auto*
qed
then show *AE x in M. subcocycle-lim-ereal u x = subcocycle-avg-ereal u* **using** *c* **by** *auto*
then show *AE x in M. subcocycle-lim u x = subcocycle-avg u*
unfolding *subcocycle-lim-def subcocycle-avg-def* **by** *auto*
qed

theorem *kingman-theorem-AE-ereal*:
assumes *subcocycle u*
shows *AE x in M.* $(\lambda n.\ u\ n\ x\ /\ n) \longrightarrow$ *subcocycle-avg-ereal u*
using *kingman-theorem-AE-nonergodic-ereal*[*OF assms*] *subcocycle-ergodic-lim-avg*(*1*)[*OF assms*] **by** *auto*

theorem *kingman-theorem*:
assumes *subcocycle u subcocycle-avg-ereal u* $> -\infty$
shows *AE x in M.* $(\lambda n.\ u\ n\ x\ /\ n) \longrightarrow$ *subcocycle-avg u*
$(\lambda n.\ (\int^+ x.\ abs(u\ n\ x\ /\ n - \textit{subcocycle-avg}\ u)\ \partial M)) \longrightarrow 0$
proof −
have ∗: *AE x in M. subcocycle-lim u x = subcocycle-avg u*
using *subcocycle-ergodic-lim-avg*(*2*)[*OF assms*(*1*)] **by** *auto*
then show *AE x in M.* $(\lambda n.\ u\ n\ x\ /\ n) \longrightarrow$ *subcocycle-avg u*
using *kingman-theorem-nonergodic*(*1*)[*OF assms*] **by** *auto*
have $(\int^+ x.\ abs(u\ n\ x\ /\ n - \textit{subcocycle-avg}\ u)\ \partial M) = (\int^+ x.\ abs(u\ n\ x\ /\ n - \textit{subcocycle-lim}\ u\ x)\ \partial M)$ **for** *n*
apply (*rule nn-integral-cong-AE*) **using** ∗ **by** *auto*
then show $(\lambda n.\ (\int^+ x.\ abs(u\ n\ x\ /\ n - \textit{subcocycle-avg}\ u)\ \partial M)) \longrightarrow 0$
using *kingman-theorem-nonergodic*(*3*)[*OF assms*] **by** *auto*

qed

end

9.7 Subocycles for invertible maps

If T is invertible, then a subcocycle u_n for T gives rise to another subcocycle for T^{-1}. Intuitively, if u is subadditive along the time interval $[0, n)$, then it should also be subadditive along the time interval $[-n, 0)$. This is true, and formalized with the following statement.

proposition (**in** *mpt*) *subcocycle-u-Tinv*:
  assumes *subcocycle u*
      invertible-qmpt
  shows *mpt.subcocycle M Tinv* ($\lambda n\ x.\ u\ n\ (((Tinv)\widehat{\ }\widehat{\ }n)\ x)$)
proof −
  have *bij*: *bij T* **using** ‹*invertible-qmpt*› **unfolding** *invertible-qmpt-def* **by** *auto*
  have *int*: *integrable M* (*u n*) **for** *n*
    using *subcocycle-integrable*[*OF assms*(*1*)] **by** *simp*
  interpret *I*: *mpt M Tinv* **using** *Tinv-mpt*[*OF assms*(*2*)] **by** *simp*
  show *I.subcocycle* ($\lambda n\ x.\ u\ n\ (((Tinv)\widehat{\ }\widehat{\ }n)\ x)$) **unfolding** *I.subcocycle-def*
  proof(*auto*)
    show *integrable M* ($\lambda x.\ u\ n\ ((Tinv\ \widehat{\ }\widehat{\ }\ n)\ x)$) **for** *n*
      using *I.Tn-integral-preserving*(*1*)[*OF int*[*of n*]] **by** *simp*
    fix *n m*::*nat* **and** *x*::*'a*
    define *y* **where** $y = (Tinv\widehat{\ }\widehat{\ }(m+n))\ x$
     have $(T\widehat{\ }\widehat{\ }m)\ y = (T\widehat{\ }\widehat{\ }m)\ ((Tinv\widehat{\ }\widehat{\ }m)\ ((Tinv\widehat{\ }\widehat{\ }n)\ x))$ **unfolding** *y-def* **by** (*simp add*: *funpow-add*)
    then have $*$: $(T\widehat{\ }\widehat{\ }m)\ y = (Tinv\widehat{\ }\widehat{\ }n)\ x$
      using *fn-o-inv-fn-is-id*[*OF bij*, *of m*] **by** (*metis Tinv-def comp-def*)
    have $u\ (n + m)\ ((Tinv\ \widehat{\ }\widehat{\ }\ (n + m))\ x) = u\ (m+n)\ y$
      unfolding *y-def* **by** (*simp add*: *add.commute*[*of n m*])
    also have ... $\leq u\ m\ y + u\ n\ ((T\widehat{\ }\widehat{\ }m)\ y)$
      using *subcocycle-ineq*[*OF* ‹*subcocycle u*›, *of m n y*] **by** *simp*
    also have ... $= u\ m\ ((Tinv\widehat{\ }\widehat{\ }(m+n))\ x) + u\ n\ ((Tinv\widehat{\ }\widehat{\ }n)\ x)$
      using $*$ *y-def* **by** *auto*
    finally show $u\ (n + m)\ ((Tinv\ \widehat{\ }\widehat{\ }\ (n + m))\ x) \leq u\ n\ ((Tinv\ \widehat{\ }\widehat{\ }\ n)\ x) + u\ m\ ((Tinv\ \widehat{\ }\widehat{\ }\ m)\ ((Tinv\ \widehat{\ }\widehat{\ }\ n)\ x))$
      by (*simp add*: *funpow-add*)
  qed
qed

The subcocycle averages for T and T^{-1} coincide.

proposition (**in** *mpt*) *subcocycle-avg-ereal-Tinv*:
  assumes *subcocycle u*
      invertible-qmpt
  shows *mpt.subcocycle-avg-ereal M* ($\lambda n\ x.\ u\ n\ (((Tinv)\widehat{\ }\widehat{\ }n)\ x)$) = *subcocycle-avg-ereal u*
proof −

have *bij*: *bij T* **using** ‹*invertible-qmpt*› **unfolding** *invertible-qmpt-def* **by** *auto*
have *int*: *integrable M* (*u n*) **for** *n*
using *subcocycle-integrable*[*OF assms*(*1*)] **by** *simp*
interpret *I*: *mpt M Tinv* **using** *Tinv-mpt*[*OF assms*(*2*)] **by** *simp*
have (*λn.* (*∫ x. u n* (((*Tinv*)^^*n*) *x*) / *n* ∂*M*)) ⟶ *I.subcocycle-avg-ereal* (*λn x. u n* (((*Tinv*)^^*n*) *x*))
using *I.subcocycle-int-tendsto-avg-ereal*[*OF subcocycle-u-Tinv*[*OF assms*]] **by** *simp*
moreover have (*∫ x. u n x* / *n* ∂*M*) = *ereal* (*∫ x. u n* (((*Tinv*)^^*n*) *x*) / *n* ∂*M*) **for** *n*
apply (*simp*)
apply (*rule disjI2*)
apply (*rule I.Tn-integral-preserving*(*2*)[*symmetric*])
apply (*simp add*: *int*)
done
ultimately have (*λn.* (*∫ x. u n x* / *n* ∂*M*)) ⟶ *I.subcocycle-avg-ereal* (*λn x. u n* (((*Tinv*)^^*n*) *x*))
by *presburger*
moreover have (*λn.* (*∫ x. u n x* / *n* ∂*M*)) ⟶ *subcocycle-avg-ereal u*
using *subcocycle-int-tendsto-avg-ereal*[*OF* ‹*subcocycle u*›] **by** *simp*
ultimately show *?thesis*
using *LIMSEQ-unique* **by** *simp*
qed

The asymptotic limit of the subcocycle is the same for T and T^{-1}. This is clear in the ergodic case, and follows from the ergodic decomposition in the general case (on a standard probability space). We give a direct proof below (on a general probability space) using the fact that the asymptotic limit is the same for the subcocycle conditioned by the invariant sigma-algebra, which is clearly the same for T and T^{-1} as it is constant along orbits.

proposition (**in** *fmpt*) *subcocycle-lim-ereal-Tinv*:
assumes *subcocycle u*
invertible-qmpt
shows *AE x in M. fmpt.subcocycle-lim-ereal M Tinv* (*λn x. u n* (((*Tinv*)^^*n*) *x*)) *x* = *subcocycle-lim-ereal u x*
proof –
have *bij*: *bij T* **using** ‹*invertible-qmpt*› **unfolding** *invertible-qmpt-def* **by** *auto*
have *int*: *integrable M* (*u n*) **for** *n*
using *subcocycle-integrable*[*OF assms*(*1*)] **by** *simp*
interpret *I*: *fmpt M Tinv* **using** *Tinv-fmpt*[*OF assms*(*2*)] **by** *simp*
have ∗: *AE x in M. real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x*
= *real-cond-exp M I.Invariants* (*u n*) *x* **for** *n*
using *I.Invariants-of-foTn int* **unfolding** *o-def* **by** *simp*
then have *AE x in M.* ∀ *n. real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x*
= *real-cond-exp M I.Invariants* (*u n*) *x*
by (*simp add*: *AE-all-countable*)
moreover have *AE x in M.* (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim-ereal u x*

using *kingman-theorem-AE-nonergodic-invariant-ereal*[*OF* ‹*subcocycle u*›] **by** *simp*
moreover have *AE x in M*. (*λn. real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x* / *n*)
⟶ *I.subcocycle-lim-ereal* (*λ n x. u n* (((*Tinv*)^^*n*) *x*)) *x*
using *I.kingman-theorem-AE-nonergodic-invariant-ereal*[*OF subcocycle-u-Tinv*[*OF assms*]] **by** *simp*
moreover
{
fix *x* **assume** *H*: ∀ *n. real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x*
= *real-cond-exp M I.Invariants* (*u n*) *x*
(*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*) ⟶ *subcocycle-lim-ereal u x*
(*λn. real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x* / *n*)
⟶ *I.subcocycle-lim-ereal* (*λ n x. u n* (((*Tinv*)^^*n*) *x*)) *x*
have *ereal*(*real-cond-exp M Invariants* (*u n*) *x* / *n*)
= *ereal*(*real-cond-exp M I.Invariants* (*λ x. u n* (((*Tinv*)^^*n*) *x*)) *x* / *n*) **for** *n*
using *H*(*1*) *Invariants-Tinv*[*OF* ‹*invertible-qmpt*›] **by** *auto*
then have (*λn. real-cond-exp M Invariants* (*u n*) *x* / *n*)
⟶ *I.subcocycle-lim-ereal* (*λ n x. u n* (((*Tinv*)^^*n*) *x*)) *x*
using *H*(*3*) **by** *presburger*
then have *I.subcocycle-lim-ereal* (*λ n x. u n* (((*Tinv*)^^*n*) *x*)) *x* = *subcocycle-lim-ereal u x*
using *H*(*2*) *LIMSEQ-unique* **by** *auto*
}
ultimately show *?thesis* **by** *auto*
qed

proposition (**in** *fmpt*) *subcocycle-lim-Tinv*:
assumes *subcocycle u*
invertible-qmpt
shows *AE x in M*. *fmpt.subcocycle-lim M Tinv* (*λn x. u n* (((*Tinv*)^^*n*) *x*)) *x* = *subcocycle-lim u x*
proof −
interpret *I*: *fmpt M Tinv* **using** *Tinv-fmpt*[*OF assms*(*2*)] **by** *simp*
show *?thesis*
unfolding *subcocycle-lim-def I.subcocycle-lim-def*
using *subcocycle-lim-ereal-Tinv*[*OF assms*] **by** *auto*
qed

end

10 Gouezel-Karlsson

theory *Gouezel-Karlsson*
imports *Asymptotic-Density Kingman*

begin

This section is devoted to the proof of the main ergodic result of the article "Subadditive and multiplicative ergodic theorems" by Gouezel and Karlsson [GK15]. It is a version of Kingman theorem ensuring that, for subadditive cocycles, there are almost surely many times n where the cocycle is nearly additive at *all* times between 0 and n.

This theorem is then used in this article to construct horofunctions characterizing the behavior at infinity of compositions of semi-contractions. This requires too many further notions to be implemented in current Isabelle/HOL, but the main ergodic result is completely proved below, in Theorem `Gouezel_Karlsson`, following the arguments in the paper (but in a slightly more general setting here as we are not making any ergodicity assumption).

To simplify the exposition, the theorem is proved assuming that the limit of the subcocycle vanishes almost everywhere, in the locale `Gouezel_Karlsson_Kingman`. The final result is proved by an easy reduction to this case.

The main steps of the proof are as follows:

- assume first that the map is invertible, and consider the inverse map and the corresponding inverse subcocycle. With combinatorial arguments that only work for this inverse subcocycle, we control the density of bad times given some allowed error $d > 0$, in a precise quantitative way, in Lemmas `upper_density_all_times` and `upper_density_large_k`. We put these estimates together in Lemma `upper_density_delta`.
- These estimates are then transfered to the original time direction and the original subcocycle in Lemma `upper_density_good_direction_invertible`. The fact that we have quantitative estimates in terms of asymptotic densities is central here, just having some infiniteness statement would not be enough.
- The invertibility assumption is removed in Lemma `upper_density_good_direction` by using the result in the natural extension.
- Finally, the main result is deduced in Lemma `infinite_AE` (still assuming that the asymptotic limit vanishes almost everywhere), and in full generality in Theorem `Gouezel_Karlsson_Kingman`.

lemma *upper-density-eventually-measure*:
  fixes $a{::}real$
  assumes [*measurable*]: $\bigwedge n.\ \{x \in space\ M.\ P\ x\ n\} \in sets\ M$
    and $emeasure\ M\ \{x \in space\ M.\ upper\text{-}asymptotic\text{-}density\ \{n.\ P\ x\ n\} < a\} > b$
  shows $\exists N.\ emeasure\ M\ \{x \in space\ M.\ \forall n \geq N.\ card\ (\{n.\ P\ x\ n\} \cap \{..<n\}) < a * n\} > b$
proof −
  define G **where** $G = \{x \in space\ M.\ upper\text{-}asymptotic\text{-}density\ \{n.\ P\ x\ n\} < a\}$

define H **where** $H = (\lambda N.\ \{x \in \mathit{space}\ M.\ \forall\, n \geq N.\ \mathit{card}\ (\{n.\ P\ x\ n\} \cap \{..<n\}) < a * n\})$
have [*measurable*]: $G \in \mathit{sets}\ M\ \bigwedge N.\ H\ N \in \mathit{sets}\ M$ **unfolding** *G-def H-def* **by** *auto*
have $G \subseteq (\bigcup N.\ H\ N)$
proof
fix x **assume** $x \in G$
then have $x \in \mathit{space}\ M$ **unfolding** *G-def* **by** *simp*
have *eventually* $(\lambda n.\ \mathit{card}(\{n.\ P\ x\ n\} \cap \{..<n\}) < a * n)$ *sequentially*
using ‹$x \in G$› **unfolding** *G-def* **using** *upper-asymptotic-densityD* **by** *auto*
then obtain N **where** $\bigwedge n.\ n \geq N \Longrightarrow \mathit{card}(\{n.\ P\ x\ n\} \cap \{..<n\}) < a * n$
using *eventually-sequentially* **by** *auto*
then have $x \in H\ N$ **unfolding** *H-def* **using** ‹$x \in \mathit{space}\ M$› **by** *auto*
then show $x \in (\bigcup N.\ H\ N)$ **by** *blast*
qed
have $b < \mathit{emeasure}\ M\ G$ **using** *assms(2)* **unfolding** *G-def* **by** *simp*
also have ... $\leq \mathit{emeasure}\ M\ (\bigcup N.\ H\ N)$
apply (*rule emeasure-mono*) **using** ‹$G \subseteq (\bigcup N.\ H\ N)$› **by** *auto*
finally have $\mathit{emeasure}\ M\ (\bigcup N.\ H\ N) > b$ **by** *simp*
moreover have $(\lambda N.\ \mathit{emeasure}\ M\ (H\ N)) \longrightarrow \mathit{emeasure}\ M\ (\bigcup N.\ H\ N)$
apply (*rule Lim-emeasure-incseq*) **unfolding** *H-def incseq-def* **by** *auto*
ultimately have *eventually* $(\lambda N.\ \mathit{emeasure}\ M\ (H\ N) > b)$ *sequentially*
by (*simp add*: *order-tendsto-iff*)
then obtain N **where** $\mathit{emeasure}\ M\ (H\ N) > b$
using *eventually-False-sequentially eventually-mono* **by** *blast*
then show *?thesis* **unfolding** *H-def* **by** *blast*
qed

locale *Gouezel-Karlsson-Kingman* = *pmpt* +
fixes $u::\mathit{nat} \Rightarrow {'a} \Rightarrow \mathit{real}$
assumes *subu*: *subcocycle u*
and *subu-fin*: *subcocycle-avg-ereal* $u > -\infty$
and *subu-0*: *AE* x *in* M. *subcocycle-lim* $u\ x = 0$
begin

lemma *int-u* [*measurable*]:
integrable $M\ (u\ n)$
using *subu* **unfolding** *subcocycle-def* **by** *auto*

Next lemma is Lemma 2.1 in [GK15].

lemma *upper-density-all-times*:
assumes $d > (0::\mathit{real})$
shows $\exists\, c> (0::\mathit{real})$.
$\mathit{emeasure}\ M\ \{x \in \mathit{space}\ M.\ \mathit{upper\text{-}asymptotic\text{-}density}\ \{n.\ \exists\, l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ c * l\} < d\} > 1 - d$
proof −
define f **where** $f = (\lambda x.\ \mathit{abs}\ (u\ 1\ x))$
have [*measurable*]: $f \in \mathit{borel\text{-}measurable}\ M$ **unfolding** *f-def* **by** *auto*

define *G* **where** *G* = {*x* ∈ *space M*. (λ*n*. *birkhoff-sum f n x* / *n*) ⟶ *real-cond-exp M Invariants f x*
∧ (λ*n*. *u n x* / *n*) ⟶ *0*}
have [*measurable*]: *G* ∈ *sets M* **unfolding** *G-def* **by** *auto*
have *AE x in M*. (λ*n*. *birkhoff-sum f n x* / *n*) ⟶ *real-cond-exp M Invariants f x*
apply (*rule birkhoff-theorem-AE-nonergodic*) **using** *subu* **unfolding** *f-def subcocycle-def* **by** *auto*
moreover have *AE x in M*. (λ*n*. *u n x* / *n*) ⟶ *0*
using *subu-0 kingman-theorem-nonergodic*(*1*)[*OF subu subu-fin*] **by** *auto*
ultimately have *AE x in M*. *x* ∈ *G* **unfolding** *G-def* **by** *auto*
then have *emeasure M G* = *1* **by** (*simp add*: *emeasure-eq-1-AE*)

define *V* **where** *V* = (λ*c x*. {*n*. ∃ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *c* ∗ *l*})
define *Good* **where** *Good* = (λ*c*. {*x* ∈ *G*. *upper-asymptotic-density* (*V c x*) < *d*})
have [*measurable*]: *Good c* ∈ *sets M* **for** *c* **unfolding** *Good-def V-def* **by** *auto*

have *I*: *upper-asymptotic-density* (*V c x*) ≤ *real-cond-exp M Invariants f x* / *c* **if** *c*>*0* *x* ∈ *G* **for** *c x*
proof −
have [*simp*]: *c*>*0* *c* ≠ *0* *c* ≥ *0* **using** ‹*c*>*0*› **by** *auto*
define *U* **where** *U* = (λ*n*. *abs*(*u 0 x*) + *birkhoff-sum f n x* − *c* ∗ *card* (*V c x* ∩ {*1*..*n*}))
have *main*: *u n x* ≤ *U n* **for** *n*
proof (*rule nat-less-induct*)
fix *n* **assume** *H*: ∀ *m*<*n*. *u m x* ≤ *U m*
consider *n* = *0* | *n*≥*1* ∧ *n* ∉ *V c x* | *n*≥*1* ∧ *n* ∈ *V c x* **by** *linarith*
then show *u n x* ≤ *U n*
proof (*cases*)
assume *n* = *0*
then show *?thesis* **unfolding** *U-def* **by** *auto*
next
assume *A*: *n*≥*1* ∧ *n* ∉ *V c x*
then have *n* ≥ *1* **by** *simp*
then have *n*−*1*<*n* **by** *simp*
have {*1*..*n*} = {*1*..*n*−*1*} ∪ {*n*} **using** ‹*1* ≤ *n*› *atLeastLessThanSuc* **by** *auto*
then have ∗: *card* (*V c x* ∩ {*1*..*n*}) = *card* (*V c x* ∩ {*1*..*n*−*1*}) **using** *A* **by** *auto*
have *u n x* ≤ *u* (*n*−*1*) *x* + *u 1* ((*T*^^(*n*−*1*)) *x*)
using ‹*n*≥*1*› *subu* **unfolding** *subcocycle-def* **by** (*metis le-add-diff-inverse2*)
also have ... ≤ *U* (*n*−*1*) + *f* ((*T*^^(*n*−*1*)) *x*) **unfolding** *f-def* **using** *H* ‹*n*−*1*<*n*› **by** *auto*
also have ... = *abs*(*u 0 x*) + *birkhoff-sum f* (*n*−*1*) *x* + *f* ((*T*^^(*n*−*1*)) *x*) − *c* ∗ *card* (*V c x* ∩ {*1*..*n*−*1*})
unfolding *U-def* **by** *auto*
also have ... = *abs*(*u 0 x*) + *birkhoff-sum f n x* − *c* ∗ *card* (*V c x* ∩ {*1*..*n*})
using ∗ *birkhoff-sum-cocycle*[*of f n*−*1 1 x*] ‹*1* ≤ *n*› **by** *auto*
also have ... = *U n* **unfolding** *U-def* **by** *simp*

finally show *?thesis* **by** *auto*
next
assume *B*: *n≥1 ∧ n ∈ V c x*
then obtain *l* **where** *l*: *l∈{1..n} u n x − u (n−l) x ≤ − c ∗ l* **unfolding** *V-def* **by** *blast*
then have *n−l < n* **by** *simp*
have *m*: *− (r ∗ ra) − r ∗ rb = − (r ∗ (rb + ra))* **for** *r ra rb::real*
by (*simp add*: *algebra-simps*)

have *card(V c x ∩ {1..n}) ≤ card ((V c x ∩ {1..n−l}) ∪ {n−l+1..n})*
by (*rule card-mono*, *auto*)
also have ... *≤ card (V c x ∩ {1..n−l}) + card {n−l+1..n}*
by (*rule card-Un-le*)
also have ... *≤ card (V c x ∩ {1..n−l}) + l* **by** *auto*
finally have *card(V c x ∩ {1..n}) ≤ card (V c x ∩ {1..n−l}) + real l* **by** *auto*
then have ∗: *−c ∗ card (V c x ∩ {1..n−l}) − c ∗ l ≤ −c ∗ card(V c x ∩ {1..n})*
using *m* **by** *auto*

have *birkhoff-sum f ((n−l) + l) x = birkhoff-sum f (n−l) x + birkhoff-sum f l ((T^^(n−l))x)*
by (*rule birkhoff-sum-cocycle*)
moreover have *birkhoff-sum f l ((T^^(n−l))x) ≥ 0*
unfolding *f-def birkhoff-sum-def* **using** *sum-nonneg* **by** *auto*
ultimately have ∗∗: *birkhoff-sum f (n−l) x ≤ birkhoff-sum f n x* **using** *l(1)* **by** *auto*

have *u n x ≤ u (n−l) x − c ∗ l* **using** *l* **by** *simp*
also have ... *≤ U (n−l) − c∗ l* **using** *H ‹n−l < n›* **by** *auto*
also have ... *= abs(u 0 x) + birkhoff-sum f (n−l) x − c ∗ card (V c x ∩ {1..n−l}) − c∗l*
unfolding *U-def* **by** *auto*
also have ... *≤ abs(u 0 x) + birkhoff-sum f n x − c ∗ card (V c x ∩ {1..n})*
using ∗ ∗∗ **by** *simp*
finally show *?thesis* **unfolding** *U-def* **by** *auto*
qed
qed

have (*λn. abs(u 0 x) ∗ (1/n) + birkhoff-sum f n x / n − u n x / n*) ⟶ *abs(u 0 x) ∗ 0 + real-cond-exp M Invariants f x − 0*
apply (*intro tendsto-intros*) **using** *‹x ∈ G›* **unfolding** *G-def* **by** *auto*
moreover have (*abs(u 0 x) + birkhoff-sum f n x − u n x)/n = abs(u 0 x) ∗ (1/n) + birkhoff-sum f n x / n − u n x / n* **for** *n*
by (*auto simp add*: *add-divide-distrib diff-divide-distrib*)
ultimately have (*λn.* (*abs(u 0 x) + birkhoff-sum f n x − u n x)/n*) ⟶ *real-cond-exp M Invariants f x*
by *auto*
then have *a*: *limsup* (*λn.* (*abs(u 0 x) + birkhoff-sum f n x − u n x)/n*) =

real-cond-exp M Invariants f x
by (*simp add*: *assms lim-imp-Limsup*)

have *c* ∗ *card* (*V c x* ∩ {*1..n*})/*n* ≤ (*abs*(*u 0 x*) + *birkhoff-sum f n x* − *u n x*)/*n* **for** *n*
using *main*[*of n*] **unfolding** *U-def* **by** (*simp add*: *divide-right-mono*)
then have *limsup* (λ*n*. *c* ∗ *card* (*V c x* ∩ {*1..n*})/*n*) ≤ *limsup* (λ*n*. (*abs*(*u 0 x*) + *birkhoff-sum f n x* − *u n x*)/*n*)
by (*simp add*: *Limsup-mono*)
then have *b*: *limsup* (λ*n*. *c* ∗ *card* (*V c x* ∩ {*1..n*})/*n*) ≤ *real-cond-exp M Invariants f x*
using *a* **by** *simp*

have *ereal*(*upper-asymptotic-density* (*V c x*)) = *limsup* (λ*n*. *card* (*V c x* ∩ {*1..n*})/*n*)
using *upper-asymptotic-density-shift*[*of V c x 1 0*] **by** *auto*
also have ... = *limsup* (λ*n*. *ereal*(*1*/*c*) ∗ *ereal*(*c* ∗ *card* (*V c x* ∩ {*1..n*})/*n*))
by *auto*
also have ... = (*1*/*c*) ∗ *limsup* (λ*n*. *c* ∗ *card* (*V c x* ∩ {*1..n*})/*n*)
by (*rule limsup-ereal-mult-left*, *auto*)
also have ... ≤ *ereal* (*1*/*c*) ∗ *real-cond-exp M Invariants f x*
by (*rule ereal-mult-left-mono*[*OF b*], *auto*)
finally show *upper-asymptotic-density* (*V c x*) ≤ *real-cond-exp M Invariants f x* / *c*
by *auto*
qed

{
fix *r*::*real*
obtain *c*::*nat* **where** *r* / *d* < *c* **using** *reals-Archimedean2* **by** *auto*
then have *r*/*d* < *real c+1* **by** *auto*
then have *r* / (*real c+1*) < *d* **using** ‹*d*>*0*› **by** (*simp add*: *divide-less-eq mult.commute*)
then have ∃ *c*::*nat*. *r* / (*real c+1*) < *d* **by** *auto*
}
then have *unG*: (⋃ *c*::*nat*. {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*c+1*) < *d*}) = *G*
by *auto*

have ∗: *r* < *d* ∗ (*real n* + *1*) **if** *m* ≤ *n r* < *d* ∗ (*real m* + *1*) **for** *m n r*
proof −
have *d* ∗ (*real m* + *1*) ≤ *d* ∗ (*real n* + *1*) **using** ‹*d*>*0*› ‹*m* ≤ *n*› **by** *auto*
then show *?thesis* **using** ‹*r* < *d* ∗ (*real m* + *1*)› **by** *auto*
qed
have (λ*c*. *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*real c+1*) < *d*})
⟶ *emeasure M* (⋃ *c*::*nat*. {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*c+1*) < *d*})
apply (*rule Lim-emeasure-incseq*) **unfolding** *incseq-def* **by** (*auto simp add*:

divide-simps ∗)
then have (λ*c*. *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*real c*+*1*) < *d*}) ⟶ *emeasure M G*
using *unG* **by** *auto*
then have (λ*c*. *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*real c*+*1*) < *d*}) ⟶ *1*
using ‹*emeasure M G* = *1*› **by** *simp*
then have *eventually* (λ*c*. *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*real c*+*1*) < *d*} > *1* − *d*) *sequentially*
apply (*rule order-tendstoD*)
apply (*insert* ‹*0*<*d*›, *auto simp add*: *ennreal-1*[*symmetric*] *ennreal-lessI simp del*: *ennreal-1*)
done
then obtain *c0* **where** *c0*: *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / (*real c0*+*1*) < *d*} > *1* − *d*
using *eventually-sequentially* **by** *auto*
define *c* **where** *c* = *real c0* + *1*
then have *c* > *0* **by** *auto*
have ∗: *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / *c* < *d*} > *1* − *d*
unfolding *c-def* **using** *c0* **by** *auto*
have {*x* ∈ *G*. *real-cond-exp M Invariants f x* / *c* < *d*} ⊆ {*x* ∈ *space M*. *upper-asymptotic-density* (*V c x*) < *d*}
apply *auto*
using *G-def* **apply** *blast*
using *I*[*OF* ‹*c*>*0*›] **by** *fastforce*
then have *emeasure M* {*x* ∈ *G*. *real-cond-exp M Invariants f x* / *c* < *d*} ≤ *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* (*V c x*) < *d*}
apply (*rule emeasure-mono*) **unfolding** *V-def* **by** *auto*
then have *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* (*V c x*) < *d*} > *1* − *d* **using** ∗ **by** *auto*
then show *?thesis* **unfolding** *V-def* **using** ‹*c*>*0*› **by** *auto*
qed

Next lemma is Lemma 2.2 in [GK15].

lemma *upper-density-large-k*:
assumes *d* > (*0*::*real*) *d* ≤ *1*
shows ∃ *k*::*nat*.
emeasure M {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *d* ∗ *l*} < *d*} > *1* − *d*
proof −
have [*simp*]: *d*>*0* *d* ≠ *0* *d* ≥ *0* **using** ‹*d*>*0*› **by** *auto*
define *rho* **where** *rho* = *d* ∗ *d* ∗ *d* / *4*
have [*simp*]: *rho* > *0* *rho* ≠ *0* *rho* ≥ *0* **unfolding** *rho-def* **using** *assms* **by** *auto*

First step: choose a time scale s at which all the computations will be done. the integral of u_s should be suitably small – how small precisely is given by ρ.

have *ennreal*(∫ *x*. *abs*(*u n x* / *n*) ∂*M*) = (∫ ⁺*x*. *abs*(*u n x* /*n* − *subcocycle-lim u x*) ∂*M*) **for** *n*

proof −
have *ennreal*($\int x$. *abs*(*u n x* / *n*) ∂M) = ($\int^+ x$. *abs*(*u n x* /*n*) ∂M)
apply (*rule nn-integral-eq-integral*[*symmetric*]) **using** *int-u* **by** *auto*
also have ... = ($\int^+ x$. *abs*(*u n x* /*n* − *subcocycle-lim u x*) ∂M)
apply (*rule nn-integral-cong-AE*) **using** *subu-0* **by** *auto*
finally show *?thesis* **by** *simp*
qed
moreover have (λn. $\int^+ x$. *abs*(*u n x* /*n* − *subcocycle-lim u x*) ∂M) $\longlongrightarrow$ *0*
by (*rule kingman-theorem-nonergodic*(*3*)[*OF subu subu-fin*])
ultimately have (λn. *ennreal*($\int x$. *abs*(*u n x* / *n*) ∂M)) $\longlongrightarrow$ *0*
by *auto*
then have (λn. ($\int x$. *abs*(*u n x* / *n*) ∂M)) $\longlongrightarrow$ *0*
by (*simp add*: *ennreal-0*[*symmetric*] *del*: *ennreal-0*)
then have *eventually* (λn. ($\int x$. *abs*(*u n x* / *n*) ∂M) < *rho*) *sequentially*
by (*rule order-tendstoD*(*2*), *auto*)
then obtain *s*::*nat* **where** [*simp*]: *s*>*0* **and** *s-int*: ($\int x$. *abs*(*u s x* / *s*) ∂M) < *rho*
by (*metis* (*mono-tags*, *lifting*) *neq0-conv eventually-sequentially gr-implies-not0 linorder-not-le of-nat-0-eq-iff order-refl zero-neq-one*)

Second step: a truncation argument, to decompose $|u_1|$ as a sum of a constant (its contribution will be small if k is large at the end of the argument) and of a function with small integral).

have (λn. ($\int x$. *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x* ∂M)) $\longrightarrow$ ($\int x$. *0* ∂M)
proof (*rule integral-dominated-convergence*[**where** *?w* = λx. *abs*(*u 1 x*)])
show *AE x in M*. *norm* (*abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x*) ≤ *abs*(*u 1 x*) **for** *n*
unfolding *indicator-def* **by** *auto*
{
fix *x*
have *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x* = (*0*::*real*) **if** *n* > *abs*(*u 1 x*) **for** *n*::*nat*
unfolding *indicator-def* **using** *that* **by** *auto*
then have *eventually* (λn. *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x* = *0*) *sequentially*
by (*metis* (*mono-tags*, *lifting*) *eventually-at-top-linorder reals-Archimedean2 less-le-trans of-nat-le-iff*)
then have (λn. *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x*) $\longrightarrow$ *0*
by (*rule tendsto-eventually*)
}
then show *AE x in M*. (λn. *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x*) $\longrightarrow$ *0*
by *simp*
qed (*auto simp add*: *int-u*)
then have *eventually* (λn. ($\int x$. *abs*(*u 1 x*) ∗ *indicator* {*x* ∈ *space M*. *abs*(*u 1 x*) ≥ *n*} *x* ∂M) < *rho*) *sequentially*
by (*rule order-tendstoD*(*2*), *auto*)

then obtain *Knat*::*nat* **where** *Knat*: $Knat > 0$ ($\int x.\ abs(u\ 1\ x) * indicator\ \{x \in space\ M.\ abs(u\ 1\ x) \geq Knat\}\ x\ \partial M) < rho$
by (*metis* (*mono-tags*, *lifting*) *eventually-sequentially gr-implies-not0 neq0-conv linorder-not-le of-nat-0-eq-iff order-refl zero-neq-one*)
define *K* **where** $K = real\ Knat$
then have [*simp*]: $K \geq 0$ $K>0$ **and** *K*: ($\int x.\ abs(u\ 1\ x) * indicator\ \{x \in space\ M.\ abs(u\ 1\ x) \geq K\}\ x\ \partial M) < rho$
using *Knat* **by** *auto*

define *F* **where** $F = (\lambda x.\ abs(u\ 1\ x) * indicator\ \{x.\ abs(u\ 1\ x) \geq K\}\ x)$
have *int-F* [*measurable*]: *integrable M F*
unfolding *F-def* **apply** (*rule Bochner-Integration.integrable-bound*[**where** *?f* $= \lambda x.\ abs(u\ 1\ x)$])
unfolding *indicator-def* **by** (*auto simp add*: *int-u*)
have ($\int x.\ F\ x\ \partial M) = (\int x.\ abs(u\ 1\ x) * indicator\ \{x \in space\ M.\ abs(u\ 1\ x) \geq K\}\ x\ \partial M)$
apply (*rule integral-cong-AE*) **unfolding** *F-def* **by** (*auto simp add*: *indicator-def*)
then have *F-int*: ($\int x.\ F\ x\ \partial M) < rho$ **using** *K* **by** *auto*
have *F-pos*: $F\ x \geq 0$ **for** *x* **unfolding** *F-def* **by** *auto*
have *u1-bound*: $abs(u\ 1\ x) \leq K + F\ x$ **for** *x*
unfolding *F-def indicator-def* **apply** (*cases* $x \in \{x \in space\ M.\ K \leq |u\ 1\ x|\}$) **by** *auto*

define *F2* **where** $F2 = (\lambda x.\ F\ x + abs(u\ s\ x/s))$
have *int-F2* [*measurable*]: *integrable M F2*
unfolding *F2-def* **using** *int-F int-u*[*of s*] **by** *auto*
have *F2-pos*: $F2\ x \geq 0$ **for** *x* **unfolding** *F2-def* **using** *F-pos* **by** *auto*
have ($\int x.\ F2\ x\ \partial M) = (\int x.\ F\ x\ \partial M) + (\int x.\ abs(u\ s\ x/s)\ \partial M)$
unfolding *F2-def* **apply** (*rule Bochner-Integration.integral-add*) **using** *int-F int-u* **by** *auto*
then have *F2-int*: ($\int x.\ F2\ x\ \partial M) < 2 * rho$
using *F-int s-int* **by** *auto*

We can now choose k, large enough. The reason for our choice will only appear at the end of the proof.

define *k* **where** $k = max\ (2 * s + 1)\ (nat(ceiling((2 * d * s + 2 * K * s)/(d/2))))$
have $k > 2 * s$ **unfolding** *k-def* **by** *auto*
have $k \geq (2 * d * s + 2 * K * s)/(d/2)$
unfolding *k-def* **by** *linarith*
then have $(2 * d * s + 2 * K * s)/k \leq d/2$
using ‹$k > 2 * s$› **by** (*simp add*: *divide-simps mult.commute*)

Third step: definition of a good set G where everything goes well.

define *G* **where** $G = \{x \in space\ M.\ (\lambda n.\ u\ n\ x\ /\ n) \longrightarrow 0$
$\wedge\ (\lambda n.\ birkhoff\text{-}sum\ (\lambda x.\ abs(u\ s\ x\ /\ s))\ n\ x\ /\ n) \longrightarrow real\text{-}cond\text{-}exp\ M\ Invariants\ (\lambda x.\ abs(u\ s\ x\ /\ s))\ x$

∧ (λn. *birkhoff-sum F n x / n*) ⟶ *real-cond-exp M Invariants F x*
∧ *real-cond-exp M Invariants F x* + *real-cond-exp M Invariants* (λx. *abs*(*u s x / s*)) *x* = *real-cond-exp M Invariants F2 x*}
have [*measurable*]: *G* ∈ *sets M* **unfolding** *G-def* **by** *auto*
have *AE x in M*. (λn. *u n x / n*) ⟶ *0*
using *kingman-theorem-nonergodic*(*1*)[*OF subu subu-fin*] *subu-0* **by** *auto*
moreover have *AE x in M*.(λn. *birkhoff-sum* (λx. *abs*(*u s x / s*)) *n x / n*) ⟶ *real-cond-exp M Invariants* (λx. *abs*(*u s x / s*)) *x*
apply (*rule birkhoff-theorem-AE-nonergodic*) **using** *int-u*[*of s*] **by** *auto*
moreover have *AE x in M*. (λn. *birkhoff-sum F n x / n*) ⟶ *real-cond-exp M Invariants F x*
by (*rule birkhoff-theorem-AE-nonergodic*[*OF int-F*])
moreover have *AE x in M*. *real-cond-exp M Invariants F x* + *real-cond-exp M Invariants* (λx. *abs*(*u s x / s*)) *x* = *real-cond-exp M Invariants F2 x*
unfolding *F2-def* **apply** (*rule AE-symmetric*[*OF real-cond-exp-add*]) **using** *int-u*[*of s*] *int-F int-u*[*of s*] **by** *auto*
ultimately have *AE x in M*. *x* ∈ *G* **unfolding** *G-def* **by** *auto*
then have *emeasure M G* = *1* **by** (*simp add*: *emeasure-eq-1-AE*)

Estimation of asymptotic densities of bad times, for points in G. There is a trivial part, named U below, that has to be treated separately because it creates problematic boundary effects.

define *U* **where** *U* = (λx. {n. ∃ l ∈ {n−s<..n}. u n x − u (n−l) x ≤ − d ∗ l})
define *V* **where** *V* = (λx. {n. ∃ l ∈ {k..n−s}. u n x − u (n−l) x ≤ − d ∗ l})

Trivial estimate for $U(x)$: this set is finite for $x \in G$.

have *densU*: *upper-asymptotic-density* (*U x*) = *0* **if** *x* ∈ *G* **for** *x*
proof −
define *C* **where** *C* = *Max* {*abs*(*u m x*) |m. m<s} + d ∗ s
have ∗: *U x* ⊆ {n. u n x ≤ C − d ∗ n}
proof (*auto*)
fix *n* **assume** *n* ∈ *U x*
then obtain *l* **where** *l*: l∈ {n−s <..n} u n x − u (n−l) x ≤ − d ∗ l **unfolding** *U-def* **by** *auto*
define *m* **where** *m* = *n*−*l*
have *m* < *s* **unfolding** *m-def* **using** *l* **by** *auto*
have *u n x* ≤ *u m x* − *d* ∗ *l* **using** *l m-def* **by** *auto*
also have ... ≤ *abs*(*u m x*) − *d* ∗ *n* + *d* ∗ *m* **unfolding** *m-def* **using** *l*
by (*simp add*: *algebra-simps of-nat-diff*)
also have ... ≤ *Max* {*abs*(*u m x*) |m. m<s} − d ∗ n + d ∗ m
using ‹*m* < *s*› **apply** (*auto*) **by** (*rule Max-ge*, *auto*)
also have ... ≤ *Max* {*abs*(*u m x*) |m. m<s} + d ∗ s − d ∗ n
using ‹*m* < *s*› ‹*d*>*0*› **by** *auto*
finally show *u n x* ≤ *C* − *d* ∗ *n*
unfolding *C-def* **by** *auto*
qed
have *eventually* (λn. *u n x / n* > −*d/2*) *sequentially*

apply (*rule order-tendstoD(1)*) **using** ‹$x \in G$› ‹$d>0$› **unfolding** *G-def* **by** *auto*

then obtain N **where** N: $\bigwedge n.\ n \geq N \Longrightarrow u\ n\ x\ /\ n > -\ d/2$
using *eventually-sequentially* **by** *auto*
{
fix n **assume** $*$: $u\ n\ x \leq C - d * n$ $n > N$
then have $n \geq N$ $n > 0$ **by** *auto*
have $2 * u\ n\ x \leq 2 * C - 2 * d * n$ **using** $*$ **by** *auto*
moreover have $2 * u\ n\ x \geq -\ d * n$ **using** N[*OF* ‹$n \geq N$›] ‹$n > 0$› **by** (*simp add*: *divide-simps*)
ultimately have $-\ d * n \leq 2 * C - 2 * d * n$ **by** *auto*
then have $d * n \leq 2 * C$ **by** *auto*
then have $n \leq 2 * C\ /\ d$ **using** ‹$d>0$› **by** (*simp add*: *mult.commute divide-simps*)
}
then have $\{n.\ u\ n\ x \leq C - d * n\} \subseteq \{..max\ (nat\ (floor(2*C/d)))\ N\}$
by (*auto*, *meson le-max-iff-disj le-nat-floor not-le*)
then have *finite* $\{n.\ u\ n\ x \leq C - d * n\}$
using *finite-subset* **by** *blast*
then have *finite* ($U\ x$) **using** $*$ *finite-subset* **by** *blast*
then show *?thesis* **using** *upper-asymptotic-density-finite* **by** *auto*
qed

Main step: control of u along the sequence $ns{+}t$, with a term $-(d/2)Card(V(x)\cap[1, ns+t])$ on the right. Then, after averaging in t, Birkhoff theorem will imply that $Card(V(x) \cap [1, n])$ is suitably small.

define Z **where** $Z = (\lambda t\ n\ x.\ Max\ \{u\ i\ x|i.\ i< s\} + (\sum i<n.\ abs(u\ s\ ((T^{\frown\frown}(i * s + t))x)))$
$+$ *birkhoff-sum* $F\ (n * s + t)\ x - (d/2) * card(V\ x \cap \{1..n * s + t\}))$
have *Main*: $u\ (n * s + t)\ x \leq Z\ t\ n\ x$ **if** $t < s$ **for** $n\ x\ t$
proof (*rule nat-less-induct*[**where** *?n* = n])
fix n **assume** H: $\forall m<n.\ u\ (m * s + t)\ x \leq Z\ t\ m\ x$
consider $n = 0 | n>0 \wedge V\ x \cap \{(n-1) * s+t<..n * s+t\} = \{\} | n>0 \wedge V\ x \cap \{(n-1) * s+t<..n * s+t\} \neq \{\}$ **by** *auto*
then show $u\ (n * s+t)\ x \leq Z\ t\ n\ x$
proof (*cases*)
assume $n = 0$
then have $V\ x \cap \{1..n * s + t\} = \{\}$ **unfolding** *V-def* **using** ‹$t<s$› ‹$k>2* s$› **by** *auto*
then have $*$: $card(V\ x \cap \{1..n * s+t\}) = 0$ **by** *simp*
have $**$: $0 \leq (\sum i<t.\ F\ ((T^{\frown\frown} i)\ x))$ **by** (*rule sum-nonneg*, *simp add*: *F-pos*)
have $u\ (n * s + t)\ x = u\ t\ x$ **using** ‹$n = 0$› **by** *auto*
also have ... $\leq Max\ \{u\ i\ x|i.\ i< s\}$ **by** (*rule Max-ge*, *auto simp add*: ‹$t<s$›)
also have ... $\leq Z\ t\ n\ x$
unfolding *Z-def birkhoff-sum-def* **using** ‹$n = 0$› $*$ $**$ **by** *auto*
finally show *?thesis* **by** *simp*
next
assume A: $n>0 \wedge V\ x \cap \{(n-1) * s+t<..n * s+t\} = \{\}$

then have *n≥1* **by** *simp*
have $n * s + t = (n-1) * s + t + s$ **using** ‹*n≥1*› **by** (*simp add*: *add.commute add.left-commute mult-eq-if*)
have $V\ x \cap \{1..n * s + t\} = V\ x \cap \{1..(n-1) * s + t\} \cup V\ x \cap \{(n-1) * s + t<..n * s + t\}$
using ‹*n≥1*› **by** (*auto*, *simp add*: *mult-eq-if*)
then have ∗: $card(V\ x \cap \{1..n * s+t\}) = card(V\ x \cap \{1..(n-1) * s+t\})$ **using** *A* **by** *auto*

have ∗∗: *birkhoff-sum F* $((n-1) * s + t)\ x \leq$ *birkhoff-sum F* $(n * s + t)\ x$
unfolding *birkhoff-sum-def* **apply** (*rule sum-mono2*)
using ‹$n * s+t = (n-1) * s+t + s$› *F-pos* **by** *auto*

have $(\sum i<n-1.\ abs(u\ s\ ((T^{\frown\frown}(i * s+t))x))) + u\ s\ ((T^{\frown\frown}((n-1) * s+t))\ x)$
$\leq (\sum i<n-1.\ abs(u\ s\ ((T^{\frown\frown}(i * s+t))x))) + abs(u\ s\ ((T^{\frown\frown}((n-1) * s+t))\ x))$ **by** *auto*
also have ... $\leq (\sum i<n.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x)))$
using ‹*n≥1*› *lessThan-Suc-atMost sum.lessThan-Suc*[*of* $\lambda i.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x))$ $n-1$, *symmetric*] **by** *auto*
finally have ∗∗∗: $(\sum i<n-1.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x))) + u\ s\ ((T^{\frown\frown}((n-1) * s+t))\ x) \leq (\sum i<n.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x)))$
by *simp*

have $u\ (n * s+t)\ x = u\ ((n-1) * s+t + s)\ x$
using ‹*n≥1*› **by** (*simp add*: *add.commute add.left-commute mult-eq-if*)
also have ... $\leq u\ ((n-1) * s+t)\ x + u\ s\ ((T^{\frown\frown}((n-1) * s+t))\ x)$
using *subcocycle-ineq*[*OF subu*, *of* $(n-1) * s+t\ s\ x$] **by** *simp*
also have ... $\leq Max\ \{u\ i\ x|i.\ i< s\} + (\sum i<n-1.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x)))$
$+$ *birkhoff-sum F* $((n-1) * s+t)\ x - (d/2) * card(V\ x \cap \{1..(n-1) * s+t\}) + u\ s\ ((T^{\frown\frown}((n-1) * s+t))\ x)$
using *H* ‹*n≥1*› **unfolding** *Z-def* **by** *auto*
also have ... $\leq Max\ \{u\ i\ x|i.\ i< s\} + (\sum i<n.\ abs(u\ s\ ((T^{\frown\frown}(i* s+t))x)))$
$+$ *birkhoff-sum F* $(n * s+t)\ x - (d/2) * card(V\ x \cap \{1..n * s+t\})$
using ∗ ∗∗ ∗∗∗ **by** *auto*
also have ... $\leq Z\ t\ n\ x$ **unfolding** *Z-def* **by** (*auto simp add*: *divide-simps*)
finally show *?thesis* **by** *simp*
next
assume *B*: $n>0 \wedge V\ x \cap \{(n-1) * s+t<..n * s+t\} \neq \{\}$
then have [*simp*]: $n>0\ n\geq 1\ n \neq 0$ **by** *auto*
obtain *m* **where** *m*: $m \in V\ x \cap \{(n-1) * s + t<..n * s + t\}$ **using** *B* **by** *blast*
then obtain *l* **where** *l*: $l \in \{k..m-s\}\ u\ m\ x - u\ (m-l)\ x \leq - d * l$ **unfolding** *V-def* **by** *auto*
then have $m-s>0$ **using** ‹$k>2* s$› **by** *auto*
then have $m-l \geq s$ **using** *l* **by** *auto*
define *p* **where** $p = (m-l-t)\ div\ s$
have *p1*: $m-l \geq p * s + t$
unfolding *p-def* **using** ‹$m-l \geq s$› ‹$s>t$› *minus-mod-eq-div-mult* [*symmetric*, *of* $m - l - t\ s$]

```
    by simp
  have p2: m−l < p∗ s + t + s
    unfolding p-def using ‹m−l ≥ s› ‹s>t›
      div-mult-mod-eq[of m−l−t s] mod-less-divisor[OF ‹s>0›, of m−l−t] by
linarith
  then have l ≥ m − p ∗ s − t −s by auto
  then have l ≥ (n−1) ∗ s + t −p ∗ s − t− s using m by auto
  then have l + 2 ∗ s≥ (n ∗ s + t) − (p ∗ s+t) by (simp add: diff-mult-distrib)
  have (p+1) ∗ s + t ≤ (n−1) ∗ s + t
    using ‹k> 2 ∗ s› m l(1) p1 by (auto simp add: algebra-simps)
  then have p+1 ≤ n−1
    using ‹s>0› by (meson add-le-cancel-right mult-le-cancel2)
  then have p ≤ n−1 p<n by auto
  have (p∗ s + t) + k ≤ (n ∗ s + t)
    using m l(1) p1 by (auto simp add: algebra-simps)
  then have (1::real) ≤ ((n ∗ s + t) − (p∗ s + t)) / k
    using ‹k > 2∗ s› by auto

  have In: u (n ∗ s + t) x ≤ u m x + (∑ i ∈ {(n−1) ∗ s + t..<n ∗ s + t}.
abs(u 1 ((T^^i) x)))
  proof (cases m = n ∗ s + t)
    case True
    have (∑ i ∈ {(n−1) ∗ s+t..<n ∗ s+t}. abs(u 1 ((T^^i) x))) ≥ 0
      by (rule sum-nonneg, auto)
    then show ?thesis using True by auto
  next
    case False
    then have m2: n ∗ s + t − m >0 (n−1) ∗ s+t ≤ m using m by auto
    have birkhoff-sum (u 1) (n ∗ s+t−m) ((T^^m) x) = (∑ i<n ∗ s+t−m. u
1 ((T^^i)((T^^m) x)))
      unfolding birkhoff-sum-def by auto
    also have ... = (∑ i<n ∗ s+t−m. u 1 ((T^^(i+m)) x))
      by (simp add: funpow-add)
    also have ... = (∑ j ∈ {m..<n ∗ s+t}. u 1 ((T^^j) x))
      by (rule sum.reindex-bij-betw, rule bij-betw-byWitness[where ?f' = λi. i
− m], auto)
    also have ... ≤ (∑ j ∈ {m..<n ∗ s+t}. abs(u 1 ((T^^j) x)))
      by (rule sum-mono, auto)
    also have ... ≤ (∑ j ∈ {(n−1) ∗ s+t..<m}. abs(u 1 ((T^^j) x))) + (∑ j
∈ {m..<n ∗ s+t}. abs(u 1 ((T^^j) x)))
      by auto
    also have ... = (∑ j ∈ {(n−1) ∗ s+t..<n ∗ s+t}. abs(u 1 ((T^^j) x)))
      apply (rule sum.atLeastLessThan-concat) using m2 by auto
    finally have ∗: birkhoff-sum (u 1) (n ∗ s+t−m) ((T^^m) x) ≤ (∑ j ∈
{(n−1) ∗ s+t..<n ∗ s+t}. abs(u 1 ((T^^j) x)))
      by auto

    have u (n ∗ s+t) x ≤ u m x + u (n ∗ s+t−m) ((T^^m) x)
      using subcocycle-ineq[OF subu, of m n ∗ s+t−m] m2 by auto
```

also have ... $\leq$ *u m x* + *birkhoff-sum* (*u 1*) (*n* $*$ *s*+*t*−*m*) ((*T*^^*m*) *x*)
using *subcocycle-bounded-by-birkhoff1*[*OF subu* ‹*n* $*$ *s*+*t* − *m* >*0*›, *of* (*T*^^*m*)*x*] **by** *simp*
finally show *?thesis* **using** $*$ **by** *auto*
qed

have *Ip*: *u* (*m*−*l*) *x* $\leq$ *u* (*p*$*$ *s*+*t*) *x* + ($\sum$ *i* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*)))
proof (*cases m*−*l* = *p*$*$ *s*+*t*)
case *True*
have ($\sum$ *i* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*))) $\geq$ *0*
by (*rule sum-nonneg*, *auto*)
then show *?thesis* **using** *True* **by** *auto*
next
case *False*
then have *m*−*l* − (*p*$*$ *s*+*t*) > *0* **using** *p1* **by** *auto*
have *I*: *p* $*$ *s* + *t* + (*m* − *l* − (*p* $*$ *s* + *t*)) = *m* − *l* **using** *p1* **by** *auto*

have *birkhoff-sum* (*u 1*) (*m*−*l* − (*p*$*$ *s*+*t*)) ((*T*^^(*p*$*$ *s*+*t*)) *x*) = ($\sum$ *i*<*m*−*l* − (*p*$*$ *s*+*t*). *u 1* ((*T*^^*i*) ((*T*^^(*p*$*$ *s*+*t*)) *x*)))
unfolding *birkhoff-sum-def* **by** *auto*
also have ... = ($\sum$ *i*<*m*−*l* − (*p*$*$ *s*+*t*). *u 1* ((*T*^^(*i*+*p*$*$ *s*+*t*)) *x*))
by (*simp add*: *funpow-add*)
also have ... = ($\sum$ *j* ∈ {*p*$*$ *s*+*t*..<*m*−*l*}. *u 1* ((*T*^^*j*) *x*))
by (*rule sum.reindex-bij-betw*, *rule bij-betw-byWitness*[**where** *?f'* = λ*i*. *i* − (*p*$*$ *s*+*t*)], *auto*)
also have ... $\leq$ ($\sum$ *j* ∈ {*p*$*$ *s*+*t*..<*m*−*l*}. *abs*(*u 1* ((*T*^^*j*) *x*)))
by (*rule sum-mono*, *auto*)
also have ... $\leq$ ($\sum$ *j* ∈ {*p*$*$ *s*+*t*..<*m*−*l*}. *abs*(*u 1* ((*T*^^*j*) *x*))) + ($\sum$ *j* ∈ {*m*−*l*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*j*) *x*)))
by *auto*
also have ... = ($\sum$ *j* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*j*) *x*)))
apply (*rule sum.atLeastLessThan-concat*) **using** *p1 p2* **by** *auto*
finally have $*$: *birkhoff-sum* (*u 1*) (*m*−*l* − (*p*$*$ *s*+*t*)) ((*T*^^(*p*$*$ *s*+*t*)) *x*)
$\leq$ ($\sum$ *j* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*j*) *x*)))
by *auto*

have *u* (*m*−*l*) *x* $\leq$ *u* (*p*$*$ *s*+*t*) *x* + *u* (*m*−*l* − (*p*$*$ *s*+*t*)) ((*T*^^(*p*$*$ *s*+*t*)) *x*)
using *subcocycle-ineq*[*OF subu*, *of p*$*$ *s*+*t m*−*l* − (*p*$*$ *s*+*t*) *x*] *I* **by** *auto*
also have ... $\leq$ *u* (*p*$*$ *s*+*t*) *x* + *birkhoff-sum* (*u 1*) (*m*−*l* − (*p*$*$ *s*+*t*)) ((*T*^^(*p*$*$ *s*+*t*)) *x*)
using *subcocycle-bounded-by-birkhoff1*[*OF subu* ‹*m*−*l* − (*p*$*$ *s*+*t*) > *0*›, *of* (*T*^^(*p*$*$ *s*+*t*)) *x*] **by** *simp*
finally show *?thesis* **using** $*$ **by** *auto*
qed

have ($\sum$ *i* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*))) $\leq$ ($\sum$ *i* ∈ {*p*$*$ *s*+*t*..<(*p*+*1*)$*$ *s*+*t*}. *K* + *F* ((*T*^^*i*) *x*))
apply (*rule sum-mono*) **using** *u1-bound* **by** *auto*

moreover have (∑ *i* ∈ {(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*))) ≤ (∑ *i* ∈ {(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *K* + *F* ((*T*^^*i*) *x*))
apply (*rule sum-mono*) **using** *u1-bound* **by** *auto*
ultimately have (∑ *i* ∈ {*p*∗ *s*+*t*..<(*p*+*1*)∗ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*))) + (∑ *i* ∈ {(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*)))
≤ (∑ *i* ∈ {*p*∗ *s*+*t*..<(*p*+*1*)∗ *s*+*t*}. *K* + *F* ((*T*^^*i*) *x*)) + (∑ *i* ∈ {(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *K* + *F* ((*T*^^*i*) *x*))
by *auto*
also have ... = *2*∗ *K*∗ *s* + (∑ *i* ∈ {*p*∗ *s*+*t*..<(*p*+*1*)∗ *s*+*t*}. *F* ((*T*^^*i*) *x*)) + (∑ *i* ∈{(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*))
by (*auto simp add*: *mult-eq-if sum.distrib*)
also have ... ≤ *2*∗ *K* ∗ *s* + (∑ *i* ∈ {*p*∗ *s*+*t*..<(*n*−*1*) ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*)) + (∑ *i* ∈{(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*))
apply (*auto*, *rule sum-mono2*) **using** ‹(*p*+*1*)∗ *s*+*t*≤(*n*−*1*) ∗ *s*+*t*› *F-pos* **by** *auto*
also have ... = *2*∗ *K* ∗ *s* + (∑ *i* ∈ {*p*∗ *s*+*t*..<*n* ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*))
apply (*auto*, *rule sum.atLeastLessThan-concat*) **using** ‹*p*≤*n*−*1*› **by** *auto*
finally have *A0*: (∑ *i* ∈ {*p*∗ *s*+*t*..<(*p*+*1*)∗ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*))) + (∑ *i* ∈ {(*n*−*1*) ∗ *s*+*t*..<*n* ∗ *s*+*t*}. *abs*(*u 1* ((*T*^^*i*) *x*)))
≤ *2*∗ *K* ∗ *s* + (∑ *i* ∈ {*p*∗ *s*+*t*..<*n* ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*))
by *simp*

have *card*(*V x* ∩ {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*}) ≤ *card* {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*} **by** (*rule card-mono*, *auto*)
have *2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s* > *0* **using** ‹*K*>*0*› ‹*s*>*0*› ‹*d*>*0*›
by (*metis add-pos-pos mult-2 mult-zero-left of-nat-0-less-iff pos-divide-less-eq times-divide-eq-right*)
then have *2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s* ≤ ((*n* ∗ *s* + *t*) − (*p*∗ *s* + *t*)) ∗ ((*2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s*) / *k*)
using ‹*1* ≤ ((*n* ∗ *s* + *t*) − (*p*∗ *s* + *t*)) / *k*› **by** (*simp add*: *le-divide-eq-1 pos-le-divide-eq*)
also have ... ≤ ((*n* ∗ *s* + *t*) − (*p*∗ *s* + *t*)) ∗ (*d*/*2*)
apply (*rule mult-left-mono*) **using** ‹(*2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s*)/*k* ≤ *d*/*2*› **by** *auto*
finally have *2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s* ≤ ((*n* ∗ *s* + *t*) − (*p*∗ *s* + *t*)) ∗ (*d*/*2*)
by *auto*
then have −*d* ∗ ((*n* ∗ *s*+*t*) − (*p*∗ *s*+*t*)) + *2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s* ≤ −*d* ∗ ((*n* ∗ *s*+*t*) − (*p*∗ *s*+*t*)) + ((*n* ∗ *s* + *t*) − (*p*∗ *s* + *t*)) ∗ (*d*/*2*)
by *linarith*
also have ... = (−*d*/*2*) ∗ *card* {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*}
by *auto*
also have ... ≤ (−*d*/*2*) ∗ *card*(*V x* ∩ {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*})
using ‹*card*(*V x* ∩ {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*}) ≤ *card* {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*}› **by** *auto*
finally have *A1*: −*d* ∗ ((*n* ∗ *s*+*t*) − (*p*∗ *s*+*t*)) + *2* ∗ *d* ∗ *s* + *2* ∗ *K* ∗ *s* ≤ (−*d*/*2*) ∗ *card*(*V x* ∩ {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*})
by *simp*

have *V x* ∩ {*1*.. *n* ∗ *s*+*t*} = *V x* ∩ {*1*..*p* ∗ *s* + *t*} ∪ *V x* ∩ {*p* ∗ *s* + *t*<.. *n*

$*$ $s+t\}$
using ‹$p * s + t + k \leq n * s + t$› **by** *auto*
then have $card\ (V\ x \cap \{1..\ n * s+t\}) = card(V\ x \cap \{1..p * s + t\} \cup V\ x \cap \{p * s + t<..\ n * s+t\})$
by *auto*
also have ... $= card\ (V\ x \cap \{1..p * s + t\}) + card\ (V\ x \cap \{p * s + t<..\ n * s+t\})$
by (*rule card-Un-disjoint*, *auto*)
finally have *A2*: $card\ (V\ x \cap \{1..p * s + t\}) + card\ (V\ x \cap \{p * s + t<..\ n * s+t\}) = card\ (V\ x \cap \{1..\ n * s+t\})$
by *simp*

have *A3*: $(\sum i<p.\ abs(u\ s\ ((T\ \hat{}\hat{}\ (i * s + t))\ x))) \leq (\sum i<n.\ abs(u\ s\ ((T\ \hat{}\hat{}\ (i * s + t))\ x)))$
apply (*rule sum-mono2*) **using** ‹$p \leq n-1$› **by** *auto*

have *A4*: $birkhoff\text{-}sum\ F\ (p * s + t)\ x + (\sum i \in \{p* s+t..<n * s+t\}.\ F\ ((T\hat{}\hat{}i)\ x)) = birkhoff\text{-}sum\ F\ (n * s + t)\ x$
unfolding *birkhoff-sum-def* **apply** (*subst atLeast0LessThan*[*symmetric*])+
apply (*rule sum.atLeastLessThan-concat*)
using ‹$p \leq n-1$› **by** *auto*

have $u\ (n * s+t)\ x \leq u\ m\ x + (\sum i \in \{(n-1) * s+t..<n * s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x)))$
using *In* **by** *simp*
also have ... $\leq (u\ m\ x - u\ (m-l)\ x) + u\ (m-l)\ x + (\sum i \in \{(n-1) * s+t..<n * s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x)))$
by *simp*
also have ... $\leq - d * l + u\ (p* s+t)\ x + (\sum i \in \{p* s+t..<(p+1)* s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x))) + (\sum i \in \{(n-1) * s+t..<n * s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x)))$
using *Ip l* **by** *auto*
also have ... $\leq - d * ((n * s+t) - (p* s+t)) + 2*d* s + u\ (p* s+t)\ x + (\sum i \in \{p* s+t..<(p+1)* s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x))) + (\sum i \in \{(n-1) * s+t..<n * s+t\}.\ abs(u\ 1\ ((T\hat{}\hat{}i)\ x)))$
using ‹$l + 2* s \geq (n * s+t) - (p* s+t)$› **apply** (*auto simp add*: *algebra-simps*)
by (*metis assms*(*1*) *distrib-left mult.commute mult-2 of-nat-add of-nat-le-iff mult-le-cancel-left-pos*)
also have ... $\leq -d * ((n * s+t) - (p* s+t)) + 2*d* s + Z\ t\ p\ x + 2* K * s + (\sum i \in \{p* s+t..<n * s+t\}.\ F\ ((T\hat{}\hat{}i)\ x))$
using *A0 H* ‹$p<n$› **by** *auto*
also have ... $\leq Z\ t\ p\ x - d/2 * card(V\ x \cap \{p * s + t<..\ n * s+t\}) + (\sum i \in \{p* s+t..<n * s+t\}.\ F\ ((T\hat{}\hat{}i)\ x))$
using *A1* **by** *auto*
also have ... $= Max\ \{u\ i\ x\ |i.\ i < s\} + (\sum i<p.\ abs(u\ s\ ((T\ \hat{}\hat{}\ (i * s + t))\ x))) + birkhoff\text{-}sum\ F\ (p * s + t)\ x$
$- d\ /\ 2 * card\ (V\ x \cap \{1..p * s + t\}) - d/2 * card(V\ x \cap \{p * s + t<..\ n * s+t\}) + (\sum i \in \{p* s+t..<n * s+t\}.\ F\ ((T\hat{}\hat{}i)\ x))$
unfolding *Z-def* **by** *auto*
also have ... $\leq Max\ \{u\ i\ x\ |i.\ i < s\} + (\sum i<n.\ abs(u\ s\ ((T\ \hat{}\hat{}\ (i * s + t))$

x)))
+ (*birkhoff-sum F* (*p* ∗ *s* + *t*) *x* + (∑ *i* ∈ {*p*∗ *s*+*t*..<*n* ∗ *s*+*t*}. *F* ((*T*^^*i*) *x*)))
− *d/2* ∗ *card* (*V x* ∩ {*1*..*p* ∗ *s* + *t*}) − *d/2* ∗ *card*(*V x* ∩ {*p* ∗ *s* + *t*<.. *n* ∗ *s*+*t*})
using *A3* **by** *auto*
also have ... = *Z t n x*
unfolding *Z-def* **using** *A2 A4* **by** (*auto simp add*: *algebra-simps*, *metis distrib-left of-nat-add*)
finally show *?thesis* **by** *simp*
qed
qed

have *Main2*: (*d/2*) ∗ *card*(*V x* ∩ {*1*..*n*}) ≤ *Max* {*u i x*|*i*. *i*< *s*} + *birkhoff-sum* (λ*x*. *abs*(*u s x*/ *s*)) (*n*+*2*∗ *s*) *x*
+ *birkhoff-sum F* (*n* + *2* ∗ *s*) *x* + (*1/s*) ∗ (∑ *i*< *2* ∗ *s*. *abs*(*u* (*n*+*i*) *x*)) **for** *n x*
proof −
define *N* **where** *N* = (*n div s*) + *1*
then have *n* ≤ *N* ∗ *s*
using ‹*s* > *0*› *dividend-less-div-times less-or-eq-imp-le* **by** *auto*
have *N* ∗ *s* ≤ *n* + *s*
by (*auto simp add*: *N-def*)
have *eq-t*: (*d/2*) ∗ *card*(*V x* ∩ {*1*..*n*}) ≤ *abs*(*u*(*N*∗ *s*+*t*) *x*) + (*Max* {*u i x*|*i*. *i*< *s*} + *birkhoff-sum F* (*n* + *2*∗ *s*) *x*)
+ (∑ *i*<*N*. *abs*(*u s* ((*T*^^(*i* ∗ *s*+*t*))*x*)))
if *t*<*s* **for** *t*
proof −
have ∗: *birkhoff-sum F* (*N* ∗ *s*+*t*) *x* ≤ *birkhoff-sum F* (*n*+ *2*∗ *s*) *x*
unfolding *birkhoff-sum-def* **apply** (*rule sum-mono2*) **using** *F-pos* ‹*N* ∗ *s* ≤ *n* + *s*› ‹*t*<*s*› **by** *auto*

have *card*(*V x* ∩ {*1*..*n*}) ≤ *card*(*V x* ∩ {*1*..*N*∗ *s*+*t*})
apply (*rule card-mono*) **using** ‹*n* ≤ *N* ∗ *s*› **by** *auto*
then have (*d/2*) ∗ *card*(*V x* ∩ {*1*..*n*}) ≤ (*d/2*) ∗ *card*(*V x* ∩ {*1*..*N*∗ *s*+*t*})
by *auto*
also have ... ≤ − *u* (*N*∗ *s*+*t*) *x* + *Max* {*u i x*|*i*. *i*< *s*} + (∑ *i*<*N*. *abs*(*u s* ((*T*^^(*i*∗ *s*+*t*))*x*))) + *birkhoff-sum F* (*N* ∗ *s*+*t*) *x*
using *Main*[*OF* ‹*t* < *s*›, *of N x*] **unfolding** *Z-def* **by** *auto*
also have ... ≤ *abs*(*u*(*N*∗ *s*+*t*) *x*) + *Max* {*u i x*|*i*. *i*< *s*} + *birkhoff-sum F* (*n* + *2*∗ *s*) *x* + (∑ *i*<*N*. *abs*(*u s* ((*T*^^(*i*∗ *s*+*t*))*x*)))
using ∗ **by** *auto*
finally show *?thesis* **by** *simp*
qed

have (∑ *t*<*s*. *abs*(*u*(*N*∗ *s*+*t*) *x*)) = (∑ *i*∈{*N*∗ *s*..<*N*∗ *s*+*s*}. *abs* (*u i x*))
by (*rule sum.reindex-bij-betw*, *rule bij-betw-byWitness*[**where** *?f'* = λ*i*. *i* − *N*∗ *s*], *auto*)
also have ... ≤ (∑ *i*∈{*n*..<*n* + *2*∗ *s*}. *abs* (*u i x*))

apply (*rule sum-mono2*) **using** ‹$n \leq N * s$› ‹$N * s \leq n + s$› **by** *auto*
also have ... = ($\sum i<2* s.$ *abs* (*u* (*n+i*) *x*))
by (*rule sum.reindex-bij-betw*[*symmetric*], *rule bij-betw-byWitness*[**where** *?f′* = $\lambda i.\ i - n$], *auto*)
finally have ∗∗: ($\sum t<s.$ *abs*(*u*(*N∗ s+t*) *x*)) ≤ ($\sum i<2* s.$ *abs* (*u* (*n+i*) *x*))
by *simp*

have ($\sum t<s.$ ($\sum i<N.$ *abs*(*u s* ((*T*^^(*i∗ s+t*))*x*)))) = ($\sum i<N* s.$ *abs*(*u s* ((*T*^^*i*) *x*)))
by (*rule sum-arith-progression*)
also have ... ≤ ($\sum i<n + 2* s.$ *abs*(*u s* ((*T*^^*i*) *x*)))
apply (*rule sum-mono2*) **using** ‹$N * s \leq n + s$› **by** *auto*
finally have ∗∗∗: ($\sum t<s.$ ($\sum i<N.$ *abs*(*u s* ((*T*^^(*i∗ s+t*))*x*)))) ≤ *s* ∗ *birkhoff-sum* ($\lambda x.$ *abs*(*u s x/ s*)) (*n+2∗ s*) *x*
unfolding *birkhoff-sum-def* **using** ‹*s>0*› **by** (*auto simp add*: *sum-divide-distrib*[*symmetric*])

have ∗∗∗∗: *s* ∗ ($\sum i<n + 2* s.$ *abs*(*u s* ((*T*^^*i*) *x*)) /*s*) = ($\sum i<n + 2* s.$ *abs*(*u s* ((*T*^^*i*) *x*)))
by (*auto simp add*: *sum-divide-distrib*[*symmetric*])

have *s* ∗ (*d/2*) ∗ *card*(*V x* ∩ {*1..n*}) = ($\sum t<s.$ (*d/2*) ∗ *card*(*V x* ∩ {*1..n*}))
by *auto*
also have ... ≤ ($\sum t<s.$ *abs*(*u*(*N∗ s+t*) *x*) + (*Max* {*u i x*|*i. i< s*} + *birkhoff-sum F* (*n + 2∗ s*) *x*)
+ ($\sum i<N.$ *abs*(*u s* ((*T*^^(*i∗ s+t*))*x*))))
apply (*rule sum-mono*) **using** *eq-t* **by** *auto*
also have ... = ($\sum t<s.$ *abs*(*u*(*N∗ s+t*) *x*)) + ($\sum t<s.$ *Max* {*u i x*|*i. i< s*} + *birkhoff-sum F* (*n + 2∗ s*) *x*) + ($\sum t<s.$ ($\sum i<N.$ *abs*(*u s* ((*T*^^(*i∗ s+t*))*x*))))
by (*auto simp add*: *sum.distrib*)
also have ... ≤ ($\sum i<2* s.$ *abs* (*u* (*n+i*) *x*)) + *s* ∗ (*Max* {*u i x*|*i. i< s*} + *birkhoff-sum F* (*n + 2∗ s*) *x*) + *s* ∗ *birkhoff-sum* ($\lambda x.$ *abs*(*u s x/ s*)) (*n+2∗ s*) *x*
using ∗∗ ∗∗∗ **by** *auto*
also have ... = *s* ∗ ((*1/s*) ∗ ($\sum i<2* s.$ *abs* (*u* (*n+i*) *x*)) + *Max* {*u i x*|*i. i< s*} + *birkhoff-sum F* (*n + 2∗ s*) *x* + *birkhoff-sum* ($\lambda x.$ *abs*(*u s x/ s*)) (*n+2∗ s*) *x*)
by (*auto simp add*: *divide-simps mult.commute distrib-left*)
finally show *?thesis*
by *auto*
qed

have *densV*: *upper-asymptotic-density* (*V x*) ≤ (*2/d*) ∗ *real-cond-exp M Invariants F2 x* **if** $x \in G$ **for** *x*
proof −
have ∗: ($\lambda n.$ *abs*(*u n x/n*)) ⟶ *0*
apply (*rule tendsto-rabs-zero*) **using** ‹$x \in G$› **unfolding** *G-def* **by** *auto*

define *Bound* **where** *Bound* = ($\lambda n.$ (*Max* {*u i x*|*i. i< s*}∗(*1/n*) + *birkhoff-sum* ($\lambda x.$ *abs*(*u s x/ s*)) (*n+2∗ s*) *x / n*
+ *birkhoff-sum F* (*n + 2∗ s*) *x / n* + (*1/s*) ∗ ($\sum i<2* s.$ *abs*(*u* (*n+i*) *x*) / *n*)))

have *Bound* ⟶ (*Max* {*u i x*|*i. i*< *s*} ∗ *0* + *real-cond-exp M Invariants* (*λx. abs*(*u s x*/*s*)) *x*
+ *real-cond-exp M Invariants F x* + (*1*/*s*) ∗ (∑ *i* < *2* ∗ *s. 0*))
unfolding *Bound-def* **apply** (*intro tendsto-intros*)
using ‹*x*∈*G*› ∗ **unfolding** *G-def* **by** *auto*
moreover have *real-cond-exp M Invariants* (*λx. abs*(*u s x*/*s*)) *x* + *real-cond-exp M Invariants F x* = *real-cond-exp M Invariants F2 x*
using ‹*x* ∈ *G*› **unfolding** *G-def* **by** *auto*
ultimately have *B-conv*: *Bound* ⟶ *real-cond-exp M Invariants F2 x* **by** *simp*

have ∗: (*d*/*2*) ∗ *card*(*V x* ∩ {*1..n*}) / *n* ≤ *Bound n* **for** *n*
proof −
have (*d*/*2*) ∗ *card*(*V x* ∩ {*1..n*}) / *n* ≤ (*Max* {*u i x*|*i. i*< *s*} + *birkhoff-sum* (*λx. abs*(*u s x*/ *s*)) (*n*+*2*∗ *s*) *x*
+ *birkhoff-sum F* (*n* + *2*∗ *s*) *x* + (*1*/*s*) ∗ (∑ *i*<*2*∗ *s. abs*(*u* (*n*+*i*) *x*)))/*n*
using *Main2*[*of x n*] **using** *divide-right-mono of-nat-0-le-iff* **by** *blast*
also have ... = *Bound n*
unfolding *Bound-def* **by** (*auto simp add*: *add-divide-distrib sum-divide-distrib*[*symmetric*])
finally show *?thesis* **by** *simp*
qed

have *ereal*(*d*/*2* ∗ *upper-asymptotic-density* (*V x*)) = *ereal*(*d*/*2*) ∗ *ereal*(*upper-asymptotic-density* (*V x*))
by *auto*
also have ... = *ereal* (*d*/*2*) ∗ *limsup*(*λn. card*(*V x* ∩ {*1..n*}) / *n*)
using *upper-asymptotic-density-shift*[*of V x 1 0*] **by** *auto*
also have ... = *limsup*(*λn. ereal* (*d*/*2*) ∗ (*card*(*V x* ∩ {*1..n*}) / *n*))
by (*rule limsup-ereal-mult-left*[*symmetric*], *auto*)
also have ... ≤ *limsup Bound*
apply (*rule Limsup-mono*) **using** ∗ *not-eventuallyD* **by** *auto*
also have ... = *ereal*(*real-cond-exp M Invariants F2 x*)
using *B-conv convergent-limsup-cl convergent-def convergent-real-imp-convergent-ereal limI* **by** *force*
finally have *d*/*2* ∗ *upper-asymptotic-density* (*V x*) ≤ *real-cond-exp M Invariants F2 x*
by *auto*
then show *?thesis*
by (*simp add*: *divide-simps mult.commute*)
qed

define *epsilon* **where** *epsilon* = *2* ∗ *rho* / *d*
have [*simp*]: *epsilon* > *0 epsilon* ≠ *0 epsilon* ≥ *0* **unfolding** *epsilon-def* **by** *auto*
have *emeasure M* {*x*∈*space M. real-cond-exp M Invariants F2 x* ≥ *epsilon*} ≤ (*1*/*epsilon*) ∗ (∫ *x. real-cond-exp M Invariants F2 x* ∂*M*)
apply (*intro integral-Markov-inequality real-cond-exp-pos real-cond-exp-int*(*1*))
by (*auto simp add*: *int-F2 F2-pos*)
also have ... = (*1*/*epsilon*) ∗ (∫ *x. F2 x* ∂*M*)
apply (*intro arg-cong*[**where** *f* = *ennreal*])

by (*simp*, *rule real-cond-exp-int*(*2*), *simp add*: *int-F2*)
also have ... < (*1*/*epsilon*) ∗ *2* ∗ *rho*
using *F2-int* **by** (*intro ennreal-lessI*) (*auto simp add*: *divide-simps*)
also have ... = *d*
unfolding *epsilon-def* **by** *auto*
finally have ∗: *emeasure M* {*x*∈*space M*. *real-cond-exp M Invariants F2 x* ≥ *epsilon*} < *d*
by *simp*

define *G2* **where** *G2* = {*x* ∈ *G*. *real-cond-exp M Invariants F2 x* < *epsilon*}
have [*measurable*]: *G2* ∈ *sets M* **unfolding** *G2-def* **by** *simp*
have *1* = *emeasure M G*
using ‹*emeasure M G* = *1*› **by** *simp*
also have ... ≤ *emeasure M* (*G2* ∪ {*x*∈*space M*. *real-cond-exp M Invariants F2 x* ≥ *epsilon*})
apply (*rule emeasure-mono*) **unfolding** *G2-def* **using** *sets.sets-into-space*[*OF* ‹*G* ∈ *sets M*›] **by** *auto*
also have ... ≤ *emeasure M G2* + *emeasure M* {*x*∈*space M*. *real-cond-exp M Invariants F2 x* ≥ *epsilon*}
by (*rule emeasure-subadditive*, *auto*)
also have ... < *emeasure M G2* + *d*
using ∗ **by** *auto*
finally have *1* − *d* < *emeasure M G2*
using *emeasure-eq-measure* ‹*d* ≤ *1*› **by** (*auto intro*!: *ennreal-less-iff*[*THEN iffD2*] *simp del*: *ennreal-plus simp add*: *ennreal-plus*[*symmetric*])

have *upper-asymptotic-density* {*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *d* ∗ *l*} < *d*
if *x* ∈ *G2* **for** *x*
proof −
have *x* ∈ *G* **using** ‹*x* ∈ *G2*› **unfolding** *G2-def* **by** *auto*
have {*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *d* ∗ *l*} ⊆ *U x* ∪ *V x*
unfolding *U-def V-def* **by** *fastforce*
then have *upper-asymptotic-density* {*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *d* ∗ *l*} ≤ *upper-asymptotic-density* (*U x* ∪ *V x*)
by (*rule upper-asymptotic-density-subset*)
also have ... ≤ *upper-asymptotic-density* (*U x*) + *upper-asymptotic-density* (*V x*)
by (*rule upper-asymptotic-density-union*)
also have ... ≤ (*2*/*d*) ∗ *real-cond-exp M Invariants F2 x*
using *densU*[*OF* ‹*x* ∈ *G*›] *densV*[*OF* ‹*x* ∈ *G*›] **by** *auto*
also have ... < (*2*/*d*) ∗ *epsilon*
using ‹*x* ∈ *G2*› **unfolding** *G2-def* **by** (*simp add*: *divide-simps*)

This is where the choice of ρ at the beginning of the proof is relevant: we choose it so that the above term is at most d.

also have ... = *d* **unfolding** *epsilon-def rho-def* **by** *auto*
finally show *?thesis* **by** *simp*
qed

then have G2 ⊆ {x ∈ space M. upper-asymptotic-density {n. ∃ l ∈ {k..n}. u n x − u (n−l) x ≤ − d ∗ l} < d}
  using sets.sets-into-space[OF ‹G2 ∈ sets M›] **by** blast
then have emeasure M G2 ≤ emeasure M {x ∈ space M. upper-asymptotic-density {n. ∃ l ∈ {k..n}. u n x − u (n−l) x ≤ − d ∗ l} < d}
  by (rule emeasure-mono, auto)
then have emeasure M {x ∈ space M. upper-asymptotic-density {n. ∃ l ∈ {k..n}. u n x − u (n−l) x ≤ − d ∗ l} < d} > 1 −d
  using ‹emeasure M G2 > 1 − d› **by** auto
then show ?thesis **by** blast
qed

The two previous lemmas are put together in the following lemma, corresponding to Lemma 2.3 in [GK15].

lemma upper-density-delta:
  fixes d::real
  assumes d > 0 d ≤ 1
  shows ∃ delta::nat⇒real. (∀ l. delta l > 0) ∧ (delta ⟶ 0) ∧
    emeasure M {x ∈ space M. ∀ (N::nat). card {n ∈{..<N}. ∃ l ∈ {1..n}. u n x − u (n−l) x ≤ − delta l ∗ l} ≤ d ∗ N} > 1 − d
proof −
  define d2 **where** d2 = d/2
  have [simp]: d2 > 0 **unfolding** d2-def **using** assms **by** simp
  then have ¬ d2 < 0 **using** not-less [of d2 0] **by** (simp add: less-le)
  have d2/2 > 0 **by** simp
  obtain c0 **where** c0: c0> (0::real) emeasure M {x ∈ space M. upper-asymptotic-density {n. ∃ l ∈ {1..n}. u n x − u (n−l) x ≤ − c0 ∗ l} < d2/2} > 1 − (d2/2)
    using upper-density-all-times[OF ‹d2/2 > 0›] **by** blast
  have ∃ N. emeasure M {x ∈ space M. ∀ n ≥ N. card ({n. ∃ l ∈ {1..n}. u n x − u (n−l) x ≤ − c0 ∗ l} ∩ {..<n}) < (d2/2) ∗ n} > 1 − (d2/2)
    apply (rule upper-density-eventually-measure) **using** c0(2) **by** auto
  then obtain N1 **where** N1: emeasure M {x ∈ space M. ∀ B ≥ N1. card ({n. ∃ l ∈ {1..n}. u n x − u (n−l) x ≤ − c0 ∗ l} ∩ {..<B}) < (d2/2) ∗ B} > 1 − (d2/2)
    by blast
  define O1 **where** O1 = {x ∈ space M. ∀ B ≥ N1. card ({n. ∃ l ∈ {1..n}. u n x − u (n−l) x ≤ − c0 ∗ l} ∩ {..<B}) < (d2/2) ∗ B}
  have [measurable]: O1 ∈ sets M **unfolding** O1-def **by** auto
  have emeasure M O1 > 1 − (d2/2) **unfolding** O1-def **using** N1 **by** auto

  have ∗: ∃ N. emeasure M {x ∈ space M. ∀ B ≥ N. card({n. ∃ l ∈ {N..n}. u n x − u (n−l) x ≤ − e ∗ l} ∩ {..<B}) < e ∗ B} > 1 − e
    if e>0 e ≤ 1 **for** e::real
  proof −
    obtain k **where** k: emeasure M {x ∈ space M. upper-asymptotic-density {n. ∃ l ∈ {k..n}. u n x − u (n−l) x ≤ − e ∗ l} < e} > 1 − e
      using upper-density-large-k[OF ‹e>0› ‹e ≤ 1›] **by** blast
    then obtain N0 **where** N0: emeasure M {x ∈ space M. ∀ B ≥ N0. card({n. ∃ l ∈ {k..n}. u n x − u (n−l) x ≤ − e ∗ l} ∩ {..<B}) < e ∗ B} > 1 − e

using *upper-density-eventually-measure*[*OF* - *k*] **by** *auto*
define *N* **where** *N* = *max k N0*
have *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *N0*. *card*({*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *e* ∗ *l*} ∩ {..<*B*}) < *e* ∗ *B*}
≤ *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *N*. *card*({*n*. ∃ *l* ∈ {*N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *e* ∗ *l*} ∩ {..<*B*}) < *e* ∗ *B*}
proof (*rule emeasure-mono*, *auto*)
fix *x B* **assume** *H*: *x* ∈ *space M* ∀ *B*≥*N0*. *card* ({*n*. ∃ *l*∈{*k*..*n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*e* ∗ *real l*)} ∩ {..<*B*}) < *e* ∗ *B N* ≤ *B*

have *card*({*n*. ∃ *l* ∈ {*N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*e* ∗ *real l*)} ∩ {..<*B*}) ≤ *card*({*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ −(*e* ∗ *real l*)} ∩ {..<*B*})
unfolding *N-def* **by** (*rule card-mono*, *auto*)
then have *real*(*card*({*n*. ∃ *l* ∈ {*N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*e* ∗ *real l*)} ∩ {..<*B*})) ≤ *card*({*n*. ∃ *l* ∈ {*k*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ −(*e* ∗ *real l*)} ∩ {..<*B*})
by *simp*
also have ... < *e* ∗ *B* **using** *H*(*2*) ‹*B*≥*N*› **unfolding** *N-def* **by** *auto*
finally show *card* ({*n*. ∃ *l*∈{*N*..*n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*e* ∗ *real l*)} ∩ {..<*B*}) < *e* ∗ *B*
by *auto*
qed
then have *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *N*. *card*({*n*. ∃ *l* ∈ {*N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *e* ∗ *l*} ∩ {..<*B*}) < *e* ∗ *B*} > *1* − *e*
using *N0* **by** *simp*
then show *?thesis* **by** *auto*
qed

define *Ne* **where** *Ne* = (λ(*e*::*real*). *SOME N*. *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *N*. *card*({*n*. ∃ *l* ∈ {*N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *e* ∗ *l*} ∩ {..<*B*}) < *e* ∗ *B*} > *1* − *e*)
have *Ne*: *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *Ne e*. *card*({*n*. ∃ *l* ∈ {*Ne e*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *e* ∗ *l*} ∩ {..<*B*}) < *e* ∗ *B*} > *1* − *e*
if *e*>*0 e* ≤ *1* **for** *e*::*real*
unfolding *Ne-def* **by** (*rule someI-ex*[*OF* ∗[*OF that*]])
define *eps* **where** *eps* = (λ(*n*::*nat*). *d2* ∗ (*1*/*2*)^*n*)
have [*simp*]: *eps n* > *0* **for** *n* **unfolding** *eps-def* **by** *auto*
then have [*simp*]: *eps n* ≥ *0* **for** *n* **by** (*rule less-imp-le*)

have *eps n* ≤ (*1* / *2*) ∗ *1* **for** *n*
unfolding *eps-def d2-def*
using ‹*d* ≤ *1*› **by** (*intro mult-mono power-le-one*) *auto*
also have . . . < *1* **by** *auto*
finally have [*simp*]: *eps n* < *1* **for** *n* **by** *simp*
then have [*simp*]: *eps n* ≤ *1* **for** *n* **by** (*rule less-imp-le*)

have (λ*n*. *d2* ∗ (*1*/*2*)^*n*) ⟶ *d2* ∗ *0*
by (*rule tendsto-mult*, *auto simp add*: *LIMSEQ-realpow-zero*)
then have *eps* ⟶ *0* **unfolding** *eps-def* **by** *auto*

define *Nf* **where** *Nf* = (λ*N*. (*if* (*N* = *0*) *then 0*
else if (*N* = *1*) *then N1* + *1*
else max (*N1*+*1*) (*Max* {*Ne*(*eps n*)|*n*. *n* ≤ *N*}) + *N*))
have *Nf N* < *Nf* (*N*+*1*) **for** *N*
proof −
consider *N* = *0* | *N* = *1* | *N*>*1* **by** *fastforce*
then show *?thesis*
proof (*cases*)
assume *N*>*1*
have *Max* {*Ne* (*eps n*) |*n*. *n* ≤ *N*} ≤ *Max* {*Ne* (*eps n*) |*n*. *n* ≤ *Suc N*}
by (*rule Max-mono*, *auto*)
then show *?thesis* **unfolding** *Nf-def* **by** *auto*
qed (*auto simp add*: *Nf-def*)
qed
then have *strict-mono Nf*
using *strict-mono-Suc-iff* **by** *auto*

define *On* **where** *On* = (λ(*N*::*nat*).
(*if* (*N* = *1*) *then O1*
else {*x* ∈ *space M*. ∀ *B* ≥ *Nf N*. *card*({*n*. ∃ *l* ∈ {*Nf N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N*) ∗ *l*} ∩ {..<*B*}) < (*eps N*) ∗ *B*}))
have [*measurable*]: *On N* ∈ *sets M* **for** *N* **unfolding** *On-def* **by** *auto*
have *emeasure M* (*On N*) > *1* − *eps N* **if** *N*>*0* **for** *N*
proof −
consider *N* = *1* | *N*>*1* **using** ‹*N*>*0*› **by** *linarith*
then show *?thesis*
proof (*cases*)
case *1*
then show *?thesis* **unfolding** *On-def eps-def* **using** ‹*emeasure M O1* > *1* − (*d2*/*2*)› **by** *auto*
next
case *2*
have *Ne* (*eps N*) ≤ *Max* {*Ne*(*eps n*)|*n*. *n* ≤ *N*}
by (*rule Max.coboundedI*, *auto*)
also have ... ≤ *Nf N* **unfolding** *Nf-def* **using** ‹*N*>*1*› **by** *auto*
finally have *Ne* (*eps N*) ≤ *Nf N* **by** *simp*
have *1* − *eps N* < *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *Ne*(*eps N*). *card*({*n*. ∃ *l* ∈ {*Ne*(*eps N*)..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N*) ∗ *l*} ∩ {..<*B*}) < (*eps N*) ∗ *B*}
by (*rule Ne*) *simp-all*
also have ... ≤ *emeasure M* {*x* ∈ *space M*. ∀ *B* ≥ *Nf N*. *card*({*n*. ∃ *l* ∈ {*Nf N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N*) ∗ *l*} ∩ {..<*B*}) < (*eps N*) ∗ *B*}
proof (*rule emeasure-mono*, *auto*)
fix *x n* **assume** *H*: *x* ∈ *space M*
∀ *n*≥*Ne* (*eps N*). *card* ({*n*. ∃ *l*∈{*Ne* (*eps N*)..*n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*eps N* ∗ *l*)} ∩ {..<*n*}) < *eps N* ∗ *n*
Nf N ≤ *n*
have *card*({*n*. ∃ *l* ∈ {*Nf N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N* ∗ *l*)} ∩ {..<*n*}) ≤ *card*({*n*. ∃ *l* ∈ {*Ne*(*eps N*)..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ −(*eps N*) ∗ *l*} ∩

{..<n})
apply (*rule card-mono*) **using** ‹*Ne* (*eps N*) ≤ *Nf N*› **by** *auto*
then have *real*(*card*({*n*. ∃ *l* ∈ {*Nf N*..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N* ∗ *l*)} ∩ {..<*n*})) ≤ *card*({*n*. ∃ *l* ∈ {*Ne*(*eps N*)..*n*}. *u n x* − *u* (*n*−*l*) *x* ≤ −(*eps N*) ∗ *l*} ∩ {..<*n*})
by *simp*
also have ... < (*eps N*) ∗ *n* **using** *H*(*2*) ‹*n* ≥ *Nf N*› ‹*Ne* (*eps N*) ≤ *Nf N*› **by** *auto*
finally show *real* (*card* ({*n*. ∃ *l*∈{*Nf N*..*n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*eps N* ∗ *l*)} ∩ {..<*n*})) < *eps N* ∗ *real n*
by *auto*
qed
also have ... = *emeasure M* (*On N*)
unfolding *On-def* **using** ‹*N*>*1*› **by** *auto*
finally show *?thesis* **by** *simp*
qed
qed
then have ∗: *emeasure M* (*On* (*N*+*1*)) > *1* − *eps* (*N*+*1*) **for** *N* **by** *simp*

define *Ogood* **where** *Ogood* = (⋂ *N*. *On* (*N*+*1*))
have [*measurable*]: *Ogood* ∈ *sets M* **unfolding** *Ogood-def* **by** *auto*
have *emeasure M Ogood* ≥ *1* − (∑ *N*. *eps*(*N*+*1*))
unfolding *Ogood-def*
apply (*intro emeasure-intersection*, *auto*)
using ∗ **by** (*auto simp add*: *eps-def summable-mult summable-divide summable-geometric less-imp-le*)
moreover have (∑ *N*. *eps*(*N*+*1*)) = *d2*
unfolding *eps-def* **apply** (*subst suminf-mult*)
using *sums-unique*[*OF power-half-series*, *symmetric*] **by** (*auto intro*!: *summable-divide summable-geometric*)
finally have *emeasure M Ogood* ≥ *1* − *d2* **by** *simp*
then have *emeasure M Ogood* > *1* − *d* **unfolding** *d2-def* **using** ‹*d*>*0*› ‹*d* ≤ *1*›
by (*simp add*: *emeasure-eq-measure field-sum-of-halves ennreal-less-iff*)

have *Ogood-union*: *Ogood* = (⋃ (*K*::*nat*). *Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)})
apply *auto* **using** *sets.sets-into-space*[*OF* ‹*Ogood* ∈ *sets M*›] **apply** *blast*
proof −
fix *x*
define *M* **where** *M* = *Max* {*abs*(*u n x* − *u* (*n*−*l*) *x*)/*l* | *n l*. *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}}
obtain *N*::*nat* **where** *N* > *M* **using** *reals-Archimedean2* **by** *blast*

have *finite* { (*n*, *l*) | *n l*. *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}}
by (*rule finite-subset*[**where** *?B* = {*1*.. *Nf 1*} × {*1*..*Nf 1*}], *auto*)
moreover have {*abs*(*u n x* − *u* (*n*−*l*) *x*)/*l* | *n l*. *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}}
= (λ (*n*, *l*). *abs*(*u n x* − *u* (*n*−*l*) *x*)/*l*)‘ { (*n*, *l*) | *n l*. *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}}

by *auto*
ultimately have *fin*: *finite* {*abs*(*u n x* − *u* (*n*−*l*) *x*)/*l* | *n l*. *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}}
by *auto*
{
fix *n l* **assume** *nl*: *n* ∈ {*1*..*Nf 1*} ∧ *l* ∈ {*1*..*n*}
then have *real l*>*0* **by** *simp*
have *abs*(*u n x* − *u* (*n*−*l*) *x*)/*l* ≤ *M*
unfolding *M-def* **apply** (*rule Max-ge*) **using** *fin nl* **by** *auto*
then have *abs*(*u n x* − *u* (*n*−*l*) *x*)/*l* < *real N* **using** ‹*N*>*M*› **by** *simp*
then have *abs*(*u n x* − *u* (*n*−*l*) *x*)< *real N* ∗ *l* **using** ‹*0* < *real l*› *pos-divide-less-eq* **by** *blast*
then have *u n x* − *u* (*n*−*l*) *x* > − (*real N* ∗ *l*) **by** *simp*
}
then have ∀ *n*∈{*Suc 0*..*Nf* (*Suc 0*)}. ∀ *l*∈{*Suc 0*..*n*}. − (*real N* ∗ *real l*) < *u n x* − *u* (*n* − *l*) *x*
by *auto*
then show ∃ *N*. ∀ *n*∈{*Suc 0*..*Nf* (*Suc 0*)}. ∀ *l*∈{*Suc 0*..*n*}. − (*real N* ∗ *real l*) < *u n x* − *u* (*n* − *l*) *x*
by *auto*
qed
have (λ*K*. *emeasure M* (*Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}))
⟶ *emeasure M* (⋃(*K*::*nat*). *Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)})
apply (*rule Lim-emeasure-incseq*, *auto*)
unfolding *incseq-def* **apply** *auto*
proof −
fix *m n x na l*
assume *m* ≤ (*n*::*nat*) ∀ *n*∈{*Suc 0*..*Nf* (*Suc 0*)}. ∀ *l*∈{*Suc 0*..*n*}. − (*real m* ∗ *real l*) < *u n x* − *u* (*n* − *l*) *x*
Suc 0 ≤ *l l* ≤ *na na* ≤ *Nf* (*Suc 0*)
then have − (*real m* ∗ *real l*) < *u na x* − *u* (*na* − *l*) *x* **by** *auto*
moreover have − (*real n* ∗ *real l*) ≤ − (*real m* ∗ *real l*) **using** ‹*m* ≤ *n*› **by** (*simp add*: *mult-mono*)
ultimately show − (*real n* ∗ *real l*) < *u na x* − *u* (*na* − *l*) *x* **by** *auto*
qed
moreover have *emeasure M* (⋃(*K*::*nat*). *Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}) > *1* − *d*
using *Ogood-union* ‹*emeasure M Ogood* > *1* − *d*› **by** *auto*
ultimately have *a*: *eventually* (λ*K*. *emeasure M* (*Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}) > *1* − *d*) *sequentially*
by (*rule order-tendstoD*(*1*))
have *b*: *eventually* (λ*K*. *K* ≥ *max c0 d2*) *sequentially*
using *eventually-at-top-linorder nat-ceiling-le-eq* **by** *blast*
have *eventually* (λ*K*. *K* ≥ *max c0 d2* ∧ *emeasure M* (*Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1*..*Nf 1*}. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}) > *1* − *d*) *sequentially*
by (*rule eventually-elim2*[*OF a b*], *auto*)

then obtain *K* **where** *K*: *K*≥*max c0 d2 emeasure M* (*Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1..Nf 1*}. ∀ *l* ∈ {*1..n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}) > *1* − *d*
using *eventually-False-sequentially eventually-elim2* **by** *blast*

define *Og* **where** *Og* = *Ogood* ∩ {*x* ∈ *space M*. ∀ *n* ∈ {*1..Nf 1*}. ∀ *l* ∈ {*1..n*}. *u n x* − *u* (*n*−*l*) *x* > − (*real K* ∗ *l*)}
have [*measurable*]: *Og* ∈ *sets M* **unfolding** *Og-def* **by** *simp*
have *emeasure M Og* > *1* − *d* **unfolding** *Og-def* **using** *K* **by** *simp*

have *fin*: *finite* {*N*. *Nf N* ≤ *n*} **for** *n*
using *pseudo-inverse-finite-set*[*OF filterlim-subseq*[*OF* ‹*strict-mono Nf*›]] **by** *auto*
define *prev-N* **where** *prev-N* = (λ*n*. *Max* {*N*. *Nf N* ≤ *n*})
define *delta* **where** *delta* = (λ*l*. *if* (*prev-N l* ≤ *1*) *then K else eps* (*prev-N l*))
have ∀ *l*. *delta l* > *0*
unfolding *delta-def* **using** ‹*K*≥*max c0 d2*› ‹*c0*>*0*› **by** *auto*

have *LIM n sequentially*. *prev-N n* :> *at-top*
unfolding *prev-N-def* **apply** (*rule tendsto-at-top-pseudo-inverse2*)
using ‹*strict-mono Nf*› **by** (*simp add*: *filterlim-subseq*)
then have *eventually* (λ*l*. *prev-N l* > *1*) *sequentially*
by (*simp add*: *filterlim-iff*)
then have *eventually* (λ*l*. *delta l* = *eps*(*prev-N l*)) *sequentially*
unfolding *delta-def* **by** (*simp add*: *eventually-mono*)
moreover have (λ*l*. *eps*(*prev-N l*)) ⟶ *0*
by (*rule filterlim-compose*[*OF* ‹*eps* ⟶ *0*› ‹*LIM n sequentially*. *prev-N n* :> *at-top*›])
ultimately have *delta* ⟶ *0* **by** (*simp add*: *filterlim-cong*)

have *delta n* ≤ *K* **for** *n*
proof −
have ∗: *d2* ∗ (*1* / *2*) ^ *prev-N n* ≤ *real K* ∗ *1*
apply (*rule mult-mono′*) **using** ‹*K* ≥ *max c0 d2*› ‹*d2*>*0*› **by** (*auto simp add*: *power-le-one less-imp-le*)
then show *?thesis* **unfolding** *delta-def* **apply** *auto* **unfolding** *eps-def* **using** ∗ **by** *auto*
qed

define *bad-times* **where** *bad-times* = (λ*x*. {*n* ∈ {*Nf 1*..}. ∃ *l*∈{*1..n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*c0* ∗ *l*)} ∪
(⋃ *N*∈{*2*..}. {*n* ∈ {*Nf N*..}. ∃ *l* ∈ {*Nf N..n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − (*eps N* ∗ *l*)}))
have *card-bad-times*: *card* (*bad-times x* ∩ {..<*B*}) ≤ *d2* ∗ *B* **if** *x* ∈ *Og* **for** *x B*
proof −
have (∑ *N*∈{..<*B*}. (*1*/(*2*::*real*))^*N*) ≤ (∑ *N*. (*1*/*2*)^*N*)
by (*rule sum-le-suminf*, *auto simp add*: *summable-geometric*)
also have ... = *2* **using** *suminf-geometric*[*of 1*/(*2*::*real*)] **by** *auto*
finally have ∗: (∑ *N*∈{..<*B*}. (*1*/(*2*::*real*))^*N*) ≤ *2* **by** *simp*

have $(\sum\ N \in \{2..<B\}.\ eps\ N * B) \leq (\sum\ N \in \{2..<B+2\}.\ eps\ N * B)$
by (*rule sum-mono2*, *auto*)
also have ... $= (\sum N \in \{2..<B+2\}.\ d2 * (1/2)\hat{\ }N * B)$
unfolding *eps-def* **by** *auto*
also have ... $= (\sum N \in \{..<B\}.\ d2 * (1/2)\hat{\ }(N+2) * B)$
by (*rule sum.reindex-bij-betw*[*symmetric*],*rule bij-betw-byWitness*[**where** *?f'* $= \lambda i.\ i-2$], *auto*)
also have ... $= (\sum N \in \{..<B\}.\ (d2 * (1/4) * B) * (1/2)\hat{\ }N)$
by (*auto*, *metis* (*lifting*) *mult.commute mult.left-commute*)
also have ... $= (d2 * (1/4) * B) * (\sum N \in \{..<B\}.\ (1/2)\hat{\ }N)$
by (*rule sum-distrib-left*[*symmetric*])
also have ... $\leq (d2 * (1/4) * B) * 2$
apply (*rule mult-left-mono*) **using** $*$ ‹$d2 > 0$› **apply** *auto*
by (*metis* ‹$0 < d2$› *mult-eq-0-iff mult-le-0-iff not-le of-nat-eq-0-iff of-nat-le-0-iff*)
finally have *I*: $(\sum\ N \in \{2..<B\}.\ eps\ N * B) \leq d2/2 * B$
by *auto*

have $x \in On\ 1$ **using** ‹$x \in Og$› **unfolding** *Og-def Ogood-def* **by** *auto*
then have $x \in O1$ **unfolding** *On-def* **by** *auto*
have *B1*: $real(card(\{n \in \{Nf\ 1..\}.\ \exists l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (c0 * l)\} \cap \{..<B\})) \leq (d2/2) * B$ **for** *B*
proof (*cases* $B \geq N1$)
case *True*
have $card(\{n \in \{Nf\ 1..\}.\ \exists l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (c0 * l)\} \cap \{..<B\})$
$\leq card(\{n.\ \exists l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (c0 * l)\} \cap \{..<B\})$
by (*rule card-mono*, *auto*)
also have ... $\leq (d2/2) * B$
using ‹$x \in O1$› **unfolding** *O1-def* **using** *True* **by** *auto*
finally show *?thesis* **by** *auto*
next
case *False*
then have $B < Nf\ 1$ **unfolding** *Nf-def* **by** *auto*
then have $\{n \in \{Nf\ 1..\}.\ \exists l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (c0 * l)\} \cap \{..<B\} = \{\}$
by *auto*
then have $card\ (\{n \in \{Nf\ 1..\}.\ \exists l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (c0 * l)\} \cap \{..<B\}) = 0$
by *auto*
also have ... $\leq (d2/2) * B$
using ‹$\neg\ d2 < 0$› **by** *simp*
finally show *?thesis* **by** *simp*
qed

have *BN*: $real(card\ (\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})) \leq eps\ N * B$ **if** $N \geq 2$ **for** *N B*
proof −
have $x \in On\ ((N-1) + 1)$ **using** ‹$x \in Og$› **unfolding** *Og-def Ogood-def* **by** *auto*

then have $x \in On\ N$ **using** ‹$N \geq 2$› **by** *auto*
show *?thesis*
proof (*cases* $B \geq Nf\ N$)
case *True*
have *card* $(\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\}) \leq$
card $(\{n.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})$
by (*rule card-mono*, *auto*)
also have ... $\leq eps\ N * B$
using ‹$x \in On\ N$› ‹$N \geq 2$› *True* **unfolding** *On-def* **by** *auto*
finally show *?thesis* **by** *simp*
next
case *False*
then have $\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\} = \{\}$
by *auto*
then have *card* $(\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\}) = 0$
by *auto*
also have ... $\leq eps\ N * B$
by (*metis* ‹$\bigwedge n.\ 0 < eps\ n$› *le-less mult-eq-0-iff mult-pos-pos of-nat-0 of-nat-0-le-iff*)
finally show *?thesis* **by** *simp*
qed
qed

{
fix N **assume** $N \geq B$
have $Nf\ N \geq B$ **using** *seq-suble*[*OF* ‹*strict-mono Nf*›, *of N*] ‹$N \geq B$› **by** *simp*
then have $\{Nf\ N..\} \cap \{..<B\} = \{\}$ **by** *auto*
then have $\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\} = \{\}$ **by** *auto*
}
then have $*$: $(\bigcup N \in \{B..\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\}) = \{\}$
by *auto*

have $\{2..\} \subseteq \{2..<B\} \cup \{B..\}$ **by** *auto*
then have $(\bigcup N \in \{2..\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})$
$\subseteq (\bigcup N \in \{2..<B\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})$
$\cup (\bigcup N \in \{B..\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})$
by *auto*
also have ... $= (\bigcup N \in \{2..<B\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq -\ (eps\ N * l)\} \cap \{..<B\})$
using $*$ **by** *auto*

finally have ∗: *bad-times* $x \cap \{..<B\} \subseteq \{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\}$
$\cup (\bigcup N\in\{2..<B\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\})$
unfolding *bad-times-def* **by** *auto*
have $card(bad\text{-}times\ x \cap \{..<B\}) \leq card(\{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\}$
$\cup (\bigcup N\in\{2..<B\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\}))$
by (*rule card-mono*[*OF* - ∗], *auto*)
also have ... $\leq card(\{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\}) +$
$card\ (\bigcup N\in\{2..<B\}.\ \{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\})$
by (*rule card-Un-le*)
also have ... $\leq card(\{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\}) +$
$(\sum N\in\{2..<B\}.\ card\ (\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\}))$
by (*simp del*: *UN-simps*, *rule card-UN-le*, *auto*)
finally have $real\ (card(bad\text{-}times\ x \cap \{..<B\})) \leq$
$real(card(\{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\})$
$+ (\sum N\in\{2..<B\}.\ card\ (\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\})))$
by (*subst of-nat-le-iff*, *simp*)
also have ... $= real(card(\{n \in \{Nf\ 1..\}.\ \exists l\in\{1..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (c0 * l)\} \cap \{..<B\}))$
$+ (\sum N\in\{2..<B\}.\ real(card\ (\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\})))$
by *auto*
also have ... $\leq (d2/2 * B) + (\sum N\in\{2..<B\}.\ real(card\ (\{n \in \{Nf\ N..\}.\ \exists l \in \{Nf\ N..n\}.\ u\ n\ x - u\ (n-l)\ x \leq - (eps\ N * l)\} \cap \{..<B\})))$
using *B1* **by** *simp*
also have ... $\leq (d2/2 * B) + (\sum N \in \{2..<B\}.\ eps\ N * B)$
apply (*simp*, *rule sum-mono*) **using** *BN* **by** *auto*
also have ... $\leq (d2/2 * B) + (d2/2*B)$
using *I* **by** *auto*
finally show *?thesis* **by** *simp*
qed

have *ineq-on-Og*: $u\ n\ x - u\ (n-l)\ x > - delta\ l * l$ **if** $l \in \{1..n\}$ $n \notin bad\text{-}times\ x$ $x \in Og$ **for** *n x l*
proof −
consider $n < Nf\ 1 \mid n \geq Nf\ 1 \wedge prev\text{-}N\ l \leq 1 \mid n \geq Nf\ 1 \wedge prev\text{-}N\ l \geq 2$ **by** *linarith*
then show *?thesis*
proof (*cases*)
assume $n < Nf\ 1$

```
      then have {N. Nf N ≤ n} = {0}
        apply auto using ‹strict-mono Nf› unfolding strict-mono-def
        apply (metis le-trans less-Suc0 less-imp-le-nat linorder-neqE-nat not-less)
        unfolding Nf-def by auto
      then have prev-N n = 0 unfolding prev-N-def by auto
      moreover have prev-N l ≤ prev-N n
       unfolding prev-N-def apply (rule Max-mono) using ‹l ∈ {1..n}› fin apply
auto
        unfolding Nf-def by auto
      ultimately have prev-N l = 0 using ‹prev-N l ≤ prev-N n› by auto
      then have delta l = K unfolding delta-def by auto
      have 1 ∉ {N. Nf N ≤ n} using fin[of n]
       by (metis (full-types) Max-ge ‹prev-N n = 0› fin not-one-le-zero prev-N-def)
      then have n < Nf 1 by auto
      moreover have n ≥ 1 using ‹l ∈ {1..n}› by auto
      ultimately have n ∈ {1..Nf 1} by auto
      then have u n x − u (n−l) x > − (real K ∗ l) using ‹x ∈ Og› unfolding
Og-def using ‹l ∈ {1..n}› by auto
      then show ?thesis using ‹delta l = K› by auto
    next
      assume H: n ≥ Nf 1 ∧ prev-N l ≤ 1
      then have delta l = K unfolding delta-def by auto
      have n ∉ {n ∈ {Nf 1..}. ∃ l∈{1..n}. u n x − u (n−l) x ≤ − (c0 ∗ l)}
        using ‹n ∉ bad-times x› unfolding bad-times-def by auto
      then have u n x − u (n−l) x > − (c0 ∗ l)
        using H ‹l ∈ {1..n}› by force
       moreover have − (c0 ∗ l) ≥ − (real K ∗ l) using K(1) by (simp add:
mult-mono)
      ultimately show ?thesis using ‹delta l = K› by auto
    next
      assume H: n ≥ Nf 1 ∧ prev-N l ≥ 2
      define N where N = prev-N l
      have N ≥ 2 unfolding N-def using H by auto
      have prev-N l ∈ {N. Nf N ≤ l}
        unfolding prev-N-def apply (rule Max-in, auto simp add: fin)
        unfolding Nf-def by auto
      then have Nf N ≤ l unfolding N-def by auto
      then have Nf N ≤ n using ‹l ∈ {1..n}› by auto
      have n ∉ {n ∈ {Nf N..}. ∃ l ∈ {Nf N..n}. u n x − u (n−l) x ≤ − (eps N ∗
l)}
        using ‹n ∉ bad-times x› ‹N≥2› unfolding bad-times-def by auto
      then have u n x − u (n−l) x > − (eps N ∗ l)
        using ‹Nf N ≤ n› ‹Nf N ≤ l› ‹l ∈ {1..n}› by force
     moreover have eps N = delta l unfolding delta-def N-def using H by auto
      ultimately show ?thesis by auto
    qed
  qed

  have Og ⊆ {x ∈ space M. ∀ (B::nat). card {n ∈{..<B}. ∃ l ∈ {1..n}. u n x − u
```

(n−l) x ≤ − delta l ∗ l} ≤ d ∗ B}
proof (*auto*)
fix *x* **assume** *x* ∈ *Og*
then show *x* ∈ *space M* **unfolding** *Og-def* **by** *auto*
next
fix *x B* **assume** *x* ∈ *Og*
have ∗: {*n. n* < *B* ∧ (∃ *l*∈{*Suc 0..n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*delta l* ∗ *real l*))} ⊆ *bad-times x* ∩ {..<*B*}
using *ineq-on-Og* ‹*x*∈*Og*› **by** *force*
have *card* {*n. n* < *B* ∧ (∃ *l*∈{*Suc 0..n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*delta l* ∗ *real l*))} ≤ *card* (*bad-times x* ∩ {..<*B*})
apply (*rule card-mono*, *simp*) **using** ∗ **by** *auto*
also have ... ≤ *d2* ∗ *B* **using** *card-bad-times* ‹*x* ∈ *Og*› **by** *auto*
also have ... ≤ *d* ∗ *B* **unfolding** *d2-def* **using** ‹*d*>*0*› **by** *auto*
finally show *card* {*n. n* < *B* ∧ (∃ *l*∈{*Suc 0..n*}. *u n x* − *u* (*n* − *l*) *x* ≤ − (*delta l* ∗ *real l*))} ≤ *d* ∗ *B*
by *simp*
qed
then have *emeasure M Og* ≤ *emeasure M* {*x* ∈ *space M*. ∀ (*B*::*nat*). *card* {*n* ∈{..<*B*}. ∃ *l* ∈ {*1..n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *delta l* ∗ *l*} ≤ *d* ∗ *B*}
by (*rule emeasure-mono*, *auto*)
then have *emeasure M* {*x* ∈ *space M*. ∀ (*B*::*nat*). *card* {*n* ∈{..<*B*}. ∃ *l* ∈ {*1..n*}. *u n x* − *u* (*n*−*l*) *x* ≤ − *delta l* ∗ *l*} ≤ *d* ∗ *B*} > *1*−*d*
using ‹*emeasure M Og* > *1* − *d*› **by** *auto*
then show *?thesis* **using** ‹*delta* ⟶ *0*› ‹∀ *l*. *delta l* > *0*› **by** *auto*
qed

We go back to the natural time direction, by using the previous result for the inverse map and the inverse subcocycle, and a change of variables argument. The price to pay is that the estimates we get are weaker: we have a control on a set of upper asymptotic density close to 1, while having a set of lower asymptotic density close to 1 as before would be stronger. This will nevertheless be sufficient for our purposes below.

lemma *upper-density-good-direction-invertible*:
assumes *invertible-qmpt*
d>(*0*::*real*) *d* ≤ *1*
shows ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
emeasure M {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1..n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} ≥ *1*−*d*} ≥ *ennreal*(*1*−*d*)
proof −
interpret *I*: *Gouezel-Karlsson-Kingman M Tinv* (λ*n x*. *u n* ((*Tinv*^^*n*) *x*))
proof
show *Tinv* ∈ *quasi-measure-preserving M M*
using *Tinv-qmpt*[*OF* ‹*invertible-qmpt*›] **unfolding** *qmpt-def qmpt-axioms-def* **by** *simp*
show *Tinv* ∈ *measure-preserving M M*
using *Tinv-mpt*[*OF* ‹*invertible-qmpt*›] **unfolding** *mpt-def mpt-axioms-def* **by** *simp*
show *mpt.subcocycle M Tinv* (λ*n x*. *u n* ((*Tinv* ^^ *n*) *x*))

using *subcocycle-u-Tinv*[*OF subu* ‹*invertible-qmpt*›] **by** *simp*
show $-\infty <$ *subcocycle-avg-ereal* ($\lambda n\ x.\ u\ n\ ((Tinv\ \widehat{}\widehat{}\ n)\ x)$)
using *subcocycle-avg-ereal-Tinv*[*OF subu* ‹*invertible-qmpt*›] *subu-fin* **by** *simp*
show *AE x in M. fmpt.subcocycle-lim M Tinv* ($\lambda n\ x.\ u\ n\ ((Tinv\ \widehat{}\widehat{}\ n)\ x)$) $x = 0$
using *subcocycle-lim-Tinv*[*OF subu* ‹*invertible-qmpt*›] *subu-0* **by** *auto*
qed
have *bij*: *bij T* **using** ‹*invertible-qmpt*› **unfolding** *invertible-qmpt-def* **by** *simp*

define *e* **where** $e = d * d\ /\ 2$
have $e>0$ $e\leq 1$ **unfolding** *e-def* **using** ‹$d>0$› ‹$d \leq 1$›
by (*auto, meson less-imp-le mult-left-le one-le-numeral order-trans*)
obtain *delta*::*nat* $\Rightarrow$ *real* **where** *d*: $\bigwedge l.$ *delta l* > 0
delta $\longrightarrow 0$
emeasure M $\{x \in$ *space M*. $\forall N.$
card $\{n \in \{..<N\}.\ \exists l\in\{1..n\}.\ u\ n\ ((Tinv\ \widehat{}\widehat{}\ n)\ x) - u\ (n - l)\ ((Tinv\ \widehat{}\widehat{}\ (n - l))\ x) \leq -$ *delta l* $*$ *real l*$\} \leq e *$ *real N*$\}$
$> 1-e$
using *I.upper-density-delta*[*OF* ‹$e>0$› ‹$e\leq 1$›] **by** *blast*

define *S* **where** $S = \{x \in$ *space M*. $\forall N.$
card $\{n \in \{..<N\}.\ \exists l\in\{1..n\}.\ u\ n\ ((Tinv\ \widehat{}\widehat{}\ n)\ x) - u\ (n - l)\ ((Tinv\ \widehat{}\widehat{}\ (n - l))\ x) \leq -$ *delta l* $*$ *real l*$\} \leq e *$ *real N*$\}$
have [*measurable*]: $S \in$ *sets M* **unfolding** *S-def* **by** *auto*
have *emeasure M S* $> 1 - e$ **unfolding** *S-def* **using** *d*(*3*) **by** *simp*

define *Og* **where** $Og = (\lambda n.\ \{x \in$ *space M*. $\forall l\in\{1..n\}.\ u\ n\ ((Tinv\ \widehat{}\widehat{}\ n)\ x) - u\ (n - l)\ ((Tinv\ \widehat{}\widehat{}\ (n - l))\ x) > -$ *delta l* $*$ *real l*$\})$
have [*measurable*]: *Og n* $\in$ *sets M* **for** *n* **unfolding** *Og-def* **by** *auto*
define *Pg* **where** $Pg = (\lambda n.\ \{x \in$ *space M*. $\forall l\in\{1..n\}.\ u\ n\ x - u\ (n - l)\ ((T\widehat{}\widehat{}l)\ x) > -$ *delta l* $*$ *real l*$\})$
have [*measurable*]: *Pg n* $\in$ *sets M* **for** *n* **unfolding** *Pg-def* **by** *auto*

define *Bad* **where** $Bad = (\lambda i$::*nat*. $\{x \in$ *space M*. $\forall N\geq i.$ *card* $\{n \in \{..<N\}.\ x \in Pg\ n\} \leq (1-d) *$ *real N*$\})$
have [*measurable*]: *Bad i* $\in$ *sets M* **for** *i* **unfolding** *Bad-def* **by** *auto*
then have *range Bad* $\subseteq$ *sets M* **by** *auto*
have *incseq Bad*
unfolding *Bad-def incseq-def* **by** *auto*
have *inc*: $\{x \in$ *space M*. *upper-asymptotic-density* $\{n.\ \forall l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ ((T\widehat{}\widehat{}l)\ x) > -$ *delta l* $* l\} < 1-d\}$
$\subseteq (\bigcup i.\ Bad\ i)$
proof
fix *x* **assume** *H*: $x \in \{x \in$ *space M*. *upper-asymptotic-density* $\{n.\ \forall l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ ((T\widehat{}\widehat{}l)\ x) > -$ *delta l* $* l\} < 1-d\}$
then have $x \in$ *space M* **by** *simp*
define *A* **where** $A = \{n.\ \forall l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ ((T\widehat{}\widehat{}l)\ x) > -$ *delta l* $* l\}$
have *upper-asymptotic-density A* $< 1-d$ **using** *H* **unfolding** *A-def* **by** *simp*

then have $\exists i. \forall N \geq i.$ *card* $(A \cap \{..<N\}) \leq (1-d)*$ *real N*
using *upper-asymptotic-densityD*[*of A 1−d*] **by** (*metis* (*no-types*, *lifting*) *eventually-sequentially less-imp-le*)
then obtain *i* **where** *card* $(A \cap \{..<N\}) \leq (1-d)*$ *real N* **if** $N \geq i$ **for** *N* **by** *blast*
moreover have $A \cap \{..<N\} = \{n.\ n<N \wedge (\forall l \in \{1..n\}.\ u\ n\ x - u\ (n-l)\ ((T^{\frown\frown}l)\ x) > -\ delta\ l * l)\}$ **for** *N*
unfolding *A-def* **by** *auto*
ultimately have $x \in Bad\ i$ **unfolding** *Bad-def Pg-def* **using** ‹$x \in space\ M$›
by *auto*
then show $x \in (\bigcup i.\ Bad\ i)$ **by** *blast*
qed

have *emeasure M* (*Og n*) $\leq$ *emeasure M* (*Pg n*) **for** *n*
proof −
have ∗: $(T^{\frown\frown}n)-`(Og\ n) \cap space\ M \subseteq Pg\ n$ **for** *n*
proof
fix *x* **assume** *x*: $x \in (T^{\frown\frown}n)-`(Og\ n) \cap space\ M$
define *y* **where** $y = (T^{\frown\frown}n)\ x$
then have $y \in Og\ n$ **using** *x* **by** *auto*
have $y \in space\ M$ **using** *sets.sets-into-space*[*OF* ‹$Og\ n \in sets\ M$›] ‹$y \in Og\ n$› **by** *auto*
have $x = (Tinv^{\frown\frown}n)\ y$
unfolding *y-def Tinv-def* **using** *inv-fn-o-fn-is-id*[*OF bij*, *of n*] **by** (*metis comp-apply*)
{
fix *l* **assume** $l \in \{1..n\}$
have $(T^{\frown\frown}l)\ x = (T^{\frown\frown}l)\ ((Tinv^{\frown\frown}l)\ ((Tinv^{\frown\frown}(n-l))y))$
apply (*subst* ‹$x = (Tinv^{\frown\frown}n)\ y$›) **using** *funpow-add*[*of l n−l Tinv*] ‹$l \in \{1..n\}$› **by** *fastforce*
then have ∗: $(T^{\frown\frown}l)\ x = (Tinv^{\frown\frown}(n-l))\ y$
unfolding *Tinv-def* **using** *fn-o-inv-fn-is-id*[*OF bij*] **by** (*metis comp-apply*)
then have $u\ n\ x - u\ (n-l)\ ((T^{\frown\frown}l)\ x) = u\ n\ ((Tinv^{\frown\frown}n)\ y) - u\ (n-l)\ ((Tinv^{\frown\frown}(n-l))\ y)$
using ‹$x = (Tinv^{\frown\frown}n)\ y$› **by** *auto*
also have ... $> -\ delta\ l * real\ l$
using ‹$y \in Og\ n$› ‹$l \in \{1..n\}$› **unfolding** *Og-def* **by** *auto*
finally have $u\ n\ x - u\ (n-l)\ ((T^{\frown\frown}l)\ x) > -\ delta\ l * real\ l$ **by** *simp*
}
then show $x \in Pg\ n$
unfolding *Pg-def* **using** *x* **by** *auto*
qed
have *emeasure M* (*Og n*) = *emeasure M* $((T^{\frown\frown}n)-`(Og\ n) \cap space\ M)$
using *T-vrestr-same-emeasure*(*2*) **unfolding** *vimage-restr-def* **by** *auto*
also have ... $\leq$ *emeasure M* (*Pg n*)
apply (*rule emeasure-mono*) **using** ∗ **by** *auto*
finally show *?thesis* **by** *simp*
qed

```
{
  fix N::nat assume N ≥ 1
  have I: card {n∈{..<N}. x ∈ Og n} ≥ (1−e) ∗ real N if x ∈ S for x
  proof −
    have x ∈ space M using ‹x ∈ S› sets.sets-into-space[OF ‹S ∈ sets M›] by auto
    have a: real (card {n. n < N ∧ (∃ l∈{Suc 0..n}. u n ((Tinv ^^ n) x) − u (n − l) ((Tinv ^^ (n − l)) x) ≤ − (delta l ∗ real l))}) ≤ e ∗ real N
      using ‹x ∈ S› unfolding S-def by auto
    have ∗: {n. n < N} = {n. n < N ∧ (∃ l∈{Suc 0..n}. u n ((Tinv ^^ n) x) − u (n − l) ((Tinv ^^ (n − l)) x) ≤ − (delta l ∗ real l))}
          ∪ {n. n < N ∧ x ∈ Og n} unfolding Og-def using ‹x ∈ space M›
      by (auto, meson atLeastAtMost-iff linorder-not-le)
    have N = card {n. n < N} by auto
    also have ... = card {n. n < N ∧ (∃ l∈{Suc 0..n}. u n ((Tinv ^^ n) x) − u (n − l) ((Tinv ^^ (n − l)) x) ≤ − (delta l ∗ real l))}
          + card {n. n < N ∧ x ∈ Og n}
      apply (subst ∗, rule card-Un-disjoint) unfolding Og-def by auto
    ultimately have real N ≤ e ∗ real N + card {n. n < N ∧ x ∈ Og n}
      using a by auto
    then show ?thesis
      by (auto simp add: algebra-simps)
  qed

  define m where m = measure M (Bad N)
  have m ≥ 0 1−m ≥ 0 unfolding m-def by auto

  have ∗: 1−e ≤ emeasure M S using ‹emeasure M S > 1 − e› by auto
  have ennreal((1−e) ∗ ((1−e) ∗ real N)) = ennreal(1−e) ∗ ennreal((1−e) ∗ real N)
    apply (rule ennreal-mult) using ‹e ≤ 1› by auto
  also have ... ≤ emeasure M S ∗ ennreal((1−e) ∗ real N)
    using mult-right-mono[OF ∗] by simp
  also have ... = (∫⁺ x∈S. ((1−e) ∗ real N) ∂M)
    by (metis ‹S ∈ events› mult.commute nn-integral-cmult-indicator)
  also have ... ≤ (∫⁺x ∈ S. ennreal(card {n∈{..<N}. x ∈ Og n}) ∂M)
    apply (rule nn-integral-mono) using I unfolding indicator-def by (simp)
  also have ... ≤ (∫⁺x ∈ space M. ennreal(card {n∈{..<N}. x ∈ Og n}) ∂M)
   by (rule nn-set-integral-set-mono, simp only: sets.sets-into-space[OF ‹S ∈ sets M›])
  also have ... = (∫⁺x. ennreal(card {n∈{..<N}. x ∈ Og n}) ∂M)
    by (rule nn-set-integral-space)
  also have ... = (∫⁺x. ennreal (∑ n∈{..<N}. ((indicator (Og n) x)::nat)) ∂M)
     apply (rule nn-integral-cong) using sum-indicator-eq-card2[of {..<N} Og] by auto
  also have ... = (∫⁺x. (∑ n∈{..<N}. indicator (Og n) x) ∂M)
    apply (rule nn-integral-cong, auto, simp only: sum-ennreal[symmetric])
   by (metis ennreal-0 ennreal-eq-1 indicator-eq-1-iff indicator-simps(2) real-of-nat-indicator)
  also have ... = (∑ n ∈{..<N}. (∫⁺x. (indicator (Og n) x) ∂M))
```

by (*rule nn-integral-sum*, *simp*)
also have ... = ($\sum$ *n* ∈{..<*N*}. *emeasure M* (*Og n*))
by *simp*
also have ... ≤ ($\sum$ *n* ∈{..<*N*}. *emeasure M* (*Pg n*))
apply (*rule sum-mono*) **using** ‹$\bigwedge$*n*. *emeasure M* (*Og n*) ≤ *emeasure M* (*Pg n*)› **by** *simp*
also have ... = ($\sum$ *n* ∈{..<*N*}. ($\int^+$*x*. (*indicator* (*Pg n*) *x*) ∂*M*))
by *simp*
also have ... = ($\int^+$*x*. ($\sum$ *n*∈{..<*N*}. *indicator* (*Pg n*) *x*) ∂*M*)
by (*rule nn-integral-sum*[*symmetric*], *simp*)
also have ... = ($\int^+$*x*. *ennreal* ($\sum$ *n*∈{..<*N*}. ((*indicator* (*Pg n*) *x*)::*nat*)) ∂*M*)
apply (*rule nn-integral-cong*, *auto*, *simp only*: *sum-ennreal*[*symmetric*])
by (*metis ennreal-0 ennreal-eq-1 indicator-eq-1-iff indicator-simps*(*2*) *real-of-nat-indicator*)
also have ... = ($\int^+$*x*. *ennreal*(*card* {*n*∈{..<*N*}. *x* ∈ *Pg n*}) ∂*M*)
apply (*rule nn-integral-cong*) **using** *sum-indicator-eq-card2*[*of* {..<*N*} *Pg*] **by** *auto*
also have ... = ($\int^+$*x* ∈ *space M*. *ennreal*(*card* {*n*∈{..<*N*}. *x* ∈ *Pg n*}) ∂*M*)
by (*rule nn-set-integral-space*[*symmetric*])
also have ... = ($\int^+$*x* ∈ *Bad N* ∪ (*space M* − *Bad N*). *ennreal*(*card* {*n*∈{..<*N*}. *x* ∈ *Pg n*}) ∂*M*)
apply (*rule nn-integral-cong*) **unfolding** *indicator-def* **by** *auto*
also have ... = ($\int^+$*x* ∈ *Bad N*. *ennreal*(*card* {*n*∈{..<*N*}. *x* ∈ *Pg n*}) ∂*M*)
+ ($\int^+$*x* ∈ *space M* − *Bad N*. *ennreal*(*card* {*n*∈{..<*N*}. *x* ∈ *Pg n*}) ∂*M*)
by (*rule nn-integral-disjoint-pair*, *auto*)
also have ... ≤ ($\int^+$*x* ∈ *Bad N*. *ennreal*((*1*−*d*) ∗ *real N*) ∂*M*) + ($\int^+$*x* ∈ *space M* − *Bad N*. *ennreal*(*real N*) ∂*M*)
apply (*rule add-mono*)
apply (*rule nn-integral-mono*, *simp add*: *Bad-def indicator-def*, *auto*)
apply (*rule nn-integral-mono*, *simp add*: *indicator-def*, *auto*)
using *card-Collect-less-nat*[*of N*] *card-mono*[*of* {*n*. *n* < *N*}] **by** (*simp add*: *Collect-mono-iff*)
also have ... = *ennreal*((*1*−*d*) ∗ *real N*) ∗ *emeasure M* (*Bad N*) + *ennreal*(*real N*) ∗ *emeasure M* (*space M* − *Bad N*)
by (*simp add*: *nn-integral-cmult-indicator*)
also have ... = *ennreal*((*1*−*d*) ∗ *real N*) ∗ *ennreal*(*m*) + *ennreal*(*real N*) ∗ *ennreal*(*1*−*m*)
unfolding *m-def* **by** (*simp add*: *emeasure-eq-measure prob-compl*)
also have ... = *ennreal*((*1*−*d*) ∗ *real N* ∗ *m* + *real N* ∗ (*1*−*m*))
using ‹*m* ≥ *0*› ‹*1*−*m* ≥ *0*› ‹*d* ≤ *1*› *ennreal-plus ennreal-mult* **by** *auto*
finally have *ennreal*((*1*−*e*) ∗ ((*1*−*e*) ∗ *real N*)) ≤ *ennreal*((*1*−*d*) ∗ *real N* ∗ *m* + *real N* ∗ (*1*−*m*))
by *simp*
moreover have (*1*−*d*) ∗ *real N* ∗ *m* + *real N* ∗ (*1*−*m*) ≥ *0*
using ‹*m* ≥ *0*› ‹*1*−*m* ≥ *0*› ‹*d* ≤ *1*› **by** *auto*
ultimately have (*1*−*e*) ∗ ((*1*−*e*) ∗ *real N*) ≤ (*1*−*d*) ∗ *real N* ∗ *m* + *real N* ∗ (*1*−*m*)
using *ennreal-le-iff* **by** *auto*
then have *0* ≤ (*e* ∗ *2* − *d* ∗ *m* − *e* ∗ *e*) ∗ *real N*

by (*auto simp add*: *algebra-simps*)
then have *0* ≤ *e* ∗ *2* − *d* ∗ *m* − *e* ∗ *e*
using ‹*N* ≥ *1*› **by** (*simp add*: *zero-le-mult-iff*)
also have ... ≤ *e* ∗ *2* − *d* ∗ *m*
using ‹*e* > *0*› **by** *auto*
finally have *m* ≤ *e* ∗ *2* / *d*
using ‹*d*>*0*› **by** (*auto simp add*: *algebra-simps divide-simps*)
then have *m* ≤ *d*
unfolding *e-def* **using** ‹*d*>*0*› **by** (*auto simp add*: *divide-simps*)
then have *emeasure M* (*Bad N*) ≤ *d*
unfolding *m-def* **by** (*simp add*: *emeasure-eq-measure ennreal-leI*)
}
then have *emeasure M* (⋃ *i*. *Bad i*) ≤ *d*
using *LIMSEQ-le-const2*[*OF Lim-emeasure-incseq*[*OF* ‹*range Bad* ⊆ *sets M*› ‹*incseq Bad*›]] **by** *auto*
then have *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} < *1*−*d*} ≤ *d*
using *emeasure-mono*[*OF inc*, *of M*] **by** *auto*
then have ∗: *measure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} < *1*−*d*} ≤ *d*
using *emeasure-eq-measure* ‹*d*>*0*› **by** *fastforce*

have {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} ≥ *1*−*d*}
= *space M* − {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} < *1*−*d*}
by *auto*
then have *measure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} ≥ *1*−*d*}
= *measure M* (*space M* − {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} < *1*−*d*})
by *simp*
also have ... = *measure M* (*space M*)
− *measure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} < *1*−*d*}
by (*rule measure-Diff*, *auto*)
also have ... ≥ *1* − *d*
using ∗ *prob-space* **by** *linarith*
finally have *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*} ≥ *1*−*d*} ≥ *1* − *d*
using *emeasure-eq-measure* **by** *auto*
then show *?thesis* **using** *d*(*1*) *d*(*2*) **by** *blast*
qed

Now, we want to remove the invertibility assumption in the previous lemma. The idea is to go to the natural extension of the system, use the result there and project it back. However, if the system is not defined on a polish space, there is no reason why it should have a natural extension, so we have first to project the original system on a polish space on which the subcocycle is defined. This system is obtained by considering the joint distribution of the subcocycle and all its iterates (this is indeed a polish system, as a space of functions from $\mathbb{N}^2$ to $\mathbb{R}$).

```
lemma upper-density-good-direction:
  assumes d>(0::real) d ≤ 1
  shows ∃ delta::nat⇒real. (∀ l. delta l > 0) ∧ (delta ⟶ 0) ∧
        emeasure M {x ∈ space M. upper-asymptotic-density {n. ∀ l ∈ {1..n}. u n
x − u (n−l) ((T^^l) x) > − delta l ∗ l} ≥ 1−d} ≥ ennreal(1−d)
proof −
  define U where U = (λx. (λn. u n x))
  define projJ where projJ = (λx. (λn. U ((T^^n)x)))
  define MJ where MJ = (distr M borel (λx. (λn. U ((T^^n)x))))
  define TJ::(nat ⇒ nat ⇒ real) ⇒ (nat ⇒ nat ⇒ real) where TJ = nat-left-shift
  have ∗: mpt-factor projJ MJ TJ
    unfolding projJ-def MJ-def TJ-def apply (rule fmpt-factor-projection)
    unfolding U-def by (rule measurable-coordinatewise-then-product, simp)
  interpret J: polish-pmpt MJ TJ
    unfolding projJ-def polish-pmpt-def apply (auto)
    apply (rule pmpt-factor) using ∗ apply simp
    unfolding polish-pmpt-axioms-def MJ-def by auto
  have [simp]: projJ ∈ measure-preserving M MJ using mpt-factorE(1)[OF ∗] by
simp
  then have [measurable]: projJ ∈ measurable M MJ by (simp add: measure-preservingE(1))
```

We define a subcocycle uJ in the projection corresponding to the original subcocycle u above. (With the natural definition, it is only a subcocycle almost everywhere.) We check that it shares most properties of u.

```
  define uJ::nat ⇒ (nat ⇒ nat ⇒ real) ⇒ real where uJ = (λn x. x 0 n)
  have [measurable]: uJ n ∈ borel-measurable borel for n
   unfolding uJ-def by (metis measurable-product-coordinates measurable-product-then-coordinatewise)
  moreover have measurable borel borel = measurable MJ borel
    apply (rule measurable-cong-sets) unfolding MJ-def by auto
  ultimately have [measurable]: uJ n ∈ borel-measurable MJ for n by fast
  have uJ-proj: u n x = uJ n (projJ x) for n x
    unfolding uJ-def projJ-def U-def by auto
  have uJ-int: integrable MJ (uJ n) for n
   apply (rule measure-preserving-preserves-integral'(1)[OF ‹projJ ∈ measure-preserving
M MJ›])
    apply (subst uJ-proj[of n, symmetric]) using int-u[of n] by auto
  have uJ-int2: (∫ x. uJ n x ∂MJ) = (∫ x. u n x ∂M) for n
    unfolding uJ-proj
   apply (rule measure-preserving-preserves-integral'(2)[OF ‹projJ ∈ measure-preserving
M MJ›])
    apply (subst uJ-proj[of n, symmetric]) using int-u[of n] by auto
  have uJ-AE: AE x in MJ. uJ (n+m) x ≤ uJ n x + uJ m ((TJ^^n) x) for m n
  proof −
   have AE x in M. uJ (n+m) (projJ x) ≤ uJ n (projJ x) + uJ m (projJ ((T^^n)
x))
```

unfolding *uJ-proj*[*symmetric*] **using** *subcocycle-ineq*[*OF subu*] **by** *auto*
moreover have *AE x in M. projJ* ((*T*^^*n*) *x*) = (*TJ*^^*n*) (*projJ x*)
using *qmpt-factor-iterates*[*OF mpt-factor-is-qmpt-factor*[*OF* ∗]] **by** *auto*
ultimately have ∗: *AE x in M. uJ* (*n*+*m*) (*projJ x*) ≤ *uJ n* (*projJ x*) + *uJ m*
((*TJ*^^*n*) (*projJ x*))
by *auto*
show *?thesis*
apply (*rule quasi-measure-preserving-AE'*[*OF measure-preserving-is-quasi-measure-preserving*[*OF*
‹*projJ* ∈ *measure-preserving M MJ*›], *OF* ∗])
by *auto*
qed
have *uJ-0*: *AE x in MJ.* (λ*n. uJ n x* / *n*) ⟶ *0*
proof −
have *AE x in M.* (λ*n. u n x* / *n*) ⟶ *subcocycle-lim u x*
by (*rule kingman-theorem-nonergodic*(*1*)[*OF subu subu-fin*])
moreover have *AE x in M. subcocycle-lim u x* = *0*
using *subu-0* **by** *simp*
ultimately have ∗: *AE x in M.* (λ*n. uJ n* (*projJ x*) / *n*) ⟶ *0*
unfolding *uJ-proj* **by** *auto*
show *?thesis*
apply (*rule quasi-measure-preserving-AE'*[*OF measure-preserving-is-quasi-measure-preserving*[*OF*
‹*projJ* ∈ *measure-preserving M MJ*›], *OF* ∗])
by *auto*
qed

Then, we go to the natural extension of TJ, to have an invertible system.

define *MI* **where** *MI* = *J.natural-extension-measure*
define *TI* **where** *TI* = *J.natural-extension-map*
define *projI* **where** *projI* = *J.natural-extension-proj*
interpret *I*: *pmpt MI TI* **unfolding** *MI-def TI-def* **by** (*rule J.natural-extension*(*1*))
have *I.mpt-factor projI MJ TJ* **unfolding** *projI-def*
using *I.mpt-factorE*(*1*) *J.natural-extension*(*3*) *MI-def TI-def* **by** *auto*
then have [*simp*]: *projI* ∈ *measure-preserving MI MJ* **using** *I.mpt-factorE*(*1*)
by *auto*
then have [*measurable*]: *projI* ∈ *measurable MI MJ* **by** (*simp add*: *measure-preservingE*(*1*))
have *I.invertible-qmpt*
using *J.natural-extension*(*2*) *MI-def TI-def* **by** *auto*

We define a natural subcocycle uI there, and check its properties.

define *uI* **where** *uI-proj*: *uI* = (λ*n x. uJ n* (*projI x*))
have [*measurable*]: *uI n* ∈ *borel-measurable MI* **for** *n* **unfolding** *uI-proj* **by** *auto*
have *uI-int*: *integrable MI* (*uI n*) **for** *n*
unfolding *uI-proj* **by** (*rule measure-preserving-preserves-integral*(*1*)[*OF* ‹*projI*
∈ *measure-preserving MI MJ*› *uJ-int*])
have (∫ *x. uJ n x* ∂*MJ*) = (∫ *x. uI n x* ∂*MI*) **for** *n*
unfolding *uI-proj* **by** (*rule measure-preserving-preserves-integral*(*2*)[*OF* ‹*projI*
∈ *measure-preserving MI MJ*› *uJ-int*])
then have *uI-int2*: (∫ *x. uI n x* ∂*MI*) = (∫ *x. u n x* ∂*M*) **for** *n*
using *uJ-int2* **by** *simp*

```
have uI-AE: AE x in MI. uI (n+m) x ≤ uI n x + uI m (((TI)^^n) x) for m n
proof -
  have AE x in MI. uJ (n+m) (projI x) ≤ uJ n (projI x) + uJ m (((TJ)^^n)
(projI x))
  apply (rule quasi-measure-preserving-AE[OF measure-preserving-is-quasi-measure-preserving[OF
‹projI ∈ measure-preserving MI MJ›]])
    using uJ-AE by auto
  moreover have AE x in MI. ((TJ)^^n) (projI x) = projI (((TI)^^n) x)
   using I.qmpt-factor-iterates[OF I.mpt-factor-is-qmpt-factor[OF ‹I.mpt-factor
projI MJ TJ›]]
    by auto
  ultimately show ?thesis unfolding uI-proj by auto
qed
have uI-0: AE x in MI. (λn. uI n x / n) ⟶ 0
  unfolding uI-proj
 apply (rule quasi-measure-preserving-AE[OF measure-preserving-is-quasi-measure-preserving[OF
‹projI ∈ measure-preserving MI MJ›]])
  using uJ-0 by simp
```

As uI is only a subcocycle almost everywhere, we correct it to get a genuine subcocycle, to which we will apply Lemma `upper_density_good_direction_invertible`.

```
obtain vI where H: I.subcocycle vI AE x in MI. ∀ n. vI n x = uI n x
  using I.subcocycle-AE[OF uI-AE uI-int] by blast
have [measurable]: ⋀n. vI n ∈ borel-measurable MI ⋀n. integrable MI (vI n)
  using I.subcocycle-integrable[OF H(1)] by auto
have (∫ x. vI n x ∂MI) = (∫ x. uI n x ∂MI) for n
  apply (rule integral-cong-AE) using H(2) by auto
then have (∫ x. vI n x ∂MI) = (∫ x. u n x ∂M) for n
  using uI-int2 by simp
then have I.subcocycle-avg-ereal vI = subcocycle-avg-ereal u
  unfolding I.subcocycle-avg-ereal-def subcocycle-avg-ereal-def by auto
then have vI-fin: I.subcocycle-avg-ereal vI > −∞ using subu-fin by simp
have AE x in MI. (λn. vI n x / n) ⟶ 0
  using uI-0 H(2) by auto
moreover have AE x in MI. (λn. vI n x / n) ⟶ I.subcocycle-lim vI x
  by (rule I.kingman-theorem-nonergodic(1)[OF H(1) vI-fin])
ultimately have vI-0: AE x in MI. I.subcocycle-lim vI x = 0
  using LIMSEQ-unique by auto

interpret GKK: Gouezel-Karlsson-Kingman MI TI vI
  apply standard
  using H(1) vI-fin vI-0 by auto
obtain delta where delta: ⋀l. delta l > 0 delta ⟶ 0
   emeasure MI {x ∈ space MI. upper-asymptotic-density {n. ∀ l∈{1..n}. − delta
l * real l < vI n x − vI (n − l) ((TI ^^ l) x)} ≥ 1 − d } ≥ 1 − d
 using GKK.upper-density-good-direction-invertible[OF ‹I.invertible-qmpt› ‹d>0›
‹d≤1›] by blast
```

Then, we need to go back to the original system, showing that the estimates for TI carry over. First, we go to TJ.

have *BJ*: *emeasure MJ* {*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d* } ≥ *1* − *d*
proof −
have ∗: *AE x in MI*. *uJ n* (*projI x*) = *vI n x* **for** *n*
using *uI-proj H*(*2*) **by** *auto*
have ∗∗: *AE x in MI*. ∀ *n*. *uJ n* (*projI x*) = *vI n x*
by (*subst AE-all-countable*, *auto intro*: ∗)
have *AE x in MI*. ∀ *m n*. *uJ n* (*projI* ((*TI*^^*m*) *x*)) = *vI n* ((*TI*^^*m*) *x*)
by (*rule I.T-AE-iterates*[*OF* ∗∗])
then have *AE x in MI*. (∀ *m n*. *uJ n* (*projI* ((*TI*^^*m*) *x*)) = *vI n* ((*TI*^^*m*) *x*)) ∧ (∀ *n*. *projI* ((*TI*^^*n*) *x*) = (*TJ*^^*n*) (*projI x*))
using *I.qmpt-factor-iterates*[*OF I.mpt-factor-is-qmpt-factor*[*OF* ‹*I.mpt-factor projI MJ TJ*›]] **by** *auto*
then obtain *ZI* **where** *ZI*: ⋀*x*. *x* ∈ *space MI* − *ZI* ⟹ (∀ *m n*. *uJ n* (*projI* ((*TI*^^*m*) *x*)) = *vI n* ((*TI*^^*m*) *x*)) ∧ (∀ *n*. *projI* ((*TI*^^*n*) *x*) = (*TJ*^^*n*) (*projI x*))
ZI ∈ *null-sets MI*
using *AE-E3* **by** *blast*

have ∗: *uJ n* (*projI x*) − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) (*projI x*)) = *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*) **if** *x* ∈ *space MI* − *ZI* **for** *x n l*
proof −
have (*TI*^^*0*) *x* = *x* (*TJ*^^*0*) (*projI x*) = (*projI x*) **by** *auto*
then show *?thesis* **using** *ZI*(*1*)[*OF that*] **by** *metis*
qed
have *projI*−‘{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space MI* − *ZI*
= {*x* ∈ *space MI* − *ZI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n* (*projI x*) − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) (*projI x*))} ≥ *1* − *d*}
by (*auto simp add*: *measurable-space*[*OF* ‹*projI* ∈ *measurable MI MJ*›])
also have ... = {*x* ∈ *space MI* − *ZI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*)} ≥ *1* − *d*}
using ∗ **by** *auto*
also have ... = {*x* ∈ *space MI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*)} ≥ *1* − *d*} − *ZI*
by *auto*
finally have ∗: *projI*−‘{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space MI* − *ZI*
= {*x* ∈ *space MI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*)} ≥ *1* − *d*} − *ZI*
by *simp*

have *emeasure MJ* {*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*}
= *emeasure MI* (*projI*−‘{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1..n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space MI*)

by (*rule measure-preservingE*(*2*)[*symmetric*], *auto*)
also have ... = *emeasure MI* ((*projI*−'{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space MI*) − *ZI*)
by (*rule emeasure-Diff-null-set*[*OF* ‹*ZI* ∈ *null-sets MI*›, *symmetric*], *measurable*)
also have ... = *emeasure MI* ({*x* ∈ *space MI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*)} ≥ *1* − *d*} − *ZI*)
using ∗ **by** *simp*
also have ... = *emeasure MI* {*x* ∈ *space MI*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *vI n x* − *vI* (*n* − *l*) ((*TI* ^^ *l*) *x*)} ≥ *1* − *d*}
by (*rule emeasure-Diff-null-set*[*OF* ‹*ZI* ∈ *null-sets MI*›], *measurable*)
also have ... ≥ *1*−*d*
using *delta*(*3*) **by** *simp*
finally show *?thesis* **by** *simp*
qed

Then, we go back to T with virtually the same argument.

have *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d* } ≥ *1* − *d*
proof −
obtain *Z* **where** *Z*: ⋀*x*. *x* ∈ *space M* − *Z* ⟹ (∀ *n*. *projJ* ((*T*^^*n*) *x*) = (*TJ*^^*n*) (*projJ x*))
Z ∈ *null-sets M*
using *AE-E3*[*OF qmpt-factor-iterates*[*OF mpt-factor-is-qmpt-factor*[*OF* ‹*mpt-factor projJ MJ TJ*›]]] **by** *blast*

have ∗: *uJ n* (*projJ x*) − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) (*projJ x*)) = *u n x* − *u* (*n* − *l*) ((*T*^^ *l*) *x*) **if** *x* ∈ *space M* − *Z* **for** *x n l*
proof −
have (*T*^^*0*) *x* = *x* (*TJ*^^*0*) (*projJ x*) = (*projJ x*) **by** *auto*
then show *?thesis* **using** *Z*(*1*)[*OF that*] *uJ-proj* **by** *metis*
qed
have *projJ*−'{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space M* − *Z*
= {*x* ∈ *space M* − *Z*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n* (*projJ x*) − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) (*projJ x*))} ≥ *1* − *d*}
by (*auto simp add*: *measurable-space*[*OF* ‹*projJ* ∈ *measurable M MJ*›])
also have ... = {*x* ∈ *space M* − *Z*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d*}
using ∗ **by** *auto*
also have ... = {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d*} − *Z*
by *auto*
finally have ∗: *projJ*−'{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space M* − *Z*
= {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d*} − *Z*

by *simp*

have *emeasure MJ* {*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*}
= *emeasure M* (*projJ*−‘{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space M*)
by (*rule measure-preservingE*(*2*)[*symmetric*], *auto*)
also have ... = *emeasure M* ((*projJ*−‘{*x* ∈ *space MJ*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *uJ n x* − *uJ* (*n* − *l*) ((*TJ* ^^ *l*) *x*)} ≥ *1* − *d*} ∩ *space M*) − *Z*)
by (*rule emeasure-Diff-null-set*[*OF* ‹*Z* ∈ *null-sets M*›, *symmetric*], *measurable*)
also have ... = *emeasure M* ({*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d*} − *Z*)
using ∗ **by** *simp*
also have ... = *emeasure M* {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l*∈{*1*..*n*}. − *delta l* ∗ *real l* < *u n x* − *u* (*n* − *l*) ((*T* ^^ *l*) *x*)} ≥ *1* − *d*}
by (*rule emeasure-Diff-null-set*[*OF* ‹*Z* ∈ *null-sets M*›], *measurable*)
finally show *?thesis* **using** *BJ* **by** *simp*
qed
then show *?thesis* **using** *delta*(*1*) *delta*(*2*) **by** *auto*
qed

From the quantitative lemma above, we deduce the qualitative statement we are after, still in the setting of the locale.

lemma *infinite-AE*:
shows *AE x in M*. ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*})
proof −
have ∃ *deltaf*::*real* ⇒ *nat* ⇒ *real*. ∀ *d*. ((*d* > *0* ∧ *d* ≤ *1*) ⟶ ((∀ *l*. *deltaf d l* > *0*) ∧ (*deltaf d* ⟶ *0*) ∧
emeasure M {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − (*deltaf d l*) ∗ *l*} ≥ *1*−*d*} ≥ *ennreal*(*1*−*d*)))
apply (*subst choice-iff* ′[*symmetric*]) **using** *upper-density-good-direction* **by** *auto*
then obtain *deltaf*::*real* ⇒ *nat* ⇒ *real* **where** *H*: ⋀*d*. *d* > *0* ∧ *d* ≤*1* ⟹ (∀ *l*. *deltaf d l* > *0*) ∧ (*deltaf d* ⟶ *0*) ∧
emeasure M {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − (*deltaf d l*) ∗ *l*} ≥ *1*−*d*} ≥ *ennreal*(*1*−*d*)
by *blast*

define *U* **where** *U* = (λ*d*. {*x* ∈ *space M*. *upper-asymptotic-density* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) > − (*deltaf d l*) ∗ *l*} ≥ *1*−*d*})
have [*measurable*]: *U d* ∈ *sets M* **for** *d*
unfolding *U-def* **by** *auto*
have ∗: *emeasure M* (*U d*) ≥ *1* − *d* **if** *d*>*0* ∧ *d*≤ *1* **for** *d*
unfolding *U-def* **using** *H that* **by** *auto*
define *V* **where** *V* = (⋃*n*::*nat*. *U* (*1*/(*n*+*2*)))
have [*measurable*]: *V* ∈ *sets M*
unfolding *V-def* **by** *auto*

```
  have a: emeasure M V ≥ 1 − 1 / (n + 2) for n::nat
  proof −
    have 1 − 1 / (n + 2) = 1 − 1 / (real n + 2)
      by auto
    also have ... ≤ emeasure M (U (1/(real n+2)))
      using *[of 1 / (real n + 2)] by auto
    also have ... ≤ emeasure M V
      apply (rule Measure-Space.emeasure-mono) unfolding V-def by auto
    finally show ?thesis by simp
  qed
  have b: (λn::nat. 1 − 1 / (n + 2)) ⟶ ennreal(1 − 0)
    by (intro tendsto-intros LIMSEQ-ignore-initial-segment)
  have emeasure M V ≥ 1 − 0
    apply (rule Lim-bounded) using a b by auto
  then have emeasure M V = 1
    by (simp add: emeasure-ge-1-iff)
  then have AE x in M. x ∈ V
    by (simp add: emeasure-eq-measure prob-eq-1)
  moreover
  {
    fix x assume x ∈ V
    then obtain n::nat where x ∈ U (1/(real n+2)) unfolding V-def by blast
    define d where d = 1/(real n + 2)
    have 0 < d d≤1 unfolding d-def by auto
    have 0 < 1−d unfolding d-def by auto
     also have ... ≤ upper-asymptotic-density {n. ∀ l ∈ {1..n}. u n x − u (n−l)
((T^^l) x) > − (deltaf d l) * l}
      using ‹x ∈ U (1/(real n+2))› unfolding U-def d-def by auto
    finally have infinite {n. ∀ l ∈ {1..n}. u n x − u (n−l) ((T^^l) x) > − (deltaf
d l) * l}
      using upper-asymptotic-density-finite by force
    then have ∃ delta::nat⇒real. (∀ l. delta l > 0) ∧ (delta ⟶ 0) ∧
          (infinite {n. ∀ l ∈ {1..n}. u n x − u (n−l) ((T^^l) x) > − delta l * l})
      using H ‹0 < d› ‹d≤1› by auto
  }
  ultimately show ?thesis by auto
qed

end
```

Finally, we obtain the full statement, by reducing to the previous situation where the asymptotic average vanishes.

```
theorem (in pmpt) Gouezel-Karlsson-Kingman:
  assumes subcocycle u subcocycle-avg-ereal u > −∞
  shows AE x in M. ∃ delta::nat⇒real. (∀ l. delta l > 0) ∧ (delta ⟶ 0) ∧
          (infinite {n. ∀ l ∈ {1..n}. u n x − u (n−l) ((T^^l) x) − l * subcocycle-lim
u x > − delta l * l})
proof −
  have [measurable]: integrable M (u n) u n ∈ borel-measurable M for n
```

using *subcocycle-integrable*[*OF assms*(*1*)] **by** *auto*

define *v* **where** *v* = *birkhoff-sum* (*λx.* −*subcocycle-lim u x*)
have *int* [*measurable*]: *integrable M* (*λx.* −*subcocycle-lim u x*)
using *kingman-theorem-nonergodic*(*2*)[*OF assms*] **by** *auto*
have *subcocycle v* **unfolding** *v-def*
apply (*rule subcocycle-birkhoff*)
using *assms* ‹*integrable M* (*λx.* −*subcocycle-lim u x*)› **unfolding** *subcocycle-def*
by *auto*
have *subcocycle-avg-ereal v* > − ∞
unfolding *v-def* **using** *subcocycle-avg-ereal-birkhoff*[*OF int*] *kingman-theorem-nonergodic*(*2*)[*OF assms*] **by** *auto*
have *AE x in M*. *subcocycle-lim v x* = *real-cond-exp M Invariants* (*λx.* −*subcocycle-lim u x*) *x*
unfolding *v-def* **by** (*rule subcocycle-lim-birkhoff*[*OF int*])
moreover have *AE x in M*. *real-cond-exp M Invariants* (*λx.* − *subcocycle-lim u x*) *x* = − *subcocycle-lim u x*
by (*rule real-cond-exp-F-meas*[*OF int*], *auto*)
ultimately have *AEv*: *AE x in M*. *subcocycle-lim v x* = − *subcocycle-lim u x*
by *auto*

define *w* **where** *w* = (*λn x.* *u n x* + *v n x*)
have *a*: *subcocycle w*
unfolding *w-def* **by** (*rule subcocycle-add*[*OF assms*(*1*) ‹*subcocycle v*›])
have *b*: *subcocycle-avg-ereal w* > −∞
unfolding *w-def* **by** (*rule subcocycle-avg-add*(*1*)[*OF assms*(*1*) ‹*subcocycle v*› *assms*(*2*) ‹*subcocycle-avg-ereal v* > − ∞›])
have *AE x in M*. *subcocycle-lim w x* = *subcocycle-lim u x* + *subcocycle-lim v x*
unfolding *w-def* **by** (*rule subcocycle-lim-add*[*OF assms*(*1*) ‹*subcocycle v*› *assms*(*2*) ‹*subcocycle-avg-ereal v* > − ∞›])
then have *c*: *AE x in M*. *subcocycle-lim w x* = *0*
using *AEv* **by** *auto*

interpret *Gouezel-Karlsson-Kingman M T w*
proof qed (*use a b c* **in** *auto*)
have *AE x in M*. ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *w n x* − *w* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*})
using *infinite-AE* **by** *auto*
moreover
{
fix *x* **assume** *H*: ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *w n x* − *w* (*n*−*l*) ((*T*^^*l*) *x*) > − *delta l* ∗ *l*})
x ∈ *space M*
have ∗: *v n x* = − *n* ∗ *subcocycle-lim u x* **for** *n*
unfolding *v-def* **using** *birkhoff-sum-of-invariants*[*OF* - ‹*x* ∈ *space M*›] **by** *auto*
have ∗∗: *v n* ((*T*^^*l*) *x*) = − *n* ∗ *subcocycle-lim u x* **for** *n l*
proof −
have *v n* ((*T*^^*l*) *x*) = − *n* ∗ *subcocycle-lim u* ((*T*^^*l*) *x*)

unfolding *v-def* **using** *birkhoff-sum-of-invariants*[*OF - T-spaceM-stable*(*2*)[*OF* ‹*x* ∈ *space M*›]] **by** *auto*
also have ... = − *n* ∗ *subcocycle-lim u x*
using *Invariants-func-is-invariant-n*[*OF subcocycle-lim-meas-Inv*(*2*) ‹*x* ∈ *space M*›] **by** *auto*
finally show *?thesis* **by** *simp*
qed
have *w n x* − *w* (*n*−*l*) ((*T*^^*l*) *x*) = *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* **if** *l* ∈ {*1*..*n*} **for** *n l*
unfolding *w-def* **using** ∗[*of n*] ∗∗[*of n*−*l l*] *that* **by** (*auto simp add*: *algebra-simps*)
then have ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > − *delta l* ∗ *l*})
using *H*(*1*) **by** *auto*
}
ultimately show *?thesis* **by** *auto*
qed

The previous theorem only contains a lower bound. The corresponding upper bound follows readily from Kingman's theorem. The next statement combines both upper and lower bounds.

theorem (**in** *pmpt*) *Gouezel-Karlsson-Kingman'*:
assumes *subcocycle u subcocycle-avg-ereal u* > −∞
shows *AE x in M*. ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l*})
proof −
{
fix *x* **assume** *x*: ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > − *delta l* ∗ *l*})
(λ*l*. *u l x*/*l*) ⟶ *subcocycle-lim u x*
then obtain *alpha*::*nat* ⇒ *real* **where** *a*: ⋀*l*. *alpha l* > *0 alpha* ⟶ *0*
infinite {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > − *alpha l* ∗ *l*}
by *auto*
define *delta*::*nat* ⇒ *real* **where** *delta* = (λ*l*. *alpha l* + *norm*(*u l x* / *l* − *subcocycle-lim u x*))
{
fix *n* **assume** ∗: ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > − *alpha l* ∗ *l*
have *H*: *x* > −*a* ⟹ *x* < *a* ⟹ *abs x* < *a* **for** *a x*::*real* **by** *simp*
have *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l* **if** *l*∈{*1*..*n*} **for** *l*
proof (*rule H*)
have *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* ≤ *u l x* − *l* ∗ *subcocycle-lim u x*
using *assms*(*1*) *subcocycle-ineq*[*OF assms*(*1*), *of l n*−*l x*] *that* **by** *auto*

also have ... ≤ *l* ∗ *norm*(*u l x*/*l* − *subcocycle-lim u x*)
using *that* **by** (*auto simp add*: *algebra-simps divide-simps*)
also have ... < *delta l* ∗ *l*
unfolding *delta-def* **using** *a*(*1*)[*of l*] *that* **by** *auto*
finally show *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* < *delta l* ∗ *l* **by** *simp*

have − (*delta l* ∗ *l*) ≤ −*alpha l* ∗ *l*
unfolding *delta-def* **by** (*auto simp add*: *algebra-simps*)
also have ... < *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*
using ∗ *that* **by** *auto*
finally show *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > −(*delta l* ∗ *l*)
by *simp*
qed
then have ∀ *l* ∈ {*1*..*n*}. *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l*
by *auto*
}
then have {*n*. ∀ *l* ∈ {*1*..*n*}. *u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x* > − *alpha l* ∗ *l*}
⊆ {*n*. ∀ *l* ∈ {*1*..*n*}. *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l*}
by *auto*
then have *infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l*}
using *a*(*3*) *finite-subset* **by** *blast*
moreover have *delta* ⟶ *0* + *0*
unfolding *delta-def* **using** *x*(*2*) **by** (*intro tendsto-intros a*(*2*) *tendsto-norm-zero LIM-zero*)
moreover have *delta l* > *0* **for** *l* **unfolding** *delta-def* **using** *a*(*1*)[*of l*] **by** *auto*
ultimately have ∃ *delta*::*nat*⇒*real*. (∀ *l*. *delta l* > *0*) ∧ (*delta* ⟶ *0*) ∧
(*infinite* {*n*. ∀ *l* ∈ {*1*..*n*}. *abs*(*u n x* − *u* (*n*−*l*) ((*T*^^*l*) *x*) − *l* ∗ *subcocycle-lim u x*) < *delta l* ∗ *l*})
by *auto*
}
then show *?thesis*
using *Gouezel-Karlsson-Kingman*[*OF assms*] *kingman-theorem-nonergodic*(*1*)[*OF assms*] **by** *auto*
qed

end

11 A theorem by Kohlberg and Neyman

theory *Kohlberg-Neyman-Karlsson*
imports *Fekete*
begin

In this section, we prove a theorem due to Kohlberg and Neyman: given a semicontraction T of a euclidean space, then $T^n(0)/n$ converges when $n \to \infty$. The proof we give is due to Karlsson. It mainly builds on subadditivity ideas. The geometry of the space is essentially not relevant except at the very end of the argument, where strict convexity comes into play.

We recall Fekete's lemma: if a sequence is subadditive (i.e., $u_{n+m} \leq u_n + u_m$), then u_n/n converges to its infimum. It is proved in a different file, but we recall the statement for self-containedness.

lemma *fekete*:
  fixes *u*::*nat* ⇒ *real*
  assumes ⋀*n m*. *u* (*m*+*n*) ≤ *u m* + *u n*
    bdd-below {*u n*/*n* | *n*. *n*>*0*}
  shows (λ*n*. *u n*/*n*) ⟶ *Inf* {*u n*/*n* | *n*. *n*>*0*}
apply (*rule subadditive-converges-bounded*) **unfolding** *subadditive-def* **using** *assms*
by *auto*

A real sequence tending to infinity has infinitely many high-scores, i.e., there are infinitely many times where it is larger than all its previous values.

lemma *high-scores*:
  fixes *u*::*nat* ⇒ *real* **and** *i*::*nat*
  assumes *u* ⟶ ∞
  shows ∃ *n* ≥ *i*. ∀ *l* ≤ *n*. *u l* ≤ *u n*
proof −
  define *M* **where** *M* = *Max* {*u l*|*l*. *l* < *i*}
  define *n* **where** *n* = *Inf* {*m*. *u m* > *M*}
  have *eventually* (λ*m*. *u m* > *M*) *sequentially*
    using *assms* **by** (*simp add*: *filterlim-at-top-dense tendsto-PInfty-eq-at-top*)
  then have {*m*. *u m* > *M*} ≠ {} **by** *fastforce*
  then have *n* ∈ {*m*. *u m* > *M*} **unfolding** *n-def* **using** *Inf-nat-def1* **by** *metis*
  then have *u n* > *M* **by** *simp*
  have *n* ≥ *i*
  proof (*rule ccontr*)
    assume ¬ *i* ≤ *n*
    then have ∗: *n* < *i* **by** *simp*
    have *u n* ≤ *M* **unfolding** *M-def* **apply** (*rule Max-ge*) **using** ∗ **by** *auto*
    then show *False* **using** ‹*u n* > *M*› **by** *auto*
  qed
  moreover have *u l* ≤ *u n* **if** *l* ≤ *n* **for** *l*
  proof (*cases l* = *n*)
    case *True*
    then show *?thesis* **by** *simp*
  next
    case *False*
    then have *l* < *n* **using** ‹*l* ≤ *n*› **by** *auto*
    then have *l* ∉ {*m*. *u m* > *M*}
      unfolding *n-def* **by** (*meson bdd-below-def cInf-lower not-le zero-le*)
    then show *?thesis* **using** ‹*u n* > *M*› **by** *auto*

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

Hahn-Banach in euclidean spaces: given a vector u, there exists a unit norm vector v such that $\langle u, v\rangle = \|u\|$ (and we put a minus sign as we will use it in this form). This uses the fact that, in Isabelle/HOL, euclidean spaces have positive dimension by definition.

lemma *select-unit-norm*:
fixes *u*::*'a*::*euclidean-space*
shows ∃ *v*. *norm v* = *1* ∧ *v* · *u* = − *norm u*
proof (*cases u* = *0*)
case *True*
then show *?thesis* **using** *norm-Basis nonempty-Basis* **by** *fastforce*
next
case *False*
show *?thesis*
apply (*rule exI*[*of* - −*u*/$_R$ *norm u*])
using *False* **by** (*auto simp add*: *dot-square-norm power2-eq-square*)
qed

We set up the assumption that we will use until the end of this file, in the following locale: we fix a semicontraction T of a euclidean space. Our goal will be to show that such a semicontraction has an asymptotic translation vector.

locale *Kohlberg-Neyman-Karlsson* =
fixes *T*::*'a*::*euclidean-space* ⇒ *'a*
assumes *semicontract*: *dist* (*T x*) (*T y*) ≤ *dist x y*
begin

The iterates of T are still semicontractions, by induction.

lemma *semicontract-Tn*:
dist ((*T*^^*n*) *x*) ((*T*^^*n*) *y*) ≤ *dist x y*
apply (*induction n*, *auto*) **using** *semicontract order-trans* **by** *blast*

The main quantity we will use is the distance from the origin to its image under T^n. We denote it by u_n. The main point is that it is subadditive by semicontraction, hence it converges to a limit A given by $Inf\{u_n/n\}$, thanks to Fekete Lemma.

definition *u*::*nat* ⇒ *real*
where *u n* = *dist 0* ((*T*^^*n*) *0*)

definition *A*::*real*
where *A* = *Inf* {*u n*/*n* | *n*. *n*>*0*}

lemma *Apos*: *A* ≥ *0*
unfolding *A-def u-def* **by** (*rule cInf-greatest*, *auto*)

lemma *Alim*:(λ*n. u n/n*) ⟶ *A*
unfolding *A-def* **proof** (*rule fekete*)
  show *bdd-below* {*u n / real n* |*n. 0 < n*}
    unfolding *u-def bdd-below-def* **by** (*rule exI*[*of - 0*], *auto*)

  fix *m n*
  have *u* (*m+n*) = *dist 0* ((*T*^^(*m+n*)) *0*)
    unfolding *u-def* **by** *simp*
  also have ... ≤ *dist 0* ((*T*^^*m*) *0*) + *dist* ((*T*^^*m*) *0*) ((*T*^^(*m+n*)) *0*)
    by (*rule dist-triangle*)
  also have ... = *dist 0* ((*T*^^*m*) *0*) + *dist* ((*T*^^*m*) *0*) ((*T*^^*m*) ((*T*^^*n*) *0*))
    by (*auto simp add*: *funpow-add*)
  also have ... ≤ *dist 0* ((*T*^^*m*) *0*) + *dist 0* ((*T*^^*n*) *0*)
    using *semicontract-Tn*[*of m*] *add-mono-thms-linordered-semiring*(*2*) **by** *blast*
  also have ... = *u m* + *u n*
    unfolding *u-def* **by** *auto*
  finally show *u* (*m+n*) ≤ *u m* + *u n* **by** *auto*
qed

The main fact to prove the existence of an asymptotic translation vector for T is the following proposition: there exists a unit norm vector v such that $T^\ell(0)$ is in the half-space at distance $A\ell$ of the origin directed by v.

The idea of the proof is to find such a vector v_i that works (with a small error $\epsilon_i > 0$) for times up to a time n_i, and then take a limit by compactness (or weak compactness, but since we are in finite dimension, compactness works fine). Times n_i are chosen to be large high scores of the sequence $u_n - (A - \epsilon_i)n$, which tends to infinity since u_n/n tends to A.

proposition *half-space*:
  ∃ *v. norm v* = *1* ∧ (∀ *l. v* · (*T* ^^ *l*) *0* ≤ − *A* ∗ *l*)
proof −
  define *eps*::*nat* ⇒ *real* **where** *eps* = (λ*i. 1/of-nat* (*i+1*))
  have *eps i* > *0* **for** *i* **unfolding** *eps-def* **by** *auto*
  have *eps* ⟶ *0*
    unfolding *eps-def* **using** *LIMSEQ-ignore-initial-segment*[*OF lim-1-over-n*, *of 1*] **by** *simp*
  have *vi*: ∃ *vi. norm vi* = *1* ∧ (∀ *l* ≤ *i. vi* · (*T* ^^ *l*) *0* ≤ (− *A* + *eps i*) ∗ *l*) **for** *i*
  proof −
    have *L*: (λ*n. ereal*(*u n* − (*A* − *eps i*) ∗ *n*)) ⟶ ∞
    proof (*rule Lim-transform-eventually*)
      have *ereal* ((*u n/n* − *A*) + *eps i*) ∗ *ereal n* = *ereal*(*u n* − (*A* − *eps i*) ∗ *n*) **if** *n* ≥ *1* **for** *n*
        using *that* **by** (*auto simp add*: *divide-simps algebra-simps*)
      then show *eventually* (λ*n. ereal* ((*u n/n* − *A*) + *eps i*) ∗ *ereal n* = *ereal*(*u n* − (*A* − *eps i*) ∗ *n*)) *sequentially*
        unfolding *eventually-sequentially* **by** *auto*

      have (λ*n.* (*ereal* ((*u n/n* − *A*) + *eps i*)) ∗ *ereal n*) ⟶ (*0* + *eps i*) ∗ ∞

apply (*intro tendsto-intros*)
using ‹*eps i* > *0*› *Alim* **by** (*auto simp add*: *LIM-zero*)
then show (λn. *ereal* (*u n* / *real n* − *A* + *eps i*) ∗ *ereal* (*real n*)) ⟶ ∞
using ‹*eps i* > *0*› **by** *simp*
qed
obtain *n* **where** *n*: *n* ≥ *i* ⋀*l*. *l* ≤ *n* ⟹ *u l* − (*A* − *eps i*) ∗ *l* ≤ *u n* − (*A* − *eps i*) ∗ *n*
using *high-scores*[*OF L*, *of i*] **by** *auto*
obtain *vi* **where** *vi*: *norm vi* = *1 vi* · ((*T*^^*n*) *0*) = − *norm* ((*T*^^*n*) *0*)
using *select-unit-norm* **by** *auto*
have *vi* · (*T* ^^ *l*) *0* ≤ (− *A* + *eps i*) ∗ *l* **if** *l* ≤ *i* **for** *l*
proof −
have ∗: *n* = *l* + (*n*−*l*) **using** *that* ‹*n* ≥ *i*› **by** *auto*
have ∗∗: *real* (*n*−*l*) = *real n* − *real l* **using** *that* ‹*n* ≥ *i*› **by** *auto*
have *vi* · (*T* ^^ *l*) *0* = *vi* · ((*T* ^^ *l*) *0* − (*T*^^*n*) *0*) + *vi* · ((*T*^^*n*) *0*)
by (*simp add*: *inner-diff-right*)
also have ... ≤ *norm vi* ∗ *norm* (((*T* ^^ *l*) *0* − (*T*^^*n*) *0*)) + *vi* · ((*T*^^*n*) *0*)
by (*simp add*: *norm-cauchy-schwarz*)
also have ... = *dist* ((*T*^^*l*)(*0*)) ((*T*^^*n*) *0*) − *norm* ((*T*^^*n*) *0*)
using *vi* **by** (*auto simp add*: *dist-norm*)
also have ... = *dist* ((*T*^^*l*)(*0*)) ((*T*^^*l*) ((*T*^^(*n*−*l*)) *0*)) − *norm* ((*T*^^*n*) *0*)
by (*metis* ∗ *funpow-add o-apply*)
also have ... ≤ *dist 0* ((*T*^^(*n*−*l*)) *0*) − *norm* ((*T*^^*n*) *0*)
using *semicontract-Tn*[*of l 0* (*T*^^(*n*−*l*)) *0*] **by** *auto*
also have ... = *u* (*n*−*l*) − *u n*
unfolding *u-def* **by** *auto*
also have ... ≤ − (*A* − *eps i*) ∗ *l*
using *n*(*2*)[*of n*−*l*] **unfolding** ∗∗ **by** (*auto simp add*: *algebra-simps*)
finally show *?thesis* **by** *auto*
qed
then show *?thesis* **using** *vi*(*1*) **by** *auto*
qed
have ∃ *V*::(*nat* ⇒ ′*a*). ∀ *i*. *norm* (*V i*) = *1* ∧ (∀ *l*≤*i*. *V i* · (*T* ^^ *l*) *0* ≤ (− *A* + *eps i*) ∗ *l*)
apply (*rule choice*) **using** *vi* **by** *auto*
then obtain *V*::*nat* ⇒ ′*a* **where** *V*: ⋀*i*. *norm* (*V i*) = *1* ⋀*l i*. *l* ≤ *i* ⟹ *V i* · (*T* ^^ *l*) *0* ≤ (− *A* + *eps i*) ∗ *l*
by *auto*

have *compact* (*sphere* (*0*::′*a*) *1*) **by** *simp*
moreover have *V i* ∈ *sphere 0 1* **for** *i* **using** *V*(*1*) **by** *auto*
ultimately have ∃ *v* ∈ *sphere 0 1*. ∃ *r*. *strict-mono r* ∧ (*V o r*) ⟶ *v*
using *compact-eq-seq-compact-metric seq-compact-def* **by** *metis*
then obtain *v r* **where** *v*: *v* ∈ *sphere 0 1 strict-mono r* (*V o r*) ⟶ *v*
by *auto*
have *v* · (*T* ^^ *l*) *0* ≤ − *A* ∗ *l* **for** *l*
proof −

have ∗: ($\lambda i.\ (-A + eps\ (r\ i)) * l - V\ (r\ i) \cdot (T\ \widehat{\ }\widehat{\ }\ l)\ 0) \longrightarrow (-A + 0) * l - v \cdot (T\ \widehat{\ }\widehat{\ }\ l)\ 0$
apply (*intro tendsto-intros*)
using ‹(V *o r*) ⟶ *v*› ‹*eps* ⟶ *0*› ‹*strict-mono r*› *LIMSEQ-subseq-LIMSEQ* **unfolding** *comp-def* **by** *auto*
have *eventually* ($\lambda i.\ (-A + eps\ (r\ i)) * l - V\ (r\ i) \cdot (T\ \widehat{\ }\widehat{\ }\ l)\ 0 \geq 0$) *sequentially*
unfolding *eventually-sequentially* **apply** (*rule exI*[*of* - *l*])
using $V(2)$[*of l*] *seq-suble*[*OF* ‹*strict-mono r*›] **apply** *auto* **using** *le-trans* **by** *blast*
then have $(-A + 0) * l - v \cdot (T\ \widehat{\ }\widehat{\ }\ l)\ 0 \geq 0$
using *LIMSEQ-le-const*[*OF* ∗, *of 0*] **unfolding** *eventually-sequentially* **by** *auto*
then show *?thesis* **by** *auto*
qed
then show *?thesis* **using** ‹$v \in$ *sphere 0 1*› **by** *auto*
qed

We can now show the existence of an asymptotic translation vector for T. It is the vector $-v$ of the previous proposition: the point $T^{\ell}(0)$ is in the half-space at distance $A\ell$ of the origin directed by v, and has norm $\sim A\ell$, hence it has to be essentially $-Av$ by strict convexity of the euclidean norm.

theorem *KNK-thm*:
convergent ($\lambda n.\ ((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n$)
proof −
obtain v **where** v: *norm* $v = 1\ \bigwedge l.\ v \cdot (T\ \widehat{\ }\widehat{\ }\ l)\ 0 \leq -A * l$
using *half-space* **by** *auto*
have ($\lambda n.\ norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n + A *_R v)\widehat{\ }2) \longrightarrow 0$
proof (*rule tendsto-sandwich*[*of* λ-. *0* - - $\lambda n.\ (norm((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n)\widehat{\ }2 - A\widehat{\ }2$])
have $norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n + A *_R v)\widehat{\ }2 \leq (norm((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n)\widehat{\ }2 - A\widehat{\ }2$ **if** $n \geq 1$ **for** n
proof −
have $norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n + A *_R v)\widehat{\ }2 = norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n)\widehat{\ }2 + A * A * (norm\ v)\widehat{\ }2 + 2 * A * inverse\ n * (v \cdot (T\widehat{\ }\widehat{\ }n)\ 0)$
unfolding *power2-norm-eq-inner* **by** (*auto simp add*: *inner-commute algebra-simps*)
also have ... $\leq norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n)\widehat{\ }2 + A * A * (norm\ v)\widehat{\ }2 + 2 * A * inverse\ n * (-A * n)$
using *mult-left-mono*[*OF v*(*2*)[*of n*] *Apos*] ‹$n \geq 1$› **by** (*auto*, *auto simp add*: *divide-simps*)
also have ... $= norm(((T\widehat{\ }\widehat{\ }n)\ 0)\ /_R\ n)\widehat{\ }2 - A * A$
using ‹$n \geq 1$› $v(1)$ **by** *auto*
finally show *?thesis* **by** (*simp add*: *power2-eq-square*)
qed
then show *eventually* ($\lambda n.\ norm\ ((T\ \widehat{\ }\widehat{\ }\ n)\ 0\ /_R\ real\ n + A *_R v)\widehat{\ }2 \leq (norm\ ((T\ \widehat{\ }\widehat{\ }\ n)\ 0)\ /_R\ real\ n)^2 - A\widehat{\ }2$) *sequentially*
unfolding *eventually-sequentially* **by** *auto*
have ($\lambda n.\ (norm\ ((T\ \widehat{\ }\widehat{\ }\ n)\ 0)\ /_R\ real\ n)\widehat{\ }2) \longrightarrow A^2$
apply (*intro tendsto-intros*)

using *Alim* **unfolding** *u-def* **by** (*auto simp add*: *divide-simps*)
then show ($\lambda n.$ (*norm* ((*T* ^^ *n*) *0*) $/_R$ *real n*)^2 − A^2) ⟶ *0*
by (*simp add*: *LIM-zero*)
qed (*auto*)
then have ($\lambda n.$ *sqrt*((*norm*(((*T*^^*n*) *0*) $/_R$ *n* + *A* $*_R$ *v*))^*2*)) ⟶ *sqrt 0*
by (*intro tendsto-intros*)
then have ($\lambda n.$ *norm*((((*T*^^*n*) *0*) $/_R$ *n*) − (− *A* $*_R$ *v*))) ⟶ *0*
by *auto*
then have ($\lambda n.$ ((*T*^^*n*) *0*) $/_R$ *n*) ⟶ − *A* $*_R$ *v*
using *Lim-null tendsto-norm-zero-iff* **by** *blast*
then show *convergent* ($\lambda n.$ ((*T*^^*n*) *0*) $/_R$ *n*)
unfolding *convergent-def* **by** *auto*
qed

end

end

12 Transfer Operator

theory *Transfer-Operator*
imports *Recurrence*
begin

context *qmpt* **begin**

The map T acts on measures by push-forward. In particular, if $fd\mu$ is absolutely continuous with respect to the reference measure μ, then its push-forward $T_*(fd\mu)$ is absolutely continuous with respect to μ, and can therefore be written as $gd\mu$ for some function g. The map $f \mapsto g$, representing the action of T on the level of densities, is called the transfer operator associated to T and often denoted by $\hat{T}$.

We first define it on nonnegative functions, using Radon-Nikodym derivatives. Then, we extend it to general real-valued functions by separating it into positive and negative parts.

The theory presents many similarities with the theory of conditional expectations. Indeed, it is possible to make a theory encompassing the two. When the map is measure preserving, there is also a direct relationship: $(\hat{T}f) \circ T$ is the conditional expectation of f with respect to $T^{-1}B$ where B is the sigma-algebra. Instead of building a general theory, we copy the proofs for conditional expectations and adapt them where needed.

12.1 The transfer operator on nonnegative functions

definition *nn-transfer-operator* :: (*'a* ⇒ *ennreal*) ⇒ (*'a* ⇒ *ennreal*)
where

nn-transfer-operator f = (*if f* ∈ *borel-measurable M then RN-deriv M* (*distr* (*density M f*) *M T*)
else (λ-. *0*))

lemma *borel-measurable-nn-transfer-operator* [*measurable*]:
nn-transfer-operator f ∈ *borel-measurable M*
unfolding *nn-transfer-operator-def* **by** *auto*

lemma *borel-measurable-nn-transfer-operator-iterates* [*measurable*]:
assumes [*measurable*]: *f* ∈ *borel-measurable M*
shows (*nn-transfer-operator*^^*n*) *f* ∈ *borel-measurable M*
by (*cases n*, *auto*)

The next lemma is arguably the most fundamental property of the transfer operator: it is the adjoint of the composition by T. If one defined it as an abstract adjoint, it would be defined on the dual of L^∞, which is a large unwieldy space. The point is that it can be defined on genuine functions, using the push-forward point of view above. However, once we have this property, we can forget completely about the definition, since this property characterizes the transfer operator, as the second lemma below shows. From this point on, we will only work with it, and forget completely about the definition using Radon-Nikodym derivatives.

lemma *nn-transfer-operator-intg*:
assumes [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
shows (∫ + *x*. *f x* ∗ *nn-transfer-operator g x* ∂*M*) = (∫ + *x*. *f* (*T x*) ∗ *g x* ∂*M*)
proof −
have ∗: *density M* (*RN-deriv M* (*distr* (*density M g*) *M T*)) = *distr* (*density M g*) *M T*
by (*rule density-RN-deriv*) (*auto intro*!: *quasi-measure-preserving-absolutely-continuous simp add*: *Tqm*)
have (∫ + *x*. *f x* ∗ *nn-transfer-operator g x* ∂*M*) = (∫ + *x*. *f x* ∂(*density M* (*RN-deriv M* (*distr* (*density M g*) *M T*))))
unfolding *nn-transfer-operator-def* **by** (*simp add*: *nn-integral-densityR*)
also have ... = (∫ + *x*. *f x* ∂(*distr* (*density M g*) *M T*))
unfolding ∗ **by** *simp*
also have ... = (∫ + *x*. *f* (*T x*) ∂(*density M g*))
by (*rule nn-integral-distr*, *auto*)
also have ... = (∫ + *x*. *f* (*T x*) ∗ *g x* ∂*M*)
by (*simp add*: *nn-integral-densityR*)
finally show *?thesis* **by** *auto*
qed

lemma *nn-transfer-operator-intTn-g*:
assumes *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
shows (∫ + *x*. *f x* ∗ (*nn-transfer-operator*^^*n*) *g x* ∂*M*) = (∫ + *x*. *f* ((*T*^^*n*) *x*) ∗ *g x* ∂*M*)
proof −
have ⋀*f g*. *f* ∈ *borel-measurable M* =⇒ *g* ∈ *borel-measurable M* =⇒ (∫ + *x*. *f x*

∗ (*nn-transfer-operator*^^*n*) *g x* ∂*M*) = (∫⁺ *x*. *f* ((*T*^^*n*) *x*) ∗ *g x* ∂*M*) **for** *n*
  proof (*induction n*)
    case (*Suc n*)
    have [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M* **by** *fact*+
    have (∫⁺ *x*. *f x* ∗ (*nn-transfer-operator* ^^ *Suc n*) *g x* ∂*M*) = (∫⁺ *x*. *f x* ∗ (*nn-transfer-operator* ((*nn-transfer-operator*^^*n*) *g*)) *x* ∂*M*)
      apply (*rule nn-integral-cong*) **using** *funpow.simps*(*2*) **unfolding** *comp-def* **by** *auto*
    also have ... = (∫⁺ *x*. *f* (*T x*) ∗ (*nn-transfer-operator*^^*n*) *g x* ∂*M*)
      by (*rule nn-transfer-operator-intg*, *auto*)
    also have ... = (∫⁺ *x*. (λ*x*. *f* (*T x*)) ((*T*^^*n*) *x*) ∗ *g x* ∂*M*)
      by (*rule Suc.IH*, *auto*)
    also have ... = (∫⁺ *x*. *f* ((*T*^^(*Suc n*)) *x*) ∗ *g x* ∂*M*)
      apply (*rule nn-integral-cong*) **using** *funpow.simps*(*2*) **unfolding** *comp-def* **by** *auto*
    finally show *?case* **by** *auto*
  qed (*simp*)
  then show *?thesis* **using** *assms* **by** *auto*
qed

lemma *nn-transfer-operator-intg-Tn*:
  assumes *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
  shows (∫⁺ *x*. (*nn-transfer-operator*^^*n*) *g x* ∗ *f x* ∂*M*) = (∫⁺ *x*. *g x* ∗ *f* ((*T*^^*n*) *x*) ∂*M*)
using *nn-transfer-operator-intTn-g*[*OF assms*, *of n*] **by** (*simp add*: *algebra-simps*)

lemma *nn-transfer-operator-charact*:
  assumes ⋀*A*. *A* ∈ *sets M* ⟹ (∫⁺ *x*. *indicator A x* ∗ *g x* ∂*M*) = (∫⁺ *x*. *indicator A* (*T x*) ∗ *f x* ∂*M*) **and**
      [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
  shows *AE x in M*. *nn-transfer-operator f x* = *g x*
proof −
  have ∗:*set-nn-integral M A g* = *set-nn-integral M A* (*nn-transfer-operator f*) **if** [*measurable*]: *A* ∈ *sets M* **for** *A*
  proof −
    have *set-nn-integral M A g* = (∫⁺ *x*. *indicator A x* ∗ *g x* ∂*M*)
      using *mult.commute* **by** *metis*
    also have ... = (∫⁺ *x*. *indicator A* (*T x*) ∗ *f x* ∂*M*)
      using *assms*(*1*) **by** *auto*
    also have ... = (∫⁺ *x*. *indicator A x* ∗ *nn-transfer-operator f x* ∂*M*)
      by (*rule nn-transfer-operator-intg*[*symmetric*], *auto*)
    finally show *?thesis*
      using *mult.commute* **by** (*metis* (*no-types*, *lifting*) *nn-integral-cong*)
  qed
  show *?thesis*
   by (*rule sigma-finite-measure.density-unique2*, *auto simp add*: *sigma-finite-measure-axioms* ∗)
qed

When T is measure-preserving, $\hat{T}(f \circ T) = f$.

lemma (**in** *mpt*) *nn-transfer-operator-foT*:
assumes [*measurable*]: $f \in$ *borel-measurable M*
shows *AE x in M. nn-transfer-operator* (*f o T*) $x = f\ x$
proof −
have ∗: ($\int^+$ *x. indicator A x* ∗ *f x* ∂*M*) = ($\int^+$ *x. indicator A* (*T x*) ∗ *f* (*T x*) ∂*M*) **if** [*measurable*]: $A \in$ *sets M* **for** *A*
by (*subst T-nn-integral-preserving*[*symmetric*]) *auto*
show *?thesis*
by (*rule nn-transfer-operator-charact*) (*auto simp add*: *assms* ∗)
qed

In general, one only has $\hat{T}(f \circ T \cdot g) = f \cdot \hat{T} g$.

lemma *nn-transfer-operator-foT-g*:
assumes [*measurable*]: $f \in$ *borel-measurable M* $g \in$ *borel-measurable M*
shows *AE x in M. nn-transfer-operator* (λ*x. f* (*T x*) ∗ *g x*) *x* = *f x* ∗ *nn-transfer-operator g x*
proof −
have ∗: ($\int^+$ *x. indicator A x* ∗ (*f x* ∗ *nn-transfer-operator g x*) ∂*M*) = ($\int^+$ *x. indicator A* (*T x*) ∗ (*f* (*T x*) ∗ *g x*) ∂*M*)
if [*measurable*]: $A \in$ *sets M* **for** *A*
by (*simp add*: *mult.assoc*[*symmetric*] *nn-transfer-operator-intg*)
show *?thesis*
by (*rule nn-transfer-operator-charact*) (*auto simp add*: *assms* ∗)
qed

lemma *nn-transfer-operator-cmult*:
assumes [*measurable*]: $g \in$ *borel-measurable M*
shows *AE x in M. nn-transfer-operator* (λ*x. c* ∗ *g x*) *x* = *c* ∗ *nn-transfer-operator g x*
apply (*rule nn-transfer-operator-foT-g*) **using** *assms* **by** *auto*

lemma *nn-transfer-operator-zero*:
AE x in M. nn-transfer-operator (λ*x. 0*) *x* = *0*
using *nn-transfer-operator-cmult*[*of* λ*x. 0 0*] **by** *auto*

lemma *nn-transfer-operator-sum*:
assumes [*measurable*]: $f \in$ *borel-measurable M* $g \in$ *borel-measurable M*
shows *AE x in M. nn-transfer-operator* (λ*x. f x* + *g x*) *x* = *nn-transfer-operator f x* + *nn-transfer-operator g x*
proof (*rule nn-transfer-operator-charact*)
fix *A* **assume** [*measurable*]: $A \in$ *sets M*
have ($\int^+$ *x. indicator A x* ∗ (*nn-transfer-operator f x* + *nn-transfer-operator g x*) ∂*M*) =
($\int^+$ *x. indicator A x* ∗ *nn-transfer-operator f x* + *indicator A x* ∗ *nn-transfer-operator g x* ∂*M*)
by (*auto simp add*: *algebra-simps*)
also have ... = ($\int^+$*x. indicator A x* ∗ *nn-transfer-operator f x* ∂*M*) + ($\int^+$*x. indicator A x* ∗ *nn-transfer-operator g x* ∂*M*)
by (*rule nn-integral-add*, *auto*)

also have ... = ($\int^+ x.$ *indicator A (T x) ∗ f x ∂M*) + ($\int^+ x.$ *indicator A (T x) ∗ g x ∂M*)
by (*simp add*: *nn-transfer-operator-intg*)
also have ... = ($\int^+ x.$ *indicator A (T x) ∗ f x + indicator A (T x) ∗ g x ∂M*)
by (*rule nn-integral-add*[*symmetric*], *auto*)
also have ... = ($\int^+ x.$ *indicator A (T x) ∗ (f x + g x) ∂M*)
by (*auto simp add*: *algebra-simps*)
finally show ($\int^+ x.$ *indicator A x ∗ (nn-transfer-operator f x + nn-transfer-operator g x) ∂M*) = ($\int^+ x.$ *indicator A (T x) ∗ (f x + g x) ∂M*)
by *simp*
qed (*auto simp add*: *assms*)

lemma *nn-transfer-operator-cong*:
assumes *AE x in M. f x = g x*
and [*measurable*]: *f ∈ borel-measurable M g ∈ borel-measurable M*
shows *AE x in M. nn-transfer-operator f x = nn-transfer-operator g x*
apply (*rule nn-transfer-operator-charact*)
apply (*auto simp add*: *nn-transfer-operator-intg assms intro*!: *nn-integral-cong-AE*)
using *assms* **by** *auto*

lemma *nn-transfer-operator-mono*:
assumes *AE x in M. f x ≤ g x*
and [*measurable*]: *f ∈ borel-measurable M g ∈ borel-measurable M*
shows *AE x in M. nn-transfer-operator f x ≤ nn-transfer-operator g x*
proof −
define *h* **where** *h* = (*λx. g x − f x*)
have [*measurable*]: *h ∈ borel-measurable M* **unfolding** *h-def* **by** *simp*
have ∗: *AE x in M. g x = f x + h x* **unfolding** *h-def* **using** *assms*(*1*) **by** (*auto simp*: *ennreal-ineq-diff-add*)
have *AE x in M. nn-transfer-operator g x = nn-transfer-operator* (*λx. f x + h x*) *x*
by (*rule nn-transfer-operator-cong*) (*auto simp add*: ∗ *assms*)
moreover have *AE x in M. nn-transfer-operator* (*λx. f x + h x*) *x = nn-transfer-operator f x + nn-transfer-operator h x*
by (*rule nn-transfer-operator-sum*) (*auto simp add*: *assms*)
ultimately have *AE x in M. nn-transfer-operator g x = nn-transfer-operator f x + nn-transfer-operator h x* **by** *auto*
then show *?thesis* **by** *force*
qed

12.2 The transfer operator on real functions

Once the transfer operator of positive functions is defined, the definition for real-valued functions follows readily, by taking the difference of positive and negative parts.

definition *real-transfer-operator* :: (*'a ⇒ real*) ⇒ (*'a ⇒ real*) **where**
real-transfer-operator f =
(*λx. enn2real*(*nn-transfer-operator* (*λx. ennreal* (*f x*)) *x*) − *enn2real*(*nn-transfer-operator* (*λx. ennreal* (−*f x*)) *x*))

lemma *borel-measurable-transfer-operator* [*measurable*]:
real-transfer-operator f ∈ *borel-measurable M*
unfolding *real-transfer-operator-def* **by** *auto*

lemma *borel-measurable-transfer-operator-iterates* [*measurable*]:
assumes [*measurable*]: *f* ∈ *borel-measurable M*
shows (*real-transfer-operator*^^*n*) *f* ∈ *borel-measurable M*
by (*cases n*, *auto*)

lemma *real-transfer-operator-abs*:
assumes [*measurable*]: *f* ∈ *borel-measurable M*
shows *AE x in M*. *abs* (*real-transfer-operator f x*) ≤ *nn-transfer-operator* (λ*x*. *ennreal* (*abs*(*f x*))) *x*
proof −
define *fp* **where** *fp* = (λ*x*. *ennreal* (*f x*))
define *fm* **where** *fm* = (λ*x*. *ennreal* (− *f x*))
have [*measurable*]: *fp* ∈ *borel-measurable M fm* ∈ *borel-measurable M* **unfolding** *fp-def fm-def* **by** *auto*
have *eq*: ⋀*x*. *ennreal* |*f x*| = *fp x* + *fm x* **unfolding** *fp-def fm-def* **by** (*simp add*: *abs-real-def ennreal-neg*)

{
fix *x* **assume** *H*: *nn-transfer-operator* (λ*x*. *fp x* + *fm x*) *x* = *nn-transfer-operator fp x* + *nn-transfer-operator fm x*
have |*real-transfer-operator f x*| ≤ |*enn2real*(*nn-transfer-operator fp x*)| + |*enn2real*(*nn-transfer-operator fm x*)|
unfolding *real-transfer-operator-def fp-def fm-def* **by** (*auto intro*: *abs-triangle-ineq4 simp del*: *enn2real-nonneg*)
from *ennreal-leI*[*OF this*]
have *abs*(*real-transfer-operator f x*) ≤ *nn-transfer-operator fp x* + *nn-transfer-operator fm x*
by *simp* (*metis add.commute ennreal-enn2real le-iff-add not-le top-unique*)
also have ... = *nn-transfer-operator* (λ*x*. *fp x* + *fm x*) *x* **using** *H* **by** *simp*
finally have *abs*(*real-transfer-operator f x*) ≤ *nn-transfer-operator* (λ*x*. *fp x* + *fm x*) *x* **by** *simp*
}
moreover have *AE x in M*. *nn-transfer-operator* (λ*x*. *fp x* + *fm x*) *x* = *nn-transfer-operator fp x* + *nn-transfer-operator fm x*
by (*rule nn-transfer-operator-sum*) (*auto simp add*: *fp-def fm-def*)
ultimately have *AE x in M*. *abs*(*real-transfer-operator f x*) ≤ *nn-transfer-operator* (λ*x*. *fp x* + *fm x*) *x*
by *auto*
then show *?thesis* **using** *eq* **by** *simp*
qed

The next lemma shows that the transfer operator as we have defined it satisfies the basic duality relation $\int \hat{T} f \cdot g = \int f \cdot g \circ T$. It follows from the same relation for nonnegative functions, and splitting into positive and negative parts.

Moreover, this relation characterizes the transfer operator. Hence, once this lemma is proved, we will never come back to the original definition of the transfer operator.

lemma *real-transfer-operator-intg-fpos*:
assumes *integrable M* ($\lambda x.$ *f* (*T x*) $*$ *g x*) **and** *f-pos*[*simp*]: $\bigwedge x.$ *f x* $\geq$ *0* **and**
[*measurable*]: *f* $\in$ *borel-measurable M g* $\in$ *borel-measurable M*
shows *integrable M* ($\lambda x.$ *f x* $*$ *real-transfer-operator g x*)
($\int$ *x. f x* $*$ *real-transfer-operator g x* ∂M) = ($\int$ *x. f* (*T x*) $*$ *g x* ∂M)
proof $-$
define *gp* **where** *gp* = ($\lambda x.$ *ennreal* (*g x*))
define *gm* **where** *gm* = ($\lambda x.$ *ennreal* ($-$ *g x*))
have [*measurable*]: *gp* $\in$ *borel-measurable M gm* $\in$ *borel-measurable M* **unfolding** *gp-def gm-def* **by** *auto*
define *h* **where** *h* = ($\lambda x.$ *ennreal*(*abs*(*g x*)))
have *hgpgm*: $\bigwedge x.$ *h x* = *gp x* + *gm x* **unfolding** *gp-def gm-def h-def* **by** (*simp add*: *abs-real-def ennreal-neg*)
have [*measurable*]: *h* $\in$ *borel-measurable M* **unfolding** *h-def* **by** *simp*
have *pos*[*simp*]: $\bigwedge x.$ *h x* $\geq$ *0* $\bigwedge x.$ *gp x* $\geq$ *0* $\bigwedge x.$ *gm x* $\geq$ *0* **unfolding** *h-def gp-def gm-def* **by** *simp-all*
have *gp-real*: $\bigwedge x.$ *enn2real*(*gp x*) = *max* (*g x*) *0*
unfolding *gp-def* **by** (*simp add*: *max-def ennreal-neg*)
have *gm-real*: $\bigwedge x.$ *enn2real*(*gm x*) = *max* ($-$*g x*) *0*
unfolding *gm-def* **by** (*simp add*: *max-def ennreal-neg*)

have ($\int^+$ *x. norm*(*f* (*T x*) $*$ *max* (*g x*) *0*) ∂M) $\leq$ ($\int^+$ *x. norm*(*f* (*T x*) $*$ *g x*) ∂M)
by (*simp add*: *nn-integral-mono*)
also have ... $<$ ∞ **using** *assms*(*1*) **by** (*simp add*: *integrable-iff-bounded*)
finally have ($\int^+$ *x. norm*(*f* (*T x*) $*$ *max* (*g x*) *0*) ∂M) $<$ ∞ **by** *simp*
then have *int1*: *integrable M* ($\lambda x.$ *f* (*T x*) $*$ *max* (*g x*) *0*) **by** (*simp add*: *integrableI-bounded*)

have ($\int^+$ *x. norm*(*f* (*T x*) $*$ *max* ($-$*g x*) *0*) ∂M) $\leq$ ($\int^+$ *x. norm*(*f* (*T x*) $*$ *g x*) ∂M)
by (*simp add*: *nn-integral-mono*)
also have ... $<$ ∞ **using** *assms*(*1*) **by** (*simp add*: *integrable-iff-bounded*)
finally have ($\int^+$ *x. norm*(*f* (*T x*) $*$ *max* ($-$*g x*) *0*) ∂M) $<$ ∞ **by** *simp*
then have *int2*: *integrable M* ($\lambda x.$ *f* (*T x*) $*$ *max* ($-$*g x*) *0*) **by** (*simp add*: *integrableI-bounded*)

have ($\int^+$*x. f x* $*$ *nn-transfer-operator h x* ∂M) = ($\int^+$*x. f* (*T x*) $*$ *h x* ∂M)
by (*rule nn-transfer-operator-intg*) *auto*
also have ... = $\int^+$ *x. ennreal* (*f* (*T x*) $*$ *max* (*g x*) *0* + *f* (*T x*) $*$ *max* ($-$ *g x*) *0*) ∂M
unfolding *h-def*
by (*intro nn-integral-cong*)(*auto simp*: *ennreal-mult*[*symmetric*] *abs-mult split*: *split-max*)

also have ... $< \infty$
using *Bochner-Integration.integrable-add*[*OF int1 int2*, *THEN integrableD*(*2*)]
by (*auto simp add*: *less-top*)
finally have ∗: ($\int^+ x.\ f\ x * $ *nn-transfer-operator h x* ∂M) $< \infty$ **by** *simp*

have ($\int^+ x.$ *norm*($f\ x *$ *real-transfer-operator g x*) ∂M) = ($\int^+ x.\ f\ x *$ *abs*(*real-transfer-operator g x*) ∂M)
by (*simp add*: *abs-mult*)
also have ... $\leq$ ($\int^+ x.\ f\ x *$ *nn-transfer-operator h x* ∂M)
proof (*rule nn-integral-mono-AE*)
{
fix *x* **assume** ∗: *abs*(*real-transfer-operator g x*) $\leq$ *nn-transfer-operator h x*
have *ennreal* ($f\ x *$ |*real-transfer-operator g x*|) = $f\ x *$ *ennreal*(|*real-transfer-operator g x*|)
by (*simp add*: *ennreal-mult*)
also have ... $\leq f\ x *$ *nn-transfer-operator h x*
using ∗ **by** (*auto intro*!: *mult-left-mono*)
finally have *ennreal* ($f\ x *$ |*real-transfer-operator g x*|) $\leq f\ x *$ *nn-transfer-operator h x*
by *simp*
}
then show *AE x in M*. *ennreal* ($f\ x *$ |*real-transfer-operator g x*|) $\leq f\ x *$ *nn-transfer-operator h x*
using *real-transfer-operator-abs*[*OF assms*(*4*)] *h-def* **by** *auto*
qed
finally have ∗∗: ($\int^+ x.$ *norm*($f\ x *$ *real-transfer-operator g x*) ∂M) $< \infty$ **using** ∗ **by** *auto*
show *integrable M* ($\lambda x.\ f\ x *$ *real-transfer-operator g x*)
using ∗∗ **by** (*intro integrableI-bounded*) *auto*

have ($\int^+ x.\ f\ x *$ *nn-transfer-operator gp x* ∂M) $\leq$ ($\int^+ x.\ f\ x *$ *nn-transfer-operator h x* ∂M)
proof (*rule nn-integral-mono-AE*)
have *AE x in M*. *nn-transfer-operator gp x* $\leq$ *nn-transfer-operator h x*
by (*rule nn-transfer-operator-mono*) (*auto simp add*: *hgpgm*)
then show *AE x in M*. $f\ x *$ *nn-transfer-operator gp x* $\leq f\ x *$ *nn-transfer-operator h x*
by (*auto simp*: *mult-left-mono*)
qed
then have *a*: ($\int^+ x.\ f\ x *$ *nn-transfer-operator gp x* ∂M) $< \infty$
using ∗ **by** *auto*
have *ennreal*(*norm*($f\ x *$ *enn2real*(*nn-transfer-operator gp x*))) $\leq f\ x *$ *nn-transfer-operator gp x* **for** *x*
by (*auto simp add*: *ennreal-mult intro*!: *mult-left-mono*)
(*metis enn2real-ennreal enn2real-nonneg le-cases le-ennreal-iff*)
then have ($\int^+ x.$ *norm*($f\ x *$ *enn2real*(*nn-transfer-operator gp x*)) ∂M) $\leq$ ($\int^+ x.\ f\ x *$ *nn-transfer-operator gp x* ∂M)
by (*simp add*: *nn-integral-mono*)
then have ($\int^+ x.$ *norm*($f\ x *$ *enn2real*(*nn-transfer-operator gp x*)) ∂M) $< \infty$

using *a* **by** *auto*
  then have *gp-int*: *integrable M* (*λx. f x* ∗ *enn2real*(*nn-transfer-operator gp x*))
by (*simp add*: *integrableI-bounded*)
  have *gp-fin*: *AE x in M. f x* ∗ *nn-transfer-operator gp x* ≠ ∞
    apply (*rule nn-integral-PInf-AE*) **using** *a* **by** *auto*

  have (∫ *x. f x* ∗ *enn2real*(*nn-transfer-operator gp x*) ∂*M*) = *enn2real* (∫ + *x. f x* ∗ *enn2real*(*nn-transfer-operator gp x*) ∂*M*)
    by (*rule integral-eq-nn-integral*) *auto*
  also have ... = *enn2real*(∫ + *x. ennreal*(*f* (*T x*) ∗ *enn2real*(*gp x*)) ∂*M*)
  proof −
    {
      fix *x* **assume** *f x* ∗ *nn-transfer-operator gp x* ≠ ∞
      then have *ennreal* (*f x* ∗ *enn2real* (*nn-transfer-operator gp x*)) = *ennreal* (*f x*) ∗ *nn-transfer-operator gp x*
          by (*auto simp add*: *ennreal-mult ennreal-mult-eq-top-iff less-top intro*!: *ennreal-mult-left-cong*)
    }
    then have *AE x in M. ennreal* (*f x* ∗ *enn2real* (*nn-transfer-operator gp x*)) = *ennreal* (*f x*) ∗ *nn-transfer-operator gp x*
      using *gp-fin* **by** *auto*
    then have (∫ + *x. f x* ∗ *enn2real*(*nn-transfer-operator gp x*) ∂*M*) = (∫ + *x. f x* ∗ *nn-transfer-operator gp x* ∂*M*)
      by (*rule nn-integral-cong-AE*)
    also have ... = (∫ + *x. f* (*T x*) ∗ *gp x* ∂*M*)
      by (*rule nn-transfer-operator-intg*) (*auto simp add*: *gp-def*)
    also have ... = (∫ + *x. ennreal*(*f* (*T x*) ∗ *enn2real*(*gp x*)) ∂*M*)
      by (*rule nn-integral-cong-AE*) (*auto simp*: *ennreal-mult gp-def*)
    finally have (∫ + *x. f x* ∗ *enn2real*(*nn-transfer-operator gp x*) ∂*M*) = (∫ + *x. ennreal*(*f* (*T x*) ∗ *enn2real*(*gp x*)) ∂*M*) **by** *simp*
    then show *?thesis* **by** *simp*
  qed
  also have ... = (∫ *x. f* (*T x*) ∗ *enn2real*(*gp x*) ∂*M*)
    by (*rule integral-eq-nn-integral*[*symmetric*]) (*auto simp add*: *gp-def*)
  finally have *gp-expr*: (∫ *x. f x* ∗ *enn2real*(*nn-transfer-operator gp x*) ∂*M*) = (∫ *x. f* (*T x*) ∗ *enn2real*(*gp x*) ∂*M*) **by** *simp*

  have (∫ +*x. f x* ∗ *nn-transfer-operator gm x* ∂*M*) ≤ (∫ +*x. f x* ∗ *nn-transfer-operator h x* ∂*M*)
  proof (*rule nn-integral-mono-AE*)
    have *AE x in M. nn-transfer-operator gm x* ≤ *nn-transfer-operator h x*
      by (*rule nn-transfer-operator-mono*) (*auto simp add*: *hgpgm*)
    then show *AE x in M. f x* ∗ *nn-transfer-operator gm x* ≤ *f x* ∗ *nn-transfer-operator h x*
      by (*auto simp*: *mult-left-mono*)
  qed
  then have *a*: (∫ +*x. f x* ∗ *nn-transfer-operator gm x* ∂*M*) < ∞
    using ∗ **by** *auto*
  have ⋀*x. ennreal*(*norm*(*f x* ∗ *enn2real*(*nn-transfer-operator gm x*))) ≤ *f x* ∗

nn-transfer-operator gm x
  by (*auto simp add*: *ennreal-mult intro*!: *mult-left-mono*)
    (*metis enn2real-ennreal enn2real-nonneg le-cases le-ennreal-iff*)
 then have ($\int^+ x.\ norm(f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x))\ \partial M) \leq (\int^+ x.\ f\ x * nn\text{-}transfer\text{-}operator\ gm\ x\ \partial M$)
  by (*simp add*: *nn-integral-mono*)
 then have ($\int^+ x.\ norm(f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x))\ \partial M) < \infty$ **using** *a* **by** *auto*
 then have *gm-int*: *integrable M* ($\lambda x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)$) **by** (*simp add*: *integrableI-bounded*)
 have *gm-fin*: *AE x in M*. $f\ x * nn\text{-}transfer\text{-}operator\ gm\ x \neq \infty$
  apply (*rule nn-integral-PInf-AE*) **using** *a* **by** *auto*

 have ($\int x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M) = enn2real\ (\int^+ x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M$)
  by (*rule integral-eq-nn-integral*) *auto*
 also have ... = $enn2real(\int^+ x.\ ennreal(f\ (T\ x) * enn2real(gm\ x))\ \partial M)$
 proof −
  {
   fix *x* **assume** $f\ x * nn\text{-}transfer\text{-}operator\ gm\ x \neq \infty$
   then have $ennreal\ (f\ x * enn2real\ (nn\text{-}transfer\text{-}operator\ gm\ x)) = ennreal\ (f\ x) * nn\text{-}transfer\text{-}operator\ gm\ x$
     by (*auto simp add*: *ennreal-mult ennreal-mult-eq-top-iff less-top intro*!: *ennreal-mult-left-cong*)
  }
  then have *AE x in M*. $ennreal\ (f\ x * enn2real\ (nn\text{-}transfer\text{-}operator\ gm\ x)) = ennreal\ (f\ x) * nn\text{-}transfer\text{-}operator\ gm\ x$
   using *gm-fin* **by** *auto*
  then have ($\int^+ x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M) = (\int^+ x.\ f\ x * nn\text{-}transfer\text{-}operator\ gm\ x\ \partial M$)
   by (*rule nn-integral-cong-AE*)
  also have ... = ($\int^+ x.\ f\ (T\ x) * gm\ x\ \partial M$)
   by (*rule nn-transfer-operator-intg*) (*auto simp add*: *gm-def*)
  also have ... = ($\int^+ x.\ ennreal(f\ (T\ x) * enn2real(gm\ x))\ \partial M$)
   by (*rule nn-integral-cong-AE*) (*auto simp*: *ennreal-mult gm-def*)
  finally have ($\int^+ x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M) = (\int^+ x.\ ennreal(f\ (T\ x) * enn2real(gm\ x))\ \partial M$) **by** *simp*
  then show *?thesis* **by** *simp*
 qed
 also have ... = ($\int x.\ f\ (T\ x) * enn2real(gm\ x)\ \partial M$)
  by (*rule integral-eq-nn-integral*[*symmetric*]) (*auto simp add*: *gm-def*)
 finally have *gm-expr*: ($\int x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M) = (\int x.\ f\ (T\ x) * enn2real(gm\ x)\ \partial M$) **by** *simp*

 have ($\int x.\ f\ x * real\text{-}transfer\text{-}operator\ g\ x\ \partial M) = (\int x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gp\ x) - f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M$)
  unfolding *real-transfer-operator-def gp-def gm-def* **by** (*simp add*: *right-diff-distrib*)
 also have ... = ($\int x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gp\ x)\ \partial M) - (\int x.\ f\ x * enn2real(nn\text{-}transfer\text{-}operator\ gm\ x)\ \partial M$)

by (*rule Bochner-Integration.integral-diff*) (*simp-all add*: *gp-int gm-int*)
also have ... = (∫ *x*. *f* (*T x*) ∗ *enn2real*(*gp x*) ∂*M*) − (∫ *x*. *f* (*T x*) ∗ *enn2real*(*gm x*) ∂*M*)
using *gp-expr gm-expr* **by** *simp*
also have ... = (∫ *x*. *f* (*T x*) ∗ *max* (*g x*) *0* ∂*M*) − (∫ *x*. *f* (*T x*) ∗ *max* (−*g x*) *0* ∂*M*)
using *gp-real gm-real* **by** *simp*
also have ... = (∫ *x*. *f* (*T x*) ∗ *max* (*g x*) *0* − *f* (*T x*) ∗ *max* (−*g x*) *0* ∂*M*)
by (*rule Bochner-Integration.integral-diff*[*symmetric*]) (*simp-all add*: *int1 int2*)
also have ... = (∫ *x*. *f* (*T x*) ∗ *g x* ∂*M*)
by (*metis* (*mono-tags*, *opaque-lifting*) *diff-0 diff-zero eq-iff max.cobounded2 max-def minus-minus neg-le-0-iff-le right-diff-distrib*)
finally show (∫ *x*. *f x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *f* (*T x*) ∗ *g x* ∂*M*)
by *simp*
qed

lemma *real-transfer-operator-intg*:
assumes *integrable M* (λ*x*. *f* (*T x*) ∗ *g x*) **and**
[*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
shows *integrable M* (λ*x*. *f x* ∗ *real-transfer-operator g x*)
(∫ *x*. *f x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *f* (*T x*) ∗ *g x* ∂*M*)
proof −
define *fp* **where** *fp* = (λ*x*. *max* (*f x*) *0*)
define *fm* **where** *fm* = (λ*x*. *max* (−*f x*) *0*)
have [*measurable*]: *fp* ∈ *borel-measurable M fm* ∈ *borel-measurable M*
unfolding *fp-def fm-def* **by** *simp-all*

have (∫⁺ *x*. *norm*(*fp* (*T x*) ∗ *g x*) ∂*M*) ≤ (∫⁺ *x*. *norm*(*f* (*T x*) ∗ *g x*) ∂*M*)
by (*simp add*: *fp-def nn-integral-mono*)
also have ... < ∞ **using** *assms*(*1*) **by** (*simp add*: *integrable-iff-bounded*)
finally have (∫⁺ *x*. *norm*(*fp* (*T x*) ∗ *g x*) ∂*M*) < ∞ **by** *simp*
then have *intp*: *integrable M* (λ*x*. *fp* (*T x*) ∗ *g x*) **by** (*simp add*: *integrableI-bounded*)
moreover have ⋀*x*. *fp x* ≥ *0* **unfolding** *fp-def* **by** *simp*
ultimately have *Rp*: *integrable M* (λ*x*. *fp x* ∗ *real-transfer-operator g x*)
(∫ *x*. *fp x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *fp* (*T x*) ∗ *g x* ∂*M*)
using *real-transfer-operator-intg-fpos* **by** *auto*

have (∫⁺ *x*. *norm*(*fm* (*T x*) ∗ *g x*) ∂*M*) ≤ (∫⁺ *x*. *norm*(*f* (*T x*) ∗ *g x*) ∂*M*)
by (*simp add*: *fm-def nn-integral-mono*)
also have ... < ∞ **using** *assms*(*1*) **by** (*simp add*: *integrable-iff-bounded*)
finally have (∫⁺ *x*. *norm*(*fm* (*T x*) ∗ *g x*) ∂*M*) < ∞ **by** *simp*
then have *intm*: *integrable M* (λ*x*. *fm* (*T x*) ∗ *g x*) **by** (*simp add*: *integrableI-bounded*)
moreover have ⋀*x*. *fm x* ≥ *0* **unfolding** *fm-def* **by** *simp*
ultimately have *Rm*: *integrable M* (λ*x*. *fm x* ∗ *real-transfer-operator g x*)
(∫ *x*. *fm x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *fm* (*T x*) ∗ *g x* ∂*M*)
using *real-transfer-operator-intg-fpos* **by** *auto*

have *integrable M* (λ*x*. *fp x* ∗ *real-transfer-operator g x* − *fm x* ∗ *real-transfer-operator*

g x)
  using *Rp*(*1*) *Rm*(*1*) *integrable-diff* **by** *simp*
 moreover have ∗: $\bigwedge x.$ *f x* ∗ *real-transfer-operator g x* = *fp x* ∗ *real-transfer-operator g x* − *fm x* ∗ *real-transfer-operator g x*
  unfolding *fp-def fm-def* **by** (*simp add*: *max-def*)
 ultimately show *integrable M* (λ*x*. *f x* ∗ *real-transfer-operator g x*)
  by *simp*

 have (∫ *x*. *f x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *fp x* ∗ *real-transfer-operator g x* − *fm x* ∗ *real-transfer-operator g x* ∂*M*)
  using ∗ **by** *simp*
 also have ... = (∫ *x*. *fp x* ∗ *real-transfer-operator g x* ∂*M*) − (∫ *x*. *fm x* ∗ *real-transfer-operator g x* ∂*M*)
  using *Rp*(*1*) *Rm*(*1*) **by** *simp*
 also have ... = (∫ *x*. *fp* (*T x*) ∗ *g x* ∂*M*) − (∫ *x*. *fm* (*T x*) ∗ *g x* ∂*M*)
  using *Rp*(*2*) *Rm*(*2*) **by** *simp*
 also have ... = (∫ *x*. *fp* (*T x*) ∗ *g x* − *fm* (*T x*) ∗ *g x* ∂*M*)
  using *intm intp* **by** *simp*
 also have ... = (∫ *x*. *f* (*T x*) ∗ *g x* ∂*M*)
   unfolding *fp-def fm-def* **by** (*metis* (*no-types*, *opaque-lifting*) *diff-0 diff-zero max.commute*
  max-def minus-minus mult.commute neg-le-iff-le right-diff-distrib)
 finally show (∫ *x*. *f x* ∗ *real-transfer-operator g x* ∂*M*) = (∫ *x*. *f* (*T x*) ∗ *g x* ∂*M*) **by** *simp*
qed

lemma *real-transfer-operator-int* [*intro*]:
 assumes *integrable M f*
 shows *integrable M* (*real-transfer-operator f*)
   (∫ *x*. *real-transfer-operator f x* ∂*M*) = (∫ *x*. *f x* ∂*M*)
using *real-transfer-operator-intg*[**where** *?f* = λ*x*. *1* **and** *?g* = *f*] *assms* **by** *auto*

lemma *real-transfer-operator-charact*:
 assumes $\bigwedge A.$ *A* ∈ *sets M* ⟹ (∫ *x*. *indicator A x* ∗ *g x* ∂*M*) = (∫ *x*. *indicator A* (*T x*) ∗ *f x* ∂*M*)
   and [*measurable*]: *integrable M f integrable M g*
 shows *AE x in M*. *real-transfer-operator f x* = *g x*
proof (*rule AE-symmetric*[*OF density-unique-real*])
 fix *A* **assume** [*measurable*]: *A* ∈ *sets M*
 have *set-lebesgue-integral M A* (*real-transfer-operator f*) = (∫ *x*. *indicator A x* ∗ *real-transfer-operator f x* ∂*M*)
  unfolding *set-lebesgue-integral-def* **by** *auto*
 also have ... = (∫ *x*. *indicator A* (*T x*) ∗ *f x* ∂*M*)
  apply (*rule real-transfer-operator-intg*, *auto*)
  by (*rule Bochner-Integration.integrable-bound*[*of* - λ*x*. *abs*(*f x*)], *auto simp add*: *assms indicator-def*)
 also have ... = *set-lebesgue-integral M A g*
  unfolding *set-lebesgue-integral-def* **using** *assms*(*1*)[*OF* ‹*A* ∈ *sets M*›] **by** *auto*
 finally show *set-lebesgue-integral M A g* = *set-lebesgue-integral M A* (*real-transfer-operator*

f)
  by *simp*
qed (*auto simp add*: *assms real-transfer-operator-int*)

lemma (**in** *mpt*) *real-transfer-operator-foT*:
 assumes *integrable M f*
 shows *AE x in M. real-transfer-operator* (*f o T*) *x* = *f x*
proof −
 have ∗: (∫ *x. indicator A x* ∗ *f x* ∂*M*) = (∫ *x. indicator A* (*T x*) ∗ *f* (*T x*) ∂*M*)
if [*measurable*]: *A* ∈ *sets M* **for** *A*
  apply (*subst T-integral-preserving*)
  using *integrable-real-mult-indicator*[*OF that assms*] **by** (*auto simp add*: *mult.commute*)
 show *?thesis*
  apply (*rule real-transfer-operator-charact*)
  using *assms* ∗ **by** (*auto simp add*: *comp-def T-integral-preserving*)
qed

lemma *real-transfer-operator-foT-g*:
 assumes [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M integrable M* (λ*x. f* (*T x*) ∗ *g x*)
 shows *AE x in M. real-transfer-operator* (λ*x. f* (*T x*) ∗ *g x*) *x* = *f x* ∗ *real-transfer-operator g x*
proof −
 have ∗: (∫ *x. indicator A x* ∗ (*f x* ∗ *real-transfer-operator g x*) ∂*M*) = (∫ *x. indicator A* (*T x*) ∗ (*f* (*T x*) ∗ *g x*) ∂*M*)
  if [*measurable*]: *A* ∈ *sets M* **for** *A*
  apply (*simp add*: *mult.assoc*[*symmetric*])
  apply (*subst real-transfer-operator-intg*)
  apply (*rule Bochner-Integration.integrable-bound*[*of* - (λ*x. f* (*T x*) ∗ *g x*)])
  by (*auto simp add*: *assms indicator-def*)
 show *?thesis*
  by (*rule real-transfer-operator-charact*) (*auto simp add*: *assms* ∗ *intro*!: *real-transfer-operator-intg*)
qed

lemma *real-transfer-operator-add* [*intro*]:
 assumes [*measurable*]: *integrable M f integrable M g*
 shows *AE x in M. real-transfer-operator* (λ*x. f x* + *g x*) *x* = *real-transfer-operator f x* + *real-transfer-operator g x*
proof (*rule real-transfer-operator-charact*)
 have *integrable M* (*real-transfer-operator f*) *integrable M* (*real-transfer-operator g*)
  using *real-transfer-operator-int*(*1*) *assms* **by** *auto*
 then show *integrable M* (λ*x. real-transfer-operator f x* + *real-transfer-operator g x*)
  by *auto*

 fix *A* **assume** [*measurable*]: *A* ∈ *sets M*
 have *intAf*: *integrable M* (λ*x. indicator A* (*T x*) ∗ *f x*)
  apply (*rule Bochner-Integration.integrable-bound*[*OF assms*(*1*)]) **unfolding**

indicator-def **by** *auto*
have *intAg*: *integrable M* (*λx. indicator A* (*T x*) ∗ *g x*)
apply (*rule Bochner-Integration.integrable-bound*[*OF assms*(*2*)]) **unfolding** *indicator-def* **by** *auto*

have (∫ *x. indicator A x* ∗ (*real-transfer-operator f x* + *real-transfer-operator g x*)∂*M*)
= (∫ *x. indicator A x* ∗ *real-transfer-operator f x* + *indicator A x*∗ *real-transfer-operator g x* ∂*M*)
by (*simp add*: *algebra-simps*)
also have ... = (∫ *x. indicator A x* ∗ *real-transfer-operator f x* ∂*M*) + (∫ *x. indicator A x* ∗ *real-transfer-operator g x* ∂*M*)
apply (*rule Bochner-Integration.integral-add*)
using *integrable-real-mult-indicator*[*OF* ‹*A* ∈ *sets M*› *real-transfer-operator-int*(*1*)[*OF assms*(*1*)]]
integrable-real-mult-indicator[*OF* ‹*A* ∈ *sets M*› *real-transfer-operator-int*(*1*)[*OF assms*(*2*)]]
by (*auto simp add*: *mult.commute*)
also have ... = (∫ *x. indicator A* (*T x*) ∗ *f x* ∂*M*) + (∫ *x. indicator A* (*T x*) ∗ *g x* ∂*M*)
using *real-transfer-operator-intg*(*2*) *assms* ‹*A* ∈ *sets M*› *intAf intAg* **by** *auto*
also have ... = (∫ *x. indicator A* (*T x*) ∗ *f x* + *indicator A* (*T x*) ∗ *g x* ∂*M*)
by (*rule Bochner-Integration.integral-add*[*symmetric*]) (*auto simp add*: *assms* ‹*A* ∈ *sets M*› *intAf intAg*)
also have ... = ∫ *x. indicator A* (*T x*) ∗ (*f x* + *g x*)∂*M*
by (*simp add*: *algebra-simps*)
finally show (∫ *x. indicator A x* ∗ (*real-transfer-operator f x* + *real-transfer-operator g x*)∂*M*) = ∫ *x. indicator A* (*T x*) ∗ (*f x* + *g x*)∂*M*
by *simp*
qed (*auto simp add*: *assms*)

lemma *real-transfer-operator-cong*:
assumes *ae*: *AE x in M. f x* = *g x* **and** [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
shows *AE x in M. real-transfer-operator f x* = *real-transfer-operator g x*
proof −
have *AE x in M. nn-transfer-operator* (*λx. ennreal* (*f x*)) *x* = *nn-transfer-operator* (*λx. ennreal* (*g x*)) *x*
apply (*rule nn-transfer-operator-cong*) **using** *assms* **by** *auto*
moreover have *AE x in M. nn-transfer-operator* (*λx. ennreal* (−*f x*)) *x* = *nn-transfer-operator* (*λx. ennreal*(−*g x*)) *x*
apply (*rule nn-transfer-operator-cong*) **using** *assms* **by** *auto*
ultimately show *AE x in M. real-transfer-operator f x* = *real-transfer-operator g x*
unfolding *real-transfer-operator-def* **by** *auto*
qed

lemma *real-transfer-operator-cmult* [*intro*, *simp*]:
fixes *c*::*real*

assumes *integrable M f*
shows *AE x in M. real-transfer-operator* (*λx. c ∗ f x*) *x* = *c ∗ real-transfer-operator f x*
by (*rule real-transfer-operator-foT-g*) (*auto simp add*: *assms borel-measurable-integrable*)

lemma *real-transfer-operator-cdiv* [*intro*, *simp*]:
fixes *c*::*real*
assumes *integrable M f*
shows *AE x in M. real-transfer-operator* (*λx. f x / c*) *x* = *real-transfer-operator f x / c*
using *real-transfer-operator-cmult*[*of - 1/c*, *OF assms*] **by** (*auto simp add*: *divide-simps*)

lemma *real-transfer-operator-diff* [*intro*, *simp*]:
assumes [*measurable*]: *integrable M f integrable M g*
shows *AE x in M. real-transfer-operator* (*λx. f x − g x*) *x* = *real-transfer-operator f x − real-transfer-operator g x*
proof −
have *AE x in M. real-transfer-operator* (*λx. f x* + (− *g x*)) *x* = *real-transfer-operator f x* + *real-transfer-operator* (*λx.* −*g x*) *x*
using *real-transfer-operator-add*[**where** *?f* = *f* **and** *?g* = *λx.* − *g x*] *assms* **by** *auto*
moreover have *AE x in M. real-transfer-operator* (*λx.* −*g x*) *x* = − *real-transfer-operator g x*
using *real-transfer-operator-cmult*[**where** *?f* = *g* **and** *?c* = −*1*] *assms*(*2*) **by** *auto*
ultimately show *?thesis* **by** *auto*
qed

lemma *real-transfer-operator-pos* [*intro*]:
assumes *AE x in M. f x* ≥ *0* **and** [*measurable*]: *f* ∈ *borel-measurable M*
shows *AE x in M. real-transfer-operator f x* ≥ *0*
proof −
define *g* **where** *g* = (*λx. max* (*f x*) *0*)
have *AE x in M. f x* = *g x* **using** *assms g-def* **by** *auto*
then have ∗: *AE x in M. real-transfer-operator f x* = *real-transfer-operator g x* **using** *real-transfer-operator-cong g-def* **by** *auto*

have ⋀*x. g x* ≥ *0* **unfolding** *g-def* **by** *simp*
then have (*λx. ennreal*(−*g x*)) = (*λx. 0*)
by (*simp add*: *ennreal-neg*)
then have *AE x in M. nn-transfer-operator* (*λx. ennreal*(−*g x*)) *x* = *0*
using *nn-transfer-operator-zero* **by** *simp*
then have *AE x in M. real-transfer-operator g x* = *enn2real*(*nn-transfer-operator* (*λx. ennreal* (*g x*)) *x*)
unfolding *real-transfer-operator-def* **by** *auto*
then have *AE x in M. real-transfer-operator g x* ≥ *0* **by** *auto*
then show *?thesis* **using** ∗ **by** *auto*
qed

lemma *real-transfer-operator-mono*:
assumes *AE x in M*. $f\ x \leq g\ x$ **and** [*measurable*]: *integrable M f integrable M g*
shows *AE x in M*. *real-transfer-operator f x* $\leq$ *real-transfer-operator g x*
proof −
have *AE x in M*. *real-transfer-operator g x* − *real-transfer-operator f x* = *real-transfer-operator* ($\lambda x.\ g\ x - f\ x$) *x*
by (*rule AE-symmetric*[*OF real-transfer-operator-diff*], *auto simp add*: *assms*)
moreover have *AE x in M*. *real-transfer-operator* ($\lambda x.\ g\ x - f\ x$) $x \geq 0$
by (*rule real-transfer-operator-pos*, *auto simp add*: *assms*(*1*))
ultimately have *AE x in M*. *real-transfer-operator g x* − *real-transfer-operator f x* ≥ 0 **by** *auto*
then show *?thesis* **by** *auto*
qed

lemma *real-transfer-operator-sum* [*intro*, *simp*]:
fixes $f::'b \Rightarrow 'a \Rightarrow real$
assumes [*measurable*]: $\bigwedge i.$ *integrable M* (*f i*)
shows *AE x in M*. *real-transfer-operator* ($\lambda x. \sum i \in I.\ f\ i\ x$) x = ($\sum i \in I.$ *real-transfer-operator* (*f i*) *x*)
proof (*rule real-transfer-operator-charact*)
fix *A* **assume** [*measurable*]: $A \in$ *sets M*
have $*$: *integrable M* ($\lambda x.$ *indicator A* (*T x*) $*$ *f i x*) **for** *i*
apply (*rule Bochner-Integration.integrable-bound*[*of - f i*]) **by** (*auto simp add*: *assms indicator-def*)
have $**$: *integrable M* ($\lambda x.$ *indicator A x* $*$ *real-transfer-operator* (*f i*) *x*) **for** *i*
apply (*rule Bochner-Integration.integrable-bound*[*of - real-transfer-operator* (*f i*)]) **by** (*auto simp add*: *assms indicator-def*)
have *inti*: ($\int x.$ *indicator A* (*T x*) $*$ *f i x* ∂M) = ($\int x.$ *indicator A x* $*$ *real-transfer-operator* (*f i*) *x* ∂M) **for** *i*
by (*rule real-transfer-operator-intg*(*2*)[*symmetric*], *auto simp add*: $*$)

have ($\int x.$ *indicator A* (*T x*) $*$ ($\sum i \in I.\ f\ i\ x$)∂M) = ($\int x.$ ($\sum i \in I.$ *indicator A* (*T x*) $*$ *f i x*)∂M)
by (*simp add*: *sum-distrib-left*)
also have ... = ($\sum i \in I.$ ($\int x.$ *indicator A* (*T x*) $*$ *f i x* ∂M))
by (*rule Bochner-Integration.integral-sum*, *simp add*: $*$)
also have ... = ($\sum i \in I.$ ($\int x.$ *indicator A x* $*$ *real-transfer-operator* (*f i*) *x* ∂M))
using *inti* **by** *auto*
also have ... = ($\int x.$ ($\sum i \in I.$ *indicator A x* $*$ *real-transfer-operator* (*f i*) *x*)∂M)
by (*rule Bochner-Integration.integral-sum*[*symmetric*], *simp add*: $**$)
also have ... = ($\int x.$ *indicator A x* $*$ ($\sum i \in I.$ *real-transfer-operator* (*f i*) *x*)∂M)
by (*simp add*: *sum-distrib-left*)
finally show ($\int x.$ *indicator A x* $*$ ($\sum i \in I.$ *real-transfer-operator* (*f i*) *x*)∂M) = ($\int x.$ *indicator A* (*T x*) $*$ ($\sum i \in I.\ f\ i\ x$)∂M) **by** *auto*
qed (*auto simp add*: *assms real-transfer-operator-int*(*1*)[*OF assms*(*1*)])
end

12.3 Conservativity in terms of transfer operators

Conservativity amounts to the fact that $\sum f(T^n x) = \infty$ for almost every x such that $f(x) > 0$, if f is nonnegative (see Lemma `recurrent_series_infinite`). There is a dual formulation, in terms of transfer operators, asserting that $\sum \hat{T}^n f(x) = \infty$ for almost every x such that $f(x) > 0$. It is proved by duality, reducing to the previous statement.

theorem (**in** *conservative*) *recurrence-series-infinite-transfer-operator*:
  assumes [*measurable*]: $f \in$ *borel-measurable M*
  shows *AE x in M. f x > 0* $\longrightarrow$ ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) $= \infty$
proof −
  define *A* **where** *A* = {*x* ∈ *space M. f x > 0*}
  have [*measurable*]: *A* ∈ *sets M*
    unfolding *A-def* **by** *auto*
  have *K*: *emeasure M* {*x* ∈ *A.* ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) ≤ *K*} = *0*
if *K* < ∞ **for** *K*
  proof (*rule ccontr*)
    assume *emeasure M* {*x* ∈ *A.* ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) ≤ *K*} ≠ *0*
    then have ∗: *emeasure M* {*x* ∈ *A.* ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) ≤ *K*}
> *0*
      using *not-gr-zero* **by** *blast*
    obtain *B* **where** *B* [*measurable*]: *B* ∈ *sets M B* ⊆ {*x* ∈ *A.* ($\sum$ *n.* (*nn-transfer-operator*^^*n*)
f x) ≤ *K*} *emeasure M B* < ∞ *emeasure M B* > *0*
      using *approx-with-finite-emeasure*[*OF* - ∗] **by** *auto*
    have *f x* > *0* **if** *x* ∈ *B* **for** *x*
      using *B*(*2*) *that* **unfolding** *A-def* **by** *auto*
    moreover have *AE x*∈*B in M.* ($\sum$ *n. indicator B* ((*T*^^*n*) *x*)) = (∞::*ennreal*)
      using *recurrence-series-infinite*[*of indicator B*] **by** (*auto simp add*: *indica-tor-def*)
    ultimately have *PInf*: *AE x*∈*B in M.* ($\sum$ *n. indicator B* ((*T*^^*n*) *x*)) ∗ *f x* =
⊤
      unfolding *ennreal-mult-eq-top-iff* **by** *fastforce*

    have ($\int^+$*x. indicator B x* ∗ ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) ∂*M*) ≤ ($\int^+$*x.*
indicator B x ∗ *K* ∂*M*)
      apply (*rule nn-integral-mono*) **using** *B*(*2*) **unfolding** *indicator-def* **by** *auto*
    also have ... = *K* ∗ *emeasure M B*
      by (*simp add*: *mult.commute nn-integral-cmult-indicator*)
    also have ... < ∞ **using** ‹*K* < ∞› *B*(*3*)
      using *ennreal-mult-eq-top-iff top.not-eq-extremum* **by** *auto*
    finally have ∗: ($\int^+$*x. indicator B x* ∗ ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*)
∂*M*) < ∞ **by** *auto*

    have ($\int^+$*x. indicator B x* ∗ ($\sum$ *n.* (*nn-transfer-operator*^^*n*) *f x*) ∂*M*)
        = ($\int^+$*x.* ($\sum$ *n. indicator B x* ∗ (*nn-transfer-operator*^^*n*) *f x*) ∂*M*)
      by *auto*
    also have ... = ($\sum$ *n.* ($\int^+$*x. indicator B x* ∗ (*nn-transfer-operator*^^*n*) *f x*
∂*M*))

```
    by (rule nn-integral-suminf, auto)
  also have ... = (∑ n. (∫⁺x. indicator B ((T^^n) x) * f x ∂M))
    using nn-transfer-operator-intTn-g by auto
  also have ... = (∫⁺x. (∑ n. indicator B ((T^^n) x) * f x) ∂M)
    by (rule nn-integral-suminf[symmetric], auto)
  also have ... = (∫⁺x. (∑ n. indicator B ((T^^n) x)) * f x ∂M)
    by auto
  finally have **: (∫⁺x. (∑ n. indicator B ((T^^n) x)) * f x ∂M) ≠ ∞
    using * by simp
  have AE x in M. (∑ n. indicator B ((T^^n) x)) * f x ≠ ∞
    by (rule nn-integral-noteq-infinite[OF - **], auto)
  then have AE x∈B in M. False
    using PInf by auto
  then have emeasure M B = 0
     by (smt (z3) AE-E B(1) Collect-mem-eq Collect-mono-iff dual-order.trans
emeasure-eq-0 subsetD sets.sets-into-space)
  then show False
    using B by auto
 qed
 have L: {x ∈ A. (∑ n. (nn-transfer-operator^^n) f x) ≤ K} ∈ null-sets M if K
< ∞ for K
   using K[OF that] by auto
 have P: AE x in M. x ∈ A ⟶ (∑ n. (nn-transfer-operator^^n) f x) ≥ K if K
< ∞ for K
   using AE-not-in[OF L[OF that]] by auto
 have AE x in M. ∀ N::nat. (x ∈ A ⟶ (∑ n. (nn-transfer-operator^^n) f x) ≥
of-nat N)
   unfolding AE-all-countable by (auto simp add: of-nat-less-top intro!: P)
 then have AE x in M. f x > 0 ⟶ (∀ N::nat. (∑ n. (nn-transfer-operator^^n)
f x) ≥ of-nat N)
   unfolding A-def by auto
 then show AE x in M. 0 < f x ⟶ (∑ n. (nn-transfer-operator ^^ n) f x) = ∞
   using ennreal-ge-nat-imp-PInf by auto
qed


end
```

13 Normalizing sequences

```
theory Normalizing-Sequences
 imports Transfer-Operator Asymptotic-Density
begin
```

In this file, we prove the main result in [Gou18]: in a conservative system, if a renormalized sequence $S_n f/B_n$ converges in distribution towards a limit which is not a Dirac mass at 0, then B_n can not grow exponentially fast. We also prove the easier result that, in a probability preserving system, normalizing sequences grow at most polynomially.

13.1 Measure of the preimages of disjoint sets.

We start with a general result about conservative maps: If A_n are disjoint sets, and P is a finite mass measure which is absolutely continuous with respect to M, then $T^{-n}A_n$ is most often small: $P(T^{-n}A_n)$ tends to 0 in Cesaro average. The proof is written in terms of densities and positive transfer operators, so we first write it in ennreal.

```
theorem (in conservative) disjoint-sets-emeasure-Cesaro-tendsto-zero:
  fixes P::'a measure and A::nat ⇒ 'a set
  assumes [measurable]: ⋀n. A n ∈ sets M
      and disjoint-family A
          absolutely-continuous M P sets P = sets M
          emeasure P (space M) ≠ ∞
  shows (λn. (∑ i<n. emeasure P (space M ∩ (T^^i)−‘(A i)))/n) ⟶ 0
proof (rule order-tendstoI)
  fix delta::ennreal assume delta > 0

  have ∃ epsilon. epsilon ≠ 0 ∧ epsilon ≠ ∞ ∧ 4 ∗ epsilon < delta
    apply (cases delta)
      apply (rule exI[of - delta/5]) using ‹delta>0› apply (auto simp add: en-
nreal-divide-eq-top-iff ennreal-divide-numeral numeral-mult-ennreal intro!: ennreal-lessI)
    apply (rule exI[of - 1]) by auto
  then obtain epsilon where epsilon ≠ 0 epsilon ≠ ∞ 4 ∗ epsilon < delta
    by auto
  then have epsilon > 0 using not-gr-zero by blast

  define L::ennreal where L = (1/epsilon) ∗ (1+ emeasure P (space M))
  have L ≠ ∞
   unfolding L-def using assms(5) divide-ennreal-def ennreal-mult-eq-top-iff ‹ep-
silon ≠ 0› by auto
  have L ≠ 0
    unfolding L-def using ‹epsilon ≠ ∞› by (simp add: ennreal-divide-eq-top-iff)
  have emeasure P (space M) ≤ epsilon ∗ L unfolding L-def
    using ‹epsilon ≠ 0› ‹epsilon ≠ ∞› ‹emeasure P (space M) ≠ ∞›
    apply (cases epsilon)
     apply (metis (no-types, lifting) add.commute add.right-neutral add-left-mono
ennreal-divide-times infinity-ennreal-def mult.left-neutral mult-divide-eq-ennreal zero-le-one)
    by simp
  then have emeasure P (space M) / L ≤ epsilon
   using ‹L ≠ 0› ‹L ≠ ∞› by (metis divide-le-posI-ennreal mult.commute not-gr-zero)
 then have c ∗ (emeasure P (space M)/L) ≤ c ∗ epsilon for c by (rule mult-left-mono,
simp)
```

We introduce the density of P.

```
  define f where f = RN-deriv M P
  have [measurable]: f ∈ borel-measurable M
    unfolding f-def by auto
  have [simp]: P = density M f
```

unfolding *f-def* **apply** (*rule density-RN-deriv*[*symmetric*]) **using** *assms* **by** *auto*
have *space P = space M*
by *auto*
interpret *Pc*: *finite-measure P*
apply *standard* **unfolding** ‹*space P = space M*› **using** *assms*(*5*) **by** *auto*

have ∗: *AE x in P*. *eventually* (λn. ($\sum$ *i<n*. (*nn-transfer-operator*^^*i*) *f x*) > *L* ∗ *f x*) *sequentially*
proof −
have *AE x in M*. *f x* ≠ ∞
unfolding *f-def* **apply** (*intro RN-deriv-finite Pc.sigma-finite-measure*)
unfolding ‹*space P = space M*› **using** *assms* **by** *auto*
moreover have *AE x in M*. *f x > 0* ⟶ ($\sum$ *n*. (*nn-transfer-operator*^^*n*) *f x*) = ∞
using *recurrence-series-infinite-transfer-operator* **by** *auto*
ultimately have *AE x in M*. *f x > 0* ⟶ (($\sum$ *n*. (*nn-transfer-operator*^^*n*) *f x*) = ∞ ∧ *f x* ≠ ∞)
by *auto*
then have *AEP*: *AE x in P*. ($\sum$ *n*. (*nn-transfer-operator*^^*n*) *f x*) = ∞ ∧ *f x* ≠ ∞
unfolding ‹*P = density M f*› **using** *AE-density*[*of f M*] **by** *auto*
moreover have *eventually* (λn. ($\sum$ *i<n*. (*nn-transfer-operator*^^*i*) *f x*) > *L* ∗ *f x*) *sequentially*
if ($\sum$ *n*. (*nn-transfer-operator*^^*n*) *f x*) = ∞ ∧ *f x* ≠ ∞ **for** *x*
proof −
have (λn. ($\sum$ *i<n*. (*nn-transfer-operator*^^*i*) *f x*)) ⟶ ($\sum$ *i*. (*nn-transfer-operator*^^*i*) *f x*)
by (*simp add*: *summable-LIMSEQ*)
moreover have ($\sum$ *i*. (*nn-transfer-operator*^^*i*) *f x*) > *L* ∗ *f x*
using *that* ‹*L* ≠ ∞› **by** (*auto simp add*: *ennreal-mult-less-top top.not-eq-extremum*)
ultimately show *?thesis*
by (*rule order-tendstoD*(*1*))
qed
ultimately show *?thesis*
by *auto*
qed
have ∃ *U N*. *U* ∈ *sets P* ∧ (∀ *n* ≥ *N*. ∀ *x* ∈ *U*. ($\sum$ *i<n*. (*nn-transfer-operator*^^*i*) *f x*) > *L* ∗ *f x*) ∧ *emeasure P* (*space P* − *U*) < *epsilon*
apply (*rule Pc.Egorov-lemma*[*OF* - ∗]) **using** ‹*epsilon*≠*0*› **by** (*auto simp add*: *zero-less-iff-neq-zero*)
then obtain *U N1* **where** [*measurable*]: *U* ∈ *sets M* **and** *U*: *emeasure P* (*space M* − *U*) < *epsilon*
⋀*n x*. *n* ≥ *N1* ⟹ *x* ∈ *U* ⟹ *L* ∗ *f x* < ($\sum$ *i<n*. (*nn-transfer-operator*^^*i*) *f x*)
unfolding ‹*sets P = sets M*› ‹*space P = space M*› **by** *blast*
have *U* ⊆ *space M* **by** (*rule sets.sets-into-space*, *simp*)

define *K* **where** *K = N1 + 1*

have *K* ≥ *N1 K* ≥ *1* **unfolding** *K-def* **by** *auto*
have ∗: *K* ∗ *emeasure P* (*space M*) / *epsilon* ≠ ∞
using ‹*emeasure P* (*space M*) ≠ ∞› ‹*epsilon* ≠ *0*› *ennreal-divide-eq-top-iff ennreal-mult-eq-top-iff* **by** *auto*
obtain *N2*::*nat* **where** *N2*: *N2* ≥ *K* ∗ *emeasure P* (*space M*) / *epsilon*
using *ennreal-archimedean*[*OF* ∗] **by** *auto*
define *N* **where** *N* = *2* ∗ *K* + *N2*
have ($\sum$ *k*∈{..<*n*}. *emeasure P* (*space M* ∩ (*T*^^*k*)−‘(*A k*))) / *n* < *delta* **if** *n* ≥ *N* **for** *n*
proof −
have *n* ≥ *2* ∗ *K of-nat n* ≥ ((*of-nat N2*)::*ennreal*) **using** *that* **unfolding** *N-def* **by** *auto*
then have *n* ≥ *K* ∗ *emeasure P* (*space M*) / *epsilon*
using *N2 order-trans* **by** *blast*
then have *K* ∗ *emeasure P* (*space M*) ≤ *n* ∗ *epsilon*
using ‹*epsilon* > *0*› ‹*epsilon* ≠ ∞›
by (*smt* (*z3*) *divide-ennreal-def divide-right-mono-ennreal ennreal-mult-divide-eq ennreal-mult-eq-top-iff infinity-ennreal-def mult.commute not-le order-le-less*)
have *n* ≥ *1* **using** ‹*n* ≥ *2* ∗ *K*› ‹*K* ≥ *1*› **by** *auto*

have ∗: (($\sum$ *k*∈{*K*..<*n*−*K*}. *indicator* (*A k*) ((*T*^^*k*) *x*))::*ennreal*) ≤ ($\sum$ *i*∈{*K*..<*n*}. *indicator* (*A* (*i*−*j*)) ((*T*^^(*i*−*j*)) *x*))
if *j* < *K* **for** *j x*
proof −
have ($\sum$ *k* ∈ {*K*..<*n*−*K*}. *indicator* (*A k*) ((*T*^^*k*) *x*)) ≤ (($\sum$ *k*∈{*K*−*j*..<*n*−*j*}. *indicator* (*A k*) ((*T*^^*k*) *x*))::*ennreal*)
apply (*rule sum-mono2*) **using** ‹*j* < *K*› **by** *auto*
also have ... = ($\sum$ *i*∈{*K*..<*n*}. *indicator* (*A* (*i*−*j*)) ((*T*^^(*i*−*j*)) *x*))
apply (*rule sum.reindex-bij-betw*[*symmetric*], *rule bij-betw-byWitness*[*of* - *λx. x*+*j*]) **using** ‹*j* < *K*› **by** *auto*
finally show *?thesis* **by** *simp*
qed

have *L* ∗ ($\sum$ *k* ∈ {*K*..<*n*−*K*}. *emeasure P* (*U* ∩ (*T*^^*k*)−‘(*A k*))) = *L* ∗ ($\sum$ *k* ∈ {*K*..<*n*−*K*}.($\int^+$*x*. *indicator* (*U* ∩ (*T*^^*k*)−‘(*A k*)) *x* ∂*P*))
by *auto*
also have ... = ($\sum$ *k* ∈ {*K*..<*n*−*K*}. ($\int^+$*x*. *L* ∗ *indicator* (*U* ∩ (*T*^^*k*)−‘(*A k*)) *x* ∂*P*))
unfolding *sum-distrib-left* **by** (*intro sum.cong nn-integral-cmult*[*symmetric*], *auto*)
also have ... = ($\sum$ *k* ∈ {*K*..<*n*−*K*}. ($\int^+$*x*. *f x* ∗ (*L* ∗ *indicator* (*U* ∩ (*T*^^*k*)−‘(*A k*)) *x*) ∂*M*))
unfolding ‹*P* = *density M f*› **by** (*intro sum.cong nn-integral-density*, *auto*)
also have ... = ($\sum$ *k* ∈ {*K*..<*n*−*K*}. ($\int^+$*x*. *f x* ∗ *L* ∗ *indicator U x* ∗ *indicator* (*A k*) ((*T*^^*k*) *x*) ∂*M*))
by (*intro sum.cong nn-integral-cong*, *auto simp add*: *algebra-simps indicator-def*)
also have ... ≤ ($\sum$ *k* ∈ {*K*..<*n*−*K*}. ($\int^+$*x*. ($\sum$ *j* ∈ {..<*K*}. (*nn-transfer-operator*^^*j*) *f x*) ∗ *indicator* (*A k*) ((*T*^^*k*) *x*) ∂*M*))

apply (*intro sum-mono nn-integral-mono*)
using $U(2)$[*OF* ‹$K \geq N1$›] **unfolding** *indicator-def* **using** *less-imp-le* **by** (*auto simp add*: *algebra-simps*)
also have ... = ($\int^+ x.$ ($\sum k \in \{K..<n-K\}$. ($\sum j \in \{..<K\}$. (*nn-transfer-operator* ^^ *j*) *f x* ∗ *indicator* (*A k*) ((*T* ^^ *k*) *x*))) ∂*M*)
apply (*subst nn-integral-sum*, *simp*) **unfolding** *sum-distrib-right* **by** *auto*
also have ... = ($\int^+ x.$ ($\sum j \in \{..<K\}$. ($\sum k \in \{K..<n-K\}$. (*nn-transfer-operator* ^^ *j*) *f x* ∗ *indicator* (*A k*) ((*T* ^^ *k*) *x*))) ∂*M*)
by (*rule nn-integral-cong*, *rule sum.swap*)
also have ... = ($\sum j \in \{..<K\}$. ($\int^+ x.$ (*nn-transfer-operator* ^^ *j*) *f x* ∗ ($\sum k \in \{K..<n-K\}$. *indicator* (*A k*) ((*T* ^^ *k*) *x*)) ∂*M*))
apply (*subst nn-integral-sum*, *simp*) **unfolding** *sum-distrib-left* **by** *auto*
also have ... ≤ ($\sum j \in \{..<K\}$. ($\int^+ x.$ (*nn-transfer-operator* ^^ *j*) *f x* ∗ ($\sum i \in \{K..<n\}$. *indicator* (*A* (*i*−*j*)) ((*T* ^^ (*i*−*j*)) *x*)) ∂*M*))
apply (*rule sum-mono*, *rule nn-integral-mono*) **using** ∗ **by** (*auto simp add*: *mult-left-mono*)
also have ... = ($\sum i \in \{K..<n\}$. ($\sum j \in \{..<K\}$. ($\int^+ x.$ (*nn-transfer-operator* ^^ *j*) *f x* ∗ *indicator* (*A* (*i*−*j*)) ((*T* ^^ (*i*−*j*)) *x*) ∂*M*)))
unfolding *sum-distrib-left* **using** *sum.swap* **by** (*subst nn-integral-sum*, *auto*)
also have ... = ($\sum i \in \{K..<n\}$. ($\sum j \in \{..<K\}$. ($\int^+ x.$ *f x* ∗ *indicator* (*A* (*i*−*j*)) ((*T* ^^ (*i*−*j*)) ((*T* ^^ *j*) *x*)) ∂*M*)))
by (*subst nn-transfer-operator-intg-Tn*, *auto*)
also have ... = ($\sum i \in \{K..<n\}$. ($\int^+ x.$ *f x* ∗ ($\sum j \in \{..<K\}$. *indicator* (*A* (*i*−*j*)) ((*T* ^^ (*i*−*j*)) ((*T* ^^ *j*) *x*))) ∂*M*))
unfolding *sum-distrib-left* **by** (*subst nn-integral-sum*, *auto*)
also have ... = ($\sum i \in \{K..<n\}$. ($\int^+ x.$ ($\sum j \in \{..<K\}$. *indicator* (*A* (*i*−*j*)) ((*T* ^^ ((*i*−*j*)+*j*)) *x*)) ∂*P*))
unfolding ‹*P* = *density M f*› *funpow-add comp-def* **apply** (*rule sum.cong*, *simp*) **by** (*rule nn-integral-density*[*symmetric*], *auto*)
also have ... = ($\sum i \in \{K..<n\}$. ($\int^+ x.$ ($\sum j \in \{..<K\}$. *indicator* (*A* (*i*−*j*)) ((*T* ^^ *i*) *x*)) ∂*P*))
by *auto*
also have ... ≤ ($\sum i \in \{K..<n\}$. ($\int^+ x.$ (*1*::*ennreal*) ∂*P*))
apply (*rule sum-mono*) **apply** (*rule nn-integral-mono*) **apply** (*rule disjoint-family-indicator-le-1*)
using *assms*(*2*) **apply** (*auto simp add*: *disjoint-family-on-def*)
by (*metis Int-iff diff-diff-cancel equals0D le-less le-trans*)
also have ... ≤ *n* ∗ *emeasure P* (*space M*)
using *assms*(*4*) **by** (*auto intro*!: *mult-right-mono*)
finally have ∗: *L* ∗ ($\sum k \in \{K..<n-K\}$. *emeasure P* (*U* ∩ (*T* ^^ *k*)−‘(*A k*))) ≤ *n* ∗ *emeasure P* (*space M*)
by *simp*
have *Ineq*: ($\sum k \in \{K..<n-K\}$. *emeasure P* (*U* ∩ (*T* ^^ *k*)−‘(*A k*))) ≤ *n* ∗ *emeasure P* (*space M*) / *L*
using *divide-right-mono-ennreal*[*OF* ∗, *of L*] ‹*L* ≠ *0*›
by (*metis* (*no-types*, *lifting*) ‹*L* ≠ ∞› *ennreal-mult-divide-eq infinity-ennreal-def mult.commute*)

have *I*: {..<*K*} ∪ {*K*..<*n*−*K*} ∪ {*n*−*K*..<*n*} = {..<*n*} **using** ‹*n* ≥ *2* ∗ *K*›

by *auto*
have ($\sum k\in\{..<n\}$. *emeasure P* (*space M* ∩ (T^^k)− '(A k))) ≤ ($\sum k\in\{..<n\}$. *emeasure P* (*U* ∩ (T^^k)− '(A k)) + *epsilon*)
proof (*rule sum-mono*)
fix *k*
have *emeasure P* (*space M* ∩ (T^^k)− '(A k)) ≤ *emeasure P* ((*U* ∩ (T^^k)− '(A k)) ∪ (*space M* − *U*))
by (*rule emeasure-mono*, *auto*)
also have ... ≤ *emeasure P* (*U* ∩ (T^^k)− '(A k)) + *emeasure P* (*space M* − *U*)
by (*rule emeasure-subadditive*, *auto*)
also have ... ≤ *emeasure P* (*U* ∩ (T^^k)− '(A k)) + *epsilon*
using *U*(*1*) **by** *auto*
finally show *emeasure P* (*space M* ∩ (*T* ^^ *k*) −' *A k*) ≤ *emeasure P* (*U* ∩ (*T* ^^ *k*) −' *A k*) + *epsilon*
by *simp*
qed
also have ... = ($\sum k\in\{..<K\} \cup \{K..<n-K\} \cup \{n-K..<n\}$. *emeasure P* (*U* ∩ ($T$^^$k$)− '($A$ k))) + ($\sum k\in\{..<n\}$. *epsilon*)
unfolding *sum.distrib I* **by** *simp*
also have ... = ($\sum k\in\{..<K\}$. *emeasure P* (*U* ∩ (T^^k)− '(A k))) + ($\sum k\in\{K..<n-K\}$. *emeasure P* (*U* ∩ (T^^k)− '(A k)))
+ ($\sum k\in\{n-K..<n\}$. *emeasure P* (*U* ∩ (T^^k)− '(A k))) + *n* ∗ *epsilon*
apply (*subst sum.union-disjoint*) **apply** *simp* **apply** *simp* **using** ‹*n* ≥ *2* ∗ *K*›
apply (*simp add*: *ivl-disj-int-one*(*2*) *ivl-disj-un-one*(*2*))
by (*subst sum.union-disjoint*, *auto*)
also have ... ≤ ($\sum k\in\{..<K\}$. *emeasure P* (*space M*)) + *n* ∗ *emeasure P* (*space M*) / *L* + ($\sum k\in\{n-K..<n\}$. *emeasure P* (*space M*)) + *n* ∗ *epsilon*
apply (*intro add-mono sum-mono Ineq emeasure-mono*) **using** ‹*U* ⊆ *space M*› **by** *auto*
also have ... = *K* ∗ *emeasure P* (*space M*) + *n* ∗ *emeasure P* (*space M*)/*L* + *K* ∗ *emeasure P* (*space M*) + *n* ∗ *epsilon*
using ‹*n* ≥ *2* ∗ *K*› **by** *auto*
also have ... ≤ *n* ∗ *epsilon* + *n* ∗ *epsilon* + *n* ∗ *epsilon* + *n* ∗ *epsilon*
apply (*intro add-mono*)
using ‹*K* ∗ *emeasure P* (*space M*) ≤ *n* ∗ *epsilon*› ‹*of-nat n* ∗ (*emeasure P* (*space M*)/*L*) ≤ *of-nat n* ∗ *epsilon*›
ennreal-times-divide **by** *auto*
also have ... = *n* ∗ (*4* ∗ *epsilon*)
by (*metis* (*no-types*, *lifting*) *add.assoc distrib-right mult.left-commute mult-2 numeral-Bit0*)
also have ... < *n* ∗ *delta*
using ‹*4* ∗ *epsilon* < *delta*› ‹*n* ≥ *1*›
by (*simp add*: *ennreal-mult-strict-left-mono ennreal-of-nat-eq-real-of-nat*)
finally show *?thesis*
apply (*subst divide-less-ennreal*)
using ‹*n* ≥ *1*› *of-nat-less-top* **by** (*auto simp add*: *mult.commute*)

qed
then show *eventually* (λ*n*. (∑ *k*∈{..<*n*}. *emeasure P* (*space M* ∩ (*T*^^*k*)− ‘(*A k*))) / *n* < *delta*) *sequentially*
unfolding *eventually-sequentially* **by** *auto*
qed (*simp*)

We state the previous theorem using measures instead of emeasures. This is clearly equivalent, but one has to play with ennreal carefully to prove it.

theorem (**in** *conservative*) *disjoint-sets-measure-Cesaro-tendsto-zero*:
fixes *P*::′*a measure* **and** *A*::*nat* ⇒ ′*a set*
assumes [*measurable*]: ⋀*n*. *A n* ∈ *sets M*
and *disjoint-family A*
absolutely-continuous M P sets P = *sets M*
emeasure P (*space M*) ≠ ∞
shows (λ*n*. (∑ *i*<*n*. *measure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n*) ⟶ *0*
proof −
have *space P* = *space M*
using *assms*(*4*) *sets-eq-imp-space-eq* **by** *blast*
moreover have *emeasure P Q* ≤ *emeasure P* (*space P*) **for** *Q*
by (*simp add*: *emeasure-space*)
ultimately have [*simp*]: *emeasure P Q* ≠ ⊤ **for** *Q*
using ‹*emeasure P* (*space M*) ≠ ∞› *neq-top-trans* **by** *auto*
have ∗: *ennreal* ((∑ *i*<*n*. *measure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n*) = (∑ *i*<*n*. *emeasure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n* **if** *n* > *0* **for** *n*
apply (*subst divide-ennreal*[*symmetric*])
apply (*auto intro*!: *sum-nonneg that simp add*: *ennreal-of-nat-eq-real-of-nat*[*symmetric*])
apply(*subst sum-ennreal*[*symmetric*], *simp*)
apply (*subst emeasure-eq-ennreal-measure*) **by** *auto*
have *eventually* (λ*n*. *ennreal* ((∑ *i*<*n*. *measure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n*) = (∑ *i*<*n*. *emeasure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n*) *sequentially*
unfolding *eventually-sequentially* **apply** (*rule exI*[*of - 1*]) **using** ∗ **by** *auto*
then have ∗: (λ*n*. *ennreal* ((∑ *i*<*n*. *measure P* (*space M* ∩ (*T*^^*i*)− ‘(*A i*)))/*n*)) ⟶ *ennreal 0*
using *disjoint-sets-emeasure-Cesaro-tendsto-zero*[*OF assms*] *tendsto-cong* **by** *force*
show *?thesis*
apply (*subst tendsto-ennreal-iff*[*symmetric*]) **using** ∗ **apply** *auto*
unfolding *eventually-sequentially* **apply** (*rule exI*[*of - 1*])
by (*auto simp add*: *divide-simps intro*!: *sum-nonneg*)
qed

As convergence to 0 in Cesaro mean is equivalent to convergence to 0 along a density one sequence, we obtain the equivalent formulation of the previous theorem.

theorem (**in** *conservative*) *disjoint-sets-measure-density-one-tendsto-zero*:
fixes *P*::′*a measure* **and** *A*::*nat* ⇒ ′*a set*
assumes [*measurable*]: ⋀*n*. *A n* ∈ *sets M*
and *disjoint-family A*
absolutely-continuous M P sets P = *sets M*

emeasure P (space M) $\neq \infty$
shows $\exists B$. *lower-asymptotic-density B* = *1* $\wedge$ (λn. *measure P* (*space M* $\cap$ (*T*^^*n*)−‘(*A n*)) ∗ *indicator B n*) ⟶ *0*
by (*rule cesaro-imp-density-one*[*OF* - *disjoint-sets-measure-Cesaro-tendsto-zero*[*OF assms*]], *simp*)

13.2 Normalizing sequences do not grow exponentially in conservative systems

We prove the main result in [Gou18]: in a conservative system, if a renormalized sequence $S_n f/B_n$ converges in distribution towards a limit which is not a Dirac mass at 0, then B_n can not grow exponentially fast. The proof is expressed in the following locale. The main theorem is Theorem `subexponential_growth` below. To prove it, we need several preliminary estimates.

We will use the fact that a real random variables which is not the Dirac mass at 0 gives positive mass to a set separated away from 0.

lemma (**in** *real-distribution*) *not-Dirac-0-imp-positive-mass-away-0*:
assumes *prob* {*0*} < *1*
shows ∃ *a*. *a* > *0* ∧ *prob* {*x*. *abs*(*x*) > *a*} > *0*
proof −
have *1* = *prob UNIV*
using *prob-space* **by** *auto*
also have ... = *prob* {*0*} + *prob* (*UNIV* −{*0*})
by (*subst finite-measure-Union*[*symmetric*], *auto*)
finally have *0* < *prob* (*UNIV* −{*0*})
using *assms* **by** *auto*
also have ... ≤ *prob* (⋃ *n*::*nat*. {*x*. *abs*(*x*)>(*1*/*2*)^*n*})
apply (*rule finite-measure-mono*)
by (*auto*, *meson one-less-numeral-iff reals-power-lt-ex semiring-norm*(*76*) *zero-less-abs-iff*)
finally have *prob* (⋃ *n*::*nat*. {*x*. *abs*(*x*)>(*1*/*2*)^*n*}) ≠ *0*
by *simp*
then have ∃ *n*. *prob* {*x*. *abs*(*x*)>(*1*/*2*)^*n*} ≠ *0*
using *measure-countably-zero*[*of* λ*n*. {*x*. *abs*(*x*)>(*1*/*2*)^*n*}] **by** *force*
then obtain *N* **where** *N*: *prob* {*x*. *abs*(*x*) > (*1*/*2*)^*N*} ≠ *0*
by *blast*
show *?thesis*
apply (*rule exI*[*of* - (*1*/*2*)^*N*]) **using** *N* **by** (*auto simp add*: *zero-less-measure-iff*)
qed

locale *conservative-limit* =
conservative M + *PS*: *prob-space P* + *PZ*: *real-distribution Z*
for *M*::'*a measure* **and** *P*::'*a measure* **and** *Z*::*real measure* +
fixes *f g*::'*a* ⇒ *real* **and** *B*::*nat* ⇒ *real*
assumes *PabsM*: *absolutely-continuous M P*
and *Bpos*: ⋀*n*. *B n* > *0*
and *M* [*measurable*]: *f* ∈ *borel-measurable M g* ∈ *borel-measurable M sets P*

= *sets M*
  and *non-trivial*: *PZ.prob* {*0*} < *1*
  and *conv*: *weak-conv-m* (*λn. distr P borel* (*λx.* (*g x* + *birkhoff-sum f n x*) / *B n*)) *Z*
begin

For measurability statements, we want every question about Z or P to reduce to a question about Borel sets of M. We add in the next lemma all the statements that are needed so that this happens automatically.

lemma *PSZ* [*simp*, *measurable-cong*]:
space P = *space M*
h ∈ *borel-measurable P* ⟷ *h* ∈ *borel-measurable M*
A ∈ *sets P* ⟷ *A* ∈ *sets M*
using *M sets-eq-imp-space-eq real-distribution-def* **by** *auto*

The first nontrivial upper bound is the following lemma, asserting that B_{n+1} can not be much larger than $\max B_i$ for $i \leq n$. This is proved by saying that $S_{n+1}f = f + (S_n f) \circ T$, and we know that $S_n f$ is not too large on a set of very large measure, so the same goes for $(S_n f) \circ T$ by a non-singularity argument. Excepted that the measure P does not have to be nonsingular for the map T, so one has to tweak a little bit this idea, using transfer operators and conservativity. This is easier to do when the density of P is bounded by 1, so we first give the proof under this assumption, and then we reduce to this case by replacing M with $M + P$ in the second lemma below.

First, let us prove the lemma assuming that the density h of P is bounded by 1.

lemma *upper-bound-C-aux*:
assumes *P* = *density M h* ⋀*x. h x* ≤ *1*
  and [*measurable*]: *h* ∈ *borel-measurable M*
shows ∃ *C*≥*1*. ∀ *n. B* (*Suc n*) ≤ *C* ∗ *Max* {*B i*|*i. i* ≤ *n*}
proof −
obtain *a0* **where** *a0*: *a0* > *0 PZ.prob* {*x. abs*(*x*) > *a0*} > *0*
  using *PZ.not-Dirac-0-imp-positive-mass-away-0*[*OF non-trivial*] **by** *blast*
define *a* **where** *a* = *a0*/*2*
have *a* > *0* **using** ‹*a0* > *0*› **unfolding** *a-def* **by** *auto*
define *alpha* **where** *alpha* = *PZ.prob* {*x. abs* (*x*) > *a0*}/*4*
have *alpha* > *0* **unfolding** *alpha-def* **using** *a0* **by** *auto*
have *PZ.prob* {*x. abs* (*x*) > *2* ∗ *a*} > *3* ∗ *alpha*
  using *a0* **unfolding** *a-def alpha-def* **by** *auto*

First step: choose K such that, with probability $1-\alpha$, one has $\sum_{1\leq k<K} h(T^k x) \geq 1$. This follows directly from conservativity.

have ∃ *K. K* ≥ *1* ∧ *PS.prob* {*x* ∈ *space M*. (∑ *i*∈{*1..*<*K*}. *h* ((*T*^^*i*) *x*)) ≥ *1*} ≥ *1* − *alpha*
proof −
  have ∗: *AE x in P. eventually* (*λn.* (∑ *i*<*n. h* ((*T*^^*i*) *x*)) > *2*) *sequentially*

proof –
have $AE\ x\ in\ M.\ h\ x > 0 \longrightarrow (\sum i.\ h\ ((T \widehat{\ \ } i)\ x)) = \infty$
using *recurrence-series-infinite* **by** *auto*
then have *AEP*: $AE\ x\ in\ P.\ (\sum i.\ h\ ((T \widehat{\ \ } i)\ x)) = \infty$
unfolding ‹$P = density\ M\ h$› **using** *AE-density*[*of h M*] **by** *auto*
moreover have *eventually* $(\lambda n.\ (\sum i<n.\ h\ ((T \widehat{\ \ } i)\ x)) > 2)$ *sequentially*
if $(\sum i.\ h\ ((T \widehat{\ \ } i)\ x)) = \infty$ **for** x
proof –
have $(\lambda n.\ (\sum i<n.\ h\ ((T \widehat{\ \ } i)\ x))) \longlongrightarrow (\sum i.\ h\ ((T \widehat{\ \ } i)\ x))$
by (*simp add*: *summable-LIMSEQ*)
moreover have $(\sum i.\ h\ ((T \widehat{\ \ } i)\ x)) > 2$
using *that* **by** *auto*
ultimately show *?thesis*
by (*rule order-tendstoD*(*1*))
qed
ultimately show *?thesis*
by *auto*
qed
have $\exists\ U\ N.\ U \in sets\ P \wedge (\forall n \geq N.\ \forall x \in U.\ (\sum i<n.\ h\ ((T \widehat{\ \ } i)\ x)) > 2) \wedge$ $emeasure\ P\ (space\ P - U) < alpha$
apply (*rule PS.Egorov-lemma*)
apply *measurable* **using** *M*(*3*) *measurable-ident-sets* **apply** *blast*
using $*$ ‹$alpha > 0$› **by** *auto*
then obtain *U N1* **where** [*measurable*]: $U \in sets\ M$ **and** *U*: $emeasure\ P\ (space\ M - U) < alpha$
$\bigwedge n\ x.\ n \geq N1 \Longrightarrow x \in U \Longrightarrow 2 < (\sum i<n.\ h\ ((T \widehat{\ \ } i)\ x))$
by *auto*
have $U \subseteq space\ M$ **by** (*rule sets.sets-into-space*, *simp*)
define K **where** $K = N1+1$
then have $K \geq 1$ **by** *auto*
have *Ux*: $(\sum i\in\{1..<K\}.\ h\ ((T \widehat{\ \ } i)\ x)) \geq 1$ **if** $x \in U$ **for** x
proof –
have $*$: $1 < t$ **if** $2 < 1 + t$ **for** *t*::*ennreal*
apply (*cases t*) **using** *that* **apply** *auto*
by (*metis ennreal-add-left-cancel-less ennreal-less-iff ennreal-numeral le-numeral-extra*(*1*) *numeral-One one-add-one*)

have $2 < (\sum i \in \{..<K\}.\ h\ ((T \widehat{\ \ } i)\ x))$
apply (*rule U*(*2*)) **unfolding** *K-def* **using** *that* **by** *auto*
also have $... = (\sum i \in \{0\}.\ h\ ((T \widehat{\ \ } i)\ x)) + (\sum i \in \{1..<K\}.\ h\ ((T \widehat{\ \ } i)\ x))$
apply (*subst sum.union-disjoint*[*symmetric*]) **apply** *simp* **apply** *simp* **apply** *simp*
apply (*rule sum.cong*) **using** ‹$K \geq 1$› **by** *auto*
also have $... = h\ x + (\sum i \in \{1..<K\}.\ h\ ((T \widehat{\ \ } i)\ x))$
by *auto*
also have $... \leq 1 + (\sum i \in \{1..<K\}.\ h\ ((T \widehat{\ \ } i)\ x))$
using *assms* **by** *auto*
finally show *?thesis* **using** *less-imp-le*[*OF* $*$] **by** *auto*
qed

have *PS.prob* {x ∈ *space M*. (∑ i∈{*1*..<K}. h ((T^^i) x)) ≥ *1*} ≥ *1* − *alpha*
proof −
have *PS.prob* (*space P* − U) < *alpha*
using U(*1*) **by** (*simp add*: *PS.emeasure-eq-measure ennreal-less-iff*)
then have *1* − *alpha* < *PS.prob* U
using *PS.prob-compl* **by** *auto*
also have ... ≤ *PS.prob* {x ∈ *space M*. (∑ i∈{*1*..<K}. h ((T^^i) x)) ≥ *1*}
apply (*rule PS.finite-measure-mono*) **using** *Ux sets.sets-into-space*[*OF* ‹U ∈ *sets M*›] **by** *auto*
finally show *?thesis* **by** *simp*
qed
then show *?thesis* **using** ‹K ≥ *1*› **by** *auto*
qed
then obtain K **where** K: K ≥ *1* *PS.prob* {x ∈ *space M*. (∑ i∈{*1*..<K}. h ((T^^i) x)) ≥ *1*} ≥ *1* − *alpha*
by *blast*

Second step: obtain D which controls the tails of the K first Birkhoff sums of f.

have ∃ D. *PS.prob* {x ∈ *space M*. ∀ k < K. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≤ D} ≥ *1* − *alpha*
proof −
have D: ∃ D. *PS.prob* {x ∈ *space P*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D} < *alpha*/K ∧ D ≥ *1* **for** k
apply (*rule PS.random-variable-small-tails*) **using** ‹K ≥ *1*› ‹*alpha* > *0*› **by** *auto*
have ∃ D. ∀ k. *PS.prob* {x ∈ *space P*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D k} < *alpha*/K ∧ D k ≥ *1*
apply (*rule choice*) **using** D **by** *auto*
then obtain D **where** D: ⋀k. *PS.prob* {x ∈ *space P*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D k} < *alpha*/K
by *blast*
define D0 **where** D0 = *Max* (D‘{..K})
have *PS.prob* {x ∈ *space M*. ∀ k < K. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≤ D0} ≥ *1* − *alpha*
proof −
have D1: *PS.prob* {x ∈ *space M*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D0} < *alpha*/K **if** k ≤ K **for** k
proof −
have D k ≤ D0
unfolding *D0-def* **apply** (*rule Max-ge*) **using** *that* **by** *auto*
have *PS.prob* {x ∈ *space M*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D0}
≤ *PS.prob* {x ∈ *space P*. *abs*(g x + *birkhoff-sum* f k x − g((T^^k) x)) ≥ D k}
apply (*rule PS.finite-measure-mono*) **using** ‹D k ≤ D0› **by** *auto*
then show *?thesis* **using** D[*of* k] **by** *auto*
qed
have *PS.prob* (⋃ k∈ {..<K}. {x ∈ *space M*. *abs*(g x + *birkhoff-sum* f k x −

$g((T\widehat{\ }\widehat{\ }k)\ x)) \geq D0\}) \leq$
$(\sum k \in \{..<K\}.\ PS.prob\ \{x \in space\ M.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \geq D0\})$
by $(rule\ PS.finite\text{-}measure\text{-}subadditive\text{-}finite,\ auto)$
also have ... $\leq (\sum k \in \{..<K\}.\ alpha/K)$
apply $(rule\ sum\text{-}mono)$ **using** $less\text{-}imp\text{-}le[OF\ D1]$ **by** $auto$
also have ... $= alpha$
using $\langle K \geq 1\rangle$ **by** $auto$
finally have $PS.prob\ (\bigcup k\in \{..<K\}.\ \{x \in space\ M.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \geq D0\}) \leq alpha$
by $simp$
then have $1 - alpha \leq 1 - PS.prob\ (\bigcup k\in \{..<K\}.\ \{x \in space\ M.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \geq D0\})$
by $simp$
also have ... $= PS.prob\ (space\ P - (\bigcup k\in \{..<K\}.\ \{x \in space\ M.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \geq D0\}))$
by $(rule\ PS.prob\text{-}compl[symmetric],\ auto)$
also have ... $\leq PS.prob\ \{x \in space\ M.\ \forall k < K.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \leq D0\}$
by $(rule\ PS.finite\text{-}measure\text{-}mono,\ auto)$
finally show $?thesis$ **by** $simp$
qed
then show $?thesis$ **by** $blast$
qed
then obtain D **where** D: $PS.prob\ \{x \in space\ M.\ \forall k < K.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \leq D\} \geq 1 - alpha$
by $blast$

Third step: obtain ϵ small enough so that, for any set U with probability less than ϵ and for any $k \leq K$, one has $\int_U \hat{T}^k h < \delta$, where δ is very small.

define $delta$ **where** $delta = alpha/(2 * K)$
then have $delta > 0$ **using** $\langle alpha > 0\rangle\ \langle K \geq 1\rangle$ **by** $auto$
have $\exists\, epsilon > (0::real).\ \forall U \in sets\ P.\ \forall k \leq K.\ emeasure\ P\ U < epsilon \longrightarrow (\int^{+} x\in U.\ ((nn\text{-}transfer\text{-}operator\widehat{\ }\widehat{\ }k)\ h)\ x\ \partial P) \leq delta$
proof $-$
have $*$: $\exists\, epsilon>(0::real).\ \forall U\in sets\ P.\ emeasure\ P\ U < epsilon \longrightarrow (\int^{+} x\in U.\ ((nn\text{-}transfer\text{-}operator\widehat{\ }\widehat{\ }k)\ h)\ x\ \partial P) < delta$
for k
proof $(rule\ small\text{-}nn\text{-}integral\text{-}on\text{-}small\text{-}sets[OF\ \text{-}\ \langle 0 < delta\rangle])$
have $(\int^{+} x.\ ((nn\text{-}transfer\text{-}operator\widehat{\ }\widehat{\ }k)\ h)\ x\ \partial P) = (\int^{+} x.\ h\ x * ((nn\text{-}transfer\text{-}operator\widehat{\ }\widehat{\ }k)\ h)\ x\ \partial M)$
unfolding $\langle P = density\ M\ h\rangle$ **by** $(rule\ nn\text{-}integral\text{-}density,\ auto)$
also have ... $\leq (\int^{+} x.\ 1 * ((nn\text{-}transfer\text{-}operator\widehat{\ }\widehat{\ }k)\ h)\ x\ \partial M)$
apply $(intro\ nn\text{-}integral\text{-}mono\ mult\text{-}right\text{-}mono)$ **using** $assms(2)$ **by** $auto$
also have ... $= (\int^{+} x.\ 1 * h\ x\ \partial M)$
by $(rule\ nn\text{-}transfer\text{-}operator\text{-}intTn\text{-}g,\ auto)$
also have ... $= emeasure\ P\ (space\ M)$
using $PS.emeasure\text{-}space\text{-}1$ **by** $(simp\ add\text{: } emeasure\text{-}density\ \langle P = density\ M\ h\rangle)$

also have ... $<$ ∞
using *PS.emeasure-space-1* **by** *simp*
finally show ($\int^+ x$. ((*nn-transfer-operator*^^*k*) *h*) *x* ∂P) $\neq$ ∞
by *auto*
qed (*simp*)
have $\exists$ *epsilon*. $\forall$ *k*. *epsilon k* $>$ (*0*::*real*) $\wedge$ ($\forall$ *U*$\in$*sets P*. *emeasure P U* $<$ *epsilon k* $\longrightarrow$ ($\int^+ x \in U$. ((*nn-transfer-operator*^^*k*) *h*) *x* ∂P) $<$ *delta*)
apply (*rule choice*) **using** $*$ **by** *blast*
then obtain *epsilon*::*nat* $\Rightarrow$ *real* **where** *E*: $\bigwedge$*k*. *epsilon k* $>$ *0*
$\bigwedge$*k U*. *U* $\in$ *sets P* $\Longrightarrow$ *emeasure P U* $<$ *epsilon k* $\Longrightarrow$ ($\int^+ x \in U$. ((*nn-transfer-operator*^^*k*) *h*) *x* ∂P) $<$ *delta*
by *blast*
define *epsilon0* **where** *epsilon0* = *Min* (*epsilon*`{..*K*})
have *epsilon0* $\in$ *epsilon*`{..*K*} **unfolding** *epsilon0-def* **by** (*rule Min-in*, *auto*)
then have *epsilon0* $>$ *0* **using** *E*(*1*) **by** *auto*
have *small-setint*: ($\int^+ x \in U$. ((*nn-transfer-operator*^^*k*) *h*) *x* ∂P) $\leq$ *delta*
if *k* $\leq$ *K U* $\in$ *sets P emeasure P U* $<$ *epsilon0* **for** *k U*
proof −
have $*$: *epsilon0* $\leq$ *epsilon k*
unfolding *epsilon0-def* **apply** (*rule Min-le*) **using** ‹*k* $\leq$ *K*› **by** *auto*
show *?thesis*
apply (*rule less-imp-le*[*OF E*(*2*)[*OF* ‹*U* $\in$ *sets P*›]])
using *ennreal-leI*[*OF* $*$] ‹*emeasure P U* $<$ *epsilon0*› **by** *auto*
qed
then show *?thesis* **using** ‹*epsilon0* $>$ *0*› **by** *auto*
qed
then obtain *epsilon*::*real* **where** *epsilon* $>$ *0* **and**
small-setint: $\bigwedge$*k U*. *k* $\leq$ *K* $\Longrightarrow$ *U* $\in$ *sets P* $\Longrightarrow$ *emeasure P U* $<$ *epsilon* $\Longrightarrow$ ($\int^+ x \in U$. ((*nn-transfer-operator*^^*k*) *h*) *x* ∂P) $\leq$ *delta*
by *blast*

Fourth step: obtain an index after which the convergence in distribution ensures that the probability to be larger than $2a$ and to be very large is comparable for $(g + S_n f)/B_n$ and for Z.

obtain *C0* **where** *PZ.prob* {*x*. *abs*(*x*) $\geq$ *C0*} $<$ *epsilon C0* $\geq$ *1*
using *PZ.random-variable-small-tails*[*OF* ‹*epsilon* $>$ *0*›, *of* λx. *x*] **by** *auto*
have *A*: *eventually* (λn. *measure* (*distr P borel* (λx. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $>$ *2* $*$ *a*} $>$ *3* $*$ *alpha*) *sequentially*
apply (*rule open-set-weak-conv-lsc*[*of* - *Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv* ‹*PZ.prob* {*x*. *abs* (*x*) $>$ *2* $*$ *a*} $>$ *3* $*$ *alpha*›)
have *B*: *eventually* (λn. *measure* (*distr P borel* (λx. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $\geq$ *C0*} $<$ *epsilon*) *sequentially*
apply (*rule closed-set-weak-conv-usc*[*of* - *Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv* ‹*PZ.prob* {*x*. *abs*(*x*) $\geq$ *C0*} $<$ *epsilon*›)
obtain *N* **where** *N*: $\bigwedge$*n*. *n* $\geq$ *N* $\Longrightarrow$ *measure* (*distr P borel* (λx. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $>$ *2* $*$ *a*} $>$ *3* $*$ *alpha*
$\bigwedge$*n*. *n* $\geq$ *N* $\Longrightarrow$ *measure* (*distr P borel* (λx. (*g x* + *birkhoff-sum f*

n x) / *B n*)) {*x*. *abs* (*x*) ≥ *C0*} < *epsilon*
  using *eventually-conj*[*OF A B*] **unfolding** *eventually-sequentially* **by** *blast*

Fifth step: obtain a trivial control on B_n for n smaller than N.

```
define C1 where C1 = Max {B k/B 0 |k. k ≤ N+K+1}
define C where C = max (max C0 C1) (max (D / (a * B 0)) (C0/a))
have C ≥ 1 unfolding C-def using ‹C0 ≥ 1› by auto
```

Now, we can put everything together. If n is large enough, we prove that $B_{n+1} \leq C \max_{i \leq n} B_i$, by contradiction.

```
have geK: B (Suc n) ≤ C * Max {B i |i. i ≤ n} if n > N + K for n
proof (rule ccontr)
  have Suc n ≥ N using that by auto
  let ?h = (λx. (g x + birkhoff-sum f (Suc n) x) / B (Suc n))
  have measure (distr P borel ?h) {x. abs (x) > 2 * a}
         = measure P (?h−‘ {x. abs (x) > 2 * a} ∩ space P)
    by (rule measure-distr, auto)
  also have ... = measure P {x ∈ space M. abs(?h x) > 2 * a}
    by (rule HOL.cong[of measure P], auto)
  finally have A: PS.prob {x ∈ space M. abs(?h x) > 2 * a} > 3 * alpha
    using N(1)[OF ‹Suc n ≥ N›] by auto

  have *: PS.prob {y ∈ space M. C0 ≤ |g y + birkhoff-sum f (Suc n − k) y| /
|B (Suc n − k)|} < epsilon
    if k ∈ {1..<K} for k
  proof −
    have Suc n − k ≥ N using that ‹n > N + K› by auto
    let ?h = (λx. (g x + birkhoff-sum f (Suc n−k) x) / B (Suc n−k))
    have measure (distr P borel ?h) {x. abs (x) ≥ C0}
           = measure P (?h−‘ {x. abs (x) ≥ C0} ∩ space P)
      by (rule measure-distr, auto)
    also have ... = measure P {x ∈ space M. abs(?h x) ≥ C0}
      by (rule HOL.cong[of measure P], auto)
    finally show ?thesis
      using N(2)[OF ‹Suc n − k ≥ N›] by auto
  qed
  have P-le-epsilon: emeasure P {y ∈ space M. C0 ≤ |g y + birkhoff-sum f (Suc
n − k) y| / |B (Suc n − k)|} < ennreal epsilon
    if k ∈ {1..<K} for k
   using *[OF that] ‹epsilon > 0› ennreal-lessI unfolding PS.emeasure-eq-measure
by auto

  assume ¬(B (Suc n) ≤ C * Max {B i |i. i ≤ n})
  then have C * Max {B i |i. i ≤ n} ≤ B (Suc n) by simp
  moreover have C * B 0 ≤ C * Max {B i |i. i ≤ n}
    apply (rule mult-left-mono, rule Max-ge) using ‹C ≥ 1› by auto
  ultimately have C * B 0 ≤ B (Suc n)
    by auto
```

have $(D\ /\ (a * B\ 0)) * B\ 0 \leq C * B\ 0$
apply (*rule mult-right-mono*) **unfolding** *C-def* **using** *Bpos*[*of 0*] **by** *auto*
then have $(D\ /\ (a * B\ 0)) * B\ 0 \leq B\ (Suc\ n)$
using ‹$C * B\ 0 \leq B\ (Suc\ n)$› **by** *simp*
then have $D \leq a * B\ (Suc\ n)$
using *Bpos*[*of 0*] ‹$a > 0$› **by** (*auto simp add*: *divide-simps algebra-simps*)

define *X* **where** $X = \{x \in space\ M.\ abs((g\ x + birkhoff\text{-}sum\ f\ (Suc\ n)\ x)/B(Suc\ n)) > 2 * a\}$
$\cap \{x \in space\ M.\ \forall k < K.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \leq D\}$
$\cap \{x \in space\ M.\ (\sum i \in \{1..<K\}.\ h\ ((T\widehat{\ }\widehat{\ }i)\ x)) \geq 1\}$
have [*measurable*]: $X \in sets\ M$ **unfolding** *X-def* **by** *auto*
have $3 * alpha + (1-alpha) + (1-alpha) \leq$
$PS.prob\ \{x \in space\ M.\ abs((g\ x + birkhoff\text{-}sum\ f\ (Suc\ n)\ x)/B(Suc\ n)) > 2 * a\}$
$+ PS.prob\ \{x \in space\ M.\ \forall k < K.\ abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)) \leq D\}$
$+ PS.prob\ \{x \in space\ M.\ (\sum i \in \{1..<K\}.\ h\ ((T\widehat{\ }\widehat{\ }i)\ x)) \geq 1\}$
using *A D K*(*2*) **by** *auto*
also have ... $\leq 2 + PS.prob\ X$
unfolding *X-def* **by** (*rule PS.sum-measure-le-measure-inter3*, *auto*)
finally have $PS.prob\ X \geq alpha$ **by** *auto*

have *I*: $(\lambda y.\ abs((g\ y + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ y)/\ B\ (Suc\ n - k)))\ ((T\widehat{\ }\widehat{\ }k)\ x) \geq C0$ **if** $x \in X$ $k \in \{1..<K\}$ **for** *x k*
proof −
have $2 * a * B(Suc\ n) \leq abs(g\ x + birkhoff\text{-}sum\ f\ (Suc\ n)\ x)$
using ‹$x \in X$› *Bpos*[*of Suc n*] **unfolding** *X-def* **by** (*auto simp add*: *divide-simps*)
also have ... $= abs(g((T\widehat{\ }\widehat{\ }k)\ x) + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ ((T\widehat{\ }\widehat{\ }k)\ x) + (g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x)))$
using ‹$n > N+K$› ‹$k \in \{1..<K\}$› *birkhoff-sum-cocycle*[*of f k Suc n − k x*] **by** *auto*
also have ... $\leq abs(g((T\widehat{\ }\widehat{\ }k)\ x) + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ ((T\widehat{\ }\widehat{\ }k)\ x)) + abs(g\ x + birkhoff\text{-}sum\ f\ k\ x - g((T\widehat{\ }\widehat{\ }k)\ x))$
by *auto*
also have ... $\leq abs(g((T\widehat{\ }\widehat{\ }k)\ x) + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ ((T\widehat{\ }\widehat{\ }k)\ x)) + D$
using ‹$x \in X$› ‹$k \in \{1..<K\}$› **unfolding** *X-def* **by** *auto*
also have ... $\leq abs(g((T\widehat{\ }\widehat{\ }k)\ x) + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ ((T\widehat{\ }\widehat{\ }k)\ x)) + a * B\ (Suc\ n)$
using ‹$D \leq a * B\ (Suc\ n)$› **by** *simp*
finally have $*$: $a * B\ (Suc\ n) \leq abs(g((T\widehat{\ }\widehat{\ }k)\ x) + birkhoff\text{-}sum\ f\ (Suc\ n - k)\ ((T\widehat{\ }\widehat{\ }k)\ x))$
by *simp*
have $(C0/a) * B\ (Suc\ n - k) \leq C * B\ (Suc\ n - k)$
apply (*rule mult-right-mono*) **unfolding** *C-def* **using** *less-imp-le*[*OF Bpos*] **by** *auto*

also have ... ≤ *C* ∗ *Max* {*B i* |*i*. *i* ≤ *n*}
apply (*rule mult-left-mono*, *rule Max-ge*) **using** ‹*k* ∈ {*1*..<*K*}› ‹*C* ≥ *1*›
by *auto*
also have ... ≤ *B* (*Suc n*)
by *fact*
finally have *C0* ∗ *B* (*Suc n* − *k*) ≤ *a* ∗ *B* (*Suc n*)
using ‹*a*>*0*› **by** (*simp add*: *divide-simps algebra-simps*)
then have *C0* ∗ *B* (*Suc n* − *k*) ≤ *abs*(*g*((*T*^^*k*) *x*) + *birkhoff-sum f* (*Suc n* −*k*) ((*T*^^*k*) *x*))
using ∗ **by** *auto*
then show *?thesis*
using *Bpos*[*of Suc n* − *k*] **by** (*simp add*: *divide-simps*)
qed
have *J*: *1* ≤ (∑*k*∈{*1*..<*K*}. (λ*y*. *h y* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *y*) ((*T*^^*k*) *x*))
if *x* ∈ *X* **for** *x*
proof −
have *x* ∈ *space M*
using ‹*x* ∈ *X*› **unfolding** *X-def* **by** *auto*
have *1* ≤ (∑*k* ∈ {*1*..<*K*}. *h* ((*T*^^*k*) *x*))
using ‹*x* ∈ *X*› **unfolding** *X-def* **by** *auto*
also have ... = (∑*k* ∈ {*1*..<*K*}. *h* ((*T*^^*k*) *x*) ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} ((*T*^^*k*) *x*))
apply (*rule sum.cong*)
unfolding *indicator-def* **using** *I*[*OF* ‹*x* ∈ *X*›] *T-spaceM-stable*(*2*)[*OF* ‹*x* ∈ *space M*›] **by** *auto*
finally show *?thesis* **by** *simp*
qed
have *ennreal alpha* ≤ *emeasure P X*
using ‹*PS.prob X* ≥ *alpha*› **by** (*simp add*: *PS.emeasure-eq-measure*)
also have ... = (∫⁺*x*. *indicator X x* ∂*P*)
by *auto*
also have ... ≤ (∫⁺*x*. (∑*k*∈{*1*..<*K*}. (λ*y*. *h y* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *y*) ((*T*^^*k*) *x*)) ∂*P*)
apply (*rule nn-integral-mono*) **using** *J* **unfolding** *indicator-def* **by** *fastforce*
also have ... = (∑*k*∈{*1*..<*K*}. (∫⁺*x*. (λ*y*. *h y* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *y*) ((*T*^^*k*) *x*) ∂*P*))
by (*rule nn-integral-sum*, *auto*)
also have ... = (∑*k*∈{*1*..<*K*}. (∫⁺*x*. (λ*y*. *h y* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *y*) ((*T*^^*k*) *x*) ∗ *h x* ∂*M*))
unfolding ‹*P* = *density M h*› **by** (*auto intro*!: *sum.cong nn-integral-densityR*[*symmetric*])
also have ... = (∑*k*∈{*1*..<*K*}. (∫⁺*x*. *h x* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *x* ∗ ((*nn-transfer-operator*^^*k*) *h*) *x* ∂*M*))
by (*auto intro*!: *sum.cong nn-transfer-operator-intTn-g*[*symmetric*])
also have ... = (∑*k*∈{*1*..<*K*}. (∫⁺*x*.

((*nn-transfer-operator* ^^ *k*) *h*) *x* ∗ *indicator* {*y* ∈ *space M*. *abs*((*g y* + *birkhoff-sum f* (*Suc n* − *k*) *y*)/ *B* (*Suc n* − *k*)) ≥ *C0*} *x* ∂*P*))
unfolding ‹*P* = *density M h*› **by** (*subst nn-integral-density*, *auto intro*!: *sum.cong simp add*: *algebra-simps*)
also have ... ≤ (∑ *k*∈{*1*..<*K*}. *ennreal delta*)
by (*rule sum-mono*, *rule small-setint*, *auto simp add*: *P-le-epsilon*)
also have ... = *ennreal* (∑ *k*∈{*1*..<*K*}. *delta*)
using *less-imp-le*[*OF* ‹*delta* > *0*›] **by** (*rule sum-ennreal*)
finally have *alpha* ≤ (∑ *k*∈{*1*..<*K*}. *delta*)
apply (*subst ennreal-le-iff*[*symmetric*]) **using** ‹*delta* > *0*› **by** *auto*
also have ... ≤ *K* ∗ *delta*
using ‹*delta* > *0*› **by** *auto*
finally show *False*
unfolding *delta-def* **using** ‹*K* ≥ *1*› ‹*alpha* > *0*› **by** (*auto simp add*: *divide-simps algebra-simps*)
qed

If n is not large, we get the same bound in a trivial way, as there are only finitely many cases to consider and we have adjusted the constant C so that it works for all of them.

have *leK*: *B* (*Suc n*) ≤ *C* ∗ *Max* {*B i* |*i*. *i* ≤ *n*} **if** *n* ≤ *N*+*K* **for** *n*
proof −
have *B* (*Suc n*)/*B 0* ≤ *Max* {*B k*/*B 0* |*k*. *k* ≤ *N*+*K*+*1*}
apply (*rule Max-ge*, *simp*) **using** ‹*n* ≤ *N*+*K*› **by** *auto*
also have ... ≤ *C* **unfolding** *C-def C1-def* **by** *auto*
finally have *B* (*Suc n*) ≤ *C* ∗ *B 0*
using *Bpos*[*of 0*] **by** (*simp add*: *divide-simps*)
also have ... ≤ *C* ∗ *Max* {*B i* |*i*. *i* ≤ *n*}
apply (*rule mult-left-mono*) **apply** (*rule Max-ge*) **using** ‹*C* ≥ *1*› **by** *auto*
finally show *?thesis* **by** *simp*
qed
have *B* (*Suc n*) ≤ *C* ∗ *Max* {*B i* |*i*. *i* ≤ *n*} **for** *n*
using *geK*[*of n*] *leK*[*of n*] **by** *force*
then show *?thesis*
using ‹*C* ≥ *1*› **by** *auto*
qed

Then, we prove the lemma without further assumptions, reducing to the previous case by replacing m with $m+P$. We do this at the level of densities since the addition of measures is not defined in the library (and it would be problematic as measures carry their sigma-algebra, so what should one do when the sigma-algebras do not coincide?)

lemma *upper-bound-C*:
∃ *C*≥*1*. ∀ *n*. *B* (*Suc n*) ≤ *C* ∗ *Max* {*B i*|*i*. *i* ≤ *n*}
proof −

We introduce the density of P, and show that it is almost everywhere finite.

define *h* **where** *h* = *RN-deriv M P*

have [*measurable*]: $h \in$ *borel-measurable M*
unfolding *h-def* **by** *auto*
have *P* [*simp*]: *P = density M h*
unfolding *h-def* **apply** (*rule density-RN-deriv*[*symmetric*]) **using** *PabsM* **by** *auto*
have *space P = space M*
by *auto*
have ∗: *AE x in M. h x* $\neq \infty$
unfolding *h-def* **apply** (*rule RN-deriv-finite*)
using *PS.sigma-finite-measure-axioms PabsM* **by** *auto*
have ∗∗: *null-sets* (*density M* (*λx. 1 + h x*)) = *null-sets M*
by (*rule null-sets-density*, *auto*)

We introduce the new system with invariant measure $M + P$, given by the density $1 + h$.

interpret *A*: *conservative density M* (*λx. 1 + h x*) *T*
apply (*rule conservative-density*) **using** ∗ **by** *auto*
interpret *B*: *conservative-limit T density M* (*λx. 1 + h x*) *P Z f g B*
apply *standard*
using *conv Bpos non-trivial absolutely-continuousI-density*[*OF* ‹*h* ∈ *borel-measurable M*›]
unfolding *absolutely-continuous-def* ∗∗ **by** *auto*

We obtain the result by applying the result above to the new dynamical system. We have to check the additional assumption that the density of P with respect to the new measure $M + P$ is bounded by 1. Since this density if $h/(1 + h)$, this is trivial modulo a computation in ennreal that is not automated (yet?).

have *z*: *1 = ennreal 1* **by** *auto*
have *Trivial*: *a* = (*1+a*) ∗ (*a*/(*1+a*)) **if** *a* ≠ ⊤ **for** *a*::*ennreal*
apply (*cases a*) **apply** *auto* **unfolding** *z ennreal-plus-if* **apply** (*subst divide-ennreal*) **apply** *simp* **apply** *simp*
apply (*subst ennreal-mult′*[*symmetric*]) **using** *that* **by** *auto*
have *Trivial2*: *a* / (*1+a*) ≤ *1* **for** *a*::*ennreal*
apply (*cases a*) **apply** *auto* **unfolding** *z ennreal-plus-if* **apply** (*subst divide-ennreal*) **by** *auto*
show *?thesis*
apply (*rule B.upper-bound-C-aux*[*of λx. h x*/(*1 + h x*)])
using ∗ *Trivial Trivial2* **by** (*auto simp add*: *density-density-eq density-unique-iff*)
qed

The second main upper bound is the following. Again, it proves that $B_{n+1} \leq L \max_{i \leq n} B_i$, for some constant L, but with two differences. First, L only depends on the distribution of Z (which is stronger). Second, this estimate is only proved along a density 1 sequence of times (which is weaker). The first point implies that this lemma will also apply to T^j, with the same L, which amounts to replacing L by $L^{1/j}$, making it in practice arbitrarily close to 1. The second point is problematic at first sight, but for the exceptional times we will use the bound of the previous lemma so this will not really create problems.

For the proof, we split the sum $S_{n+1}f$ as $S_n f + f \circ T^n$. If B_{n+1} is much larger than B_n, we deduce that $S_n f$ is much smaller than $S_{n+1}f$ with large probability, which means that $f \circ T^n$ is larger than anything that has been seen before. Since preimages of distinct events have a measure that tends to 0 along a density 1 subsequence, this can only happen along a density 0 subsequence.

```
lemma upper-bound-L:
  fixes a::real and L::real and alpha::real
  assumes a > 0 alpha > 0 L > 3
          PZ.prob {x. abs (x) > 2 ∗ a} > 3 ∗ alpha
          PZ.prob {x. abs (x) ≥ (L−1) ∗ a} < alpha
  shows ∃ A. lower-asymptotic-density A = 1 ∧ (∀ n∈A. B (Suc n) ≤ L ∗ Max {B
i|i. i ≤ n})
proof −
  define m where m = (λn. Max {B i|i. i ≤ n})
  define K where K = (λn::nat. {x ∈ space M. abs(f x) ∈ {a ∗ L ∗ m n <..< a
∗ L ∗ m (Suc n)}})
  have [measurable]: K n ∈ sets M for n
    unfolding K-def by auto
  have ∗: m n ≤ m p if n ≤ p for n p
    unfolding m-def K-def using that by (auto intro!: Max-mono)
  have K n ∩ K p = {} if n < p for n p
  proof (auto simp add: K-def)
    fix x assume |f x| < a ∗ L ∗ m (Suc n) a ∗ L ∗ m p < |f x|
    moreover have a ∗ L ∗ m (Suc n) ≤ a ∗ L ∗ m p
      using ∗[of Suc n p] that ‹a > 0› ‹L > 3› by auto
    ultimately show False by auto
  qed
  then have disjoint-family K
    unfolding disjoint-family-on-def using nat-neq-iff by auto

  have ∃ A0. lower-asymptotic-density A0 = 1 ∧
          (λn. measure P (space M ∩ (T^^n)−‘(K n)) ∗ indicator A0 n) ⟶ 0
    apply (rule disjoint-sets-measure-density-one-tendsto-zero) apply fact+
    using PabsM by auto
  then obtain A0 where A0: lower-asymptotic-density A0 = 1 (λn. measure P
(space M ∩ (T^^n)−‘(K n)) ∗ indicator A0 n) ⟶ 0
    by blast
  obtain N0 where N0: ⋀n. n ≥ N0 ⟹ measure P (space M ∩ (T^^n)−‘(K
n)) ∗ indicator A0 n < alpha
   using order-tendstoD(2)[OF A0(2) ‹alpha > 0›] unfolding eventually-sequentially
by blast

  have A: eventually (λn. measure (distr P borel (λx. (g x + birkhoff-sum f n x) /
B n)) {x. abs (x) > 2 ∗ a} > 3 ∗ alpha) sequentially
    apply (rule open-set-weak-conv-lsc[of - Z])
```

by (*auto simp add*: *PZ.real-distribution-axioms conv assms*)
have *B*: *eventually* (λ*n*. *measure* (*distr P borel* (λ*x*. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $\geq$ (*L*− *1*) ∗ *a*} < *alpha*) *sequentially*
apply (*rule closed-set-weak-conv-usc*[*of* - *Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv assms*)
obtain *N* **where** *N*: $\bigwedge$*n*. *n* $\geq$ *N* $\Longrightarrow$ *measure* (*distr P borel* (λ*x*. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) > *2* ∗ *a*} > *3* ∗ *alpha*
$\bigwedge$*n*. *n* $\geq$ *N* $\Longrightarrow$ *measure* (*distr P borel* (λ*x*. (*g x* + *birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $\geq$ (*L*−*1*) ∗ *a*} < *alpha*
using *eventually-conj*[*OF A B*] **unfolding** *eventually-sequentially* **by** *blast*

have *I*: *PS.prob* {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f n x*) / *B n*) < (*L*−*1*) ∗ *a*} > *1* − *alpha* **if** *n* $\geq$ *N* **for** *n*
proof −
let *?h* = (λ*x*. (*g x* + *birkhoff-sum f n x*) / *B n*)
have *measure* (*distr P borel ?h*) {*x*. *abs* (*x*) $\geq$ (*L*−*1*) ∗ *a*}
= *measure P* (*?h*−‘ {*x*. *abs* (*x*) $\geq$ (*L*−*1*) ∗ *a*} ∩ *space P*)
by (*rule measure-distr*, *auto*)
also have ... = *measure P* {*x* ∈ *space M*. *abs*(*?h x*) $\geq$ (*L*−*1*) ∗ *a*}
by (*rule HOL.cong*[*of measure P*], *auto*)
finally have *A*: *PS.prob* {*x* ∈ *space M*. *abs*(*?h x*) $\geq$ (*L*−*1*) ∗ *a*} < *alpha*
using *N*(*2*)[*OF that*] **by** *auto*
have ∗: {*x* ∈ *space M*. *abs*(*?h x*) < (*L*−*1*) ∗ *a*} = *space M* − {*x* ∈ *space M*. *abs*(*?h x*) $\geq$ (*L*−*1*) ∗ *a*}
by *auto*
show *?thesis*
unfolding ∗ **using** *A PS.prob-compl* **by** *auto*
qed

have *Main*: *PS.prob* (*space M* ∩ (*T*^^*n*)−‘(*K n*)) > *alpha* **if** *n* $\geq$ *N B* (*Suc n*) > *L* ∗ *m n* **for** *n*
proof −
have *Suc n* $\geq$ *N* **using** *that* **by** *auto*
let *?h* = (λ*x*. (*g x* + *birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*))
have *measure* (*distr P borel ?h*) {*x*. *abs* (*x*) > *2* ∗ *a*}
= *measure P* (*?h*−‘ {*x*. *abs* (*x*) > *2* ∗ *a*} ∩ *space P*)
by (*rule measure-distr*, *auto*)
also have ... = *measure P* {*x* ∈ *space M*. *abs*(*?h x*) > *2* ∗ *a*}
by (*rule HOL.cong*[*of measure P*], *auto*)
finally have *A*: *PS.prob* {*x* ∈ *space M*. *abs*(*?h x*) > *2* ∗ *a*} > *3* ∗ *alpha*
using *N*(*1*)[*OF* ‹*Suc n* $\geq$ *N*›] **by** *auto*

define *X* **where** *X* = {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f n x*) / *B n*) < (*L*−*1*) ∗ *a*}
∩ {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) < (*L*−*1*) ∗ *a*}
∩ {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) > *2* ∗ *a*}
have (*1* − *alpha*) + (*1* − *alpha*) + *3* ∗ *alpha* <

PS.prob {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f n x*) / *B n*) < (*L*−*1*) ∗ *a*}
+ *PS.prob* {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) < (*L*−*1*) ∗ *a*}
+ *PS.prob* {*x* ∈ *space M*. *abs*((*g x* + *birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) > *2* ∗ *a*}
using *A I*[*OF* ‹*n* ≥ *N*›] *I*[*OF* ‹*Suc n* ≥ *N*›] **by** *auto*
also have ... ≤ *2* + *PS.prob X*
unfolding *X-def* **by** (*rule PS.sum-measure-le-measure-inter3*, *auto*)
finally have *PS.prob X* > *alpha* **by** *auto*

have *X* ⊆ *space M* ∩ (*T*^^*n*)−‘(*K n*)
proof
have ∗: *B i* ≤ *m n* **if** *i* ≤ *n* **for** *i*
unfolding *m-def* **by** (*rule Max-ge*, *auto simp add*: *that*)
have ∗∗: *B i* ≤ *B* (*Suc n*) **if** *i* ≤ *Suc n* **for** *i*
proof (*cases i* ≤ *n*)
case *True*
have *m n* ≤ *B* (*Suc n*) / *L*
using ‹*L* ∗ *m n* < *B* (*Suc n*)› ‹*L* > *3*› **by** (*simp add*: *divide-simps algebra-simps*)
also have ... ≤ *B* (*Suc n*)
using *Bpos*[*of Suc n*] ‹*L* > *3*› **by** (*simp add*: *divide-simps algebra-simps*)
finally show *?thesis* **using** ∗[*OF True*] **by** *simp*
next
case *False*
then show *?thesis*
using ‹*i* ≤ *Suc n*› *le-SucE* **by** *blast*
qed
have *m* (*Suc n*) = *B* (*Suc n*)
unfolding *m-def* **by** (*rule Max-eqI*, *auto simp add*: ∗∗)

fix *x* **assume** *x* ∈ *X*
then have *abs* (*g x* + *birkhoff-sum f n x*) < (*L*−*1*) ∗ *a* ∗ *B n*
unfolding *X-def* **using** *Bpos*[*of n*] **by** (*auto simp add*: *algebra-simps divide-simps*)
also have ... ≤ *L* ∗ *a* ∗ *m n*
using ∗[*of n*] ‹*L* > *3*› ‹*a* > *0*› *Bpos*[*of n*] **by** (*auto intro*!: *mult-mono*)
also have ... ≤ *a* ∗ *B* (*Suc n*)
using ‹*B* (*Suc n*) > *L* ∗ *m n*› *less-imp-le* ‹*a* > *0*› **by** *auto*
finally have *A*: *abs* (*g x* + *birkhoff-sum f n x*) < *a* ∗ *B* (*Suc n*)
by *simp*

have *B*: *abs*(*g x* + *birkhoff-sum f* (*Suc n*) *x*) < (*L*−*1*) ∗ *a* ∗ *B* (*Suc n*)
using ‹*x* ∈ *X*› **unfolding** *X-def* **using** *Bpos*[*of Suc n*] **by** (*auto simp add*: *algebra-simps divide-simps*)
have ∗: *f*((*T*^^*n*) *x*) = (*g x* + *birkhoff-sum f* (*Suc n*) *x*) − (*g x* + *birkhoff-sum f n x*)
apply (*auto simp add*: *algebra-simps*)

by (*metis add.commute birkhoff-sum-1*(*2*) *birkhoff-sum-cocycle plus-1-eq-Suc*)
have *abs*(*f*((*T*^^*n*) *x*)) ≤ *abs* (*g x* + *birkhoff-sum f* (*Suc n*) *x*) + *abs*(*g x* + *birkhoff-sum f n x*)
unfolding ∗ **by** *simp*
also have ... < (*L*−*1*) ∗ *a* ∗ *B* (*Suc n*) + *a* ∗ *B* (*Suc n*)
using *A B* **by** *auto*
also have ... = *L* ∗ *a* ∗ *m* (*Suc n*)
using ‹*m* (*Suc n*) = *B* (*Suc n*)› **by** (*simp add*: *algebra-simps*)
finally have *Z1*: *abs*(*f*((*T*^^*n*) *x*)) < *L* ∗ *a* ∗ *m* (*Suc n*)
by *simp*

have *2* ∗ *a* ∗ *B* (*Suc n*) < *abs* (*g x* + *birkhoff-sum f* (*Suc n*) *x*)
using ‹*x* ∈ *X*› **unfolding** *X-def* **using** *Bpos*[*of Suc n*] **by** (*auto simp add*: *algebra-simps divide-simps*)
also have ... = *abs*(*f*((*T*^^*n*) *x*) + (*g x* + *birkhoff-sum f n x*))
unfolding ∗ **by** *auto*
also have ... ≤ *abs*(*f*((*T*^^*n*) *x*)) + *abs* (*g x* + *birkhoff-sum f n x*)
by *auto*
also have ... < *abs*(*f*((*T*^^*n*) *x*)) + *a* ∗ *B* (*Suc n*)
using *A* **by** *auto*
finally have *abs*(*f*((*T*^^*n*) *x*)) > *a* ∗ *B* (*Suc n*)
by *simp*
then have *Z2*: *abs*(*f*((*T*^^*n*) *x*)) > *a* ∗ *L* ∗ *m n*
using *mult-strict-left-mono*[*OF* ‹*B* (*Suc n*) > *L* ∗ *m n*› ‹*a* > *0*›] **by** *auto*

show *x* ∈ *space M* ∩ (*T* ^^ *n*) −‘ *K n*
using *Z1 Z2* ‹*x* ∈ *X*› **unfolding** *K-def X-def* **by** (*auto simp add*: *algebra-simps*)
qed
have *PS.prob X* ≤ *PS.prob* (*space M* ∩ (*T*^^*n*)−‘(*K n*))
by (*rule PS.finite-measure-mono*, *fact*, *auto*)
then show *alpha* < *PS.prob* (*space M* ∩ (*T* ^^ *n*) −‘ *K n*)
using ‹*alpha* < *PS.prob X*› **by** *simp*
qed
define *A* **where** *A* = *A0* ∩ {*N* + *N0*..}
have *lower-asymptotic-density A* = *1*
unfolding *A-def* **by** (*rule lower-asymptotic-density-one-intersection*, *fact*, *simp*)
moreover have *B* (*Suc n*) ≤ *L* ∗ *m n* **if** *n* ∈ *A* **for** *n*
proof (*rule ccontr*)
assume ¬(*B* (*Suc n*) ≤ *L* ∗ *m n*)
then have *L* ∗ *m n* < *B* (*Suc n*) *n* ≥ *N n* ≥ *N0*
using ‹*n* ∈ *A*› **unfolding** *A-def* **by** *auto*
then have *PS.prob* (*space M* ∩ (*T*^^*n*)−‘(*K n*)) > *alpha*
using *Main* **by** *auto*
moreover have *PS.prob* (*space M* ∩ (*T*^^*n*)−‘(*K n*)) ∗ *indicator A0 n* < *alpha*
using *N0*[*OF* ‹*n* ≥ *N0*›] **by** *simp*
moreover have *indicator A0 n* = (*1*::*real*)
using ‹*n* ∈ *A*› **unfolding** *A-def indicator-def* **by** *auto*
ultimately show *False*

by *simp*
qed
ultimately show *?thesis*
unfolding *m-def* **by** *blast*
qed

Now, we combine the two previous statements to prove the main theorem.

theorem *subexponential-growth*:
$(\lambda n.\ max\ 0\ (ln\ (B\ n)\ /n)) \longrightarrow 0$
proof −
obtain *a0* **where** *a0*: $a0 > 0$ $PZ.prob\ \{x.\ abs\ (x) > a0\} > 0$
using *PZ.not-Dirac-0-imp-positive-mass-away-0*[*OF non-trivial*] **by** *blast*
define *a* **where** $a = a0/2$
have $a > 0$ **using** ‹$a0 > 0$› **unfolding** *a-def* **by** *auto*
define *alpha* **where** $alpha = PZ.prob\ \{x.\ abs\ (x) > a0\}/4$
have $alpha > 0$ **unfolding** *alpha-def* **using** *a0* **by** *auto*
have $PZ.prob\ \{x.\ abs\ (x) > 2 * a\} > 3 * alpha$
using *a0* **unfolding** *a-def alpha-def* **by** *auto*

obtain *C0* **where** *C0*: $PZ.prob\ \{x.\ abs(x) \geq C0\} < alpha$ $C0 \geq 3 * a$
using *PZ.random-variable-small-tails*[*OF* ‹$alpha > 0$›, *of* $\lambda x.\ x$] **by** *auto*
define *L* **where** $L = C0/a + 1$
have $PZ.prob\ \{x.\ abs(x) \geq (L-1) * a\} < alpha$
unfolding *L-def* **using** *C0* ‹$a>0$› **by** *auto*
have $L > 3$
unfolding *L-def* **using** *C0* ‹$a > 0$› **by** (*auto simp add*: *divide-simps*)

obtain *C* **where** *C*: $\bigwedge n.\ B\ (Suc\ n) \leq C * Max\ \{B\ i|i.\ i \leq n\}$ $C \geq 1$
using *upper-bound-C* **by** *blast*
have *C2*: $B\ n \leq C * Max\ \{B\ i|i.\ i < n\}$ **if** $n > 0$ **for** *n*
proof −
obtain *m* **where** *m*: $n = Suc\ m$
using ‹$0 < n$› *gr0-implies-Suc* **by** *auto*
have ∗: $i \leq m \longleftrightarrow i < Suc\ m$ **for** *i* **by** *auto*
show *?thesis* **using** *C*(*1*)[*of m*] **unfolding** *m* ∗ **by** *auto*
qed

have *Mainj*: *eventually* $(\lambda n.\ ln\ (B\ n)\ /\ n \leq (1+ln\ L)/j)$ *sequentially* **if** $j > 0$ **for** *j*::*nat*
proof −
have ∗: $\exists A.\ lower\text{-}asymptotic\text{-}density\ A = 1 \wedge (\forall n \in A.\ B\ (j * Suc\ n + k) \leq L * Max\ \{B\ (j * i + k)\ |i.\ i \leq n\})$ **for** *k*
proof −
interpret *Tj0*: *conservative M* $(T \widehat{\ }\widehat{\ } j)$ **using** *conservative-power*[*of j*] **by** *auto*
have ∗: $g\ x + birkhoff\text{-}sum\ f\ k\ x + Tj0.birkhoff\text{-}sum\ (\lambda x.\ birkhoff\text{-}sum\ f\ j\ ((T \widehat{\ }\widehat{\ } k)\ x))\ n\ x = g\ x + birkhoff\text{-}sum\ f\ (j * n + k)\ x$ **for** *x n*
proof −
have $birkhoff\text{-}sum\ f\ (j * n + k)\ x = (\sum i \in \{..<k\} \cup \{k..<j * n + k\}.\ f\ ((T \widehat{\ }\widehat{\ } i)\ x))$

unfolding *birkhoff-sum-def* **by** (*rule sum.cong*, *auto*)
also have ... = ($\sum$ *i* ∈ {..<*k*}. *f* ((*T* ^^ *i*) *x*)) + ($\sum$ *i* ∈ {*k*..<*j* ∗ *n* + *k*}. *f* ((*T* ^^ *i*) *x*))
by (*auto intro*!: *sum.union-disjoint*)
also have ... = *birkhoff-sum f k x* + ($\sum$ *s*<*j*. $\sum$ *i*<*n*. *f* ((*T* ^^ (*i* ∗ *j* + *s*)) ((*T*^^*k*) *x*)))
apply (*subst sum-arith-progression*)
unfolding *birkhoff-sum-def Tj0.birkhoff-sum-def funpow-mult funpow-add'*[*symmetric*]
by (*auto simp add*: *algebra-simps intro*!: *sum.reindex-bij-betw*[*symmetric*] *bij-betw-byWitness*[*of* - λ*a*. *a*−*k*])
also have ... = *birkhoff-sum f k x* + *Tj0.birkhoff-sum* (λ*x*. *birkhoff-sum f j* ((*T* ^^ *k*) *x*)) *n x*
unfolding *birkhoff-sum-def Tj0.birkhoff-sum-def funpow-mult funpow-add'*[*symmetric*]
by (*auto simp add*: *algebra-simps intro*!: *sum.swap*)
finally show *?thesis* **by** *simp*
qed
interpret *Tj*: *conservative-limit T*^^*j M P Z* λ*x*. *birkhoff-sum f j* ((*T*^^*k*) *x*) λ*x*. *g x* + *birkhoff-sum f k x* λ*n*. *B* (*j* ∗ *n* + *k*)
apply *standard*
using *PabsM Bpos non-trivial conv* ‹*j*>*0*› **unfolding** ∗ **by** (*auto intro*!: *weak-conv-m-subseq strict-monoI*)
show *?thesis*
apply (*rule Tj.upper-bound-L*[*OF* ‹*a* > *0*› ‹*alpha* > *0*›]) **by** *fact*+
qed
have ∃ *A*. ∀ *k*. *lower-asymptotic-density* (*A k*) = *1* ∧ (∀ *n*∈*A k*. *B* (*j* ∗ *Suc n* + *k*) ≤ *L* ∗ *Max* {*B* (*j* ∗ *i* + *k*) |*i*. *i* ≤ *n*})
apply (*rule choice*) **using** ∗ **by** *auto*
then obtain *A* **where** *A*: ⋀*k*. *lower-asymptotic-density* (*A k*) = *1* ⋀*k n*. *n* ∈ *A k* ⟹ *B* (*j* ∗ *Suc n* + *k*) ≤ *L* ∗ *Max* {*B* (*j* ∗ *i* + *k*) |*i*. *i* ≤ *n*}
by *blast*
define *Aj* **where** *Aj* = (⋂*k*<*j*. *A k*)
have *lower-asymptotic-density Aj* = *1*
unfolding *Aj-def* **using** *A*(*1*) **by** (*simp add*: *lower-asymptotic-density-one-finite-Intersection*)
define *Bj* **where** *Bj* = *UNIV* − *Aj*
have *upper-asymptotic-density Bj* = *0*
using ‹*lower-asymptotic-density Aj* = *1*›
unfolding *Bj-def lower-upper-asymptotic-density-complement* **by** *simp*

define *M* **where** *M* = (λ*n*. *Max* {*B p* |*p*. *p* < (*n*+*1*) ∗ *j*})
have *B 0* ≤ *M n* **for** *n*
unfolding *M-def* **apply** (*rule Max-ge*, *auto*, *rule exI*[*of* - *0*]) **using** ‹*j* > *0*› **by** *auto*
then have *Mpos*: *M n* > *0* **for** *n*
by (*metis Bpos not-le not-less-iff-gr-or-eq order.strict-trans*)
have *M-L*: *M* (*Suc n*) ≤ *L* ∗ *M n* **if** *n* ∈ *Aj* **for** *n*
proof −
have ∗: *B s* ≤ *L* ∗ *M n* **if** *s* < (*n*+*2*) ∗ *j* **for** *s*
proof (*cases s* < (*n*+*1*) ∗ *j*)
case *True*

have $B\ s \le M\ n$
unfolding *M-def* **apply** (*rule Max-ge*) **using** *True* **by** *auto*
also have ... $\le L * M\ n$ **using** ‹$L > 3$› ‹$M\ n > 0$› **by** *auto*
finally show *?thesis* **by** *simp*
next
case *False*
then obtain k **where** $k < j$ $s = (n+1) * j + k$ **using** ‹$s < (n+2) * j$› *le-Suc-ex* **by** *force*
then have $B\ s = B\ (j * Suc\ n + k)$ **by** (*auto simp add*: *algebra-simps*)
also have ... $\le L * Max\ \{B\ (j * i + k)\ |i.\ i \le n\}$
using $A(2)[of\ n\ k]$ ‹$n \in Aj$› **unfolding** *Aj-def* **using** ‹$k < j$› **by** *auto*
also have ... $\le L * Max\ \{B\ a|a.\ a < (n+1) * j\}$
apply (*rule mult-left-mono*, *rule Max-mono*) **using** ‹$L>3$› **proof** (*auto*)
fix i **assume** $i \le n$ **show** $\exists\, a.\ B\ (j * i + k) = B\ a \land a < j + n * j$
apply (*rule exI*[*of - j * i + k*]) **using** ‹$k < j$› ‹$i \le n$›
by (*auto simp add*: *add-mono-thms-linordered-field*(*3*) *algebra-simps*)
qed
finally show *?thesis* **unfolding** *M-def* **by** *auto*
qed
show *?thesis*
unfolding *M-def* **apply** (*rule Max.boundedI*)
using $*$ **unfolding** *M-def* **using** ‹$j > 0$› **by** *auto*
qed
have *M-C*: $M\ (Suc\ n) \le C\hat{\ }j * M\ n$ **for** n
proof −
have *I*: $Max\ \{B\ s|s.\ s < (n+1) * j + k\} \le C\hat{\ }k * M\ n$ **for** k
proof (*induction k*)
case *0*
show *?case*
apply (*rule Max.boundedI*) **unfolding** *M-def* **using** ‹$j > 0$› **by** *auto*
next
case (*Suc k*)
have $*$: $B\ s \le C * C\ \hat{}\ k * M\ n$ **if** $s < Suc\ (j + n * j + k)$ **for** s
proof (*cases* $s < j + n * j + k$)
case *True*
then have $B\ s \le C\hat{\ }k * M\ n$ **using** *iffD1*[*OF Max-le-iff*, *OF - - Suc.IH*] **by** *auto*
also have ... $\le C * C\hat{\ }k * M\ n$ **using** ‹$C \ge 1$› ‹$M\ n > 0$› **by** *auto*
finally show *?thesis* **by** *simp*
next
case *False*
then have $s = j + n * j + k$ **using** *that* **by** *auto*
then have $B\ s \le C * Max\ \{B\ s|s.\ s < (n+1) * j + k\}$ **using** *C2*[*of s*] **using** ‹$j > 0$› **by** *auto*
also have ... $\le C * C\hat{\ }k * M\ n$ **using** *Suc.IH* ‹$C \ge 1$› **by** *auto*
finally show *?thesis* **by** *simp*
qed
show *?case*
apply (*rule Max.boundedI*) **using** ‹$j > 0$› $*$ **by** *auto*

```
    qed
   show ?thesis using I[of j] unfolding M-def by (auto simp add: algebra-simps)
  qed
  have I: ln (M n) ≤ ln (M 0) + n * ln L + card (Bj ∩ {..<n}) * ln (C^j) for
n
  proof (induction n)
    case 0
    show ?case by auto
  next
    case (Suc n)
    show ?case
    proof (cases n ∈ Bj)
      case True
      then have *: Bj ∩ {..<Suc n} = Bj ∩ {..<n} ∪ {n} by auto
      have **: card (Bj ∩ {..<Suc n}) = card (Bj ∩ {..<n}) + card {n}
        unfolding * by (rule card-Un-disjoint, auto)

      have ln (M (Suc n)) ≤ ln (C^j * M n)
        using M-C ‹⋀n. 0 < M n› less-le-trans ln-le-cancel-iff by blast
      also have ... = ln (M n) + ln (C^j)
        using ‹C ≥ 1› ‹0 < M n› ln-mult by auto
      also have ... ≤ ln (M 0) + n * ln L + card (Bj ∩ {..<n}) * ln (C^j) + ln
(C^j)
        using Suc.IH by auto
      also have ... = ln (M 0) + n * ln L + card (Bj ∩ {..<Suc n}) * ln (C^j)
        using ** by (auto simp add: algebra-simps)
      also have ... ≤ ln (M 0) + (Suc n) * ln L + card (Bj ∩ {..<Suc n}) * ln
(C^j)
        using ‹L > 3› by auto
      finally show ?thesis by auto
    next
      case False
      have M (Suc n) ≤ L * M n
        apply (rule M-L) using False unfolding Bj-def by auto
      then have ln (M (Suc n)) ≤ ln (L * M n)
        using ‹⋀n. 0 < M n› less-le-trans ln-le-cancel-iff by blast
      also have ... = ln (M n) + ln L
        using ‹L > 3› ‹0 < M n› ln-mult by auto
      also have ... ≤ ln (M 0) + Suc n * ln L + card (Bj ∩ {..<n}) * ln (C^j)
        using Suc.IH by (auto simp add: algebra-simps)
      also have ... ≤ ln (M 0) + Suc n * ln L + card (Bj ∩ {..<Suc n}) * ln
(C^j)
        using ‹C ≥ 1› by (auto intro!: mult-right-mono card-mono)
      finally show ?thesis by auto
    qed
  qed
  have ln (M n)/n ≤ ln (M 0)* (1/n) + ln L + (card (Bj ∩ {..<n})/n) * ln
(C^j) if n≥1 for n
    using that apply (auto simp add: algebra-simps divide-simps)
```

by (*metis* (*no-types*, *opaque-lifting*) *I add.assoc mult.commute mult-left-mono of-nat-0-le-iff semiring-normalization-rules*(*34*))
then have *A*: *eventually* (λ*n*. *ln* (*M n*)/*n* ≤ *ln* (*M 0*)∗ (*1*/*n*) + *ln L* + (*card* (*Bj* ∩ {..<*n*})/*n*) ∗ *ln* (*C*^*j*)) *sequentially*
unfolding *eventually-sequentially* **by** *blast*

have ∗: (λ*n*. *ln* (*M 0*)∗(*1*/*n*) + *ln L* + (*card* (*Bj* ∩ {..<*n*})/*n*) ∗ *ln* (*C*^*j*)) ⟶ *ln* (*M 0*) ∗ *0* + *ln L* + *0* ∗*ln* (*C*^*j*)
by (*intro tendsto-intros upper-asymptotic-density-zero-lim*, *fact*)
have *B*: *eventually* (λ*n*. *ln* (*M 0*)∗(*1*/*n*) + *ln L* + (*card* (*Bj* ∩ {..<*n*})/*n*) ∗ *ln* (*C*^*j*) < *1* + *ln L*) *sequentially*
by (*rule order-tendstoD*[*OF* ∗], *auto*)
have *eventually* (λ*n*. *ln* (*M n*)/*n* < *1* + *ln L*) *sequentially*
using *eventually-conj*[*OF A B*] **by** (*simp add*: *eventually-mono*)
then obtain *N* **where** *N*: ⋀*n*. *n* ≥ *N* ⟹ *ln* (*M n*)/*n* < *1* + *ln L*
unfolding *eventually-sequentially* **by** *blast*
have *ln* (*B p*) / *p* ≤ (*1*+*ln L*) / *j* **if** *p* ≥ (*N*+*1*) ∗ *j* **for** *p*
proof −
define *n* **where** *n* = *p div j*
have *n* ≥ *N*+*1* **unfolding** *n-def* **using** *that*
by (*metis* ‹*0* < *j*› *div-le-mono div-mult-self-is-m*)
then have *n* ≥ *N n* ≥ *1* **by** *auto*
have ∗: *p* < (*n*+*1*) ∗ *j n* ∗ *j* ≤ *p*
unfolding *n-def* **using** ‹*j* > *0*› *dividend-less-div-times* **by** *auto*
have *B p* ≤ *M n*
unfolding *M-def* **apply** (*rule Max-ge*) **using** ∗ **by** *auto*
then have *ln* (*B p*) ≤ *ln* (*M n*)
using *Bpos Mpos ln-le-cancel-iff* **by** *blast*
also have ... ≤ *n* ∗ (*1*+*ln L*)
using *N*[*OF* ‹*n* ≥ *N*›] ‹*n* ≥ *1*› **by** (*auto simp add*: *divide-simps algebra-simps*)
also have ... ≤ (*p*/*j*) ∗ (*1*+*ln L*)
apply (*rule mult-right-mono*) **using** ∗(*2*) ‹*j* > *0*› ‹*L* > *3*›
apply (*auto simp add*: *divide-simps algebra-simps*)
using *of-nat-mono* **by** *fastforce*
finally show *?thesis*
using ‹*j* > *0*› *that* **by** (*simp add*: *algebra-simps divide-simps*)
qed
then show *eventually* (λ*p*. *ln* (*B p*) / *p* ≤ (*1*+*ln L*)/*j*) *sequentially*
unfolding *eventually-sequentially* **by** *auto*
qed
show (λ*n*. *max 0* (*ln* (*B n*) / *real n*)) ⟶ *0*
proof (*rule order-tendstoI*)
fix *e*::*real* **assume** *e* > *0*
have ∗: (λ*j*. (*1*+*ln L*) ∗ (*1*/*j*)) ⟶ (*1*+*ln L*) ∗ *0*
by (*intro tendsto-intros*)
have *eventually* (λ*j*. (*1*+*ln L*) ∗ (*1*/*j*) < *e*) *sequentially*
apply (*rule order-tendstoD*[*OF* ∗]) **using** ‹*e* > *0*› **by** *auto*
then obtain *j*::*nat* **where** *j*: *j* > *0* (*1*+*ln L*) ∗ (*1*/*j*) < *e*
unfolding *eventually-sequentially* **using** *add.right-neutral le-eq-less-or-eq* **by**

fastforce
  show *eventually* (*λn. max 0* (*ln* (*B n*) / *real n*) < *e*) *sequentially*
    using *Mainj*[*OF* ‹*j* > *0*›] *j*(*2*) ‹*e* > *0*› **by** (*simp add*: *eventually-mono*)
  qed (*simp add*: *max.strict-coboundedI1*)
qed

end

13.3 Normalizing sequences grow at most polynomially in probability preserving systems

In probability preserving systems, normalizing sequences grow at most polynomially. The proof, also given in [Gou18], is considerably easier than the conservative case. We prove that $B_{n+1} \leq CB_n$ (more precisely, this only holds if B_{n+1} is large enough), by arguing that $S_{n+1}f = S_nf + f \circ T^n$, where $f \circ T^n$ is negligible if B_{n+1} is large thanks to the measure preservation. We also prove that $B_{2n} \leq EB_n$, by writing $S_{2n}f = S_nf + S_nf \circ T^n$ and arguing that the two terms on the right have the same distribution. Finally, combining these two estimates, the polynomial growth follows readily.

locale *pmpt-limit* =
  pmpt M + *PZ*: *real-distribution Z*
    for *M*::*'a measure* **and** *Z*::*real measure* +
  fixes *f*::*'a* ⇒ *real* **and** *B*::*nat* ⇒ *real*
  assumes *Bpos*: ⋀*n. B n* > *0*
    and *M* [*measurable*]: *f* ∈ *borel-measurable M*
    and *non-trivial*: *PZ.prob* {*0*} < *1*
    and *conv*: *weak-conv-m* (*λn. distr P borel* (*λx.* (*birkhoff-sum f n x*) / *B n*)) *Z*
begin

First, we prove that $B_{n+1} \leq CB_n$ if B_{n+1} is large enough.

lemma *upper-bound-CD*:
  ∃ *C D.* (∀ *n. B* (*Suc n*) ≤ *D* ∨ *B* (*Suc n*) ≤ *C* ∗ *B n*) ∧ *C* ≥ *1*
proof −
  obtain *a* **where** *a*: *a* > *0 PZ.prob* {*x. abs* (*x*) > *a*} > *0*
    using *PZ.not-Dirac-0-imp-positive-mass-away-0*[*OF non-trivial*] **by** *blast*
  define *alpha* **where** *alpha* = *PZ.prob* {*x. abs* (*x*) > *a*}/*4*
  have *alpha* > *0* **unfolding** *alpha-def* **using** *a* **by** *auto*
  have *A*: *PZ.prob* {*x. abs* (*x*) > *a*} > *3* ∗ *alpha*
    using *a* **unfolding** *alpha-def* **by** *auto*

  obtain *C0* **where** *C0*: *PZ.prob* {*x. abs*(*x*) ≥ *C0*} < *alpha C0* ≥ *a*
    using *PZ.random-variable-small-tails*[*OF* ‹*alpha* > *0*›, *of λx. x*] **by** *auto*

  have *A*: *eventually* (*λn. measure* (*distr M borel* (*λx.* (*birkhoff-sum f n x*) / *B n*))
{*x. abs* (*x*) > *a*} > *3* ∗ *alpha*) *sequentially*
    apply (*rule open-set-weak-conv-lsc*[*of - Z*])

by (*auto simp add*: *PZ.real-distribution-axioms conv A*)
have *B*: *eventually* (λn. *measure* (*distr M borel* (λx. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) $\geq$ *C0*} < *alpha*) *sequentially*
apply (*rule closed-set-weak-conv-usc*[*of* - *Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv C0*)
obtain *N* **where** *N*: $\bigwedge n$. $n \geq N \Longrightarrow$ *measure* (*distr M borel* (λx. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs x* > *a*} > *3* $*$ *alpha*
$\bigwedge n$. $n \geq N \Longrightarrow$ *measure* (*distr M borel* (λx. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs x* $\geq$ *C0*} < *alpha*
using *eventually-conj*[*OF A B*] **unfolding** *eventually-sequentially* **by** *blast*

obtain *Cf* **where** *Cf*: *prob* {*x* $\in$ *space M*. *abs*(*f x*) $\geq$ *Cf*} < *alpha Cf* $\geq$ *1*
using *random-variable-small-tails*[*OF* ‹*alpha* > *0*› *M*] **by** *auto*
have *Main*: *B* (*Suc n*) $\leq$ (*2*$*$*C0*/*a*) $*$ *B n* **if** *n* $\geq$ *N B* (*Suc n*) $\geq$ *2* $*$ *Cf*/*a* **for** *n*
proof −
have *Suc n* $\geq$ *N* **using** *that* **by** *auto*
let *?h* = (λx. (*birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*))
have *measure* (*distr M borel ?h*) {*x*. *abs* (*x*) > *a*}
= *measure M* (*?h*−‘ {*x*. *abs* (*x*) > *a*} $\cap$ *space M*)
by (*rule measure-distr*, *auto*)
also have ... = *prob* {*x* $\in$ *space M*. *abs*(*?h x*) > *a*}
by (*rule HOL.cong*[*of measure M*], *auto*)
finally have *A*: *prob* {*x* $\in$ *space M*. *abs*(*?h x*) > *a*} > *3* $*$ *alpha*
using *N*(*1*)[*OF* ‹*Suc n* $\geq$ *N*›] **by** *auto*

let *?h* = (λx. (*birkhoff-sum f n x*) / *B n*)
have *measure* (*distr M borel ?h*) {*x*. *abs* (*x*) $\geq$ *C0*}
= *measure M* (*?h*−‘ {*x*. *abs* (*x*) $\geq$ *C0*} $\cap$ *space M*)
by (*rule measure-distr*, *auto*)
also have ... = *measure M* {*x* $\in$ *space M*. *abs*(*?h x*) $\geq$ *C0*}
by (*rule HOL.cong*[*of measure M*], *auto*)
finally have *B0*: *prob* {*x* $\in$ *space M*. *abs*(*?h x*) $\geq$ *C0*} < *alpha*
using *N*(*2*)[*OF* ‹*n* $\geq$ *N*›] **by** *auto*
have $*$: {*x* $\in$ *space M*. *abs*(*?h x*) < *C0*} = *space M* − {*x* $\in$ *space M*. *abs*(*?h x*) $\geq$ *C0*}
by *auto*
have *B*: *prob* {*x* $\in$ *space M*. *abs*(*?h x*) < *C0*} > *1*− *alpha*
unfolding $*$ **using** *B0 prob-compl* **by** *auto*

have *prob* {*x* $\in$ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) $\geq$ *Cf*} = *prob* ((*T*^^*n*)−‘{*x* $\in$ *space M*. *abs*(*f x*) $\geq$ *Cf*} $\cap$ *space M*)
by (*rule HOL.cong*[*of prob*], *auto*)
also have ... = *prob* {*x* $\in$ *space M*. *abs*(*f x*) $\geq$ *Cf*}
using *T-vrestr-same-measure*(*2*)[*of* {*x* $\in$ *space M*. *abs*(*f x*) $\geq$ *Cf*} *n*]
unfolding *vimage-restr-def* **by** *auto*
finally have *C0*: *prob* {*x* $\in$ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) $\geq$ *Cf*} < *alpha*
using *Cf* **by** *simp*
have $*$: {*x* $\in$ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) < *Cf*} = *space M* − {*x* $\in$ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) $\geq$ *Cf*}

by *auto*
have *C*: *prob* {*x* ∈ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) < *Cf*} > *1*− *alpha*
unfolding ∗ **using** *C0 prob-compl* **by** *auto*

define *X* **where** *X* = {*x* ∈ *space M*. *abs*((*birkhoff-sum f n x*) / *B n*) < *C0*}
∩ {*x* ∈ *space M*. *abs*((*birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) > *a*}
∩ {*x* ∈ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) < *Cf*}
have (*1* − *alpha*) + *3* ∗ *alpha* + (*1* − *alpha*) <
prob {*x* ∈ *space M*. *abs*((*birkhoff-sum f n x*) / *B n*) < *C0*}
+ *prob* {*x* ∈ *space M*. *abs*((*birkhoff-sum f* (*Suc n*) *x*) / *B* (*Suc n*)) > *a*}
+ *prob* {*x* ∈ *space M*. *abs*(*f* ((*T*^^*n*) *x*)) < *Cf*}
using *A B C* **by** *auto*
also have ... ≤ *2* + *prob X*
unfolding *X-def* **by** (*rule sum-measure-le-measure-inter3*, *auto*)
finally have *prob X* > *alpha* **by** *auto*
then have *X* ≠ {} **using** ‹*alpha* > *0*› **by** *auto*
then obtain *x* **where** *x* ∈ *X* **by** *auto*
have ∗: *abs*(*birkhoff-sum f n x*) ≤ *C0* ∗ *B n*
abs(*birkhoff-sum f* (*Suc n*) *x*) ≥ *a* ∗ *B* (*Suc n*)
abs(*f*((*T*^^*n*) *x*)) ≤ *Cf*
using ‹*x* ∈ *X*› *Bpos*[*of n*] *Bpos*[*of Suc n*] **unfolding** *X-def* **by** (*auto simp add*: *divide-simps*)
have *a* ∗ *B* (*Suc n*) ≤ *abs*(*birkhoff-sum f* (*Suc n*) *x*)
using ∗ **by** *simp*
also have ... = *abs*(*birkhoff-sum f n x* + *f* ((*T*^^*n*) *x*))
by (*metis Groups.add-ac*(*2*) *One-nat-def birkhoff-sum-1*(*3*) *birkhoff-sum-cocycle plus-1-eq-Suc*)
also have ... ≤ *C0* ∗ *B n* + *Cf*
using ∗ **by** *auto*
also have ... ≤ *C0* ∗ *B n* + (*a*/*2*) ∗ *B* (*Suc n*)
using ‹*B* (*Suc n*) ≥ *2* ∗ *Cf*/*a*› ‹*a* > *0*› **by** (*auto simp add*: *divide-simps algebra-simps*)
finally show *B* (*Suc n*) ≤ (*2* ∗ *C0*/*a*) ∗ *B n*
using ‹*a* > *0*› **by** (*auto simp add*: *divide-simps algebra-simps*)
qed
define *C1* **where** *C1* = *Max* {*B*(*Suc n*)/*B n* |*n*. *n* ≤ *N*}
have ∗: *B* (*Suc n*) ≤ *max* ((*2* ∗ *C0*/*a*)) *C1* ∗ *B n* **if** *B* (*Suc n*) > *2* ∗ *Cf*/*a* **for** *n*
proof (*cases n* > *N*)
case *True*
then show *?thesis*
using *Main*[*OF less-imp-le*[*OF* ‹*n* > *N*›] *less-imp-le*[*OF that*]] *Bpos*[*of n*]
by (*meson max.cobounded1 order-trans mult-le-cancel-right-pos*)
next
case *False*
then have *n* ≤ *N* **by** *simp*
have *B*(*Suc n*)/*B n* ≤ *C1*
unfolding *C1-def* **apply** (*rule Max-ge*) **using** ‹*n* ≤ *N*› **by** *auto*

then have *B* (*Suc n*) ≤ *C1* ∗ *B n*
using *Bpos*[*of n*] **by** (*simp add*: *divide-simps*)
then show *?thesis*
using *Bpos*[*of n*] **by** (*meson max.cobounded2 order-trans mult-le-cancel-right-pos*)
qed
show *?thesis*
apply (*rule exI*[*of - max* ((*2* ∗ *C0*/*a*)) *C1*], *rule exI*[*of - 2* ∗ *Cf*/*a*])
using ∗ *linorder-not-less* ‹*C0* ≥ *a*› ‹*a* > *0*› **by** (*auto intro*!: *max.coboundedI1*)
qed

Second, we prove that $B_{2n} \leq EB_n$.

lemma *upper-bound-E*:
∃ *E*. ∀ *n*. *B* (*2* ∗ *n*) ≤ *E* ∗ *B n*
proof −
obtain *a* **where** *a*: *a* > *0 PZ.prob* {*x*. *abs* (*x*) > *a*} > *0*
using *PZ.not-Dirac-0-imp-positive-mass-away-0*[*OF non-trivial*] **by** *blast*
define *alpha* **where** *alpha* = *PZ.prob* {*x*. *abs* (*x*) > *a*}/*4*
have *alpha* > *0* **unfolding** *alpha-def* **using** *a* **by** *auto*
have *A*: *PZ.prob* {*x*. *abs* (*x*) > *a*} > *3* ∗ *alpha*
using *a* **unfolding** *alpha-def* **by** *auto*

obtain *C0* **where** *C0*: *PZ.prob* {*x*. *abs*(*x*) ≥ *C0*} < *alpha C0* ≥ *a*
using *PZ.random-variable-small-tails*[*OF* ‹*alpha* > *0*›, *of* λ*x*. *x*] **by** *auto*

have *A*: *eventually* (λ*n*. *measure* (*distr M borel* (λ*x*. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) > *a*} > *3* ∗ *alpha*) *sequentially*
apply (*rule open-set-weak-conv-lsc*[*of - Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv A*)
have *B*: *eventually* (λ*n*. *measure* (*distr M borel* (λ*x*. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs* (*x*) ≥ *C0*} < *alpha*) *sequentially*
apply (*rule closed-set-weak-conv-usc*[*of - Z*])
by (*auto simp add*: *PZ.real-distribution-axioms conv C0*)
obtain *N* **where** *N*: ⋀*n*. *n* ≥ *N* ⟹ *measure* (*distr M borel* (λ*x*. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs x* > *a*} > *3* ∗ *alpha*
⋀*n*. *n* ≥ *N* ⟹ *measure* (*distr M borel* (λ*x*. (*birkhoff-sum f n x*) / *B n*)) {*x*. *abs x* ≥ *C0*} < *alpha*
using *eventually-conj*[*OF A B*] **unfolding** *eventually-sequentially* **by** *blast*

have *Main*: *B* (*2* ∗ *n*) ≤ (*2*∗*C0*/*a*) ∗ *B n* **if** *n* ≥ *N* **for** *n*
proof −
have *2* ∗ *n* ≥ *N* **using** *that* **by** *auto*
let *?h* = (λ*x*. (*birkhoff-sum f* (*2* ∗ *n*) *x*) / *B* (*2* ∗ *n*))
have *measure* (*distr M borel ?h*) {*x*. *abs* (*x*) > *a*}
= *measure M* (*?h*−‘ {*x*. *abs* (*x*) > *a*} ∩ *space M*)
by (*rule measure-distr*, *auto*)
also have ... = *prob* {*x* ∈ *space M*. *abs*(*?h x*) > *a*}
by (*rule HOL.cong*[*of measure M*], *auto*)
finally have *A*: *prob* {*x* ∈ *space M*. *abs*((*birkhoff-sum f* (*2* ∗ *n*) *x*) / *B* (*2* ∗ *n*)) > *a*} > *3* ∗ *alpha*

using *N(1)[OF ‹2 ∗ n ≥ N›]* **by** *auto*

let *?h = (λx. (birkhoff-sum f n x) / B n)*
have *measure (distr M borel ?h) {x. abs (x) ≥ C0}*
= measure M (?h−‘ {x. abs (x) ≥ C0} ∩ space M)
by *(rule measure-distr, auto)*
also have *... = measure M {x ∈ space M. abs(?h x) ≥ C0}*
by *(rule HOL.cong[of measure M], auto)*
finally have *B0: prob {x ∈ space M. abs(?h x) ≥ C0} < alpha*
using *N(2)[OF ‹n ≥ N›]* **by** *auto*
have *∗: {x ∈ space M. abs(?h x) < C0} = space M − {x ∈ space M. abs(?h x) ≥ C0}*
by *auto*
have *B: prob {x ∈ space M. abs((birkhoff-sum f n x) / B n) < C0} > 1− alpha*
unfolding *∗* **using** *B0 prob-compl* **by** *auto*

have *prob {x ∈ space M. abs(?h ((T^^n) x)) < C0} = prob ((T^^n)−‘{x ∈ space M. abs(?h x) < C0} ∩ space M)*
by *(rule HOL.cong[of prob], auto)*
also have *... = prob {x ∈ space M. abs(?h x) < C0}*
using *T-vrestr-same-measure(2)[of {x ∈ space M. abs(?h x) < C0} n]*
unfolding *vimage-restr-def* **by** *auto*
finally have *C: prob {x ∈ space M. abs((birkhoff-sum f n ((T^^n) x)) / B n) < C0} > 1− alpha*
using *B* **by** *simp*

define *X* **where** *X = {x ∈ space M. abs((birkhoff-sum f n x) / B n) < C0}*
∩ {x ∈ space M. abs((birkhoff-sum f (2∗ n) x) / B (2∗ n)) > a}
∩ {x ∈ space M. abs((birkhoff-sum f n ((T^^n) x)) / B n) < C0}
have *(1 − alpha) + 3 ∗ alpha + (1 − alpha) <*
prob {x ∈ space M. abs((birkhoff-sum f n x) / B n) < C0}
+ prob {x ∈ space M. abs((birkhoff-sum f (2∗ n) x) / B (2∗ n)) > a}
+ prob {x ∈ space M. abs((birkhoff-sum f n ((T^^n) x)) / B n) < C0}
using *A B C* **by** *auto*
also have *... ≤ 2 + prob X*
unfolding *X-def* **by** *(rule sum-measure-le-measure-inter3, auto)*
finally have *prob X > alpha* **by** *auto*
then have *X ≠ {}* **using** *‹alpha > 0›* **by** *auto*
then obtain *x* **where** *x ∈ X* **by** *auto*
have *∗: abs(birkhoff-sum f n x) ≤ C0 ∗ B n*
abs((birkhoff-sum f (2 ∗ n) x)) ≥ a ∗ B (2 ∗ n)
abs((birkhoff-sum f n ((T^^n) x))) ≤ C0 ∗ B n
using *‹x ∈ X› Bpos[of n] Bpos[of 2∗ n]* **unfolding** *X-def* **by** *(auto simp add: divide-simps)*
have *a ∗ B (2 ∗ n) ≤ abs(birkhoff-sum f (2 ∗ n) x)*
using *∗* **by** *simp*
also have *... = abs(birkhoff-sum f n x + birkhoff-sum f n ((T^^n) x))*

unfolding *birkhoff-sum-cocycle*[*of f n n x*, *symmetric*] **by** (*simp add*: *mult-2*)
also have ... ≤ *2* ∗ *C0* ∗ *B n*
using ∗ **by** *auto*
finally show *B* (*2* ∗ *n*) ≤ (*2* ∗ *C0*/*a*) ∗ *B n*
using ‹*a* > *0*› **by** (*auto simp add*: *divide-simps algebra-simps*)
qed
define *C1* **where** *C1* = *Max* {*B*(*2* ∗ *n*)/*B n* |*n*. *n* ≤ *N*}
have ∗: *B* (*2*∗*n*) ≤ *max* ((*2* ∗ *C0*/*a*)) *C1* ∗ *B n* **for** *n*
proof (*cases n* > *N*)
case *True*
then show *?thesis*
using *Main*[*OF less-imp-le*[*OF* ‹*n* > *N*›]] *Bpos*[*of n*]
by (*meson max.cobounded1 order-trans mult-le-cancel-right-pos*)
next
case *False*
then have *n* ≤ *N* **by** *simp*
have *B*(*2*∗*n*)/*B n* ≤ *C1*
unfolding *C1-def* **apply** (*rule Max-ge*) **using** ‹*n* ≤ *N*› **by** *auto*
then have *B* (*2*∗*n*) ≤ *C1* ∗ *B n*
using *Bpos*[*of n*] **by** (*simp add*: *divide-simps*)
then show *?thesis*
using *Bpos*[*of n*] **by** (*meson max.cobounded2 order-trans mult-le-cancel-right-pos*)
qed
show *?thesis*
apply (*rule exI*[*of* - *max* ((*2* ∗ *C0*/*a*)) *C1*])
using ∗ **by** *auto*
qed

Finally, we combine the estimates in the two lemmas above to show that B_n grows at most polynomially.

theorem *polynomial-growth*:
∃ *C K*. ∀ *n*>*0*. *B n* ≤ *C* ∗ (*real n*)^*K*
proof −
obtain *C D* **where** *C*: *C* ≥ *1* ⋀*n*. *B* (*Suc n*) ≤ *D* ∨ *B* (*Suc n*) ≤ *C* ∗ *B n*
using *upper-bound-CD* **by** *blast*
obtain *E* **where** *E*: ⋀*n*. *B* (*2* ∗ *n*) ≤ *E* ∗ *B n*
using *upper-bound-E* **by** *blast*
have *E* ≥ *1* **using** *E*[*of 0*] *Bpos*[*of 0*] **by** *auto*

obtain *k*::*nat* **where** *log 2* (*max C E*) ≤ *k*
using *real-arch-simple*[*of log 2* (*max C E*)] **by** *blast*
then have *max C E* ≤ *2*^*k*
by (*meson less-log-of-power not-less one-less-numeral-iff semiring-norm*(*76*))
then have *C* ≤ *2*^*k E* ≤ *2*^*k*
by *auto*

define *P* **where** *P* = *max D* (*B 0*)
have *P* > *0* **unfolding** *P-def* **using** *Bpos*[*of 0*] **by** *auto*
have *Main*: ⋀*n*. *n* < *2*^*r* ⟹ *B n* ≤ *P* ∗ *2*^(*2* ∗ *k* ∗ *r*) **for** *r*

proof (*induction r*)
  case *0*
  fix *n*::*nat* **assume** $n < 2\hat{}0$
  then show $B\ n \leq P * 2\ \hat{}\ (2 * k * 0)$
    unfolding *P-def* **by** *auto*
next
  case (*Suc r*)
  fix *n*::*nat* **assume** $n < 2\hat{}(Suc\ r)$
  consider *even n* | $B\ n \leq D$ | *odd n* $\wedge$ $B\ n > D$ **by** *linarith*
  then show $B\ n \leq P * 2\ \hat{}\ (2 * k * Suc\ r)$
  proof (*cases*)
    case *1*
    then obtain *m* **where** *m*: $n = 2 * m$ **by** (*rule evenE*)
    have $m < 2\hat{}r$
      using ‹$n < 2\hat{}(Suc\ r)$› **unfolding** *m* **by** *auto*
    then have $*$: $B\ m \leq P * 2\hat{}(2 * k * r)$
      using *Suc.IH* **by** *auto*
    have $B\ n \leq E * B\ m$
      unfolding *m* **using** *E* **by** *simp*
    also have ... $\leq 2\hat{}k * B\ m$
      apply (*rule mult-right-mono*[*OF - less-imp-le*[*OF Bpos*[*of m*]]])
      using ‹$E \leq 2\hat{}k$› **by** *simp*
    also have ... $\leq 2\hat{}k * (P * 2\hat{}(2 * k * r))$
      apply (*rule mult-left-mono*[*OF* $*$]) **by** *auto*
    also have ... $= P * 2\hat{}(2 * k * r + k)$
      by (*auto simp add*: *algebra-simps power-add*)
    also have ... $\leq P * 2\hat{}(2 * k * Suc\ r)$
      apply (*rule mult-left-mono*) **using** ‹$P > 0$› **by** *auto*
    finally show *?thesis* **by** *simp*
  next
    case *2*
    have $D \leq P * 1$
      unfolding *P-def* **by** *auto*
    also have ... $\leq P * 2\hat{}(2 * k * Suc\ r)$
      by (*rule mult-left-mono*[*OF - less-imp-le*[*OF* ‹$P > 0$›]], *auto*)
    finally show *?thesis* **using** *2* **by** *simp*
  next
    case *3*
    then obtain *m* **where** *m*: $n = 2 * m + 1$
      using *oddE* **by** *blast*
    have $m < 2\hat{}r$
      using ‹$n < 2\hat{}(Suc\ r)$› **unfolding** *m* **by** *auto*
    then have $*$: $B\ m \leq P * 2\hat{}(2 * k * r)$
      using *Suc.IH* **by** *auto*
    have $B\ n > D$ **using** *3* **by** *auto*
    then have $B\ n \leq C * B\ (2 * m)$
      unfolding *m* **using** *C*(*2*)[*of* $2 * m$] **by** *auto*
    also have ... $\leq C * (E * B\ m)$
      apply (*rule mult-left-mono*) **using** ‹$C \geq 1$› *E*[*of m*] **by** *auto*

also have ... ≤ *2^k ∗ (2^k ∗ B m)*
apply (*intro mult-mono*) **using** ‹*C ≤ 2^k*› ‹*C ≥ 1*› ‹*E ≥ 1*› ‹*E ≤ 2^k*› *Bpos*[*of m*] **by** *auto*
also have ... ≤ *2^k ∗ (2^k ∗ (P ∗ 2^(2 ∗ k ∗ r)))*
apply (*intro mult-left-mono*) **using** ∗ **by** *auto*
also have ... = *P ∗ 2^(2 ∗ k ∗ Suc r)*
using ‹*P > 0*› **by** (*simp add*: *algebra-simps divide-simps mult-2-right power-add*)
finally show *?thesis* **by** *simp*
qed
qed
have *I*: *B n ≤ (P ∗ 2^(2 ∗ k)) ∗ n^(2 ∗ k)* **if** *n > 0* **for** *n*
proof −
define *r::nat* **where** *r = nat(floor(log 2 (real n)))*
have ∗: *int r = floor(log 2 (real n))*
unfolding *r-def* **using** ‹*0 < n*› **by** *auto*
have *I*: *2^r ≤ n ∧ n < 2^(r+1)*
using *floor-log-nat-eq-powr-iff*[*OF - ‹n > 0›, of 2 r*] ∗ **by** *auto*
then have *B n ≤ P ∗ 2^(2 ∗ k ∗ (r+1))*
using *Main*[*of n r+1*] **by** *auto*
also have ... = *(P ∗ 2^(2 ∗ k)) ∗ ((2^r)^(2∗k))*
by (*simp add*: *power-add power-mult*[*symmetric*])
also have ... ≤ *(P ∗ 2^(2 ∗ k)) ∗ n^(2 ∗ k)*
apply (*rule mult-left-mono*) **using** *I* ‹*P > 0*› **by** (*auto simp add*: *power-mono*)
finally show *?thesis* **by** *simp*
qed
show *?thesis*
proof (*intro exI*)
show ∀ *n>0. B n ≤ P ∗ 2 ^ (2 ∗ k) ∗ real n ^ (2 ∗ k)*
using *I* **by** *auto*
qed
qed

end

end

References

[GK15] Sébastien Gouëzel and Anders Karlsson, *Subadditive and multiplicative ergodic theorems*, preprint, 2015.

[Gou18] Sébastien Gouëzel, *Growth of normalizing sequences in limit theorems for conservative maps*, preprint, 2018.