Regular Variation

Michikazu Hirata

November 4, 2026

Abstract

This entry formalizes regularly varying functions. Roughly speaking, a regularly varying function is a function whose behavior at infinity is similar to that of a power-law function. Regularly varying functions are typically used in extreme value theory, a branch of statistics. We formalize the definition of regularly varying functions and standard results about them, including:

- the uniform convergence theorem,
- Karamata's representation theorem, and
- Karamata's theorem.

Contents

1 Lemmas **2**
- 1.1 General Lemmas . . . 2
- 1.2 Limit . . . 9
- 1.3 Integrals on Sets . . . 14
- 1.4 Integral of Composite Functions $g(\int_{t\in A_x} f(t)\,\mathrm{d}t)$ . . . 25
- 1.5 Convergence Theorems with Real Index . . . 36
- 1.6 Change of Variables w.r.t. Lebesgue Integration on a Set . . . 39
- 1.7 Integral of Power Functions . . . 43
- 1.8 Uniform Convergence . . . 48

2 Regularly Varying Functions **52**
- 2.1 Definition and Basic Properties . . . 52
  - 2.1.1 Regularly Varying Functions . . . 52
  - 2.1.2 Slowly Varying Functions . . . 57
- 2.2 Uniform Convergence Theorem on Intervals . . . 64
- 2.3 Weakening the Definition of Regularly Varying Functions . . . 76
- 2.4 Karamata's Representation Theorem . . . 83
  - 2.4.1 Theorems Obtained from Karamata's Representation Theorem . . . 99
  - 2.4.2 Uniform Convergence Theorem for $\rho \neq 0$ . . . 101

2.4.3 Potter Bounds 108
2.5 Karamata's Theorem 112
2.5.1 Direct Half 112
2.5.2 Converse Half 133
2.6 Properties Related to Derivative 137

1 Lemmas

theory *Lemmas-Regular-Variation*
imports
HOL−Probability.Probability
Standard-Borel-Spaces.StandardBorel
begin

1.1 General Lemmas

lemma *image-mono-on-continuous-on-Icc*:
fixes $f :: 'a::linear\text{-}continuum\text{-}topology \Rightarrow 'b :: linorder\text{-}topology$
assumes *ab*:$a \leq b$ **and** *mono*:*mono-on* $\{a..b\}$ f
and *cont*:*continuous-on* $\{a..b\}$ f
shows $f \text{ ` } \{a..b\} = \{f\ a..f\ b\}$
using *IVT'*[*OF - - ab cont*] *mono-onD*[*OF mono*] **by** *fastforce*

lemma *image-mono-on-continuous-on-Ic*:
fixes $f :: 'a::linear\text{-}continuum\text{-}topology \Rightarrow 'b :: linorder\text{-}topology$
assumes *mono*:*mono-on* $\{a..\}$ f
and *cont*:*continuous-on* $\{a..\}$ f
and *lim*:*filterlim f at-top at-top*
shows $f \text{ ` } \{a..\} = \{f\ a..\}$
proof −
have $f \text{ ` } \{a..\} = f \text{ ` } (\bigcup b{\in}\{a..\}.\ \{a..b\})$
by *fastforce*
also have ... $= (\bigcup b{\in}\{a..\}.\ f \text{ ` } \{a..b\})$
by *blast*
also have ... $= (\bigcup b{\in}\{a..\}.\ \{f\ a..f\ b\})$
proof −
have $\bigwedge b.\ a \leq b \Longrightarrow f \text{ ` } \{a..b\} = \{f\ a..\ f\ b\}$
by(*intro image-mono-on-continuous-on-Icc mono-on-subset*[*OF mono*]
continuous-on-subset[*OF cont*]) *auto*
thus *?thesis*
by *fast*
qed
also have ... $= \{f\ a..\}$
proof *safe*
fix *y*
assume *y*:$f\ a \leq y$
obtain *b* **where** *b*: $a \leq b\ y \leq f\ b$
by (*metis eventually-at-top-linorder filterlim-at-top lim nle-le*)

```
    thus y ∈ (⋃ b∈{a..}. {f a..f b})
      using y by fastforce
  qed auto
  finally show ?thesis .
qed

lemma image-mono-on-continuous-on-Ioo:
  fixes f :: 'a::{linear-continuum-topology} ⇒ 'b :: linorder-topology
  assumes ab:a ≤ b and mono:strict-mono-on {a..b} f
    and cont:continuous-on {a..b} f
  shows f ‘ {a<..<b} = {f a<..<f b}
proof −
  have f ‘ {a<..<b} = f ‘ {a..b} − {f a, f b}
    apply safe
      apply fastforce
     apply (metis ab mono atLeastAtMost-iff greaterThanLessThan-iff le-less less-le
                  strict-mono-on-less)
    apply (metis (mono-tags, opaque-lifting) ab mono
        atLeastAtMost-iff greaterThanLessThan-iff less-le
        monotone-onD order.refl)
    apply (metis atLeastAtMost-iff greaterThanLessThan-iff imageI less-le)
    done
  also have ... = {f a..f b} − {f a, f b}
    by(simp add: image-mono-on-continuous-on-Icc
        [OF ab strict-mono-on-imp-mono-on[OF mono] cont])
  also have ... = {f a<..<f b}
    by fastforce
  finally show ?thesis .
qed

lemma image-mono-on-continuous-on-Ioc:
  fixes f :: 'a::{linear-continuum-topology} ⇒ 'b :: linorder-topology
  assumes ab:a ≤ b and mono:strict-mono-on {a..b} f
    and cont:continuous-on {a..b} f
  shows f ‘ {a<..b} = {f a<..f b}
proof −
  have f ‘ {a<..b} = f ‘ {a..b} − {f a}
    apply safe
      apply fastforce
    apply (metis ab mono atLeastAtMost-iff greaterThanAtMost-iff less-le order.refl
                 strict-mono-on-eqD)
    apply (metis atLeastAtMost-iff greaterThanAtMost-iff image-eqI less-le)
    done
  also have ... = {f a..f b} − {f a}
    by(simp add: image-mono-on-continuous-on-Icc
        [OF ab strict-mono-on-imp-mono-on[OF mono] cont])
  also have ... = {f a<..f b}
    by fastforce
  finally show ?thesis .
```

qed

lemma *image-mono-on-continuous-on-greaterThan*:
fixes $f :: 'a::\{linear\text{-}continuum\text{-}topology\} \Rightarrow 'b :: linorder\text{-}topology$
assumes *mono*:*strict-mono-on* $\{a..\}$ f
and *cont*:*continuous-on* $\{a..\}$ f
and *lim*:*filterlim* f *at-top at-top*
shows $f\ `\ \{a<..\} = \{f\ a<..\}$
proof −
have $f\ `\ \{a<..\} = f\ `\ \{a..\} - \{f\ a\}$
apply *safe*
apply *fastforce*
apply (*metis mono atLeast-iff less-le nle-le strict-mono-on-eq*)
apply (*metis greaterThan-iff imageI less-le*)
done
also have $... = \{f\ a..\} - \{f\ a\}$
by(*simp add*: *image-mono-on-continuous-on-Ic*
[*OF strict-mono-on-imp-mono-on*[*OF mono*] *cont lim*])
also have $... = \{f\ a<..\}$
by *fastforce*
finally show *?thesis* **.**
qed

lemma *image-antimono-on-continuous-on-Icc*:
fixes $f :: 'a::linear\text{-}continuum\text{-}topology \Rightarrow 'b :: linorder\text{-}topology$
assumes *ab*:$a \leq b$ **and** *mono*:*antimono-on* $\{a..b\}$ f
and *cont*:*continuous-on* $\{a..b\}$ f
shows $f\ `\ \{a..b\} = \{f\ b..f\ a\}$
using *IVT2'*[*OF - - ab cont*] *monotone-onD*[*OF mono*] **by** *fastforce*

lemma *image-antimono-on-continuous-on-Ioc*:
fixes $f :: 'a::\{linear\text{-}continuum\text{-}topology\} \Rightarrow 'b :: linorder\text{-}topology$
assumes *ab*:$a \leq b$ **and** *mono*:*strict-antimono-on* $\{a..b\}$ f
and *cont*:*continuous-on* $\{a..b\}$ f
shows $f\ `\ \{a<..b\} = \{f\ b..<f\ a\}$
proof −
have $f\ `\ \{a<..b\} = f\ `\ \{a..b\} - \{f\ a\}$
apply *safe*
apply *fastforce*
apply (*metis* (*mono-tags*, *lifting*) *assms*(*1*,*2*) *atLeastAtMost-iff*
greaterThanAtMost-iff less-le monotone-onD order.refl)
apply (*metis atLeastAtMost-iff greaterThanAtMost-iff imageI less-le*)
done
also have $... = \{f\ b..f\ a\} - \{f\ a\}$
using *mono*
by(*subst image-antimono-on-continuous-on-Icc*[*OF ab - cont*])
(*auto simp*: *monotone-on-def le-less*)
also have $... = \{f\ b..<f\ a\}$
by *fastforce*

finally show *?thesis* **.**
qed

lemma *image-mult-greaterThanLessThan*[*simp*]:
fixes *c* :: *real*
assumes *0* < *c*
shows (∗) *c* ‘ {*a*<..<*b*} = {*c* ∗ *a*<..<*c* ∗ *b*}
proof −
have (∗) *c* ‘ {*a*<..<*b*} = (∗) *c* ‘ *ball* ((*b* + *a*) / *2*) ((*b* − *a*) / *2*)
by(*rule arg-cong*[**where** *f*=*image* (λ*x*. *c* ∗ *x*)])
(*simp add*: *ball-eq-greaterThanLessThan add-divide-distrib*[*symmetric*]
diff-divide-distrib[*symmetric*])
also have ... = *ball* (*c* ∗ (*b* + *a*) / *2*) (*c* ∗ (*b* − *a*) / *2*)
using *ball-scale*[*of c* (*b* + *a*) / *2* (*b* − *a*) / *2*] *assms* **by** *simp*
also have ... = {*c* ∗ *a*<..<*c* ∗ *b*}
by(*simp add*: *ball-eq-greaterThanLessThan add-divide-distrib*[*symmetric*]
distrib-left[*symmetric*] *diff-divide-distrib*[*symmetric*] *right-diff-distrib*[*symmetric*])
finally show *?thesis* **.**
qed

lemma *image-times-atLeastLessThan*[*simp*]:
fixes *c* :: *real*
assumes *0* < *c*
shows (∗) *c* ‘ {*a*<..*b*} = {*c* ∗ *a*<..*c* ∗ *b*}
proof −
have (∗) *c* ‘ {*a*<..*b*} = (∗) *c* ‘ ({*a*<..<*b*+*1*} ∩ {*a*−*1*..*b*})
by(*rule arg-cong*[**where** *f*=*image* (λ*x*. *c* ∗ *x*)]) *auto*
also have ... = (∗) *c* ‘ {*a*<..<*b*+*1*} ∩ (∗) *c* ‘ {*a*−*1*..*b*}
by(*subst image-Int*) (*use assms* **in** *auto*)
also have ... = {*c* ∗ *a*<..<*c* ∗ (*b* + *1*)} ∩ {*c* ∗ (*a* − *1*)..*c* ∗ *b*}
using *assms* **by** *simp*
also have ... = {*c* ∗ *a*<..*c* ∗ *b*}
unfolding *set-eq-iff greaterThanLessThan-iff Int-iff atLeastAtMost-iff greaterThanAtMost-iff*
by(*intro allI iffI*) (*use assms* **in** *argo*)+
finally show *?thesis* **.**
qed

lemma *image-power-Icc-pos*:
assumes *0* ≤ *ϱ*
and *0* < *a* **and** *a* ≤ *b*
shows (λ*x*::*real*. *x powr ϱ*) ‘ {*a*..*b*} = {*a powr ϱ*..*b powr ϱ*}
using *assms*
by(*intro image-mono-on-continuous-on-Icc mono-onI powr-mono2 continuous-on-powr*)
auto

lemma *image-inverse-power-Icc-pos*:
assumes *0* ≤ *ϱ*
and *0* < *a* **and** *a* ≤ *b*

shows $(\lambda x{::}real.\ 1\ /\ x\ powr\ \varrho)\ `\ \{a..b\} = \{1\ /\ b\ powr\ \varrho..1\ /\ a\ powr\ \varrho\}$
using *assms*
by(*intro image-antimono-on-continuous-on-Icc monotone-onI divide-left-mono powr-mono2*
continuous-on-divide continuous-on-powr)
auto

lemma *image-power-Icc-neg*:
assumes $\varrho \leq 0$
and $0 < a$ **and** $a \leq b$
shows $(\lambda x{::}real.\ x\ powr\ \varrho)\ `\ \{a..b\} = \{b\ powr\ \varrho..a\ powr\ \varrho\}$
using *assms*
by(*intro image-antimono-on-continuous-on-Icc monotone-onI powr-mono2' continuous-on-powr*)
auto

lemma *image-inverse-power-Icc-neg*:
assumes $\varrho \leq 0$
and $0 < a$ **and** $a \leq b$
shows $(\lambda x{::}real.\ 1\ /\ x\ powr\ \varrho)\ `\ \{a..b\} = \{1\ /\ a\ powr\ \varrho..1\ /\ b\ powr\ \varrho\}$
using *assms*
by(*intro image-mono-on-continuous-on-Icc continuous-on-divide*
divide-left-mono mono-onI powr-mono2' continuous-on-powr)
auto

lemma *image-times-atLeastLessThan'*[*simp*]:
fixes $c :: real$
assumes $0 < c$
shows $(\lambda x.\ x * c)\ `\ \{a<..b\} = \{c * a<..c * b\}$
using *assms* **by**(*simp add*: *mult.commute*[*of* - *c*])

lemma *bounded-times-comp*:
fixes $f\ g{::}'a \Rightarrow 'b{::}real\text{-}normed\text{-}field$
assumes *bounded* $(f\ `\ S)$
and *bounded* $(g\ `\ S)$
shows *bounded* $((\lambda x.\ f\ x * g\ x)\ `\ S)$
proof −
obtain $a\ b$ **where** $\bigwedge x.\ x \in S \Longrightarrow norm\ (f\ x) \leq a\ \bigwedge x.\ x \in S \Longrightarrow norm\ (g\ x) \leq b$
$\bigwedge x.\ x \in S \Longrightarrow 0 \leq a$
using *assms* **by** (*metis bounded-pos less-eq-real-def rev-image-eqI*)
thus *?thesis*
by(*auto simp*: *bounded-iff norm-mult intro*!: *exI*[**where** $x{=}a * b$] *mult-mono*)
qed

lemma *antimono-on-iff-uminus-mono-on*:
fixes $f :: - \Rightarrow - :: ordered\text{-}ab\text{-}group\text{-}add$
shows *antimono-on* $A\ f \longleftrightarrow$ *mono-on* $A\ (\lambda x.\ -f\ x)$
by(*auto simp*: *monotone-on-def*)

lemma *filterlim-Pair-component*:
LIM x F. f x :> G $\Longrightarrow$ *LIM x H. g x :> I* $\Longrightarrow$ *LIM* $(x,\ y)$ $F \times_F H.\ (f\ x,\ g\ y)$:> $G \times_F I$
by (*metis filterlim-def filtermap-prod-filter map-prod-def prod-filter-mono*)

lemma *eventually-eventually-prod-filter-principal1*:
eventually P $(F \times_F$ *principal A*$)$ $\longleftrightarrow$ *eventually* $(\lambda x.\ \forall\, y{\in}A.\ P\ (x,\ y))$ *F*
proof
assume *h*:$\forall_F$ *x in F.* $\forall\, y{\in}A.\ P\ (x,\ y)$
show *eventually P* $(F \times_F$ *principal A*$)$
unfolding *eventually-prod-filter*
by(*intro exI*[**where** $x{=}\lambda x.\ \forall\, y{\in}A.\ P\ (x,\ y)$] *exI*[**where** $x{=}\lambda y.\ y \in A$])
(*use h* **in** *auto simp*: *eventually-principal*)
qed(*auto dest*: *eventually-eventually-prod-filter1 simp*: *eventually-principal*)

lemma *eventually-eventually-prod-filter-principal2*:
eventually P (*principal A* $\times_F$ *G*) $\longleftrightarrow$ *eventually* $(\lambda y.\ \forall\, x{\in}A.\ P\ (x,\ y))$ *G*
proof
assume *h*:$\forall_F$ *y in G.* $\forall\, x{\in}A.\ P\ (x,\ y)$
show *eventually P* (*principal A* $\times_F$ *G*)
unfolding *eventually-prod-filter*
by(*intro exI*[**where** $x{=}\lambda y.\ \forall\, x{\in}A.\ P\ (x,\ y)$] *exI*[**where** $x{=}\lambda x.\ x \in A$])
(*use h* **in** *auto simp*: *eventually-principal*)
qed(*auto dest*: *eventually-eventually-prod-filter2 simp*: *eventually-principal*)

lemma *prod-filter-distrib-right*:
$F \times_F (G \sqcup H) = F \times_F G \sqcup F \times_F H$
proof(*rule filter-eqI*)
fix *P*
show *eventually P* $(F \times_F (G \sqcup H)) =$
eventually P $(F \times_F G \sqcup F \times_F H)$
proof
assume *eventually P* $(F \times_F (G \sqcup H))$
then obtain *Pf Pg* **where** *eventually Pf F eventually Pg G eventually Pg H*
$\forall\, x\ y.\ Pf\ x \longrightarrow Pg\ y \longrightarrow P\ (x,\ y)$
by(*auto simp*: *eventually-prod-filter eventually-sup*)
thus *eventually P* $(F \times_F G \sqcup F \times_F H)$
by(*auto simp*: *eventually-sup eventually-prod-filter*)
next
assume *eventually P* $(F \times_F G \sqcup F \times_F H)$
then obtain *Pf Pg Pf′ Ph* **where** *h*:
eventually Pf F eventually Pg G $\forall\, x\ y.\ Pf\ x \longrightarrow Pg\ y \longrightarrow P\ (x,\ y)$
eventually Pf′ F eventually Ph H $\forall\, x\ y.\ Pf'\ x \longrightarrow Ph\ y \longrightarrow P\ (x,\ y)$
by(*auto simp*: *eventually-sup eventually-prod-filter*)
thus *eventually P* $(F \times_F (G \sqcup H))$
unfolding *eventually-prod-filter*
by(*intro exI*[**where** $x{=}\lambda x.\ Pf\ x \wedge Pf'\ x$] *exI*[**where** $x{=}\lambda y.\ Pg\ y \vee Ph\ y$])

```
      (auto simp: eventually-sup eventually-conj eventually-mono)
  qed
qed

lemma suminf-const-infinity:
  fixes a :: ennreal
  assumes a:0 < a
  shows (∑ n. a) = ⊤
proof −
  have ∗:(⋃ n. {real-of-int (from-nat-into UNIV n)..<real-of-int (from-nat-into
UNIV n + 1)})
        = UNIV
  proof safe
    fix x :: real
    have ∃ n. real-of-int (from-nat-into UNIV n) ≤ x
          ∧ x < real-of-int (from-nat-into UNIV n + 1)
     by (metis UNIV-I countableI-type floor-correct from-nat-into-to-nat-on)
    thm from-nat-into-to-nat-on
   thus x ∈ (⋃ n. {real-of-int (from-nat-into UNIV n)..<real-of-int (from-nat-into
UNIV n + 1)})
      by auto
  qed auto
  have ∗∗:disjoint-family
            (λn. {real-of-int (from-nat-into UNIV n)..<real-of-int (from-nat-into
UNIV n + 1)})
  proof −
    have ⋀n m. n ≠ m ⟹ from-nat-into UNIV n ≠ (from-nat-into UNIV m ::
int)
      by simp
    thus ?thesis
      unfolding disjoint-family-on-def
      by (metis Int-emptyI atLeastLessThan-iff floor-exists1)
  qed
  have (∑ n. a) = (∑ n.
         (∫+ x∈{real-of-int (from-nat-into UNIV n)..<real-of-int (from-nat-into
UNIV n + 1)}.
              a ∂lborel))
    by(auto intro!: suminf-cong simp: nn-integral-cmult-indicator)
  also have ... = (∫+ x∈⋃ n.
          {real-of-int (from-nat-into UNIV n)..<real-of-int (from-nat-into UNIV n
+ 1)}.
           a ∂lborel)
    by(rule nn-integral-disjoint-family[symmetric])
      (use ∗∗ in auto)
  also have ... = (∫+ x∈UNIV. a ∂lborel)
    unfolding ∗ by fastforce
  also have (∫+ x∈UNIV. a ∂lborel) = a ∗ emeasure lborel UNIV
    by(simp add: nn-integral-cmult-indicator)
  also have ... = ⊤
```

using *a* **by** *force*
finally show *?thesis*
by *simp*
qed

1.2 Limit

lemma *tendsto-sandwich-mult-const*:
fixes *c* :: *real*
assumes *ev*: $\bigwedge e$::*real*. $0 < e \Longrightarrow$ *eventually* (λn. $f\ e \leq g\ n\ /\ c \wedge g\ n\ /\ c \leq h\ e$) *net*
and *limf*:($f \longrightarrow 1$) (*at-right 0*) **and** *limh*:($h \longrightarrow 1$) (*at-right 0*)
shows ($g \longrightarrow c$) *net*
proof −
have ((λn. $g\ n\ /\ c$) $\longrightarrow 1$) *net*
unfolding *tendsto-iff dist-real-def*
proof *safe*
fix *e* :: *real*
assume *e*:$0 < e$
obtain *a* **where** *a-pos*:$0 < a$ **and** *a*:$\bigwedge a'$. $0 < a' \Longrightarrow a' < a \Longrightarrow |f\ a' - 1| < e$
using *limf e* **unfolding** *tendsto-iff eventually-at-right*[*OF e*]
by(*auto simp*: *dist-real-def*)
obtain *b* **where** *b-pos*:$0 < b$ **and** *b*:$\bigwedge b'$. $0 < b' \Longrightarrow b' < b \Longrightarrow |h\ b' - 1| < e$
using *limh e* **unfolding** *tendsto-iff eventually-at-right*[*OF e*]
by(*auto simp*: *dist-real-def*)
show $\forall_F\ x$ *in net*. $|g\ x\ /\ c - 1| < e$
proof(*safe intro*!: *eventually-mono*[*OF ev*[*of min a b / 2*]])
fix *x*
assume *le1*:$f\ (min\ a\ b\ /\ 2) \leq g\ x\ /\ c$
and *le2*:$g\ x\ /\ c \leq h\ (min\ a\ b\ /\ 2)$
have $1 - e < f\ (min\ a\ b\ /\ 2)$
using *a*[*of min a b / 2*] *a-pos b-pos* **by** *argo*
also have ... $\leq g\ x\ /\ c$
by *fact*
finally have $*$:$1 - e < g\ x\ /\ c$ **.**
have $g\ x\ /\ c \leq h\ (min\ a\ b\ /\ 2)$
by *fact*
also have ... $<\ e + 1$
using *b*[*of min a b / 2*] *a-pos b-pos* **by** *argo*
finally show $|g\ x\ /\ c - 1| < e$
using $*$ **by** *auto*
qed(*use a-pos b-pos* **in** *argo*)
qed
from *asymp-equivI'*[*OF this*]
show *?thesis*
by(*auto simp*: *asymp-equivD-const*)
qed

lemma *asymp-equiv-Limsup-cong-ereal*:

```
  assumes equiv:f ∼[F] g
    and nonneg1:∀ F x in F. 0 ≤ f x
    and nonneg2:∀ F x in F. 0 ≤ g x
  shows Limsup F (λx. ereal (f x)) = Limsup F (λx. ereal (g x))
proof(cases F = ⊥)
  assume F:F ≠ ⊥
  have ∗:Limsup F (λx. ereal (f x)) ≤ Limsup F (λx. ereal (g x))
    if equiv:f ∼[F] g
    and nonneg1:∀ F x in F. 0 ≤ f x
    and nonneg2:∀ F x in F. 0 ≤ g x for f g
  proof −
    have Limsup1:0 ≤ Limsup F (λx. ereal (f x))
      by(auto intro!: le-Limsup nonneg1 F)
    have Limsup2:0 ≤ Limsup F (λx. ereal (g x))
      by(auto intro!: le-Limsup nonneg2 F)
    have le:Limsup F (λx. ereal (f x))
       ≤ ereal c ∗ Limsup F (λx. ereal (g x)) if c:1 < c for c :: real
    proof −
      have Limsup F (λx. ereal (f x)) ≤ Limsup F (λx. ereal c ∗ ereal (g x))
        by(intro Limsup-mono eventually-mono[OF
              eventually-conj[OF eventually-conj[OF nonneg1 nonneg2]
                asymp-equiv-imp-eventually-le[OF equiv c]]])
          auto
      also have ... = ereal c ∗ Limsup F (λx. ereal (g x))
        by(subst Limsup-ereal-mult-left) (use F c in auto)
      finally show ?thesis .
    qed
    have ∗:((λc. ereal c ∗ Limsup F (λx. ereal (g x)))
                  ⟶ Limsup F (λx. ereal (g x))) (at-right 1)
    proof −
      have ((λc. ereal c) ⟶ ereal 1) (at-right 1)
        by simp
      from tendsto-mult-ereal[OF this tendsto-const[of Limsup F (λx. ereal (g x))]]
      show ?thesis
        by simp
    qed
    show ?thesis
      using le
      by(intro tendsto-lowerbound[OF ∗] eventually-at-rightI[where b=2]) auto
  qed
  show ?thesis
    by(intro antisym ∗ assms asymp-equiv-symI)
qed simp

lemma asymp-equiv-Liminf-cong-ereal:
  assumes equiv:f ∼[F] g
    and nonneg1:∀ F x in F. 0 ≤ f x
    and nonneg2:∀ F x in F. 0 ≤ g x
  shows Liminf F (λx. ereal (f x)) = Liminf F (λx. ereal (g x))
```

```
proof(cases F = ⊥)
  assume F:F ≠ ⊥
  have *:Liminf F (λx. ereal (f x)) ≤ Liminf F (λx. ereal (g x))
    if equiv:f ∼[F] g
    and nonneg1:∀⇩F x in F. 0 ≤ f x
    and nonneg2:∀⇩F x in F. 0 ≤ g x for f g
  proof −
    have le:Liminf F (λx. ereal (f x))
       ≤ ereal c * Liminf F (λx. ereal (g x)) if c:1 < c for c :: real
    proof −
      have Liminf F (λx. ereal (f x)) ≤ Liminf F (λx. ereal c * ereal (g x))
        by(intro Liminf-mono eventually-mono[OF
              eventually-conj[OF eventually-conj[OF nonneg1 nonneg2]
                asymp-equiv-imp-eventually-le[OF equiv c]]])
          auto
      also have ... = ereal c * Liminf F (λx. ereal (g x))
        by(subst Liminf-ereal-mult-left) (use F c in auto)
      finally show ?thesis .
    qed
    have *:((λc. ereal c * Liminf F (λx. ereal (g x)))
            ⟶ Liminf F (λx. ereal (g x))) (at-right 1)
    proof −
      have ((λc. ereal c) ⟶ ereal 1) (at-right 1)
        by simp
      from tendsto-mult-ereal[OF this tendsto-const[of Liminf F (λx. ereal (g x))]]
      show ?thesis
        by simp
    qed
    show ?thesis
      using le
      by(intro tendsto-lowerbound[OF *] eventually-at-rightI[where b=2]) auto
  qed
  show ?thesis
    by(intro antisym * assms asymp-equiv-symI)
qed simp

lemma tendsto-at-topI-sequentially-real':
  fixes f :: real ⇒ real
  assumes mono: mono-on {K..} f
    and limseq: (λn. f (real n)) ⟶ y
  shows (f ⟶ y) at-top
proof −
  have mono2:mono (λx. if x < K then f K else f x)
    using mono-onD[OF mono] by(auto intro!: monoI)
  obtain N where K < real N
    using reals-Archimedean2 by blast
  hence (λn. (λx. if x < K then f K else f x) (real n)) ⟶ y
    by(auto intro!: tendsto-cong[THEN iffD1,OF - limseq]
        eventually-at-top-linorderI[where c=N])
```

```
  hence lim:((λx. if x < K then f K else f x) ⟶ y) at-top
    by(auto intro!: tendsto-at-topI-sequentially-real mono2)
  show ?thesis
    by(auto intro!: tendsto-cong[THEN iffD1,OF - lim]
        eventually-at-top-linorderI[where c=K])
qed

lemma mono-convergent-real-le:
  fixes f :: real ⇒ real
  assumes mono:mono-on {K..} f
    and le:⋀r. K ≤ r ⟹ f r ≤ A
  shows ∃ l. (f ⟶ l) at-top ∧ (∀ r≥K. f r ≤ l)
proof −
  have inc: incseq (λn. f (K + real n))
    by(auto intro!: monoI mono-onD[OF mono])
  obtain L where L:(λn. f (K + real n)) ⟶ L ⋀i. f (K + real i) ≤ L
    using incseq-convergent[OF inc,of A] le by auto
  hence ((λr. f (K + r)) ⟶ L) at-top
    using mono-onD[OF mono]
    by(auto intro!: tendsto-at-topI-sequentially-real'[of 0 λr. f (K + r)]
        mono-onI)
  moreover have LIM r at-top. r − K :> at-top
    using filterlim-tendsto-add-at-top-iff[where f=λx. − K,THEN iffD2]
        filterlim-ident by auto
  ultimately have 1:(f ⟶ L) at-top
    using filterlim-compose[of λr. f (K + r) nhds L at-top λr. −K + r at-top]
    by simp
  moreover have f r ≤ L if r:K ≤ r for r
  proof(rule ccontr)
    assume ¬ f r ≤ L
    hence ∀ F x in at-top. f x < f r
      using order-topology-class.order-tendstoD[OF 1] by simp
    moreover have ∀ F x in at-top. f r ≤ f x
      using mono-onD[OF mono,of r] r
      by(auto intro!: eventually-at-top-linorderI[where c=r])
    ultimately show False
      unfolding eventually-at-top-linorder
      by (metis nle-le not-less)
  qed
  ultimately show ?thesis
    by blast
qed

lemma mono-convergent-real-less:
  fixes f :: real ⇒ real
  assumes mono:mono-on {K<..} f
    and less:⋀r. K < r ⟹ f r ≤ A
  shows ∃ l. (f ⟶ l) at-top ∧ (∀ r>K. f r ≤ l)
proof −
```

```
  have ∃ l. (f ⟶ l) at-top ∧ (∀ r≥K + 1. f r ≤ l)
    by(auto intro!: mono-convergent-real-le less mono-onI mono-onD[OF mono])
  then obtain L where L:(f ⟶ L) at-top
    by blast
  have 0 < e ⟹ ∃ l. (f ⟶ l) at-top ∧ (∀ r≥K+e. f r ≤ l) for e :: real
    by(auto intro!:  mono-convergent-real-le less mono-onI mono-onD[OF mono])
  then obtain l where l':⋀e. 0 < e ⟹ (f ⟶ l e) at-top ∧ (∀ r≥K+e. f r ≤
l e)
    by metis
  hence ⋀e. 0 < e ⟹ l e = L
    by(auto intro!: tendsto-unique[OF - - L])
  hence l:⋀e r. 0 < e ⟹ K + e ≤ r ⟹ f r ≤ L
    using l' by fast
  hence ⋀r. K < r ⟹ f r ≤ L
    by (metis not-le field-le-epsilon nle-le)
  with L show ?thesis by blast
qed

lemma at-top-real-iff-all-sequentially:
  fixes f :: real ⇒ 'b :: topological-space
  shows (f ⟶ l) at-top
      ⟷ (∀ a. filterlim a at-top sequentially ⟶ incseq a
            ⟶ (λn. (f (a n))) ⟶ l)
proof
  assume h:∀ a. filterlim a at-top sequentially ⟶ (incseq a) ⟶ (λn. f (a n))
⟶ l
  show (f ⟶ l) at-top
  proof (rule ccontr)
    assume c:¬ (f ⟶ l) at-top
    then obtain S where S:open S l ∈ S⋀R. ∃ r ≥ R. f r ∉ S
      by(fastforce simp: tendsto-def eventually-at-top-linorder)
    define a where a ≡ rec-nat (SOME r. r ≥ 0 ∧ f r ∉ S)
                       (λn an. SOME r. r > max an (real (Suc n)) ∧ f r ∉ S)
    have a-inc:incseq a
    proof(rule incseq-SucI)
      fix n
      have aSn:a (Suc n) = (SOME r.  r > max (a n) (real (Suc n)) ∧ f r ∉ S)
        by(simp add: a-def)
      obtain r where r:r ≥ max (a n + 1) (Suc (Suc n)) f r ∉ S
        using S by blast
      show a n ≤ a (Suc n)
        unfolding aSn
        by(rule someI2[where a=r]) (use r in auto)
    qed
    have a-ge-n:a n ≥ real n ∧ f (a n) ∉ S for n
    proof(cases n)
      case 0
      obtain r where r:r ≥ 0 f r ∉ S
        using S by blast
```

```
    show ?thesis
      unfolding a-def 0 nat.rec
      by(rule someI2[where a=r]) (use r in auto)
  next
    case n':(Suc n')
    then have aSn':a (Suc n') = (SOME r.  r > max (a n') (real (Suc n')) ∧ f
r ∉ S)
      by(simp add: a-def)
    obtain r where r:r ≥ max (a n' + 1) (Suc (Suc n')) f r ∉ S
      using S by blast
    show ?thesis
      unfolding n' aSn'
      by(rule someI2[where a=r]) (use r in auto)
  qed
  hence a-ge-n:a n ≥ real n and fan-nin-S:f (a n) ∉ S for n
    by auto
  hence filterlim a at-top sequentially
    unfolding filterlim-at-top
    by (metis order.trans eventually-sequentially of-nat-mono real-arch-simple)
  hence (λn. f (a n)) ⟶⟶ l
    using h a-inc by simp
  thus False
    using S fan-nin-S
    by (meson lim-explicit order.refl)
 qed
qed(use filterlim-compose in blast)
lemma at-top-real-iff-all-sequentially':
 fixes f :: real ⇒ 'b :: topological-space
 shows (f ⟶⟶ l) at-top
  ⟷ (∀ a. filterlim a at-top sequentially ⟶ (∀ n. a n ≥ K)
      ⟶ incseq a ⟶ (λn. (f (a n))) ⟶⟶ l)
 unfolding at-top-real-iff-all-sequentially
proof safe
 fix a :: nat ⇒ real
 assume h:∀ a. filterlim a at-top sequentially ⟶ (∀ n. K ≤ a n)
          ⟶ incseq a⟶ (λn. f (a n)) ⟶⟶ l
   and  a:filterlim a at-top sequentially incseq a
 then obtain N where N:⋀n. n ≥ N ⟹ a n ≥ K
   by (metis eventually-at-top-linorder filterlim-at-top)
 have a':filterlim (λn. a (n + N)) at-top sequentially incseq (λn. a (n + N))
   using a  filterlim-compose[OF - filterlim-add-const-nat-at-top]
   by(auto simp: incseq-Suc-iff)
 show (λn. f (a n)) ⟶⟶ l
   by(rule LIMSEQ-offset[where k=N]) (use h N a' in fastforce)
qed simp
```

1.3 Integrals on Sets

lemma *set-borel-measurable-iff-restrict-space*:

fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
assumes [*measurable*]:$A \in$ *sets* M
shows *set-borel-measurable* M A $f \longleftrightarrow f \in$ *borel-measurable* (*restrict-space* M A)
by (*simp add*: *borel-measurable-restrict-space-iff set-borel-measurable-def*)

lemma *set-borel-measurable-add*:
fixes $f :: - \Rightarrow -$ :: {*second-countable-topology*, *topological-monoid-add*, *real-vector*}
assumes *set-borel-measurable* M A f
and *set-borel-measurable* M A g
shows *set-borel-measurable* M A $(\lambda x.\ f\ x + g\ x)$
using *assms* **by**(*simp add*: *set-borel-measurable-def scaleR-add-right*)

lemma *set-borel-measurable-scaleR*:
fixes $g :: - \Rightarrow -$ :: {*second-countable-topology*, *real-normed-vector*}
assumes *set-borel-measurable* M A f
and *set-borel-measurable* M A g
shows *set-borel-measurable* M A $(\lambda x.\ f\ x *_R g\ x)$
proof −
have $*$:$(\lambda x.$ *indicat-real* A $x *_R f\ x *_R g\ x)$
$= (\lambda x.$ (*indicat-real* A $x * f\ x$) $*_R$ (*indicat-real* A $x *_R g\ x$))
by(*auto simp*: *indicator-def*)
show *?thesis*
unfolding *set-borel-measurable-def* $*$
by(*rule borel-measurable-scaleR*) (*use assms* **in** *auto simp*: *set-borel-measurable-def*)
qed

lemma *set-borel-measurable-times*:
fixes $f :: 'a \Rightarrow 'b$::{*second-countable-topology*, *real-normed-algebra*}
assumes *set-borel-measurable* M A f
and *set-borel-measurable* M A g
shows *set-borel-measurable* M A $(\lambda x.\ f\ x * g\ x)$
proof −
have $*$:$(\lambda x.$ *indicat-real* A $x *_R (f\ x * g\ x))$
$= (\lambda x.$ (*indicat-real* A $x *_R f\ x$) $*$ (*indicat-real* A $x *_R g\ x$))
by(*auto simp*: *indicator-def*)
show *?thesis*
unfolding *set-borel-measurable-def* $*$
by(*rule borel-measurable-times*) (*use assms* **in** *auto simp*: *set-borel-measurable-def*)
qed

lemma *set-borel-measurable-restrict-space-iff*:
fixes $f :: 'a \Rightarrow 'b$::*real-normed-vector*
assumes [*measurable*]:$A \in$ *sets* M
shows *set-borel-measurable* (*restrict-space* M A) A $f \longleftrightarrow$ *set-borel-measurable* M A f
proof
assume *set-borel-measurable* (*restrict-space* M A) A f
moreover have $(\lambda x.$ *indicat-real* A $x *_R f\ x)$
$= (\lambda x.$ *if* $x \in A$ *then indicat-real* A $x *_R f\ x$ *else* 0)

by *auto*
ultimately show *set-borel-measurable M A f*
by(*auto simp*: *set-borel-measurable-def measurable-restrict-space-iff*[*symmetric*])
qed(*auto simp*: *set-borel-measurable-def measurable-restrict-space1*)

lemma *set-borel-measurable-comp'*:
fixes *f* :: *'a* ⇒ *'b*::*real-normed-vector*
assumes [*measurable*]: *A* ∈ *sets M*
and *set-borel-measurable N* (*g* ' *A*) *f*
and [*measurable*]: *g* ∈ *measurable M N*
shows *set-borel-measurable M A* (λ*x*. *f* (*g x*))
proof −
have *x* ∈ *space* (*restrict-space M A*) ⟹
f (*g x*) = ((λ*x*. *indicat-real* (*g* ' *A*) *x* ∗$_R$ (*f x*)) ∘ *g*) *x* **for** *x*
by *simp*
moreover have ... ∈ *borel-measurable* (*restrict-space M A*)
by(*rule measurable-comp*[**where** *N*=*N*])
(*use assms* **in** *auto simp*: *measurable-restrict-space1 set-borel-measurable-def*)
ultimately show *?thesis*
by(*simp del*: *comp-apply space-restrict-space2*
add: *set-borel-measurable-iff-restrict-space*
cong: *measurable-cong*)
qed

lemma *set-borel-measurable-comp*:
fixes *f* :: *-* ⇒ *-* ::*real-normed-vector* **and** *g* :: *-* ⇒ *-*::*real-vector*
assumes [*measurable*]: *A* ∈ *sets M*
and *set-borel-measurable borel* (*g* ' *A*) *f*
and *set-borel-measurable M A g*
shows *set-borel-measurable M A* (λ*x*. *f* (*g x*))
proof −
have *x* ∈ *space* (*restrict-space M A*) ⟹
f (*g x*) = ((λ*x*. *indicat-real* (*g* ' *A*) *x* ∗$_R$ (*f x*))
∘ (λ*x*. *indicat-real A x* ∗$_R$ (*g x*))) *x* **for** *x*
by *simp*
moreover have ... ∈ *borel-measurable* (*restrict-space M A*)
by(*rule measurable-comp*[**where** *N*=*restrict-space borel* (*g* ' *A*)])
(*use assms* **in** *auto simp*: *set-borel-measurable-def*
measurable-restrict-space2-iff measurable-restrict-space1)
ultimately show *?thesis*
by(*simp del*: *comp-apply space-restrict-space2*
add: *set-borel-measurable-iff-restrict-space*
cong: *measurable-cong*)
qed

lemma *set-integrable-set-borel-measurable*:
fixes *f* :: *'a* ⇒ *'b*::{*banach*, *second-countable-topology*}
assumes *set-integrable M S f*
shows *set-borel-measurable M S f*

using *assms integrable-iff-bounded set-borel-measurable-def set-integrable-def*
by *blast*

lemma *set-integrable-uminus*:
fixes $f :: 'a \Rightarrow 'b$::{*banach*, *second-countable-topology*}
shows *set-integrable M S* $(\lambda x.\ -f\ x) \longleftrightarrow$ *set-integrable M S f*
by(*auto simp*: *set-integrable-def*)

lemma *set-integrable-const*:
fixes $a :: 'b$::{*banach*, *second-countable-topology*}
assumes *emeasure M S* $< \top$ $S \in$ *sets M*
shows *set-integrable M S* $(\lambda x.\ a)$
using *assms* **by**(*auto simp*: *set-integrable-def*)

lemma *set-integrable-zero*[*simp*]: *set-integrable M S* $(\lambda x.\ 0)$
by(*auto simp*: *set-integrable-def*)

lemma *set-integral-cong*:
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x = g\ x$
shows $(\int x{\in}A.\ f\ x\ \partial M) = (\int x{\in}A.\ g\ x\ \partial M)$
unfolding *set-lebesgue-integral-def*
by(*auto intro*!: *Bochner-Integration.integral-cong assms simp*: *indicator-def*)

lemma *set-integrable-bounded*:
fixes $f :: 'a \Rightarrow 'b$::{*banach*, *second-countable-topology*}
assumes *set-borel-measurable M S f*
and *S*:*emeasure M S* $< \top$ $S \in$ *sets M*
and *le*:$\bigwedge x.\ x \in S \Longrightarrow$ *norm* $(f\ x) \leq K$
shows *set-integrable M S f*
unfolding *set-integrable-def integrable-iff-bounded*
proof
have $(\int^+ x.$ *ennreal* $($*norm* $($*indicat-real S x* $*_R$ $f\ x))\ \partial M)$
$\leq (\int^+ x.$ *ennreal* $($*norm* $($*indicat-real S x* $*_R$ $K))\ \partial M)$
using *assms* **by**(*fastforce intro*!: *nn-integral-mono simp*: *indicator-def*)
also have $\ldots = (\int^+ x{\in}S.$ *ennreal K* $\partial M)$
using *order.trans*[*OF norm-ge-zero le*]
by(*auto intro*!: *nn-integral-cong ennreal-inj*[*THEN iffD2*] *simp*: *indicator-def*)
also have $\ldots < \infty$
using *S ennreal-less-top ennreal-mult-less-top*
by(*simp add*: *nn-integral-cmult-indicator*)
finally show $(\int^+ x.$ *ennreal* $($*norm* $($*indicat-real S x* $*_R$ $f\ x))\ \partial M) < \infty$.
qed(*use assms* **in** *simp add*: *set-borel-measurable-def*)

lemma *set-integral-subset-AE*:
fixes $f :: - \Rightarrow$ *real*
assumes *integ*:*set-integrable M B f*
and *AB*:$A \in$ *sets M* $A \subseteq B$
and *nonneg*:*AE* $x{\in}$ $B - A$ *in M*. $0 \leq f\ x$
shows $(\int x{\in}A.\ f\ x\ \partial M) \leq (\int x{\in}B.\ f\ x\ \partial M)$

using *set-integrable-subset*[*OF integ AB*] *assms*
unfolding *set-lebesgue-integral-def set-integrable-def*
by(*intro Bochner-Integration.integral-mono-AE eventually-mono*[*OF nonneg*])
(*auto simp*: *indicator-def*)

lemma *set-integral-subset*:
fixes *f* :: - ⇒ *real*
assumes *integ*:*set-integrable M B f*
and *AB*:*A* ∈ *sets M A* ⊆ *B*
and *nonneg*:⋀*x*. *x*∈ *B* − *A* ⟹ *0* ≤ *f x*
shows (∫ *x*∈*A*. *f x* ∂*M*) ≤ (∫ *x*∈*B*. *f x* ∂*M*)
by(*auto intro*!: *set-integral-subset-AE assms*)

lemma *set-integral-empty*[*simp*]:(∫ *x*∈{}. *f x* ∂*M*) = *0*
by(*simp add*: *set-lebesgue-integral-def*)

lemma *set-integral-null-sets*:
assumes *N* ∈ *null-sets M*
shows (∫ *x*∈*N*. *f x* ∂*M*) = *0*
by(*auto simp*: *set-lebesgue-integral-def indicator-def*
intro!: *integral-eq-zero-AE AE-I'*[*OF assms*])

corollary *LBINT-singleton*[*simp*]:(∫ *x*∈{*a*}. *f x* ∂*lborel*) = *0*
by(*auto intro*!: *set-integral-null-sets*)

lemma *not-set-integrable-integral-eq*:
assumes ¬ *set-integrable M A f*
shows (∫ *x*∈*A*. *f x* ∂*M*) = *0*
using *assms*
by(*simp add*: *set-integrable-def set-lebesgue-integral-def not-integrable-integral-eq*)

The following is more general than *set-integrable lborel* {*?x*} *?f*.

lemma *set-borel-integrable-singleton'*[*simp*]: *set-integrable lborel* {*a*} *f*
unfolding *set-integrable-def*
by(*rule integrable-cong-AE*[**where** *f*=λ*x*. *0*,*THEN iffD1*
,*OF - measurable-cong*[**where** *f*=λ*x*. *if x* = *a then f a else 0*,*THEN iffD1*]])
(*auto intro*!: *AE-I'*[**where** *N*={*a*}])

lemma *set-integrable-Ioc-eq-Icc*:
fixes *a* :: - :: *linorder-topology*
fixes *f* :: - ⇒ - :: {*banach*, *second-countable-topology*}
shows *set-integrable lborel* {*a*<..*b*} *f* ⟷ *set-integrable lborel* {*a*..*b*} *f*
proof
assume *set-integrable lborel* {*a*<..*b*} *f*
from *set-integrable-Un*[*OF set-borel-integrable-singleton'*[*of a f*] *this*]
have ∗:*set-integrable lborel* ({*a*} ∪ {*a*<..*b*}) *f*
by *simp*
show *set-integrable lborel* {*a*..*b*} *f*
by(*auto intro*!: *set-integrable-subset*[*OF* ∗])

next
assume ∗:*set-integrable lborel* {*a*..*b*} *f*
show *set-integrable lborel* {*a*<..*b*} *f*
by(*auto intro*!: *set-integrable-subset*[*OF* ∗])
qed

lemma *LBINT-Ioc-eq-Icc*:
fixes *f* :: - :: *linorder-topology* ⇒ - :: {*banach*, *second-countable-topology*}
shows (∫ *x*∈{*a*<..*b*}. *f x* ∂*lborel*) = (∫ *x*∈{*a*..*b*}. *f x* ∂*lborel*)
proof(*cases set-integrable lborel* {*a*<..*b*} *f* ∧ *a* ≤ *b*)
assume [*simp*]:*set-integrable lborel* {*a*<..*b*} *f* ∧ *a* ≤ *b*
have (∫ *x*∈{*a*<..*b*}. *f x* ∂*lborel*) = (∫ *x*∈{*a*<..*b*}. *f x* ∂*lborel*) + (∫ *x*∈{*a*}. *f x* ∂*lborel*)
by *simp*
also have ... = (∫ *x*∈{*a*<..*b*} ∪ {*a*}. *f x* ∂*lborel*)
by(*subst set-integral-Un*) *auto*
also have ... = (∫ *x*∈{*a*..*b*}. *f x* ∂*lborel*)
by(*auto intro*!: *arg-cong*[**where** *f*=λ*B*. (∫ *x*∈*B*. *f x* ∂*lborel*)])
finally show *?thesis* **.**
qed (*auto simp*: *set-integrable-Ioc-eq-Icc not-set-integrable-integral-eq*)

lemma *set-integrable-atLeast-eq*:
fixes *a* :: - :: *linorder-topology*
fixes *f* :: - ⇒ - :: {*banach*, *second-countable-topology*}
shows *set-integrable lborel* {*a*<..} *f* ⟷ *set-integrable lborel* {*a*..} *f*
proof
assume *set-integrable lborel* {*a*<..} *f*
from *set-integrable-Un*[*OF set-borel-integrable-singleton*′[*of a f*] *this*]
have ∗:*set-integrable lborel* ({*a*} ∪ {*a*<..}) *f*
by *simp*
show *set-integrable lborel* {*a*..} *f*
by(*auto intro*!: *set-integrable-subset*[*OF* ∗])
next
assume ∗:*set-integrable lborel* {*a*..} *f*
show *set-integrable lborel* {*a*<..} *f*
by(*auto intro*!: *set-integrable-subset*[*OF* ∗])
qed

lemma *set-integral-atLeast-eq*:
fixes *f* :: *real* ⇒ - :: {*banach*, *second-countable-topology*}
shows (∫ *x*∈{*a*<..}. *f x* ∂*lborel*) = (∫ *x*∈{*a*..}. *f x* ∂*lborel*)
proof(*cases set-integrable lborel* {*a*<..} *f*)
assume [*simp*]:*set-integrable lborel* {*a*<..} *f*
have (∫ *x*∈{*a*<..}. *f x* ∂*lborel*) = (∫ *x*∈{*a*<..}. *f x* ∂*lborel*) + (∫ *x*∈{*a*}. *f x* ∂*lborel*)
by *simp*
also have ... = (∫ *x*∈{*a*<..} ∪ {*a*}. *f x* ∂*lborel*)
by(*subst set-integral-Un*) *auto*
also have ... = (∫ *x*∈{*a*..}. *f x* ∂*lborel*)
by(*auto intro*!: *arg-cong*[**where** *f*=λ*B*. (∫ *x*∈*B*. *f x* ∂*lborel*)])

finally show *?thesis* **.**
qed (*auto simp*: *set-integrable-atLeast-eq not-set-integrable-integral-eq*)

lemma *set-nn-integral-atLeast-eq*:
fixes *a* :: - :: *linorder-topology*
shows $(\int^+ x{\in}\{a<..\}.\ f\ x\ \partial lborel) = (\int^+ x{\in}\{a..\}.\ f\ x\ \partial lborel)$
apply(*safe intro*!: *nn-integral-cong-AE AE-I'*[**where** $N{=}\{a\}$])
apply (*simp add*: *countable-imp-null-set-lborel*)
apply (*metis atLeast-iff greaterThan-iff indicator-simps order-less-le*)
done

lemma *set-integrable-atMost-eq*:
fixes *b* :: - :: *linorder-topology*
fixes *f* :: - ⇒ - :: {*banach*, *second-countable-topology*}
shows *set-integrable lborel* $\{..<b\}$ *f* ⟷ *set-integrable lborel* $\{..b\}$ *f*
proof
assume *set-integrable lborel* $\{..<b\}$ *f*
from *set-integrable-Un*[*OF set-borel-integrable-singleton'*[*of b f*] *this*]
have ∗:*set-integrable lborel* $(\{b\} \cup \{..<b\})$ *f*
by *simp*
show *set-integrable lborel* $\{..b\}$ *f*
by(*auto intro*!: *set-integrable-subset*[*OF* ∗])
next
assume ∗:*set-integrable lborel* $\{..b\}$ *f*
then show *set-integrable lborel* $\{..<b\}$ *f*
by(*rule set-integrable-subset*) *auto*
qed

lemma *set-integral-atMost-eq*:
fixes *f* :: *real* ⇒ - :: {*banach*, *second-countable-topology*}
shows $(\int x{\in}\{..<b\}.\ f\ x\ \partial lborel) = (\int x{\in}\{..b\}.\ f\ x\ \partial lborel)$
proof(*cases set-integrable lborel* $\{..<b\}$ *f*)
assume [*simp*]:*set-integrable lborel* $\{..<b\}$ *f*
have $(\int x{\in}\{..<b\}.\ f\ x\ \partial lborel) = (\int x{\in}\{..<b\}.\ f\ x\ \partial lborel) + (\int x{\in}\{b\}.\ f\ x\ \partial lborel)$
by *simp*
also have ... = $(\int x{\in}\{..<b\} \cup \{b\}.\ f\ x\ \partial lborel)$
by(*subst set-integral-Un*) *auto*
also have ... = $(\int x{\in}\{..b\}.\ f\ x\ \partial lborel)$
by(*auto intro*!: *arg-cong*[**where** $f{=}\lambda B.\ (\int x{\in}B.\ f\ x\ \partial lborel)$])
finally show *?thesis* **.**
qed (*auto simp*: *set-integrable-atMost-eq not-set-integrable-integral-eq*)

lemma *set-nn-integral-atMost-eq*:
fixes *b* :: - :: *linorder-topology*
shows $(\int^+ x{\in}\{..<b\}.\ f\ x\ \partial lborel) = (\int^+ x{\in}\{..b\}.\ f\ x\ \partial lborel)$
apply(*safe intro*!: *nn-integral-cong-AE AE-I'*[**where** $N{=}\{b\}$])
apply (*simp add*: *countable-imp-null-set-lborel*)
apply (*metis atMost-iff indicator-simps lessThan-iff less-le*)
done

lemma *AE-False-iff-False*:
assumes $0 < \mathit{emeasure}\ M\ A$
shows $(\mathit{AE}\ x{\in}A\ \mathit{in}\ M.\ \mathit{False}) \longleftrightarrow \mathit{False}$
proof
assume $\mathit{AE}\ x{\in}A\ \mathit{in}\ M.\ \mathit{False}$
thus *False*
using *AE-iff-null-sets assms emeasure-notin-sets less-irrefl null-setsD1*
by *metis*
qed *simp*

lemma *nn-integral-less'*:
assumes [*measurable*]: $f \in \mathit{borel\text{-}measurable}\ M\ g \in \mathit{borel\text{-}measurable}\ M$
assumes *f*: $(\int^+ x.\ f\ x\ \partial M) \neq \infty$
and *ord1*: $\mathit{AE}\ x \in A\ \mathit{in}\ M.\ f\ x < g\ x$ **and** *ord2*:$\mathit{AE}\ x\ \mathit{in}\ M.\ f\ x \leq g\ x$
and *pos*:$0 < \mathit{emeasure}\ M\ A$
shows $(\int^+ x.\ f\ x\ \partial M) < (\int^+ x.\ g\ x\ \partial M)$
proof(*rule nn-integral-less*)
have *A*:$A \in \mathit{sets}\ M$
using *pos emeasure-notin-sets* **by** *fastforce*
show $\neg\ (\mathit{AE}\ x\ \mathit{in}\ M.\ g\ x \leq f\ x)$
proof
assume *h*:$\mathit{AE}\ x\ \mathit{in}\ M.\ g\ x \leq f\ x$
have $\mathit{AE}\ x{\in}A\ \mathit{in}\ M.\ \mathit{False}$
by(*rule eventually-mono*[*OF eventually-conj*[*OF h ord1*]]) *auto*
thus *False*
using *pos AE-False-iff-False* **by** *blast*
qed
qed(*use assms* **in** *auto*)

lemma *set-nn-integral-less'*:
fixes *f* :: - ⇒ *ennreal*
assumes $(\lambda x.\ f\ x * \mathit{indicator}\ A\ x) \in \mathit{borel\text{-}measurable}\ M$
and $(\lambda x.\ g\ x * \mathit{indicator}\ A\ x) \in \mathit{borel\text{-}measurable}\ M$
and $A \in \mathit{sets}\ M$
and $B \subseteq A$
and *le*:$\mathit{AE}\ x{\in}B\ \mathit{in}\ M.\ f\ x < g\ x\ \mathit{AE}\ x{\in}A\ \mathit{in}\ M.\ f\ x \leq g\ x$
and $(\int^+ x{\in}A.\ f\ x\ \partial M) \neq \infty$
and $0 < \mathit{emeasure}\ M\ B$
shows $(\int^+ x{\in}A.\ f\ x\ \partial M) < (\int^+ x{\in}A.\ g\ x\ \partial M)$
using *assms*
by(*intro nn-integral-less'*[**where** $A{=}B$] *eventually-mono*[*OF AE-conjI*[*OF le*]])
(*auto simp*: *indicator-def*)

lemma *set-nn-integral-less*:
fixes *f* :: - ⇒ *ennreal*
assumes $(\lambda x.\ f\ x * \mathit{indicator}\ A\ x) \in \mathit{borel\text{-}measurable}\ M$
and $(\lambda x.\ g\ x * \mathit{indicator}\ A\ x) \in \mathit{borel\text{-}measurable}\ M$
and $\mathit{AE}\ x{\in}A\ \mathit{in}\ M.\ f\ x < g\ x$

```
    and (∫⁺x∈A. f x ∂M) ≠ ∞
    and 0 < emeasure M A
  shows (∫⁺x∈A. f x ∂M) < (∫⁺x∈A. g x ∂M)
  using emeasure-neq-0-sets[of M A] assms
  by(auto intro!: set-nn-integral-less'[of f A M g A])

lemma set-nn-integral-zero-less:
  assumes [measurable]:(λx. g x * indicator A x) ∈ borel-measurable M
    and pos:AE x∈A in M. 0 < g x
    and meas:0 < emeasure M A
  shows 0 < (∫⁺x∈A. g x ∂M)
  using set-nn-integral-less[where f=λx. 0 and g=g and A=A] assms
  by fastforce

lemma set-integral-zero-less:
  fixes f :: - ⇒ real
  assumes integ:set-integrable M A f
    and le:AE x∈A in M. 0 < f x
    and meas:0 < emeasure M A
  shows 0 < (∫ x∈A. f x ∂M)
proof −
  have ∗:enn2real (∫⁺x∈A. ennreal (f x) ∂M) = (∫ x∈A. f x ∂M)
  proof −
    have enn2real (∫⁺x∈A. ennreal (f x) ∂M)
        = enn2real (∫⁺x. ennreal (indicator A x ∗ f x) ∂M)
      by(auto intro!: arg-cong[where f=enn2real] nn-integral-cong simp: indicator-def)
    also have ... = enn2real (ennreal (∫ x. indicator A x ∗ f x ∂M))
      using integ le
      by(subst nn-integral-eq-integral) (auto simp: set-integrable-def)
    also have ... = (∫ x∈A. f x ∂M)
      using le
      by(subst enn2real-ennreal)
        (auto simp: set-lebesgue-integral-def intro!: integral-nonneg-AE)
    finally show ?thesis .
  qed
  have [measurable]:(λx. ennreal (f x) ∗ indicator A x) ∈ borel-measurable M
    by (smt (verit, ccfv-threshold) integ mult.commute indicator-mult-ennreal
      integrable-iff-bounded measurable-compose measurable-cong measurable-ennreal
        real-scaleR-def set-integrable-def)
  have 0 < (∫⁺x∈A. ennreal (f x) ∂M)
    using assms
    by(auto intro!: set-nn-integral-zero-less simp: set-integrable-def)
  hence 0 < enn2real (∫⁺x∈A. ennreal (f x) ∂M)
    unfolding enn2real-positive-iff
    by (smt (verit, best) integ ennreal-0 ennreal-less-zero-iff ennreal-mult-top
      ennreal-zero-less-mult-iff indicator-def indicator-def indicator-def indicator-def
        indicator-eq-0-iff indicator-eq-0-iff indicator-simps indicator-times-eq-if
        integrable-iff-bounded less-top measurable-ennreal mult.right-neutral
```

mult-eq-0-iff mult-eq-0-iff mult-zero-right nless-le nn-integral-mono not-le real-norm-def real-scaleR-def set-integrable-def)
also have ... = $(\int x{\in}A.\ f\ x\ \partial M)$
unfolding $*$ **..**
finally show *?thesis* **.**
qed

lemma *set-integral-zero-less'*:
fixes f :: - $\Rightarrow$ *real*
assumes *integ*:*set-integrable M A f*
and *le*:$AE\ x{\in}B$ *in* $M.\ 0 < f\ x$
and *le2*:$AE\ x{\in}A$ *in* $M.\ 0 \leq f\ x$
and [*measurable*]:$A \in$ *sets M*
and *AB*:$B \subseteq A$
and *meas*:$0 <$ *emeasure M B*
shows $0 < (\int x{\in}A.\ f\ x\ \partial M)$
proof −
have [*measurable*]:$B \in$ *sets M*
using *meas emeasure-notin-sets* **by** *fastforce*
have $0 < (\int x{\in}B.\ f\ x\ \partial M)$
by(*auto intro*!: *set-integral-zero-less set-integrable-subset*[*OF integ*] *assms*)
also have ... $\leq (\int x{\in}A.\ f\ x\ \partial M)$
using *AB*
by(*auto intro*!: *set-integral-subset-AE integ eventually-mono*[*OF le2*])
finally show *?thesis* **.**
qed

lemma *set-integral-less'*:
fixes f :: - $\Rightarrow$ *real*
assumes *f*:*set-integrable M A f*
and *g*:*set-integrable M A g*
and [*measurable*]:$A \in$ *sets M*
and *AB*:$B \subseteq A$
and *le*:$AE\ x{\in}B$ *in* $M.\ f\ x < g\ x$ **and** *le2*:$AE\ x{\in}A$ *in* $M.\ f\ x \leq g\ x$
and *meas*:$0 <$ *emeasure M B*
shows $(\int x{\in}A.\ f\ x\ \partial M) < (\int x{\in}A.\ g\ x\ \partial M)$
proof −
have $0 < (\int x{\in}A.\ g\ x - f\ x\ \partial M)$
using *assms*
by(*intro set-integral-zero-less'*[**where** *B*=*B*] *set-integral-diff*) *auto*
also have ... = $(\int x{\in}A.\ g\ x\ \partial M) - (\int x{\in}A.\ f\ x\ \partial M)$
using *assms* **by** *auto*
finally show *?thesis*
by *simp*
qed

lemma *set-integral-less*:
fixes f :: - $\Rightarrow$ *real*
assumes *f*:*set-integrable M A f*

and *g*:*set-integrable M A g*
and *le*:*AE x*∈*A in M*. *f x* < *g x*
and *meas*:*0* < *emeasure M A*
shows (∫ *x*∈*A*. *f x* ∂*M*) < (∫ *x*∈*A*. *g x* ∂*M*)
using *assms*
by(*intro set-integral-less'*[**where** *B*=*A*])
(*auto simp*: *emeasure-neq-0-sets*)

corollary *set-itegral-strict-mono-Icc*:
fixes *f* :: *real* ⇒ *real*
fixes *a b* :: *real*
assumes *integ*:*set-integrable lborel* {*a*..*b*} *f*
and *le*:*AE x*∈{*a*<..*b*} *in lborel*. *0* < *f x*
shows *strict-mono-on* {*a*..*b*} (λ*x*. (∫ *t*∈{*a*<..*x*}. *f t* ∂*lborel*))
proof(*safe intro*!: *strict-mono-onI*)
fix *r s*
assume *r* ∈ {*a*..*b*} *s* ∈ {*a*..*b*} **and** *rs*[*arith*]:*r* < *s*
then have *rab*[*arith*]:*a* ≤ *r r* ≤ *b* **and** *sab*[*arith*]:*a* ≤ *s s* ≤ *b*
by *simp-all*
with *rs* **have** (∫ *t*∈{*a*<..*r*}. *f t* ∂*lborel*) < (∫ *t*∈{*a*<..*r*}. *f t* ∂*lborel*) + (∫ *t*∈{*r*<..*s*}. *f t* ∂*lborel*)
by(*auto intro*!: *set-integral-zero-less set-integrable-subset*[*OF integ*] *eventually-mono*[*OF le*])
also have ... = (∫ *t*∈{*a*<..*s*}. *f t* ∂*lborel*)
by(*subst set-integral-Un*[*symmetric*])
(*auto intro*!: *set-integrable-subset*[*OF integ*] *arg-cong*[**where** *f*=λ*A*. (∫ *t*∈*A*. *f t* ∂*lborel*)])
finally show (∫ *t*∈{*a*<..*r*}. *f t* ∂*lborel*) <(∫ *t*∈{*a*<..*s*}. *f t* ∂*lborel*) .
qed

corollary *set-itegral-tail-strict-antimono*:
fixes *f* :: *real* ⇒ *real*
assumes *integ*:*set-integrable lborel* {*a*..} *f*
and *le*:*AE x*∈{*a*<..} *in lborel*. *0* < *f x*
shows *strict-antimono-on* {*a*..} (λ*x*. (∫ *t*∈{*x*<..}. *f t* ∂*lborel*))
proof(*safe intro*!: *monotone-onI*)
fix *x y*
assume *x*[*arith*]:*a* ≤ *x* **and** *y*[*arith*]:*a* ≤ *y* **and** *xy*[*arith*]:*x* < *y*
have (∫ *t*∈{*y*<..}. *f t* ∂*lborel*) < (∫ *t*∈{*x*<..*y*}. *f t* ∂*lborel*) + (∫ *t*∈{*y*<..}. *f t* ∂*lborel*)
by(*auto intro*!: *set-integral-zero-less set-integrable-subset*[*OF integ*] *eventually-mono*[*OF le*])
also have ... = (∫ *t*∈{*x*<..}. *f t* ∂*lborel*)
by(*subst set-integral-Un*[*symmetric*])
(*auto intro*!: *set-integrable-subset*[*OF integ*] *arg-cong*[**where** *f*=λ*A*. (∫ *t*∈*A*. *f t* ∂*lborel*)])
finally show (∫ *t*∈{*y*<..}. *f t* ∂*lborel*) < (∫ *t*∈{*x*<..}. *f t* ∂*lborel*) .
qed

1.4 Integral of Composite Functions $g(\int_{t\in A_x} f(t)\,\mathrm{d}t)$

lemma *continuous-on-LBINT-Ioc*:
fixes f :: *real* ⇒ *real*
assumes f: ⋀b. $a \le b \Longrightarrow$ *set-integrable lborel* {a..b} f
shows *continuous-on UNIV* (λb. *LBINT* x:{a<..b}. $f\,x$)
by (*simp add*: *LBINT-Ioc-eq-Icc continuous-on-LBINT f*)

corollary *continuous-on-LBINT-tail*:
fixes f :: *real* ⇒ *real*
assumes f: *set-integrable lborel* {a..} f
shows *continuous-on* {a..} (λb. *LBINT* x:{b<..}. $f\,x$)
proof(*subst continuous-on-cong*)
fix b
assume b:$b \in$ {a..}
then have (*LBINT* x:{b<..}. $f\,x$) + (*LBINT* x:{a<..b}. $f\,x$) = (*LBINT* x:{a<..}. $f\,x$)
by(*subst set-integral-Un*[*symmetric*])
(*auto intro*!: *set-integrable-subset*[*OF f*] *arg-cong*[**where** f=λB. (*LBINT* x:B. $f\,x$)])
thus (*LBINT* x:{b<..}. $f\,x$) = (*LBINT* x:{a<..}. $f\,x$) − (*LBINT* x:{a<..b}. $f\,x$)
by *argo*
next
show *continuous-on* {a..} (λb. (*LBINT* x:{a<..}. $f\,x$) − (*LBINT* x:{a<..b}. $f\,x$))
by(*auto intro*!: *continuous-on-diff continuous-on-subset*[*OF continuous-on-LBINT-Ioc*] *set-integrable-subset*[*OF f*])
qed *simp*

corollary *continuous-on-LBINT-com*:
fixes f :: *real* ⇒ *real*
assumes *integrable lborel f*
shows *isCont* (λb. *LBINT* x:{..b}. $f\,x$) x
proof(*rule continuous-on-interior*)
have f:*set-integrable lborel UNIV f*
using *assms* **by**(*simp add*: *set-integrable-def*)
show *continuous-on* {$x-1$<..<$x+1$} (λb. *LBINT* x:{..b}. $f\,x$)
thm *continuous-on-eq-continuous-at*[*OF open-UNIV*]
proof(*subst continuous-on-cong*)
fix b
assume $b \in$ {$x - 1$<..<$x + 1$}
thus (*LBINT* x:{..b}. $f\,x$) = (*LBINT* x:{..$x-1$}. $f\,x$) + (*LBINT* x:{$x-1$<..b}. $f\,x$)
by(*subst set-integral-Un*[*symmetric*])
(*auto intro*!: *set-integrable-subset*[*OF f*] *arg-cong*[**where** f=λB. (*LBINT* x:B. $f\,x$)])
next
show *continuous-on* {$x-1$<..<$x+1$} (λb. (*LBINT* x:{..$x-1$}. $f\,x$) + (*LBINT* x:{$x-1$<..b}. $f\,x$))
by(*auto intro*!: *continuous-on-subset*[*OF continuous-on-LBINT-Ioc*] *continuous-on-add set-integrable-subset*[*OF f*])

qed *simp*
qed *simp*

lemma *sets-restrict-space-sigma-sets*:
assumes *sets M* = *sigma-sets* Ω *E*
and *A*:*A* ∈ *sets M* **and** *Ao*:*A* ⊆ Ω
shows *sets* (*restrict-space M A*) = *sigma-sets A* ((∩) *A* ' *E*)
by (*metis assms sets-restrict-space sigma-sets-Int*)

lemma *Int-stable-image-inter*:
assumes *Int-stable A*
shows *Int-stable* ((∩) *a* ' *A*)
proof(*intro Int-stableI-image*)
fix *i j*
assume *i* ∈ *A j* ∈ *A*
thus ∃ *k*∈*A*. *a* ∩ *i* ∩ (*a* ∩ *j*) = *a* ∩ *k*
using *Int-stableD*[*OF assms*]
by(*auto intro*!: *bexI*[**where** *x*=*i* ∩ *j*])
qed

lemma
fixes *f* **and** *g*:: *real* ⇒ *real* **and** *A B*
defines *F* ≡ λ*x*::*real*. (∫ *t*∈*A x*. *f t* ∂*lborel*)
defines *M* ≡ *density* (*restrict-space lborel B*) *f*
assumes *f-pos*:*AE x* ∈ *B in lborel*. 0 ≤ *f x*
and *f-int*:⋀*x*. *x* ∈ *X* ⟹ *set-integrable lborel* (*A x*) *f*
and *F-meas*:*F* ∈ (*restrict-space lborel B*) →$_M$ *restrict-space borel C*
and *B*:*F* '*B* = *C* **and** [*measurable*]:*B* ∈ *sets borel C* ∈ *sets borel* ⋀*x*. *A x* ∈ *sets borel*
and *AB*:*B* ⊆ *A a* **and** *a*:*a* ∈ *X*
and *meq*:*restrict-space lborel C* = *distr M* (*restrict-space borel C*) *F*
shows *set-integral-compose-integral*:
g ∈ *borel-measurable* (*restrict-space borel C*)
⟹ (∫ *t*∈*B*. *g* (*F t*) ∗ *f t* ∂*lborel*) = (∫ *t*∈*C*. *g t* ∂*lborel*)
and *set-integrable-compose-integral*:
set-integrable lborel C g
⟹ *set-integrable lborel B* (λ*t*. *g* (*F t*) ∗ *f t*)
proof −
have [*simp*]:*AE x in restrict-space lborel B*. 0 ≤ *f x*
by (*simp add*: *AE-restrict-space-iff f-pos*)
have *space-M*:*space M* = *B*
by(*simp add*: *M-def*)
have *indicat-eq*:*x* ∈ *B* ⟹ *indicat-real B x* = *indicat-real C* (*F x*) **for** *x*
using *B* **by** *auto*

have *F-meas1*:*F* ∈ *M* →$_M$ *restrict-space borel C*
using *F-meas* **by**(*simp add*: *M-def*)

have *f-intbc*[*measurable*]:*integrable* (*restrict-space lborel B*) *f*

```
  using set-integrable-subset[OF f-int[OF a] - AB]
  by(simp add: set-integrable-eq)

show (∫ t∈B. g (F t) * f t ∂lborel) = (∫ t∈C. g t ∂lborel)
  if g-meas:g ∈ borel-measurable (restrict-space borel C)
proof −
  have [measurable]:(λx. indicat-real B x * g (F x))
                    ∈ borel-measurable (restrict-space lborel B)
    using measurable-compose[OF F-meas1 g-meas]
      measurable-restrict-space1[of - - borel B,OF measurable-id]
    by(auto simp add: M-def)
  have [measurable]:(λt. indicat-real C t * g t)
                    ∈ borel-measurable (restrict-space borel C)
    by (simp add: borel-measurable-times g-meas measurable-restrict-space1)
  have (∫ t∈B. g (F t) * f t ∂lborel) = (∫ t∈B. g (F t) ∂M)
    unfolding set-lebesgue-integral-def M-def
    by(subst integral-density)
      (auto simp: integral-restrict-space indicator-times-eq-if
        intro!: f-pos Bochner-Integration.integral-cong[OF refl])
  also have ... = (∫ t. indicat-real C t * g t ∂distr M (restrict-space borel C) F)
    unfolding set-lebesgue-integral-def using F-meas1
    by(subst integral-distr)
    (auto intro!: Bochner-Integration.integral-cong simp: space-M indicat-eq[symmetric])
  thm meq
  also have ... = (∫ t∈C. g t ∂(restrict-space lborel C))
    by(simp add: meq[symmetric] set-lebesgue-integral-def)
  also have ... = (∫ t∈C. g t ∂lborel)
    unfolding set-lebesgue-integral-def
    by(subst integral-restrict-space)
      (auto intro!: Bochner-Integration.integral-cong simp: indicator-def)
  finally show (∫ t∈B. g (F t) * f t ∂lborel) = (∫ t∈C. g t ∂lborel) .
qed
assume g-integ:set-integrable lborel C g
have g-meas:g ∈ borel-measurable (restrict-space borel C)
  using g-integ by(simp add: set-integrable-eq)

have [measurable]:(λx. indicat-real B x * g (F x))
                  ∈ borel-measurable (restrict-space lborel B)
  using measurable-compose[OF F-meas1 g-meas]
    measurable-restrict-space1[of - - borel B,OF measurable-id]
  by(auto simp add: M-def)
have [measurable]:(λt. indicat-real C t * g t)
                  ∈ borel-measurable (restrict-space borel C)
  by (simp add: borel-measurable-times g-meas measurable-restrict-space1)

have set-integrable (restrict-space lborel C) C g
  unfolding set-integrable-def
  by(rule Bochner-Integration.integrable-cong[where f=g,OF refl,THEN iffD1])
    (use g-integ in simp-all add: set-integrable-eq)
```

hence *1*:*integrable* (*distr M* (*restrict-space borel C*) *F*) (λ*t*. *indicat-real C t* ∗ *g t*)
by(*simp add*: *meq*[*symmetric*] *set-integrable-def*)
have *set-integrable M B* (λ*t*. *g* (*F t*))
using *F-meas1*
by(*auto intro*!:
Bochner-Integration.integrable-cong[*THEN iffD2*,*OF refl - integrable-distr*[*OF - 1*]]
simp: *set-integrable-def indicat-eq*[*symmetric*] *space-M*)
hence *integrable* (*restrict-space lborel B*)
(λ*t*. *f t* ∗$_R$ (*indicat-real B t* ∗ *g* (*F t*)))
by(*intro integrable-density*[*THEN iffD1*])
(*auto simp*: *set-integrable-def M-def intro*!: *f-pos*)
hence *2*:*integrable lborel*
(λ*t*. *indicat-real B t* ∗$_R$ *f t* ∗$_R$ (*indicat-real B t* ∗ *g* (*F t*)))
by(*subst integrable-restrict-space*[*symmetric*]) *auto*
show *set-integrable lborel B* (λ*t*. *g* (*F t*) ∗ *f t*)
unfolding *set-integrable-def*
by(*auto intro*!: *Bochner-Integration.integrable-cong*[*THEN iffD1*,*OF refl - 2*]
simp : *indicator-def*)
qed

lemma
fixes *f* **and** *g*:: *real* ⇒ *real* **and** *a b c*::*real*
defines *F* ≡ λ*x*. (∫ *t*∈{*a*<..*x*}. *f t* ∂*lborel*)
assumes *f-pos*:⋀*x*. *a* < *x* ⟹ 0 < *f x*
and *f-int*:⋀*d*. *a* ≤ *d* ⟹ *set-integrable lborel* {*a*..*d*} *f*
and *ab*[*arith*]:*a* ≤ *b* **and** *bc*[*arith*]:*b* ≤ *c*
shows *set-integral-compose-integral-Ioc*:
g ∈ *borel-measurable* (*restrict-space borel* {*F b*<..*F c*})
⟹ (∫ *t*∈{*b*<..*c*}. *g* (*F t*) ∗ *f t* ∂*lborel*) = (∫ *t*∈{*F b*<..*F c*}. *g t* ∂*lborel*)
and *set-integrable-compose-integral-Ioc*:
set-integrable lborel {*F b*<..*F c*} *g*
⟹ *set-integrable lborel* {*b*<..*c*} (λ*t*. *g* (*F t*) ∗ *f t*)
proof −
define *M* **where** *M* ≡ *density* (*restrict-space lborel* {*b*<..*c*}) *f*
have *space-M*:*space M* = {*b*<..*c*}
by(*simp add*: *M-def*)
have *F-strict-mono*:⋀*d*. *a* ≤ *d* ⟹ *strict-mono-on* {*a*..*d*} *F*
using *f-pos* **unfolding** *F-def*
by(*intro set-itegral-strict-mono-Icc f-int*) *auto*
have *F-0*:⋀*d*. *d* ≤ *a* ⟹ *F d* = 0
by(*simp add*: *F-def*)
have *F-pos*:*a* < *d* ⟹ 0 < *F d* **for** *d*
unfolding *F-def* **using** *f-pos*
by(*intro set-integral-zero-less set-integrable-subset*[*OF f-int*[*of d*]]) *auto*
have *F-cont*:*continuous-on UNIV F*
unfolding *F-def*
by(*intro continuous-on-LBINT-Ioc f-int*)

```
  have im-F:b ≤ d ⟹ F ‘ {b<..d} = {F b<..F d} for d
   by(intro image-mono-on-continuous-on-Ioc monotone-on-subset[OF F-strict-mono[of
d]]
        continuous-on-subset[OF F-cont])
     auto
  have im-F2:b ≤ d ⟹ F ‘ {b..d} = {F b..F d} for d
    by(intro image-mono-on-continuous-on-Icc
        strict-mono-on-imp-mono-on monotone-on-subset[OF F-strict-mono[of d]]
        continuous-on-subset[OF F-cont])
     auto
  have F-meas1:F ∈ M →M restrict-space borel {F b<..F c}
    using im-F[OF bc] borel-measurable-continuous-onI[OF F-cont]
    by(auto simp add: M-def intro!: measurable-restrict-space3)
  have F-meas2:F ∈ restrict-space borel {b<..c} →M restrict-space borel {F b<..F
c}
    using F-meas1 by(simp add: M-def)
  have f-intbc[measurable]:integrable (restrict-space lborel {b<..c}) f
    using set-integrable-subset[where B={b<..c},OF f-int[of c]]
    by(fastforce simp: set-integrable-eq)
  have bij:bij-betw F {b<..c} {F b<..F c}
  proof −
    have inj-on F {b<..c}
      by(subst injective-eq-monotone-map)
        (auto intro!: disjI1 monotone-on-subset[OF F-strict-mono[of c]]
          continuous-on-subset[OF F-cont])
    from inj-on-imp-bij-betw[OF this]
    show ?thesis
      unfolding im-F[OF bc] by simp
  qed

  have inv-mono:x′ ≤ y′
    if mono:F x′ ≤ F y′
      and x′[arith]:b ≤ x′ x′ ≤ c
      and y′[arith]: b ≤ y′ y′ ≤ c for x′ y′
  proof(rule ccontr)
    assume ¬ x′ ≤ y′
    then have F y′ < F x′
      using strict-mono-onD[OF F-strict-mono[of c],of y′ x′]
      by simp
    thus False
      using mono by simp
  qed
  have inv-strict-mono:x′ < y′
    if mono:F x′ < F y′
      and x′[arith]:b ≤ x′ x′ ≤ c
      and y′[arith]: b ≤ y′ y′ ≤ c for x′ y′
    by (metis inv-mono leD le-less that)
  have eq-indicat:b < x ⟹ x ≤ c ⟹
                    indicat-real {F b<..F c} (F x) = indicat-real {b<..c} x for x
```

```
    by (auto simp: indicator-def
        intro!: strict-mono-onD[OF F-strict-mono[of c]]
        mono-onD[OF strict-mono-on-imp-mono-on[OF F-strict-mono[of c]]])

  have ex-inv:F b ≤ x ⟹ x ≤ F c ⟹ ∃x'. b ≤ x' ∧ x' ≤ c ∧ F x' = x for x
    by (metis (mono-tags, lifting) bc atLeastAtMost-iff im-F2 image-iff)
   have eq:emeasure (distr M (restrict-space borel {F b<..F c}) F) {x<..y} =
ennreal (y − x)
    if xy[arith]:x ≤ y F b ≤ x y ≤ F c for x y
  proof −
    obtain x' y' where x'[arith]:b ≤ x' x' ≤ c and y'[arith]:b ≤ y' y' ≤ c
      and x'-eq:F x' = x and y'-eq:F y' = y
      using ex-inv[of x] ex-inv[of y] by auto
    have [arith]:x' ≤ y'
      by(auto intro!: inv-mono[of x' y'] simp: x'-eq y'-eq)
    have [measurable]:{x<..y} ∈ sets (restrict-space borel {F b<..F c})
      by(subst sets-restrict-space-iff) auto
    have [measurable]:{x'<..y'} ∈ sets (restrict-space borel {b<..c})
      by(subst sets-restrict-space-iff) auto
    have eq-set:F −‘ {x<..y} ∩ {b<..c} = {x'<..y'}
      using x'-eq y'-eq inv-mono[of - y'] inv-strict-mono[of x']
      by(auto intro!: strict-mono-onD[OF F-strict-mono[of c]]
          mono-onD[OF strict-mono-on-imp-mono-on[OF F-strict-mono[of c]]])
    have emeasure (distr M (restrict-space borel {F b<..F c}) F) {x<..y}
        = emeasure M (F −‘ {x<..y} ∩ space M)
      by(intro emeasure-distr F-meas1) simp
    also have ... = emeasure M {x'<..y'}
      by(simp add: M-def eq-set)
    also have ... = (∫⁺x∈{x'<..y'}. ennreal (f x) ∂restrict-space lborel {b<..c})
      unfolding M-def
      by(subst emeasure-density) auto
    also have ... = ennreal (∫ x∈{x'<..y'}. f x∂restrict-space lborel {b<..c})
      by(intro nn-set-integral-eq-set-integral f-intbc)
        (auto intro!: less-imp-le[OF f-pos])
    also have ... = ennreal (∫ x∈{x'<..y'}. f x∂lborel)
      by(subst set-integral-restrict-space)
        (auto intro!: arg-cong[where f=ennreal] Bochner-Integration.integral-cong
          simp: set-lebesgue-integral-def indicator-def)
    also have ... = ennreal (∫ x. indicator {a<..y'} x * f x − indicator {a<..x'} x
* f x∂lborel)
      by(auto intro!: arg-cong[where f=ennreal] Bochner-Integration.integral-cong
          simp: set-lebesgue-integral-def indicator-def)
    also have ... = ennreal (F y' − F x')
      using set-integrable-subset[where B={a<..y'},OF f-int[of c]]
        set-integrable-subset[where B={a<..x'},OF f-int[of c]]
      unfolding F-def set-lebesgue-integral-def real-scaleR-def set-integrable-def
     by(fastforce intro!: arg-cong[where f=ennreal] Bochner-Integration.integral-diff)
    finally show ?thesis
      by(simp add: x'-eq y'-eq)
```

qed

have *meq*:*restrict-space lborel* {*F b*<..*F c*} = *distr M* (*restrict-space borel* {*F b*<..*F c*}) *F*
proof(*rule measure-eqI-generator-eq-countable*
[**where** Ω={*F b*<..*F c*} **and** *E*=(∩) {*F b*<..*F c*} ‘ (*range* (λ(*a*, *b*). {*a* <.. *b*::*real*}))
and *A*=(∩) {*F b*<..*F c*} ‘ ((λ(*a*, *b*). {*a* <.. *b*::*real*}) ‘ (ℚ × ℚ))])
show *Int-stable* ((∩) {*F b*<..*F c*} ‘ *range* (λ(*a*, *b*). {*a*<..*b*}))
by(*intro Int-stable-image-inter*) (*fastforce intro*!: *Int-stableI*)
next
have ∗:*sets* (*distr M* (*restrict-space borel* {*F b*<..*F c*}) *F*) =
sets (*restrict-space lborel* {*F b*<..*F c*})
by(*simp add*: *M-def cong*: *sets-restrict-space-cong*)

show *sets* (*restrict-space lborel* {*F b*<..*F c*}) =
sigma-sets {*F b*<..*F c*} ((∩) {*F b*<..*F c*} ‘ *range* (λ(*a*, *b*). {*a*<..*b*}))
by(*intro sets-restrict-space-sigma-sets*[**where** Ω=*UNIV*])
(*auto simp*: *borel-sigma-sets-Ioc*)
thus *sets* (*distr M* (*restrict-space borel* {*F b*<..*F c*}) *F*) =
sigma-sets {*F b*<..*F c*} ((∩) {*F b*<..*F c*} ‘ *range* (λ(*a*, *b*). {*a*<..*b*}))
unfolding ∗ **by** *meson*
next
fix *X*
assume *X*:*X* ∈ (∩) {*F b*<..*F c*} ‘ *range* (λ(*a*, *b*). {*a*<..*b*})
then obtain *x y* **where** *X* = {*x*<..*y*} *F b* ≤ *x y* ≤ *F c*
by *fastforce*
thus *emeasure* (*restrict-space lborel* {*F b*<..*F c*}) *X* =
emeasure (*distr M* (*restrict-space borel* {*F b*<..*F c*}) *F*) *X*
by(*cases x* ≤ *y*, *subst emeasure-restrict-space*)
(*auto simp add*: *eq emeasure-restrict-space*)
next
have *real-of-int* ⌊*F b*⌋ ∈ ℚ *real-of-int* ⌈*F c*⌉ ∈ ℚ
{*F b*<..*F c*} ∩ {⌊*F b*⌋<..⌈*F c*⌉} = {*F b*<..*F c*}
by *auto*
thus ⋃ ((∩) {*F b*<..*F c*} ‘ (λ(*a*, *b*). {*a*<..*b*}) ‘ (ℚ × ℚ)) = {*F b*<..*F c*}
by *blast*
next
fix *a*
assume *a*:*a* ∈ (∩) {*F b*<..*F c*} ‘ (λ(*a*, *b*). {*a*<..*b*}) ‘ (ℚ × ℚ)
then have *bounded a*
using *bounded-Ioc* **by** *blast*
from *emeasure-bounded-finite*[*OF this*] *a*
show *emeasure* (*restrict-space lborel* {*F b*<..*F c*}) *a* ≠ ∞
by(*subst emeasure-restrict-space*) *auto*
qed(*auto intro*!: *countable-rat*)

have *f-int′*:⋀*x*. *x* ∈ {*a*..} ⟹ *set-integrable lborel* {*a*..*x*} *f*
by (*simp add*: *f-int*)

```
  have c-in:c ∈ {a..}
    by simp
  have f-nonneg:AE x ∈ {b<..c} in lborel. 0 ≤ f x
    by(auto intro!: less-imp-le f-pos)
  show (∫ t∈{b<..c}. g (F t) * f t ∂lborel) = (∫ t∈{F b<..F c}. g t ∂lborel)
    if g-meas:g ∈ borel-measurable (restrict-space borel {F b<..F c})
    using
      set-integral-compose-integral[where A=λx. {a..x} and B={b<..c}
        and C={F b<..F c} and a=c and X={a..} and f=f and g=g
        , OF f-nonneg f-int' - - - - - - c-in - g-meas] meq F-meas2 im-F[of c]
    by (simp add: LBINT-Ioc-eq-Icc greaterThanAtMost-subseteq-atLeastAtMost-iff
F-def M-def)

  show set-integrable lborel {b<..c} (λt. g (F t) * f t)
    if g-integ:set-integrable lborel {F b<..F c} g
    using
      set-integrable-compose-integral[where A=λx. {a..x} and B={b<..c}
        and C={F b<..F c} and a=c and X={a..} and f=f and g=g
        , OF f-nonneg f-int' - - - - - - c-in - g-integ] meq F-meas2 im-F[of c]
    by (simp add: LBINT-Ioc-eq-Icc greaterThanAtMost-subseteq-atLeastAtMost-iff
F-def M-def)
qed


lemma
  fixes f and g:: real ⇒ real and a b c::real
  defines F ≡ λx. (∫ t∈{x<..}. f t ∂lborel)
  assumes f-pos:⋀x. a < x ⟹ 0 < f x
    and f-int:set-integrable lborel {a..} f
    and ab[arith]:a ≤ b and bc[arith]:b ≤ c
  shows set-integral-compose-integral-tail:
    g ∈ borel-measurable (restrict-space borel {F c<..F b})
      ⟹ (∫ t∈{b<..c}. g (F t) * f t ∂lborel) = (∫ t∈{F c<..F b}. g t ∂lborel)
    and set-integrable-compose-integral-tail:
    set-integrable lborel {F c<..F b} g
      ⟹ set-integrable lborel {b<..c} (λt. g (F t) * f t)
proof −
  have g-meas':g ∈ borel-measurable (restrict-space borel {F c..F b})
    if h:g ∈ borel-measurable (restrict-space borel {F c<..F b})
  proof −
    have ∗:− {F c} ∩ {F c..F b} = {F c<..F b}
      by auto
    have g = (λx. if x ∈ −{F c} then g x else g (F c))
      by auto
    also have ... ∈ borel-measurable (restrict-space borel {F c..F b})
      by(subst measurable-restrict-space-iff[symmetric])
        (auto simp: Int-commute sets-restrict-space restrict-restrict-space ∗ h)
    finally show ?thesis .
  qed
```

define M **where** $M \equiv$ *density* (*restrict-space lborel* $\{b..c\}$) f
have *space-M*:*space* $M = \{b..c\}$
by(*simp add*: *M-def*)
have *F-strict-mono*:*strict-antimono-on* $\{a..\}$ F
using *f-pos* **unfolding** *F-def*
by(*intro set-itegral-tail-strict-antimono f-int*) *auto*
hence *F-mono*:*antimono-on* $\{a..\}$ F
using *strict-antimono-iff-antimono* **by** *blast*
have *F-pos*:$a < d \Longrightarrow 0 < F\ d$ **for** d
unfolding *F-def* **using** *f-pos*
by(*intro set-integral-zero-less set-integrable-subset*[*OF f-int*]
order.strict-trans2[*of - emeasure lborel* $\{d<..d + 1\}$
emeasure lborel $\{d<..\}$,*OF - emeasure-mono*])
auto
have *F-cont*:*continuous-on* $\{a..\}$ F
unfolding *F-def*
by(*intro continuous-on-LBINT-tail f-int*)
have *im-F*:$b \leq d \Longrightarrow F\ `\ \{b..d\} = \{F\ d..F\ b\}$ **for** d
using *monotone-on-subset*[*OF F-strict-mono*]
by(*intro image-antimono-on-continuous-on-Icc*
continuous-on-subset[*OF F-cont*])
(*auto simp*: *strict-antimono-iff-antimono*)

have *F-meas1*:$F \in M \rightarrow_M$ *restrict-space borel* $\{F\ c..F\ b\}$
using *im-F*[*OF bc*]
borel-measurable-continuous-on-restrict[*OF continuous-on-subset*[**where** t=$\{b..c\}$,*OF F-cont*]]
by(*fastforce simp add*: *M-def intro*!: *measurable-restrict-space2*)
hence *F-meas2*:$F \in$ *restrict-space borel* $\{b..c\} \rightarrow_M$ *restrict-space borel* $\{F\ c..F\ b\}$
by(*simp add*: *M-def*)
have *f-intbc*[*measurable*]:*integrable* (*restrict-space lborel* $\{b..c\}$) f
using *set-integrable-subset*[**where** B=$\{b..c\}$,*OF f-int*]
by(*fastforce simp*: *set-integrable-eq*)

have *inv-mono*:$y' \leq x'$
if *mono*:$F\ x' \leq F\ y'$
and x'[*arith*]:$b \leq x'\ x' \leq c$
and y'[*arith*]: $b \leq y'\ y' \leq c$ **for** $x'\ y'$
proof(*rule ccontr*)
assume $\neg\ y' \leq x'$
thus *False*
using *monotone-onD*[*OF F-strict-mono*,*of* $x'\ y'$] *mono*
by *simp*
qed
have *inv-strict-mono*:$y' < x'$
if *mono*:$F\ x' < F\ y'$
and x'[*arith*]:$b \leq x'\ x' \leq c$
and y'[*arith*]: $b \leq y'\ y' \leq c$ **for** $x'\ y'$

by (*metis inv-mono leD le-less that*)

have *ex-inv*:$F\ c \le x \Longrightarrow x \le F\ b \Longrightarrow \exists x'.\ b \le x' \wedge x' \le c \wedge F\ x' = x$ **for** x
by (*metis* (*mono-tags*, *lifting*) *im-F*[*OF bc*] *atLeastAtMost-iff image-iff*)

have *eq*:*emeasure* (*distr M* (*restrict-space borel* $\{F\ c..F\ b\}$) *F*) $\{x..y\} = ennreal\ (y - x)$
if *xy*[*arith*]:$x \le y\ F\ c \le x\ y \le F\ b$ **for** $x\ y$
proof −
obtain $x'\ y'$ **where** x'[*arith*]:$b \le x'\ x' \le c$ **and** y'[*arith*]:$b \le y'\ y' \le c$
and x'-*eq*:$F\ x' = x$ **and** y'-*eq*:$F\ y' = y$
using *ex-inv*[*of x*] *ex-inv*[*of y*] **by** *auto*
have [*arith*]:$y' \le x'$
by(*auto intro*!: *inv-mono*[*of x' y'*] *simp*: x'-*eq* y'-*eq*)
have [*measurable*]:$\{x..y\} \in sets\ (restrict\text{-}space\ borel\ \{F\ c..F\ b\})$
by(*subst sets-restrict-space-iff*) *simp-all*
have [*measurable*]:$\{y'..x'\} \in sets\ (restrict\text{-}space\ borel\ \{b..c\})$
by(*subst sets-restrict-space-iff*) *auto*
have *eq-set*:$F\ -`\ \{x..y\} \cap \{b..c\} = \{y'..x'\}$
using x'-*eq* y'-*eq inv-mono*[*of - y'*] *inv-mono*[*of x'*]
by(*fastforce intro*!: *monotone-onD*[*OF F-mono*])
have *emeasure* (*distr M* (*restrict-space borel* $\{F\ c..F\ b\}$) *F*) $\{x..y\}$
$= emeasure\ M\ (F\ -`\ \{x..y\} \cap space\ M)$
by(*intro emeasure-distr F-meas1*) *simp*
also have ... $= emeasure\ M\ \{y'..x'\}$
by(*simp add*: *M-def eq-set*)
also have ... $= (\int^+ x{\in}\{y'..x'\}.\ ennreal\ (f\ x)\ \partial restrict\text{-}space\ lborel\ \{b..c\})$
unfolding *M-def*
by(*subst emeasure-density*) *auto*
also have ... $= ennreal\ (\int x{\in}\{y'..x'\}.\ f\ x \partial restrict\text{-}space\ lborel\ \{b..c\})$
by(*intro nn-set-integral-eq-set-integral f-intbc*)
(*auto intro*!: *AE-I'*[**where** $N{=}\{b\}$] *less-imp-le*[*OF f-pos*]
simp: *countable-imp-null-set-lborel null-sets-restrict-space*)
also have ... $= ennreal\ (\int x{\in}\{y'..x'\}.\ f\ x \partial lborel)$
by(*subst set-integral-restrict-space*)
(*auto intro*!: *arg-cong*[**where** $f{=}ennreal$] *Bochner-Integration.integral-cong*
simp: *set-lebesgue-integral-def indicator-def*)
also have ... $= ennreal\ (\int x.\ indicator\ \{y'..\}\ x * f\ x - indicator\ \{x'{<}..\}\ x * f\ x \partial lborel)$
by(*auto intro*!: *arg-cong*[**where** $f{=}ennreal$] *Bochner-Integration.integral-cong*
simp: *set-lebesgue-integral-def indicator-def*)
also have ... $= ennreal\ ((\int x.\ indicator\ \{y'..\}\ x * f\ x\ \partial lborel)$
$- (\int x.\ indicator\ \{x'{<}..\}\ x * f\ x \partial lborel))$
using *set-integrable-subset*[**where** $B{=}\{y'..\}$,*OF f-int*]
set-integrable-subset[**where** $B{=}\{x'{<}..\}$,*OF f-int*]
by(*fastforce intro*!: *arg-cong*[**where** $f{=}ennreal$] *Bochner-Integration.integral-diff*
simp: *set-integrable-def*)
also have ... $= ennreal\ (F\ y' - F\ x')$
unfolding *F-def set-integral-atLeast-eq*[*of y'*]

by(*simp add*: *set-lebesgue-integral-def*)
finally show *?thesis*
by(*simp add*: *x′-eq y′-eq*)
qed

have *meq*:*restrict-space lborel* {*F c..F b*} = *distr M* (*restrict-space borel* {*F c..F b*}) *F*
proof(*rule measure-eqI-generator-eq-countable*
[**where** Ω={*F c..F b*} **and** *E*=(∩) {*F c..F b*} ‘ (*range* (λ(*a*, *b*). {*a* .. *b*::*real*}))
and *A*=(∩) {*F c..F b*} ‘ ((λ(*a*, *b*). {*a* .. *b*::*real*}) ‘ (ℚ × ℚ))])
show *Int-stable* ((∩) {*F c..F b*} ‘ *range* (λ(*a*, *b*). {*a..b*}))
by(*intro Int-stable-image-inter*) (*fastforce intro*!: *Int-stableI*)
next
have ∗:*sets* (*distr M* (*restrict-space borel* {*F c..F b*}) *F*) =
sets (*restrict-space lborel* {*F c..F b*})
by(*simp add*: *M-def cong*: *sets-restrict-space-cong*)
show *sets* (*restrict-space lborel* {*F c..F b*}) =
sigma-sets {*F c..F b*} ((∩) {*F c..F b*} ‘ *range* (λ(*a*, *b*). {*a..b*}))
by(*intro sets-restrict-space-sigma-sets*[**where** Ω=*UNIV*])
(*auto simp*: *borel-eq-atLeastAtMost*)
thus *sets* (*distr M* (*restrict-space borel* {*F c..F b*}) *F*) =
sigma-sets {*F c..F b*} ((∩) {*F c..F b*} ‘ *range* (λ(*a*, *b*). {*a..b*}))
unfolding ∗ **by** *meson*
next
fix *X*
assume *X*:*X* ∈ (∩) {*F c..F b*} ‘ *range* (λ(*a*, *b*). {*a..b*})
then obtain *x y* **where** *X* = {*x..y*} *F c* ≤ *x y* ≤ *F b*
by *fastforce*
thus *emeasure* (*restrict-space lborel* {*F c..F b*}) *X* =
emeasure (*distr M* (*restrict-space borel* {*F c..F b*}) *F*) *X*
by(*cases x* ≤ *y*, *subst emeasure-restrict-space*)
(*auto simp add*: *eq emeasure-restrict-space*)
next
have *real-of-int* ⌊*F c*⌋ ∈ ℚ *real-of-int* ⌈*F b*⌉ ∈ ℚ
{*F c..F b*} ∩ {⌊*F c*⌋..⌈*F b*⌉} = {*F c..F b*}
by *auto*
thus ⋃ ((∩) {*F c..F b*} ‘ (λ(*a*, *b*). {*a..b*}) ‘ (ℚ × ℚ)) = {*F c..F b*}
by *blast*
next
fix *a*
assume *a*:*a* ∈ (∩) {*F c..F b*} ‘ (λ(*a*, *b*). {*a..b*}) ‘ (ℚ × ℚ)
then have *bounded a*
by *auto*
from *emeasure-bounded-finite*[*OF this*] *a*
show *emeasure* (*restrict-space lborel* {*F c..F b*}) *a* ≠ ∞
by(*subst emeasure-restrict-space*) *auto*
qed(*auto intro*!: *countable-rat*)

have *f-int′*:⋀*x*. *x* ∈ {*a..*} ⟹ *set-integrable lborel* {*x..*} *f*

by (*auto intro*!: *set-integrable-subset*[*OF f-int*])
have *a-in*:$a \in \{a..\}$
by *simp*
have *f-nonneg*:*AE* x∈{$b..c$} *in lborel*. $0 \leq f\ x$
using *less-imp-le*[*OF f-pos*]
by(*auto intro*!: *AE-I'*[**where** N={b}])

show $(\int t$∈{b<..c}. $g\ (F\ t) * f\ t\ \partial lborel) = (\int t$∈{$F\ c$<..$F\ b$}. $g\ t\ \partial lborel)$
if *g-meas*:$g \in$ *borel-measurable* (*restrict-space borel* {$F\ c$<..$F\ b$})
using *set-integral-compose-integral*[**where** A=λx. {$x..$} **and** B={$b..c$}
and C={$F\ c..F\ b$} **and** a=a **and** X={$a..$} **and** f=f **and** g=g
,*OF f-nonneg f-int' - - - - - - a-in - g-meas'*[*OF g-meas*]] *meq F-meas2 im-F*[*of c*]
by(*simp add*: *F-def M-def LBINT-Ioc-eq-Icc*[*symmetric*] *set-integral-atLeast-eq*)

show *set-integrable lborel* {b<..c} (λt. $g\ (F\ t) * f\ t$)
if *integ*:*set-integrable lborel* {$F\ c$<..$F\ b$} g
using *set-integrable-compose-integral*[**where** A=λx. {$x..$} **and** B={$b..c$}
and C={$F\ c..F\ b$} **and** a=a **and** X={$a..$} **and** f=f **and** g=g
,*OF f-nonneg f-int' - - - - - - a-in -] meq F-meas2 im-F*[*of c*] *integ*
by(*simp add*: *F-def M-def LBINT-Ioc-eq-Icc*[*symmetric*]
set-integral-atLeast-eq set-integrable-Ioc-eq-Icc)
qed

1.5 Convergence Theorems with Real Index

lemma *borel-measurable-LIMSEQ-metric-at-top*:
fixes f :: *real* ⇒ $'a$ ⇒ $'b$ :: *metric-space*
assumes [*measurable*]: $\bigwedge r.\ K \leq r \Longrightarrow f\ r \in$ *borel-measurable M*
assumes *lim*: $\bigwedge x.\ x \in$ *space M* $\Longrightarrow$ $((\lambda r.\ f\ r\ x) \longrightarrow g\ x)$ *at-top*
shows $g \in$ *borel-measurable M*
by(*rule borel-measurable-LIMSEQ-metric*[**where** f=$\lambda n.\ f\ (real\ n + K)$])
(*use assms* **in** *auto simp*: *at-top-real-iff-all-sequentially add.commute*[*of - K*]
intro!: *filterlim-compose*[*OF - filterlim-tendsto-add-at-top*[*OF tendsto-const*[*of K*]
filterlim-real-sequentially]])

lemma *nn-integral-at-top-real*:
assumes *mono-on* {$K..$} f $\bigwedge r.\ r \geq K \Longrightarrow f\ r \in$ *borel-measurable M*
and u: $\bigwedge x.\ ((\lambda r$::*real*. $f\ r\ x) \longrightarrow u\ x)$ *at-top*
shows $((\lambda r.\ integral^N\ M\ (f\ r)) \longrightarrow integral^N\ M\ u)$ *at-top*
using *assms*
by(*auto intro*!: *nn-integral-LIMSEQ*
simp: *at-top-real-iff-all-sequentially'*[**where** K=K] *monotone-on-def*)

theorem *nn-integral-dominated-convergence-at-top*:
assumes
$\bigwedge r$::*real*. $r \geq K \Longrightarrow u\ r \in$ *borel-measurable M*
$u' \in$ *borel-measurable M* $w \in$ *borel-measurable M*

and $\bigwedge r.\ r \geq K \Longrightarrow AE\ x\ in\ M.\ u\ r\ x \leq w\ x$
and $(\int^+ x.\ w\ x\ \partial M) < \infty$
and $AE\ x\ in\ M.\ ((\lambda r.\ u\ r\ x) \longrightarrow u'\ x)\ at\text{-}top$
shows $((\lambda r.\ (\int^+ x.\ u\ r\ x\ \partial M)) \longrightarrow (\int^+ x.\ u'\ x\ \partial M))\ at\text{-}top$
using *assms*
by(*force intro*!: *nn-integral-dominated-convergence*
simp: *at-top-real-iff-all-sequentially'*[**where** $K{=}K$])

lemma
fixes $f :: real \Rightarrow 'a \Rightarrow real$
assumes $\bigwedge r.\ K \leq r \Longrightarrow integrable\ M\ (f\ r)$ **and** *mono*: $AE\ x\ in\ M.\ mono\text{-}on\ \{K..\}\ (\lambda r.\ f\ r\ x)$
and $AE\ x\ in\ M.\ ((\lambda r.\ f\ r\ x) \longrightarrow u\ x)\ at\text{-}top$
and $((\lambda r.\ integral^L\ M\ (f\ r)) \longrightarrow x)\ at\text{-}top$
and $u{:}u \in borel\text{-}measurable\ M$
shows *integrable-monotone-convergence-real*: $integrable\ M\ u$ (**is** *?g1*)
and *integral-monotone-convergence-real*: $integral^L\ M\ u = x$ (**is** *?g2*)
and *has-bochner-integral-monotone-convergence-real*: $has\text{-}bochner\text{-}integral\ M\ u\ x$ (**is** *?g3*)
proof −
have *fl*:*filterlim* $(\lambda n.\ K + real\ n)\ at\text{-}top\ sequentially$
by(*rule filterlim-tendsto-add-at-top*) (*use filterlim-real-sequentially* **in** *auto*)
have *inc*: $incseq\ (\lambda n.\ K + real\ n)$
by(*auto intro*!: *monoI*)
have $\bigwedge n.\ integrable\ M\ (f\ (K + real\ n))\ AE\ x\ in\ M.\ incseq\ (\lambda n.\ f\ (K + real\ n)\ x)$
$AE\ x\ in\ M.\ (\lambda n.\ f\ (K + real\ n)\ x) \longrightarrow u\ x$
$(\lambda n.\ integral^L\ M\ (f\ (K + real\ n))) \longrightarrow x$
using *assms fl inc* **by**(*auto simp*: *at-top-real-iff-all-sequentially mono-on-def*)
thus *?g1 ?g2 ?g3*
using *u integrable-monotone-convergence*[**where** $f{=}\lambda n.\ f\ (K + real\ n)$]
integral-monotone-convergence[**where** $f{=}\lambda n.\ f\ (K + real\ n)$]
has-bochner-integral-monotone-convergence [**where** $f{=}\lambda n.\ f\ (K + real\ n)$]
by *blast*+
qed

lemma
fixes $f :: 'a \Rightarrow 'b{::}\{banach,\ second\text{-}countable\text{-}topology\}$ **and** $w :: 'a \Rightarrow real$
assumes $f \in borel\text{-}measurable\ M\ \bigwedge r{::}real.\ K \leq r \Longrightarrow s\ r \in borel\text{-}measurable\ M$
$integrable\ M\ w$
assumes *lim*: $AE\ x\ in\ M.\ ((\lambda r.\ s\ r\ x) \longrightarrow f\ x)\ at\text{-}top$
assumes *bound*: $\bigwedge r.\ K \leq r \Longrightarrow AE\ x\ in\ M.\ norm\ (s\ r\ x) \leq w\ x$
shows *integrable-dominated-convergence-real*: $integrable\ M\ f$
and *integrable-dominated-convergence-real2*: $\bigwedge r.\ K \leq r \Longrightarrow integrable\ M\ (s\ r)$
and *integral-dominated-convergence-real*: $((\lambda r.\ integral^L\ M\ (s\ r)) \longrightarrow integral^L\ M\ f)\ at\text{-}top$
proof −
have *fl*:*filterlim* $(\lambda n.\ r + real\ n)\ at\text{-}top\ sequentially$ **for** r
by(*rule filterlim-tendsto-add-at-top*) (*use filterlim-real-sequentially* **in** *auto*)

have *inc*: *incseq* ($\lambda n.\ r + real\ n$) **for** r
by(*auto intro*!: *monoI*)
show *integrable M f*
by(*rule integrable-dominated-convergence*[**where** $s{=}\lambda n.\ s\ (K + real\ n)$ **and** $w{=}w$])
(*use assms fl*[*of K*] *inc*[*of K*] *bound*[*of K + real* -]
in *auto simp*: *at-top-real-iff-all-sequentially*)
have *dct*:$f \in$ *borel-measurable M* $\Longrightarrow$ ($\bigwedge i.\ s\ (r + real\ i) \in$ *borel-measurable M*)
$\Longrightarrow$ *integrable M w* $\Longrightarrow$ *AE x in M*. ($\lambda i.\ s\ (r + real\ i)\ x$) $\longrightarrow f\ x$
$\Longrightarrow$ ($\bigwedge i.$ *AE x in M*. *norm* ($s\ (r + real\ i)\ x$) $\leq w\ x$) $\Longrightarrow$ *integrable M* ($s\ r$) **for** r
using *integrable-dominated-convergence2*[**where** $s{=}\lambda n.\ s\ (r + real\ n)$ **and** $i{=}0$]
by *simp*
show $K \leq r \Longrightarrow$ *integrable M* ($s\ r$) **for** r
by(*rule dct*) (*use assms fl*[*of r*] *inc*[*of r*]
in *auto simp*: *at-top-real-iff-all-sequentially*)
show (($\lambda r.$ *integral*^{L} *M* ($s\ r$)) $\longrightarrow$ *integral*^{L} *M f*) *at-top*
unfolding *at-top-real-iff-all-sequentially*′[**where** $K{=}K$]
by(*intro allI impI integral-dominated-convergence*[**where** $w{=}w$])
(*use assms* **in** *auto simp*: *at-top-real-iff-all-sequentially*)
qed

corollary
fixes f :: *real* $\Rightarrow$ *real*
assumes *f-nonneg*: $\bigwedge x.\ K < x \Longrightarrow 0 \leq f\ x$
and *integrable*: $\bigwedge r.\ K < r \Longrightarrow$ *set-integrable M* $\{K{<}..r\}$ *f*
and *le*:$\bigwedge r.\ K < r \Longrightarrow (\int x{\in}\{K{<}..r\}.\ f\ x\ \partial M) \leq G$
shows *interval-integral-inf*:
(($\lambda r.\ (\int x{\in}\{K{<}..r\}.\ f\ x\ \partial M)$) $\longrightarrow$ ($\int x{\in}\{K{<}..\}.\ f\ x\ \partial M$)) *at-top*
and *interval-integral-inf-integrable*:
set-integrable M $\{K{<}..\}$ *f*
and *measurable-integral-inf*:
set-borel-measurable M $\{K{<}..\}$ *f*
proof −
have *1*:$\bigwedge r.\ K + 1 \leq r \Longrightarrow$ *integrable M* ($\lambda x.$ *indicat-real* $\{K{<}..r\}\ x * f\ x$)
using *integrable* **by**(*auto simp*: *set-integrable-def*)
have *2*: *AE x in M*. *mono-on* $\{K + 1..\}$ ($\lambda r.$ *indicat-real* $\{K{<}..r\}\ x * f\ x$)
by(*auto intro*!: *mono-onI f-nonneg simp*: *indicator-def*)
have *3*: *AE x in M*.
(($\lambda r.$ *indicat-real* $\{K{<}..r\}\ x * f\ x$) $\longrightarrow$ *indicator* $\{K{<}..\}\ x * f\ x$) *at-top*
proof(*rule AE-I2*)
fix x
show (($\lambda r.$ *indicat-real* $\{K{<}..r\}\ x * f\ x$) $\longrightarrow$ *indicat-real* $\{K{<}..\}\ x * f\ x$) *at-top*
by(*auto intro*!: *tendstoI eventually-at-top-linorderI*[**where** $c{=}x$]
simp: *dist-real-def indicator-def*)
qed
have *mono-on* $\{K{<}..\}$ ($\lambda r.$ ($\int x{\in}\{K{<}..r\}.\ f\ x\ \partial M$))
using *integrable f-nonneg*

by(*auto intro*!: *mono-onI Bochner-Integration.integral-mono*
simp: *set-lebesgue-integral-def set-integrable-def indicator-def*)
from *mono-convergent-real-less*[*OF this le*]
obtain *l* **where** *l*:((λ*r*. (∫ *x*∈{*K*<..*r*}. *f x* ∂*M*)) ⟶ *l*) *at-top*
by *blast*
show *set-borel-measurable M* {*K*<..} *f*
unfolding *set-borel-measurable-def*
proof(*rule borel-measurable-LIMSEQ-metric*)
fix *x*
obtain *n* **where** *x* ≤ *K* + *real n*
by (*meson abs-diff-le-iff real-arch-simple*)
thus (λ*i*. *indicat-real* {*K*<..*K* + *real i* + *1*} *x* ∗ *f x*)
⟶ *indicat-real* {*K*<..} *x* ∗$_R$ *f x*
by(*auto intro*!:*LIMSEQ-I exI*[**where** *x*=*n*] *simp*: *indicator-def*)
qed(*use integrable* **in** *auto simp*: *set-integrable-def*)
thus ((λ*r*. (∫ *x*∈{*K*<..*r*}. *f x* ∂*M*)) ⟶ (∫ *x*∈{*K*<..}. *f x* ∂*M*)) *at-top*
and *set-integrable M* {*K*<..} *f*
using *integrable-monotone-convergence-real*[*OF 1 2 3*,*of l*] *l*
integral-monotone-convergence-real[*OF 1 2 3*,*of l*]
unfolding *set-lebesgue-integral-def set-borel-measurable-def set-integrable-def*
by *auto*
qed

corollary *interval-integral-inf2*:
fixes *f* :: *real* ⇒ *real*
assumes *f-nonneg*: ⋀*x*. *K* < *x* ⟹ *0* ≤ *f x*
and *integrable*: ⋀*r*. *K* < *r* ⟹ *set-integrable M* {*K*<..*r*} *f*
and *integrable2*:*set-integrable M* {*K*<..} *f*
shows ((λ*r*. (∫ *x*∈{*K*<..*r*}. *f x* ∂*M*)) ⟶ (∫ *x*∈{*K*<..}. *f x* ∂*M*)) *at-top*
proof(*rule interval-integral-inf*)
fix *r*
assume *r*:*K* < *r*
show (∫ *x*∈{*K*<..*r*}. *f x* ∂*M*) ≤ (∫ *x*∈{*K*<..}. *f x* ∂*M*)
using *integrable*[*OF r*] *integrable2 f-nonneg*
by(*auto intro*!: *Bochner-Integration.integral-mono*
simp: *set-integrable-def set-lebesgue-integral-def indicator-def*)
qed(*use assms* **in** *auto*)

1.6 Change of Variables w.r.t. Lebesgue Integration on a Set

lemma *lebesgue-integral-change-of-variables-real1*:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S*[*measurable*]: *S* ∈ *sets borel*
and *g*[*measurable*]:*g* ∈ *borel-measurable* (*restrict-space borel S*)
and *g'*[*measurable*]:*g'* ∈ *borel-measurable* (*restrict-space borel S*)
and *der-g*: ⋀*x*. *x* ∈ *S* ⟹ (*g has-field-derivative g' x*) (*at x within S*)
and *inj*: *inj-on g S*
and *integrable*: *set-integrable lborel* (*g* ‘ *S*) *f*

shows $(\int x\in g\ `\ S.\ f\ x\ \partial lborel) = (\int x\in S.\ |g'\ x| * f\ (g\ x)\ \partial lborel)$
proof −
have $*$:*set-borel-measurable borel S* $(\lambda x.\ f\ (g\ x))$
using *set-borel-measurable-comp*[*of S borel g f*]
set-integrable-set-borel-measurable[*OF integrable*] *g*
by (*simp add*: *borel-measurable-restrict-space-iff set-borel-measurable-def*)
hence [*measurable*]:$(\lambda x.\ indicat\text{-}real\ S\ x * (|g'\ x| * f\ (g\ x))) \in borel\text{-}measurable\ borel$
proof −
have *set-borel-measurable borel S* $(\lambda x.\ |g'\ x|)$
by(*auto simp*: *set-borel-measurable-iff-restrict-space*)
from *set-borel-measurable-times*[*OF this* $*$]
show *?thesis*
by(*simp add*: *set-borel-measurable-def*)
qed
have S':$S \in sets\ lebesgue$
using *S sets-lborel* **by** *blast*
have *h*:*f absolutely-integrable-on g* ` *S*
using *integrable* **by**(*auto simp*: *set-integrable-def integrable-completion*)
hence h':$(\lambda x.\ |g'\ x| * f\ (g\ x))$ *absolutely-integrable-on S*
using *absolutely-integrable-change-of-variables-real*[*OF S' der-g inj*]
by *force*
have $(\int x\in g\ `\ S.\ f\ x\ \partial lborel) = integral\ (g\ `\ S)\ f$
by(*intro set-borel-integral-eq-integral integrable*)
also have ... $= integral\ S\ (\lambda x.\ |g'\ x| * f\ (g\ x))$
using *has-absolute-integral-change-of-variables-1'*[*OF S' der-g inj*] *h'* **by** *blast*
also have ... $= (\int x\in S.\ |g'\ x| * f\ (g\ x)\ \partial lborel)$
by(*subst set-borel-integral-eq-integral*)
(*use h'* **in***auto simp*: *set-integrable-def integrable-completion*)
finally show *?thesis* **.**
qed

lemma *lebesgue-integral-change-of-variables-real2*:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S*[*measurable*]: $S \in sets\ borel$
and *f*:*set-borel-measurable borel* $(g\ `\ S)\ f$
and *der-g*: $\bigwedge x.\ x \in S \Longrightarrow (g\ has\text{-}field\text{-}derivative\ g'\ x)\ (at\ x\ within\ S)$
and *inj*: *inj-on g S*
and *integrable*:*set-integrable lborel S* $(\lambda x.\ |g'\ x| * f\ (g\ x))$
shows $(\int x\in g\ `\ S.\ f\ x\ \partial lborel) = (\int x\in S.\ |g'\ x| * f\ (g\ x)\ \partial lborel)$
proof −
have S':$S \in sets\ lebesgue$
using *S sets-lborel* **by** *blast*
have [*measurable*]:$(\lambda x.\ indicat\text{-}real\ (g\ `\ S)\ x * f\ x) \in borel\text{-}measurable\ borel$
using *f* **by**(*simp add*: *set-borel-measurable-def*)
have h':$(\lambda x.\ |g'\ x| * f\ (g\ x))$ *absolutely-integrable-on S*
using *integrable* **by**(*auto simp*: *set-integrable-def integrable-completion*)
hence *h*:*f absolutely-integrable-on g* ` *S*
using *absolutely-integrable-change-of-variables-real*[*OF S' der-g inj*]

by *force*
have *set-integrable lborel* (*g* ‘ *S*) *f*
using *h* **by**(*auto simp*: *set-integrable-def integrable-completion*)
hence (∫ *x*∈ *g* ‘ *S*. *f x* ∂*lborel*) = *integral* (*g* ‘ *S*) *f*
by(*intro set-borel-integral-eq-integral*)
also have ... = *integral S* (λ*x*. |*g*′ *x*| ∗ *f* (*g x*))
using *has-absolute-integral-change-of-variables-1*′[*OF S*′ *der-g inj*] *h*′ **by** *blast*
also have ... = (∫ *x*∈*S*. |*g*′ *x*| ∗ *f* (*g x*) ∂*lborel*)
by(*simp add*: *integrable set-borel-integral-eq-integral*)
finally show *?thesis* **.**
qed

lemma *lebesgue-integrable-change-of-variables-real*:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S*[*measurable*]: *S* ∈ *sets borel*
and *f*:*set-borel-measurable borel* (*g* ‘ *S*) *f*
and *g*[*measurable*]:*g* ∈ *borel-measurable* (*restrict-space borel S*)
and *g*′[*measurable*]:*g*′ ∈ *borel-measurable* (*restrict-space borel S*)
and *der-g*: ⋀*x*. *x* ∈ *S* ⟹ (*g has-field-derivative g*′ *x*) (*at x within S*)
and *inj*: *inj-on g S*
shows *set-integrable lborel* (*g* ‘ *S*) *f*
⟷ *set-integrable lborel S* (λ*x*. |*g*′ *x*| ∗ *f* (*g x*))
proof −
have ∗:*set-borel-measurable borel S* (λ*x*. *f* (*g x*))
using *set-borel-measurable-comp*[*of S borel g f*] *f g*
by (*simp add*: *borel-measurable-restrict-space-iff set-borel-measurable-def*)
have [*measurable*]:(λ*x*. *indicat-real S x* ∗ (|*g*′ *x*| ∗ *f* (*g x*))) ∈ *borel-measurable borel*
proof −
have *set-borel-measurable borel S* (λ*x*. |*g*′ *x*|)
by(*auto simp*: *set-borel-measurable-iff-restrict-space*)
from *set-borel-measurable-times*[*OF this* ∗]
show *?thesis*
by(*simp add*: *set-borel-measurable-def*)
qed
have [*measurable*]:(λ*x*. *indicat-real* (*g* ‘ *S*) *x* ∗ *f x*) ∈ *borel-measurable borel*
using *f* **by**(*simp add*: *set-borel-measurable-def*)
show *?thesis*
using *absolutely-integrable-change-of-variables-real*[*OF - der-g inj*,*of f*] *S*
by(*auto simp*: *set-integrable-def integrable-completion*)
qed

lemma *nn-integral-change-of-variables-real1*:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S*[*measurable*]: *S* ∈ *sets borel*
and *g*[*measurable*]:*g* ∈ *borel-measurable* (*restrict-space borel S*)
and *g*′[*measurable*]:*g*′ ∈ *borel-measurable* (*restrict-space borel S*)
and *der-g*: ⋀*x*. *x* ∈ *S* ⟹ (*g has-field-derivative g*′ *x*) (*at x within S*)
and *inj*: *inj-on g S*

and *integrable*: *set-integrable lborel* (*g* ‘ *S*) *f*
and *nonneg*:*AE x*∈*S in lborel. 0* ≤ *f* (*g x*) *AE x*∈*g* ‘ *S in lborel. 0* ≤ *f x*
shows (∫ +*x*∈ *g* ‘ *S. ennreal* (*f x*) *∂lborel*) = (∫ +*x*∈*S. ennreal* (|*g*′ *x*| ∗ *f* (*g x*)) *∂lborel*)
proof −
have *f*:*set-borel-measurable borel* (*g* ‘ *S*) *f*
using *integrable* **by**(*auto simp*: *set-integrable-def set-borel-measurable-def*)
have *integrable2*:*set-integrable lborel S* (λ*x*. |*g*′ *x*| ∗ *f* (*g x*))
using *lebesgue-integrable-change-of-variables-real assms f* **by** *blast*
have (∫ +*x*∈ *g* ‘ *S. ennreal* (*f x*) *∂lborel*)
= (∫ + *x. ennreal* (*indicat-real* (*g* ‘ *S*) *x* ∗ *f x*) *∂lborel*)
using *mult.commute*
by(*auto simp*: *indicator-mult-ennreal*[*symmetric*] *intro*!: *nn-integral-cong*)
also have ... = *ennreal* (∫ *x. indicat-real* (*g* ‘ *S*) *x* ∗ *f x ∂lborel*)
by(*rule nn-integral-eq-integral*)
(*use integrable nonneg* **in** *auto simp*: *set-integrable-def*)
also have ... = *ennreal* (∫ *x*∈ *g* ‘ *S. f x ∂lborel*)
by(*simp add*: *set-lebesgue-integral-def*)
also have ... = *ennreal* (∫ *x*∈*S*. |*g*′ *x*| ∗ *f* (*g x*) *∂lborel*)
by(*intro arg-cong*[**where** *f*=*ennreal*] *lebesgue-integral-change-of-variables-real1 assms*)
also have ... = *ennreal* (∫ *x. indicat-real S x* ∗ (|*g*′ *x*| ∗ *f* (*g x*)) *∂lborel*)
by(*simp add*: *set-lebesgue-integral-def*)
also have ... = (∫ + *x. ennreal* (*indicat-real S x* ∗ (|*g*′ *x*| ∗ *f* (*g x*))) *∂lborel*)
by(*rule nn-integral-eq-integral*[*symmetric*])
(*use integrable2 nonneg* **in** *auto simp*: *set-integrable-def*)
also have ... = (∫ +*x*∈*S. ennreal* (|*g*′ *x*| ∗ *f* (*g x*)) *∂lborel*)
using *mult.commute*
by(*auto simp*: *indicator-mult-ennreal*[*symmetric*] *intro*!: *nn-integral-cong*)
finally show *?thesis* **.**
qed

lemma *nn-integral-change-of-variables-real1* ′:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S* ∈ *sets borel*
and *g* ∈ *borel-measurable* (*restrict-space borel S*)
and *g*′ ∈ *borel-measurable* (*restrict-space borel S*)
and ⋀*x*. *x* ∈ *S* ⟹ (*g has-field-derivative g*′ *x*) (*at x within S*)
and *inj-on g S*
and *set-integrable lborel* (*g* ‘ *S*) *f*
and *nonneg*:⋀*x*. *x* ∈ *S* ⟹ *0* ≤ *f* (*g x*)
shows (∫ +*x*∈ *g* ‘ *S. ennreal* (*f x*) *∂lborel*) = (∫ +*x*∈*S. ennreal* (|*g*′ *x*| ∗ *f* (*g x*)) *∂lborel*)
by(*auto intro*!: *nn-integral-change-of-variables-real1 assms*)

lemma *nn-integral-change-of-variables-real2*:
fixes *f* :: *real* ⇒ *real* **and** *g* :: *real*⇒ *real*
assumes *S*[*measurable*]: *S* ∈ *sets borel*
and *f*:*set-borel-measurable borel* (*g* ‘ *S*) *f*

and g[*measurable*]:$g \in$ *borel-measurable* (*restrict-space borel* S)
and g'[*measurable*]:$g' \in$ *borel-measurable* (*restrict-space borel* S)
and *der-g*: $\bigwedge x.\ x \in S \Longrightarrow (g$ *has-field-derivative* $g'\ x)$ (*at* x *within* S)
and *inj*: *inj-on* $g\ S$
and *integrable*:*set-integrable lborel* $S\ (\lambda x.\ |g'\ x| * f\ (g\ x))$
and *nonneg*:*AE* $x \in S$ *in lborel*. $0 \leq f\ (g\ x)$ *AE* $x \in g$ ‘ S *in lborel*. $0 \leq f\ x$
shows $(\int^+ x \in g$ ‘ S. *ennreal* $(f\ x)\ \partial lborel) = (\int^+ x \in S$. *ennreal* $(|g'\ x| * f\ (g\ x))$ $\partial lborel)$
by(*auto intro*!: *nn-integral-change-of-variables-real1 assms*
lebesgue-integrable-change-of-variables-real[*THEN iffD2*])

lemma *nn-integral-change-of-variables-real2'*:
fixes $f :: real \Rightarrow real$ **and** $g :: real \Rightarrow real$
assumes S[*measurable*]: $S \in$ *sets borel*
and f:*set-borel-measurable borel* $(g$ ‘ $S)\ f$
and g[*measurable*]:$g \in$ *borel-measurable* (*restrict-space borel* S)
and g'[*measurable*]:$g' \in$ *borel-measurable* (*restrict-space borel* S)
and *der-g*: $\bigwedge x.\ x \in S \Longrightarrow (g$ *has-field-derivative* $g'\ x)$ (*at* x *within* S)
and *inj*: *inj-on* $g\ S$
and *integrable*:*set-integrable lborel* $S\ (\lambda x.\ |g'\ x| * f\ (g\ x))$
and *nonneg*:$\bigwedge x.\ x \in S \Longrightarrow 0 \leq f\ (g\ x)$
shows $(\int^+ x \in g$ ‘ S. *ennreal* $(f\ x)\ \partial lborel) = (\int^+ x \in S$. *ennreal* $(|g'\ x| * f\ (g\ x))$ $\partial lborel)$
by(*auto intro*!: *nn-integral-change-of-variables-real2 assms*)

1.7 Integral of Power Functions

lemma *set-integrable-inverse*:
fixes $a\ b\ c :: real$
assumes *a-pos*:$0 < a$
shows *set-integrable lborel* $\{a..b\}\ (\lambda x.\ c\ /\ x)$
using *a-pos*
by(*cases* $a \leq b$)
(*auto intro*!: *set-integrable-bounded*[**where** K=(*abs* c) / a] *divide-mono*
simp: *lborel-eq-real set-borel-measurable-def emeasure-interval-measure-Icc*)

lemma *set-integrable-inverse'*:
fixes $a\ b\ c :: real$
assumes $0 < a$
shows *set-integrable lborel* $\{a<..b\}\ (\lambda x.\ c\ /\ x)$
by(*auto intro*!: *set-integrable-subset*[*OF set-integrable-inverse*] *assms*)

corollary *set-integral-inverse*:
fixes $a\ b\ c :: real$
assumes *a-pos*:$0 < a$
and *a-le-b*:$a \leq b$
shows $(\int x \in \{a<..b\}.\ c\ /\ x\ \partial lborel) = c * (ln\ b - ln\ a)$
proof −
have *ln-im*:*ln* ‘ $\{a<..b\} = \{ln\ a<..ln\ b\}$

```
    using assms
    by(intro image-mono-on-continuous-on-Ioc strict-mono-on-imp-inj-on
        monotone-onI ln-strict-mono continuous-on-ln)
      auto
  have (∫ x∈{a<..b}. c / x ∂lborel) = (∫ x∈{a<..b}. |1 / x| * c ∂lborel)
    using assms by(auto intro!: Bochner-Integration.integral-cong
        simp: set-lebesgue-integral-def indicator-def)
  also have ... = (∫ x∈ln ` {a<..b}. c ∂lborel)
    using a-pos a-le-b
    by(intro lebesgue-integral-change-of-variables-real2[symmetric]
        has-field-derivative-at-within[OF DERIV-ln-divide]
        strict-mono-on-imp-inj-on monotone-onI ln-strict-mono)
      (fastforce intro!: integrableI-bounded-set[where A={a<..b} and B=|c| / a]
        divide-mono simp: set-borel-measurable-def set-integrable-def ln-im)+
  also have ... = c * (ln b − ln a)
    by(simp add: interval-integral-Ioc[symmetric] ln-mono[OF a-le-b a-pos] ln-im)
  finally show ?thesis .
qed

lemma set-integrable-powr-Icc:
  fixes c :: real
  assumes c:0 < c
  shows set-integrable lborel {c..d} (λx. x powr a)
proof(cases c ≤ d)
  assume cd:c ≤ d
  thus set-integrable lborel {c..d} (λx. x powr a)
    using c
    by(cases a = − 1)
      (auto intro!: borel-integrable-atLeastAtMost isCont-divide continuous-powr
        simp del: times-divide-eq-left simp add: set-integrable-def mult.commute)
qed simp

lemma set-integral-powr-Icc:
  fixes c :: real
  assumes a:− 1 ≠ a
    and c:0 < c
    and cd:c ≤ d
  shows (∫ x∈{c..d}. x powr a ∂lborel) = d powr (a + 1) / (a + 1) − c powr (a
+ 1) / (a + 1)
proof −
  have (∫ x∈{c..d}. x powr a ∂lborel) = (∫ x. x powr a * indicat-real {c..d} x
∂lborel)
    by(simp add: set-lebesgue-integral-def mult.commute)
  also have ... = d powr (a + 1) / (a + 1) − c powr (a + 1) / (a + 1)
  proof(rule integral-FTC-Icc-nonneg)
    fix x
    assume x ∈ {c..d}
    then show ((λy. (y powr (a + 1)) / (a + 1)) has-real-derivative x powr a) (at
x)
```

```
    using has-real-derivative-powr[of x a + 1] c a
      DERIV-cmult[OF has-real-derivative-powr[of x a + 1],where c=1 / (a +
1)]
    by auto
  qed(use cd in auto)
  finally show (∫ x∈{c..d}. x powr a ∂lborel)
              = d powr (a + 1) / (a + 1) − c powr (a + 1) / (a + 1) .
qed

corollary set-integrable-powr-Ioc:
  fixes c :: real
  assumes c:0 < c
  shows set-integrable lborel {c<..d} (λx. x powr a)
  by(auto intro!: set-integrable-subset[OF set-integrable-powr-Icc[OF c]])

corollary set-integral-powr-Ioc:
  fixes c :: real
  shows a ≠ − 1 ⟹ 0 < c ⟹ c ≤ d ⟹
   (∫ x∈{c<..d}. x powr a ∂lborel) = d powr (a + 1) / (a + 1) − c powr (a + 1)
/ (a + 1)
  by(auto intro!: set-integrable-subset[OF set-integrable-powr-Icc[of a c]]
    simp: interval-integral-Ioc[symmetric] interval-integral-Icc set-integral-powr-Icc)

lemma
  fixes c :: real
  assumes a:− 1 < a
  shows set-integrable-powr-from-0:set-integrable lborel {0<..c} (λx. x powr a)
    and set-integral-powr-from-0:0 ≤ c ⟹
   (∫ x∈{0<..c}. x powr a ∂lborel) = c powr (a + 1) / (a + 1)
proof −
  have ∗:set-integrable lborel {0<..c} (λx. x powr a)
        (∫ x∈{0<..c}. x powr a ∂lborel) = c powr (a + 1) / (a + 1) if c:0 < c
  proof −
    obtain N where N:0 < N 1 / real N < c
      using c by (metis less-Suc-eq-0-disj nat-approx-posE)
    hence cn0: ⋀n. 1 / real (n + N) < c
      by (metis basic-trans-rules(21) inverse-of-nat-le le-add2 not-gr0)
    have cnle0:⋀n. 1 / real (n + N) > 0
      using N by simp
    have dec:decseq (λn. 1 / real (n + N))
      using N by (metis (mono-tags, lifting) add-le-mono1
                    antimonoI inverse-of-nat-le not-add-less2)
    have 1:integrable lborel (λx. indicator {1 / real (n + N)..c} x ∗ x powr a) for
n
      using set-integrable-powr-Icc[of 1 / real (n + N) c] N
      by(auto simp: set-integrable-def)
    have 2:AE x in lborel. incseq (λn. indicator {1 / real (n + N)..c} x ∗ x powr
a)
      using decseq-SucD[OF dec]
```

by(*intro AE-I2 incseq-SucI mult-mono order.refl indicator-pos-le powr-ge-zero indicator-leI*)
(*meson atLeastAtMost-iff order.trans*)
have *3*:*AE x in lborel.* (λ*n. indicator* {*1 / real* (*n* + *N*)..*c*} *x* ∗ *x powr a*)
⟶ *indicator* {*0*<..*c*} *x* ∗ *x powr a*
proof(*intro AE-I2 tendsto-mult-right*)
fix *x*
consider *0* < *x x* ≤ *c* | *x* ≤ *0* | *c* < *x*
by *linarith*
then show (λ*n. indicator* {*1 / real* (*n* + *N*)..*c*} *x*) ⟶ *indicator* {*0*<..*c*} *x*
proof *cases*
assume *x*:*0* < *x x* ≤ *c*
then obtain *N'* **where** *1 / real* (*N'* + *N*) ≤ *x*
by (*smt* (*verit*) *inverse-of-nat-le le-add1 less-Suc-eq nat-approx-posE not-gr0*)
hence ⋀*n. N'* ≤ *n* ⟹ *1 / real* (*n* + *N*) ≤ *x*
using *N*
by (*metis add-mono-thms-linordered-semiring*(*3*) *inverse-of-nat-le not-add-less2 order.trans*)
with *x* **have** ∀ *F n in sequentially. indicator* {*1 / real* (*n* + *N*)..*c*} *x* = *1*
by(*auto intro*!: *eventually-sequentiallyI*[**where** *c*=*N'*] *simp*: *indicator-def*)
thus *?thesis*
using *x atLeastAtMost-iff greaterThanAtMost-iff tendsto-eventually* **by** *fastforce*
next
assume *x*:*x* ≤ *0*
then have ∗:*indicator* {*1 / real* (*n* + *N*)..*c*} *x* = *0* **for** *n*
using *cnle0*[*of n*] *c*
by (*meson atLeastAtMost-iff indicator-eq-0-iff not-less order.strict-trans1*)
thus *?thesis*
using *x* **unfolding** ∗ **by** *force*
qed *simp*
qed
have *4*:(λ*n.* (∫ *x. indicator* {*1 / real* (*n* + *N*)..*c*} *x* ∗ *x powr a* ∂*lborel*))
⟶ *c powr* (*a* + *1*) / (*a* + *1*)
proof −
have (∫ *x. indicator* {*1 / real* (*n* + *N*)..*c*} *x* ∗ *x powr a* ∂*lborel*)
= *c powr* (*a* + *1*) / (*a* + *1*) − (*1 / real* (*n* + *N*)) *powr* (*a* + *1*) / (*a* + *1*) **for** *n*
using *cn0 a cnle0*
by(*subst set-integral-powr-Icc*[*symmetric*])
(*auto simp*: *set-lebesgue-integral-def less-imp-le*)
moreover have (λ*n. c powr* (*a* + *1*) / (*a* + *1*) − (*1 / real* (*n* + *N*)) *powr* (*a* + *1*) / (*a* + *1*))
⟶ *c powr* (*a* + *1*) / (*a* + *1*)
by(*rule LIM-zero-cancel*)
(*use a* **in** *auto simp*: *tendsto-minus-cancel-left*[*symmetric*]
simp del: *of-nat-add intro*!: *lim-const-over-n tendsto-divide-zero*

```
                                    tendsto-zero-powrI LIMSEQ-ignore-initial-segment[where
a=0])
      ultimately show ?thesis
        by presburger
    qed
    show set-integrable lborel {0<..c} (λx. x powr a)
      and (∫ x∈{0<..c}. x powr a ∂lborel) = c powr (a + 1) / (a + 1)
      using integral-monotone-convergence[OF 1 2 3 4]
        integrable-monotone-convergence[OF 1 2 3 4]
      unfolding set-integrable-def set-lebesgue-integral-def
      by fastforce+
  qed
  show set-integrable lborel {0<..c} (λx. x powr a)
    by(cases 0 < c) (use ∗ in auto)
  show 0 ≤ c ⟹
   (∫ x∈{0<..c}. x powr a ∂lborel) = c powr (a + 1) / (a + 1)
    by(cases 0 < c) (use ∗ in auto)
qed

lemma
  fixes c :: real
  assumes a:a < − 1
    and c:0 < c
  shows set-integrable-powr-inf:set-integrable lborel {c<..} (λx. x powr a)
    and set-integral-powr-inf:(∫ x∈{c<..}. x powr a ∂lborel) = − (c powr (a + 1)
/ (a + 1))
proof −
  show set-integrable lborel {c<..} (λx. x powr a)
    using a c
    by(intro interval-integral-inf-integrable[where G=− (c powr (a + 1) / (a +
1))]
        set-integrable-powr-Ioc)
      (auto simp: set-integral-powr-Ioc[of a c] intro!: divide-nonneg-neg)
  have ((λr. (∫ x∈{c<..r}. x powr a ∂lborel)) ⟶ (∫ x∈{c<..}. x powr a ∂lborel))
at-top
    using a c
    by(intro interval-integral-inf[where G=− (c powr (a + 1) / (a + 1))]
        set-integrable-powr-Ioc)
      (auto simp: set-integral-powr-Ioc[of a c] intro!: divide-nonneg-neg)
  moreover have ((λr. (∫ x∈{c<..r}. x powr a ∂lborel))
                  ⟶ − (c powr (a + 1) / (a + 1))) at-top
  proof −
    have ∀F r in at-top. (∫ x∈{c<..r}. x powr a ∂lborel)
                        = r powr (a + 1) / (a + 1) − c powr (a + 1) / (a + 1)
      unfolding eventually-at-top-linorder
      using a c by(auto simp add: set-integral-powr-Ioc intro!: exI[where x=c])
    moreover have ((λr::real. r powr (a + 1) / (a + 1) − c powr (a + 1) / (a
+ 1))
                    ⟶ − (c powr (a + 1) / (a + 1))) at-top
```

using *a*
by(*intro LIM-zero-cancel*[**where** *l*=− (*c powr* (*a* + *1*) / (*a* + *1*))])
(*auto intro*!: *tendsto-divide-zero tendsto-neg-powr filterlim-ident*)
ultimately show *?thesis*
by(*simp cong*: *tendsto-cong*)
qed
ultimately show (∫ *x*∈{*c*<..}. *x powr a* ∂*lborel*) = − (*c powr* (*a* + *1*) / (*a* + *1*))
using *tendsto-unique trivial-limit-at-top-linorder* **by** *blast*
qed

1.8 Uniform Convergence

corollary *uniform-limit-uminus-iff*:
fixes *f* :: - ⇒ - :: *real-normed-vector*
shows *uniform-limit A* (λ*i x*. − *fi i x*) (λ*x*. − *f x*) *F*
⟷ *uniform-limit A fi f F*
using *uniform-limit-uminus* **by** *fastforce*

lemma *uniform-lim-mult-bdd1*:
fixes *f* :: - ⇒ - :: *real-normed-field*
assumes *ul*:*uniform-limit A fi f F*
and *bdd*:*bounded* (*g* ‘ *A*)
shows *uniform-limit A* (λ*i x*. *g x* ∗ *fi i x*) (λ*x*. *g x* ∗ *f x*) *F*
unfolding *uniform-limit-iff*
proof *safe*
fix *e* :: *real*
assume *e*:*0* < *e*
obtain *K* **where** *K*:⋀*x*. *x* ∈ *A* ⟹ *norm* (*g x*) ≤ *K 0* < *K*
by (*metis bdd bounded-pos image-eqI*)
hence *eK*:*0* < *e* / *K*
using *e* **by** *simp*
show ∀ *F i in F*. ∀ *x*∈*A*. *dist* (*g x* ∗ *fi i x*) (*g x* ∗ *f x*) < *e*
by(*intro eventually-mono*[*OF uniform-limitD*[*OF ul eK*]] *ballI*)
(*smt* (*verit*, *best*) *K*(*1*) *dist-mult-left e frac-le*
nonzero-mult-div-cancel-left norm-ge-zero zero-less-mult-iff)
qed

lemma *uconvergence-on-bdd-times2*:
fixes *f* :: - ⇒ - :: *real-normed-field*
assumes *uniform-limit A fi f F*
and *bounded* (*g* ‘ *A*)
shows *uniform-limit A* (λ*i x*. *fi i x* ∗ *g x*) (λ*x*. *f x* ∗ *g x*) *F*
using *uniform-lim-mult-bdd1*[*OF assms*] **by**(*simp add*: *mult.commute*)

lemma *uniform-limit-eventually-in*:
assumes *lim*: *uniform-limit A g g′ F*
and *e*:*0* < *e*
and *d*: ⋀*x y*. *x* ∈ *A* ⟹ *dist* (*g′ x*) *y* < *e* ⟹ *y* ∈ *B*

shows $\forall_F\ t\ in\ F.\ \forall y{\in}A.\ g\ t\ y \in B$
by(*rule eventually-mono*[*OF uniform-limitD*[*OF lim e*]])
(*use d* **in** *simp add*: *Met-TC.commute*)

lemma *mono-on-UCT*:
fixes $Ui :: - \Rightarrow real \Rightarrow real$
assumes *ab*:$a \leq b$
and *mono*:$\forall_F\ i\ in\ F.\ mono\text{-}on\ \{a..b\}\ (Ui\ i)$ **and** *monoU*:*mono-on* $\{a..b\}\ U$
and U:*continuous-on* $\{a..b\}\ U$
and *lim*: $\bigwedge x.\ a \leq x \Longrightarrow x \leq b \Longrightarrow ((\lambda i.\ Ui\ i\ x) \longrightarrow U\ x)\ F$
shows *uniform-limit* $\{a..b\}\ Ui\ U\ F$
proof(*intro uniform-limitI*)
fix $e :: real$
assume *e*:$0 < e$
hence e':$0 < e\ /\ 2$
by *simp*
then obtain d' **where** d': $d' > 0$
$\bigwedge x\ y.\ a \leq x \Longrightarrow x \leq b \Longrightarrow a \leq y \Longrightarrow y \leq b$
$\Longrightarrow |x - y| < d' \Longrightarrow |U\ x - U\ y| < e\ /\ 2$
using *compact-uniformly-continuous*[*OF U compact-interval*[*of a b*]]
unfolding *uniformly-continuous-on-def*
by (*metis atLeastAtMost-iff dist-real-def*)
define d **where** $d \equiv min\ d'\ (1\ /\ 2)$
have *d-less*:$d < 1$ **and** *d-pos*:$0 < d$
and d:$\bigwedge x\ y.\ a \leq x \Longrightarrow x \leq b \Longrightarrow a \leq y \Longrightarrow y \leq b$
$\Longrightarrow |x - y| < d \Longrightarrow |U\ x - U\ y| < e\ /\ 2$
using d' **by**(*auto simp*: *d-def*)
have $*$:$\{a..b\} \subseteq (\bigcup\ ((\lambda x.\ \{x - d\ /\ 4 <..< x + d\ /\ 4\})\ `\ \{a..b\}))$
using *d-pos* **by** *fastforce*
then obtain K **where** *finK*:*finite K* **and** *K-sub*:$K \subseteq \{a..b\}$
and K: $\{a..b\} \subseteq (\bigcup x{\in}K.\ \{x - d\ /\ 4 <..< x + d\ /\ 4\})$
using *compactE-image*[*OF compact-interval - **] **by** *force*
hence K': $(\bigcup x{\in}K.\ \{x - d\ /\ 4 <..< x + d\ /\ 4\} \cap \{a..b\}) = \{a..b\}$
(**is** $(\bigcup x{\in}K.\ ?A\ x) = \{a..b\}$)
by *blast*
have *K-ne*:$K \neq \{\}$
using *K ab* **by** *fastforce*
have $\bigwedge x.\ x \in K \Longrightarrow a \leq max\ (x - d\ /\ 4)\ a\ \bigwedge x.\ x \in K \Longrightarrow max\ (x - d\ /\ 4)\ a \leq b$
$\bigwedge x.\ x \in K \Longrightarrow a \leq (min\ (x + d\ /\ 4)\ b)\ \bigwedge x.\ x \in K \Longrightarrow (min\ (x + d\ /\ 4)\ b) \leq b$
using *ab K-sub d-pos* **by** *auto*
hence $\forall x{\in}K.\ \forall_F\ i\ in\ F.\ |Ui\ i\ (max\ (x - d\ /\ 4)\ a) - U\ (max\ (x - d\ /\ 4)\ a)| < e\ /\ 2$
and $\forall x{\in}K.\ \forall_F\ i\ in\ F.\ |Ui\ i\ (min\ (x + d\ /\ 4)\ b) - U\ (min\ (x + d\ /\ 4)\ b)| < e\ /\ 2$
using e' *lim*
unfolding *tendsto-iff dist-norm real-norm-def* **by** *fast+*
hence $\forall_F\ i\ in\ F.\ \forall x{\in}K.\ |Ui\ i\ (max\ (x - d\ /\ 4)\ a) - U\ (max\ (x - d\ /\ 4)\ a)|$

$< e / 2$
and $\forall_F\ i\ in\ F.\ \forall x{\in}K.\ |Ui\ i\ (min\ (x + d / 4)\ b) - U\ (min\ (x + d / 4)\ b)|$
$< e / 2$
using *finK* **by**(*auto simp*: *eventually-ball-finite*)
hence $*{:}\forall_F\ i\ in\ F.\ \forall x{\in}K.\ |Ui\ i\ (max\ (x - d / 4)\ a) - U\ (max\ (x - d / 4)\ a)| < e / 2$
$\wedge\ |Ui\ i\ (min\ (x + d / 4)\ b) - U\ (min\ (x + d / 4)\ b)| < e / 2$
using *eventually-elim2* **by** *fastforce*
show $\forall_F\ i\ in\ F.\ \forall x{\in}\{a..b\}.\ dist\ (Ui\ i\ x)\ (U\ x) < e$
proof(*safe intro*!: *eventually-mono*[**where** $Q{=}\lambda i.\ \forall x{\in}\{a..b\}.\ dist\ (Ui\ i\ x)\ (U\ x) < e$,
OF eventually-conj[*OF* $*$ *mono*]])
fix i **and** x :: *real*
assume $i{:}\forall y{\in}K.\ |Ui\ i\ (max\ (y - d / 4)\ a) - U\ (max\ (y - d / 4)\ a)| < e / 2$
$\wedge\ |Ui\ i\ (min\ (y + d / 4)\ b) - U\ (min\ (y + d / 4)\ b)| < e / 2$
and $x{:}x \in \{a..b\}$ **and** $monoUi{:}mono\text{-}on\ \{a..b\}\ (Ui\ i)$
then obtain y **where** $y{:}y \in K\ x \in \{y - d / 4 <..< y + d / 4\} \cap \{a..b\}$
using K' **by** *blast*
have $Ui\ i\ x - U\ x \le Ui\ i\ (min\ (y + d / 4)\ b) - U\ (max\ (y - d / 4)\ a)$
using *mono-onD*[*OF monoUi*] *y mono-onD*[*OF monoU*]
by(*auto intro*!: *diff-mono*)
also have $... \le |Ui\ i\ (min\ (y + d / 4)\ b) - U\ (min\ (y + d / 4)\ b)| + |U\ (min\ (y + d / 4)\ b) - U\ (max\ (y - d / 4)\ a)|$
by *simp*
also have $... < e / 2 + e / 2$
using *d*[*of min* $(y + d / 4)\ b$ *max* $(y - d / 4)\ a$] *y i K-sub d-less d-pos*
by(*intro add-strict-mono*) *auto*
also have $... = e$
by *simp*
finally have *1*:$Ui\ i\ x - U\ x < e$ **.**
have $- e = - e / 2 - e / 2$
by *simp*
also have $... < - |Ui\ i\ (max\ (y - d / 4)\ a) - U\ (max\ (y - d / 4)\ a)| - |U\ (max\ (y - d / 4)\ a) - U\ (min\ (y + d / 4)\ b)|$
using *y i K-sub d-less d-pos*
by(*intro diff-strict-mono d*) *auto*
also have $... \le Ui\ i\ (max\ (y - d / 4)\ a) - U\ (min\ (y + d / 4)\ b)$
by *linarith*
also have $... \le Ui\ i\ x - U\ x$
using *mono-onD*[*OF monoUi*] *y mono-onD*[*OF monoU*]
by(*auto intro*!: *diff-mono*)
finally have *2*:$- e < Ui\ i\ x - U\ x$ **.**
show $dist\ (Ui\ i\ x)\ (U\ x) < e$
using *1 2* **by** (*simp add*: *dist-real-def*)
qed
qed

lemma *mono-on-UCT-atLeast*:

```
  fixes f :: 'a ⇒ real ⇒ real and g :: real ⇒ real and F :: 'a filter
  assumes mono: ∀⇩F i in F. mono-on {a..} (f i)
    and bdd: ∀⇩F i in F. ∀ x≥a. f i x ≤ B
    and gmono: mono-on {a..} g
    and gcont: continuous-on {a..} g
    and glim: (g ⟶⟶ B) at-top
    and lim: ⋀x. a ≤ x ⟹ ((λi. f i x) ⟶⟶ g x) F
  shows uniform-limit {a..} f g F
proof(safe intro!: uniform-limitI)
  fix e :: real
  assume e[arith]:0 < e
  have gB:g x ≤ B if x:a ≤ x for x
    using mono-onD[OF gmono] x
    by(auto intro!: tendsto-lowerbound[OF glim] eventually-at-top-linorderI[where
c=x])
  obtain b' where b':⋀x. b' ≤ x ⟹ |g x − B| < e / 2
    using tendstoD[OF glim half-gt-zero[OF e]]
    unfolding dist-real-def eventually-at-top-linorder
    by blast
  define b where b ≡ max b' a
  have b:⋀x. b ≤ x ⟹ B − g x < e / 2 and ab:a ≤ b
    using gB b' by(fastforce simp: b-def simp del: less-divide-eq-numeral1)+
  have e / 2 − (B − g b) > 0
    using b[OF order.refl] by simp
  from tendstoD[OF lim[OF ab] this]
  have ∀⇩F i in F. |f i b − g b| < e / 2 − (B − g b)
    by(auto simp: dist-real-def)
  hence ∗:∀⇩F i in F. B − f i b < e / 2
    by eventually-elim argo
  have 1:∀⇩F i in F. ∀ x∈{b..}. dist (f i x) (g x) < e
  proof(safe intro!: eventually-mono[OF eventually-conj[OF ∗ eventually-conj[OF
bdd mono]]])
    fix i x
    assume h:B − f i b < e / 2
      and mono:mono-on {a..} (f i)
      and bx:b ≤ x and bdd:∀ y≥a. f i y ≤ B
    have dist (f i x) (g x) ≤ |B − g x| + |B − f i x|
      by(simp add: dist-real-def)
    also have ... = B − g x + (B − f i x)
      using bdd bx gB ab by simp
    also have ... ≤ B − g x + (B − f i b)
      using mono-onD[OF mono] ab bx by simp
    finally show dist (f i x) (g x) < e
      using b[OF bx] h by linarith
  qed
  have uniform-limit {a..b} f g F
    by(intro mono-on-UCT[OF ab eventually-mp[OF - mono]
        mono-on-subset[OF gmono] continuous-on-subset[OF gcont] lim] eventual-
lyI)
```

(*auto simp*: *mono-on-subset*)
hence *2*:$\forall_F$ *i in F*. $\forall x{\in}\{a..b\}$. *dist* (*f i x*) (*g x*) < *e*
using *uniform-limitD e* **by** *blast*
show $\forall_F$ *i in F*. $\forall x{\in}\{a..\}$. *dist* (*f i x*) (*g x*) < *e*
using *1 2* **by** *eventually-elim auto*
qed

lemma *antimono-on-UCT*:
fixes *Ui* :: - ⇒ *real* ⇒ *real*
assumes *ab*:*a* ≤ *b*
and *mono*:$\forall_F$ *i in F*. *antimono-on* {*a..b*} (*Ui i*) **and** *monoU*:*antimono-on* {*a..b*} *U*
and *U*:*continuous-on* {*a..b*} *U*
and *lim*: ⋀*x*. *a* ≤ *x* ⟹ *x* ≤ *b* ⟹ ((λ*i*. *Ui i x*) ⟶ *U x*) *F*
shows *uniform-limit* {*a..b*} *Ui U F*
using *assms mono-on-UCT*[**where** *Ui*=λ*i x*. − *Ui i x* **and** *U*=λ*x*. − *U x*]
by(*auto simp*: *antimono-on-iff-uminus-mono-on uniform-limit-uminus-iff*
tendsto-minus-cancel continuous-on-minus)

end

2 Regularly Varying Functions

theory *Regular-Variation*
imports
Lemmas-Regular-Variation
Cauchy-Functional-Equation.Cauchy-Functional-Equation
begin

We refer to Chapter 1 of the book by Bingham et al. [1], and Chapter 2 of the book by Resnick [3].

2.1 Definition and Basic Properties

2.1.1 Regularly Varying Functions

definition *regularly-varying* :: (*real* ⇒ *real*) ⇒ *real* ⇒ *bool* **where**
regularly-varying U ϱ ⟷
U ∈ *borel-measurable* (*restrict-space borel* {*0*<..}) ∧
U ‘ {*0*<..} ⊆ {*0*<..} ∧
(∀ *x*>*0*. ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr ϱ*) *at-top*)

lemma *regularly-varyingI*:
assumes *U* ∈ *borel-measurable* (*restrict-space borel* {*0*<..})
and ⋀*r*. *0* < *r* ⟹ *0* < *U r*
and ⋀*x*. *0* < *x* ⟹ ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr ϱ*) *at-top*
shows *regularly-varying U ϱ*
using *assms* **by**(*auto simp*: *regularly-varying-def*)

lemma *regularly-varyingI''*:
assumes *set-borel-measurable borel* {*0*<..} *U*
and ⋀*r*. *0* < *r* ⟹ *0* < *U r*
and ⋀*x*. *0* < *x* ⟹ ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr* ϱ) *at-top*
shows *regularly-varying U* ϱ
using *assms* **by**(*auto intro*!: *regularly-varyingI*
simp: *set-borel-measurable-iff-restrict-space*)

lemma *regularly-varyingI-total*:
assumes *U* ∈ *borel-measurable borel*
and ⋀*r*. *0* < *r* ⟹ *0* < *U r*
and ⋀*x*. *0* < *x* ⟹ ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr* ϱ) *at-top*
shows *regularly-varying U* ϱ
using *assms* **by**(*auto intro*!: *regularly-varyingI measurable-restrict-space1*)

lemma
assumes *regularly-varying U* ϱ
shows *regularly-varying-meas'*:*set-borel-measurable borel* {*0*<..} *U*
and *regularly-varying-meas*:*U* ∈ *borel-measurable* (*restrict-space borel* {*0*<..})
and *regularly-varying-pos*:⋀*r*. *0* < *r* ⟹ *0* < *U r*
and *regularly-varying-lim*:⋀*x*. *x* > *0* ⟹ ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr* ϱ) *at-top*
using *assms*
by(*auto simp*: *regularly-varying-def set-borel-measurable-iff-restrict-space*)

lemma *regularly-varying-meas-log-exp*:
assumes *regularly-varying U* ϱ
shows (λ*t*. *ln* (*U* (*exp t*))) ∈ *borel-measurable borel*
by(*intro borel-measurable-ln measurable-compose*[*OF - regularly-varying-meas*[*OF assms*]]
measurable-restrict-space2) *auto*

lemma
assumes *regularly-varying U* ϱ
shows *regularly-varying-meas-o*:
⋀*a*. *0* ≤ *a* ⟹ (λ*x*. *indicat-real* {*a*<..} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-c*:
⋀*a*. *0* < *a* ⟹ (λ*x*. *indicat-real* {*a*..} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-Ioo*:
⋀*a b*. *0* ≤ *a* ⟹ (λ*x*. *indicat-real* {*a*<..<*b*} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-Ico*:
⋀*a b*. *0* < *a* ⟹ (λ*x*. *indicat-real* {*a*..<*b*} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-Ioc*:
⋀*a b*. *0* ≤ *a* ⟹ (λ*x*. *indicat-real* {*a*<..*b*} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-Icc*:
⋀*a b*. *0* < *a* ⟹ (λ*x*. *indicat-real* {*a*..*b*} *x* ∗ *U x*) ∈ *borel-measurable borel*
and *regularly-varying-meas-o-ennreal*:
⋀*a*. *0* ≤ *a* ⟹ (λ*x*. *ennreal* (*U x*) ∗ *indicator* {*a*<..} *x*) ∈ *borel-measurable borel*

and *regularly-varying-meas-c-ennreal*:
$\bigwedge a.\ 0 < a \Longrightarrow (\lambda x.\ \ ennreal\ (U\ x) * indicator\ \{a..\}\ x) \in borel\text{-}measurable\ borel$
and *regularly-varying-meas-Ioo-ennreal*:
$\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a<..<b\}\ x) \in borel\text{-}measurable\ borel$
and *regularly-varying-meas-Ico-ennreal*:
$\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a..<b\}\ x) \in borel\text{-}measurable\ borel$
and *regularly-varying-meas-Ioc-ennreal*:
$\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a<..b\}\ x) \in borel\text{-}measurable\ borel$
and *regularly-varying-meas-Icc-ennreal*:
$\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a..b\}\ x) \in borel\text{-}measurable\ borel$
proof −
have $*$:$(\lambda x.\ indicat\text{-}real\ A\ x * U\ x) \in borel\text{-}measurable\ borel$
if A:$A \in sets\ borel\ A \subseteq \{0<..\}$ **for** A
proof −
have $(\lambda x.\ indicat\text{-}real\ A\ x * U\ x) = (\lambda x.\ indicat\text{-}real\ \{0<..\}\ x *\ \ U\ x * indicat\text{-}real\ A\ x)$
by *standard* (*use* A **in** *auto simp*: *indicator-eq-0-iff indicator-eq-1-iff*)
moreover have $... \in borel\text{-}measurable\ borel$
using *regularly-varying-meas'*[*OF assms*] A
by(*auto simp*: *set-borel-measurable-def*)
ultimately show *?thesis*
by *argo*
qed
have $**$:$(\lambda x.\ ennreal\ (U\ x) * indicator\ A\ x) \in borel\text{-}measurable\ borel$
if A:$A \in sets\ borel\ A \subseteq \{0<..\}$ **for** A
using *measurable-compose*[*OF* $*$[*OF* A] *measurable-ennreal*]
by(*simp add*: *mult.commute ennreal-mult''*[*OF indicator-pos-le*] *ennreal-indicator*)
show$\bigwedge a.\ 0 \leq a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a<..\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a.\ 0 < a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a..\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a<..<b\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a..<b\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a<..b\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ indicat\text{-}real\ \{a..b\}\ x * U\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a.\ 0 \leq a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a<..\}\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a.\ 0 < a \Longrightarrow (\lambda x.\ \ ennreal\ (U\ x) * indicator\ \{a..\}\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a<..<b\}\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a..<b\}\ x) \in borel\text{-}measurable\ borel$
and $\bigwedge a\ b.\ 0 \leq a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a<..b\}\ x) \in borel\text{-}measurable$

borel
  and $\bigwedge a\ b.\ 0 < a \Longrightarrow (\lambda x.\ ennreal\ (U\ x) * indicator\ \{a..b\}\ x) \in borel\text{-}measurable$
borel
  by(*auto intro*!: $*$ $**$)
qed

lemma *regularly-varying-asymp-equiv*:
  assumes *regularly-varying U ϱ*
    and *x*:$0 < x$
  shows $(\lambda t.\ U\ (t * x)) \sim[at\text{-}top]\ (\lambda t.\ x\ powr\ \varrho * U\ t)$
  using *x* **by**(*auto intro*!: *asymp-equivI'-const regularly-varying-lim assms*)

lemma *regularly-varying-def2*:
regularly-varying U ϱ $\longleftrightarrow$
  $U \in borel\text{-}measurable\ (restrict\text{-}space\ borel\ \{0<..\}) \wedge$
  $U\ `\ \{0<..\} \subseteq \{0<..\} \wedge$
  $(\forall\, x{>}0.\ (\lambda t.\ U\ (t * x)) \sim[at\text{-}top]\ (\lambda t.\ x\ powr\ \varrho * U\ t))$
proof *safe*
  assume [*simp*]:$U \in borel\text{-}measurable$
        $(restrict\text{-}space\ borel\ \{0<..\})$
    and *h''*:$U\ `\ \{0<..\} \subseteq \{0<..\}$
    and *h*:$\forall\, x{>}0.\ (\lambda t.\ U\ (t * x)) \sim[at\text{-}top]\ (\lambda t.\ x\ powr\ \varrho * U\ t)$
  hence *h'*:$\bigwedge x\ y.\ x > 0 \Longrightarrow y > 0 \Longrightarrow \neg\ (U\ (y * x) = 0 \wedge x\ powr\ \varrho * U\ y = 0)$
    by *fastforce*
  have $((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow x\ powr\ \varrho)\ at\text{-}top$ **if** *x*:$x > 0$ **for** *x*
  proof −
    have $(\lambda t.\ U\ (t * x)) \sim[at\text{-}top]\ (\lambda t.\ x\ powr\ \varrho * U\ t)$
      using *x h* **by** *blast*
    hence $((\lambda y.\ U\ (y * x)\ /\ (x\ powr\ \varrho * U\ y)) \longrightarrow 1)\ at\text{-}top$
    using *x h'*[*of x*] **by**(*fastforce intro*!: *asymp-equivD-strong simp*: *eventually-at-top-dense*)
    from *tendsto-mult-right*[**where** *c*=*x powr ϱ*,*OF this*]
    show *?thesis*
      using *x* **by** *simp*
  qed
  thus *regularly-varying U ϱ*
    using *h''* **by**(*auto intro*!: *regularly-varyingI*)
qed(*auto simp*: *regularly-varying-asymp-equiv regularly-varying-meas regularly-varying-pos*)

lemma *regularly-varying-cong*:
  assumes $\bigwedge x.\ 0 < x \Longrightarrow U\ x = T\ x$
  shows *regularly-varying U ϱ* $\longleftrightarrow$ *regularly-varying T ϱ*
proof −
  have *regularly-varying T ϱ*
    if *U*:*regularly-varying U ϱ* **and** *x*:$\bigwedge x.\ 0 < x \Longrightarrow U\ x = T\ x$ **for** *U T*
  proof(*rule regularly-varyingI*)
    show $T \in borel\text{-}measurable$
        $(restrict\text{-}space\ borel\ \{0<..\})$
      using *regularly-varying-meas*[*OF U*] *x*
      by(*simp cong*: *measurable-cong'*)

```
  next
    show ((λt. T (t * x) / T t) ⟶ x powr ϱ) at-top if x-pos:0 < x for x
      by(rule tendsto-cong[THEN iffD1,OF - regularly-varying-lim[OF U]])
        (use x-pos x in auto intro!: eventually-at-top-linorderI[where c=1])
  qed(use U x regularly-varying-pos in fastforce)
  thus ?thesis
    using assms by auto
qed

lemma regularly-varying-ln-exp-iff:
regularly-varying U ϱ ⟷
   U ∈ borel-measurable (restrict-space borel {0<..}) ∧
   U ‘ {0<..} ⊆ {0<..} ∧
   (∀ x. ((λt. ln (U (exp (t + x))) − ln (U (exp t))) ⟶ ϱ * x) at-top)
  unfolding regularly-varying-def
proof safe
  fix x
  assume  U ‘ {0<..} ⊆ {0<..}
        ∀ y>0. ((λt. U (t * y) / U t) ⟶ y powr ϱ) at-top
  hence U:⋀x. 0 < x ⟹ 0 < U x
    and lim:⋀y. 0 < y ⟹ ((λt. U (t * y) / U t) ⟶ y powr ϱ) at-top
    by auto
  hence ⋀y. 0 < y ⟹ ((λt. U (exp t * y) / U (exp t)) ⟶ y powr ϱ) at-top
    using exp-at-top filterlim-compose by fast
  hence ((λt. U (exp t * exp x) / U (exp t)) ⟶ exp x powr ϱ) at-top
    using exp-gt-zero by blast
  hence ((λt. U (exp (t + x)) / U (exp t)) ⟶ exp (x * ϱ)) at-top
    using exp-gt-zero by (simp add: exp-add exp-powr-real)
  hence ((λt. ln (U (exp (t + x)) / U (exp t))) ⟶ ln (exp (x * ϱ))) at-top
    by(intro tendsto-ln) auto
  thus ((λt. ln (U (exp (t + x))) − ln (U (exp t))) ⟶ ϱ * x) at-top
    by(subst ln-divide-pos[symmetric]) (use U in auto simp: mult.commute)
next
  fix x :: real
  assume h:U ‘ {0<..} ⊆ {0<..}
        ∀ y. ((λt. ln (U (exp (t + y))) − ln (U (exp t))) ⟶ ϱ * y) at-top
   and x:0 < x
  hence U:⋀x. 0 < x ⟹ 0 < U x
    by blast
  obtain a where a:x = exp a
    using x exp-total by blast
  have ((λt. ln (U (exp (t + a))) − ln (U (exp t))) ⟶ ϱ * a) at-top
    using h by fast
  hence ((λt. ln (U (exp (t + a)) / U (exp t))) ⟶ ϱ * a) at-top
    by(subst ln-divide-pos) (use U in auto)
  hence ((λt. ln (U (exp (t + a)) / U (exp t))) ⟶ ln (x powr ϱ)) at-top
    by(simp add: a)
  from tendsto-exp[OF this]
  have ((λt. U (exp (t + a)) / U (exp t)) ⟶ x powr ϱ) at-top
```

using *exp-ln*[*OF powr-gt-zero*[*of x*,*THEN iffD2*]] *x*
by(*simp add*: *exp-ln*[*OF divide-pos-pos*[*OF U*[*OF exp-gt-zero*] *U*[*OF exp-gt-zero*]]])
hence ∗:((λ*t*. *U* (*exp t* ∗ *exp a*) / *U* (*exp t*)) ⟶ *x powr* ϱ) *at-top*
by (*simp add*: *mult-exp-exp*)
have ((λ*t*. *U* (*t* ∗ *exp a*) / *U t*) ⟶ *x powr* ϱ) *at-top*
by(*rule tendsto-cong*[*THEN iffD2*,*OF - filterlim-compose*[*OF* ∗ *ln-at-top*]])
(*auto intro*!: *eventually-at-top-linorderI*[**where** *c=1*])
thus ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶ *x powr* ϱ) *at-top*
by(*simp add*: *a*)
qed

lemma *regularly-varying-ln-exp*:
assumes *regularly-varying U* ϱ
shows ((λ*t*. *ln* (*U* (*exp* (*t* + *x*))) − *ln* (*U* (*exp t*))) ⟶ ϱ ∗ *x*) *at-top*
using *assms* **by**(*simp add*: *regularly-varying-ln-exp-iff*)

2.1.2 Slowly Varying Functions

abbreviation *slowly-varying L* ≡ *regularly-varying L 0*

lemma *slowly-varyingI*:
assumes *L* ∈ *borel-measurable* (*restrict-space borel* {*0*<..})
and ⋀*r*. *0* < *r* ⟹ *0* < *L r*
and ⋀*x*. *0* < *x* ⟹ ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶ *1*) *at-top*
shows *slowly-varying L*
by(*auto intro*!: *regularly-varyingI assms*)

lemma *slowly-varyingI-total*:
assumes *L* ∈ *borel-measurable borel*
and ⋀*r*. *0* < *r* ⟹ *0* < *L r*
and ⋀*x*. *0* < *x* ⟹ ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶ *1*) *at-top*
shows *slowly-varying L*
by(*auto intro*!: *regularly-varyingI-total assms*)

lemma *slowly-varying-lim*:
slowly-varying L ⟹ *0* < *x* ⟹ ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶ *1*) *at-top*
by(*auto dest*: *regularly-varying-lim*)

lemma *slowly-varying-asymp-equiv*:
assumes *slowly-varying L*
and *x*:*0* < *x*
shows (λ*t*. *L* (*t* ∗ *x*)) ∼[*at-top*] (λ*t*. *L t*)
using *assms* **by**(*auto dest*: *regularly-varying-asymp-equiv*)

lemma *slowly-varying-def2*:
slowly-varying L ⟷
L ∈ *borel-measurable* (*restrict-space borel* {*0*<..}) ∧
L ‘ {*0*<..} ⊆ {*0*<..} ∧
(∀ *x*>*0*. (λ*t*. *L* (*t* ∗ *x*)) ∼[*at-top*] (λ*t*. *L t*))

unfolding *regularly-varying-def2* **by** *auto*

lemma *slowly-varying-ln-exp*:
assumes *slowly-varying L*
shows $((\lambda t.\ ln\ (L\ (exp\ (t + x))) - ln\ (L\ (exp\ t))) \longrightarrow 0)\ at\text{-}top$
using *regularly-varying-ln-exp*[*OF assms*] **by** *simp*

lemma *slowly-varying-const*[*simp*]: $0 < c \Longrightarrow$ *slowly-varying* $(\lambda x.\ c)$
by(*auto intro*!: *slowly-varyingI-total*)

lemma *regularly-varying-powr*[*simp*]: *regularly-varying* $(\lambda x.\ x\ powr\ \varrho)\ \varrho$
by (*auto simp*: *powr-mult*
intro!: *regularly-varyingI-total tendsto-eventually*
eventually-at-top-linorderI[**where** *c=1*])

lemma *regularly-varying-id*[*simp*]: *regularly-varying* $(\lambda x.\ x)\ 1$
by(*auto intro*!: *regularly-varyingI-total simp*: *tendsto-iff frequently-def*)

lemma *slowly-varying-1*: *slowly-varying* $(\lambda x.\ 1)$
by *simp*

lemma *regularly-varying-mult*:
assumes *U*:*regularly-varying U* ϱ
and *V*:*regularly-varying V* η
shows *regularly-varying* $(\lambda x.\ U\ x * V\ x)\ (\varrho + \eta)$
proof(*rule regularly-varyingI*)
fix x :: *real*
assume $0 < x$
then have $*$:$((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow x\ powr\ \varrho)\ at\text{-}top$
and $**$:$((\lambda t.\ V\ (t * x)\ /\ V\ t) \longrightarrow x\ powr\ \eta)\ at\text{-}top$
using *assms* **by**(*auto simp*: *regularly-varying-lim*)
show $((\lambda t.\ U\ (t * x) * V\ (t * x)\ /\ (U\ t * V\ t)) \longrightarrow x\ powr\ (\varrho + \eta))\ at\text{-}top$
using *tendsto-mult*[*OF* $* **$] **by**(*simp add*: *powr-add*)
qed(*use assms* **in**
auto simp: *regularly-varying-pos regularly-varying-meas*
intro!: *mult-pos-pos borel-measurable-times*)

corollary
shows *regularly-varying-cmult*:
$0 < c \Longrightarrow$ *regularly-varying U* $\varrho \Longrightarrow$ *regularly-varying* $(\lambda x.\ c * U\ x)\ \varrho$
and *regularly-varying-multc*:
$0 < c \Longrightarrow$ *regularly-varying U* $\varrho \Longrightarrow$ *regularly-varying* $(\lambda x.\ U\ x * c)\ \varrho$
using *regularly-varying-mult*[*OF - slowly-varying-const*[*of c*],*of U* ϱ]
by(*auto simp*: *mult.commute*)

lemma *regularly-varying-inverse*:
assumes *regularly-varying U* ϱ
shows *regularly-varying* $(\lambda x.\ 1\ /\ U\ x)\ (-\ \varrho)$
proof(*rule regularly-varyingI*)

fix x :: *real*
assume [*arith*]:$0 < x$
then have $*$:$((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow x\ powr\ \varrho)$ *at-top*
by(*simp add*: *regularly-varying-lim assms*)
show $((\lambda t.\ 1\ /\ U\ (t * x)\ /\ (1\ /\ U\ t)) \longrightarrow x\ powr\ -\ \varrho)$ *at-top*
using *tendsto-inverse*[*OF* $*$] **by**(*auto simp*: *powr-minus*[*symmetric*])
qed(*use assms* **in** *auto dest*: *regularly-varying-meas regularly-varying-pos*)

corollary *regularly-varying-inverse-iff*:
regularly-varying $(\lambda x.\ 1\ /\ U\ x)\ (-\ \varrho) \longleftrightarrow$ *regularly-varying* $U\ \varrho$
using *regularly-varying-inverse*[*of* $\lambda x.\ 1\ /\ U\ x\ -\ \varrho$]
regularly-varying-inverse[*of* $U\ \varrho$] **by** *fastforce*

lemma *regularly-varying-divide*:
assumes U:*regularly-varying* $U\ \varrho$
and V:*regularly-varying* $V\ \eta$
shows *regularly-varying* $(\lambda x.\ U\ x\ /\ V\ x)\ (\varrho\ -\ \eta)$
using *regularly-varying-mult*[*OF* U *regularly-varying-inverse*[*OF* V]]
by *simp*

lemma *regularly-varying-comp-cmult*:
assumes *rv*:*regularly-varying* $U\ \varrho$
and c:$0 < c$
shows *regularly-varying* $(\lambda x.\ U\ (c * x))\ \varrho$
proof(*rule regularly-varyingI*)
show $(\lambda x.\ U\ (c * x)) \in$ *borel-measurable* (*restrict-space borel* $\{0<..\}$)
by(*intro measurable-compose*[*OF - regularly-varying-meas*[*OF rv*]] *measurable-restrict-space3*)
(*use* c **in** *auto*)
show $((\lambda t.\ U\ (c * (t * x))\ /\ U\ (c * t)) \longrightarrow x\ powr\ \varrho)$ *at-top* **if** x:$0 < x$ **for** x
unfolding *mult.assoc*[*symmetric*]
by(*rule filterlim-compose*[*OF regularly-varying-lim*[*OF rv x*]])
(*use* c *filterlim-ident filterlim-tendsto-pos-mult-at-top* **in** *fastforce*)
qed(*use assms* **in** *auto simp*: *regularly-varying-pos*)

lemma *regularly-varying-comp-multc*:
assumes *regularly-varying* $U\ \varrho$
and $0 < c$
shows *regularly-varying* $(\lambda x.\ U\ (x * c))\ \varrho$
using *regularly-varying-comp-cmult*[*OF assms*] **by**(*simp add*: *mult.commute*)

lemma *regularly-varying-iff-slowly-varying-divide-powr*:
regularly-varying $U\ \varrho \longleftrightarrow$ *slowly-varying* $(\lambda x.\ U\ x\ /\ x\ powr\ \varrho)$
proof
assume *slowly-varying* $(\lambda x.\ U\ x\ /\ x\ powr\ \varrho)$
from *regularly-varying-mult*[*OF this regularly-varying-powr*[*of* ϱ],*simplified*]
show *regularly-varying* $U\ \varrho$
by(*simp cong*: *regularly-varying-cong*)
next
assume *regularly-varying* $U\ \varrho$

```
  from regularly-varying-divide[OF this regularly-varying-powr[of ϱ]]
  show slowly-varying (λx. U x / x powr ϱ)
    by simp
qed

corollary regularly-varying-decomp:
  assumes regularly-varying U ϱ
  obtains L where
   slowly-varying L
   ⋀x. 0 < x ⟹ U x = x powr ϱ * L x
  using assms regularly-varying-iff-slowly-varying-divide-powr by fastforce

lemma slowly-varying-ln:slowly-varying (λx. ln (1 + x))
proof(safe intro!: slowly-varyingI-total)
  fix x :: real
  assume x[arith]:0 < x
  have tendsto-add':(f ⟶ (0::real)) F ⟹ (g ⟶ b) F
                    ⟹ ((λx::real. f x + g x) ⟶ b) F for f g b F
    using tendsto-add by fastforce
  show ((λt. ln (1 + t * x) / ln (1 + t)) ⟶ 1) at-top
  proof(rule Lim-transform-eventually)
    show ((λt. ln (1 / (1 + t) + (t * x) / (1 + t)) / ln (1 + t) + 1) ⟶ 1)
at-top
    proof −
      have ((λy. y * x / (1 + y)) ⟶ x) at-top
      proof(rule Lim-transform-eventually)
        have ((λy. x / (1 / y + 1)) ⟶ x / 1) at-top
          by(intro tendsto-divide tendsto-add')
            (use filterlim-ident real-tendsto-divide-at-top in fastforce)+
        thus ((λy. x / (1 / y + 1)) ⟶ x) at-top
          by simp
        show ∀F y in at-top. x / (1 / y + 1) = y * x / (1 + y)
          by(rule eventually-at-top-linorderI[where c=1])
         (smt (verit, ccfv-SIG) Groups.mult-ac(2) add-divide-eq-if-simps(1) div-by-1
              divide-divide-eq-right)
      qed
      show ?thesis
        apply(rule tendsto-add')
         apply(rule real-tendsto-divide-at-top[where c=ln x])
          apply(rule tendsto-ln)
           apply(rule tendsto-add')
            apply(rule real-tendsto-divide-at-top[where c=1])
             apply simp
        using filterlim-ident filterlim-tendsto-add-at-top
           apply fastforce
          apply fact
         apply arith
        using filterlim-compose filterlim-ident filterlim-tendsto-add-at-top ln-at-top
         apply blast
```

```
      apply simp
      done
  qed
 next
  show ∀F y in at-top. ln (1 / (1 + y) + y * x / (1 + y)) / ln (1 + y) + 1 =
                       ln (1 + y * x) / ln (1 + y)
  proof(rule eventually-at-top-linorderI[where c=1])
    fix y :: real
    assume y[arith]: 1 ≤ y
    have ln (1 / (1 + y) + y * x / (1 + y)) / ln (1 + y) + 1
       = ln (1 / (1 + y) + y * x / (1 + y)) / ln (1 + y) + ln (1 + y) / ln (1
+ y)
      by simp
    also have ... = (ln (1 / (1 + y) + y * x / (1 + y)) + ln (1 + y)) / ln (1
+ y)
      by algebra
    also have ... = ln ((1 / (1 + y) + y * x / (1 + y)) * (1 + y)) / ln (1 + y)
      by(subst ln-mult-pos) (auto simp add: add-sign-intros(2))
    also have ... = ln (1 + y * x) / ln (1 + y)
      by (metis add-divide-distrib divide-cancel-right ln-0 nonzero-eq-divide-eq)
    finally show ln (1 / (1 + y) + y * x / (1 + y)) / ln (1 + y) + 1 =
                 ln (1 + y * x) / ln (1 + y) .
  qed
 qed
qed auto
```

Exercise 2.8 of the book [3].

```
lemma slowly-varying-tendsto-gr-le-eq:
  fixes U :: 'a :: linordered-field ⇒ 'b::{real-normed-div-algebra, field}
  shows (∀x>1. ((λt. U (t * x) / U t) ⟶ 1) at-top)
    ⟷ (∀x. 0 < x ⟶ x < 1 ⟶ ((λt. U (t * x) / U t) ⟶ 1) at-top)
proof safe
  assume h:(∀x>1. ((λt. U (t * x) / U t) ⟶ 1) at-top)
  fix x :: 'a
  assume x[arith]:0 < x x < 1
  then have 1 < 1 / x
    by simp
  hence ((λt. U (t * (1 / x)) / U t) ⟶ 1) at-top
    using h by blast
  hence ((λt. U ((t * x) * (1 / x)) / U (t * x)) ⟶ 1) at-top
    by(rule filterlim-compose)
      (use x in
           auto simp: filterlim-at-top eventually-at-top-linorder pos-divide-le-eq[of
x,symmetric])
  hence *:((λt. U t / U (t * x)) ⟶ 1 / 1) at-top
    by simp
  thus ((λt. U (t * x) / U t) ⟶ 1) at-top
    using tendsto-inverse[OF *] by simp
next
```

assume *h*:∀ *x*>*0*. *x* < *1* ⟶ ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶⟶ *1*) *at-top*
fix *x* :: *'a*
assume *x*[*arith*]:*1* < *x*
then have *0* < *1* / *x 1* / *x* < *1*
by *simp-all*
hence ((λ*t*. *U* (*t* ∗ (*1* / *x*)) / *U t*) ⟶⟶ *1*) *at-top*
using *h* **by** *blast*
hence ((λ*t*. *U* ((*t* ∗ *x*) ∗ (*1* / *x*)) / *U* (*t* ∗ *x*)) ⟶⟶ *1*) *at-top*
by(*rule filterlim-compose*)
(*use x* **in**
auto simp: *filterlim-at-top eventually-at-top-linorder pos-divide-le-eq*[*of x*,*symmetric*])
hence ∗:((λ*t*. *U t* / *U* (*t* ∗ *x*)) ⟶⟶ *1* / *1*) *at-top*
by *simp*
thus ((λ*t*. *U* (*t* ∗ *x*) / *U t*) ⟶⟶ *1*) *at-top*
using *tendsto-inverse*[*OF* ∗] **by** *simp*
qed

lemma *slowly-varying-iff-all-gr-1*:
assumes *L* ∈ *borel-measurable* (*restrict-space borel* {*0*<..})
and ⋀*x*. *0* < *x* ⟹ *0* < *L x*
shows *slowly-varying L* ⟷ (∀ *x*>*1*. ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶⟶ *1*) *at-top*)
proof
assume *h*:∀ *x*>*1*. ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶⟶ *1*) *at-top*
show *slowly-varying L*
proof(*rule slowly-varyingI*)
fix *x* :: *real*
have ∗:*x* = *1* ⟹ ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶⟶ *1*) *at-top*
using *assms*
by(*intro tendsto-eventually eventually-at-top-linorderI*[**where** *c*=*1*])
(*metis divide-self less-irrefl mult-1-right less-le-trans zero-less-one*)
assume *0* < *x*
then consider *0* < *x x* < *1* | *x* = *1* | *1* < *x*
by *linarith*
thus ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶⟶ *1*) *at-top*
by *cases* (*use h slowly-varying-tendsto-gr-le-eq assms* ∗ **in** *auto*)
qed(*use assms* **in** *auto*)
qed(*auto intro*!: *slowly-varying-lim*)

lemma *slowly-varying-iff-ne-1*:
assumes *L-meas*:*L* ∈ *borel-measurable* (*restrict-space borel* {*0*<..})
and *L-pos*:⋀*x*. *0* < *x* ⟹ *0* < *L x*
and *L-mono*:*mono-on* {*0*<..} *L*
shows *slowly-varying L* ⟷ (∃ *x*≠*1*. *x* > *0* ∧ ((λ*t*. *L* (*t* ∗ *x*) / *L t*) ⟶⟶ *1*) *at-top*)
proof *safe*
fix *x'* :: *real*
assume *x'-ne1*:*x'* ≠ *1*
and *x'-pos*:*0* < *x'*

```
  and lim':((λt. L (t * x') / L t) ⟶ 1) at-top
 have lim'':((λt. L (t * (1 / x')) / L t) ⟶ 1) at-top
  using filterlim-compose[OF lim'
          filterlim-compose[OF filterlim-ident
                  filterlim-tendsto-pos-mult-at-top[OF - - filterlim-ident,
                    where c=1 / x' and f=λx. 1 / x']]]
       tendsto-inverse-real x'-pos
  by fastforce
 define x where x ≡ if 1 < x' then x' else 1 / x'
 have x1[arith]:1 < x and x-pos[arith]:0 < x and x-pos'[arith]:0 < 1 / x 1 / x
< 1
  using x'-ne1 x'-pos
  by(auto simp: x-def)
 have lim-x1:((λt. L (t * x) / L t) ⟶ 1) at-top
  and lim-x2:((λt. L (t * (1 / x)) / L t) ⟶ 1) at-top
  using lim' lim'' by(auto simp: x-def)
 have lim-y1:((λt. L (t * y) / L t) ⟶ 1) at-top
  if y[arith]:1 / x ≤ y y ≤ x for y
  by(rule tendsto-sandwich[OF - - lim-x2 lim-x1])
   (auto intro!: eventually-at-top-linorderI[where c=1] less-imp-le[OF L-pos]
   divide-right-mono mono-onD[OF L-mono] mult-pos-pos simp del: times-divide-eq-right)
 have lim-y:1 ≤ y ⟹ y ≤ x^n ⟹ ((λt. L (t * y) / L t) ⟶ 1) at-top for y
n
 proof(induction n arbitrary: y)
  case 0
  then show ?case
   using L-pos
   by(auto intro!: tendsto-cong[THEN iffD1,OF - tendsto-const]
      eventually-at-top-linorderI[where c=1] simp: order.strict-iff-order)
 next
  fix y n
  assume ih:⋀z. 1 ≤ z ⟹ z ≤ x ^ n ⟹
            ((λt. L (t * z) / L t) ⟶ 1) at-top
   and y1[arith]:1 ≤ y and y2:y ≤ x ^ Suc n
  show ((λt. L (t * y) / L t) ⟶ 1) at-top
  proof(cases x < y)
   assume [arith]:x < y
   then have y:1 ≤ y / x y / x ≤ x ^ n
    using y2 by (auto simp add: cross3-simps pos-divide-le-eq)
   note lim-ih =ih[OF this]
   have 1:∀F t in at-top. L (t * y) / L t
                  = (L (t * (y / x)) / L t) * (L (t * y) / L (t * y * 1 / x))
    using  y order.strict-iff-order[THEN iffD1,OF L-pos]
    by(auto intro!: eventually-at-top-linorderI[where c=1])
   have ((λt. (L (t * (y / x)) / L t) * (L (t * y) / L (t * y * 1 / x))) ⟶ 1
* 1) at-top
    using tendsto-inverse[OF filterlim-compose[OF lim-x2
      filterlim-at-top-mult-tendsto-pos[OF tendsto-const[of y] - filterlim-ident]]]
    by(intro tendsto-mult lim-ih) simp
```

thus *?thesis*
using *tendsto-cong*[*OF 1*] **by** *auto*
qed(*auto intro*!: *lim-y1*)
qed
show *slowly-varying L*
proof(*safe intro*!: *slowly-varying-iff-all-gr-1*[*THEN iffD2*] *assms*)
fix *y* :: *real*
assume *y*:*1* < *y*
then obtain *n* **where** *n*:*y* ≤ *x* ^ *n*
using *power-of-nat-log-ge x1* **by** *blast*
thus ((λ*t*. *L* (*t* ∗ *y*) / *L t*) ⟶ *1*) *at-top*
using *y* **by**(*auto intro*!: *lim-y*)
qed
qed(*auto simp*: *slowly-varying-lim intro*!: *exI*[**where** *x*=*2*])

2.2 Uniform Convergence Theorem on Intervals

theorem *slowly-varying-UCT*:
assumes *sv*:*slowly-varying L*
and *a*: *0* < *a*
shows *uniform-limit* {*a*..*b*} (λ*t x*. *L* (*t* ∗ *x*) / *L t*) (λ*x*. *1*) *at-top*
proof(*cases a* < *b*)
assume *a-le-b*:*a* < *b*
have *lna-le-lnb*:*ln a* < *ln b*
using *a-le-b a* **by** *simp*
have *eb*: ∀ *F i in at-top*.
∀ *x*∈{*ln a*..*ln b*}. *ln* (*L* (*exp i* ∗ *exp x*) / *L* (*exp i*)) ∈ {− *1*..*1*} (**is** *?eb*)
and *uniform-limit* {*ln a*..*ln b*}
(λ*t x*. *ln* (*L* (*exp* (*t* + *x*)) / *L* (*exp t*))) (λ*x*. *0*) *at-top* (**is** *?ul*)
proof −
show *g1*:*?ul*
proof(*safe intro*!: *uniform-limitI*)
fix *e* :: *real*
assume *e*:*0* < *e*
let *?h* = λ*t*. *ln* (*L* (*exp t*))
let *?A* = *abs* (*ln b*) + *abs* (*ln a*)
have *A*:*0* < *?A*
using *a lna-le-lnb* **by** *linarith*
note [*measurable*] = *regularly-varying-meas-log-exp*[*OF sv*]
define *I* **where** *I* ≡ λ*t*. {*t*..*t* + *2* ∗ *?A*}
define *E* **where** *E* ≡ λ*t*. *I t* ∩ {*x*. *e* / *2* ≤ |*?h x* − *?h t*|}
define *E*′ **where** *E*′ ≡ λ*t*. *I 0* ∩ {*x*. *e* / *2* ≤ |*?h* (*x* + *t*) − *?h t*|}
have [*measurable*]: ⋀*t*. *I t* ∈ *sets borel* ⋀*t*. *E t* ∈ *sets borel*
⋀*t*. *E*′ *t* ∈ *sets borel*
by(*auto simp*: *E-def E*′*-def I-def*)
have *measE-eq-measE*′:*measure lborel* (*E t*) = *measure lborel* (*E*′ *t*) **for** *t*
proof −
{
fix *x*

```
      have x ∈ (λx. x − t) ‘ E t ⟷ x + t ∈ (+) t ‘ (λx. x − t) ‘ E t
        by (simp add: image-iff)
      also have ... ⟷ x + t ∈ E t
        by(auto simp: image-image)
      also have x + t ∈ E t ⟷ x ∈ E′ t
        by(auto simp: E-def E′-def I-def)
      finally have x ∈ (λx. x − t) ‘ E t ⟷ x ∈ E′ t .
    }
    hence 1:(λx. x − t) ‘ E t = E′ t
      by blast
    have measure lborel (E t) = measure lebesgue (E t)
      by simp
    also have ... = measure lebesgue ((λx. x − t) ‘ E t)
      by(simp add: measure-translation-subtract)
    also have ... = measure lebesgue (E′ t)
      by (simp add: 1)
    also have ... = measure lborel (E′ t)
      by simp
    finally show ?thesis .
  qed
  have ((λt. integral^L lborel (indicat-real (E′ t)))
        ⟶ integral^L lborel (λx::real. 0)) at-top
  proof(rule integral-dominated-convergence-real)
    show AE x in lborel. norm (indicat-real (E′ t) (x :: real))
                  ≤ indicat-real {0..2 * ?A} x for t
      by(auto simp: E′-def I-def indicator-def)
  next
    have ∀_F t in at-top. |?h (x + t) − ?h t| < e / 2 for x
      using tendstoD[OF tendsto-rabs[OF regularly-varying-ln-exp[OF sv,of x]]
          half-gt-zero[OF e]]
      by(simp add: dist-real-def add.commute)
    then obtain T where T:⋀x t. T x ≤ t ⟹ |?h (x + t) − ?h t| < e / 2
      unfolding eventually-at-top-linorder by meson
    hence ∀_F t in at-top. indicat-real (E′ t) x = 0 for x
      unfolding eventually-at-top-linorder
      by(fastforce intro!: exI[where x=T x] simp: indicator-def E′-def)
    hence ((λt. indicat-real (E′ t) x) ⟶ 0) at-top for x
      by (simp add: tendsto-eventually)
    thus AE x in lborel. ((λt. indicat-real (E′ t) x) ⟶ 0) at-top
      by blast
  qed (auto simp: emeasure-lborel-Icc-eq)
  from tendstoD[OF this min-less-iff-conj[THEN iffD2
      ,OF conjI[OF half-gt-zero[OF e] half-gt-zero[OF A]]]]
  obtain t0 where t0:⋀t. t0 ≤ t ⟹ measure lborel (E t) < min (e / 2) (?A
/ 2)
    unfolding measE-eq-measE′ dist-real-def eventually-at-top-linorder
    by auto
  show ∀_F i in at-top. ∀x∈{ln a..ln b}. dist (ln (L (exp (i + x)) / L (exp i)))
0 < e
```

```
proof(safe intro!: eventually-at-top-linorderI[where c=t0 + |ln b| + |ln a|])
  fix t x
  assume t:t0 + |ln b| + |ln a| ≤ t and x:x ∈ {ln a..ln b}
  have (I (t + x) ∩ I t) − (E (t + x) ∪ E t) ≠ {}
  proof
    assume ne:I (t + x) ∩ I t − (E (t + x) ∪ E t) = {}
    have 0 ≤ x ⟹ I (t + x) ∩ I t = {t + x ..t + 2∗?A}
         x ≤ 0 ⟹ I (t + x) ∩ I t = {t..t + x + 2∗?A}
      unfolding Int-atLeastAtMost I-def using x lna-le-lnb by force+
    hence 0 ≤ x ⟹ measure lborel (I (t + x) ∩ I t) = 2 ∗ ?A − x
          x ≤ 0 ⟹ measure lborel (I (t + x) ∩ I t) = 2 ∗ ?A + x
      using x lna-le-lnb by auto
    hence ∗:measure lborel (I (t + x) ∩ I t) ≥ ?A
      using x by fastforce
    have measure lborel (E (t + x) ∪ E t)
         ≤ measure lborel (E (t + x)) + measure lborel (E t)
      by(simp add: measure-Un-le)
    also have ... < ?A / 2 + ?A / 2
      by(rule add-strict-mono)
        (use t0[of t] t t0[of t + x] x in fastforce)+
    finally have ∗∗:measure lborel (E (t + x) ∪ E t) < ?A
      by argo
    have 0 = ?A − ?A
      by simp
    also have ... < measure lborel (I (t + x) ∩ I t) − measure lborel (E (t +
x) ∪ E t)
      using ∗ ∗∗ by argo
    also have ... ≤ measure lborel ((I (t + x) ∩ I t) − (E (t + x) ∪ E t))
      by (metis Diff-Diff-Int E-def I-def ‹⋀t. E t ∈ sets borel›
          ‹⋀t. I t ∈ sets borel› compact-Icc fmeasurable.Un fmeasurableI2
          fmeasurable-compact inf-le1 measure-diff-le-measure-setdiff
          sets.Diff sets-lborel)
    also have ... = 0
      unfolding ne by simp
    finally show False
      by simp
  qed
  then obtain k where k:k ∈ I (t + x) k ∈ I t
    k ∉ {y. e / 2 ≤ |?h y − ?h (t + x)|} k ∉ {y. e / 2 ≤ |?h y − ?h t|}
    unfolding E-def by blast
  have |ln (L (exp (t + x)) / L (exp t)) − 0| = |?h (t + x) − ?h t|
 by(auto intro!: ln-divide-pos arg-cong[where f=abs] regularly-varying-pos[OF
sv])
  also have ... ≤ |?h k − ?h (t + x) | + |?h k − ?h t|
    by force
  also have ... < e / 2 + e / 2
    by(intro add-strict-mono) (use k in fastforce)+
  finally show dist (ln (L (exp (t + x)) / L (exp t))) 0 < e
    by (simp add: dist-real-def)
```

```
    qed
  qed
  show ?eb
    by(rule eventually-mono[OF  uniform-limitD[OF g1,of 1]])
      (auto simp:  exp-add)
 qed
 hence uniform-limit {ln a..ln b} (λt x. ln (L (exp t * exp x) / L (exp t))) (λx.
0) at-top
  by (simp add: exp-add)
 hence uniform-limit {ln a..ln b}
       (λi x. exp (ln (L (exp i * exp x) / L (exp i)))) (λx. exp 0) at-top
  by(safe intro!: compact-uniformly-continuous continuous-on-exp
      eb uniform-limit-compose-uniformly-continuous-on[where B={−1..1} and
g' = λx. 0])
     auto
 hence uniform-limit {ln a..ln b} (λi x. L (exp i * exp x) / L (exp i)) (λx. 1)
at-top
  using exp-ln[OF divide-pos-pos[OF regularly-varying-pos[OF sv] regularly-varying-pos[OF
sv]]]
  by auto
 hence uniform-limit {a..b} (λi x. L (exp i * exp (ln x)) / L (exp i)) (λx. 1)
at-top
  using uniform-limit-compose'[where A={ln a..ln b}
    and f=λi x. L (exp i * exp x) / L (exp i)
    and g=λx. 1 and h=ln and B={a..b}] a by auto
 hence uniform-limit {a..b} (λi x. L (exp i * x) / L (exp i)) (λx. 1) at-top
  by(intro uniform-limit-cong[of - - λi x. L (exp i * x) / L (exp i),THEN iffD1])
    (use a in auto)
 hence uniform-limit {a..b} (λi x. L (exp (ln i) * x) / L (exp (ln i))) (λx. 1)
at-top
  by(intro filterlim-compose[OF - ln-at-top])
 thus uniform-limit {a..b} (λi x. L (i * x) / L i) (λx. 1) at-top
  by(intro uniform-limit-cong[of - - λi x. L (i * x) / L i,THEN iffD1]
      eventually-at-top-linorderI[where c=1]) auto
next
 assume ¬ a < b
 then consider a = b | b < a
  by fastforce
 thus ?thesis
  by cases (use a slowly-varying-lim[OF sv] in auto)
qed

corollary regularly-varying-UCT-interval:
 assumes sv:regularly-varying U ϱ
  and a: 0 < a
 shows uniform-limit {a..b} (λt x. U (t * x) / U t) (λx. x powr ϱ) at-top
proof −
 obtain L where L-sv: slowly-varying L
  and eq-L: ⋀x. 0 < x ⟹ U x = x powr ϱ * L x
```

```
    using regularly-varying-decomp[OF sv] by blast
  have 1:uniform-limit {a..b} (λt x. x powr ϱ * (L (t * x) / L t)) (λx. x powr ϱ *
1) at-top
    by(intro uniform-lim-mult-bdd1[OF slowly-varying-UCT] assms L-sv
        compact-imp-bounded compact-continuous-image continuous-on-powr)
      (use a in auto)
  have 2:uniform-limit {a..b} (λt x. (x powr ϱ * t powr ϱ * L (t * x) / (t powr ϱ
* L t)))
                              (λx. x powr ϱ) at-top
   by(auto intro!: uniform-limit-cong[THEN iffD1,OF - - 1] eventually-at-top-linorderI[of
1])
  show ?thesis
    using a
   by(auto intro!: uniform-limit-cong[THEN iffD1,OF - - 2] eventually-at-top-linorderI[of
1]
       simp: powr-mult eq-L)
qed

corollary regularly-varying-UCT-interval-ln-exp:
  assumes rv:regularly-varying U ϱ
  defines h ≡ λx. ln (U (exp x))
  shows uniform-limit {a..b} (λt x. h (t + x) − h t) (λx. ϱ * x) at-top
proof −
  have 1:uniform-limit {exp a..exp b} (λt x. U (t * x) / U t) (λx. x powr ϱ) at-top
    by(auto intro!: regularly-varying-UCT-interval[OF rv])
  have uniform-limit {exp a..exp b} (λt x. ln (U (t * x) / U t)) (λx. ln (x powr
ϱ)) at-top
  proof(intro uniform-limit-compose-uniformly-continuous-on[OF 1]
            uniform-limit-eventually-in[OF regularly-varying-UCT-interval[OF rv]])
    define e where e ≡ min (exp (a * ϱ)) (exp (b * ϱ))
    have e:0 < e
      by(simp add: e-def)
    define e′ where e′ ≡ e / 2
    have e′:e′ < e 0 < e′
      using dense e′-def e by auto
    have pos:0 < min (exp (a * ϱ)) (exp (b * ϱ)) − e′
      using e′ by(auto simp: e-def)
    thus uniformly-continuous-on
          {min (exp (a * ϱ)) (exp (b * ϱ)) − e′..max (exp (a * ϱ)) (exp (b * ϱ)) +
e′} ln
         (is uniformly-continuous-on ?B ln)
      using e′ by(auto intro!: continuous-on-ln compact-uniformly-continuous)
    show 0 < e′
      by fact
    fix x y
    assume x:x ∈ {exp a .. exp b}
      and d:dist (x powr ϱ) y < e′
    have a-le-b:a ≤ b
      by(rule ccontr) (use x in auto)
```

```
    have xin:x powr ϱ ∈ {min (exp (a * ϱ)) (exp (b * ϱ))..max (exp (a * ϱ)) (exp
(b * ϱ))}
      (is - ∈ ?C)
    proof(cases 0 ≤ ϱ)
      assume h:0 ≤ ϱ
      then have mono:mono-on {exp a .. exp b} (λx. x powr ϱ)
        by(auto intro!: mono-onI powr-mono2 order.trans[OF exp-ge-zero])
      have 1:min (exp (a * ϱ)) (exp (b * ϱ)) = exp (a * ϱ)
        and 2: max (exp (a * ϱ)) (exp (b * ϱ)) = exp (b * ϱ)
        using h a-le-b by (simp-all add: mult-le-cancel-right)
      have (λx. x powr ϱ) ` {exp a .. exp b} = {exp (a * ϱ) .. exp (b * ϱ)}
        unfolding exp-powr-real[symmetric]
        by(intro image-mono-on-continuous-on-Icc mono continuous-on-powr)
          (use a-le-b in auto)
      thus ?thesis
        using x by(auto simp: 1 2)
    next
      assume h:¬ 0 ≤ ϱ
      then have mono:mono-on {exp a .. exp b} (λx. − (x powr ϱ))
        by(intro mono-onI)
         (metis atLeastAtMost-iff basic-trans-rules(22) exp-gt-zero neg-le-iff-le nle-le
            powr-mono2')
      have 1:min (exp (a * ϱ)) (exp (b * ϱ)) = exp (b * ϱ)
        and 2: max (exp (a * ϱ)) (exp (b * ϱ)) = exp (a * ϱ)
        using h a-le-b by (simp-all add: mult-le-cancel-right)
      have (λx. x powr ϱ) ` {exp a .. exp b} = uminus ` (λx. − (x powr ϱ)) ` {exp
a .. exp b}
        by force
      also have ... = uminus ` {− (exp (a * ϱ))..− (exp (b * ϱ))}
        unfolding exp-powr-real[symmetric]
        by(intro image-mono-on-continuous-on-Icc mono continuous-on-minus
              continuous-on-powr arg-cong[where f=(`) uminus])
          (use a-le-b in auto)
      also have ... = {exp (b * ϱ) .. exp (a * ϱ)}
        by auto
      finally show ?thesis
        using x by(auto simp: 1 2)
    qed
    show y ∈ ?B
      using xin d by(auto simp: dist-real-def)
  qed auto
  hence uniform-limit {exp a..exp b}
        (λt x. ln (U (exp t * x) / U (exp t))) (λx. ϱ * ln x) at-top
    using filterlim-compose[OF - exp-at-top] by auto
  hence uniform-limit {a..b}
        (λt x. ln (U (exp t * exp x) / U (exp t))) (λx. ϱ * ln (exp x)) at-top
    by(intro uniform-limit-compose'[where h=exp]) auto
  hence uniform-limit {a..b} (λt x. ln (U (exp (t + x)) / U (exp t))) (λx. ϱ * x)
at-top
```

by (*simp add*: *exp-add*)
thus *uniform-limit* {*a*..*b*} (λ*t x*. *h* (*t* + *x*) − *h t*) (λ*x*. *ϱ* ∗ *x*) *at-top*
using *regularly-varying-pos*[*OF rv exp-gt-zero*] *ln-divide-pos*
by(*simp add*: *h-def cong*: *uniform-limit-cong′*)
qed

corollary *slowly-varying-UCT-interval-ln-exp*:
assumes *sv*:*slowly-varying L*
defines *h* ≡ λ*x*. *ln* (*L* (*exp x*))
shows *uniform-limit* {*a*..*b*} (λ*t x*. *h* (*t* + *x*) − *h t*) (λ*x*. *0*) *at-top*
using *regularly-varying-UCT-interval-ln-exp*[*OF sv*] **by**(*simp add*: *h-def*)

corollary *regularly-varying-compose*:
assumes *rv1*:*regularly-varying U ϱ*
and *rv2*:*regularly-varying V η*
and *lim*:*filterlim U at-top at-top*
shows *regularly-varying* (λ*x*. *V* (*U x*)) (*ϱ* ∗ *η*)
proof(*rule regularly-varyingI*)
fix *x* ::*real*
assume *x*:*0* < *x*
hence *x powr ϱ* > *0*
by *simp*
then obtain *i* **where** *i-pos*:*0* < *i* **and** *i*: *0* < *x powr ϱ* − *i*
by (*meson diff-gt-0-iff-gt field-lbound-gt-zero*)
show ((λ*t*. *V* (*U* (*t* ∗ *x*)) / *V* (*U t*)) ⟶ *x powr* (*ϱ* ∗ *η*)) *at-top*
unfolding *tendsto-iff dist-real-def*
proof *safe*
fix *e* :: *real*
assume *e*:*0* < *e*
from *regularly-varying-UCT-interval*[*OF rv2 i*,*of x powr ϱ* + *i*] *half-gt-zero*[*OF e*]
obtain *Tv* **where** *Tv*:⋀*y t*. *Tv* ≤ *t* ⟹ *x powr ϱ* − *i* ≤ *y* ⟹ *y* ≤ *x powr ϱ* + *i*
⟹ |*V* (*t* ∗ *y*) / *V t* − *y powr η*| < *e* / *2*
unfolding *uniform-limit-iff dist-real-def eventually-at-top-linorder*
by (*meson atLeastAtMost-iff*)
obtain *T1* **where** *T1*:⋀*t*. *T1* ≤ *t* ⟹ *Tv* ≤ *U t*
using *lim* **unfolding** *filterlim-at-top eventually-at-top-linorder*
by *fastforce*
obtain *T2* **where** *T2*:⋀*t*. *T2* ≤ *t* ⟹ |*U* (*t* ∗ *x*) / *U t* − *x powr ϱ*| < *i*
using *regularly-varying-lim*[*OF rv1 x*] *i-pos*
by(*fastforce simp*: *tendsto-iff eventually-at-top-linorder dist-real-def*)
obtain *T3* **where** *T3*:⋀*t*. *T3* ≤ *t* ⟹ |(*U* (*t* ∗ *x*) / *U t*) *powr η* − *x powr* (*ϱ* ∗ *η*)| < *e* / *2*
using *tendstoD*[*OF tendsto-powr*[*OF regularly-varying-lim*[*OF rv1 x*]
tendsto-const[*of η*]] *half-gt-zero*[*OF e*]]
x
by(*fastforce simp*: *eventually-at-top-linorder powr-powr dist-real-def*)
define *T* **where** *T* ≡ *max* (*max 1 T1*) (*max T2 T3*)

```
    show ∀F t in at-top. |V (U (t * x)) / V (U t) - x powr (ϱ * η)| < e
    proof(safe intro!: eventually-at-top-linorderI[where c=T])
      fix t
      assume t:T ≤ t
      then have t-pos:0 < t and Tvt:Tv ≤ U t
        and ti:|U (t * x) / U t - x powr ϱ| < i
        and tpowr:|(U (t * x) / U t) powr η -  x powr (ϱ * η)| < e / 2
        using T1 T2 T3 by(auto simp: T-def)
      have |V (U (t * x)) / V (U t) - x powr (ϱ * η)|
            = |V (U t * (U (t * x) / U t)) / V (U t) - x powr (ϱ * η)|
        using regularly-varying-pos[OF rv1 t-pos] by simp
      also have ... ≤ |V (U t * (U (t * x) / U t)) / V (U t) - (U (t * x) / U t)
powr η|
                        + |(U (t * x) / U t) powr η - x powr (ϱ * η)|
        by linarith
      also have ... < e / 2 + e / 2
        using ti by(intro add-strict-mono tpowr Tv Tvt) auto
      finally show |V (U (t * x)) / V (U t) - x powr (ϱ * η)| < e
        by argo
    qed
  qed
qed(use measurable-compose[OF
      measurable-restrict-space2[OF - regularly-varying-meas[OF rv1]]
      regularly-varying-meas[OF rv2]]
  in auto intro!: regularly-varying-pos[OF rv1] regularly-varying-pos[OF rv2])


lemma
  assumes sv:slowly-varying L
  defines h ≡ λx. ln (L (exp x))
  shows slowly-varying-eventually-bounded-interval:
    ∀F x in at-top. ∀a≥x. ∀b≥a. bounded (L ‘ {a..b})
    and slowly-varying-eventually-bounded-interval-strong:
    ∀F x in at-top. ∀a≥x. ∀b≥a. ∃A>0. ∃B. L ‘ {a..b} ⊆ {A..B}
    and slowly-varying-eventually-bounded-interval-ln-exp:
    ∀F x in at-top. ∀a≥x. ∀b≥a. bounded (h ‘ {a..b})
proof −
  have ∃T::real. ∀t≥T. ∀n. t ≤ T + Suc n ⟶ |h t| < Suc n + |h T|
  proof −
    from uniform-limitD[where e=1,OF slowly-varying-UCT-interval-ln-exp[OF
sv,of 0 1]]
    obtain T where T':⋀t x. t ≥ T ⟹ 0 ≤ x ⟹ x ≤ 1 ⟹ |h (t + x) - h t|
< 1
      by(fastforce simp: eventually-at-top-linorder h-def)
    have ⋀t. T ≤ t ⟹ t ≤ T + Suc n ⟹ |h t| < Suc n + |h T| for n
    proof(induction n)
      case ih:(Suc n')
      then consider t ≤ T + Suc n' | T + Suc n' ≤ t t ≤ T + Suc (Suc n')
        by argo
```

thus *?case*
proof *cases*
assume t:$T + Suc\ n' \leq t\ t \leq T + Suc\ (Suc\ n')$
define t' **where** $t' \equiv t - 1$
have t':$T \leq t'\ t' \leq T + Suc\ n'$
using t **by**(*auto simp*: t'-*def*)
then have $|h\ t| < 1 + |h\ t'|$
using T'[*of* t' *1*] **by**(*simp add*: t'-*def*)
also have $... < real\ (Suc\ (Suc\ n')) + |h\ T|$
using *ih* t' **by** *fastforce*
finally show *?thesis* **.**
qed(*use ih* **in** *fastforce*)
qed(*use* T'[*of* T *- - T*] **in** *fastforce*)
thus *?thesis*
by *blast*
qed
then obtain T :: *real* **where** T:$\bigwedge t\ n.\ T \leq t \Longrightarrow t \leq T + Suc\ n \Longrightarrow |h\ t| < Suc\ n + |h\ T|$
by *fast*
show h:$\forall_F\ x$ *in at-top.* $\forall\, a{\geq}x.\ \forall\, b{\geq}a.$ *bounded* $(h\ `\ \{a..b\})$
proof(*intro eventually-at-top-linorderI*[**where** c=T] *allI impI*)
fix $t\ a\ b$ :: *real*
assume *tab*:$T \leq t\ t \leq a\ a \leq b$
then obtain N **where** N:$b \leq T + Suc\ N$
by (*metis Multiseries-Expansion.intyness-simps*(*5*) *add.commute*
add-mono-thms-linordered-semiring(*1*) *arithmetic-simps*(*49*) *diff-le-eq*
not0-implies-Suc of-nat-0-le-iff real-arch-simple)
show *bounded* $(h\ `\ \{a..b\})$
using *tab N*
by(*auto intro*!: *boundedI*[**where** B=$Suc\ N + |h\ T|$]
less-imp-le[*of - Suc N* + $|h\ T|$] *T simp del*: *of-nat-Suc*)
qed
have $*$:$\forall_F\ x$ *in at-top.* $\forall\, a{\geq}x.\ \forall\, b{\geq}a.\ \exists\, A{>}0.\ \exists\, B.\ (\lambda x.\ L\ (exp\ x))\ `\ \{a..b\} \subseteq \{A..B\}$
proof(*safe intro*!: *eventually-mono*[*OF h*])
fix $t\ a\ b$
assume $\forall\, a{\geq}t.\ \forall\, b{\geq}a.$ *bounded* $(h\ `\ \{a..b\})$
and $t \leq a\ a \leq b$
then obtain $A\ B$ **where** AB:$h\ `\ \{a..b\} \subseteq \{A..B\}$
unfolding *bounded-subset-cball cball-eq-atLeastAtMost* **by** *blast*
show $\exists\, A{>}0.\ \exists\, B.\ (\lambda x.\ L\ (exp\ x))\ `\ \{a..b\} \subseteq \{A..B\}$
by(*intro exI*[**where** x=*exp A*] *conjI*[*OF - exI*[**where** x=*exp B*]] *image-subsetI exp-gt-zero*)
(*metis AB regularly-varying-pos*[*OF sv exp-gt-zero*] *h-def*
atLeastAtMost-iff exp-le-cancel-iff exp-ln-iff image-subset-iff)
qed
then obtain T **where** T:
$\bigwedge a\ b.\ T \leq a \Longrightarrow a \leq b \Longrightarrow \exists\, A{>}0.\ \exists\, B.\ (\lambda x.\ L\ (exp\ x))\ `\ \{a..b\} \subseteq \{A..B\}$
unfolding *eventually-at-top-linorder* **by** *metis*

```
  have **:a ≤ b ⟹ exp ‘ {a..b} = {exp a..exp b} for a b :: real
    by(intro image-mono-on-continuous-on-Icc)
      (auto intro!: continuous-on-exp ord.mono-onI)
  show ***:∀F x in at-top. ∀ a≥x. ∀ b≥a. ∃ A>0. ∃ B. L ‘ {a..b} ⊆ {A..B}
  proof(safe intro!: eventually-at-top-linorderI[where c=exp T])
    fix t a b :: real
    assume tab:exp T ≤ t t ≤ a a ≤ b
    hence tab0:0 < t 0 < a 0 < b
      using exp-gt-zero[of T] by linarith+
    hence lnab:T ≤ ln a ln a ≤ ln b
      using tab ln-ge-iff by(auto intro!: ln-mono)
    then obtain A B where AB:0 < A (λx. L (exp x)) ‘ {ln a..ln b} ⊆ {A..B}
      using T by meson
    moreover have (λx. L (exp x)) ‘ {ln a..ln b} = L ‘ {a..b}
      unfolding h-def image-image[symmetric]
      by(simp add: ** lnab tab0)
    ultimately show ∃ A>0. ∃ B. L ‘ {a..b} ⊆ {A..B}
      by fast
  qed
  show ∀F x in at-top. ∀ a≥x. ∀ b≥a. bounded (L ‘ {a..b})
    by(intro eventually-mono[OF ***])
      (meson bounded-closed-interval bounded-subset)
qed

lemma
  assumes rv:regularly-varying U ϱ
  defines h ≡ λx. ln (U (exp x))
  shows regularly-varying-eventually-bounded-interval:
    ∀F x in at-top. ∀ a≥x. ∀ b. bounded (U ‘ {a..b})
    and regularly-varying-eventually-bounded-interval-strong:
    ∀F x in at-top. ∀ a≥x. ∀ b. ∃ A>0. ∃ B. U ‘ {a..b} ⊆ {A..B}
    and regularly-varying-eventually-bounded-interval-ln-exp:
    ∀F x in at-top. ∀ a≥x. ∀ b. bounded (h ‘ {a..b})
    and regularly-varying-eventually-locally-integrable:
    ∀F x in at-top. ∀ a≥x. ∀ b. set-integrable lborel {a..b} U
    and regularly-varying-eventually-locally-integrable-ln-exp:
    ∀F x in at-top. ∀ a≥x. ∀ b. set-integrable lborel {a..b} h
proof −
  define L where L ≡ λx. U x / x powr ϱ
  define h' where h' ≡ λx. ln (L (exp x))
  have sv:slowly-varying L
    by(simp add: regularly-varying-iff-slowly-varying-divide-powr[of U,symmetric]
L-def rv)
  have U-L:⋀x. 0 < x ⟹ U x = x powr ϱ * L x
    by(auto simp: L-def)
  have h-h':h = (λx. ϱ * x + h' x)
    using regularly-varying-pos[OF sv exp-gt-zero]
    by(auto simp add: h'-def h-def U-L ln-mult less-le)
  hence 1:∀F x in at-top. ∀ a≥x. ∀ b≥a. bounded (L ‘ {a..b})
```

and *2*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b{\geq}a.\ \exists\, A{>}0.\ \exists\, B.\ L\ `\ \{a..b\} \subseteq \{A..B\}$
and *3*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b{\geq}a.$ *bounded* $(h'\ `\ \{a..b\})$
using *slowly-varying-eventually-bounded-interval*
slowly-varying-eventually-bounded-interval-strong
slowly-varying-eventually-bounded-interval-ln-exp sv
by(*auto simp*: *h'-def*)
have ∗:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(L\ `\ \{a..b\})$
by(*intro eventually-mono*[*OF 1*])
(*metis atLeastatMost-empty-iff2 bounded-closed-interval image-empty*)
have ∗∗:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.\ \exists\, A{>}0.\ \exists\, B.\ L\ `\ \{a..b\} \subseteq \{A..B\}$
by(*intro eventually-mono*[*OF 2*])
(*metis atLeastatMost-empty' empty-iff exp-gt-zero image-subset-iff*)
have ∗∗∗:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(h'\ `\ \{a..b\})$
by(*intro eventually-mono*[*OF 3*])
(*metis atLeastatMost-empty-iff2 bounded-closed-interval image-empty*)+
have *bdd1*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $((\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\})$
by(*auto intro*!: *compact-imp-bounded compact-continuous-image continuous-on-powr*
eventually-at-top-linorderI[**where** *c=1*])
have *bdd2*:*bounded* $((\lambda x.\ \varrho * x)\ `\ \{a..b\})$ **for** *a b*
by (*metis bounded-closed-interval bounded-scaling image-cong real-scaleR-def*)
have *U'*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $((\lambda x.\ x\ powr\ \varrho * L\ x)\ `\ \{a..b\})$
by(*intro eventually-mono*[*OF eventually-conj*[*OF* ∗ *bdd1*]] *bounded-times-comp allI impI*)
auto
show *g1*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(U\ `\ \{a..b\})$
proof(*intro eventually-at-top-linorderI*[**where** *c=1*,*THEN eventually-mp*[*OF - U'*]] *impI allI*)
fix *x a b* :: *real*
assume *h*:$1 \leq x\ \forall\, a{\geq}x.\ \forall\, b.$ *bounded* $((\lambda x.\ x\ powr\ \varrho * L\ x)\ `\ \{a..b\})\ x \leq a$
then have $(\lambda x.\ x\ powr\ \varrho * L\ x)\ `\ \{a..b\} = U\ `\ \{a..b\}$
using *U-L* **by** *auto*
thus *bounded* $(U\ `\ \{a..b\})$
using *h* **by** *metis*
qed
show *g2*:$\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(h\ `\ \{a..b\})$
using *bdd2* **by**(*auto simp*: *h-h' intro*!: *bounded-plus-comp eventually-mono*[*OF* ∗∗∗])
show $\forall_F$ *x in at-top.* $\forall\, a{\geq}x.\ \forall\, b.$ *set-integrable lborel* $\{a..b\}$ *U*
proof(*safe intro*!: *eventually-at-top-linorderI*[**where** *c=1*,*THEN eventually-mp*[*OF - g1*]])
fix *a b x* :: *real*
assume *h*:$1 \leq x\ \forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(U\ `\ \{a..b\})\ x \leq a$
then obtain *K* **where** $\forall\, x{\in}\{a..b\}.\ |U\ x| \leq K$
by(*fastforce simp*: *bounded-iff*)
thus *set-integrable lborel* $\{a..b\}$ *U*
using *regularly-varying-meas-Icc*[*OF rv*] *h*
by(*auto simp*: *set-integrable-def emeasure-lborel-Icc-eq*
intro!: *integrableI-bounded-set*[**where** *A*=$\{a..b\}$ **and** *B*=*K*])
qed

show $\forall_F\ x$ *in at-top*. $\forall\, a{\geq}x.\ \forall\, b.$ *set-integrable lborel* $\{a..b\}\ h$
proof(*safe intro*!: *eventually-mono*[*OF g2*])
fix $a\ b\ x$ :: *real*
assume $h{:}\forall\, a{\geq}x.\ \forall\, b.$ *bounded* $(h\ `\ \{a..b\})\ x \leq a$
then obtain K **where** $\forall\, x{\in}\{a..b\}.\ |h\ x| \leq K$
by(*fastforce simp*: *bounded-iff*)
thus *set-integrable lborel* $\{a..b\}\ h$
using *regularly-varying-meas-log-exp*[*OF rv*]
by(*auto simp*: *set-integrable-def emeasure-lborel-Icc-eq h-def*
intro!: *integrableI-bounded-set*[**where** $A{=}\{a..b\}$ **and** $B{=}K$])
qed
have $****{:}\forall_F\ x$ *in at-top*. $\forall\, a{\geq}x.\ \forall\, b{\geq}a.\ \exists\, A{>}0.\ \exists\, B.\ U\ `\ \{a..b\} \subseteq \{A..B\}$
proof(*safe intro*!: *eventually-mp*[*OF eventually-at-top-linorderI*[*of 1*] $**$])
fix $x\ a\ b$ :: *real*
assume *xab*:$1 \leq x\ x \leq a\ a \leq b\ \forall\, a{\geq}x.\ \forall\, b.\ \exists\, A{>}0.\ \exists\, B.\ L\ `\ \{a..b\} \subseteq \{A..B\}$
then obtain $A\ B$ **where** $A{:}A > 0$ **and** $AB{:}L\ `\ \{a..b\} \subseteq \{A..B\}$
by *blast*
have $a{:}0 < a$
using *xab* **by** *linarith*
have $\exists\, C{>}0.\ \exists\, D.\ (\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\} = \{C..D\}$
proof(*cases* $0 \leq \varrho$)
assume $0 \leq \varrho$
then have $(\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\} = \{a\ powr\ \varrho..b\ powr\ \varrho\}$
using *xab*
by(*intro image-mono-on-continuous-on-Icc xab mono-onI*
powr-mono2 continuous-on-powr) *auto*
thus *?thesis*
using *a*
by(*intro exI*[**where** $x{=}a\ powr\ \varrho$] *conjI*[*OF - exI*[**where** $x{=}b\ powr\ \varrho$]]) *auto*
next
assume $\neg\ 0 \leq \varrho$
then have $(\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\} = \{b\ powr\ \varrho..a\ powr\ \varrho\}$
using *xab*
by(*intro image-antimono-on-continuous-on-Icc monotone-onI powr-mono2'*
continuous-on-powr) *auto*
thus *?thesis*
using *xab*
by(*intro exI*[**where** $x{=}b\ powr\ \varrho$] *conjI*[*OF - exI*[**where** $x{=}a\ powr\ \varrho$]]) *auto*
qed
then obtain $C\ D$ **where** $CD{:}0 < C\ (\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\} = \{C..D\}$
by *blast*
have $U\ `\ \{a..b\} \subseteq (\lambda(x,\ y).\ x * y)\ `\ ((\lambda x.\ x\ powr\ \varrho)\ `\ \{a..b\} \times L\ `\ \{a..b\})$
using *a U-L* **by** *auto*
also have $... \subseteq (\lambda(x,\ y).\ x * y)\ `\ (\{C..D\} \times \{A..B\})$
using *AB CD* **by** *fast*
also have $... \subseteq \{C{*}A..D{*}B\}$
using *A CD mult-mono* **by** *fastforce*
finally show $\exists\, A{>}0.\ \exists\, B.\ U\ `\ \{a..b\} \subseteq \{A..B\}$
using *A CD* **by**(*intro exI*[**where** $x{=}C{*}A$]) *auto*

qed
show $\forall_F\ x$ *in at-top*. $\forall\, a{\geq}x.\ \forall\, b.\ \exists\, A{>}0.\ \exists\, B.\ U\ `\ \{a..b\} \subseteq \{A..B\}$
by(*rule eventually-mono*[*OF* ****]])
(*metis atLeastatMost-empty-iff dual-order.refl empty-iff image-subset-iff*)
qed

2.3 Weakening the Definition of Regularly Varying Functions

We prove Proposition 2.3 in [3]. In the definition of a regularly varying function, $U(tx)/U(t)$ needs to converge to x^ρ as $t \to \infty$. However, we can weaken this assumption to: $U(tx)/U(t)$ converges to some function h.

lemma
assumes [*measurable*]:$U \in$ *borel-measurable* (*restrict-space borel* $\{0{<}..\}$)
and U:$\bigwedge x.\ 0 < x \Longrightarrow 0 < U\ x$
and *lim*:$\bigwedge x.\ x > 0 \Longrightarrow ((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow h\ x)$ *at-top*
shows *divide-limit-power*:$\bigwedge x.\ x > 0 \Longrightarrow h\ x = x\ powr\ ln\ (h\ (exp\ 1))$
and *divide-limit-imp-regularly-varying*:*regularly-varying U* $(ln\ (h\ (exp\ 1)))$
and *divide-limit-imp-regularly-varying-ex*:$\exists\, \varrho.$ *regularly-varying U* ϱ
proof −
have *lim'*:$0 < x \Longrightarrow 0 < y \Longrightarrow$
$((\lambda t.\ U\ (t * x * y)\ /\ U\ (t * x)) \longrightarrow h\ y)$ *at-top* **for** $x\ y$
by(*rule filterlim-compose*[*of* - - *at-top* $\lambda t.\ t * x$])
(*auto intro*!: *lim eventually-at-top-linorderI*[**where** *c*=- / *x*]
simp: *divide-le-eq filterlim-at-top*)
have *cong*:$\bigwedge x\ y.\ 0 < x \Longrightarrow 0 < y \Longrightarrow$
$\forall_F\ t$ *in at-top*. $U\ (t * (x * y))\ /\ U\ t$
$= U\ (t * x)\ /\ U\ t * (U\ (t * x * y)\ /\ U\ (t * x))$
using *U* **by**(*intro eventually-at-top-linorderI*[**where** *c*=*1*])
(*simp add*: *mult.assoc less-le*)
have *lim1*:$\bigwedge x\ y.\ 0 < x \Longrightarrow 0 < y \Longrightarrow$
$((\lambda t.\ U\ (t * (x * y))\ /\ U\ t) \longrightarrow h\ (x * y))$ *at-top*
using *lim* **by** *simp*
have *lim2*:$\bigwedge x\ y.\ 0 < x \Longrightarrow 0 < y \Longrightarrow$
$((\lambda t.\ U\ (t * x)\ /\ U\ t * (U\ (t * x * y)\ /\ U\ (t * x))) \longrightarrow h\ x * h\ y)$ *at-top*
by(*intro tendsto-mult lim lim'*) *simp-all*
have *h-eq*:$0 < x \Longrightarrow 0 < y \Longrightarrow h\ (x * y) = h\ x * h\ y$ **for** $x\ y$
using *lim1*[*of x y*] *lim2*[*of x y*]
by(*intro tendsto-unique*) (*auto simp*: *tendsto-cong*[*OF cong*[*of x y*]])
have [*measurable*]:$(*)\ r \in$ *restrict-space borel* $\{0{::}real{<}..\} \to_M$ *restrict-space borel* $\{0{<}..\}$
if *r*:$r > 0$ **for** *r*
using *r* **by**(*auto intro*!: *measurable-restrict-space3*)
have *h-meas*[*measurable*]:$h \in$ *borel-measurable* (*restrict-space borel* $\{0{<}..\}$)
by(*rule borel-measurable-LIMSEQ-metric-at-top*[**where** $f{=}\lambda t\ x.\ U\ (t * x)\ /\ U\ t$ **and** *K*=*1*])
(*auto intro*!: *lim*)

```
  have h-pos:0 < h x if x:0 < x for x
  proof(rule ccontr)
    assume cc:¬ 0 < h x
    moreover have 0 ≤ h x
      by(rule tendsto-lowerbound[OF lim[OF x]])
        (use U in auto intro!: eventually-at-top-linorderI[where c=1]
                               order.strict-implies-order simp: x)
    ultimately have h x = 0
      by linarith
    hence h 1 = 0
      using h-eq[of x 1 / x] x by simp
    hence ((λt. U t / U t) ⟶ 0) at-top
      using lim[of 1] by auto
    moreover have ((λt. U t / U t) ⟶ 1) at-top
      by(intro tendstoI eventually-at-top-linorderI[where c=1])
        (metis U dist-self divide-self less-irrefl order-less-le-trans zero-less-one)
    ultimately show False
      using tendsto-unique by force
  qed
  hence h 1 = 1
    using h-eq[of 1 1] by (metis less-irrefl mult-cancel-right1 zero-less-one)
  define g where g ≡ λr. ln (h (exp r))
  have g-meas:g ∈ borel-measurable borel
    unfolding g-def
    by(rule measurable-compose[where N=borel,OF measurable-compose
          [where g=h and N=restrict-space borel {0<..}]])
      (auto simp: measurable-restrict-space2-iff)
  have g-eq:g (x + y) = g x + g y for x y
     by(simp add: g-def exp-add h-eq[OF exp-gt-zero exp-gt-zero] ln-mult-pos[OF
h-pos h-pos])
  have g = (λx. x * g 1)
    using g-meas g-eq Cauchy-equation-measurable' by blast
  hence x * g 1 = ln (h (exp x)) for x
    by (metis g-def)
  hence x > 0 ⟹ ln x * g 1 = ln (h x) for x
    by simp
  thus hx:x > 0 ⟹ h x = x powr ln (h (exp 1)) for x
    by (simp add: h-pos powr-def g-def mult.commute)
  show regularly-varying U (ln (h (exp 1)))
    using assms by(auto intro!: regularly-varyingI simp: hx[symmetric])
  thus ∃ϱ. regularly-varying U ϱ
    by blast
qed

lemma
  assumes mono:mono-on {0<..} U and U-pos: ⋀x. 0 < x ⟹ 0 < U x
    and b-pos:∀F n in sequentially. 0 < b n
    and l-pos:∀F n in sequentially. 0 < l n
    and lim-b:filterlim b at-top sequentially
```

and *l-equiv*:$l \sim[sequentially]\ (\lambda n.\ l\ (Suc\ n))$
and *lim*:$\bigwedge x.\ 0 < x \Longrightarrow (\lambda n.\ l\ n * U\ (b\ n * x)) \longrightarrow h\ x$
and *h-pos*: $\bigwedge x.\ 0 < x \Longrightarrow 0 < h\ x$
shows *regularly-varying-sequentialy-lim*:
$\bigwedge x.\ 0 < x \Longrightarrow ((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow h\ x\ /\ h\ 1)\ at\text{-}top$
and *regularly-varying-sequentialy*:*regularly-varying U* $(ln\ (h\ (exp\ 1)\ /\ h\ 1))$
and *regularly-varying-sequentialy-ex*:$\exists \varrho.$ *regularly-varying U* ϱ
proof −
have *l-Spos*:$\forall_F\ n$ *in sequentially.* $0 < l\ (Suc\ n)$
using *l-pos le-Suc-eq* **by**(*auto simp*: *eventually-sequentially*)
have *lim-l*:$(\lambda n.\ l\ n\ /\ l\ (Suc\ n)) \longrightarrow 1$
using *l-pos eventually-mono*
by(*fastforce intro*!: *asymp-equivD-strong*[*OF l-equiv*])
have *lim-S*:$\bigwedge x.\ 0 < x \Longrightarrow (\lambda n.\ l\ (Suc\ n) * U\ (b\ (Suc\ n) * x)) \longrightarrow h\ x$
using *lim* **by**(*auto intro*!: *LIMSEQ-Suc*)
define *n* **where** $n \equiv (\lambda t.\ LEAST\ m.\ t < b\ (m + 1))$
have *n-less-b*:$t < b\ (n\ t + 1)$ **for** *t*
unfolding *n-def*
by(*rule LeastI-ex*)
(*metis lim-b eventually-sequentially filterlim-at-top-dense nat-le-iff-add*)
have *b-le-n*:$b\ (n\ t) \leq t$ **if** *t*:$b\ 0 \leq t$ **for** *t*
by(*rule ccontr*)
(*metis Least-le Suc-n-not-le-n n-def not0-implies-Suc not-le Suc-eq-plus1 t*)
have *lim-n*:*filterlim n sequentially at-top*
proof(*rule ccontr*)
assume ¬ *filterlim n sequentially at-top*
then obtain *M* **where** *M'*:$\bigwedge T.\ \exists t{\geq}T.\ n\ t + 1 \leq M + 1$
unfolding *filterlim-at-top eventually-at-top-linorder*
by (*meson add-right-mono linear*)
define *bM* **where** $bM \equiv Max\ (b\ `\ \{..M + 1\})$
have $\bigwedge T.\ \exists t{\geq}T.\ b\ (n\ t + 1) \leq bM$
using *M'* **unfolding** *bM-def*
by (*meson Max.coboundedI atMost-iff finite-atMost finite-imageI image-eqI*)
then obtain *t* **where** $bM \leq t\ b\ (n\ t + 1) \leq bM$
by *blast*
thus *False*
using *n-less-b*[*of t*] **by** *linarith*
qed
have *b-pos'*:$\forall_F\ t$ *in at-top.* $0 < b\ (n\ t)$
and *l-pos'*:$\forall_F\ t$ *in at-top.* $0 < l\ (n\ t)$
and *l-Spos'*:$\forall_F\ t$ *in at-top.* $0 < l\ (Suc\ (n\ t))$
and *lim-b'*:*filterlim* $(\lambda t.\ b\ (n\ t))\ at\text{-}top\ at\text{-}top$
and *lim-l'*:$((\lambda t.\ l\ (n\ t)\ /\ l\ (Suc\ (n\ t))) \longrightarrow 1)\ at\text{-}top$
and *lim'*:$\bigwedge x.\ 0 < x \Longrightarrow ((\lambda t.\ l\ (n\ t) * U\ (b\ (n\ t) * x)) \longrightarrow h\ x)\ at\text{-}top$
and *lim-S'*:$\bigwedge x.\ 0 < x \Longrightarrow ((\lambda t.\ l\ (Suc\ (n\ t)) * U\ (b\ (Suc\ (n\ t)) * x)) \longrightarrow h\ x)\ at\text{-}top$
by(*auto intro*!: *b-pos l-pos lim-b lim-l eventually-compose-filterlim*[*OF - lim-n*] *filterlim-compose*[*OF - lim-n*] *l-Spos lim lim-S*)
show $*$:$((\lambda t.\ U\ (t * x)\ /\ U\ t) \longrightarrow h\ x\ /\ h\ 1)\ at\text{-}top$ **if** *x*:$0 < x$ **for** *x*

```
  proof(rule tendsto-sandwich)
    show ∀F t in at-top. (l (Suc (n t)) / l (n t))
                         * (l (n t) * U (b (n t) * x) / (l (Suc (n t)) * U (b (Suc (n
t)))))
                        ≤ U (t * x) / U t (is ∀F t in at-top. ?lhs t ≤ -)
    proof(intro impI eventually-mp[OF eventually-at-top-linorderI[where c=max
(b 0) 1]
          eventually-conj[OF b-pos' eventually-conj[OF l-pos' l-Spos']]])
      fix t
      assume t: max (b 0) 1 ≤ t and h:0 < b (n t) ∧  0 < l (n t) ∧ 0 < l (Suc
(n t))
      have ?lhs t = U (b (n t) * x) / U (b (Suc (n t)))
        using h by simp
      also have ... ≤ U (t * x) / U t
        by(rule frac-le)
          (use U-pos x h t n-less-b[of t] mono-onD[OF mono] b-le-n[of t]
              in auto intro!: order.strict-implies-order[of 0])
      finally show ?lhs t ≤ U (t * x) / U t .
    qed
  next
    show ∀F t in at-top. U (t * x) / U t
                        ≤ (l (n t) / l (Suc (n t)))
                        * (l (Suc (n t)) * U (b (Suc (n t)) * x) / (l (n t) * U (b (n
t))))
                         (is ∀F t in at-top. - ≤ ?rhs t)
    proof(intro impI eventually-mp[OF eventually-at-top-linorderI[where c=max
(b 0) 1]
          eventually-conj[OF b-pos' eventually-conj[OF l-pos' l-Spos']]])
      fix t
      assume t: max (b 0) 1 ≤ t and h:0 < b (n t) ∧  0 < l (n t) ∧ 0 < l (Suc
(n t))
      have U (t * x) / U t ≤ U (b (Suc (n t)) * x) / U (b (n t))
        by(rule frac-le)
          (use U-pos x h t n-less-b[of t] mono-onD[OF mono] b-le-n[of t]
              in auto intro!: order.strict-implies-order[of 0])
      also have ... ≤ ?rhs t
        using h by simp
      finally show U (t * x) / U t ≤ ?rhs t .
    qed
  next
    have 1:((λt. l (Suc (n t)) / l (n t)) ⟶ 1) at-top
      using tendsto-divide[OF tendsto-const[of 1] lim-l'] by simp
    have 2:((λt. l (n t) * U (b (n t) * x)) ⟶ h x) at-top
      using x lim' by simp
    have 3:((λt. l (Suc (n t)) * U (b (Suc (n t)))) ⟶ h 1) at-top
      using lim-S'[of 1] by simp
    show ((λt. l (Suc (n t)) / l (n t) *
              (l (n t) * U (b (n t) * x) / (l (Suc (n t)) * U (b (Suc (n t))))))
          ⟶ h x / h 1) at-top
```

```
    using tendsto-mult[OF 1 tendsto-divide[OF 2 3]] h-pos[of 1] by argo
  next
    thm lim-l'
    have 2:((λt. l (Suc (n t)) * U (b (Suc (n t)) * x)) ⟶ h x) at-top
      using x lim-S' by simp
    have 3:((λt. l (n t) * U (b (n t))) ⟶ h 1) at-top
      using lim'[of 1] by simp
    show ((λt. l (n t) / l (Suc (n t)) *
          (l (Suc (n t)) * U (b (Suc (n t)) * x) / (l (n t) * U (b (n t))))))
           ⟶ h x / h 1) at-top
      using tendsto-mult[OF lim-l' tendsto-divide[OF 2 3]] h-pos[of 1] by argo
  qed
  show regularly-varying U (ln (h (exp 1) / h 1))
    by(auto intro!: divide-limit-imp-regularly-varying
        borel-measurable-mono-on-fnc mono U-pos *)
  thus ∃ ϱ. regularly-varying U ϱ
    by blast
qed

lemma regularly-varying-integral-infinity:
  assumes rv:regularly-varying U ϱ
  assumes rho:- 1 < ϱ
  shows (∫⁺ t∈{a<..}. ennreal (U t) ∂lborel) = ⊤
proof -
  have [measurable]:⋀n. (λx. ennreal (U x) * indicator {2^n<..2 * 2^n} x)
                        ∈ borel-measurable borel
    (λx. ennreal (U x) * indicator {0<..} x) ∈ borel-measurable borel
    by(auto intro!: regularly-varying-meas-Ioc-ennreal
                    regularly-varying-meas-o-ennreal rv)
  have 2 powr ϱ > 1 / 2
   by (metis rho powr-less-cancel-iff powr-neg-one rel-simps(49,51) semiring-norm(76))
  with regularly-varying-lim[where x=2,OF rv]
  obtain s' where s':⋀t. s' ≤ t ⟹ 1 / 2 < U (t * 2) / U t
    by (metis (lifting) eventually-at-top-linorder rel-simps(51) order-tendstoD(1))
  obtain s'' where s'': ⋀t k. s'' ≤ t ⟹ set-integrable lborel {t..k} U
    using regularly-varying-eventually-locally-integrable[OF rv]
    by (metis (lifting) eventually-at-top-linorder)
  define s where s ≡ max (max 1 a) (max s' s'')
  have s-pos:0 < s and s:⋀t. s ≤ t ⟹ U t < 2 * U (t * 2)
    and s2:⋀t k. s ≤ t ⟹ set-integrable lborel {t..k} U
    using s' s'' regularly-varying-pos[OF rv] by(fastforce simp: s-def)+
  have as:a ≤ s
    by(simp add: s-def)
  obtain N where N:s ≤ 2 ^ N
    using power-of-nat-log-ge [of 2 s] by auto
  hence N':⋀n. N ≤ n ⟹ 2 ^ n ≤ t ⟹ U t < 2 * U (t * 2) for t
    using s[of t] s-pos order.trans less-num-simps
    by (metis power-increasing power-one-right two-realpow-ge-one)
  have integrable:N ≤ n ⟹ set-integrable lborel {2^n<..2 * 2^n} U for n
```

```
    using s2[OF order.trans[OF N]]
   by(auto intro!: set-integrable-subset[where A={2^n..2 * 2^n} and B={2^n<..2
* 2^n}])
  have less':(∫⁺ t∈{2^n<..2 * 2^n}. U t ∂lborel) < (∫⁺ t∈{2*2^n<..4 * 2^n}.
U t ∂lborel)
    if n:N ≤ n for n
  proof −
    have (∫⁺ t∈{2^n<..2 * 2^n}. ennreal (U t) ∂lborel)
          < (∫⁺ t∈{2^n<..2 * 2^n}. ennreal (2 * U (t * 2)) ∂lborel)
    proof(rule set-nn-integral-less)
      have (∫⁺x∈{2 ^ n<..2 * 2 ^ n}. ennreal (U x) ∂lborel)
            ≤ (∫⁺x. ennreal |indicat-real {2 ^ n<..2 * 2 ^ n} x * U x| ∂lborel)
        by(auto intro!: nn-integral-mono arg-cong[where f=ennreal]
         simp: ennreal-indicator[symmetric] ennreal-mult''[OF indicator-pos-le,symmetric]
           mult.commute)
      also have ... < ∞
      using integrable[OF n] by(auto simp: set-integrable-def integrable-iff-bounded)
      finally show (∫⁺x∈{2 ^ n<..2 * 2 ^ n}. ennreal (U x) ∂lborel) ≠ ∞
        by order
    next
      show AE x∈{2 ^ n<..2 * 2 ^ n} in lborel. ennreal (U x)
                          < ennreal (2 * U (x * 2))
        by(fastforce intro!: ennreal-lessI N'[OF n] regularly-varying-pos[OF rv])
      show (λx. ennreal (2 * U (x * 2)) * indicator {2 ^ n<..2 * 2 ^ n} x)
            ∈ borel-measurable lborel
        by(auto intro!: regularly-varying-meas-Ioc-ennreal
                        regularly-varying-comp-multc regularly-varying-cmult rv)
    qed auto
    also have ... = (∫⁺x∈{2 ^ n<..2 * 2 ^ n}. ennreal (|2| * U (x * 2)) ∂lborel)
      by simp
   also have ... = (∫⁺x∈(λx. x * 2) ‘ {2 ^ n<..2 * 2 ^ n}. ennreal (U x) ∂lborel)
      by(rule nn-integral-change-of-variables-real1'[symmetric])
        (use integrable[of n + 1] n in
         auto intro!: measurable-restrict-space1 inj-on-mult less-imp-le
                      regularly-varying-pos[OF rv] simp: mult-commute-abs)
    also have ... = (∫⁺ t∈{2*2^n<..4 * 2^n}. U t ∂lborel)
      by simp
    finally show ?thesis .
  qed
  have less:(∫⁺ t∈{2^N<..2 * 2^N}. U t ∂lborel) < (∫⁺ t∈{2*2^n<..4 * 2^n}.
U t ∂lborel)
    if n:N ≤ n for n
  proof(rule nat-induct-at-least[OF n])
    fix m
    assume m:N ≤ m
      and (∫⁺x∈{2 ^ N<..2 * 2 ^ N}. ennreal (U x) ∂lborel)
          < (∫⁺x∈{2 * 2 ^ m<..4 * 2 ^ m}. ennreal (U x) ∂lborel)
    hence (∫⁺x∈{2 ^ N<..2 * 2 ^ N}. ennreal (U x) ∂lborel)
          < (∫⁺x∈{2 * 2 ^ m<..4 * 2 ^ m}. ennreal (U x) ∂lborel)
```

by *simp*
also have ... = $(\int^+ x{\in}\{2 \mathbin{\hat{}} Suc\ m{<}..2 * 2 \mathbin{\hat{}} Suc\ m\}.\ ennreal\ (U\ x)\ \partial lborel)$
by *simp*
also have ... < $(\int^+ x{\in}\{2 * 2 \mathbin{\hat{}} Suc\ m{<}..4 * 2 \mathbin{\hat{}} Suc\ m\}.\ ennreal\ (U\ x)\ \partial lborel)$
by(*intro less'*) (*use m* **in** *auto*)
finally show $(\int^+ x{\in}\{2 \mathbin{\hat{}} N{<}..2 * 2 \mathbin{\hat{}} N\}.\ ennreal\ (U\ x)\ \partial lborel)$
$< (\int^+ x{\in}\{2 * 2 \mathbin{\hat{}} Suc\ m{<}..4 * 2 \mathbin{\hat{}} Suc\ m\}.\ ennreal\ (U\ x)\ \partial lborel)$.
qed(*use less'* **in** *auto*)
have *disj*:*disjoint-family* $(\lambda n.\ \{2 * 2 \mathbin{\hat{}} (n + N) {<}..4 * 2 \mathbin{\hat{}} (n + N)::real\})$
unfolding *disjoint-family-on-def*
proof *safe*
fix *m n x*
assume *mn*:$m \neq n$
and *x*:$x \in \{2 * 2 \mathbin{\hat{}} (m + N){<}..4 * 2 \mathbin{\hat{}} (m + N)::real\}$
$x \in \{2 * 2 \mathbin{\hat{}} (n + N){<}..4 * 2 \mathbin{\hat{}} (n + N)\}$
have *1*:$m < n \Longrightarrow (2 :: real) \mathbin{\hat{}} (m + 1 + (N + 1)) \leq 2 \mathbin{\hat{}} (n + N + 1)$
by(*intro power-increasing*) (*auto simp*: *less-eq-Suc-le*)
have *2*:$n < m \Longrightarrow (2 :: real) \mathbin{\hat{}} (n + 1 + (N + 1)) \leq 2 \mathbin{\hat{}} (m + N + 1)$
by(*intro power-increasing*) (*auto simp*: *less-eq-Suc-le*)
consider $m < n \mid n < m$
using *mn* **by** *linarith*
thus $x \in \{\}$
by *cases* (*use 1 2 x* **in** *auto*)
qed
have *subst*:$(\bigcup m.\ \{2{*}2\hat{}(m + N){<}..4 * 2\hat{}(m + N)\}) \subseteq \{a{<}..\}$
proof *safe*
fix *x m*
assume *x*: $x \in \{2 * 2 \mathbin{\hat{}} (m + N){<}..4 * 2 \mathbin{\hat{}} (m + N) :: real\}$
have $(2 :: real) \mathbin{\hat{}} N \leq 2 * 2 \mathbin{\hat{}} (m + N)$
by(*auto simp del*: *power-Suc simp*: *power-Suc*[*symmetric*])
thus $a < x$
using *as N x* **by** *simp*
qed
have $\top = (\sum m.\ (\int^+ t{\in}\{2 \mathbin{\hat{}} N{<}..2 * 2 \mathbin{\hat{}} N\}.\ ennreal\ (U\ t)\ \partial lborel))$
by(*intro suminf-const-infinity*[*symmetric*] *set-nn-integral-zero-less AE-I2*)
(*use regularly-varying-pos*[*OF rv*] **in** *auto*)
also have ... $\leq (\sum m.\ (\int^+ t{\in}\{2{*}2\hat{}(m + N){<}..4 * 2\hat{}(m + N)\}.\ U\ t\ \partial lborel))$
by(*auto intro*!: *suminf-le less-imp-le*[*OF less*])
also have ... =
$(\sum m.\ (\int^+ t{\in}\{2{*}2\hat{}(m + N){<}..4 * 2\hat{}(m + N)\}.\ ennreal\ (U\ t) * indicator\ \{0{<}..\}\ t\ \partial lborel))$
by(*auto intro*!: *suminf-cong nn-integral-cong ennreal-mult-right-cong*
simp: *indicator-eq-0-iff*)
also have ... =
$(\int^+ t{\in}\bigcup m.\ \{2{*}2\hat{}(m + N){<}..4 * 2\hat{}(m + N)\}.\ ennreal\ (U\ t) * indicator\ \{0{<}..\}\ t\ \partial lborel)$
by(*rule nn-integral-disjoint-family*[*symmetric*]) (*auto simp*: *disj*)
also have ... $\leq (\int^+ t{\in}\{a{<}..\}.\ ennreal\ (U\ t)\ \partial lborel)$
using *subst* **by**(*intro nn-integral-mono*) (*fastforce simp*: *indicator-def*)

finally show *?thesis*
using *neq-top-trans* **by** *blast*
qed

2.4 Karamata's Representation Theorem

Lemma 1.3.1 in [1].

lemma *Karamata-representation-ln-exp-slowly-varying*:
fixes L
defines $h \equiv (\lambda x.\ ln\ (L\ (exp\ x)))$
assumes *sv*:*slowly-varying* L
and A: $0 < A$
and *bdd*:$\bigwedge a\ b.\ A \leq a \Longrightarrow \exists X{>}0.\ \exists Y.\ L\ `\ \{a..b\} \subseteq \{X..Y\}$
shows $\exists c\ e\ C.$
$(\forall x \geq ln\ A.\ h\ x = c\ x + (\int u{\in}\{ln\ A{<}..x\}.\ e\ u\ \partial lborel))$
$\wedge\ (\forall x \geq ln\ A.\ set\text{-}integrable\ lborel\ \{ln\ A{<}..x\}\ e)$
$\wedge\ e \in borel\text{-}measurable\ borel$
$\wedge\ (e \longrightarrow 0)\ at\text{-}top$
$\wedge\ bounded\ (e\ `\ \{ln\ A..\})$
$\wedge\ c \in borel\text{-}measurable\ borel$
$\wedge\ (c \longrightarrow C)\ at\text{-}top$
proof −
have [*measurable*]:$h \in borel\text{-}measurable\ borel$
using *regularly-varying-meas-log-exp sv* **by**(*simp add*: *h-def*)
have *bddh*:*bounded* $(h\ `\ \{a..b\})$ **if** a:$ln\ A \leq a$ **for** $a\ b$
proof(*cases* $a \leq b$)
assume b:$a \leq b$
then obtain $A'\ B$ **where** A': $A \leq A'\ a = ln\ A'$ **and** B:$A' \leq B\ b = ln\ B$
by (*metis a exp-gt-zero landau-omega.R-trans ln-exp ln-mono nle-le*)
hence A'-*pos*:$0 < A'$ **and** B-*pos*:$0 < B$
using A **by** *auto*
obtain $X\ Y$ **where** XY:$X > 0\ X \leq Y\ L\ `\ \{A'..B\} \subseteq \{X..Y\}$
using *bdd* $A'\ B$ **by** (*meson* A'-*pos atLeastatMost-subset-iff order.trans*)
have $exp\ `\ \{a..b\} = \{A'..B\}$
unfolding $A'\ B$
by(*subst image-mono-on-continuous-on-Icc*)
(*use* $A'\ B$ A'-*pos B-pos* **in**
auto intro!: *mono-on-subset*[*OF Multiseries-Expansion-Bounds.mono-exp-real*]
continuous-on-exp)
hence $h\ `\ \{a..b\} = (\lambda x.\ ln\ (L\ x))\ `\ \{A'..B\}$
by (*metis* (*no-types*, *lifting*) *ext h-def image-image*)
also have ... $= ln\ `\ (L\ `\ \{A'..B\})$
by *blast*
also have ... $\subseteq ln\ `\ \{X..Y\}$
using XY **by** *blast*
also have ... $= \{ln\ X..ln\ Y\}$
using XY
by(*intro image-mono-on-continuous-on-Icc XY continuous-on-ln mono-onI ln-mono*)

```
      auto
  finally show ?thesis
    by (metis bounded-closed-interval bounded-subset)
 qed simp
 have integrableh:set-integrable lborel {a<..b} h if a:ln A ≤ a for a b
  unfolding set-integrable-def
 proof(safe intro!: integrableI-bounded)
  obtain K where K:⋀x. a ≤ x ⟹ x ≤ b ⟹ |h x|  ≤ K
   using bddh[OF a] by (metis atLeastAtMost-iff bounded-real image-eqI)
  have (∫⁺ x. ennreal (norm (indicat-real {a<..b} x *R h x)) ∂lborel)
      = (∫⁺ x∈{a<..b}. ennreal |h x| ∂lborel)
   by(auto intro!: nn-integral-cong simp: indicator-times-eq-if)
  also have ... ≤ (∫⁺ x∈{a..b}. ennreal K ∂lborel)
   using K by(auto intro!: nn-integral-mono simp: indicator-def ennreal-leI)
  also have ... =  ennreal K * (∫⁺ x. indicator {a..b} x ∂lborel)
   by(subst nn-integral-cmult) simp-all
  also have ... < ∞
   by (simp add: emeasure-lborel-Icc-eq ennreal-mult-less-top)
  finally show (∫⁺ x. ennreal (norm (indicat-real {a<..b} x *R h x)) ∂lborel)
< ∞ .
 qed simp
 define C where C ≡ (∫ t∈{ln A<..ln A + 1}. h t ∂lborel)
 define c where c ≡ (λx. C + (∫ t∈{x<..x+ 1}. h x − h t ∂lborel))
 define e where e ≡ (λx. h (x + 1) − h x)
 have integrableh':set-integrable lborel {ln A<..x} (λt. h (t + 1)) for x
 proof −
  have (λt. indicat-real {ln A + 1<..x + 1} (1 + t) * h (1 + t))
      = (λt. indicat-real {ln A<..x} t * h (t + 1))
   by standard (auto simp: indicator-def add.commute)
  thus ?thesis
   using integrableh[of ln A + 1 x + 1]
   by(auto simp:
     lborel-integrable-real-affine-iff[symmetric,
      where f=λt. indicat-real {ln A+1<..x+1} t * h t and t=1 and c=1]
      set-integrable-def)
 qed
 have *:∀x ≥ ln A. h x = c x + (∫ u∈{ln A<..x}. e u ∂lborel)
 proof safe
  fix x
  assume x:ln A ≤ x
  have h x = h x + (∫ t∈{ln A<..ln A + 1} ∪ {ln A + 1<.. x+ 1}. h t ∂lborel)
            − (∫ t∈{ln A<..x} ∪ {x<..x+ 1}. h t ∂lborel)
   using x by(auto intro!: arg-cong[where f=λA. set-lebesgue-integral lborel A
h])
  also have ... = h x + (∫ t∈{ln A<..ln A + 1}. h t ∂lborel)
                + (∫ t∈{ln A + 1<.. x+ 1}. h t ∂lborel)
                − (∫ t∈{ln A<..x} ∪ {x<..x+ 1}. h t ∂lborel)
   by(subst set-integral-Un) (auto intro!: integrableh)
  also have ... = h x + (∫ t∈{ln A<..ln A + 1}. h t ∂lborel)
```

$+ (\int t\in\{ln\ A + 1<..\ x+ 1\}.\ h\ t\ \partial lborel)$
$- ((\int t\in\{ln\ A<..x\}.\ h\ t\ \partial lborel)$
$+ (\int t\in\{x<..x+ 1\}.\ h\ t\ \partial lborel))$
by(*subst set-integral-Un*) (*auto intro*!: *integrableh x*)
also have ... = $h\ x + (\int t\in\{ln\ A<..ln\ A + 1\}.\ h\ t\ \partial lborel)$
$+ (\int t\in\{ln\ A<..\ x\}.\ h\ (t + 1)\ \partial lborel)$
$- ((\int t\in\{ln\ A<..x\}.\ h\ t\ \partial lborel)$
$+ (\int t\in\{x<..x+ 1\}.\ h\ t\ \partial lborel))$
unfolding *set-lebesgue-integral-def real-scaleR-def*
by(*subst lborel-integral-real-affine*
[**where** $f=\lambda t.$ *indicat-real* $\{ln\ A+1<..x+1\}\ t * h\ t$ **and** $t=1$ **and** $c=1$])
(*auto intro*!: *Bochner-Integration.integral-cong simp*: *indicator-def add.commute*)
also have ... = $C + (h\ x + (\int t\in\{x<..x+ 1\}.\ - h\ t\ \partial lborel))$
$+ ((\int t\in\{ln\ A<..\ x\}.\ h\ (t + 1)\ \partial lborel)$
$+ (\int t\in\{ln\ A<..x\}.\ - h\ t\ \partial lborel))$
by(*simp add*: *set-integral-uminus*[*OF integrableh*] *x C-def*)
also have ... = $C + ((\int t\in\{x<..x+ 1\}.\ h\ x\ \partial lborel) + (\int t\in\{x<..x+ 1\}.\ - h\ t\ \partial lborel))$
$+ ((\int t\in\{ln\ A<..\ x\}.\ h\ (t + 1)\ \partial lborel)$
$+ (\int t\in\{ln\ A<..x\}.\ - h\ t\ \partial lborel))$
by(*auto simp*: *set-lebesgue-integral-def*)
also have ... = $C + ((\int t\in\{x<..x+ 1\}.\ h\ x + (-\ h\ t)\ \partial lborel))$
$+ ((\int t\in\{ln\ A<..\ x\}.\ h\ (t + 1)\ \partial lborel)$
$+ (\int t\in\{ln\ A<..x\}.\ - h\ t\ \partial lborel))$
by(*subst set-integral-add*)
(*auto intro*!: *integrableh x set-integrable-const simp*: *set-integrable-uminus*)
also have ... = $c\ x + ((\int t\in\{ln\ A<..\ x\}.\ h\ (t + 1) + (-\ h\ t)\ \partial lborel))$
by(*subst set-integral-add*[**where** $f=\lambda t.\ h\ (t + 1)$])
(*auto intro*!: *integrableh x integrableh'* *simp*: *set-integrable-uminus c-def*)
finally show $h\ x = c\ x + (\int u\in\{ln\ A<..x\}.\ e\ u\ \partial lborel)$
by(*simp add*: *e-def*)
qed
have $**$:$\forall\ x\geq ln\ A.$ *set-integrable lborel* $\{ln\ A<..x\}\ e$
by(*auto simp*: *e-def intro*!: *integrableh' integrableh*)
have $***$:$e \in$ *borel-measurable borel*
by(*simp add*: *e-def*)
have $****$:$(e \longrightarrow 0)$ *at-top*
using *slowly-varying-ln-exp*[*OF sv*] **by**(*simp add*: *e-def h-def*)
have $*****$:*bounded* $(e\ `\ \{ln\ A..\})$
proof −
obtain X **where** X:$\bigwedge x.\ X < x \Longrightarrow |e\ x| < 1$
using *tendstoD*[*OF* $***$,*of 1*] **unfolding** *eventually-at-top-dense*
by *auto*
have *1*:*bounded* $(e\ `\ \{ln\ A..X\})$
unfolding *e-def*
by(*intro bounded-minus-comp bounded-subset*[**where** $T=h\ `\ \{ln\ A..X\}$ **and** $S=h\ `\ \{ln\ A<..X\}$]
bounded-subset[**where** $T=(\lambda x.\ h\ (x + 1))\ `\ \{ln\ A..X\}$
and $S=(\lambda x.\ h\ (x + 1))\ `\ \{ln\ A<..X\}$])

```
      (auto simp: image-image[symmetric,where g=λx. x + 1] intro!: bddh)
    have 2:bounded (e ‘ {X<..})
      using X by(fastforce intro!: boundedI[where B=2])
    show ?thesis
      by(rule bounded-subset[where T=e ‘ {ln A..X} ∪ e ‘ {X<..}])
        (use 1 2 in fastforce)+
  qed
  have ****:c ∈ borel-measurable borel
  proof −
    have Measurable.pred (borel ⨂M borel) (λx::real × real. snd x ∈ {fst x<..fst
x + 1})
      by simp
    thus ?thesis
      unfolding c-def set-lebesgue-integral-def
      by measurable
  qed
  have (c ⟶ C) at-top
  proof −
    have ∀F x in at-top. (∫ t∈{x<..x+ 1}. h x − h t ∂lborel)
                       = (∫ t∈{0<..1}. h x − h (x + t) ∂lborel)
    proof(safe intro!: eventually-at-top-linorderI[where c=ln A])
      fix x
      assume x:ln A ≤ x
      thus (∫ t∈{x<..x+ 1}. h x − h t ∂lborel) = (∫ t∈{0<..1}. h x − h (x + t)
∂lborel)
        unfolding set-lebesgue-integral-def
        by(subst lborel-integral-real-affine[where c=1 and t=x])
          (auto intro!: Bochner-Integration.integral-cong simp: indicator-def)
    qed
    moreover have ((λx. (∫ t∈{0<..1}. h x − h (x + t) ∂lborel)) ⟶ 0) at-top
    proof(rule tendstoI)
      fix e :: real
      assume e:0 < e
      define e′ where e′ ≡ e / 2
      have e′:0 < e′
        using e by(simp add: e′-def)
      obtain X where X:⋀x t. X ≤ x ⟹ 0 ≤ t ⟹ t ≤ 1 ⟹ |h x − h (x + t)|
< e′
        using uniform-limitD[OF slowly-varying-UCT-interval-ln-exp[OF sv,of 0 1]
e′]
        by(fastforce simp: h-def dist-real-def eventually-at-top-linorder abs-minus-commute)
      show ∀F x in at-top. dist (∫ t∈{0<..1}. h x − h (x + t) ∂lborel) 0 < e
      proof(safe intro!: eventually-at-top-linorderI[where c=X])
        fix x
        assume x:X ≤ x
        then have [simp]:set-integrable lborel {0<..1} (λt. |h x − h (x + t)|)
          using less-imp-le[OF X]
          by(auto intro!: integrableI-bounded-set[where A={0<..1} and B=e′]
              simp: set-integrable-def)
```

```
      have dist (∫ t∈{0<..1}. h x − h (x + t) ∂lborel) 0
            ≤ (∫ t∈{0<..1}. |h x − h (x + t)| ∂lborel)
        by(auto simp: set-lebesgue-integral-def abs-mult-pos')
      also have ... ≤ (∫ t∈{0::real<..1}. e' ∂lborel)
      by(auto intro!: set-integral-mono less-imp-le[OF X] x integrableh set-integrable-const)
      also have ... < e
        by(simp add: set-lebesgue-integral-def e'-def e)
      finally show dist (∫ t∈{0<..1}. h x − h (x + t) ∂lborel) 0 < e .
    qed
  qed
  ultimately have ((λx. (∫ t∈{x<..x+ 1}. h x − h t ∂lborel)) ⟶ 0) at-top
    by(simp cong: tendsto-cong)
  thus ?thesis
    by (metis (lifting) ext c-def group-cancel.rule0 tendsto-add-const-iff)
 qed
 thus ?thesis
   using * ** *** **** ***** ****** by blast
qed

lemma Karamata-representation-ln-exp2:
 fixes U
 defines h ≡ (λx. ln (U (exp x)))
 assumes rv:regularly-varying U ϱ
   and A: 0 < A
   and bdd:⋀a b. A ≤ a ⟹ ∃X>0. ∃Y. U ‘ {a..b} ⊆ {X..Y}
 shows ∃c e C.
          (∀x ≥ ln A. h x = c x + ϱ * ln A + (∫ u∈{ln A<..x}. ϱ +  e u ∂lborel))
          ∧ (∀x ≥ ln A. set-integrable lborel {ln A<..x} e)
          ∧ e ∈ borel-measurable borel
          ∧ (e ⟶ 0) at-top
          ∧ bounded (e ‘ {ln A..})
          ∧ c ∈ borel-measurable borel
          ∧ (c ⟶ C) at-top
proof −
 define L where L ≡ λx. U x / x powr ϱ
 define h' where h' ≡ λx. ln (L (exp x))
 have sv:slowly-varying L
   by(simp add: regularly-varying-iff-slowly-varying-divide-powr[of U,symmetric]
L-def rv)
 have U-L:⋀x. 0 < x ⟹ U x = x powr ϱ * L x
   by(auto simp: L-def)
 have h-h':h = (λx. ϱ * x + h' x)
   using regularly-varying-pos[OF sv exp-gt-zero]
   by(auto simp add: h'-def h-def U-L ln-mult less-le)
 have ∃X>0. ∃Y. L ‘ {a..b} ⊆ {X..Y} if Aa:A ≤ a for a b
 proof −
   obtain X' Y' where X'-pos:X' > 0
     and X': ⋀x. a ≤ x ⟹ x ≤ b ⟹ X' ≤ U x
     and Y':⋀x. a ≤ x ⟹ x ≤ b ⟹ U x ≤ Y'
```

```
    using bdd[OF Aa,of b] by fastforce
  consider ϱ ≤ 0 a ≤ b | 0 ≤ ϱ a ≤ b | b < a
    by linarith
  thus ?thesis
  proof cases
    assume h: ϱ ≤ 0 a ≤ b
    have X' / a powr ϱ ≤ U x / x powr ϱ
      and U x / x powr ϱ ≤ Y' / b powr ϱ if a ≤ x x ≤ b for x
      using X'[OF that] Y'[OF that] Aa A X'-pos h that
      by(auto intro!: frac-le powr-mono2')
    thus ?thesis
      using X'-pos h Aa A
      by(intro exI[where x=X' / a powr ϱ,OF conjI[OF - exI[where x=Y' / b
powr ϱ]]])
        (auto simp: L-def)
  next
    assume h: ϱ ≥ 0 a ≤ b
    have X' / b powr ϱ ≤ U x / x powr ϱ
      and U x / x powr ϱ ≤ Y' / a powr ϱ if a ≤ x x ≤ b for x
      using X'[OF that] Y'[OF that] Aa A X'-pos h that
      by(auto intro!: frac-le powr-mono2)
    thus ?thesis
      using X'-pos h Aa A
      by(intro exI[where x=X' / b powr ϱ,OF conjI[OF - exI[where x=Y' / a
powr ϱ]]])
        (auto simp: L-def)
  qed (auto intro!: exI[where x=1])
qed
from Karamata-representation-ln-exp-slowly-varying[OF sv A this]
obtain c e C where
  h'-eq:⋀x. ln A ≤ x ⟹ h' x = c x + (∫ u∈{ln A<..x}. e u ∂lborel)
  and e-integrable:⋀x. ln A ≤ x ⟹ set-integrable lborel {ln A<..x} e
  and e-meas[measurable]:e ∈ borel-measurable borel
  and e-lim:(e ⟶ 0) at-top
  and e-bdd:bounded (e ` {ln A..})
  and c-meas[measurable]:c ∈ borel-measurable borel
  and c-lim:(c ⟶ C) at-top
  by(auto simp: h'-def)
have h-eq:h x = c x + ϱ * ln A + (∫ u∈{ln A<..x}. ϱ +  e u ∂lborel) if x:ln A
≤ x for x
proof −
  have h x = h' x + ϱ * x
    by(simp add: h-h')
  also have ... = c x + (∫ u∈{ln A<..x}. e u ∂lborel) + ϱ * x
    by(simp add: x h'-eq)
  also have ... = c x + (∫ u∈{ln A<..x}. ϱ ∂lborel) + (∫ u∈{ln A<..x}. e u
∂lborel) + ϱ * ln A
    by(simp add: interval-integral-Ioc[OF x,symmetric] right-diff-distrib')
  also have ... = c x + ϱ * ln A + (∫ u∈{ln A<..x}. ϱ +  e u ∂lborel)
```

by (*rule sym*) (*auto intro*!: *e-integrable set-integrable-const simp*: *x*)
finally show *?thesis* **.**
qed
thus *?thesis*
using *e-integrable e-lim c-lim c-meas e-meas e-bdd* **by** *blast*
qed

lemma *Karamata-representation-ln-exp*:
fixes *U*
defines $h \equiv (\lambda x.\ ln\ (U\ (exp\ x)))$
assumes *rv*:*regularly-varying U* ϱ
shows $\exists\, A{>}0.\ \exists\, c\ e\ C.$
$(\forall\, x \geq ln\ A.\ h\ x = c\ x + \varrho * ln\ A + (\int u{\in}\{ln\ A{<}..x\}.\ \varrho + e\ u\ \partial lborel))$
$\wedge\ (\forall\, x \geq ln\ A.\ set\text{-}integrable\ lborel\ \{ln\ A{<}..x\}\ e)$
$\wedge\ e \in borel\text{-}measurable\ borel$
$\wedge\ (e \longrightarrow 0)\ at\text{-}top$
$\wedge\ bounded\ (e\ `\ \{ln\ A..\})$
$\wedge\ c \in borel\text{-}measurable\ borel$
$\wedge\ (c \longrightarrow C)\ at\text{-}top$
proof −
obtain *A* **where** $0 < A\ \bigwedge a\ b.\ A \leq a \Longrightarrow \exists\, X{>}0.\ \exists\, Y.\ U\ `\ \{a..b\} \subseteq \{X..Y\}$
using *regularly-varying-eventually-bounded-interval-strong*[*OF rv*]
unfolding *eventually-at-top-linorder*
by (*meson exp-gt-self exp-gt-zero less-eq-real-def*)
from *Karamata-representation-ln-exp2*[*OF rv this*] *this*
show *?thesis*
unfolding *h-def* **by** *blast*
qed

This equation for *ln* is useful for transforming equations in proofs.

lemma *Karamata-representation-ex1*:
assumes *rv*:*regularly-varying U* ϱ
and *A*:$0 < A$
and *bdd*:$\bigwedge a\ b.\ A \leq a \Longrightarrow \exists\, X{>}0.\ \exists\, Y.\ U\ `\ \{a..b\} \subseteq \{X..Y\}$
shows $\exists\, c\ r\ C.$
$(\forall\, x \geq A.\ U\ x = c\ x * exp\ (\int t{\in}\{A{<}..x\}.\ r\ t\ /\ t\ \partial lborel))$
$\wedge\ (\forall\, x \geq A.\ set\text{-}integrable\ lborel\ \{A{<}..x\}\ (\lambda t.\ r\ t\ /\ t))$
$\wedge\ r \in borel\text{-}measurable\ borel$
$\wedge\ (r \longrightarrow \varrho)\ at\text{-}top$
$\wedge\ bounded\ (r\ `\ \{A..\})$
$\wedge\ c \in borel\text{-}measurable\ borel$
$\wedge\ (\forall\, x.\ 0 < c\ x)$
$\wedge\ (c \longrightarrow C)\ at\text{-}top$
$\wedge\ 0 < C$
proof −
define *h* **where** $h \equiv \lambda x.\ ln\ (U\ (exp\ x))$
from *Karamata-representation-ln-exp2*[*OF assms*]
obtain $c'\ e\ C'$ **where**
h-eq:$\bigwedge x.\ ln\ A \leq x \Longrightarrow h\ x = c'\ x + \varrho * ln\ A + (\int u{\in}\{ln\ A{<}..x\}.\ \varrho + e\ u$

∂lborel)
and *e-integrable*:⋀*x*. *ln A* ≤ *x* ⟹ *set-integrable lborel* {*ln A*<..*x*} *e*
and *e-meas*[*measurable*]:*e* ∈ *borel-measurable borel*
and *e-lim*:(*e* ⟶ *0*) *at-top*
and *e-bdd*:*bounded* (*e* ‘ {*ln A*..})
and *c-meas*[*measurable*]:*c′* ∈ *borel-measurable borel*
and *c-lim*:(*c′* ⟶ *C′*) *at-top*
unfolding *h-def* **by** *blast*
define *C* **where** *C* ≡ *exp C′* ∗ (*A powr ϱ*)
define *c* **where** *c* ≡ λ*x*. *exp* (*c′* (*ln x*)) ∗ *A powr ϱ*
define *r* **where** *r* ≡ λ*x*. *ϱ* + *e* (*ln x*)
have *im-ln*:*A* ≤ *x* ⟹ *ln* ‘ {*A*<..*x*} = {*ln A*<..*ln x*} **for** *x*
using *A*
by(*intro image-mono-on-continuous-on-Ioc continuous-on-ln strict-mono-onI ln-strict-mono*)
auto
have *im-ln2*:*ln* ‘ {*A*..} = {*ln A*..}
using *A*
by(*intro image-mono-on-continuous-on-Ic continuous-on-ln mono-onI ln-mono ln-at-top*)
auto
have ∗:∀ *x* ≥ *A*. *set-integrable lborel* {*A*<..*x*} (λ*t*. *r t* / *t*)
∧ *U x* = *c x* ∗ *exp* (∫ *t*∈{*A*<..*x*}. *r t* / *t ∂lborel*)
proof *safe*
fix *x*
assume *x*:*A* ≤ *x*
have [*simp*]:*set-integrable lborel* ({*ln A*<..*ln x*}) *e*
unfolding *im-ln*[*OF x*]
by(*auto intro*!: *e-integrable ln-mono x A*)
have *1*:*set-integrable lborel* {*A*<..*x*} (λ*x*. |*1* / *x*| ∗ *e* (*ln x*))
using *A x*
by(*intro lebesgue-integrable-change-of-variables-real*
[**where** *g′*=λ*x*. *1* / *x* **and** *f*=*e*,*THEN iffD1*]
strict-mono-on-imp-inj-on)
(*auto simp*: *set-borel-measurable-def im-ln*[*OF x*] *DERIV-ln-divide*
has-field-derivative-at-within
intro!: *measurable-restrict-space1 strict-mono-onI ln-strict-mono*)
have *integ1*:*set-integrable lborel* {*A*<..*x*} (λ*t*. *e* (*ln t*) / *t*)
by(*rule set-integrable-cong*[*THEN iffD1*,*OF - - - 1*]) (*use A* **in** *auto*)
have *integ2*:*set-integrable lborel* {*A*<..*x*} (λ*t*. *ϱ* / *t*)
unfolding *set-integrable-def*
using *x A* **by**(*auto intro*!: *integrableI-bounded-set*[**where** *A*={*A*<..*x*} **and** *B*=|*ϱ*| / *A*]
simp: *divide-mono*)
show *set-integrable lborel* {*A*<..*x*} (λ*t*. *r t* / *t*)
by(*auto simp*: *r-def add-divide-distrib integ1 integ2*)
have (∫ *u*∈{*ln A*<..*ln x*}. *ϱ* + *e u ∂lborel*) = (∫ *u*∈*ln* ‘ {*A*<..*x*}. *ϱ* + *e u ∂lborel*)
by(*simp add*: *im-ln x*)
also have ... = (∫ *x*∈{*A*<..*x*}. |*1* / *x*| ∗ (*ϱ* + *e* (*ln x*)) *∂lborel*)

using *A x*
by(*intro lebesgue-integral-change-of-variables-real1*
[**where** *g'*=λ*x*. *1 / x* **and** *f*=λ*x*. *ϱ + e x*] *strict-mono-on-imp-inj-on*)
(*auto simp*: *set-borel-measurable-def im-ln*[*OF x*] *DERIV-ln-divide*
has-field-derivative-at-within
intro!: *set-integrable-const measurable-restrict-space1 strict-mono-onI ln-strict-mono*)
finally have *eq*:(∫ *u*∈{*ln A*<..*ln x*}. *ϱ + e u* ∂*lborel*) = (∫ *t*∈{*A*<..*x*}. *r t / t*
∂*lborel*)
using *A* **by**(*auto simp*: *r-def intro*!: *set-lebesgue-integral-cong*)
have *U x* = *exp* (*h* (*ln x*))
using *A x regularly-varying-pos*[*OF rv*] **by**(*simp add*: *h-def*)
also have ... = *exp* (*c'* (*ln x*) + *ϱ* ∗ *ln A* + (∫ *u*∈{*ln A*<..*ln x*}. *ϱ + e u* ∂*lborel*))
using *A x* **by**(*simp add*: *h-eq*)
also have ... = *c x* ∗ *exp* (∫ *u*∈{*ln A*<..*ln x*}. *ϱ + e u* ∂*lborel*)
using *A* **by**(*simp add*: *c-def exp-add ln-powr*[*symmetric*] *del*: *ln-powr*)
also have ... = *c x* ∗ *exp* (∫ *t*∈{*A*<..*x*}. *r t / t* ∂*lborel*)
by(*simp add*: *eq*)
finally show *U x* = *c x* ∗ *exp* (∫ *t*∈{*A*<..*x*}. *r t / t* ∂*lborel*) **.**
qed
have ∗∗:*r* ∈ *borel-measurable borel*
by(*simp add*: *r-def*)
have ∗∗∗:(*r* ⟶ *ϱ*) *at-top*
unfolding *r-def*
by(*rule filterlim-compose*[*OF - ln-at-top*])
(*use tendsto-add*[*OF tendsto-const e-lim*] **in** *simp*)
have ∗∗∗∗:*bounded* (*r* ‘ {*A*..})
unfolding *r-def image-image*[*symmetric*,**where** *g*=*ln* **and** *f*=λ*x*. *ϱ + e x*]
im-ln2
by(*auto intro*!: *bounded-plus-comp*[*OF boundedI e-bdd*])
have ∗∗∗∗∗:*c* ∈ *borel-measurable borel*
by (*simp add*: *c-def*)
have ∗∗∗∗∗∗:∀ *x*. *0* < *c x*
using *A* **by**(*simp add*: *c-def*)
have *cC*:(*c* ⟶ *C*) *at-top*
unfolding *c-def C-def*
by(*rule filterlim-compose*[*OF - ln-at-top*])
(*auto intro*!: *tendsto-mult-right tendsto-exp c-lim*)
have *C*:*0* < *C*
using *A* **by**(*simp add*: *C-def*)
show *?thesis*
using ∗ ∗∗ ∗∗∗ ∗∗∗∗ ∗∗∗∗∗ ∗∗∗∗∗∗ *C cC* **by** *blast*
qed

lemma *Karamata-representation-ex*:
assumes *rv*:*regularly-varying U ϱ*
shows ∃ *A*>*0*. ∃ *c r C*.
(∀ *x* ≥ *A*. *U x* = *c x* ∗ *exp* (∫ *t*∈{*A*<..*x*}. *r t / t* ∂*lborel*))
∧ (∀ *x* ≥ *A*. *set-integrable lborel* {*A*<..*x*} (λ*t*. *r t / t*))
∧ *r* ∈ *borel-measurable borel*

$\wedge$ $(r \longrightarrow \varrho)$ *at-top*
$\wedge$ *bounded* $(r \mathbin{`} \{A..\})$
$\wedge$ $c \in$ *borel-measurable borel*
$\wedge$ $(\forall x.\ 0 < c\ x)$
$\wedge$ $(c \longrightarrow C)$ *at-top*
$\wedge$ $0 < C$

proof −
obtain A **where** $0 < A$ $\bigwedge a\ b.\ A \leq a \Longrightarrow \exists X{>}0.\ \exists Y.\ U \mathbin{`} \{a..b\} \subseteq \{X..Y\}$
using *regularly-varying-eventually-bounded-interval-strong*[*OF rv*]
unfolding *eventually-at-top-linorder*
by (*meson exp-gt-self exp-gt-zero less-eq-real-def*)
from *Karamata-representation-ex1*[*OF rv this*] *this*
show *?thesis*
by *blast*
qed

corollary *Karamata-representation-ex-total*:
assumes *rv*:*regularly-varying U* ϱ
shows $\exists c\ r\ C.$
$(\forall x.\ U\ x = c\ x * exp\ (\int t{\in}\{0{<}..x\}.\ r\ t\ /\ t\ \partial lborel))$
$\wedge$ $(\forall x\ y.$ *set-integrable lborel* $\{..y\}$ $(\lambda t.\ r\ t\ /\ t))$
$\wedge$ $r \in$ *borel-measurable borel*
$\wedge$ $(r \longrightarrow \varrho)$ *at-top*
$\wedge$ *bounded* (*range r*)
$\wedge$ $c \in$ *borel-measurable* (*restrict-space borel* $\{0{<}..\}$)
$\wedge$ $(\forall x{>}0.\ 0 < c\ x)$
$\wedge$ $(c \longrightarrow C)$ *at-top*
$\wedge$ $0 < C$

proof −
from *Karamata-representation-ex*[*OF rv*]
obtain $A\ c'\ r'\ C$ **where** *A-pos*:$0 < A$
and *eq'*: $\bigwedge x.\ A \leq x \Longrightarrow U\ x = c'\ x * exp\ (\int t{\in}\{A{<}..x\}.\ r'\ t\ /\ t\ \partial lborel)$
and *integ'*:$\bigwedge x.\ A \leq x \Longrightarrow$ *set-integrable lborel* $\{A{<}..x\}$ $(\lambda t.\ r'\ t\ /\ t)$
and *r'*[*measurable*]:$r' \in$ *borel-measurable borel*
and *lim-r'*:$(r' \longrightarrow \varrho)$ *at-top*
and *bdd*:*bounded* $(r' \mathbin{`} \{A..\})$
and *c'*[*measurable*]:$c' \in$ *borel-measurable borel*
and *c'-pos*:$\bigwedge x.\ 0 < c'\ x$
and *lim-c'*:$(c' \longrightarrow C)$ *at-top*
and *C-pos*:$0 < C$
by *blast*
define *c* **where** $c \equiv \lambda x.$ *if* $A < x$ *then* $c'\ x$ *else* $U\ x$
define *r* **where** $r \equiv \lambda x.$ *if* $A < x$ *then* $r'\ x$ *else* 0
have *r-integral*:$(\int t{\in}\{0{<}..x\}.\ r\ t\ /\ t\ \partial lborel) = 0$ **if** *Ax*:$x \leq A$ **for** x
proof −
have $(\int t{\in}\{0{<}..x\}.\ r\ t\ /\ t\ \partial lborel) = (\int t{\in}\{0{<}..x\}.\ 0\ \partial lborel)$
unfolding *set-lebesgue-integral-def* **using** *Ax*
by(*intro Bochner-Integration.integral-cong*)
(*auto simp*: *r-def indicator-def*)

```
    thus ?thesis
      by simp
  qed
  have r-integrable1:x ≤ A ⟹ set-integrable lborel {..x} (λt. r t / t) for x
   by(subst set-integrable-cong[where f'=λx. 0,OF refl refl])
     (auto simp: r-def)
  hence r-integrable1'[simp]:x ≤ A ⟹ set-integrable lborel {0<..x} (λt. r t / t)
for x
    using set-integrable-subset[where A={..x} and B={0<..x}]
    by fastforce
  have r-integrabl2[simp]: A ≤ x ⟹ set-integrable lborel {A<..x} (λt. r t / t) for
x
    by(subst set-integrable-cong[where f'=λt. r' t / t,OF refl refl])
      (auto simp: r-def integ')
  have 1:U x = c x * exp (∫ t∈{0<..x}. r t / t ∂lborel) for x
  proof(cases A ≤ x)
    assume Ax:A ≤ x
    have (∫ t∈{0<..x}. r t / t ∂lborel) = (∫ t∈{0<..A}∪{A<..x}. r t / t ∂lborel)
      using Ax A-pos by(intro arg-cong[where f=λB. (∫ t∈B. r t / t ∂lborel)])
auto
    also have ... = (∫ t∈{0<..A}. r t / t ∂lborel) + (∫ t∈{A<..x}. r t / t ∂lborel)
      by(subst set-integral-Un) (auto simp: Ax)
    also have ... = (∫ t∈{A<..x}. r t / t ∂lborel)
      by(simp add: r-integral)
    also have ... = (∫ t∈{A<..x}. r' t / t ∂lborel)
     by(auto intro!: Bochner-Integration.integral-cong simp: set-lebesgue-integral-def
r-def)
    finally show ?thesis
      using Ax by(simp add: c-def eq')
  qed(auto simp: c-def r-integral)
  have 2:set-integrable lborel {..x} (λt. r t / t) for x
  proof −
    consider A ≤ x | x ≤ A
      by fastforce
    then show ?thesis
    proof cases
      assume Ax:A ≤ x
      then have [simp]:{..A} ∪ {A<..x} = {..x}
        by auto
      with set-integrable-Un[OF r-integrable1[OF order.refl] r-integrabl2[OF Ax]]
      show ?thesis
        by simp
    qed fact
  qed
  have 3:r ∈ borel-measurable borel
    by(simp add: r-def)
  have 4:(r ⟶ ϱ) at-top
    by(rule tendsto-cong[THEN iffD1,OF eventually-at-top-linorderI[where c=A
+ 1]] lim-r')
```

```
    (simp add: r-def)
  have 5:bounded (range r)
  proof −
    have eq:r ‘ {A..} ∪ r ‘ {..<A} = range r
      by fastforce
    have ∗:bounded (r ‘ {A..})
      using bdd bounded-iff by(auto simp: r-def bounded-subset image-mono)
    have ∗∗:bounded (r ‘ {..<A})
      by(auto simp: bounded-iff r-def)
    have bounded (r ‘ {A..} ∪ r ‘ {..<A})
      by(simp add: ∗ ∗∗)
    thus ?thesis
      by(simp add: eq)
  qed
  have 6:c ∈ borel-measurable (restrict-space borel {0<..})
    unfolding c-def
  proof(subst measurable-If)
    show c′ ∈ borel-measurable (restrict-space borel {0<..})
      by(auto intro!: measurable-restrict-space1)
    show {x ∈ space (restrict-space borel {0<..}). A < x} ∈ sets (restrict-space
borel {0<..})
      using borel-measurable-const measurable-id measurable-inequality-set-flipped
        measurable-restrict-space1 by blast
  qed(use regularly-varying-meas[OF rv] in auto)
  have 7:∀ x>0. 0 < c x
    using c′-pos regularly-varying-pos[OF rv] by(simp add: c-def)
  have 8:(c ⟶ C) at-top
    by(rule tendsto-cong[THEN iffD1,OF eventually-at-top-linorderI[where c=A
+ 1] lim-c′])
      (simp add: c-def)
  show ?thesis
    using 1 2 3 4 5 6 7 8 C-pos by blast
qed

lemma Karamata-representation-iff:
  assumes U-meas:U ∈ borel-measurable (restrict-space borel {0<..})
    and U-pos:⋀x. 0 < x ⟹ 0 < U x
  shows regularly-varying U ϱ ⟷
    (∃ A>0. ∃ c r C.
            (∀ x ≥ A. U x = c x ∗ exp (∫ t∈{A<..x}. r t / t ∂lborel))
          ∧ (∀ x ≥ A. set-integrable lborel {A<..x} (λt. r t / t))
          ∧ (r ⟶ ϱ) at-top
          ∧ (c ⟶ C) at-top
          ∧ 0 < C)
proof(safe intro!: regularly-varyingI)
  fix A c r C x
  assume A:0 < A and x-pos:(0 :: real) < x
    and eq-U:∀ x≥A. U x =
            c x ∗ exp (∫ t∈{A<..x}. r t / t ∂lborel)
```

```
    and integ:∀ x≥A. set-integrable lborel {A<..x}
              (λt. r t / t)
    and lim-r:(r ⟶ ϱ) at-top
    and C-pos:0 < C
    and lim-c:(c ⟶ C) at-top
  have lim-powr1:((λs. x powr s) ⟶ 1) (at-right 0)
   using tendsto-powr[OF tendsto-const[of x] tendsto-ident-at[of 0]] x-pos by simp
  have lim-powr2:((λs. x powr (− s)) ⟶ 1) (at-right 0)
   using tendsto-powr[OF tendsto-const[of x] tendsto-minus[OF tendsto-ident-at[of
0]]] x-pos
    by simp
  have ∗:∀ F t in at-top. U (t ∗ x) / U t
        = (c (t ∗ x) / c t)
         ∗ exp ((∫ s∈{A<..t ∗ x}. r s / s ∂lborel) − (∫ s∈{A<..t}. r s / s ∂lborel))
    using x-pos A eq-U
    by(auto intro!: eventually-at-top-linorderI[where c=max (A / x) A]
       simp: divide-le-eq exp-diff)
  have ∗∗:((λt. c (t ∗ x) / c t) ⟶ C / C) at-top
    using C-pos
    by(intro tendsto-divide filterlim-ident lim-c tendsto-const
         filterlim-compose[OF lim-c filterlim-at-top-mult-tendsto-pos[OF - x-pos]])
simp
  have ∗∗∗:((λt. exp ((∫ s∈{A<..t ∗ x}. r s / s ∂lborel) − (∫ s∈{A<..t}. r s / s
∂lborel)))
          ⟶ x powr ϱ) at-top
  proof −
    consider x < 1 | x = 1 | 1 < x
      by linarith
    then show ?thesis
    proof cases
      assume x:x < 1
      show ?thesis
      proof(rule tendsto-sandwich-mult-const[OF - lim-powr1 lim-powr2])
        fix e :: real
        assume e:0 < e
        obtain T1 where T1:⋀t. T1 ≤ t ⟹ r t ≤ ϱ + e ⋀t. T1 ≤ t ⟹ ϱ − e
≤ r t
         using tendstoD[OF lim-r e] unfolding eventually-at-top-linorder dist-real-def
          by (metis abs-diff-le-iff less-eq-real-def)
        define T where T ≡ max (A / x) (T1 / x)
        show ∀ F t in at-top.
            x powr e ≤ exp ((∫ s∈{A<..t ∗ x}. r s / s ∂lborel)
                          − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
      ∧ exp ((∫ s∈{A<..t ∗ x}. r s / s ∂lborel)
            − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
        ≤ x powr − e
        proof(safe intro!: eventually-at-top-linorderI[where c=T])
          fix t
          assume t:T ≤ t
```

then have *t-pos*:$0 < t$ **and** *tx*:$t * x \leq t$
using *A x-pos x*
by(*auto simp*: *T-def order.strict-trans2*[*OF divide-pos-pos*[*OF A x-pos*]])
hence *Atx*:$A \leq t * x$ **and** *At*:$A \leq t$
using *A t x-pos x*
by(*auto simp*: *T-def divide-le-eq intro*!: *order.trans*[*of A t * x t*] *mult-left-le*)
have *t-le*:$\bigwedge s.\ t * x \leq s \Longrightarrow r\ s \leq \varrho + e$ **and** *le-t*:$\bigwedge s.\ t * x \leq s \Longrightarrow \varrho - e \leq r\ s$
using *t x-pos x* **by**(*auto simp*: *T-def divide-le-eq intro*!: *T1*)
have *exp* $((\int s{\in}\{A{<}..t * x\}.\ r\ s\ /\ s\ \partial lborel) - (\int s{\in}\{A{<}..t\}.\ r\ s\ /\ s\ \partial lborel))$
$=$ *exp* $((\int s{\in}\{t * x{<}..t\}.\ -\ r\ s\ /\ s\ \partial lborel))$
unfolding *set-lebesgue-integral-def*
using *integ At Atx tx*
by(*subst Bochner-Integration.integral-diff*[*symmetric*])
(*auto simp*: *set-lebesgue-integral-def set-integrable-def indicator-def*
simp del: *integral-minus Bochner-Integration.integral-diff*
intro!: *Bochner-Integration.integral-cong*)
also have ... $\leq$ *exp* $((\int s{\in}\{t * x{<}..t\}.\ -\ (\varrho - e)\ /\ s\ \partial lborel))$
using *integ At Atx A*
by(*auto simp*: *set-integrable-uminus simp del*: *minus-diff-eq*
intro!: *le-t set-integral-mono divide-right-mono set-integrable-inverse'*
set-integrable-subset[**where** $A{=}\{A{<}..t\}$ **and** $B{=}\{t * x{<}..t\}$])
also have ... $=$ *exp* $((\varrho - e) * ln\ x)$
using *A Atx x-pos t-pos x*
by(*subst set-integral-inverse*) (*auto simp*: *left-diff-distrib' ln-mult*)
also have ... $=$ *x powr* $(\varrho - e)$
by (*metis nless-le powr-def x-pos*)
finally have *exp* $((\int s{\in}\{A{<}..t * x\}.\ r\ s\ /\ s\ \partial lborel)$
$-\ (\int s{\in}\{A{<}..t\}.\ r\ s\ /\ s\ \partial lborel)) \leq x\ powr\ (\varrho - e)$ **.**
from *divide-right-mono*[*OF this*,*of x powr* ϱ]
show *exp* $((\int s{\in}\{A{<}..t * x\}.\ r\ s\ /\ s\ \partial lborel)$
$-\ (\int s{\in}\{A{<}..t\}.\ r\ s\ /\ s\ \partial lborel))\ /\ x\ powr\ \varrho \leq x\ powr - e$
using *x-pos* **by**(*auto simp*: *powr-diff*[*symmetric*])
have $x\ powr\ (\varrho + e) =$ *exp* $((\varrho + e) * ln\ x)$
using *x-pos* **by**(*simp add*: *powr-def*)
also have ... $=$ *exp* $((\int s{\in}\{t{*}\ x{<}..t\}.\ -\ (\varrho + e)\ /\ s\ \partial lborel))$
using *t-pos A Atx tx*
by(*subst set-integral-inverse*) (*auto simp*: *ln-mult left-diff-distrib' distrib-right*)
also have ... $\leq$ *exp* $((\int s{\in}\{t * x{<}..t\}.\ -\ r\ s\ /\ s\ \partial lborel))$
using *integ At Atx A*
by(*auto simp*: *set-integrable-uminus simp del*: *minus-add-distrib*
intro!: *t-le set-integral-mono divide-right-mono set-integrable-inverse'*
set-integrable-subset[**where** $A{=}\{A{<}..t\}$ **and** $B{=}\{t * x{<}..t\}$])
also have ... $=$ *exp* $((\int s{\in}\{A{<}..t * x\}.\ r\ s\ /\ s\ \partial lborel)$
$-\ (\int s{\in}\{A{<}..t\}.\ r\ s\ /\ s\ \partial lborel))$
unfolding *set-lebesgue-integral-def*
using *integ At Atx tx*

```
        by(subst Bochner-Integration.integral-diff[symmetric])
          (auto simp: set-lebesgue-integral-def set-integrable-def indicator-def
            simp del: integral-minus Bochner-Integration.integral-diff
            intro!: Bochner-Integration.integral-cong)
      finally have x powr (ϱ + e) ≤ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                                        − (∫ s∈{A<..t}. r s / s ∂lborel)) .
      from divide-right-mono[OF this,of x powr ϱ]
      show x powr e ≤ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                         − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
        using x-pos by(auto simp: powr-diff[symmetric])
    qed
  qed
 next
  assume x:1 < x
  show ?thesis
  proof(rule tendsto-sandwich-mult-const[OF - lim-powr2 lim-powr1])
    fix e :: real
    assume e:0 < e
    obtain T1 where T1:⋀t. T1 ≤ t ⟹ r t ≤ ϱ + e ⋀t. T1 ≤ t ⟹ ϱ − e
≤ r t
     using tendstoD[OF lim-r e] unfolding eventually-at-top-linorder dist-real-def
      by (metis abs-diff-le-iff less-eq-real-def)
    define T where T ≡ max A T1
    show ∀ F t in at-top.
        x powr (− e) ≤ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                          − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
  ∧ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
        − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
    ≤ x powr e
    proof(safe intro!: eventually-at-top-linorderI[where c=T])
      fix t
      assume t:T ≤ t
      then have t-pos:0 < t and tx:t ≤ t * x
        using A x-pos x
        by(auto simp: T-def order.strict-trans2[OF divide-pos-pos[OF A x-pos]])
      hence Atx:A ≤ t * x and At:A ≤ t
        using t tx by(auto simp: T-def order.trans)
      have t-le:⋀s. t ≤ s ⟹ r s ≤ ϱ + e and le-t:⋀s. t ≤ s ⟹ ϱ − e ≤ r s
        using t by(auto simp: T-def intro!:T1)
        have exp ((∫ s∈{A<..t * x}. r s / s ∂lborel) − (∫ s∈{A<..t}. r s / s
∂lborel))
          = exp ((∫ s∈{t<..t * x}. r s / s ∂lborel))
        unfolding set-lebesgue-integral-def
        using integ At Atx tx
        by(subst Bochner-Integration.integral-diff[symmetric])
          (auto simp: set-lebesgue-integral-def set-integrable-def indicator-def
            simp del: integral-minus Bochner-Integration.integral-diff
            intro!: Bochner-Integration.integral-cong)
      also have ... ≤ exp ((∫ s∈{t<..t * x}. (ϱ + e) / s ∂lborel))
```

```
        using integ At Atx A tx
        by(fastforce simp:
            intro!: t-le set-integral-mono divide-right-mono set-integrable-inverse'
            set-integrable-subset[where A={A<..t*x} and B={t<..t * x}])
      also have ... = exp ((ϱ + e) * ln x)
        using A Atx x-pos t-pos x
        by(subst set-integral-inverse) (auto simp: left-diff-distrib' ln-mult)
      also have ... = x powr (ϱ + e)
        by (metis nless-le powr-def x-pos)
      finally have exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                        − (∫ s∈{A<..t}. r s / s ∂lborel)) ≤ x powr (ϱ + e) .
      from divide-right-mono[OF this,of x powr ϱ]
      show exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ ≤ x powr e
        using x-pos by(auto simp: powr-diff[symmetric])
      have x powr (ϱ − e) = exp ((ϱ − e) * ln x)
        using x-pos by(simp add: powr-def)
      also have ... = exp ((∫ s∈{t<..t*x}. (ϱ − e) / s ∂lborel))
        using t-pos A Atx tx
          by(subst set-integral-inverse) (auto simp: ln-mult left-diff-distrib' dis-
trib-right)
      also have ... ≤ exp ((∫ s∈{t<..t * x}. r s / s ∂lborel))
        using integ At Atx A tx
        by(fastforce simp:
            intro!: le-t set-integral-mono divide-right-mono set-integrable-inverse'
            set-integrable-subset[where A={A<..t*x} and B={t<..t * x}])
      also have ... = exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                           − (∫ s∈{A<..t}. r s / s ∂lborel))
        unfolding set-lebesgue-integral-def
        using integ At Atx tx
        by(subst Bochner-Integration.integral-diff[symmetric])
          (auto simp: set-lebesgue-integral-def set-integrable-def indicator-def
            simp del: integral-minus Bochner-Integration.integral-diff
            intro!: Bochner-Integration.integral-cong)
      finally have x powr (ϱ − e) ≤ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                                         − (∫ s∈{A<..t}. r s / s ∂lborel)) .
      from divide-right-mono[OF this,of x powr ϱ]
      show x powr (− e) ≤ exp ((∫ s∈{A<..t * x}. r s / s ∂lborel)
                               − (∫ s∈{A<..t}. r s / s ∂lborel)) / x powr ϱ
        using x-pos by(auto simp: powr-diff[symmetric])
    qed
  qed
 qed simp
qed
have ((λt. (c (t * x) / c t)
      * exp ((∫ s∈{A<..t * x}. r s / s ∂lborel) − (∫ s∈{A<..t}. r s / s ∂lborel)))
     ⟶ x powr ϱ) at-top
  using tendsto-mult[OF ** ***] C-pos by simp
thus ((λt. U (t * x) / U t) ⟶ x powr ϱ) at-top
```

using *tendsto-cong* ∗ **by** *fast*
qed(*use assms Karamata-representation-ex* **in** *blast*)+

2.4.1 Theorems Obtained from Karamata's Representation Theorem

lemma
assumes *rv*:*regularly-varying U ϱ*
shows *regularly-varying-pos-at-top*:*0 < ϱ* ⟹ *filterlim U at-top at-top*
and *regularly-varying-neg-at-top*:*ϱ < 0* ⟹ (*U* ⟶ *0*) *at-top*
proof −
let *?h* = λ*x*. *ln* (*U* (*exp x*))
from *Karamata-representation-ln-exp*[*OF rv*]
obtain *A c e C* **where** *A*:*0 < A*
and *eq-h*: ⋀*x*. *ln A* ≤ *x* ⟹ *?h x* = *c x* + *ϱ* ∗ *ln A* + (∫ *u*∈{*ln A*<..*x*}. *ϱ* + *e u* ∂*lborel*)
and *integrable*: ⋀*x*. *ln A* ≤ *x* ⟹ *set-integrable lborel* {*ln A*<..*x*} *e*
and *lim-e*: (*e* ⟶ *0*) *at-top*
and *lim-c*: (*c* ⟶ *C*) *at-top*
by *blast*
obtain *X1* **where** *X1*:⋀*x*. *X1* ≤ *x* ⟹ *C* − *1* < *c x* ⋀*x*. *X1* ≤ *x* ⟹ *c x* < *C* + *1*
using *tendstoD*[*OF lim-c*,*of 1*]
unfolding *eventually-at-top-linorder dist-real-def abs-less-iff*
by *fastforce*
have *lim-h-pos*: *0 < ϱ* ⟹ *filterlim ?h at-top at-top*
proof −
assume *rho*:*0 < ϱ*
obtain *X2* **where** *X2*:⋀*x*. *X2* ≤ *x* ⟹ |*e x*| < *ϱ* / *2*
using *tendstoD*[*OF lim-e half-gt-zero*[*OF rho*]]
by (*metis* (*no-types*, *lifting*) *eventually-at-top-linorder norm-conv-dist real-norm-def*)
define *X* **where** *X* ≡ *max* (*ln A*) (*max X1 X2*)
have *le-X*:*ln A* ≤ *X X1* ≤ *X X2* ≤ *X*
by(*simp-all add*: *X-def*)
have *X*:⋀*x*. *X* ≤ *x* ⟹ − *ϱ* / *2* ≤ *e x* ⋀*x*. *X* ≤ *x* ⟹ *C* − *1* ≤ *c x*
using *X2 X1 X-def* **by** *force*+
have ∀$_F$ *x in at-top*. *?h x* ≥ *C* − *1* + *ϱ* ∗ *ln A* + (∫ *u*∈{*ln A*<..*X*}. *ϱ* + *e u* ∂*lborel*)
+ (− *ϱ* / *2*) ∗ *X* + (*ϱ* / *2*) ∗ *x* (**is** ∀$_F$ *x in at-top*. *?h x* ≥ *?f x*)
proof(*rule eventually-at-top-linorderI*[**where** *c*=*X*])
fix *x*
assume *x*:*X* ≤ *x*
have *?f x* = *C* − *1* + *ϱ* ∗ *ln A* + (∫ *u*∈{*ln A*<..*X*}. *ϱ* + *e u* ∂*lborel*) + (*ϱ* / *2*) ∗ (*x* − *X*)
by *argo*
also have ... = *C* − *1* + *ϱ* ∗ *ln A*
+ ((∫ *u*∈{*ln A*<..*X*}. *ϱ* + *e u* ∂*lborel*) + (∫ *u*∈{*X*<..*x*}. *ϱ* / *2* ∂*lborel*))

```
    by(simp add: interval-integral-Ioc[OF x,symmetric])
  also have ... ≤ C − 1 + ϱ ∗ ln A
                + ((∫ u∈{ln A<..X}. ϱ + e u ∂lborel) + (∫ u∈{X<..x}. ϱ + e
u ∂lborel))
    using X x le-X by(auto simp del: set-integral-divide-zero
      simp: real-0-le-add-iff minus-divide-le-eq
      intro!: set-integral-mono-banach set-integrable-subset[OF integrable[of x]]
      set-integrable-const)
  also have ... = C − 1 + ϱ ∗ ln A + ((∫ u∈{ln A<..X} ∪ {X<..x}. ϱ + e u
∂lborel))
    using le-X x by(auto intro!: set-integral-Un[symmetric]
      set-integrable-subset[OF integrable[of x]] set-integrable-const)
  also have ... = C − 1 + ϱ ∗ ln A + ((∫ u∈{ln A<..x}. ϱ + e u ∂lborel))
    using x le-X by(subst ivl-disj-un-two) auto
  also have ... ≤ ?h x
    using le-X x X by(simp add: eq-h[of x])
  finally show ?f x ≤ ?h x .
 qed
 moreover have filterlim ?f at-top at-top
  by(intro filterlim-tendsto-add-at-top
     filterlim-tendsto-pos-mult-at-top[OF - half-gt-zero[OF rho]] filterlim-ident)
    blast+
 ultimately show ?thesis
  using filterlim-at-top-mono by fast
qed
show filterlim U at-top at-top if rho:0 < ϱ
proof −
 note filterlim-compose[OF exp-at-top filterlim-compose[OF lim-h-pos[OF rho]
ln-at-top]]
 moreover have ∀ F x in at-top. exp (?h (ln x)) = U x
 using regularly-varying-pos[OF rv] by(auto intro!: eventually-at-top-linorderI[where
c=1])
 ultimately show ?thesis
  by(simp cong: filterlim-cong)
qed
have lim-h-neg: ϱ < 0 ⟹ filterlim ?h at-bot at-top
proof −
 assume rho:ϱ < 0
 obtain X2 where X2:⋀x. X2 ≤ x ⟹ |e x| < − ϱ / 2
  using tendstoD[OF lim-e half-gt-zero[OF neg-0-less-iff-less[THEN iffD2,OF
rho]]]
 by (metis (no-types, lifting) eventually-at-top-linorder norm-conv-dist real-norm-def)
 define X where X ≡ max (ln A) (max X1 X2)
 have le-X:ln A ≤ X X1 ≤ X X2 ≤ X
  by(simp-all add: X-def)
 have X:⋀x. X ≤ x ⟹ e x ≤ − ϱ / 2 ⋀x. X ≤ x ⟹ c x ≤ C + 1
  using X2 X1 X-def by force+
 have ∀ F x in at-top. ?h x ≤ C + 1 + ϱ ∗ ln A + (∫ u∈{ln A<..X}. ϱ + e u
∂lborel)
```

```
                                + (− ϱ / 2) * X + (ϱ / 2) * x
        (is ∀_F x in at-top. ?h x ≤ ?g x)
    proof(rule eventually-at-top-linorderI[where c=X])
      fix x
      assume x:X ≤ x
      have ?h x ≤ C + 1 + ϱ * ln A + ((∫ u∈{ln A<..x}. ϱ + e u ∂lborel))
        using le-X x X by(simp add: eq-h[of x])
      also have ... = C + 1 + ϱ * ln A + ((∫ u∈{ln A<..X} ∪ {X<..x}. ϱ + e u
∂lborel))
        using x le-X by(subst ivl-disj-un-two) auto
      also have ... = C + 1 + ϱ * ln A
                      + ((∫ u∈{ln A<..X}. ϱ + e u ∂lborel) + (∫ u∈{X<..x}. ϱ + e
u ∂lborel))
        using le-X x by(auto intro!: set-integral-Un
            set-integrable-subset[OF integrable[of x]] set-integrable-const)
      also have ... ≤ C + 1 + ϱ * ln A
                      + ((∫ u∈{ln A<..X}. ϱ + e u ∂lborel) + (∫ u∈{X<..x}. ϱ / 2
∂lborel))
        using X x le-X by(auto simp del: set-integral-divide-zero
            simp: real-add-le-0-iff le-minus-divide-eq
            intro!: set-integral-mono-banach set-integrable-subset[OF integrable[of x]]
            set-integrable-const)
      also have ... = ?g x
        by(simp add: interval-integral-Ioc[OF x,symmetric] right-diff-distrib)
      finally show ?h x ≤ ?g x .
    qed
    moreover have filterlim ?g at-bot at-top
      using rho
      by (intro filterlim-tendsto-add-at-bot-iff[THEN iffD2]
          filterlim-tendsto-neg-mult-at-bot[where c=ϱ / 2] filterlim-ident)
          fastforce+
    ultimately show ?thesis
      using filterlim-at-bot-mono by fast
  qed
  show (U ⟶ 0) at-top if rho:ϱ < 0
  proof −
     note filterlim-compose[OF exp-at-bot filterlim-compose[OF lim-h-neg[OF rho]
ln-at-top]]
    moreover have ∀_F x in at-top. exp (?h (ln x)) = U x
    using regularly-varying-pos[OF rv] by(auto intro!: eventually-at-top-linorderI[where
c=1])
    ultimately show ?thesis
      by(simp cong: filterlim-cong)
  qed
qed
```

2.4.2 Uniform Convergence Theorem for $\rho \neq 0$

corollary *regularly-varying-pos-UCT*:

```
  assumes rv:regularly-varying U ϱ
    and bdd:⋀x. 0 < x ⟹ bounded (U ` {0<..x})
    and rho:0 < ϱ
    and a: 0 < a
  shows uniform-limit {0<..a} (λt x. U (t * x) / U t) (λx. x powr ϱ) at-top
  unfolding uniform-limit-iff dist-real-def
proof safe
  fix e :: real
  assume e:0 < e
  define e' where e' ≡ min e (1 / 2)
  have e'-pos:0 < e' and e'-le-1:e' < 1
    using e by(simp-all add: e'-def)
  obtain i where i-pos:0 < i and i:ϱ − i > 0
    using rho dense by auto
  from Karamata-representation-ex[OF rv]
  obtain A c r C where A-pos:0 < A
    and eq-U:⋀x. A ≤ x ⟹ U x = c x * exp (∫ t∈{A<..x}. r t / t ∂lborel)
    and integ:⋀x. A ≤ x ⟹ set-integrable lborel {A<..x} (λt. r t / t)
    and lim-r:(r ⟶ ϱ) at-top
    and r-meas[measurable]: r ∈ borel-measurable borel
    and lim-c:(c ⟶ C) at-top
    and C-pos:0 < C
    by blast
  define X where X ≡ min (min (1 / 2) ((e' / 4) powr (1 / ϱ))) ((e' / 16) powr
(1 / (ϱ − i)))
  have X-pos:0 < X and X-le-1:X < 1
    and X-e':X powr ϱ < e' / 2
   using e'-pos rho by(auto simp: X-def powr-powr simp del: less-divide-eq-numeral1
        intro!: order.strict-trans1[OF powr-mono2[OF - - min.coboundedI2[THEN
min.coboundedI1]]])
  have X-e2':X powr (ϱ − i) < e' / 8
   using e'-pos e'-le-1 i by(auto simp: X-def powr-powr simp del: less-divide-eq-numeral1
       intro!: order.strict-trans1[OF powr-mono2[OF - - min.coboundedI2]]
       order.strict-trans2[OF powr-less-mono'[where x=e' / 8 and a = i]])
  have X-e:⋀x. 0 < x ⟹ x ≤ X ⟹ x powr ϱ < e / 2
   using X-e' by (smt (verit, del-insts) rho e'-def field-sum-of-halves powr-mono2)
  have X-e2:x powr (ϱ − i) < e / 8 if xX:x ≤ X and x:0 < x for x
    using i
    by(intro order.strict-trans2[OF order.strict-trans1
        [OF powr-mono2[OF - less-imp-le[OF x] xX,of ϱ − i] X-e2']])
      (simp-all add: e'-def)
  obtain T1 where T1:⋀x. T1 ≤ x ⟹ ϱ − i ≤ r x ⋀x. T1 ≤ x ⟹ 0 < r x
    using tendstoD[OF lim-r i-pos] tendstoD[OF lim-r rho] rho i-pos i
    unfolding eventually-at-top-linorder dist-real-def abs-less-iff
    by fastforce
  obtain T2 where T2':⋀x. T2 ≤ x ⟹ |c x − C| ≤ C / 2
    using tendstoD[OF lim-c half-gt-zero[OF C-pos]]
    unfolding eventually-at-top-linorder dist-real-def
    by fastforce
```

```
  have T2:C / 2 ≤ c x c x ≤ 2 * C if x:T2 ≤ x for x
    using T2'[OF x] by argo+

  define L where L ≡ λx. U x / x powr ϱ
  have sv:slowly-varying L
     by(simp add: regularly-varying-iff-slowly-varying-divide-powr[of U,symmetric]
L-def rv)
  have 0 < e / a powr ϱ
    using e a by auto
  from uniform-limitD[OF slowly-varying-UCT[OF sv X-pos,of a] this]
  obtain T3 where T3:⋀t x. T3 ≤ t ⟹ X ≤ x ⟹ x ≤ a
                        ⟹ |L (t * x) / L t − 1| < e / a powr ϱ
    unfolding dist-real-def eventually-at-top-linorder
    by (meson atLeastAtMost-iff)
  define T' where T' ≡ max (max (max (A / X) A) T3) (max T1 T2)
  have T':A ≤ T' 0 < T' A / X≤ T'
    and T'-le1:⋀x. T' ≤ x ⟹ ϱ − i ≤ r x ⋀x. T' ≤ x ⟹ 0 < r x
    and T'-le2:⋀x. T' ≤ x ⟹ C / 2 ≤ c x
    and T'-le2': ⋀x. T' ≤ x ⟹ c x ≤ 2 * C
    and T'-le3:⋀t x. T' ≤ t ⟹ X ≤ x ⟹ x ≤ a ⟹ |L (t * x) / L t − 1| < e
/ a powr ϱ
    using T1 T2 T3 A-pos by(auto simp: T'-def)
  from bdd[of T'] regularly-varying-pos[OF rv] T'
  obtain K where K:⋀x. 0 < x ⟹ x ≤ T' ⟹ |U x| ≤ K
    unfolding bounded-iff by fastforce
  have ((λx. K / U x) ⟶ 0) at-top
   by(intro real-tendsto-divide-at-top regularly-varying-pos-at-top[OF rv rho]) auto
  then obtain T'' where T'':⋀t. T'' ≤ t ⟹ |K / U t| < e / 2
    by (metis (mono-tags, lifting) divide-pos-pos e eventually-at-top-linorder
      less-le norm-conv-dist real-norm-def tendstoD zero-le-numeral zero-neq-numeral)
  define T where T ≡ max T' T''
  have T:T' ≤ T T'' ≤ T
    by(simp-all add: T-def)
  show ∀⇩F t in at-top. ∀x∈{0<..a}. |U (t * x) / U t − x powr ϱ| < e
  proof(safe intro!: eventually-at-top-linorderI[where c=T])
    fix t x
    assume t:T ≤ t and x ∈ {0<..a}
    then have x-pos:0 < x and x-a: x ≤ a
      and T't:T' ≤ t and t-pos:0 < t and T''t:T'' ≤ t
      using T A-pos by (auto simp: T'-def)
    then consider x ≤ X T' ≤ t * x | x ≤ X t * x ≤ T' | X ≤ x
      by fastforce
    thus |U (t * x) / U t − x powr ϱ| < e
    proof cases
      assume x:X ≤ x
      have |U (t * x) / U t − x powr ϱ| = x powr ϱ * |L (t * x) / L t − 1|
     using T T' t x-pos by(simp add: L-def powr-mult abs-mult-pos' right-diff-distrib)
      also have ... ≤ a powr ϱ * |L (t * x) / L t − 1|
        using x-pos x-a rho by(intro mult-mono powr-mono2) auto
```

```
    also have ... < e
      using a mult-le-less-imp-less[OF order.refl[of a powr ϱ] T'-le3[OF T't x x-a]] by force
    finally show ?thesis .
  next
    assume x:x ≤ X and txT:t * x ≤ T'
    have |U (t * x) / U t| ≤ |K / U t|
      using K[OF - txT] x-pos t-pos by (simp add: divide-right-mono)
    also have ... < e / 2
      by(intro T'' T''t)
    finally have *:|U (t * x) / U t| < e / 2 .
    have |U (t * x) / U t - x powr ϱ| ≤ |U (t * x) / U t| + |x powr ϱ|
      by argo
    also have ... < e / 2 + e / 2
      unfolding abs-of-nonneg[OF powr-ge-zero]
      by(intro add-strict-mono * X-e x-pos x)
    finally show ?thesis
      by simp
  next
    assume x:x ≤ X and Ttx:T' ≤ t * x
    have Atx:A ≤ t * x and At:A ≤ t and ttx:t * x ≤ t and tx-pos:0 < t * x
      using x T' t T Ttx X-pos A-pos x-pos X-le-1 by auto
    have im-ln:ln ` {t * x<..t} = {ln (t * x) <..ln t}
      using ttx tx-pos
    by(intro image-mono-on-continuous-on-Ioc ttx monotone-onI continuous-on-ln ln-strict-mono)
       auto
    have U (t * x) / U t = c (t * x) * exp (∫ u∈{A<..t * x}. r u / u ∂lborel)
                           / (c t * exp (∫ u∈{A<..t}. r u / u ∂lborel))
      by(simp add: eq-U Atx At)
    also have ... = (c (t * x) / c t) * exp ((∫ u∈{A<..t * x}. r u / u ∂lborel)
                                            - (∫ u∈{A<..t}. r u / u ∂lborel))
      by (simp add: exp-diff)
    also have ... ≤ (2 * C / (C / 2)) * exp ((∫ u∈{A<..t * x}. r u / u ∂lborel)
                                            - (∫ u∈{A<..t}. r u / u ∂lborel))
      using frac-le[OF - T'-le2'[OF Ttx] - T'-le2[OF T't]] C-pos by(intro mult-mono) auto
    also have ... = 4 * exp ((∫ u. indicat-real {A<..t * x} u * r u / u
                                - indicat-real {A<..t} u * r u / u ∂lborel))
      using integ[OF Atx] integ[OF At] C-pos
      by(auto simp: set-lebesgue-integral-def set-integrable-def)
    also have ... = 4 * exp (∫ u∈{t*x<..t}. - (r u / u) ∂lborel)
      using ttx Atx At
      by(auto intro!: Bochner-Integration.integral-cong
          simp: set-lebesgue-integral-def indicator-def simp del: integral-minus)
    also have ... ≤ 4 * exp (∫ u∈{t*x<..t}. - (ϱ - i) / u ∂lborel)
      unfolding divide-minus-left
      using Atx ttx tx-pos Ttx rho i T'-le1
      by(intro mult-mono exp-mono set-integral-mono
```

```
        set-integrable-subset[THEN set-integrable-uminus[THEN iffD2],OF integ,of
t])
        (fastforce simp: set-integrable-def
          intro!: integrableI-bounded-set[where A={t * x<..t} and B=|ϱ − i| / (t
* x)]
           divide-mono)+
    also have ... = 4 * exp (− (ϱ − i) * (ln t − ln (t * x)))
      by(subst set-integral-inverse) (use ttx tx-pos in auto)
    also have ... = 4 * exp (− (ϱ − i) * ln (1 / x))
      using A-pos At Atx
      by(subst ln-divide-pos[symmetric]) auto
    also have ... = 4 * exp ((ϱ − i) * ln x)
      by(simp add: x-pos ln-divide-pos[of 1 x] minus-mult-right mult.commute)
    also have ... = 4 * x powr (ϱ − i)
      using x-pos i by(simp add: powr-def X-def)
    also have ... < e / 2
      using x X-e2 x-pos by force
    finally have *:|U (t * x) / U t| < e / 2
      by (simp add: regularly-varying-pos[OF rv] abs-of-pos t-pos tx-pos)
    have |U (t * x) / U t − x powr ϱ| ≤ |U (t * x) / U t| + |x powr ϱ|
      by linarith
    also have ... < e / 2 + e / 2
      using * X-e[OF x-pos x] by(intro add-strict-mono) auto
    finally show ?thesis
      by fastforce
  qed
 qed
qed

corollary regularly-varying-neg-UCT:
 assumes rv:regularly-varying U ϱ
   and rho:ϱ < 0
   and a: 0 < a
 shows uniform-limit {a..} (λt x. U (t * x) / U t) (λx. x powr ϱ) at-top
 unfolding uniform-limit-iff dist-real-def
proof safe
 fix e :: real
 assume e:0 < e
 obtain i where i-pos:0 < i and i:ϱ + i < 0
   by (metis rho neg-0-less-iff-less real-add-less-0-iff real-arch-inverse)
 from Karamata-representation-ex[OF rv]
 obtain A c r C where A-pos:0 < A
   and eq-U:⋀x. A ≤ x ⟹ U x = c x * exp (∫ t∈{A<..x}. r t / t ∂lborel)
   and integ:⋀x. A ≤ x ⟹ set-integrable lborel {A<..x} (λt. r t / t)
   and lim-r:(r ⟶ ϱ) at-top
   and lim-c:(c ⟶ C) at-top
   and C-pos:0 < C
   by blast
 from tendstoD[OF tendsto-neg-powr[OF rho filterlim-ident] half-gt-zero[OF e]]
```

obtain *X1* **where** *X1*:$\bigwedge x.\ X1 \leq x \Longrightarrow x\ powr\ \varrho < e\ /\ 2$
unfolding *eventually-at-top-linorder dist-real-def* **by** *auto*
from *tendstoD*[*OF tendsto-mult-right-zero*[*OF tendsto-neg-powr*[*OF i filterlim-ident*],*of 4*]
half-gt-zero[*OF e*]]
obtain *X2* **where** *X2*:$\bigwedge x.\ X2 \leq x \Longrightarrow 4 * x\ powr\ (\varrho + i) < e\ /\ 2$
unfolding *eventually-at-top-linorder dist-real-def* **by** *auto*
define *X* **where** $X \equiv max\ (max\ 1\ a)\ (max\ X1\ X2)$
have *X*:$\bigwedge x.\ X \leq x \Longrightarrow x\ powr\ \varrho < e\ /\ 2\ \bigwedge x.\ X \leq x \Longrightarrow 4 * x\ powr\ (\varrho + i) < e\ /\ 2$
$\bigwedge x.\ X \leq x \Longrightarrow 0 < x\ powr\ \varrho\ \bigwedge x.\ X \leq x \Longrightarrow 0 < 4 * x\ powr\ \varrho$
using *X1 X2* **by**(*auto simp*: *X-def*)
obtain *T1* **where** *T1*:$\bigwedge x.\ T1 \leq x \Longrightarrow r\ x \leq \varrho + i$
using *tendstoD*[*OF lim-r i-pos*] **unfolding** *eventually-at-top-linorder dist-real-def abs-less-iff*
by *fastforce*
obtain *T2* **where** *T2'*:$\bigwedge x.\ T2 \leq x \Longrightarrow |c\ x - C| \leq C\ /\ 2$
using *tendstoD*[*OF lim-c half-gt-zero*[*OF C-pos*]]
unfolding *eventually-at-top-linorder dist-real-def*
by *fastforce*
have *T2*:$C\ /\ 2 \leq c\ x\ c\ x \leq 2 * C$ **if** *x*:$T2 \leq x$ **for** *x*
using *T2'*[*OF x*] **by** *argo+*

define *L* **where** $L \equiv \lambda x.\ U\ x\ /\ x\ powr\ \varrho$
have *sv*:*slowly-varying L*
by(*simp add*: *regularly-varying-iff-slowly-varying-divide-powr*[*of U*,*symmetric*] *L-def rv*)
have $0 < e\ /\ a\ powr\ \varrho$
using *e a* **by** *auto*
from *uniform-limitD*[*OF slowly-varying-UCT*[*OF sv a*,*of X*] *this*]
obtain *T3* **where** *T3*:$\bigwedge t\ x.\ T3 \leq t \Longrightarrow a \leq x \Longrightarrow x \leq X \Longrightarrow |L\ (t * x)\ /\ L\ t - 1| < e\ /\ a\ powr\ \varrho$
unfolding *dist-real-def eventually-at-top-linorder*
by (*meson atLeastAtMost-iff*)
define *T* **where** $T \equiv max\ (max\ A\ T3)\ (max\ T1\ T2)$
have *T*:$A \leq T\ 0 < T$
and *T-le1*:$\bigwedge x.\ T \leq x \Longrightarrow r\ x \leq \varrho + i$
and *T-le2*:$\bigwedge x.\ T \leq x \Longrightarrow C\ /\ 2 \leq c\ x$ **and** *T-le2'*: $\bigwedge x.\ T \leq x \Longrightarrow c\ x \leq 2 * C$
and *T-le3*:$\bigwedge t\ x.\ T \leq t \Longrightarrow a \leq x \Longrightarrow x \leq X \Longrightarrow |L\ (t * x)\ /\ L\ t - 1| < e\ /\ a\ powr\ \varrho$
using *T1 T2 T3 A-pos* **by**(*auto simp*: *T-def*)
show $\forall_F\ t\ in\ at\text{-}top.\ \forall x \in \{a..\}.\ |U\ (t * x)\ /\ U\ t - x\ powr\ \varrho| < e$
proof(*safe intro*!: *eventually-at-top-linorderI*[**where** *c*=*T*])
fix *t x*
assume *t*:$T \leq t$ **and** $a \leq x$
then consider $a \leq x\ x \leq X \mid X \leq x$
by *fastforce*
thus $|U\ (t * x)\ /\ U\ t - x\ powr\ \varrho| < e$

proof *cases*
assume $x{:}a \leq x\ x \leq X$
have $|U\ (t * x)\ /\ U\ t - x\ powr\ \varrho| = x\ powr\ \varrho * |L\ (t * x)\ /\ L\ t - 1|$
using $T\ t\ x\ a$ **by**(*simp add*: *L-def powr-mult abs-mult-pos' right-diff-distrib*)
also have $... \leq a\ powr\ \varrho * |L\ (t * x)\ /\ L\ t - 1|$
using $x\ a\ rho$ **by**(*intro mult-mono powr-mono2'*) *auto*
also have $... < e$
using *a mult-le-less-imp-less*[*OF order.refl*[*of a powr* ϱ] *T-le3*[*OF t x*]] **by** *force*
finally show *?thesis* **.**
next
assume $x{:}X \leq x$
have $Atx{:}A \leq t * x$ **and** $At{:}A \leq t$ **and** $ttx{:}t \leq t * x$
using $x\ t$ *mult-mono*[*of A t 1 x*] *A-pos* **by**(*auto simp*: *X-def T-def*)
hence $Ttx{:}T \leq t * x$
using t **by** *linarith*
have *im-ln*:$ln\ `\ \{t<..t * x\} = \{ln\ t <..ln\ (t * x)\}$
using *At A-pos*
by(*intro image-mono-on-continuous-on-Ioc ttx monotone-onI continuous-on-ln ln-strict-mono*)
auto
have $U\ (t * x)\ /\ U\ t = c\ (t * x) * exp\ (\int u{\in}\{A<..t * x\}.\ r\ u\ /\ u\ \partial lborel)$
$/\ (c\ t * exp\ (\int u{\in}\{A<..t\}.\ r\ u\ /\ u\ \partial lborel))$
by(*simp add*: *eq-U Atx At*)
also have $... = (c\ (t * x)\ /\ c\ t) * exp\ ((\int u{\in}\{A<..t * x\}.\ r\ u\ /\ u\ \partial lborel)$
$- (\int u{\in}\{A<..t\}.\ r\ u\ /\ u\ \partial lborel))$
by (*simp add*: *exp-diff*)
also have $... \leq (2 * C\ /\ (C\ /\ 2)) * exp\ ((\int u{\in}\{A<..t * x\}.\ r\ u\ /\ u\ \partial lborel)$
$- (\int u{\in}\{A<..t\}.\ r\ u\ /\ u\ \partial lborel))$
using *frac-le*[*OF - T-le2'*[*OF Ttx*] *- T-le2*[*OF t*]] *C-pos* **by**(*intro mult-mono*) *auto*
also have $... = 4 * exp\ ((\int u.\ indicat\text{-}real\ \{A<..t * x\}\ u * r\ u\ /\ u$
$- indicat\text{-}real\ \{A<..t\}\ u * r\ u\ /\ u\ \partial lborel))$
using *integ*[*OF Atx*] *integ*[*OF At*] *C-pos*
by(*auto simp*: *set-lebesgue-integral-def set-integrable-def*)
also have $... = 4 * exp\ (\int u{\in}\{t<..t * x\}.\ r\ u\ /\ u\ \partial lborel)$
using *ttx Atx At* **by**(*auto intro*!: *Bochner-Integration.integral-cong simp*: *set-lebesgue-integral-def indicator-def*)
also have $... \leq 4 * exp\ (\int u{\in}\{t<..t * x\}.\ (\varrho + i)\ /\ u\ \partial lborel)$
using *ttx At T-le1 T t Atx*
by(*fastforce intro*!: *set-integral-mono set-integrable-subset*[*OF integ*,*of* $t * x$] *divide-right-mono integrableI-bounded-set*[**where** $A{=}\{t<..t * x\}$ **and** $B{=}|\varrho + i|\ /\ t$] *divide-mono simp*: *set-borel-measurable-def set-integrable-def*)
also have $... = 4 * exp\ ((\varrho + i) * (ln\ (t * x) - ln\ t))$
by(*subst set-integral-inverse*) (*use A-pos At ttx* **in** *auto*)
also have $... = 4 * exp\ ((\varrho + i) * ln\ x)$
using *A-pos At Atx*
by(*subst ln-divide-pos*[*symmetric*]) *auto*

also have ... = *4* ∗ *x powr* (*ϱ* + *i*)
using *x a* **by**(*simp add*: *powr-def X-def*)
also have ... < *e* / *2*
using *X x* **by** *blast*
finally have ∗:|*U* (*t* ∗ *x*) / *U t*| < *e* / *2*
by (*smt* (*verit*, *best*) *A-pos At rv divide-pos-pos regularly-varying-pos ttx*)
have |*U* (*t* ∗ *x*) / *U t* − *x powr ϱ*| ≤ |*U* (*t* ∗ *x*) / *U t*| + |*x powr ϱ*|
by *linarith*
also have ... < *e* / *2* + *e* / *2*
using ∗ *X*[*OF x*] **by**(*intro add-strict-mono*) *auto*
finally show *?thesis*
by *fastforce*
qed
qed
qed

2.4.3 Potter Bounds

corollary
assumes *rv*:*regularly-varying U ϱ*
and *A*:*1* < *A* **and** *d*:*0* < *d*
shows *Potter-bounds1*:
∃ *T*>*0*. ∀ *x*≥*T*. ∀ *y*≥*x*. *U y* / *U x* ≤ *A* ∗ (*y* / *x*) *powr* (*ϱ* + *d*) (**is** *?g1*)
and *Potter-bounds2*:
∃ *T*>*0*. ∀ *y*≥*T*. ∀ *x*≥*y*. *U y* / *U x* ≤ *A* ∗ (*y* / *x*) *powr* (*ϱ* − *d*) (**is** *?g2*)
and *Potter-bounds*:
∃ *T*>*0*. ∀ *x*≥*T*. ∀ *y*≥*T*. *U y* / *U x* ≤ *A* ∗ *max* ((*y* / *x*) *powr* (*ϱ* + *d*)) ((*y* / *x*) *powr* (*ϱ* − *d*))
(**is** *?g3*)
proof −
from *Karamata-representation-ex*[*OF rv*]
obtain *B c r C* **where** *A-pos*:*0* < *B*
and *eq-U*:⋀*x*. *B* ≤ *x* ⟹ *U x* = *c x* ∗ *exp* (∫ *t*∈{*B*<..*x*}. *r t* / *t* ∂*lborel*)
and *integ*:⋀*x*. *B* ≤ *x* ⟹ *set-integrable lborel* {*B*<..*x*} (λ*t*. *r t* / *t*)
and *lim-r*:(*r* ⟶ *ϱ*) *at-top*
and *lim-c*:(*c* ⟶ *C*) *at-top*
and *C-pos*:*0* < *C*
by *blast*
obtain *T1* **where** *T1*:⋀*t*. *T1* ≤ *t* ⟹ *r t* ≤ *ϱ* + *d* ⋀*t*. *T1* ≤ *t* ⟹ *ϱ* − *d* ≤ *r t*
using *tendstoD*[*OF lim-r d*] **unfolding** *eventually-at-top-linorder dist-real-def*
by (*metis abs-diff-le-iff less-eq-real-def*)
have *pos*:*C* ∗ (*A* − *1*) / (*A* + *1*) > *0*
using *A C-pos* **by** *fastforce*
obtain *T2* **where** ⋀*t*. *T2* ≤ *t* ⟹ |*c t* − *C*| ≤ *C* ∗ (*A* − *1*) / (*A* + *1*)
using *tendstoD*[*OF lim-c pos*] **unfolding** *eventually-at-top-linorder dist-real-def*
by *fastforce*
hence *T2'*:*C* − *C* ∗ (*A* − *1*) / (*A* + *1*) ≤ *c t c t* ≤ *C* + *C* ∗ (*A* − *1*) / (*A* + *1*)
if *t*:*T2* ≤ *t* **for** *t*

```
  using t by(auto simp: abs-diff-le-iff)
 have ct-pos:0 < c t if t:T2 ≤ t for t
  using T2'[OF t] C-pos A
  by (smt (verit, ccfv-threshold) mult.commute le-divide-eq le-divide-eq-1
      nonzero-mult-divide-mult-cancel-left2)
 have T2:c y / c x ≤ A if x:T2 ≤ x and y:T2 ≤ y for x y
 proof −
  have c y / c x ≤ (C + C * (A − 1) / (A + 1)) / (C − C * (A − 1) / (A +
1))
   using T2'[OF x] T2'[OF y] ct-pos[OF y] C-pos A
   by(auto intro!: divide-mono simp del: times-divide-eq-right
     simp: times-divide-eq-right[symmetric])
  also have ... = (C * (1 + (A − 1) / (A + 1))) / (C * (1 − (A − 1) / (A +
1)))
   by (simp add: distrib-left right-diff-distrib')
  also have ... = (C * ((A + 1) / (A + 1) + (A − 1) / (A + 1)))
                / (C * ((A + 1) / (A + 1) − (A − 1) / (A + 1)))
   using A by simp
  also have ... = A
   unfolding add-divide-distrib[symmetric] diff-divide-distrib[symmetric]
   using C-pos A by simp
  finally show ?thesis .
 qed
 define T where T ≡ max (max 1 B) (max T1 T2)
 have T-pos: 0 < T and TB:B ≤ T
  and Tr1:⋀t. T ≤ t ⟹ r t ≤ ϱ + d and Tr2:⋀t. T ≤ t ⟹ ϱ − d ≤ r t
  and Tc:⋀x y. T ≤ x ⟹ T ≤ y ⟹ c y / c x ≤ A
  using T1 T2 by(auto simp: T-def)
 have 1:U y / U x ≤ A * (y / x) powr (ϱ + d) if x:T ≤ x and xy:x ≤ y for x y
 proof −
  have y:T ≤ y
   using x xy by simp
  have U y / U x = c y * exp (∫ t∈{B<..y}. r t / t ∂lborel)
                 / (c x * exp (∫ t∈{B<..x}. r t / t ∂lborel))
   using x y eq-U TB by simp
  also have ... = (c y / c x)
                  * exp ((∫ t∈{B<..y}. r t / t ∂lborel) − (∫ t∈{B<..x}. r t / t
∂lborel))
   by (simp add: exp-diff)
  also have ... ≤ A * exp ((∫ t∈{B<..y}. r t / t ∂lborel) − (∫ t∈{B<..x}. r t /
t ∂lborel))
   using Tc[OF x y] A by(intro mult-mono) auto
  also have ... = A * exp ((∫ t. indicat-real {B<..y} t * r t / t
                                − indicat-real {B<..x} t * r t / t ∂lborel))
   using integ[of x] integ[of y] TB x y A
   by(auto simp: set-lebesgue-integral-def set-integrable-def)
  also have ... = A * exp ((∫ t∈{x<..y}. r t / t ∂lborel))
   using x y TB xy by(auto intro!: Bochner-Integration.integral-cong
     simp: set-lebesgue-integral-def indicator-def)
```

```
  also have ... ≤ A * exp ((∫ t∈{x<..y}. (ϱ + d) / t ∂lborel))
    using A x TB T-pos Tr1 y
    by(auto intro!: set-integral-mono set-integrable-subset[OF integ[of y]]
        set-integrable-inverse' divide-right-mono)
  also have ... = A * exp ((ϱ + d) * ln (y / x))
    using x y xy T-pos
    by(subst set-integral-inverse) (auto simp: ln-divide-pos)
  also have ... = A * (y / x) powr (ϱ + d)
    using x y T-pos by(simp add: powr-def)
  finally show ?thesis .
 qed
 have 2:U y / U x ≤ A * (y / x) powr (ϱ - d) if yx:y ≤ x and y:T ≤ y for x y
 proof -
  have x:T ≤ x
    using yx y by simp
  have U y / U x = c y * exp (∫ t∈{B<..y}. r t / t ∂lborel)
                 / (c x * exp (∫ t∈{B<..x}. r t / t ∂lborel))
    using x y eq-U TB by simp
  also have ... = (c y / c x)
                  * exp ((∫ t∈{B<..y}. r t / t ∂lborel) - (∫ t∈{B<..x}. r t / t
∂lborel))
    by (simp add: exp-diff)
  also have ... ≤ A * exp ((∫ t∈{B<..y}. r t / t ∂lborel) - (∫ t∈{B<..x}. r t /
t ∂lborel))
    using Tc[OF x y] A by(intro mult-mono) auto
  also have ... = A * exp ((∫ t. indicat-real {B<..y} t * r t / t
                                - indicat-real {B<..x} t * r t / t ∂lborel))
    using integ[of x] integ[of y] TB x y A
    by(auto simp: set-lebesgue-integral-def set-integrable-def)
  also have ... = A * exp ((∫ t∈{y<..x}. - r t / t ∂lborel))
    using x y TB yx A by(auto intro!: Bochner-Integration.integral-cong
        simp: set-lebesgue-integral-def indicator-def simp del: integral-minus)
  also have ... ≤ A * exp ((∫ t∈{y<..x}. - (ϱ - d) / t ∂lborel))
    using A x TB T-pos Tr2 y yx
    by(auto intro!: set-integral-mono set-integrable-subset[OF integ[of x]]
        set-integrable-inverse' divide-right-mono simp: set-integrable-uminus
        simp del: minus-diff-eq)
  also have ... = A * exp((ϱ - d) * ln (y / x))
    using x y yx T-pos
    by(subst set-integral-inverse)
      (auto simp: ln-divide-pos left-diff-distrib right-diff-distrib)
  also have ... = A * (y / x) powr (ϱ - d)
    using x y T-pos by(simp add: powr-def)
  finally show ?thesis .
 qed
 have 3:U y / U x ≤ A * max ((y / x) powr (ϱ + d)) ((y / x) powr (ϱ - d))
  if x:T ≤ x and y:T ≤ y for x y
 proof(cases x ≤ y)
  assume xy:x ≤ y
```

```
    then have U y / U x ≤ A * (y / x) powr (ϱ + d)
      using 1 x by blast
    thus ?thesis
      using T-pos xy d powr-mono x by force
  next
    assume yx:¬ x ≤ y
    then have U y / U x ≤ A * (y / x) powr (ϱ − d)
      using 2 y by simp
    thus ?thesis
      by (smt (verit, best) A mult-mono powr-ge-zero)
  qed
  show ?g1 ?g2 ?g3
    using 1 2 3 T-pos by blast+
qed

corollary
  assumes rv:regularly-varying U ϱ
    and A:A < 1 and d:0 < d
  shows Potter-bounds-lower1:
    ∃ T>0. ∀ x≥T. ∀ y≥x. A * (y / x) powr (ϱ − d) ≤ U y / U x (is ?g1)
    and Potter-bounds-lower2:
    ∃ T>0. ∀ y≥T. ∀ x≥y. A * (y / x) powr (ϱ + d) ≤ U y / U x (is ?g2)
    and Potter-bounds-lower:
    ∃ T>0. ∀ x≥T. ∀ y≥T. A * min ((y / x) powr (ϱ + d)) ((y / x) powr (ϱ − d))
≤ U y / U x
    (is ?g3)
proof −
  show g1:?g1
  proof(cases A ≤ 0)
    assume A ≤ 0
    thus ?thesis
      by (smt (verit, best) divide-pos-pos mult-cancel-right2 mult-less-cancel-left
          powr-non-neg regularly-varying-pos rv)
  next
    assume A-pos:¬ A ≤ 0
    then have invA:1 < 1 / A
      using A by force
    from Potter-bounds2[OF rv this d]
    obtain T where T-pos:T > 0 and T:⋀x y. y ≥ T ⟹ x ≥ y ⟹
                    U y / U x ≤ 1 / A * (y / x) powr (ϱ − d)
      by fast
    have A * (y / x) powr (ϱ − d) ≤ U y / U x if x:T ≤ x and xy:x ≤ y for x y
    proof −
      have 1 / (1 / A * (x / y) powr (ϱ − d)) ≤ 1 / (U x / U y)
        using A-pos regularly-varying-pos[OF rv] T-pos x xy
        by(intro divide-left-mono T[OF x xy] T[OF x xy]) auto
      thus ?thesis
        by (simp add: powr-divide)
    qed
```

```
    thus ?thesis
      using T-pos by blast
  qed
  show g2:?g2
  proof(cases A ≤ 0)
    assume A ≤ 0
    thus ?thesis
      by (smt (verit, best) divide-pos-pos mult-cancel-right2 mult-less-cancel-left
          powr-non-neg regularly-varying-pos rv)
  next
    assume A-pos:¬ A ≤ 0
    then have invA:1 < 1 / A
      using A by force
    from Potter-bounds1[OF rv this d]
    obtain T where T-pos:T > 0 and T:⋀x y. x ≥ T ⟹ y ≥ x ⟹
                    U y / U x ≤ 1 / A * (y / x) powr (ϱ + d)
      by fast
    have A * (y / x) powr (ϱ + d) ≤ U y / U x if y:T ≤ y and yx:y ≤ x for x y
    proof −
      have 1 / (1 / A * (x / y) powr (ϱ + d)) ≤ 1 / (U x / U y)
        using A-pos regularly-varying-pos[OF rv] T-pos y yx
        by(intro divide-left-mono T[OF y yx]) auto
      thus ?thesis
        by (simp add: powr-divide)
    qed
    thus ?thesis
      using T-pos by blast
  qed
  from g1 g2 obtain T1 T2 where
    T1:T1 > 0 ∀ x≥T1. ∀ y≥x. A * (y / x) powr (ϱ − d) ≤ U y / U x
    and T2:T2 > 0 ∀ y≥T2. ∀ x≥y. A * (y / x) powr (ϱ + d) ≤ U y / U x
    by auto
  thus ?g3
    by(safe intro!: exI[where x=max T1 T2] max.strict-coboundedI1)
      (smt (verit, best) assms(3) divide-pos-pos less-divide-eq-1-pos powr-less-mono
powr-mono')
qed
```

2.5 Karamata's Theorem

2.5.1 Direct Half

```
lemma Karamata-theorem-half1:
  assumes rv: regularly-varying U ϱ
    and a: 0 ≤ a and rho:− 1 < ϱ
    and integrable:⋀c d. a ≤ c ⟹ set-integrable lborel {c<..d} U
  shows ((λt. (∫ x∈{a<..t}. U x ∂lborel) / (t * U t)) ⟶ 1 / (ϱ + 1)) at-top
proof −
  obtain d where d-pos:0 < d and d:d < ϱ + 1
    by (metis rho minus-less-iff real-0-less-add-iff real-arch-inverse)
```

with *Potter-bounds2*[*OF rv*,*of 2 d*]
obtain T' **where** T':$T' > 0$ $\bigwedge x\ y.\ T' \leq y \Longrightarrow y \leq x \Longrightarrow U\ y\ /\ U\ x \leq 2 * (y\ /\ x)\ powr\ (\varrho - d)$
by *auto*
define T **where** $T \equiv max\ T'\ a$
have *T-pos*:$0 < T$ **and** aT:$a \leq T$
and T:$\bigwedge x\ y.\ T \leq y \Longrightarrow y \leq x \Longrightarrow U\ y\ /\ U\ x \leq 2 * (y\ /\ x)\ powr\ (\varrho - d)$
using T' **by**(*auto simp*: *T-def*)
have *eq*:$(\int x{\in}\{T{<}..t\}.\ U\ x\ \partial lborel)\ /\ (t * U\ t) = (\int s{\in}\{T/t{<}..1\}.\ U\ (t * s)\ /\ U\ t\ \partial lborel)$
if t:$T \leq t$ **for** t
proof −
have $(\int x{\in}\{T{<}..t\}.\ U\ x\ \partial lborel) = t * (\int s.\ indicat\text{-}real\ \{T{<}..t\}\ (t * s) *_R U\ (t * s)\ \partial lborel)$
using *lborel-integral-real-affine*[*of t* $\lambda x.\ indicat\text{-}real\ \{T{<}..t\}\ x * U\ x\ 0$] *T-pos t*
by(*auto simp*: *set-lebesgue-integral-def*)
also have $... = t * (\int s{\in}\{T/t{<}..1\}.\ U\ (t * s)\ \partial lborel)$
unfolding *set-lebesgue-integral-def* **using** *T-pos t*
by(*intro arg-cong*[**where** $f{=}(*)$ t]
Bochner-Integration.integral-cong arg-cong[**where** $f{=}\lambda y.\ y *_R U\ (t * -)$])
(*auto simp*: *mult.commute indicator-def pos-divide-less-eq*)
finally show *?thesis*
using *T-pos t* **by** *simp*
qed
have *set-integrable lborel* $\{T/t{<}..1\}$ $(\lambda s.\ U\ (t * s)\ /\ U\ t)$
if t:$T < t$ **for** t
proof −
have $*$:$indicat\text{-}real\ \{T{<}..t\}\ (t * x) = indicat\text{-}real\ \{T/t{<}..1\}\ x$ **for** x
using *T-pos t* **by**(*auto simp*: *indicator-def divide-less-eq mult.commute*)
have *integrable lborel* $(\lambda x.\ indicat\text{-}real\ \{T{<}..t\}\ (t * x) * U\ (t * x))$
using *lborel-integrable-real-affine*[*of* $\lambda x.\ indicat\text{-}real\ \{T{<}..t\}\ x * U\ x\ t\ 0$]
T-pos aT integrable[*of T t*] *t*
by(*auto simp*: *set-integrable-def*)
thus *?thesis*
unfolding $*$ **by**(*simp add*: *set-integrable-def*)
qed
hence $*$:$(\lambda s.\ indicat\text{-}real\ \{T/t{<}..1\}\ s * U\ (t * s)\ /\ U\ t) \in borel\text{-}measurable\ lborel$
if t:$T + 1 \leq t$ **for** t
using t *T-pos* **by**(*auto simp*: *set-integrable-def*)
have $**$:$integrable\ lborel\ (\lambda s.\ indicator\ \{0{<}..1\}\ s * 2 * s\ powr\ (\varrho - d))$
using *d set-integrable-mult-right-iff*[*THEN iffD2*,*OF - set-integrable-powr-from-0*[*of* $\varrho - d$]]
by(*auto simp*: *set-integrable-def mult.assoc*)
have $***$:*AE s in lborel.*
$((\lambda t.\ indicat\text{-}real\ \{T/t{<}..1\}\ s * U\ (t * s)\ /\ U\ t) \longrightarrow indicat\text{-}real\ \{0{<}..1\}\ s * s\ powr\ \varrho)\ at\text{-}top$
proof(*rule AE-I2*)
fix s :: *real*

```
    consider 0 < s s ≤ 1 | s ≤ 0 | 1 < s
      by linarith
    then show ((λt. indicat-real {T / t<..1} s * U (t * s) / U t)
          ⟶ indicat-real {0<..1} s * s powr ϱ) at-top
    proof cases
      assume s:0 < s s ≤ 1
      then obtain X where X:0 < X T / X < s
          by (metis linordered-field-no-ub mult.commute not-less-iff-gr-or-eq or-
der.strict-trans
          pos-divide-less-eq)
      hence ⋀t. X ≤ t ⟹ T / t < s
        by (metis T-pos order.strict-trans frac-less2 less-eq-real-def)
      hence ∀F t in at-top. indicat-real {T / t<..1} s * U (t * s) / U t = U (t *
s) / U t
        using s by(auto intro!: eventually-at-top-linorderI[where c=X])
      moreover have ((λt. U (t * s) / U t) ⟶ indicat-real {0<..1} s * s powr
ϱ) at-top
        using regularly-varying-lim[OF rv] s
        by(simp add: indicator-def)
      ultimately show ?thesis
        by(simp only: tendsto-cong)
    next
      assume s:s ≤ 0
      hence ∀F t in at-top. indicat-real {T / t<..1} s * U (t * s) / U t = 0
        by(intro eventually-at-top-linorderI[where c=1])
          (simp add: linorder-class.not-less[of T / - s]
            T-pos dual-order.trans[of 0 T / - s]
            less-eq-real-def[of 0])
      thus ?thesis
        using s by (simp add: tendsto-eventually)
    qed simp
  qed
  have ****:AE x in lborel. norm (indicat-real {T / t<..1} x * U (t * x) / U t)
                  ≤ indicat-real {0<..1} x * 2 * x powr (ϱ − d)
    if t:T + 1 ≤ t for t
  proof(rule AE-I2)
    fix s::real
    consider T / t < s s ≤ 1 | s ≤ T / t | 1 < s
      by linarith
    then show norm (indicat-real {T / t<..1} s * U (t * s) / U t)
                  ≤ indicat-real {0<..1} s * 2 * s powr (ϱ − d)
    proof cases
      assume s:T / t < s s ≤ 1
      then have Tts:T ≤ t * s t * s ≤ t and s-pos:s > 0
        using t T-pos by(auto simp: mult.commute pos-divide-less-eq
          order.strict-trans[of 0 T / t s])
      have norm (indicat-real {T / t<..1} s * U (t * s) / U t)
          = U (t * s) / U t
        using s T-pos t s-pos
```

```
      by(auto simp del: abs-divide intro!: abs-of-pos
          regularly-varying-pos[OF rv] divide-pos-pos)
    also have ... ≤ indicat-real {0<..1} s * 2 * s powr (ϱ - d)
      using s-pos T[OF Tts] t s T-pos by auto
    finally show ?thesis .
  qed auto
 qed
 have lim:((λt. (∫ s∈{T/t<..1}. U (t * s) / U t ∂lborel))
          ⟶ (∫ s∈{0<..1}. s powr ϱ ∂lborel)) at-top
  using integral-dominated-convergence-real[OF - * ** *** ****]
  by(auto simp: set-lebesgue-integral-def)
 have cong:∀ F t in at-top.
       (∫ x∈{a<..t}. U x ∂lborel) / (t * U t)
        = (∫ x∈{a<..T}. U x ∂lborel)/ (t * U t) + (∫ s∈{T/t<..1}. U (t * s) /
U t ∂lborel)
 proof(safe intro!: eventually-at-top-linorderI[where c=T])
  fix t
  assume t:T ≤ t
   with aT have (∫ x∈{a<..t}. U x ∂lborel) = (∫ x∈{a<..T} ∪ {T<..t}. U x
∂lborel)
    by(intro arg-cong[where f=λB. set-lebesgue-integral lborel B -]) auto
  also have ... = (∫ x∈{a<..T}. U x ∂lborel) + (∫ x∈{T<..t}. U x ∂lborel)
    by(intro set-integral-Un integrable aT) auto
  finally show (∫ x∈{a<..t}. U x ∂lborel) / (t * U t)
       = (∫ x∈{a<..T}. U x ∂lborel)/ (t * U t) + (∫ s∈{T/t<..1}. U (t * s) /
U t ∂lborel)
    by(simp add: add-divide-distrib eq[OF t])
 qed
 have lim2:((λt. (∫ x∈{a<..T}. U x ∂lborel)/ (t * U t)
              + (∫ s∈{T/t<..1}. U (t * s) / U t ∂lborel))
          ⟶ 0 + 1 / (ϱ + 1)) at-top
  using rho lim set-integral-powr-from-0
  by(intro tendsto-add real-tendsto-divide-at-top
      regularly-varying-pos-at-top[where ϱ=1 + ϱ] regularly-varying-mult rv)
     auto
 show ?thesis
  using lim2 cong tendsto-cong
  by fastforce
qed


lemma Karamata-theorem-half-uminus1:
 assumes rv:regularly-varying U (- 1)
  and a: 0 ≤ a
  and integrable:⋀c d. a ≤ c ⟹ set-integrable lborel {c<..d} U
 shows filterlim (λt. (∫ x∈{a<..t}. U x ∂lborel) / (t * U t)) at-top at-top
 unfolding filterlim-at-top
proof safe
 fix Z :: real
```

define Z' **where** $Z' \equiv max\ (Z + 1)\ 2$
define c **where** $c \equiv exp\ (-\ Z')$
have *c-pos*:$0 < c$ **and** *c-1*:$c < 1$
by (*simp-all add*: *c-def Z'-def*)
obtain *T1* **where** *T1*:$\bigwedge t.\ T1 \leq t \Longrightarrow a \leq c * t$
by (*metis mult.commute c-pos pos-divide-le-eq*)
obtain *T2* **where** *T2*:$\bigwedge t\ x.\ T2 \leq t \Longrightarrow c \leq x \Longrightarrow x \leq 1$
$\Longrightarrow |U\ (t * x)\ /\ U\ t - x\ powr\ (-\ 1)| < 1$
using *uniform-limitD*[*OF regularly-varying-UCT-interval*[*OF rv c-pos*,*of 1*],*of 1*]
by(*fastforce simp*: *eventually-at-top-linorder dist-real-def*)
define T **where** $T \equiv max\ 1\ (max\ T1\ T2)$
have *Tact*:$\bigwedge t.\ T \leq t \Longrightarrow a \leq c * t$
by(*auto simp*: *T-def intro*!: *T1*)
have *Tle*:$\bigwedge t\ x.\ T \leq t \Longrightarrow c \leq x \Longrightarrow x \leq 1 \Longrightarrow 1\ /\ x - 1 \leq U\ (t * x)\ /\ U\ t$
using *T2 T-def c-pos* **by** *fastforce*
show $\forall_F\ t$ *in at-top*. $Z \leq (\int x{\in}\{a{<}..t\}.\ U\ x\ \partial lborel)\ /\ (t * U\ t)$
proof(*safe intro*!: *eventually-at-top-linorderI*[**where** $c{=}T$])
fix t
assume t:$T \leq t$
then have *t-pos*:$0 < t$
by(*simp add*: *T-def*)
have *integ1*[*simp*]:*set-integrable lborel* $\{c * t{<}..t\}$ U
by(*auto intro*!: *assms Tact*[*OF t*])
have *integ2*[*simp*]:*set-integrable lborel* $\{c{<}..1\}$ $(\lambda x.\ U\ (t * x)\ /\ U\ t)$
proof −
have $*$:*indicat-real* $\{c * t{<}..t\}$ $(t * x)$ = *indicat-real* $\{c{<}..1\}$ x **for** x
using *t-pos* **by**(*auto simp*: *indicator-def divide-less-eq mult.commute*)
have *integrable lborel* $(\lambda x.\ indicat\text{-}real\ \{c * t{<}..t\}\ (t * x) * U\ (t * x))$
using *lborel-integrable-real-affine*[*of* $\lambda x.\ indicat\text{-}real\ \{c * t{<}..t\}\ x * U\ x\ t\ 0$]
integ1 t-pos
unfolding *set-integrable-def*
by *simp*
thus *?thesis*
unfolding $*$ **by**(*simp add*: *set-integrable-def*)
qed
note [*simp*] = *set-integral-diff*[*OF set-integrable-inverse'*[*OF c-pos*]
set-integrable-const[*of lborel* $\{c{<}..1\}$]]

have $Z \leq Z' - 1$
by(*simp add*: *Z'-def*)
also have ... $= -\ ln\ c - 1$
by(*simp add*: *c-def*)
also have ... $\leq -\ ln\ c - (1 - c)$
using *c-1 c-pos* **by** *argo*
also have ... $= (\int x{\in}\{c{<}..1\}.\ 1\ /\ x\ \partial lborel) - (1 - c)$
using *c-pos c-1* **by**(*subst set-integral-inverse*) *auto*
also have ... $= (\int x{\in}\{c{<}..1\}.\ 1\ /\ x\ \partial lborel) - (\int x{\in}\{c{<}..1\}.\ 1\ \partial lborel)$
using *c-1* **by**(*subst set-integral-const*) *auto*

```
    also have ... = (∫ x∈{c<..1}. 1 / x - 1 ∂lborel)
      using c-1 by auto
    also have ... ≤ (∫ x∈{c<..1}. U (t * x) / U t ∂lborel)
      using t c-1 by(intro set-integral-mono Tle) auto
    also have ... = (∫ x∈{c * t<..t}. U x ∂lborel) / (t * U t)
    proof -
      have (∫ x∈{c * t<..t}. U x ∂lborel)
            = t * (∫ s. indicat-real {c * t<..t} (t * s) *R U (t * s) ∂lborel)
        using lborel-integral-real-affine[of t λx. indicat-real {c*t<..t} x * U x 0]
t-pos
        by(auto simp: set-lebesgue-integral-def)
      also have ... = t * (∫ s∈{c<..1}. U (t * s) ∂lborel)
        unfolding set-lebesgue-integral-def using t-pos
        by(intro arg-cong[where f=(*) t]
          Bochner-Integration.integral-cong arg-cong[where f=λy. y *R U (t * -)])
          (auto simp: mult.commute indicator-def pos-divide-less-eq)
      finally show ?thesis
        using t-pos by simp
    qed
    also have ... ≤ (∫ x∈{a<..t}. U x ∂lborel) / (t * U t)
      using t-pos less-imp-le[OF regularly-varying-pos[OF rv]]
        integrable[OF order.refl] Tact[OF t] a
      by(intro divide-right-mono set-integral-subset) auto
    finally show Z ≤ (∫ x∈{a<..t}. U x ∂lborel) / (t * U t) .
  qed
qed

corollary Karamata-theorem-half1-inverse:
  assumes rv: regularly-varying U ϱ
    and rho:- 1 ≤ ϱ and a: 0 ≤ a
    and integrable:⋀c d. a ≤ c ⟹ set-integrable lborel {c<..d} U
  shows ((λt. (t * U t) / (∫ x∈{a<..t}. U x ∂lborel)) ⟶ ϱ + 1) at-top
proof(cases ϱ = -1)
    show ϱ = -1 ⟹ ?thesis
      using tendsto-inverse-0-at-top[OF Karamata-theorem-half-uminus1[OF - a
integrable]] rv
      by simp
qed(use assms tendsto-inverse[OF Karamata-theorem-half1] in auto)

lemma Karamata-theorem-half1-ex:
  assumes rv: regularly-varying U ϱ
    and rho:- 1 ≤ ϱ
  shows ∃ a. ((λt. (t * U t) / (∫ x∈{a<..t}. U x ∂lborel)) ⟶ ϱ + 1) at-top
proof -
  from regularly-varying-eventually-locally-integrable[OF rv]
  obtain x' where x':⋀a b. x' ≤ a ⟹ set-integrable lborel {a..b} U
    by (metis (lifting) eventually-at-top-linorder)
  define x where x ≡ max 0 x'
  have x:0 ≤ x ⋀c d. x ≤ c ⟹ set-integrable lborel {c<..d} U
```

by(*auto intro*!: *set-integrable-subset*[**where** *A*={-..-} **and** *B*={-<..-}] *x′ simp*: *x-def*)
from *Karamata-theorem-half1-inverse*[*OF rv rho this*]
show *?thesis*
by *fast*
qed

lemma *regularly-varying-integral-limit*:
assumes *rv*: *regularly-varying U ϱ*
and *rho*:− *1* ≤ *ϱ* **and** *a*: *0* ≤ *a*
and *integrable*:⋀*c d*. *a* ≤ *c* ⟹ *set-integrable lborel* {*c*<..*d*} *U*
and *x*: *0* < *x*
shows ((λ*t*. (∫ *s*∈{*a*<..*t* ∗ *x*}. *U s* ∂*lborel*) / (∫ *s*∈{*a*<..*t*}. *U s* ∂*lborel*))
⟶ *x powr* (*ϱ* + *1*)) *at-top*
(**is** (*?g* ⟶ *x powr* (*ϱ* + *1*)) *at-top*)
proof(*cases ϱ* = −*1*)
assume *ϱ* ≠ − *1*
then have *rho′*:− *1* < *ϱ*
using *rho* **by** *force*
have (λ*t*. (∫ *s*∈{*a*<..*t* ∗ *x*}. *U s* ∂*lborel*) / (∫ *s*∈{*a*<..*t*}. *U s* ∂*lborel*)) ∼[*at-top*]
(λ*t*. (*t* ∗ *x* ∗ *U* (*t* ∗ *x*)) / (*t* ∗ *U t*))
using *asymp-equiv-divide*[*OF asymp-equivI′-const*[*OF*
filterlim-compose[*OF Karamata-theorem-half1*[*OF rv a rho′ integrable*]
filterlim-tendsto-pos-mult-at-top[*OF tendsto-const x filterlim-ident*]]]
asymp-equivI′-const[*OF Karamata-theorem-half1*[*OF rv a rho′ integrable*]]] *rho′*
by(*simp add*: *mult.commute*)
moreover have (λ*t*. (*t* ∗ *x* ∗ *U* (*t* ∗ *x*)) / (*t* ∗ *U t*))
∼[*at-top*] (λ*t*. (*x* ∗ (*U* (*t* ∗ *x*)) / *U t*))
using *x regularly-varying-pos*[*OF rv*] *regularly-varying-pos*[*OF rv mult-pos-pos*[*OF - x*]]
by(*intro asymp-equivI′ tendstoI eventually-at-top-linorderI*[**where** *c*=*1*])
(*smt* (*verit*, *ccfv-threshold*) *dist-real-def divide-divide-eq-right*
divide-divide-times-eq divide-eq-1-iff nonzero-mult-div-cancel-left)
ultimately have (λ*t*. (∫ *s*∈{*a*<..*t* ∗ *x*}. *U s* ∂*lborel*) / (∫ *s*∈{*a*<..*t*}. *U s* ∂*lborel*))
∼[*at-top*] (λ*t*. (*x* ∗ (*U* (*t* ∗ *x*)) / *U t*))
using *asymp-equiv-trans* **by** *blast*
moreover have ((λ*t*. (*x* ∗ (*U* (*t* ∗ *x*)) / *U t*)) ⟶ *x powr* (*ϱ* + *1*)) *at-top*
using *tendsto-mult*[*OF tendsto-const*[*of x*] *regularly-varying-lim*[*OF rv x*]]
powr-mult-base′ x
by (*simp add*: *add.commute powr-mult-base*)
ultimately show *?thesis*
using *tendsto-asymp-equiv-cong* **by** *blast*
next
assume *rho′*:*ϱ* = − *1*
show *?thesis*
proof(*cases set-integrable lborel* {*a*<..} *U*)
assume *h*:*set-integrable lborel* {*a*<..} *U*
then have ∗:((λ*t*. (∫ *s*∈{*a*<..*t*}. *U s* ∂*lborel*)) ⟶ (∫ *s*∈{*a*<..}. *U s* ∂*lborel*)) *at-top*

```
    using a
    by(intro interval-integral-inf2 less-imp-le[OF regularly-varying-pos[OF rv]]
        set-integrable-subset[OF h])
       auto
  have 0 < (∫ s∈{a<..}. U s ∂lborel)
    using a
    by(auto intro!: set-integral-zero-less h regularly-varying-pos[OF rv]
       order.strict-trans2[where b=emeasure lborel {a<..a+1} and c=emeasure
lborel {a<..},
         OF - emeasure-mono])
  thus ?thesis
    using tendsto-divide[OF filterlim-compose[OF
        * filterlim-tendsto-pos-mult-at-top[OF tendsto-const x filterlim-ident]]
       *] x
    by(simp add: mult.commute rho')
 next
  assume h:¬ set-integrable lborel {a<..} U
  have integ-infa:(∫⁺s∈{a<..}. ennreal (U s) ∂lborel) = ⊤
  proof −
    have ¬ (∫⁺ x. ennreal |indicat-real {a<..} x * U x| ∂lborel) < ∞
      using h regularly-varying-meas-o[OF rv a]
      by(auto simp: set-integrable-def integrable-iff-bounded)
    moreover have (∫⁺ x. ennreal |indicat-real {a<..} x * U x| ∂lborel)
                  = (∫⁺s∈{a<..}. U s ∂lborel)
      using a regularly-varying-pos[OF rv]
      by(auto intro!: nn-integral-cong simp: indicator-def abs-of-pos)
    ultimately show ?thesis
      using top.not-eq-extremum by fastforce
  qed
  have filterlim1:filterlim (λt. (∫ s∈{a<..t}. U s ∂lborel)) at-top at-top
  proof −
    have 1:∀ F t in at-top.
           ennreal (∫ s∈{a<..t}. U s ∂lborel) = (∫⁺s∈{a<..t}. ennreal (U s)
∂lborel)
      using integrable[OF order.refl] a
      unfolding nn-integral-set-ennreal set-lebesgue-integral-def
      apply(intro eventually-at-top-linorderI[where c=a + 1])
      by(subst nn-integral-eq-integral)
        (auto simp: set-integrable-def mult.commute indicator-def
          less-imp-le[OF regularly-varying-pos[OF rv]])
    have 2:((λt. (∫⁺s∈{a<..t}. ennreal (U s) ∂lborel))
             ⟶ (∫⁺s∈{a<..}. ennreal (U s) ∂lborel)) at-top
    proof(rule nn-integral-at-top-real[where K=0])
      show mono-on {0..} (λr x. ennreal (U x) * indicator {a<..r} x)
        by(auto intro!: mono-onI le-funI simp: indicator-def)
    next
     show ⋀t. (λx. ennreal (U x) * indicator {a<..t} x) ∈ borel-measurable lborel
        using regularly-varying-meas-Ioc-ennreal[OF rv a] by simp
    next
```

fix y
show $((\lambda t.\ ennreal\ (U\ y) * indicator\ \{a<..t\}\ y) \longrightarrow$
$ennreal\ (U\ y) * indicator\ \{a<..\}\ y)\ at\text{-}top$
proof(*cases* $a < y$)
assume $a < y$
then obtain T **where** $\bigwedge t.\ T \leq t \Longrightarrow y \leq t$
by *simp*
thus *?thesis*
unfolding *mult.commute*[**where** *a*=*ennreal* (*U y*)] *indicator-mult-ennreal*
by(*intro tendsto-ennrealI tendstoI eventually-at-top-linorderI*[**where** *c*=*T*])
(*auto simp*: *dist-real-def indicator-def*)
qed *simp*
qed
show *?thesis*
unfolding *ennreal-tendsto-top-eq-at-top*[*symmetric*]
using *1 2 integ-infa*
by (*simp add*: *tendsto-cong*)
qed
have *filterlim2*:*filterlim* $(\lambda t.\ (\int s\in\{X<..t\}.\ U\ s\ \partial lborel))\ at\text{-}top\ at\text{-}top$
if *X*:$a \leq X$ **for** X
proof −
have $\forall_F\ t\ in\ at\text{-}top.\ (\int s\in\{a<..t\}.\ U\ s\ \partial lborel)$
$= (\int s\in\{a<..X\}.\ U\ s\ \partial lborel) + (\int s\in\{X<..t\}.\ U\ s\ \partial lborel)$
using *X* **unfolding** *eventually-at-top-linorder*
by(*subst set-integral-Un*[*symmetric*])
(*auto intro*!: *integrable arg-cong*[**where** *f*=$\lambda B.\ (\int s\in B.\ U\ s\ \partial lborel)$] *exI*[**where** *x*=*X*])
hence $\forall_F\ t\ in\ at\text{-}top.\ (\int s\in\{X<..t\}.\ U\ s\ \partial lborel)$
$= -(\int s\in\{a<..X\}.\ U\ s\ \partial lborel) + (\int s\in\{a<..t\}.\ U\ s\ \partial lborel)$
by (*rule eventually-mono*) *simp*
moreover have *filterlim* $(\lambda t.\ -(\int s\in\{a<..X\}.\ U\ s\ \partial lborel) + (\int s\in\{a<..t\}.\ U\ s\ \partial lborel))$
at-top at-top
by(*subst filterlim-tendsto-add-at-top-iff*)
(*auto simp*: *filterlim1*)
ultimately show *?thesis*
by(*simp add*: *filterlim-cong*)
qed
have *integ-pos*:$\bigwedge b\ c.\ a \leq b \Longrightarrow b < c \Longrightarrow 0 < (\int s\in\{b<..c\}.\ U\ s\ \partial lborel)$
using *a*
by(*intro set-integral-zero-less set-integrable-subset*[*OF integrable*])
(*auto intro*!: *regularly-varying-pos*[*OF rv*])
have *integ-le-0*:$\forall_F\ t\ in\ at\text{-}top.\ 0 < (\int s\in\{X<..t\}.\ U\ s\ \partial lborel)$ **if** *X*:$a \leq X$ **for** X
using *filterlim-at-top-dense*[*THEN iffD1*,*OF filterlim2*[*OF X*]]
by *fast*
have *asymp-eq*:$(\lambda t.\ (\int s\in\{a<..t\}.\ U\ s\ \partial lborel)) \sim[at\text{-}top]\ (\lambda t.\ \int s\in\{X<..t\}.\ U$

```
s ∂lborel)
    if X:a ≤ X for X
  proof −
    have ∀F t in at-top. (∫ s∈{a<..t}. U s ∂lborel)
                       = (∫ s∈{a<..X}. U s ∂lborel) + (∫ s∈{X<..t}. U s ∂lborel)
      using X unfolding eventually-at-top-linorder
      by(subst set-integral-Un[symmetric])
          (auto intro!: integrable arg-cong[where f=λB. (∫ s∈B. U s ∂lborel)]
exI[where x=X])
    hence ∀F t in at-top. (∫ s∈{a<..t}. U s ∂lborel) / (∫ s∈{X<..t}. U s ∂lborel)
                       = (∫ s∈{a<..X}. U s ∂lborel) / (∫ s∈{X<..t}. U s ∂lborel)
+ 1
      by(rule eventually-mono[OF eventually-conj[OF integ-le-0[OF X]]])
        (use filterlim-at-top[THEN iffD1,OF filterlim2[OF X]]
         in simp add: add-divide-distrib)
   moreover have ((λt. (∫ s∈{a<..X}. U s ∂lborel) / (∫ s∈{X<..t}. U s ∂lborel)
+ 1)
                   ⟶ 0 + 1) at-top
      by(intro tendsto-add real-tendsto-divide-at-top filterlim2 X) auto
    ultimately show ?thesis
      by(auto intro!: asymp-equivI' simp: tendsto-cong)
  qed
  have asymp-eq':(λt. (∫ s∈{a<..t * x}. U s ∂lborel))
                 ∼[at-top] (λt. ∫ s∈{X<..t * x}. U s ∂lborel)
    if X:a ≤ X for X
    using filterlim-tendsto-pos-mult-at-top[OF tendsto-const x filterlim-ident]
    by(intro asymp-equiv-compose'[OF asymp-eq] X)
      (simp add: mult.commute)
  have le1:1
       ≤ Liminf at-top (λt. ereal (?g t))
         + ereal e
    and le2:Limsup at-top (λt. ereal (?g t))
       ≤ 1 + ereal e
      if e:0 < e for e
  proof −
    define e' where e' ≡ e / x
    have e':0 < e'
      using e x by(simp add: e'-def)
    obtain T' where T':⋀t. T' ≤ t ⟹ |U (t * x) / U t − 1 / x| < e'
      using tendstoD[OF regularly-varying-lim[OF rv x] e'] x
      by(auto simp: rho' eventually-at-top-linorder dist-real-def)
    define T where T ≡ max 1 (max (a + 1) (max (a / x) T'))
    have T-pos:0 < T and aT2:a + 1 ≤ T and TT':T' ≤ T and a / x ≤ T
      using x by(auto simp: T-def)
    hence aTx:a ≤ T * x and aT1:a ≤ T
      using x a pos-divide-le-eq by auto
    have T1:T ≤ t ⟹ U (t * x) ≤ U t * (1 / x + e') for t
     using T'[OF order.trans[OF TT'],of t] T-pos regularly-varying-pos[OF rv,of
t]
```

by(*auto simp*: *abs-less-iff diff-less-eq add.commute mult.commute pos-divide-less-eq*)
have *T2*:$T \leq t \Longrightarrow U\ t * (1\ /\ x - e') \leq U\ (t * x)$ **for** t
using *T'*[*OF order.trans*[*OF TT'*],*of t*] *T-pos regularly-varying-pos*[*OF rv*,*of t*]
by (*metis* (*no-types*, *lifting*) *linorder-not-le abs-diff-less-iff dual-order.trans mult.commute mult-imp-div-pos-less not-less-iff-gr-or-eq*)
have *integT*:*set-integrable lborel* $\{T<..y\}$ $(\lambda t.\ U\ (t * x))$ **for** y
proof −
have *1*:*indicat-real* $\{T * x<..y * x\}$ $(s * x) * U\ (s * x)$
$=$ *indicat-real* $\{T<..y\}$ $s * U\ (s * x)$ **for** s
using x **by**(*auto simp*: *indicator-def*)
have *integrable lborel* $(\lambda t.$ *indicat-real* $\{T * x<..y * x\}$ $t * U\ t)$
using *integrable*[*OF aTx*] **unfolding** *set-integrable-def* **by** *simp*
from *lborel-integrable-real-affine*[**where** $c{=}x$ **and** $t{=}0$,*OF this*] x
have *integrable lborel* $(\lambda s.$ *indicat-real* $\{T * x<..y * x\}$ $(s * x) * U\ (s * x))$
by(*simp add*: *mult.commute*)
thus *?thesis*
unfolding *1* **by**(*simp add*: *set-integrable-def*)
qed
have *Limsup at-top* $(\lambda t.$ *ereal* $(?g\ t))$
$=$ *Limsup at-top* $(\lambda t.$ *ereal* $((\int s{\in}\{T * x<..t * x\}.\ U\ s\ \partial lborel)$
$/\ (\int s{\in}\{T<..t\}.\ U\ s\ \partial lborel)))$
using *aTx aT2 aT1 a T-pos x*
by(*intro asymp-equiv-Limsup-cong-ereal*[*OF asymp-equiv-divide*[*OF asymp-eq'*[*OF aTx*]
asymp-eq[*OF aT1*]]]
eventually-at-top-linorderI[**where** $c{=}max\ (T * x + 1)\ (T + 1)$]
less-imp-le divide-pos-pos)
(*auto intro*!: *integ-pos order.strict-trans1*
[*OF aTx mult-less-cancel-right-pos*[*THEN iffD2*,*OF x*]])
also have ... $=$ *Limsup at-top*
$(\lambda t.$ *ereal* $(x * (\int s{\in}\{T<..t\}.\ U\ (s * x)\ \partial lborel)$
$/\ (\int s{\in}\{T<..t\}.\ U\ s\ \partial lborel)))$
proof(*intro Limsup-eq eventually-at-top-linorderI*[**where** $c{=}T * x + 1$])
fix y
assume y:$T * x + 1 \leq y$
have $(\int s{\in}\{T * x<..y * x\}.\ U\ s\ \partial lborel)$
$=$ $x * (\int s.$ *indicat-real* $\{T * x<..y * x\}$ $(s * x) * U\ (s * x)\ \partial lborel)$
using x **unfolding** *set-lebesgue-integral-def*
by(*subst lborel-integral-real-affine*[*of x* $\lambda t.$ *indicat-real* $\{T{*}x<..y{*}x\}$ $t *_R U\ t\ 0$])
(*auto simp*: *mult.commute*)
also have ... $= x * (\int s{\in}\{T<..y\}.\ U\ (s * x)\ \partial lborel)$
using x
by(*auto simp*: *set-lebesgue-integral-def indicator-def*
intro!: *Bochner-Integration.integral-cong*)
finally show *ereal* $((\int s{\in}\{T * x<..y * x\}.\ U\ s\ \partial lborel)\ /\ (\int s{\in}\{T<..y\}.\ U\ s\ \partial lborel))$
$=$ *ereal* $(x * (\int s{\in}\{T<..y\}.\ U\ (s * x)\ \partial lborel)\ /\ (\int s{\in}\{T<..y\}.\ U$

s ∂lborel))
by *presburger*
qed
also have ... ≤ *Limsup at-top* (λ*t*. *ereal* (*x* ∗ (∫ *s*∈{*T*<..*t*}. *U s* ∗ (*1* / *x* + *e′*) *∂lborel*)
/ (∫ *s*∈{*T*<..*t*}. *U s ∂lborel*)))
using *x*
by(*auto intro*!: *Limsup-mono eventually-at-top-linorderI*[**where** *c*=*T* + *1*]
T1 mult-left-mono divide-right-mono set-integral-mono integrable[*OF aT1*]
less-imp-le[*OF integ-pos*[*OF aT1*]] *integT*
simp del: *set-integral-mult-left*)
also have ... = *Limsup at-top* (λ*t*::*real*. *ereal* (*1* + *x* ∗ *e′*))
using *x*
by(*intro Limsup-eq eventually-mono*[*OF integ-le-0*[*OF aT1*]])
(*simp add*: *distrib-left*[*of x*])
also have ... = *ereal* (*1* + *e*)
using *x* **by**(*simp add*: *e′-def Limsup-const*)
also have ... = *1* + *ereal e*
by *simp*
finally show *Limsup at-top* (λ*t*. *ereal* (*?g t*)) ≤ *1* + *ereal e* **.**
have *1* − *ereal e* = *ereal* (*1* − *e*)
by (*simp add*: *one-ereal-def*)
also have ... = *Liminf at-top* (λ*t*::*real*. *ereal* (*1* − *x* ∗ *e′*))
using *x* **by**(*simp add*: *e′-def Liminf-const*)
also have ... = *Liminf at-top* (λ*t*. *ereal* (*x* ∗ (∫ *s*∈{*T*<..*t*}. *U s* ∗ (*1* / *x* − *e′*) *∂lborel*)
/ (∫ *s*∈{*T*<..*t*}. *U s ∂lborel*)))
using *x e′*
by(*intro Liminf-eq eventually-mono*[*OF integ-le-0*[*OF aT1*]])
(*simp add*: *right-diff-distrib*[*of x*])
also have ... ≤ *Liminf at-top* (λ*t*. *ereal* (*x* ∗ (∫ *s*∈{*T*<..*t*}. *U* (*s* ∗ *x*) *∂lborel*)
/ (∫ *s*∈{*T*<..*t*}. *U s ∂lborel*)))
using *x*
by(*auto intro*!: *Liminf-mono eventually-at-top-linorderI*[**where** *c*=*T* + *1*]
T2 mult-left-mono divide-right-mono set-integral-mono integrable[*OF aT1*]
less-imp-le[*OF integ-pos*[*OF aT1*]] *integT*
simp del: *set-integral-mult-left*)
also have ... = *Liminf at-top* (λ*t*. *ereal* ((∫ *s*∈{*T* ∗ *x*<..*t* ∗ *x*}. *U s ∂lborel*)
/ (∫ *s*∈{*T*<..*t*}. *U s ∂lborel*)))
proof(*intro Liminf-eq eventually-at-top-linorderI*[**where** *c*=*T* ∗ *x* + *1*])
fix *y*
assume *y*:*T* ∗ *x* + *1* ≤ *y*
have (∫ *s*∈{*T* ∗ *x*<..*y* ∗ *x*}. *U s ∂lborel*)
= *x* ∗ (∫ *s*. *indicat-real* {*T* ∗ *x*<..*y* ∗ *x*} (*s* ∗ *x*) ∗ *U* (*s* ∗ *x*) *∂lborel*)
using *x* **unfolding** *set-lebesgue-integral-def*
by(*subst lborel-integral-real-affine*
[*of x* λ*t*. *indicat-real* {*T*∗*x*<..*y*∗*x*} *t* ∗$_R$ *U t 0*])
(*auto simp*: *mult.commute*)
also have ... = *x* ∗ (∫ *s*∈{*T*<..*y*}. *U* (*s* ∗ *x*) *∂lborel*)

```
        using x
        by(auto simp: set-lebesgue-integral-def indicator-def
            intro!: Bochner-Integration.integral-cong)
      finally show ereal (x * (∫ s∈{T<..y}. U (s * x) ∂lborel) / (∫ s∈{T<..y}.
U s ∂lborel))
                  = ereal ((∫ s∈{T * x<..y * x}. U s ∂lborel) / (∫ s∈{T<..y}. U
s ∂lborel))
        by presburger
    qed
    also have ... = Liminf at-top (λt. ereal (?g t))
      using aTx aT2 aT1 a T-pos x
    by(intro asymp-equiv-Liminf-cong-ereal[OF asymp-equiv-divide[OF asymp-eq'[OF
aTx]
            asymp-eq[OF aT1]],symmetric]
          eventually-at-top-linorderI[where c=max (T * x + 1) (T + 1)]
          less-imp-le divide-pos-pos)
        (auto intro!: integ-pos order.strict-trans1[OF aTx
          mult-less-cancel-right-pos[THEN iffD2,OF x]] )
    finally
    show 1
        ≤ Liminf at-top (λt. ereal (?g t)) + ereal e
      by (smt (verit, best) ‹1 - ereal e = ereal (1 - e)› add-mono one-ereal-def
order.refl
          plus-ereal.simps(1))
  qed
  have le3:1 ≤ Liminf at-top (λt. ereal (?g t))
    by(rule ereal-le-epsilon2) (intro le1)
  moreover have le4:Limsup at-top (λt. ereal (?g t)) ≤ 1
    by(rule ereal-le-epsilon2) (intro le2)
  ultimately
  have Liminf at-top (λt. ereal (?g t)) = ereal 1
    and Limsup at-top (λt. ereal (?g t)) = ereal 1
    using Liminf-le-Limsup[of at-top]
    by (metis (no-types, lifting) one-ereal-def order.antisym order-trans
        trivial-limit-at-top-linorder)+
  hence ((λt. ereal (?g t)) ⟶ ereal 1) at-top
    by(intro Liminf-eq-Limsup) auto
  thus ?thesis
    using x by(simp add: rho')
 qed
qed

lemma
 assumes rv: regularly-varying U ϱ
   and rho:ϱ < - 1
 shows Karamata-theorem-half2:((λt. (∫ x∈{t<..}. U x ∂lborel) / (t * U t))
                              ⟶ - 1 / (ϱ + 1)) at-top
   and regularly-varying-tail-integrable:∃ T>0. set-integrable lborel {T<..} U
proof -
```

obtain i **where** $i\text{-}pos{:}0 < i$ **and** $i{:}\varrho + i < -1$
by (*metis add-minus-cancel rho dense less-add-same-cancel1*)
from *Potter-bounds1*[*OF rv - i-pos*,*of 2*]
obtain T **where** $T\text{-}pos{:}0 < T$
and $T{:}\bigwedge x\ y.\ T \le x \Longrightarrow x \le y \Longrightarrow U\ y\ /\ U\ x \le 2 * (y\ /\ x)\ powr\ (\varrho + i)$
by *auto*
have $*{:}(\lambda s.\ indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) \in borel\text{-}measurable\ lborel$
if $t{:}T \le t$ **for** t
proof −
have [*measurable*]:(∗) $t \in restrict\text{-}space\ borel\ \{0<..\} \to_M restrict\text{-}space\ borel\ \{0<..\}$
using t *T-pos*
by(*auto intro*!: *measurable-restrict-space3*)
note [*measurable*] = *regularly-varying-meas*[*OF rv*]
have [*measurable*]:$(\lambda s.\ U\ (t * s)) \in borel\text{-}measurable\ (restrict\text{-}space\ borel\ \{1<..\})$
by(*rule measurable-restrict-mono*[**where** $A{=}\{0<..\}$]) *measurable*
have $(\lambda s.\ indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) =$
$(\lambda s.\ if\ s \in \{1<..\}\ then\ U\ (t * s)\ /\ U\ t\ else\ 0)$
by *fastforce*
also have ... ∈ *borel-measurable lborel*
by(*intro measurable-restrict-space-iff*[*THEN iffD1*]) *auto*
finally show *?thesis* **.**
qed
have $**{:}\ integrable\ lborel\ (\lambda x.\ indicat\text{-}real\ \{1<..\}\ x * 2 * x\ powr\ (\varrho + i))$
using *set-integrable-mult-right-iff*[*THEN iffD2*,*OF - set-integrable-powr-inf*[*OF i*]]
by(*simp add*: *set-integrable-def mult.assoc*)
have $***{:}AE\ s\ in\ lborel.\ ((\lambda t.\ indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) \longrightarrow indicat\text{-}real\ \{1<..\}\ s * s\ powr\ \varrho)\ at\text{-}top$
by(*auto simp*: *indicator-def regularly-varying-lim*[*OF rv*])
have $****{:}AE\ s\ in\ lborel.$
$norm\ (indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) \le indicat\text{-}real\ \{1<..\}\ s * 2 * s\ powr\ (\varrho + i)$
if $t{:}T \le t$ **for** t
proof(*rule AE-I2*)
fix s
show $norm\ (indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) \le indicat\text{-}real\ \{1<..\}\ s * 2 * s\ powr\ (\varrho + i)$
proof(*cases* $1 < s$)
assume $s{:}1 < s$
then have $norm\ (indicat\text{-}real\ \{1<..\}\ s * U\ (t * s)\ /\ U\ t) = U\ (t * s)\ /\ U\ t$
using *regularly-varying-pos*[*OF rv*] t *T-pos*
by(*fastforce simp del*: *abs-divide*
intro!: *abs-of-pos regularly-varying-pos*[*OF rv*] *divide-pos-pos*)
also have ... ≤ $indicat\text{-}real\ \{1<..\}\ s * 2 * s\ powr\ (\varrho + i)$
using *T*[*OF t*,*of* $t * s$] s *T-pos* t **by** *auto*
finally show *?thesis* **.**

```
    qed simp
  qed
  have lim:((λt. (∫ x∈{1<..}. U (t * x) / U t ∂lborel)) ⟶ − 1 / (ϱ + 1))
at-top
    using integral-dominated-convergence-real[OF - * ** *** ****]
        set-integral-powr-inf[OF rho]
    by(auto simp: set-lebesgue-integral-def)
  have eq:∀ F t in at-top. (∫ x∈{t<..}. U x ∂lborel) / (t * U t)
                        = (∫ x∈{1<..}. U (t * x) / U t ∂lborel)
  proof(rule eventually-at-top-linorderI[where c=1])
    fix t :: real
    assume t:1 ≤ t
    have (∫ x∈{t<..}. U x ∂lborel) = t * (∫ s. indicat-real {t<..} (t * s) *R U (t
* s) ∂lborel)
      using lborel-integral-real-affine[of t λx. indicat-real {t<..} x * U x 0] t
      by(simp add: set-lebesgue-integral-def)
    also have ... = t * (∫ x∈{1<..}. U (t * x) ∂lborel)
      using t
      by(auto intro!: Bochner-Integration.integral-cong
        simp: set-lebesgue-integral-def mult.commute indicator-def pos-divide-less-eq)
    finally show (∫ x∈{t<..}. U x ∂lborel) / (t * U t) = (∫ x∈{1<..}. U (t * x)
/ U t ∂lborel)
      using t by simp
  qed
  show ((λt. (∫ x∈{t<..}. U x ∂lborel) / (t * U t)) ⟶ − 1 / (ϱ + 1)) at-top
    using lim eq tendsto-cong by fast
  have set-integrable lborel {1<..} (λx. U (t * x) / U t) if t:T ≤ t for t
    using ** ****[OF t] t
    by(auto intro!: Bochner-Integration.integrable-bound
        [where f=λs. indicat-real {1<..} s * 2 * s powr (ϱ + i),OF - * ]
        simp: set-integrable-def)
  from this[OF order.refl]
  have integrable lborel (λx. indicat-real {1<..} (x / T) * U x / U T)
    using lborel-integrable-real-affine
      [of λx. indicat-real {1<..} x * U (T * x) / U T 1 / T 0]
        T-pos by(auto simp: set-integrable-def)
  from this[THEN integrable-divide[where y=1 / U T]] regularly-varying-pos[OF
rv T-pos]
  have integrable lborel (λx. indicat-real {1<..} (x / T) * U x)
    by simp
  moreover have indicat-real {1<..} (x / T) * U x = indicat-real {T<..} x * U
x for x
    using T-pos regularly-varying-pos[OF rv] by(auto simp: indicator-def)
  ultimately have set-integrable lborel {T<..} U
    by (simp add: set-integrable-def cong: Bochner-Integration.integrable-cong)
  thus ∃ T>0. set-integrable lborel {T<..} U
    using T-pos by fast
qed
```

lemma *Karamata-theorem-half-uminus2-at-top*:
assumes *rv*: *regularly-varying U (− 1)*
and *a*:*0 ≤ a*
and *integ*:*set-integrable lborel {a<..} U*
shows *filterlim (λt. (∫ x∈{t<..}. U x ∂lborel) / (t * U t)) at-top at-top*
unfolding *filterlim-at-top*
proof *safe*
fix *Z* :: *real*
define *Z′* **where** *Z′ ≡ max (Z + 1) 2*
define *A* **where** *A ≡ exp Z′*
have *A1*:*1 ≤ A*
by(*simp add*: *A-def Z′-def*)
obtain *T1* **where** *T1*:⋀*x t. T1 ≤ t ⟹ 1 ≤ x ⟹ |U (t * x) / U t − x powr (− 1)| < 1 / A*
using *uniform-limitD*[*OF regularly-varying-neg-UCT*[*OF rv*,*of 1*],*of 1 / A*] *A1*
by(*fastforce simp*: *eventually-at-top-linorder dist-real-def*)
define *T* **where** *T ≡ max (max 1 a) T1*
have *Tle*:⋀*t x. T ≤ t ⟹ 1 ≤ x ⟹ 1 / x − 1 / A ≤ U (t * x) / U t*
using *T1 T-def* **by** *fastforce*
show ∀ *F t in at-top. Z ≤ (∫ x∈{t<..}. U x ∂lborel) / (t * U t)*
proof(*rule eventually-at-top-linorderI*[**where** *c*=*T*])
fix *t*
assume *t*:*T ≤ t*
hence *t-pos*:*0 < t*
by(*simp add*: *T-def*)

have *integ1*[*simp*]: *set-integrable lborel {t<..} U*
using *t* **by**(*auto intro*!: *set-integrable-subset*[*OF integ*] *simp*: *T-def*)
have *integ2*[*simp*]:*set-integrable lborel {1<..} (λx. U (t * x) / U t)*
proof −
have *:*indicat-real {t<..} (t * x) = indicat-real {1<..} x* **for** *x*
using *t-pos* **by**(*auto simp*: *indicator-def divide-less-eq mult.commute*)
have *integrable lborel (λx. indicat-real {t<..} (t * x) * U (t * x))*
using *lborel-integrable-real-affine*[*of λx. indicat-real {t<..} x * U x t 0*]
integ1 t-pos **unfolding** *set-integrable-def*
by *simp*
thus *?thesis*
unfolding * **by**(*simp add*: *set-integrable-def*)
qed
have *integ3*[*simp*]:*set-integrable lborel {1<..A} (λx. U (t * x) / U t)*
by(*auto intro*!: *set-integrable-subset*[*OF integ2*])
note [*simp*] = *set-integral-diff*[*OF set-integrable-inverse′*[*of 1*]
set-integrable-const[*of lborel {1<..A} 1 / A*]]

have *Z ≤ ln A − 1*
by(*simp add*: *A-def Z′-def*)
also have *... ≤ ln A − (1 − 1 / A)*
using *A1* **by** *simp*
also have *... = (∫ x∈{1<..A}. 1 / x ∂lborel) − (1 − 1 / A)*

```
    using A1 by(subst set-integral-inverse) auto
  also have ... = (∫ x∈{1<..A}. 1 / x ∂lborel) − (∫ x∈{1<..A}. 1 / A ∂lborel)
    using A1 by(subst set-integral-const) (auto simp: diff-divide-distrib)
  also have ... = (∫ x∈{1<..A}. 1 / x − 1 / A ∂lborel)
    using A1 by simp
  also have ... ≤ (∫ x∈{1<..A}. U (t * x) / U t ∂lborel)
    using A1 by(intro set-integral-mono Tle t) auto
  also have ... ≤ (∫ x∈{1<..}. U (t * x) / U t ∂lborel)
    using t-pos
    by(intro set-integral-subset)
      (auto intro!: divide-pos-pos less-imp-le[of 0] regularly-varying-pos[OF rv])
  also have ... = (∫ x∈{t<..}. U x ∂lborel) / (t * U t)
  proof −
    have (∫ x∈{t<..}. U x ∂lborel)
          = t * (∫ s. indicat-real {t<..} (t * s) *R U (t * s) ∂lborel)
      using lborel-integral-real-affine[of t λx. indicat-real {t<..} x * U x 0] t-pos
      by(simp add: set-lebesgue-integral-def)
    also have ... = t * (∫ x∈{1<..}. U (t * x) ∂lborel)
      using t-pos
      by(auto intro!: Bochner-Integration.integral-cong
       simp: set-lebesgue-integral-def mult.commute indicator-def pos-divide-less-eq)
    finally show ?thesis
      using t-pos by simp
  qed
  finally show Z ≤ (∫ x∈{t<..}. U x ∂lborel) / (t * U t) .
 qed
qed

corollary Karamata-theorem-half-uminus2:
 assumes rv: regularly-varying U (− 1)
   and a:0 ≤ a
   and integ:set-integrable lborel {a<..} U
 shows ((λt. (t * U t) / (∫ x∈{t<..}. U x ∂lborel)) ⟶ 0) at-top
 using real-tendsto-divide-at-top[OF tendsto-const[of 1]
    Karamata-theorem-half-uminus2-at-top[OF assms]]
 by simp

lemma regularly-varying-integral-tail-limit:
 assumes rv: regularly-varying U ϱ
   and rho:ϱ < − 1
   and x: 0 < x
 shows ((λt. (∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel))
       ⟶ x powr (ϱ + 1)) at-top
proof −
 have (λt. (∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel)) ∼[at-top]
      (λt. (t * x * U (t * x)) / (t * U t))
   using asymp-equiv-divide[OF asymp-equivI'-const[OF
        filterlim-compose[OF Karamata-theorem-half2[OF rv rho]
          filterlim-tendsto-pos-mult-at-top[OF tendsto-const x filterlim-ident]]]
```

```
      asymp-equivI'-const[OF Karamata-theorem-half2[OF rv rho]]] rho
    by(simp add: mult.commute)
  moreover have (λt. (t * x * U (t * x)) / (t * U t))
              ~[at-top] (λt. (x * (U (t * x)) / U t))
   using x regularly-varying-pos[OF rv] regularly-varying-pos[OF rv mult-pos-pos[OF
- x]]
    by(intro asymp-equivI' tendstoI eventually-at-top-linorderI[where c=1])
      (smt (verit, ccfv-threshold) dist-real-def divide-divide-eq-right
        divide-divide-times-eq divide-eq-1-iff nonzero-mult-div-cancel-left)
  ultimately have (λt. (∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel))
                ~[at-top] (λt. (x * (U (t * x)) / U t))
    using asymp-equiv-trans by blast
  moreover have ((λt. (x * (U (t * x)) / U t)) ⟶ x powr (ϱ + 1)) at-top
    using tendsto-mult[OF tendsto-const[of x] regularly-varying-lim[OF rv x]]
      powr-mult-base' x
    by (simp add: add.commute powr-mult-base)
  ultimately show ?thesis
    using tendsto-asymp-equiv-cong by blast
qed

lemma regularly-varying-integral-tail-limit2:
  assumes rv: regularly-varying U (− 1)
    and integ:set-integrable lborel {a<..} U
    and x:0 < x
  shows ((λt. (∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel)) ⟶ 1)
at-top
  unfolding tendsto-iff eventually-at-top-linorder dist-real-def
proof safe
  define a' where a' ≡ max 1 a
  have a':0 < a'
    by(simp add: a'-def)
  have integ':⋀t. a' ≤ t ⟹ set-integrable lborel {t<..} U
    by(auto intro!: set-integrable-subset[OF integ] simp: a'-def)
  fix e :: real
  assume e:0 < e
  define e' where e' ≡ e / (2 * x)
  have e':0 < e'
    using x e by(simp add: e'-def)
  obtain T' where T':⋀t. T' ≤ t ⟹ |U (t * x) / U t − 1 / x| < e'
    using tendstoD[OF regularly-varying-lim[OF rv x] e'] x
    by(auto simp: eventually-at-top-linorder dist-real-def)
  define T where T ≡ max a' (max (a' / x) T')
  have T-pos:0 < T and aT2:a' ≤ T and TT':T' ≤ T and a' / x ≤ T
    using a' x by(auto simp: T-def)
  hence aTx:a' ≤ T * x and aT1:a' ≤ T
    using x a' pos-divide-le-eq by auto
  have T1:T ≤ t ⟹ U (t * x) ≤ U t * (1 / x + e') for t
    using T'[OF order.trans[OF TT'],of t] T-pos regularly-varying-pos[OF rv,of t]
   by(auto simp: abs-less-iff diff-less-eq add.commute mult.commute pos-divide-less-eq)
```

have *T2*:$T \leq t \Longrightarrow U\ t * (1\ /\ x - e') \leq U\ (t * x)$ **for** t
using *T'*[*OF order.trans*[*OF TT'*],*of t*] *T-pos regularly-varying-pos*[*OF rv*,*of t*]
by (*metis* (*no-types*, *lifting*) *linorder-not-le abs-diff-less-iff dual-order.trans mult.commute mult-imp-div-pos-less not-less-iff-gr-or-eq*)
have *integT*:*set-integrable lborel* {*T*<..} (λ*t*. *U* (*t* * *x*))
proof −
have *1*:*indicat-real* {*T* * *x*<..} (*s* * *x*) * *U* (*s* * *x*)
= *indicat-real* {*T*<..} *s* * *U* (*s* * *x*) **for** *s*
using *x* **by**(*auto simp*: *indicator-def*)
have *integrable lborel* (λ*t*. *indicat-real* {*T* * *x*<..} *t* * *U t*)
using *integ'*[*OF aTx*] **unfolding** *set-integrable-def* **by** *simp*
from *lborel-integrable-real-affine*[**where** *c*=*x* **and** *t*=*0*,*OF this*] *x*
have *integrable lborel* (λ*s*. *indicat-real* {*T* * *x*<..} (*s* * *x*) * *U* (*s* * *x*))
by(*simp add*: *mult.commute*)
thus *?thesis*
unfolding *1* **by**(*simp add*: *set-integrable-def*)
qed
show ∃ *T*. ∀ *t*≥*T*. |(∫ *s*∈{*t* * *x*<..}. *U s* ∂*lborel*) / (∫ *s*∈{*t*<..}. *U s* ∂*lborel*) − *1*| < *e*
proof(*safe intro*!: *exI*[**where** *x*=*T*])
fix *t*
assume *t*:*T* ≤ *t*
then have *t-pos*:*0* < *t*
using *T-pos* **by** *simp*
have *integ-pos*[*arith*]:*0* < (∫ *s*∈{*t*<..}. *U s* ∂*lborel*)
using *t-pos*
by(*intro set-integral-zero-less integ' order.trans*[*OF aT1 t*] *emeasure-mono order.strict-trans2*[*of 0 emeasure lborel* {*t*<..*t*+*1*} *emeasure lborel* {*t*<..}])
(*auto intro*!: *regularly-varying-pos*[*OF rv*])
have *eq*:(∫ *s*∈{*t* * *x*<..}. *U s* ∂*lborel*) = *x* * (∫ *s*∈{*t*<..}. *U* (*s* * *x*) ∂*lborel*)
proof −
have (∫ *s*∈{*t* * *x*<..}. *U s* ∂*lborel*)
= *x* * (∫ *s*. *indicat-real* {*t* * *x*<..} (*s* * *x*) * *U* (*s* * *x*) ∂*lborel*)
using *lborel-integral-real-affine*[*of x* λ*s*. *indicat-real* {*t***x*<..} *s* * *U s 0*] *x*
by(*simp add*: *set-lebesgue-integral-def mult.commute*)
also have ... = *x* * (∫ *s*∈{*t*<..}. *U* (*s* * *x*) ∂*lborel*)
using *x t-pos regularly-varying-pos*[*OF rv*]
by(*auto simp*: *set-lebesgue-integral-def indicator-def intro*!: *Bochner-Integration.integral-cong*)
finally show *?thesis* **.**
qed
have (∫ *s*∈{*t* * *x*<..}. *U s* ∂*lborel*) / (∫ *s*∈{*t*<..}. *U s* ∂*lborel*)
= *x* * (∫ *s*∈{*t*<..}. *U* (*s* * *x*) ∂*lborel*) / (∫ *s*∈{*t*<..}. *U s* ∂*lborel*)
by(*simp add*: *eq*)
also have ... ≤ *x* * (∫ *s*∈{*t*<..}. *U s* * (*1* / *x* + *e'*) ∂*lborel*) / (∫ *s*∈{*t*<..}. *U s* ∂*lborel*)
using *x t aT1*
by(*auto simp del*: *set-integral-mult-left intro*!: *divide-right-mono set-integral-mono set-integrable-subset*[*OF integT*]

```
        set-integrable-mult-left[OF integ'] T1)
    also have ... < 1 + e
      using x e by(simp add: distrib-left[of x] e'-def)
    finally have *:(∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel)
        < 1 + e .
    have 1 - e < x * (∫ s∈{t<..}. U s * (1 / x - e') ∂lborel) / (∫ s∈{t<..}. U s
∂lborel)
      using x e by(simp add: right-diff-distrib'[of x] e'-def)
    also have ... ≤ x * (∫ s∈{t<..}. U (s * x) ∂lborel) / (∫ s∈{t<..}. U s ∂lborel)
      using x t aT1
      by(auto simp del: set-integral-mult-left
        intro!: divide-right-mono set-integral-mono set-integrable-subset[OF integT]
          set-integrable-mult-left[OF integ'] T2)
    also have ... = (∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel)
      by(simp add: eq)
    finally show |(∫ s∈{t * x<..}. U s ∂lborel) / (∫ s∈{t<..}. U s ∂lborel) - 1| <
e
      using * by linarith
  qed
qed

lemma regularly-varying-integral-measurable:
  assumes rv: regularly-varying U ϱ
    and a:0 ≤ a
  shows (λx. (∫ s∈{a<..x}. U s ∂lborel)) ∈ borel-measurable borel
proof -
  define U' where U' ≡ λx. if x ≤ a then 0 else U x
  have [measurable]:U' ∈ borel-measurable borel
    unfolding U'-def using regularly-varying-meas[OF rv] a
    by(subst measurable-If-restrict-space-iff)
      (auto intro!: measurable-restrict-mono[where A={0<..} and B={x. ¬ x ≤
a}])
  have *:(λx. (∫ s∈{a<..x}. U' s ∂lborel)) = (λx. (∫ s∈{a<..x}. U s ∂lborel))
    by (auto intro!: set-lebesgue-integral-cong simp: U'-def)
  have (λx. (∫ s∈{a<..x}. U' s ∂lborel)) ∈ borel-measurable borel
    unfolding set-lebesgue-integral-def
    by measurable auto
  thus ?thesis
    unfolding * by simp
qed

lemma regularly-varying-tail-integral-measurable:
  assumes rv: regularly-varying U ϱ
  shows (λx. (∫ s∈{x<..}. U s ∂lborel)) ∈ borel-measurable (restrict-space borel
{0<..})
proof -
  define U' where U' ≡ λx. if 0 < x then U x else 0
  have [measurable]:U' ∈ borel-measurable borel
    unfolding U'-def using regularly-varying-meas[OF rv]
```

```
    by(subst measurable-If-restrict-space-iff)
     (auto intro!: measurable-restrict-mono[where A={0<..} and B={x. 0 < x}])
  have (λx. (∫ s∈{x<..}. U' s ∂lborel)) ∈ borel-measurable borel
    unfolding set-lebesgue-integral-def
    by measurable auto
  hence *:(λx. (∫ s∈{x<..}. U' s ∂lborel)) ∈ borel-measurable (restrict-space borel
{0<..})
    using measurable-restrict-space1 by blast
  have ⋀x. x ∈ space (restrict-space borel {0<..})
          ⟹ (∫ s∈{x<..}. U' s ∂lborel) = (∫ s∈{x<..}. U s ∂lborel)
    by(auto intro!: set-lebesgue-integral-cong simp: U'-def)
  from measurable-cong[OF this] *
  show ?thesis
    by fast
qed

lemma regularly-varying-integral:
  assumes rv:regularly-varying U ϱ
    and rho:- 1 ≤ ϱ
    and integrable:⋀x. set-integrable lborel {0<..x} U
  shows regularly-varying (λx. (∫ t∈{0<..x}. U t ∂lborel)) (ϱ + 1)
  using regularly-varying-pos[OF rv]
  by(intro regularly-varyingI[OF - - regularly-varying-integral-limit] rho
      measurable-restrict-space1[OF regularly-varying-integral-measurable[OF rv]]
      rv set-integral-zero-less integrable set-integrable-subset[OF integrable])
     fastforce+

lemma regularly-varying-tail-integral:
  assumes rv:regularly-varying U ϱ
    and rho:ϱ ≤ - 1
    and integrable:set-integrable lborel {0<..} U
  shows regularly-varying (λx. (∫ t∈{x<..}. U t ∂lborel)) (ϱ + 1)
proof -
  have 1:0 < x ⟹ 0 < (∫ t∈{x<..}. U t ∂lborel) for x
    by(intro set-integral-zero-less set-integrable-subset[OF integrable]
        order.strict-trans2[of 0 emeasure lborel {x<..x+1}
                            emeasure lborel {x<..}] emeasure-mono)
      (auto intro!: regularly-varying-pos[OF rv])
  consider ϱ < - 1 | ϱ = - 1
    using rho by argo
  thus ?thesis
  proof cases
    assume rho:ϱ = - 1
    show regularly-varying (λx. set-lebesgue-integral lborel {x<..} U) (ϱ + 1)
      using rv 1
      by(auto intro!: slowly-varyingI
          [OF - - regularly-varying-integral-tail-limit2[OF - integrable]]
        regularly-varying-tail-integral-measurable[OF rv] rv simp: rho)
  qed(intro regularly-varyingI[OF - - regularly-varying-integral-tail-limit]
```

regularly-varying-tail-integral-measurable[*OF rv*] *rv 1*)
qed

2.5.2 Converse Half

lemma
assumes *U-pos*:⋀*r*. *0* < *r* ⟹ *0* < *U r*
and *U-meas*:*U* ∈ *borel-measurable* (*restrict-space borel* {*0*<..}) **and** *rho*:− *1* < *ϱ* **and** *a*[*arith*]:*0* ≤ *a*
and *lim*: ((λ*t*. (*t* ∗ *U t*) / (∫ *x*∈{*a*<..*t*}. *U x* ∂*lborel*)) ⟶ *ϱ* + *1*) *at-top*
shows *Karamata-theorem-converse1*:*regularly-varying U ϱ*
and *Karamata-theorem-converse1-integrable*:
⋀*c d*. *a* ≤ *c* ⟹ *set-integrable lborel* {*c*..*d*} *U*
proof −
show *integrable*:*set-integrable lborel* {*c*..*d*} *U* **if** *le*:*a* ≤ *c* **for** *c d*
proof −
obtain *T* **where** *T*:⋀*t*. *T* ≤ *t* ⟹ *0* < *t* ∗ *U t* / (∫ *x*∈{*a*<..*t*}. *U x* ∂*lborel*)
using *order-tendstoD*[**where** *a*=*0*,*OF lim*] *rho*
by(*auto simp*: *eventually-at-top-linorder*)
have [*arith*]:*a* < *T*
by(*rule ccontr*) (*use T*[*of T*] **in** *simp*)
have *T* ≤ *t* ⟹ *0* < (∫ *x*∈{*a*<..*t*}. *U x* ∂*lborel*) **for** *t*
using *T*[*of t*] *U-pos*[*of t*]
by(*auto simp*: *zero-less-divide-iff mult-less-0-iff*)
hence *T* ≤ *t* ⟹ *set-integrable lborel* {*a*<..*t*} *U* **for** *t*
using *not-set-integrable-integral-eq* **by** *fastforce*
hence *T* ≤ *t* ⟹ *set-integrable lborel* {*a*..*t*} *U* **for** *t*
using *set-integrable-Ioc-eq-Icc* **by** *blast*
thus *set-integrable lborel* {*c*..*d*} *U*
by (*metis* (*mono-tags*, *opaque-lifting*) *atLeastAtMost-borel atLeastatMost-subset-iff*
le le-cases set-integrable-subset sets-lborel)
qed
let *?F* = λ*x*. (∫ *t*∈{*a*<..*x*}. *U t* ∂*lborel*)
define *g* **where** *g* ≡ λ*t*. (*t* ∗ *U t*) / *?F t*
have *F-pos*[*simp*]:*a* < *x* ⟹ *0* < *?F x* **for** *x*
using *U-pos*
by(*intro set-integral-zero-less set-integrable-subset*[*OF integrable*[*of a x*]]) *auto*
obtain *b* **where** *b*[*arith*]:*a* < *b*
using *linordered-field-no-ub* **by** *blast*
have *lnF-eq*: *ln* (*?F b*) + (∫ *t*∈{*b*<..*x*}. *g t* / *t* ∂*lborel*) = *ln* (*?F x*)
if *x*[*arith*]:*b* ≤ *x* **for** *x*
proof −
have (∫ *t*∈{*b*<..*x*}. *g t* / *t* ∂*lborel*) = (∫ *t*∈{*b*<..*x*}. *1* / *?F t* ∗ *U t* ∂*lborel*)
using *U-pos*
by(*intro set-integral-cong*) (*auto simp*: *g-def*)
also have ... = (∫ *t*∈{*?F b*<..*?F x*}. *1* / *t* ∂*lborel*)
by(*intro set-integral-compose-integral-Ioc U-pos integrable measurable-restrict-space1*)
auto
also have ... = *ln* (*?F x*) − *ln* (*?F b*)

```
    by(subst set-integral-inverse)
      (auto intro!: set-integral-subset set-integrable-subset
        [OF integrable[of a x]] less-imp-le[OF U-pos])
  finally show ?thesis
    by simp
qed
have g-integ:set-integrable lborel {b<..x} (λt. g t / t)
  if x[arith]:b ≤ x for x
proof −
  have 1:set-integrable lborel {b<..x} (λt. 1 / ?F t * U t)
  by(intro set-integrable-compose-integral-Ioc U-pos integrable set-integrable-inverse′)
      auto
  have 2:⋀t. t ∈ {b<..x} ⟹ 1 / ?F t * U t = g t / t
    by(simp add: g-def)
  show ?thesis
    using set-integrable-cong[THEN iffD1,OF refl refl 2 1]
    by fastforce
qed
define r where r ≡ λx. g x − 1
define c where c ≡ λx. g x * ?F b / b
define C where C ≡ (ϱ + 1) * ?F b / b
have 1:∀ x≥b. U x = c x * exp (∫ t∈{b<..x}. r t / t ∂lborel)
proof safe
  fix x
  assume x[arith]:b ≤ x
  have [arith]:0 < ?F x
    by simp
  have U x = g x * ?F x / x
    by(simp add: g-def)
  also have ... = g x * exp (− ln x) * exp (ln (?F x))
    by(simp add: ln-inverse[symmetric] divide-real-def)
  also have ... = g x * exp (ln (1 / b) − (∫ t∈{b<..x}. 1 / t ∂lborel))
                    * exp (ln (?F b) + (∫ t∈{b<..x}. g t / t ∂lborel))
    by(simp add: set-integral-inverse lnF-eq ln-divide-pos)
  also have ... = c x * exp ((∫ t∈{b<..x}. g t / t ∂lborel) − (∫ t∈{b<..x}. 1 / t
∂lborel))
    by(simp add: c-def exp-diff exp-add)
  also have ... = c x * exp (∫ t∈{b<..x}. r t / t ∂lborel)
    using g-integ[OF x] set-integrable-inverse′
    by(auto simp: r-def diff-divide-distrib)
  finally show U x = c x * exp (∫ t∈{b<..x}. r t / t ∂lborel) .
qed
have 2:∀ x≥b. set-integrable lborel {b<..x} (λt. r t / t)
  unfolding r-def diff-divide-distrib
  by(intro allI impI set-integral-diff set-integrable-inverse′ g-integ) auto
have 3:(r ⟶ ϱ) at-top
  using tendsto-diff[OF lim tendsto-const[where k=1]]
  by(simp add: r-def g-def)
have 4:(c ⟶ C) at-top
```

```
    unfolding c-def C-def times-divide-eq-right[symmetric]
    by(intro tendsto-mult) (auto simp: g-def lim)
  have 5:0 < C
    using rho by(simp add: C-def)
  show regularly-varying U ϱ
    using 1 2 3 4 5
    by(subst Karamata-representation-iff[OF U-meas U-pos])
      (auto intro!:  exI[where x=b,OF conjI
          [OF - exI[where x=c,OF exI[where x=r,OF exI[where x=C]]]]]
        simp del: ex-simps)
qed

lemma
  assumes U-pos:⋀r. 0 < r ⟹ 0 < U r
    and U-meas:U ∈ borel-measurable (restrict-space borel {0<..})
    and rho:- 1 > ϱ
    and lim: ((λt. t * U t / (∫ x∈{t<..}. U x ∂lborel)) ⟶ - (ϱ + 1)) at-top
  shows Karamata-theorem-converse2:regularly-varying U ϱ
    and Karamata-theorem-converse2-itegrable: ∃a>0. set-integrable lborel {a..} U
proof -
  show ∃a>0. set-integrable lborel {a..} U
  proof -
   obtain T' where T':⋀t. T' ≤ t ⟹ 0 < (t * U t) / (∫ x∈{t<..}. U x ∂lborel)
     using order-tendstoD[where a=0,OF lim] rho
     by(auto simp: eventually-at-top-linorder)
    define a where a ≡ max T' 1
    have [arith]:0 < a
     by(simp add: a-def)
    have 0 < (∫ x∈{a<..}. U x ∂lborel)
     using T'[of a] U-pos[of a]
     by(auto simp: a-def zero-less-divide-iff mult-less-0-iff)
    thus ?thesis
     using not-set-integrable-integral-eq
    by(fastforce intro!: exI[where x=a] simp: set-integrable-atLeast-eq[symmetric])
  qed
  then obtain a where a-pos[arith]:0 < a
    and integrable:set-integrable lborel {a..} U
    by blast
  let ?F = λx. (∫ t∈{x<..}. U t ∂lborel)
  define g where g ≡ λx. - x * U x / ?F x
  have F-pos[simp]:a ≤ x ⟹ 0 < ?F x for x
    using U-pos
    by(intro set-integral-zero-less set-integrable-subset[OF integrable]
        order.strict-trans2[of 0 emeasure lborel {x<..x+1}
                               emeasure lborel {x<..},OF - emeasure-mono])
      auto
  have g-integ[simp]:set-integrable lborel {a<..x} (λt. g t / t)
    if x[arith]:a ≤ x for x
  proof -
```

```
  have 1:set-integrable lborel {a<..x} (λt. − 1 / ?F t ∗ U t)
    by(intro set-integrable-compose-integral-tail[where a=a]
        U-pos integrable set-integrable-inverse′)
      auto
  have 2:⋀t. t ∈ {a<..x} ⟹ − 1 / ?F t ∗ U t = g t / t
    by(simp add: g-def)
  show ?thesis
    using set-integrable-cong[THEN iffD1,OF refl refl 2 1]
    by fastforce
qed

have lnF-eq: ln (?F a) + (∫ t∈{a<..x}. g t / t ∂lborel) = ln (?F x)
  if x[arith]:a ≤ x for x
proof −
  have (∫ t∈{a<..x}. − g t / t ∂lborel) = (∫ t∈{a<..x}. 1 / ?F t ∗ U t ∂lborel)
    using U-pos
    by(intro set-integral-cong) (auto simp: g-def)
  also have ... = (∫ t∈{?F x<..?F a}. 1 / t ∂lborel)
    by(intro set-integral-compose-integral-tail[where a=a]
        integrable measurable-restrict-space1 U-pos)
      auto
  also have ... = ln (?F a) − ln (?F x)
    by(subst set-integral-inverse)
      (auto intro!: set-integral-subset set-integrable-subset[OF integrable]
        less-imp-le[OF U-pos])
  finally show ?thesis
    by (simp add: set-integral-uminus)
qed
define r where r ≡ λx. g x − 1
define c where c ≡ λx. − g x ∗ ?F a / a
define C where C ≡ − (ϱ + 1) ∗ ?F a / a
have 1:∀ x≥a. U x = c x ∗ exp (∫ t∈{a<..x}. r t / t ∂lborel)
proof safe
  fix x
  assume x[arith]:a ≤ x
  have [arith]:0 < ?F x
    by simp
  have U x = − g x ∗ ?F x / x
    by(simp add: g-def)
  also have ... = − g x ∗ exp (− ln x) ∗ exp (ln (?F x))
    by(simp add: ln-inverse[symmetric] divide-real-def)
  thm ln-inverse[symmetric]
  also have ... = − g x ∗ exp (ln (1 / a) − (∫ t∈{a<..x}. 1 / t ∂lborel))
                        ∗ exp (ln (?F a) + (∫ t∈{a<..x}. g t / t ∂lborel))
    by(simp add: set-integral-inverse lnF-eq ln-divide-pos)
  also have ... = c x ∗ exp ((∫ t∈{a<..x}. g t / t ∂lborel) − (∫ t∈{a<..x}. 1 / t
∂lborel))
    by(simp add: c-def exp-diff exp-add)
  also have ... = c x ∗ exp (∫ t∈{a<..x}. r t / t ∂lborel)
```

```
    using g-integ[OF x] set-integrable-inverse'
    by(auto simp: r-def diff-divide-distrib)
  finally show U x = c x * exp (∫ t∈{a<..x}. r t / t ∂lborel) .
 qed
 have 2:∀ x≥a. set-integrable lborel {a<..x} (λt. r t / t)
  unfolding r-def diff-divide-distrib
  by(intro allI impI set-integral-diff set-integrable-inverse' g-integ) auto
 have 3:(r ⟶ ϱ) at-top
  using tendsto-diff[OF lim tendsto-const[where k=- 1]]
  by(simp add: r-def g-def tendsto-minus-cancel-left)
 have 4:(c ⟶ C) at-top
  unfolding c-def C-def times-divide-eq-right[symmetric] using lim
  by(intro tendsto-mult)
    (auto simp: g-def )
 have 5:0 < C
  using rho by(simp add: C-def)
 show regularly-varying U ϱ
  using 1 2 3 4 5
  by(subst Karamata-representation-iff[OF U-meas U-pos])
    (auto intro!:  exI[where x=a,OF conjI[OF - exI[where x=c,
          OF exI[where x=r,OF exI[where x=C]]]]]
      simp del: ex-simps)
qed
```

2.6 Properties Related to Derivative

Proposition 2.5 (a) [3].

```
lemma deriv-ratio-regularly-varying-uniform:
 fixes U U' :: real ⇒ real and ϱ a b :: real
 assumes t0: 0 < t0
  and deriv: ⋀t. t0 < t ⟹ (U has-real-derivative U' t) (at t)
  and U'cont: continuous-on {t0<..} U'
  and Upos: ⋀t. t0 < t ⟹ 0 < U t
  and lim: ((λt. t * U' t / U t) ⟶ ϱ) at-top
  and a: 0 < a
 shows uniform-limit {a..b} (λt x. U (t * x) / U t) (λx. x powr ϱ) at-top
proof −
 have Ucont:continuous-on {t0<..} U
  by(auto intro!: has-real-derivative-imp-continuous-on[where f'=U'] deriv)
 define V where V ≡ λx. if x ∈ {t0<..} then U x else 1
 have V-eq-U:⋀x. t0 < x ⟹ V x = U x
  by(simp add: V-def)
 have cong:∀ F t in at-top. ∀ x∈{a..b}. V (t * x) / V t = U (t * x) / U t
 proof(safe intro!: eventually-at-top-linorderI[where c=max (t0 + 1) ((t0 + 1)
/ a)])
  fix t x
  assume max (t0 + 1) ((t0 + 1) / a) ≤ t x ∈ {a..b}
  then have t0 < t t0 < t * x
   using a t0 by(auto intro!: order.strict-trans2[of t0 t0 + 1 t * x]
```

```
        order.trans[of t0 + 1 t * a t * x] simp: pos-divide-le-eq )
    thus V (t * x) / V t = U (t * x) / U t
      by(simp add: V-eq-U)
  qed
  have rv:regularly-varying V ϱ
  proof(subst Karamata-representation-iff)
    show V ∈ borel-measurable (restrict-space borel {0<..})
      unfolding V-def
      by(intro measurable-restrict-space1 borel-measurable-continuous-on-if Ucont)
auto
    show V-pos:⋀x. 0 < V x
      by(auto simp: V-def Upos)
    define A where A ≡ t0 + 1
    have A-pos[arith]:0 < A and t0A[arith]:t0 < A
      using t0 by(simp-all add: A-def)
    define r where r ≡ λx. x * U' x / U x
    have r-lim:(r ⟶ ϱ) at-top
      unfolding r-def by fact
    have deriv':A ≤ x ⟹ ((λx. ln (U x)) has-real-derivative r x / x) (at x) for x
      by(intro DERIV-cong[where X=1 / U x * U' x and Y=r x / x] deriva-
tive-intros deriv)
        (auto simp: r-def Upos)
    have rcont:continuous-on {t0<..} (λx. r x / x)
      unfolding r-def using Upos A-pos t0A t0
      by(intro continuous-intros continuous-on-subset[OF U'cont]
          continuous-on-subset[OF Ucont] ballI dual-order.strict-implies-not-eq) auto
    have (∫ t∈{A<..x}. r t / t ∂lborel) = ln (V x) − ln (V A) if x:A ≤ x for x
    proof −
     have (∫ t∈{A<..x}. r t / t ∂lborel) = (∫ t. r t / t * indicator {A..x} t ∂lborel)
      unfolding LBINT-Ioc-eq-Icc by(simp add: set-lebesgue-integral-def mult.commute)
      also have ... = ln (U x) − ln (U A)
        using x
        by(intro integral-FTC-Icc-real deriv' continuous-on-interior[OF rcont])
          (auto simp: interior-open)
      finally show ?thesis
        using x by(simp add: V-eq-U)
    qed
    hence eq:A ≤ x ⟹ V x = V A * exp (∫ t∈{A<..x}. r t / t ∂lborel) for x
      using V-pos[of A] by(auto simp add: exp-diff V-pos[of x])
    have integ:set-integrable lborel {A<..x} (λt. r t / t) if A ≤ x for x
      by(auto intro!: set-integrable-subset[OF borel-integrable-atLeastAtMost'[of A
x]]
          continuous-on-subset[OF rcont])
    show ∃ A>0. ∃ c r C.
            (∀ x ≥ A. V x = c x * exp (∫ t∈{A<..x}. r t / t ∂lborel))
          ∧ (∀ x ≥ A. set-integrable lborel {A<..x} (λt. r t / t))
          ∧ (r ⟶ ϱ) at-top
          ∧ (c ⟶ C) at-top
          ∧ 0 < C
```

```
    using eq integ r-lim V-pos[of A] tendsto-const[of V A] A-pos
    by fast
 qed
 hence uniform-limit {a..b} (λt x. V (t * x) / V t) (λx. x powr ϱ) at-top
  by(simp add: a regularly-varying-UCT-interval)
 with cong show ?thesis
  using uniform-limit-cong by fast
qed

corollary deriv-ratio-regularly-varying:
 fixes U U' :: real ⇒ real and γ x :: real
 assumes 0 < t0
  and ⋀t. t0 < t ⟹ (U has-real-derivative U' t) (at t)
  and continuous-on {t0<..} U'
  and ⋀t. t0 < t ⟹ 0 < U t
  and ((λt. t * U' t / U t) ⟶ γ) at-top
  and 0 < x
 shows ((λt. U (t * x) / U t) ⟶ x powr γ) at-top
 by(auto intro!: tendsto-uniform-limitI[OF
      deriv-ratio-regularly-varying-uniform[where a=x and b=x,OF assms]])

lemma deriv-ratio-regularly-varying-uminus:
 fixes U u :: real ⇒ real and γ L x :: real
 assumes t0: 0 < t0
  and Uderiv: ⋀t. t0 < t ⟹ (U has-real-derivative u t) (at t)
  and ucont: continuous-on {t0<..} u
  and Ult: ⋀t. t0 < t ⟹ U t < L
  and lim: ((λt. t * u t / (L − U t)) ⟶ − γ) at-top
  and x: 0 < x
 shows ((λt. (L − U (t * x)) / (L − U t)) ⟶ x powr γ) at-top
 using DERIV-diff[OF DERIV-const[of L] Uderiv] ucont tendsto-minus[OF lim]
 by(auto intro!: deriv-ratio-regularly-varying[where U'=λx. − u x,OF t0] continuous-minus
                 Ult x simp: continuous-on-eq-continuous-within)
```

Corollary 1.1.10 [2].

```
lemma deriv-diff-ratio-regularly-varying:
 fixes U u u' :: real ⇒ real and γ x :: real
 assumes t0: 0 < t0
  and U-deriv: ⋀t. t0 < t ⟹ (U has-real-derivative u t) (at t)
  and u-deriv: ⋀t. t0 < t ⟹ (u has-real-derivative u' t) (at t)
  and u'-cont: continuous-on {t0<..} u'
  and u-pos: ⋀t. t0 < t ⟹ 0 < u t
  and lim: ((λt. t * u' t / u t) ⟶ γ − 1) at-top
  and x: 0 < x
 shows ((λt. (U (t * x) − U t) / (t * u t))
        ⟶ (if γ = 0 then ln x else (x powr γ − 1) / γ)) at-top
proof −
 have u-cont:continuous-on {t0<..} u
```

```
    by(intro has-real-derivative-imp-continuous-on[where f'=u'] u-deriv) auto
  define v where v ≡ λx. (if x ∈ {t0<..} then u x else 1)
  have v-eq:⋀t. t0 < t ⟹ v t = u t
    by(simp add: v-def)
  have v-pos:⋀t. 0 < v t
    using u-pos by(simp add: v-def)
  have [measurable]:v ∈ borel-measurable borel
    unfolding v-def
     by(intro measurable-restrict-space1 borel-measurable-continuous-on-if u-cont)
auto
  have rv:regularly-varying v (γ − 1)
  proof(safe intro!: regularly-varyingI-total)
    fix y::real
    assume 0 < y
    then show ((λt. v (t ∗ y) / v t) ⟶ y powr (γ − 1)) at-top
      by(intro Lim-transform-eventually
          [OF deriv-ratio-regularly-varying[OF t0 u-deriv u'-cont u-pos lim]]
          eventually-at-top-linorderI[where c=max (t0 + 1) (t0 / y + 1)]
          arg-cong2[where f=divide,OF v-eq[symmetric] v-eq[symmetric]])
        (auto simp: field-simps)
  qed (use v-pos in auto)
  obtain t' where t':⋀x y. t' ≤ x ⟹ x ≤ y ⟹ v y / v x ≤ 2 ∗ (y / x) powr γ
    using Potter-bounds1[where A=2 and d=1,OF rv] by auto
  obtain t'' where t'':⋀x y. t'' ≤ y ⟹ y ≤ x ⟹ v y / v x ≤ 2 ∗ (y / x) powr
(γ − 2)
    using Potter-bounds2[where A=2 and d=1,OF rv] by auto
  define T where T ≡ max (max (t0 + 1) (t0 / x + 1)) (max t' (t'' / x + 1))
  have T:t0 < T ⋀t. T ≤ t ⟹ t0 < t ⋀t. T ≤ t ⟹ t0 < t ∗ x t' ≤ T
    ⋀t. T ≤ t ⟹ t'' ≤ t ∗ x
    and T-pos:0 < T
    using x t0 by(auto simp: T-def field-simps)
  have div-eq1:T ≤ t ⟹ 1 ≤ s ⟹ v (t ∗ s)/ v t = u (t ∗ s) / u t for t s
    using T-pos T
     by(intro arg-cong2[where f=divide,OF v-eq v-eq] order.strict-trans2[of t0 t t
∗ s])
       auto
  have div-eq2:T ≤ t ⟹ x ≤ s ⟹ v (t ∗ s)/ v t = u (t ∗ s) / u t for t s
    using T-pos T x
     by(intro arg-cong2[where f=divide,OF v-eq v-eq] order.strict-trans2[of t0 t ∗
x t ∗ s])
       auto
  have leq1:T ≤ x ⟹ x ≤ y ⟹ v y / v x ≤ 2 ∗ (y / x) powr γ for x y
    using t' t'' T by simp-all
  consider x < 1 | 1 < x | x = 1
    by linarith
  thus ?thesis
  proof cases
    assume x1:1 < x
    have ∀F t in at-top. (U (t ∗ x) − U t) / (t ∗ u t) = (∫ s∈{1..x}. v (t ∗ s) / v
```

```
t ∂lborel)
  proof(safe intro!: eventually-at-top-linorderI[where c=T])
    fix t
    assume t:T ≤ t
    hence t-pos:0 < t and t0t:t0 < t
      using T-pos T by simp-all
    have (U (t * x) - U t) / (t * u t)
          = (∫ s. v s / (t * v t) * indicat-real {t..t * x} s ∂lborel)
      using U-deriv x1 t-pos v-pos t0t
      by(subst integral-FTC-Icc-nonneg[where F=λx. U x / (t * v t)])
        (auto simp: v-eq[symmetric] diff-divide-distrib
           intro!: divide-nonneg-pos less-imp-le DERIV-cdivide)
    also have ... = (∫ s∈{1..x}. v (t * s) / v t ∂lborel)
      using t-pos v-pos
      by(subst lborel-integral-real-affine
          [of t λs. v s / (t * v t) * indicat-real {t..t * x} s 0])
        (auto simp: set-lebesgue-integral-def indicator-def
          intro!: Bochner-Integration.integral-cong)
    finally show (U (t * x) - U t) / (t * u t) = (∫ s∈{1..x}. v (t * s) / v t
∂lborel) .
  qed
  moreover have ((λt. (∫ s∈{1..x}. v (t * s) / v t  ∂lborel))
                  ⟶ (∫ s∈{1..x}. s powr (γ - 1) ∂lborel)) at-top
    unfolding set-lebesgue-integral-def
  proof(rule integral-dominated-convergence-real[OF - - - AE-I2 AE-I2])
    fix t s
    assume T ≤ t
    thus norm (indicat-real {1..x} s *R (v (t * s) / v t))
          ≤ 2 * (indicator {1..x} s *R s powr γ)
      using v-pos leq1[of t t * s] T-pos
      by(auto simp: indicator-def abs-of-pos)
  next
    fix y
    show ((λt. indicat-real {1..x} y *R (v (t * y) / v t))
          ⟶ indicat-real {1..x} y *R y powr (γ - 1)) at-top
      using regularly-varying-lim[OF rv,of y] by(auto simp: indicator-def)
  qed(auto intro!: borel-integrable-compact continuous-on-powr continuous-on-mult
       simp del: real-scaleR-def)
  ultimately have lim':((λt. (U (t * x) - U t) / (t * u t))
                  ⟶ (∫ s∈{1..x}. s powr (γ - 1) ∂lborel)) at-top
    using tendsto-cong by fast
  have (∫ s∈{1..x}. s powr (γ - 1) ∂lborel) = (if γ = 0 then ln x else (x powr
γ - 1) / γ)
  proof(cases γ = 0)
    assume gamma:γ = 0
   then have (∫ s∈{1..x}. s powr (γ - 1) ∂lborel) = (∫ s∈{1<..x}. 1 / s ∂lborel)
      by(auto intro!: set-integral-cong simp: LBINT-Ioc-eq-Icc )
    also have ... = ln x
      using x1 by(simp add: set-integral-inverse)
```

```
    finally show ?thesis
      using gamma by simp
  qed(use x1 in auto simp: set-integral-powr-Icc diff-divide-distrib)
  with lim' show ?thesis
    by simp
next
  assume x1:x < 1
  have ∀F t in at-top. (U (t * x) - U t) / (t * u t) = - (∫ s∈{x..1}. v (t * s)
/ v t ∂lborel)
  proof(safe intro!: eventually-at-top-linorderI[where c=T])
    fix t
    assume t:T ≤ t
    hence t-pos:0 < t and t0t:t0 < t * x t0 < t
      using T-pos T by simp-all
    have (U (t * x) - U t) / (t * u t)
        = - (∫ s. v s / (t * v t) * indicat-real {t * x..t} s ∂lborel)
      using x1 t-pos v-pos t0t U-deriv
      by(subst integral-FTC-Icc-nonneg[where F=λx. U x / (t * v t)])
        (auto simp: v-eq[symmetric] diff-divide-distrib
          intro!: divide-nonneg-pos less-imp-le DERIV-cdivide)
    also have ... = - (∫ s∈{x..1}. v (t * s) / v t ∂lborel)
      using t-pos v-pos
      by(subst lborel-integral-real-affine
          [of t λs. v s / (t * v t) * indicat-real {t * x..t} s 0])
        (auto simp: set-lebesgue-integral-def indicator-def
          intro!: Bochner-Integration.integral-cong)
    finally show (U (t * x) - U t) / (t * u t) = - (∫ s∈{x..1}. v (t * s) / v t
∂lborel) .
  qed
  moreover have ((λt. (∫ s∈{x..1}. v (t * s) / v t ∂lborel))
                  ⟶ (∫ s∈{x..1}. s powr (γ - 1) ∂lborel)) at-top
    unfolding set-lebesgue-integral-def
  proof(rule integral-dominated-convergence-real[OF - - - AE-I2 AE-I2])
    fix t s
    assume t:T ≤ t
    hence x ≤ s ⟹ t'' ≤ t * s
      using T-pos t x by(auto intro!: order.trans[OF T(5)[OF t]] mult-mono)
    thus norm (indicat-real {x..1} s *R (v (t * s) / v t))
        ≤ 2 * (indicator {x..1} s *R s powr (γ - 2))
      using v-pos t''[of t * s t] T-pos t
      by(auto simp: indicator-def abs-of-pos)
  next
    fix y
    show ((λt. indicat-real {x..1} y *R (v (t * y) / v t))
        ⟶ indicat-real {x..1} y *R y powr (γ - 1)) at-top
      using regularly-varying-lim[OF rv,of y] x by(auto simp: indicator-def)
  qed(use x in auto intro!: borel-integrable-compact continuous-on-powr continu-
ous-on-mult
      simp del: real-scaleR-def)
```

```
    ultimately have lim':((λt. (U (t * x) − U t) / (t * u t))
                    ⟶ − (∫ s∈{x..1}. s powr (γ − 1) ∂lborel)) at-top
      using tendsto-cong tendsto-minus by fast
    have (∫ s∈{x..1}. s powr (γ − 1) ∂lborel) = − (if γ = 0 then ln x else (x powr
γ − 1) / γ)
    proof(cases γ = 0)
      assume gamma:γ = 0
     then have (∫ s∈{x..1}. s powr (γ − 1) ∂lborel) = (∫ s∈{x<..1}. 1 / s ∂lborel)
        using x by(auto intro!: set-integral-cong simp: LBINT-Ioc-eq-Icc)
      also have ... = − ln x
        using x x1 by(simp add: set-integral-inverse)
      finally show ?thesis
        using gamma by simp
    qed(use x1 x in auto simp: set-integral-powr-Icc diff-divide-distrib)
    with lim' show ?thesis
      by simp
  qed simp
qed

end
```

References

[1] N. H. Bingham, C. M. Goldie, and J. L. Teugels. *Regular Variation*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1987.

[2] L. de Haan and A. Ferreira. *Extreme Value Theory: An Introduction*. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[3] S. I. Resnick. *Heavy-Tail Phenomena*. Springer New York, NY, 2007.