Actuarial Mathematics

Yosuke Ito

February 6, 2026

Abstract

Actuarial Mathematics is a theory in applied mathematics, which is mainly used for determining the prices of insurance products and evaluating the liability of a company associating with insurance contracts. It is related to calculus, probability theory and financial theory, etc.

In this entry, I formalize the very basic part of Actuarial Mathematics in Isabelle/HOL. It includes the theory of interest, survival model, and life table. The theory of interest deals with interest rates, present value factors, an annuity certain, etc. The survival model is a probabilistic model that represents the human mortality. The life table is based on the survival model and used for practical calculations.

I have already formalized the basic part of Actuarial Mathematics in Coq (https://github.com/Yosuke-Ito-345/Actuary) in a purely axiomatic manner. In contrast, Isabelle formalization is based on the probability theory and the survival model is developed as generally as possible. Such rigorous and general formulation seems very rare; at least I cannot find any similar documentation on the web.

This formalization in Isabelle is still at an early stage, and I cannot guarantee the backward compatibility in the future development. If you heavily depend on the "Actuarial Mathematics" library, please let me know.

Contents

1 Preliminary Lemmas **1**
1.1 Lemmas on *indicator* for a Linearly Ordered Type 12

2 Additional Lemmas for the *HOL−Analysis* Library **15**
2.1 Set Lebesgue Integrability on Affine Transformation 30
2.2 Set Lebesgue Integral on Affine Transformation 32
2.3 Alternative Integral Test 34
2.4 Interchange of Differentiation and Lebesgue Integration . . . 36

3 Additional Lemmas for the *HOL−Probability* Library **45**
3.1 More Properties of *cdf*'s . . . 46
3.2 Conditional Probability Space . . . 54
3.3 Complementary Cumulative Distribution Function . . . 59
3.4 Hazard Rate . . . 63

4 Theory of Interest **70**

5 Survival Model **88**
5.1 General Theory of Survival Model . . . 88
5.1.1 Introduction of Survival Function for X . . . 88
5.1.2 Introdution of Future-Lifetime Random Variable $T(x)$ 91
5.1.3 Actuarial Notations on the Survival Model . . . 92
5.1.4 Properties of Survival Function for $T(x)$ . . . 93
5.1.5 Properties of $\$p$-$\{t\&x\}$ . . . 98
5.1.6 Properties of Survival Function for X . . . 99
5.1.7 Introduction of Cumulative Distributive Function for X 99
5.1.8 Properties of Cumulative Distributive Function for $T(x)$ 100
5.1.9 Properties of $\$q$-$\{t\&x\}$ . . . 102
5.1.10 Properties of Cumulative Distributive Function for X 104
5.1.11 Relations between $\$p$-$\{t\&x\}$ and $\$q$-$\{t\&x\}$ . . . 105
5.1.12 Properties of Life Expectation . . . 106
5.2 Piecewise Differentiable Survival Function . . . 120
5.2.1 Properties of Survival Function for X . . . 120
5.2.2 Properties of Cumulative Distributive Function for X 121
5.2.3 Introduction of Probability Density Functions of X and $T(x)$ . . . 122
5.2.4 Properties of Survival Function for $T(x)$ . . . 123
5.2.5 Properties of Cumulative Distributive Function for $T(x)$ 125
5.2.6 Properties of Probability Density Function of $T(x)$ . . 126
5.2.7 Properties of Probability Density Function of X . . . 132
5.2.8 Relations between Life Expectation and Probability Density Function . . . 133
5.2.9 Introduction of Force of Mortality . . . 136
5.2.10 Properties of Force of Mortality . . . 138
5.2.11 Properties of Curtate Life Expectation . . . 150
5.3 Limited Survival Function . . . 151

6 Life Table **153**
6.1 Basic Properties of Life Table . . . 153
6.2 Construction of Survival Model from Life Table . . . 158
6.2.1 Relations between Life Table and Survival Function for X . . . 159

6.2.2 Relations between Life Table and Cumulative Distributive Function for X 160
6.2.3 Relations between Life Table and Survival Function for $T(x)$ 161
6.2.4 Relations between Life Table and Cumulative Distributive Function for $T(x)$ 163
6.2.5 Life Table and Actuarial Notations 163
6.3 Piecewise Differentiable Life Table 167
6.4 Interpolations 173
6.5 Limited Life Table 189

7 Examples **190**

theory *Preliminaries*
imports *HOL−Library.Rewrite HOL−Library.Extended-Nonnegative-Real HOL−Library.Extended-Real HOL−Probability.Probability*
begin
declare [[*show-types*]]
notation *powr* (**infixr** ‹.^› *80*)

1 Preliminary Lemmas

lemma *Collect-conj-eq2*: $\{x \in A.\ P\ x \wedge Q\ x\} = \{x \in A.\ P\ x\} \cap \{x \in A.\ Q\ x\}$
by *blast*

lemma *vimage-compl-atMost*:
fixes $f :: 'a \Rightarrow 'b$::*linorder*
shows $-(f\ -`\ \{..y\}) = f\ -`\ \{y<..\}$
by *fastforce*

context *linorder*
begin

lemma *Icc-minus-Ico*:
fixes $a\ b$
assumes $a \leq b$
shows $\{a..b\} - \{a..<b\} = \{b\}$
proof
{ fix x **assume** $x \in \{a..b\} - \{a..<b\}$
hence $x \in \{b\}$ **by** *force* **}**
thus $\{a..b\} - \{a..<b\} \subseteq \{b\}$ **by** *blast*
next
show $\{b\} \subseteq \{a..b\} - \{a..<b\}$ **using** *assms* **by** *simp*
qed

lemma *Icc-minus-Ioc*:
fixes $a\ b$
assumes $a \leq b$

shows $\{a..b\} - \{a<..b\} = \{a\}$
proof
{ fix x **assume** $x \in \{a..b\} - \{a<..b\}$
hence $x \in \{a\}$ **by** *force* **}**
thus $\{a..b\} - \{a<..b\} \subseteq \{a\}$ **by** *blast*
next
show $\{a\} \subseteq \{a..b\} - \{a<..b\}$ **using** *assms* **by** *simp*
qed

lemma *Int-atLeastAtMost-Unbounded*[*simp*]: $\{a..\}$ *Int* $\{..b\} = \{a..b\}$
by *auto*

lemma *Int-greaterThanAtMost-Unbounded*[*simp*]: $\{a<..\}$ *Int* $\{..b\} = \{a<..b\}$
by *auto*

lemma *Int-atLeastLessThan-Unbounded*[*simp*]: $\{a..\}$ *Int* $\{..<b\} = \{a..<b\}$
by *auto*

lemma *Int-greaterThanLessThan-Unbounded*[*simp*]: $\{a<..\}$ *Int* $\{..<b\} = \{a<..<b\}$
by *auto*

end

lemma *Ico-real-nat-disjoint*:
disjoint-family (λn::*nat*. $\{a + real\ n\ ..<\ a + real\ n + 1\}$) **for** *a*::*real*
proof −
{ fix $m\ n$ :: *nat*
assume $\{a + real\ m\ ..<\ a + real\ m + 1\} \cap \{a + real\ n\ ..<\ a + real\ n + 1\} \neq \{\}$
then obtain *x*::*real*
where $x \in \{a + real\ m\ ..<\ a + real\ m + 1\} \cap \{a + real\ n\ ..<\ a + real\ n + 1\}$ **by** *force*
hence $m = n$ **by** *simp* **}**
thus *?thesis* **unfolding** *disjoint-family-on-def* **by** *blast*
qed

corollary *Ico-nat-disjoint*: *disjoint-family* (λn::*nat*. $\{real\ n\ ..<\ real\ n + 1\}$)
using *Ico-real-nat-disjoint*[*of 0*] **by** *simp*

lemma *Ico-real-nat-union*: $(\bigcup n$::*nat*. $\{a + real\ n\ ..<\ a + real\ n + 1\}) = \{a..\}$
for *a*::*real*
proof
show $(\bigcup n$::*nat*. $\{a + real\ n\ ..<\ a + real\ n + 1\}) \subseteq \{a..\}$ **by** *auto*
next
show $\{a..\} \subseteq (\bigcup n$::*nat*. $\{a + real\ n\ ..<\ a + real\ n + 1\})$
proof
fix x **assume** $x \in \{a..\}$
hence $a \leq x$ **by** *simp*

```
    hence nat ⌊x−a⌋ ≤ x−a ∧ x−a < nat ⌊x−a⌋ + 1 by linarith
    hence a + nat ⌊x−a⌋ ≤ x ∧ x < a + nat ⌊x−a⌋ + 1 by auto
    thus x ∈ (⋃ n::nat. {a + real n ..< a + real n + 1}) by auto
  qed
qed

corollary Ico-nat-union: (⋃ n::nat. {real n ..< real n + 1}) = {0..}
  using Ico-real-nat-union[of 0] by simp

lemma Ico-nat-union-finite: (⋃ (n::nat)<m. {real n ..< real n + 1}) = {0..<m}
proof
  show (⋃ (n::nat)<m. {real n ..< real n + 1}) ⊆ {0..<m} by auto
next
  show {0..<m} ⊆ (⋃ (n::nat)<m. {real n ..< real n + 1})
  proof
    fix x::real
    assume ⋆: x ∈ {0..<m}
    hence nat ⌊x⌋ < m using of-nat-floor by fastforce
    moreover with ⋆ have nat ⌊x⌋ ≤ x ∧ x < nat ⌊x⌋ + 1
      by (metis Nat.add-0-right atLeastLessThan-iff le-nat-floor
          less-one linorder-not-le nat-add-left-cancel-le of-nat-floor)
    ultimately show x ∈ (⋃ (n::nat)<m. {real n ..< real n + 1})
      unfolding atLeastLessThan-def by force
  qed
qed

lemma seq-part-multiple: fixes m n :: nat assumes m ≠ 0 defines A ≡ λi::nat.
{i∗m ..< (i+1)∗m}
  shows ∀ i j. i ≠ j ⟶ A i ∩ A j = {} and (⋃ i<n. A i) = {..< n∗m}
proof −
  { fix i j :: nat
    have i ≠ j ⟹ A i ∩ A j = {}
    proof (erule contrapos-np)
      assume A i ∩ A j ≠ {}
      then obtain k where k ∈ A i ∩ A j by blast
      hence i∗m < (j+1)∗m ∧ j∗m < (i+1)∗m unfolding A-def by force
      hence i < j+1 ∧ j < i+1 using mult-less-cancel2 by blast
      thus i = j by force
    qed }
  thus ∀ i j. i ≠ j ⟶ A i ∩ A j = {} by blast
next
  show (⋃ i<n. A i) = {..< n∗m}
  proof
    show (⋃ i<n. A i) ⊆ {..< n∗m}
    proof
      fix x::nat
      assume x ∈ (⋃ i<n. A i)
     then obtain i where i-n: i < n and i-x: x < (i+1)∗m unfolding A-def by
force
```

```
      hence i+1 ≤ n by linarith
      hence x < n*m by (meson less-le-trans mult-le-cancel2 i-x)
      thus x ∈ {..< n*m}
        using diff-mult-distrib mult-1 i-n by auto
    qed
  next
    show {..< n*m} ⊆ (⋃ i<n. A i)
    proof
      fix x::nat
      let ?i = x div m
      assume x ∈ {..< n*m}
      hence ?i < n by (simp add: less-mult-imp-div-less)
      moreover have ?i*m ≤ x ∧ x < (?i+1)*m
        using assms div-times-less-eq-dividend dividend-less-div-times by auto
      ultimately show x ∈ (⋃ i<n. A i) unfolding A-def by force
    qed
  qed
qed

lemma frontier-Icc-real: frontier {a..b} = {a, b} if a ≤ b for a b :: real
  unfolding frontier-def using that by force

lemma(in field) divide-mult-cancel[simp]: fixes a b assumes b ≠ 0
  shows a / b * b = a
  by (simp add: assms)

lemma inverse-powr: (1/a).^b = a.^-b if a > 0 for a b :: real
  by (smt (verit) that powr-divide powr-minus-divide powr-one-eq-one)

lemma powr-eq-one-iff-gen[simp]: a.^x = 1 ⟷ x = 0 if a > 0 a ≠ 1 for a x ::
real
  by (metis powr-eq-0-iff powr-inj powr-zero-eq-one that)

lemma powr-less-cancel2: 0 < a ⟹ 0 < x ⟹ 0 < y ⟹ x.^a < y.^a ⟹ x <
y
  for a x y ::real
proof -
  assume a-pos: 0 < a and x-pos: 0 < x and y-pos: 0 < y
  show x.^a < y.^a ⟹ x < y
  proof (erule contrapos-pp)
    assume ¬ x < y
    hence x ≥ y by fastforce
    hence x.^a ≥ y.^a
    proof (cases x = y)
      case True
      thus ?thesis by simp
    next
      case False
      hence x.^a > y.^a
```

```
      using ‹x ≥ y› powr-less-mono2 a-pos y-pos by auto
    thus ?thesis by auto
  qed
  thus ¬ x.^a < y.^a by fastforce
 qed
qed

lemma geometric-increasing-sum-aux: (1−r)^2 * (∑k<n. (k+1)*r^k) = 1 −
(n+1)*r^n + n*r^(n+1)
 for n::nat and r::real
proof (induct n)
 case 0
 thus ?case by simp
next
 case (Suc n)
 thus ?case
  apply (simp only: sum.lessThan-Suc)
  apply (subst distrib-left)
  apply (subst Suc.hyps)
  by (subst power2-diff, simp add: field-simps power2-eq-square)
qed

lemma geometric-increasing-sum: (∑k<n. (k+1)*r^k) = (1 − (n+1)*r^n +
n*r^(n+1)) / (1−r)^2
 if r ≠ 1 for n::nat and r::real
 by (subst geometric-increasing-sum-aux[THEN sym], simp add: that)

lemma Reals-UNIV[simp]: ℝ = {x::real. True}
 unfolding Reals-def by auto

lemma Lim-cong:
 assumes ∀F x in F. f x = g x
 shows Lim F f = Lim F g
 unfolding t2-space-class.Lim-def using tendsto-cong assms by fastforce

lemma LIM-zero-iff': ((λx. l − f x) ⟶ 0) F = (f ⟶ l) F
 for f :: 'a ⇒ 'b::real-normed-vector
 unfolding tendsto-iff dist-norm
 by (rewrite minus-diff-eq[THEN sym], rewrite norm-minus-cancel) simp

lemma antimono-onI:
 (⋀r s. r ∈ A ⟹ s ∈ A ⟹ r ≤ s ⟹ f r ≥ f s) ⟹ antimono-on A f
 by (rule monotone-onI)

lemma antimono-onD:
 ⟦antimono-on A f; r ∈ A; s ∈ A; r ≤ s⟧ ⟹ f r ≥ f s
 by (rule monotone-onD)

lemma antimono-imp-mono-on: antimono f ⟹ antimono-on A f
```

by (*simp add*: *antimonoD antimono-onI*)

lemma *antimono-on-subset*: *antimono-on A f* $\Longrightarrow$ $B \subseteq A$ $\Longrightarrow$ *antimono-on B f*
by (*rule monotone-on-subset*)

lemma *mono-on-antimono-on*:
fixes *f* :: *'a*::*order* $\Rightarrow$ *'b*::*ordered-ab-group-add*
shows *mono-on A f* $\longleftrightarrow$ *antimono-on A* ($\lambda r.\ -\ f\ r$)
by (*simp add*: *monotone-on-def*)

corollary *mono-antimono*:
fixes *f* :: *'a*::*order* $\Rightarrow$ *'b*::*ordered-ab-group-add*
shows *mono f* $\longleftrightarrow$ *antimono* ($\lambda r.\ -\ f\ r$)
by (*rule mono-on-antimono-on*)

lemma *mono-on-at-top-le*:
fixes *a* :: *'a*::*linorder* **and** *b* :: *'b*::{*order-topology*, *linordered-ab-group-add*}
and *f* :: *'a* $\Rightarrow$ *'b*
assumes *f-mono*: *mono-on* {*a*..} *f* **and** *f-to-l*: (*f* $\longrightarrow$ *l*) *at-top*
shows $\bigwedge x.\ x \in \{a..\} \Longrightarrow f\ x \leq l$
proof (*unfold atomize-ball*)
{ fix *b* **assume** *b-a*: $b \geq a$ **and** *fb-l*: $\neg\ f\ b \leq l$
let *?eps* = $f\ b - l$
have *lim-top*: $\bigwedge S.$ *open S* $\Longrightarrow$ $l \in S$ $\Longrightarrow$ *eventually* ($\lambda x.\ f\ x \in S$) *at-top*
using *assms tendsto-def* **by** *auto*
have *eventually* ($\lambda x.\ f\ x \in \{l - ?eps <..< l + ?eps\}$) *at-top*
using *fb-l* **by** (*intro lim-top*; *force*)
then obtain *N* **where** *fn-in*: $\bigwedge n.\ n \geq N \Longrightarrow f\ n \in \{l - ?eps <..< l + ?eps\}$
using *eventually-at-top-linorder* **by** *metis*
let *?n* = *max b N*
have *f ?n* < *f b* **using** *fn-in* **by** *force*
moreover have *f ?n* $\geq$ *f b* **using** *f-mono b-a* **by** (*simp add*: *le-max-iff-disj mono-on-def*)
ultimately have *False* **by** *simp* **}**
thus $\forall x{\in}\{a..\}.\ f\ x \leq l$
apply −
by (*rule notnotD*, *rewrite Set.ball-simps*) *auto*
qed

corollary *mono-at-top-le*:
fixes *b* :: *'b*::{*order-topology*, *linordered-ab-group-add*} **and** *f* :: *'a*::*linorder* $\Rightarrow$ *'b*
assumes *mono f* **and** (*f* $\longrightarrow$ *b*) *at-top*
shows $\bigwedge x.\ f\ x \leq b$
using *mono-on-at-top-le assms* **by** (*metis atLeast-iff mono-imp-mono-on nle-le*)

lemma *mono-on-at-bot-ge*:
fixes *a* :: *'a*::*linorder* **and** *b* :: *'b*::{*order-topology*, *linordered-ab-group-add*}
and *f* :: *'a* $\Rightarrow$ *'b*
assumes *f-mono*: *mono-on* {..*a*} *f* **and** *f-to-l*: (*f* $\longrightarrow$ *l*) *at-bot*

shows $\bigwedge x.\ x \in \{..a\} \Longrightarrow f\ x \geq l$
proof (*unfold atomize-ball*)
{ fix b **assume** *b-a*: $b \leq a$ **and** *fb-l*: $\neg\ f\ b \geq l$
let *?eps* $= l - f\ b$
have *lim-bot*: $\bigwedge S.$ *open* $S \Longrightarrow l \in S \Longrightarrow$ *eventually* $(\lambda x.\ f\ x \in S)$ *at-bot*
using *assms tendsto-def* **by** *auto*
have *eventually* $(\lambda x.\ f\ x \in \{l - ?eps <..< l + ?eps\})$ *at-bot*
using *fb-l* **by** (*intro lim-bot*; *force*)
then obtain N **where** *fn-in*: $\bigwedge n.\ n \leq N \Longrightarrow f\ n \in \{l - ?eps <..< l + ?eps\}$
using *eventually-at-bot-linorder* **by** *metis*
let *?n* $=$ *min b N*
have f *?n* $> f\ b$ **using** *fn-in* **by** *force*
moreover have f *?n* $\leq f\ b$ **using** *f-mono b-a* **by** (*simp add*: *min.coboundedI1 mono-onD*)
ultimately have *False* **by** *simp* **}**
thus $\forall x{\in}\{..a\}.\ f\ x \geq l$
apply −
by (*rule notnotD*, *rewrite Set.ball-simps*) *auto*
qed

corollary *mono-at-bot-ge*:
fixes $b :: 'b{::}\{$*order-topology*, *linordered-ab-group-add*$\}$ **and** $f :: 'a{::}$*linorder* $\Rightarrow 'b$
assumes *mono* f **and** $(f \longrightarrow b)$ *at-bot*
shows $\bigwedge x.\ f\ x \geq b$
using *mono-on-at-bot-ge assms* **by** (*metis atMost-iff mono-imp-mono-on nle-le*)

lemma *antimono-on-at-top-ge*:
fixes $a :: 'a{::}$*linorder* **and** $b :: 'b{::}\{$*order-topology*, *linordered-ab-group-add*$\}$
and $f :: 'a \Rightarrow 'b$
assumes *f-antimono*: *antimono-on* $\{a..\}$ f **and** *f-to-l*: $(f \longrightarrow l)$ *at-top*
shows $\bigwedge x.\ x \in \{a..\} \Longrightarrow f\ x \geq l$
proof (*unfold atomize-ball*)
{ fix b **assume** *b-a*: $b \geq a$ **and** *fb-l*: $\neg\ f\ b \geq l$
let *?eps* $= l - f\ b$
have *lim-top*: $\bigwedge S.$ *open* $S \Longrightarrow l \in S \Longrightarrow$ *eventually* $(\lambda x.\ f\ x \in S)$ *at-top*
using *assms tendsto-def* **by** *auto*
have *eventually* $(\lambda x.\ f\ x \in \{l - ?eps <..< l + ?eps\})$ *at-top*
using *fb-l* **by** (*intro lim-top*; *force*)
then obtain N **where** *fn-in*: $\bigwedge n.\ n \geq N \Longrightarrow f\ n \in \{l - ?eps <..< l + ?eps\}$
using *eventually-at-top-linorder* **by** *metis*
let *?n* $=$ *max b N*
have f *?n* $> f\ b$ **using** *fn-in* **by** *force*
moreover have f *?n* $\leq f\ b$ **using** *f-antimono b-a*
by (*simp add*: *antimono-onD le-max-iff-disj*)
ultimately have *False* **by** *simp* **}**
thus $\forall x{\in}\{a..\}.\ f\ x \geq l$
apply −
by (*rule notnotD*, *rewrite Set.ball-simps*) *auto*
qed

corollary *antimono-at-top-le*:
fixes $b :: 'b::\{order\text{-}topology, linordered\text{-}ab\text{-}group\text{-}add\}$ **and** $f :: 'a::linorder \Rightarrow 'b$
assumes *antimono f* **and** $(f \longrightarrow b)$ *at-top*
shows $\bigwedge x.\ f\ x \geq b$
using *antimono-on-at-top-ge assms antimono-imp-mono-on* **by** *blast*

lemma *antimono-on-at-bot-ge*:
fixes $a :: 'a::linorder$ **and** $b :: 'b::\{order\text{-}topology, linordered\text{-}ab\text{-}group\text{-}add\}$
and $f :: 'a \Rightarrow 'b$
assumes *f-antimono*: *antimono-on* $\{..a\}$ *f* **and** *f-to-l*: $(f \longrightarrow l)$ *at-bot*
shows $\bigwedge x.\ x \in \{..a\} \Longrightarrow f\ x \leq l$
proof (*unfold atomize-ball*)
{ fix *b* **assume** *b-a*: $b \leq a$ **and** *fb-l*: $\neg\ f\ b \leq l$
let *?eps* $= f\ b - l$
have *lim-bot*: $\bigwedge S.$ *open* $S \Longrightarrow l \in S \Longrightarrow$ *eventually* $(\lambda x.\ f\ x \in S)$ *at-bot*
using *assms tendsto-def* **by** *auto*
have *eventually* $(\lambda x.\ f\ x \in \{l - ?eps <..< l + ?eps\})$ *at-bot*
using *fb-l* **by** (*intro lim-bot*; *force*)
then obtain *N* **where** *fn-in*: $\bigwedge n.\ n \leq N \Longrightarrow f\ n \in \{l - ?eps <..< l + ?eps\}$
using *eventually-at-bot-linorder* **by** *metis*
let *?n* $=$ *min b N*
have *f ?n* $< f\ b$ **using** *fn-in* **by** *force*
moreover have *f ?n* $\geq f\ b$ **using** *f-antimono b-a* **by** (*simp add*: *min.coboundedI1 antimono-onD*)
ultimately have *False* **by** *simp* **}**
thus $\forall x{\in}\{..a\}.\ f\ x \leq l$
apply $-$
by (*rule notnotD*, *rewrite Set.ball-simps*) *auto*
qed

corollary *antimono-at-bot-ge*:
fixes $b :: 'b::\{order\text{-}topology, linordered\text{-}ab\text{-}group\text{-}add\}$ **and** $f :: 'a::linorder \Rightarrow 'b$
assumes *antimono f* **and** $(f \longrightarrow b)$ *at-bot*
shows $\bigwedge x.\ f\ x \leq b$
using *antimono-on-at-bot-ge assms antimono-imp-mono-on* **by** *blast*

lemma *continuous-cdivide*:
fixes *c*::*'a*::*real-normed-field*
assumes $c \neq 0$ *continuous F f*
shows *continuous F* $(\lambda x.\ f\ x\ /\ c)$
using *assms continuous-mult-right* **by** (*rewrite field-class.field-divide-inverse*) *auto*

lemma *continuous-mult-left-iff*:
fixes *c*::*'a*::*real-normed-field*
assumes $c \neq 0$
shows *continuous F f* $\longleftrightarrow$ *continuous F* $(\lambda x.\ c * f\ x)$
using *continuous-mult-left continuous-cdivide assms* **by** *force*

lemma *continuous-mult-right-iff*:
fixes *c*::*'a*::*real-normed-field*
assumes *c* ≠ *0*
shows *continuous F f* ⟷ *continuous F* (λ*x*. *f x* ∗ *c*)
using *continuous-mult-right continuous-cdivide assms* **by** *force*

lemma *continuous-cdivide-iff*:
fixes *c*::*'a*::*real-normed-field*
assumes *c* ≠ *0*
shows *continuous F f* ⟷ *continuous F* (λ*x*. *f x* / *c*)
proof
assume *continuous F f*
thus *continuous F* (λ*x*. *f x* / *c*)
by (*intro continuous-cdivide*) (*simp add*: *assms*)
next
assume *continuous F* (λ*x*. *f x* / *c*)
hence *continuous F* (λ*x*. *f x* / *c* ∗ *c*) **using** *continuous-mult-right* **by** *fastforce*
thus *continuous F f* **using** *assms* **by** *force*
qed

lemma *continuous-cong*:
assumes *eventually* (λ*x*. *f x* = *g x*) *F f* (*Lim F* (λ*x*. *x*)) = *g* (*Lim F* (λ*x*. *x*))
shows *continuous F f* ⟷ *continuous F g*
unfolding *continuous-def* **using** *assms filterlim-cong* **by** *force*

lemma *continuous-at-within-cong*:
assumes *f x* = *g x eventually* (λ*x*. *f x* = *g x*) (*at x within s*)
shows *continuous* (*at x within s*) *f* ⟷ *continuous* (*at x within s*) *g*
proof (*cases* ‹*x* ∈ *closure* (*s* − {*x*})›)
case *True*
thus *?thesis*
using *assms* **apply** (*intro continuous-cong*, *simp*)
by (*rewrite Lim-ident-at*, *simp add*: *at-within-eq-bot-iff*)+ *simp*
next
case *False*
hence *at x within s* = *bot* **using** *not-trivial-limit-within* **by** *blast*
thus *?thesis* **by** *simp*
qed

lemma *continuous-within-shift*:
fixes *a x* :: *'a* :: {*topological-ab-group-add*, *t2-space*}
and *s* :: *'a set*
and *f* :: *'a* ⇒ *'b*::*topological-space*
shows *continuous* (*at x within s*) (λ*x*. *f* (*x*+*a*)) ⟷ *continuous* (*at* (*x*+*a*) *within plus a* ‘ *s*) *f*
proof
assume *continuous* (*at x within s*) (λ*x*. *f* (*x*+*a*))
moreover have *continuous* (*at* (*x*+*a*) *within plus a* ‘ *s*) (*plus* (−*a*))

by (*simp add*: *continuous-at-imp-continuous-at-within*)
moreover have *plus* (−*a*) ‘ *plus a* ‘ *s* = *s* **by** *force*
ultimately show *continuous* (*at* (*x*+*a*) *within plus a* ‘ *s*) *f*
using *continuous-within-compose* **unfolding** *comp-def* **by** *force*
next
assume *continuous* (*at* (*x*+*a*) *within plus a* ‘ *s*) *f*
moreover have *continuous* (*at x within s*) (*plus a*)
by (*simp add*: *continuous-at-imp-continuous-at-within*)
ultimately show *continuous* (*at x within s*) (λ*x*. *f* (*x*+*a*))
using *continuous-within-compose* **unfolding** *comp-def* **by** (*force simp add*: *add.commute*)
qed

lemma *isCont-shift*:
fixes *a x* :: ′*a* :: {*topological-ab-group-add*, *t2-space*}
and *f* :: ′*a* ⇒ ′*b*::*topological-space*
shows *isCont* (λ*x*. *f* (*x*+*a*)) *x* ⟷ *isCont f* (*x*+*a*)
using *continuous-within-shift* **by** *force*

lemma *has-real-derivative-at-split*:
(*f has-real-derivative D*) (*at x*) ⟷
(*f has-real-derivative D*) (*at-left x*) ∧ (*f has-real-derivative D*) (*at-right x*)
proof −
have *at x* = *at x within* ({..<*x*} ∪ {*x*<..}) **by** (*simp add*: *at-eq-sup-left-right at-within-union*)
thus (*f has-real-derivative D*) (*at x*) ⟷
(*f has-real-derivative D*) (*at-left x*) ∧ (*f has-real-derivative D*) (*at-right x*)
using *Lim-within-Un has-field-derivative-iff* **by** *fastforce*
qed

lemma *DERIV-cmult-iff*:
assumes *c* ≠ *0*
shows (*f has-field-derivative D*) (*at x within s*) ⟷
((λ*x*. *c* ∗ *f x*) *has-field-derivative c* ∗ *D*) (*at x within s*)
proof
assume (*f has-field-derivative D*) (*at x within s*)
thus ((λ*x*. *c* ∗ *f x*) *has-field-derivative c* ∗ *D*) (*at x within s*) **using** *DERIV-cmult*
by *force*
next
assume ((λ*x*. *c* ∗ *f x*) *has-field-derivative c* ∗ *D*) (*at x within s*)
hence ((λ*x*. *c* ∗ *f x* / *c*) *has-field-derivative c* ∗ *D* / *c*) (*at x within s*)
using *DERIV-cdivide assms* **by** *blast*
thus (*f has-field-derivative D*) (*at x within s*) **by** (*simp add*: *assms field-simps*)
qed

lemma *DERIV-cmult-right-iff*:
assumes *c* ≠ *0*
shows (*f has-field-derivative D*) (*at x within s*) ⟷
((λ*x*. *f x* ∗ *c*) *has-field-derivative D* ∗ *c*) (*at x within s*)

by (*rewrite DERIV-cmult-iff*[*of c*], *simp-all add*: *assms mult-ac*)

lemma *DERIV-cdivide-iff*:
assumes $c \neq 0$
shows (*f has-field-derivative D*) (*at x within s*) $\longleftrightarrow$
((λ*x*. *f x* / *c*) *has-field-derivative D* / *c*) (*at x within s*)
apply (*rewrite field-class.field-divide-inverse*)+
using *DERIV-cmult-right-iff assms inverse-nonzero-iff-nonzero* **by** *blast*

lemma *DERIV-ln-divide-chain*:
fixes *f* :: *real* ⇒ *real*
assumes *f x* > *0* **and** (*f has-real-derivative D*) (*at x within s*)
shows ((λ*x*. *ln* (*f x*)) *has-real-derivative* (*D* / *f x*)) (*at x within s*)
proof −
have *DERIV ln* (*f x*) :> *1* / *f x* **using** *assms* **by** (*intro DERIV-ln-divide*) *blast*
thus *?thesis* **using** *DERIV-chain2 assms* **by** *fastforce*
qed

lemma *inverse-fun-has-integral-ln*:
fixes *f* :: *real* ⇒ *real* **and** *f'* :: *real* ⇒ *real*
assumes *a* ≤ *b* **and**
⋀*x*. *x*∈{*a*..*b*} ⟹ *f x* > *0* **and**
continuous-on {*a*..*b*} *f* **and**
⋀*x*. *x*∈{*a*<..<*b*} ⟹ (*f has-real-derivative f' x*) (*at x*)
shows ((λ*x*. *f' x* / *f x*) *has-integral* (*ln* (*f b*) − *ln* (*f a*))) {*a*..*b*}
proof −
have *continuous-on* {*a*..*b*} (λ*x*. *ln* (*f x*)) **using** *assms* **by** (*intro continuous-intros*; *force*)
moreover have ⋀*x*. *x*∈{*a*<..<*b*} ⟹ ((λ*x*. *ln* (*f x*)) *has-vector-derivative f' x* / *f x*) (*at x*)
apply (*rewrite has-real-derivative-iff-has-vector-derivative*[*THEN sym*])
using *assms* **by** (*intro DERIV-ln-divide-chain*; *simp*)
ultimately show *?thesis* **using** *assms* **by** (*intro fundamental-theorem-of-calculus-interior*; *simp*)
qed

lemma *DERIV-fun-powr2*:
fixes *a*::*real*
assumes *a-pos*: *a* > *0*
and *f*: *DERIV f x* :> *r*
shows *DERIV* (λ*x*. *a*.^(*f x*)) *x* :> *a*.^(*f x*) ∗ *r* ∗ *ln a*
proof −
let *?g* = (λ*x*. *a*)
have *g*: *DERIV ?g x* :> *0* **by** *simp*
have *pos*: *?g x* > *0* **by** (*simp add*: *a-pos*)
show *?thesis*
using *DERIV-powr*[*OF g pos f*] *a-pos* **by** (*auto simp add*: *field-simps*)
qed

lemma *has-real-derivative-powr2*:
assumes *a-pos*: $a > 0$
shows $((\lambda x.\ a.\hat{}x)$ *has-real-derivative* $a.\hat{}x * ln\ a)\ (at\ x)$
proof −
let *?f* $= (\lambda x.\ x$::*real*$)$
have *f*: *DERIV ?f x :> 1* **by** *simp*
thus *?thesis* **using** *DERIV-fun-powr2*[*OF a-pos f*] **by** *simp*
qed

lemma *field-differentiable-shift*:
$(f$ *field-differentiable* $(at\ (x + z))) = ((\lambda x.\ f\ (x + z))$ *field-differentiable* $(at\ x))$
unfolding *field-differentiable-def* **using** *DERIV-shift* **by** *force*

1.1 Lemmas on *indicator* for a Linearly Ordered Type

lemma *indicator-Icc-shift*:
fixes *a b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{a..b\}\ x =$ *indicator* $\{t+a..t+b\}\ (t+x)$
by (*simp add*: *indicator-def*)

lemma *indicator-Icc-shift-inverse*:
fixes *a b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{a-t..b-t\}\ x =$ *indicator* $\{a..b\}\ (t+x)$
by (*metis add.commute diff-add-cancel indicator-Icc-shift*)

lemma *indicator-Ici-shift*:
fixes *a t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{a..\}\ x =$ *indicator* $\{t+a..\}\ (t+x)$
by (*simp add*: *indicator-def*)

lemma *indicator-Ici-shift-inverse*:
fixes *a t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{a-t..\}\ x =$ *indicator* $\{a..\}\ (t+x)$
by (*metis add.commute diff-add-cancel indicator-Ici-shift*)

lemma *indicator-Iic-shift*:
fixes *b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{..b\}\ x =$ *indicator* $\{..t+b\}\ (t+x)$
by (*simp add*: *indicator-def*)

lemma *indicator-Iic-shift-inverse*:
fixes *b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{..b-t\}\ x =$ *indicator* $\{..b\}\ (t+x)$
by (*metis add.commute diff-add-cancel indicator-Iic-shift*)

lemma *indicator-Icc-reverse*:
fixes *a b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* $\{a..b\}\ x =$ *indicator* $\{t-b..t-a\}\ (t-x)$

by (*metis add-le-cancel-left atLeastAtMost-iff diff-add-cancel indicator-simps le-diff-eq*)

lemma *indicator-Icc-reverse-inverse*:
fixes *a b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* {*t*−*b*..*t*−*a*} *x* = *indicator* {*a*..*b*} (*t*−*x*)
by (*metis add-diff-cancel-left' diff-add-cancel indicator-Icc-reverse*)

lemma *indicator-Ici-reverse*:
fixes *a t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* {*a*..} *x* = *indicator* {..*t*−*a*} (*t*−*x*)
by (*simp add*: *indicator-def*)

lemma *indicator-Ici-reverse-inverse*:
fixes *b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* {*t*−*b*..} *x* = *indicator* {..*b*} (*t*−*x*)
by (*metis add-diff-cancel-left' diff-add-cancel indicator-Ici-reverse*)

lemma *indicator-Iic-reverse*:
fixes *b t x* :: *'a*::*linordered-ab-group-add*
shows *indicator* {..*b*} *x* = *indicator* {*t*−*b*..} (*t*−*x*)
by (*simp add*: *indicator-def*)

lemma *indicator-Iic-reverse-inverse*:
fixes *a t x* :: *'a*::*linordered-field*
shows *indicator* {..*t*−*a*} *x* = *indicator* {*a*..} (*t*−*x*)
by (*simp add*: *indicator-def*)

lemma *indicator-Icc-affine-pos*:
fixes *a b c t x* :: *'a*::*linordered-field*
assumes *c* > *0*
shows *indicator* {*a*..*b*} *x* = *indicator* {*t*+*c*∗*a*..*t*+*c*∗*b*} (*t* + *c*∗*x*)
unfolding *indicator-def* **using** *assms* **by** *simp*

lemma *indicator-Icc-affine-pos-inverse*:
fixes *a b c t x* :: *'a*::*linordered-field*
assumes *c* > *0*
shows *indicator* {(*a*−*t*)/*c*..(*b*−*t*)/*c*} *x* = *indicator* {*a*..*b*} (*t* + *c*∗*x*)
using *indicator-Icc-affine-pos*[**where** *a*=(*a*−*t*)/*c* **and** *b*=(*b*−*t*)/*c* **and** *c*=*c* **and** *t*=*t*] *assms* **by** *simp*

lemma *indicator-Ici-affine-pos*:
fixes *a c t x* :: *'a*::*linordered-field*
assumes *c* > *0*
shows *indicator* {*a*..} *x* = *indicator* {*t*+*c*∗*a*..} (*t* + *c*∗*x*)
unfolding *indicator-def* **using** *assms* **by** *simp*

lemma *indicator-Ici-affine-pos-inverse*:
fixes *a c t x* :: *'a*::*linordered-field*
assumes *c* > *0*

shows *indicator* $\{(a-t)/c..\}$ x = *indicator* $\{a..\}$ $(t + c*x)$
using *indicator-Ici-affine-pos*[**where** $a=(a-t)/c$ **and** $c=c$ **and** $t=t$] *assms* **by** *simp*

lemma *indicator-Iic-affine-pos*:
fixes b c t x :: $'a$::*linordered-field*
assumes $c > 0$
shows *indicator* $\{..b\}$ x = *indicator* $\{..t+c*b\}$ $(t + c*x)$
unfolding *indicator-def* **using** *assms* **by** *simp*

lemma *indicator-Iic-affine-pos-inverse*:
fixes b c t x :: $'a$::*linordered-field*
assumes $c > 0$
shows *indicator* $\{..(b-t)/c\}$ x = *indicator* $\{..b\}$ $(t + c*x)$
using *indicator-Iic-affine-pos*[**where** $b=(b-t)/c$ **and** $c=c$ **and** $t=t$] *assms* **by** *simp*

lemma *indicator-Icc-affine-neg*:
fixes a b c t x :: $'a$::*linordered-field*
assumes $c < 0$
shows *indicator* $\{a..b\}$ x = *indicator* $\{t+c*b..t+c*a\}$ $(t + c*x)$
unfolding *indicator-def* **using** *assms* **by** *auto*

lemma *indicator-Icc-affine-neg-inverse*:
fixes a b c t x :: $'a$::*linordered-field*
assumes $c < 0$
shows *indicator* $\{(b-t)/c..(a-t)/c\}$ x = *indicator* $\{a..b\}$ $(t + c*x)$
using *indicator-Icc-affine-neg*[**where** $a=(b-t)/c$ **and** $b=(a-t)/c$ **and** $c=c$ **and** $t=t$] *assms* **by** *simp*

lemma *indicator-Ici-affine-neg*:
fixes a c t x :: $'a$::*linordered-field*
assumes $c < 0$
shows *indicator* $\{a..\}$ x = *indicator* $\{..t+c*a\}$ $(t + c*x)$
unfolding *indicator-def* **using** *assms* **by** *simp*

lemma *indicator-Ici-affine-neg-inverse*:
fixes b c t x :: $'a$::*linordered-field*
assumes $c < 0$
shows *indicator* $\{(b-t)/c..\}$ x = *indicator* $\{..b\}$ $(t + c*x)$
using *indicator-Ici-affine-neg*[**where** $a=(b-t)/c$ **and** $c=c$ **and** $t=t$] *assms* **by** *simp*

lemma *indicator-Iic-affine-neg*:
fixes b c t x :: $'a$::*linordered-field*
assumes $c < 0$
shows *indicator* $\{..b\}$ x = *indicator* $\{t+c*b..\}$ $(t + c*x)$
unfolding *indicator-def* **using** *assms* **by** *simp*

lemma *indicator-Iic-affine-neg-inverse*:
fixes *a c t x* :: *'a*::*linordered-field*
assumes *c* < *0*
shows *indicator* {..(*a*−*t*)/*c*} *x* = *indicator* {*a*..} (*t* + *c*∗*x*)
using *indicator-Iic-affine-neg*[**where** *b*=(*a*−*t*)/*c* **and** *c*=*c* **and** *t*=*t*] *assms* **by** *simp*

2 Additional Lemmas for the *HOL−Analysis* Library

lemma *differentiable-eq-field-differentiable-real*:
fixes *f* :: *real* ⇒ *real*
shows *f differentiable F* ⟷ *f field-differentiable F*
unfolding *field-differentiable-def differentiable-def has-real-derivative*
using *has-real-derivative-iff* **by** *presburger*

lemma *differentiable-on-eq-field-differentiable-real*:
fixes *f* :: *real* ⇒ *real*
shows *f differentiable-on s* ⟷ (∀ *x*∈*s*. *f field-differentiable* (*at x within s*))
unfolding *differentiable-on-def* **using** *differentiable-eq-field-differentiable-real* **by** *simp*

lemma *differentiable-on-cong* :
assumes ⋀*x*. *x*∈*s* ⟹ *f x* = *g x* **and** *f differentiable-on s*
shows *g differentiable-on s*
proof −
{ fix *x* **assume** *x*∈*s*
hence *f differentiable at x within s* **using** *assms* **unfolding** *differentiable-on-def* **by** *simp*
from *this* **obtain** *D* **where** (*f has-derivative D*) (*at x within s*)
unfolding *differentiable-def* **by** *blast*
hence (*g has-derivative D*) (*at x within s*)
using *has-derivative-transform assms* ‹*x*∈*s*› **by** *metis*
hence *g differentiable at x within s* **unfolding** *differentiable-def* **by** *blast* **}**
hence ∀ *x*∈*s*. *g differentiable at x within s* **by** *simp*
thus *?thesis* **unfolding** *differentiable-on-def* **by** *simp*
qed

lemma *C1-differentiable-imp-deriv-continuous-on*:
f C1-differentiable-on S ⟹ *continuous-on S* (*deriv f*)
using *C1-differentiable-on-eq field-derivative-eq-vector-derivative* **by** *auto*

lemma *deriv-shift*:
assumes *f field-differentiable at* (*x*+*a*)
shows *deriv f* (*x*+*a*) = *deriv* (λ*s*. *f* (*x*+*s*)) *a*
proof −
have (*f has-field-derivative deriv f* (*x*+*a*)) (*at* (*x*+*a*))
using *DERIV-deriv-iff-field-differentiable assms*
by *force*
hence ((λ*s*. *f* (*x*+*s*)) *has-field-derivative deriv f* (*x*+*a*)) (*at a*)

```
    using DERIV-at-within-shift has-field-derivative-at-within by blast
  moreover have ((λs. f (x+s)) has-field-derivative deriv (λs. f (x+s)) a) (at a)
    using DERIV-imp-deriv calculation by fastforce
  ultimately show ?thesis using DERIV-unique by force
qed

lemma piecewise-differentiable-on-cong:
  assumes f piecewise-differentiable-on i
    and ⋀x. x ∈ i ⟹ f x = g x
  shows g piecewise-differentiable-on i
proof −
  have continuous-on i g
   using continuous-on-cong-simp assms piecewise-differentiable-on-imp-continuous-on
by force
  moreover have ∃ S. finite S ∧ (∀ x ∈ i − S. g differentiable (at x within i))
  proof −
    from assms piecewise-differentiable-on-def
    obtain S where fin: finite S and ∀ x ∈ i − S. f differentiable (at x within i)
by metis
    hence ⋀x. x ∈ i − S ⟹ f differentiable (at x within i) by simp
    hence ⋀x. x ∈ i − S ⟹ g differentiable (at x within i)
      using has-derivative-transform assms by (metis DiffD1 differentiable-def)
    with fin show ?thesis by blast
  qed
  ultimately show ?thesis unfolding piecewise-differentiable-on-def by simp
qed

lemma differentiable-piecewise:
  assumes continuous-on i f
    and f differentiable-on i
  shows f piecewise-differentiable-on i
 unfolding piecewise-differentiable-on-def using assms differentiable-onD by auto

lemma piecewise-differentiable-scaleR:
  assumes f piecewise-differentiable-on S
  shows (λx. a *R f x) piecewise-differentiable-on S
proof −
  from assms obtain T where fin: finite T ⋀x. x ∈ S − T ⟹ f differentiable
at x within S
    unfolding piecewise-differentiable-on-def by blast
  hence ⋀x. x ∈ S − T ⟹ (λx. a *R f x) differentiable at x within S
    using differentiable-scaleR by simp
  moreover have continuous-on S (λx. a *R f x)
   using assms continuous-on-scaleR continuous-on-const piecewise-differentiable-on-def
by blast
  ultimately show (λx. a *R f x) piecewise-differentiable-on S
    using fin piecewise-differentiable-on-def by blast
qed
```

lemma *differentiable-on-piecewise-compose*:
assumes *f piecewise-differentiable-on S*
and *g differentiable-on f ' S*
shows *g ∘ f piecewise-differentiable-on S*
proof −
from *assms* **obtain** *T* **where** *fin*: *finite T* $\bigwedge x.\ x \in S - T \Longrightarrow$ *f differentiable at x within S*
unfolding *piecewise-differentiable-on-def* **by** *blast*
hence $\bigwedge x.\ x \in S - T \Longrightarrow$ *g ∘ f differentiable at x within S*
by (*meson DiffD1 assms differentiable-chain-within differentiable-onD image-eqI*)
hence $\exists T.$ *finite T* $\wedge$ $(\forall x{\in}S{-}T.$ *g ∘ f differentiable at x within S*) **using** *fin*
by *blast*
moreover have *continuous-on S* (*g ∘ f*)
using *assms continuous-on-compose differentiable-imp-continuous-on*
unfolding *piecewise-differentiable-on-def* **by** *blast*
ultimately show *?thesis*
unfolding *piecewise-differentiable-on-def* **by** *force*
qed

lemma *MVT-order-free*:
fixes *r h* :: *real*
defines $I \equiv \{r..r{+}h\} \cup \{r{+}h..r\}$
assumes *continuous-on I f* **and** *f differentiable-on interior I*
obtains *t* **where** $t \in \{0{<}..{<}1\}$ **and** *f* $(r{+}h) - f\ r = h * deriv\ f\ (r + t{*}h)$
proof −
consider (*h-pos*) $h > 0$ | (*h-0*) $h = 0$ | (*h-neg*) $h < 0$ **by** *force*
thus *?thesis*
proof *cases*
case *h-pos*
hence $r < r{+}h$ $I = \{r..r{+}h\}$ **unfolding** *I-def* **by** *simp-all*
moreover hence *interior I* $= \{r{<}..{<}r{+}h\}$ **by** *simp*
moreover hence $\bigwedge x.\ [\![r < x;\ x < r{+}h]\!] \Longrightarrow$ *f differentiable* (*at x*)
using *assms* **by** (*simp add*: *differentiable-on-eq-differentiable-at*)
ultimately obtain *z* **where** $r < z \wedge z < r{+}h \wedge f\ (r{+}h) - f\ r = h * deriv\ f\ z$
using *MVT assms* **by** (*smt* (*verit*) *DERIV-imp-deriv*)
moreover hence $(z{-}r)\ /\ h \in \{0{<}..{<}1\}$ **by** *simp*
moreover have $z = r + (z{-}r)/h * h$ **using** *h-pos* **by** *simp*
ultimately show *?thesis* **using** *that* **by** *presburger*
next
case *h-0*
have $1/2 \in \{0{::}real{<}..{<}1\}$ **by** *simp*
moreover have *f* $(r{+}h) - f\ r = 0$ **using** *h-0* **by** *simp*
moreover have $h * deriv\ f\ (r + (1/2){*}h) = 0$ **using** *h-0* **by** *simp*
ultimately show *?thesis* **using** *that* **by** *presburger*
next case *h-neg*
hence $r{+}h < r$ $I = \{r{+}h..r\}$ **unfolding** *I-def* **by** *simp-all*
moreover hence *interior I* $= \{r{+}h{<}..{<}r\}$ **by** *simp*
moreover hence $\bigwedge x.\ [\![r{+}h < x;\ x < r]\!] \Longrightarrow$ *f differentiable* (*at x*)

```
    using assms by (simp add: differentiable-on-eq-differentiable-at)
   ultimately obtain z where r+h < z ∧ z < r ∧ f r − f (r+h) = −h ∗ deriv
f z
    using MVT assms by (smt (verit) DERIV-imp-deriv)
   moreover hence (z−r) / h ∈ {0<..<1} by (simp add: neg-less-divide-eq)
   moreover have z = r + (z−r)/h ∗ h using h-neg by simp
   ultimately show ?thesis using that mult-minus-left by fastforce
  qed
qed

lemma integral-combine2:
  fixes f :: real ⇒ 'a::banach
  assumes a ≤ c c ≤ b
    and f integrable-on {a..c} f integrable-on {c..b}
  shows integral {a..c} f + integral {c..b} f = integral {a..b} f
  apply (rule integral-unique[THEN sym])
  apply (rule has-integral-combine[of a c b], simp-all add: assms)
  using has-integral-integral assms by auto

lemma has-integral-null-interval: fixes a b :: real and f::real ⇒ real assumes a
≥ b
  shows (f has-integral 0) {a..b}
  using assms content-real-eq-0 by blast

lemma has-integral-interval-reverse: fixes f :: real ⇒ real and a b :: real
  assumes a ≤ b
    and continuous-on {a..b} f
  shows ((λx. f (a+b−x)) has-integral (integral {a..b} f)) {a..b}
proof −
  let ?g = λx. a + b − x
  let ?g' = λx. −1
  have g-C0: continuous-on {a..b} ?g using continuous-on-op-minus by simp
  have Dg-g': ⋀x. x∈{a..b} ⟹ (?g has-field-derivative ?g' x) (at x within {a..b})
    by (auto intro!: derivative-eq-intros)
  show ?thesis
    using has-integral-substitution-general
      [of {} a b ?g a b f, simplified, OF assms g-C0 Dg-g', simplified]
   apply (simp add: has-integral-null-interval[OF assms(1), THEN integral-unique])
    by (simp add: has-integral-neg-iff)
qed

lemma FTC-real-deriv-has-integral:
  fixes F :: real ⇒ real
  assumes a ≤ b
    and F piecewise-differentiable-on {a<..<b}
    and continuous-on {a..b} F
  shows (deriv F has-integral F b − F a) {a..b}
proof −
  obtain S where fin: finite S and
```

```
    diff: ⋀x. x ∈ {a<..<b} − S ⟹ F differentiable at x within {a<..<b} − S
    using assms unfolding piecewise-differentiable-on-def
    by (meson Diff-subset differentiable-within-subset)
  hence ⋀x. x ∈ {a<..<b} − S ⟹ (F has-real-derivative deriv F x) (at x)
  proof −
    fix x assume x-in: x ∈ {a<..<b} − S
    have open ({a<..<b} − S)
      using fin finite-imp-closed by (metis open-Diff open-greaterThanLessThan)
    hence at x within {a<..<b} − S = at x by (meson x-in at-within-open)
    hence F differentiable at x using diff x-in by (smt (verit))
    thus (F has-real-derivative deriv F x) (at x)
      using DERIV-deriv-iff-real-differentiable by simp
  qed
  thus ?thesis
    by (intro fundamental-theorem-of-calculus-interior-strong[where S=S];
        simp add: assms fin has-real-derivative-iff-has-vector-derivative)
qed

lemma integrable-spike-cong:
  assumes negligible S ⋀x. x ∈ T − S ⟹ g x = f x
  shows f integrable-on T ⟷ g integrable-on T
  using integrable-spike assms by force

lemma has-integral-powr2-from-0:
  fixes a c :: real
  assumes a-pos: a > 0 and a-neq-1: a ≠ 1 and c-nneg: c ≥ 0
  shows ((λx. a.^x) has-integral ((a.^c − 1) / (ln a))) {0..c}
proof −
  have ((λx. a.^x) has-integral ((a.^c)/(ln a) − (a.^0)/(ln a))) {0..c}
  proof (rule fundamental-theorem-of-calculus[OF c-nneg])
    fix x::real
    assume x ∈ {0..c}
    show ((λy. a.^y / ln a) has-vector-derivative a.^x) (at x within {0..c})
      apply (insert has-real-derivative-powr2[OF a-pos, of x])
      apply (drule DERIV-cdivide[where c = ln a], simp add: assms)
     apply (rule has-vector-derivative-within-subset[where S=UNIV and T={0..c}],
auto)
      by (rule iffD1[OF has-real-derivative-iff-has-vector-derivative])
  qed
  thus ?thesis
    using assms powr-zero-eq-one by (simp add: field-simps)
qed

lemma integrable-on-powr2-from-0:
  fixes a c :: real
  assumes a-pos: a > 0 and a-neq-1: a ≠ 1 and c-nneg: c ≥ 0
  shows (λx. a.^x) integrable-on {0..c}
 using has-integral-powr2-from-0[OF assms] unfolding integrable-on-def by blast
```

lemma *integrable-on-powr2-from-0-general*:
fixes *a c* :: *real*
assumes *a-pos*: $a > 0$ **and** *c-nneg*: $c \geq 0$
shows $(\lambda x.\ a.\hat{}x)$ *integrable-on* $\{0..c\}$
proof (*cases* $a = 1$)
case *True*
thus *?thesis*
using *has-integral-const-real* **by** *auto*
next
case *False*
thus *?thesis*
using *has-integral-powr2-from-0 False assms* **by** *auto*
qed

lemma *has-bochner-integral-power*:
fixes *a b* :: *real* **and** *k* :: *nat*
assumes $a \leq b$
shows *has-bochner-integral lborel* $(\lambda x.\ x\hat{}k * \textit{indicator}\ \{a..b\}\ x)$ $((b\hat{}(k+1) - a\hat{}(k+1)) / (k+1))$
proof −
have $\bigwedge x.\ ((\lambda x.\ x\hat{}(k+1) / (k+1))\ \textit{has-real-derivative}\ x\hat{}k)\ (\textit{at}\ x)$
using *DERIV-pow* **by** (*intro derivative-eq-intros*) *auto*
hence *has-bochner-integral lborel* $(\lambda x.\ x\hat{}k * \textit{indicator}\ \{a..b\}\ x)$ $(b\hat{}(k+1)/(k+1) - a\hat{}(k+1)/(k+1))$
by (*intro has-bochner-integral-FTC-Icc-real*; *simp add*: *assms*)
thus *?thesis* **by** (*simp add*: *diff-divide-distrib*)
qed

corollary *integrable-power*: $(a{::}\textit{real}) \leq b \Longrightarrow \textit{integrable lborel}\ (\lambda x.\ x\hat{}k * \textit{indicator}\ \{a..b\}\ x)$
using *has-bochner-integral-power integrable.intros* **by** *blast*

lemma *has-integral-set-integral-real*:
fixes $f{::}'a{::}\textit{euclidean-space} \Rightarrow \textit{real}$ **and** $A :: 'a\ \textit{set}$
assumes *f*: *set-integrable lborel A f*
shows (*f has-integral* (*set-lebesgue-integral lborel A f*)) *A*
using *assms has-integral-integral-real*[**where** $f{=}\lambda x.\ \textit{indicat-real}\ A\ x * f\ x$]
unfolding *set-integrable-def set-lebesgue-integral-def*
by *simp* (*smt* (*verit*, *ccfv-SIG*) *has-integral-cong has-integral-restrict-UNIV indicator-times-eq-if*)

lemma *set-borel-measurable-lborel*:
set-borel-measurable lborel A f $\longleftrightarrow$ *set-borel-measurable borel A f*
unfolding *set-borel-measurable-def* **by** *simp*

lemma *restrict-space-whole*[*simp*]: *restrict-space M* (*space M*) = *M*
unfolding *restrict-space-def* **by** (*simp add*: *measure-of-of-measure*)

lemma *deriv-measurable-real*:
fixes $f :: real \Rightarrow real$
assumes *f differentiable-on S open S f* $\in$ *borel-measurable borel*
shows *set-borel-measurable borel S* (*deriv f*)
proof −
have $\bigwedge x.\ x \in S \Longrightarrow deriv\ f\ x = lim\ (\lambda i.\ (f\ (x + 1\ /\ Suc\ i) - f\ x)\ /\ (1\ /\ Suc\ i))$
proof −
fix *x* **assume** *x-S*: $x \in S$
hence *f field-differentiable* (*at x within S*)
using *differentiable-on-eq-field-differentiable-real assms* **by** *simp*
hence (*f has-field-derivative deriv f x*) (*at x*)
using *assms DERIV-deriv-iff-field-differentiable x-S at-within-open* **by** *force*
hence $(\lambda h.\ (f\ (x+h) - f\ x)\ /\ h) -0\to deriv\ f\ x$ **using** *DERIV-def* **by** *auto*
hence $\forall\, d.\ (\forall\, i.\ d\ i \in UNIV - \{0::real\}) \longrightarrow d \longrightarrow 0 \longrightarrow$
$((\lambda h.\ (f\ (x+h) - f\ x)\ /\ h) \circ d) \longrightarrow deriv\ f\ x$
using *tendsto-at-iff-sequentially* **by** *blast*
moreover have $\forall\, i.\ 1\ /\ Suc\ i \in UNIV - \{0::real\}$ **by** *simp*
moreover have $(\lambda i.\ 1\ /\ Suc\ i) \longrightarrow 0$ **using** *LIMSEQ-Suc lim-const-over-n* **by** *blast*
ultimately have $((\lambda h.\ (f\ (x + h) - f\ x)\ /\ h) \circ (\lambda i.\ 1\ /\ Suc\ i)) \longrightarrow deriv\ f\ x$ **by** *auto*
thus $deriv\ f\ x = lim\ (\lambda i.\ (f\ (x + 1\ /\ Suc\ i) - f\ x)\ /\ (1\ /\ Suc\ i))$
unfolding *comp-def* **by** (*simp add*: *limI*)
qed
moreover have $(\lambda x.\ indicator\ S\ x * lim\ (\lambda i.\ (f\ (x + 1\ /\ Suc\ i) - f\ x)\ /\ (1\ /\ Suc\ i)))$
$\in$ *borel-measurable borel*
using *assms* **by** (*measurable*, *simp*, *measurable*)
ultimately show *?thesis*
unfolding *set-borel-measurable-def measurable-cong*
by *simp* (*smt* (*verit*) *indicator-simps*(*2*) *measurable-cong mult-eq-0-iff*)
qed

lemma *piecewise-differentiable-on-deriv-measurable-real*:
fixes $f :: real \Rightarrow real$
assumes *f piecewise-differentiable-on S open S f* $\in$ *borel-measurable borel*
shows *set-borel-measurable borel S* (*deriv f*)
proof −
from *assms* **obtain** *T* **where** *fin*: *finite T* **and**
diff: $\bigwedge x.\ x \in S - T \Longrightarrow$ *f differentiable at x within S*
unfolding *piecewise-differentiable-on-def* **by** *blast*
with *assms* **have** *open* (*S* − *T*) **using** *finite-imp-closed* **by** *blast*
moreover hence *f differentiable-on* (*S* − *T*)
unfolding *differentiable-on-def* **using** *assms* **by** (*metis Diff-iff at-within-open diff*)
ultimately have *set-borel-measurable borel* (*S* − *T*) (*deriv f*)
by (*intro deriv-measurable-real*; *simp add*: *assms*)

thus *?thesis*
unfolding *set-borel-measurable-def* **apply** *simp*
apply (*rule measurable-discrete-difference*
[**where** $X{=}T$ **and** $f{=}\lambda x.$ *indicat-real* $(S - T)$ $x * deriv\ f\ x$], *simp-all*)
using *fin uncountable-infinite* **apply** *blast*
by (*simp add*: *indicator-diff*)
qed

lemma *borel-measurable-antimono*:
fixes f :: *real* ⇒ *real*
shows *antimono* $f \Longrightarrow f \in$ *borel-measurable borel*
using *borel-measurable-mono* **by** (*smt* (*verit*, *del-insts*) *borel-measurable-uminus-eq monotone-on-def*)

lemma *set-borel-measurable-restrict-space-iff*:
fixes f :: *'a* ⇒ *'b*::*real-normed-vector*
assumes Ω[*measurable*, *simp*]: $\Omega \cap$ *space* $M \in$ *sets* M
shows $f \in$ *borel-measurable* (*restrict-space* M Ω) ⟷ *set-borel-measurable* M Ω f
using *assms borel-measurable-restrict-space-iff set-borel-measurable-def* **by** *blast*

lemma *set-integrable-restrict-space-iff*:
fixes f :: *'a* ⇒ *'b*::{*banach*, *second-countable-topology*}
assumes $A \in$ *sets* M
shows *set-integrable* M A f ⟷ *integrable* (*restrict-space* M A) f
unfolding *set-integrable-def* **using** *assms*
by (*rewrite integrable-restrict-space*; *simp*)

lemma *set-lebesgue-integral-restrict-space*:
fixes f :: *'a* ⇒ *'b*::{*banach*, *second-countable-topology*}
assumes $A \in$ *sets* M
shows *set-lebesgue-integral* M A f = *integral*^{L} (*restrict-space* M A) f
unfolding *set-lebesgue-integral-def* **using** *assms integral-restrict-space*
by (*metis* (*mono-tags*) *sets.Int-space-eq2*)

lemma *distr-borel-lborel*: *distr M borel f* = *distr M lborel f*
by (*metis distr-cong sets-lborel*)

lemma *AE-translation*:
assumes *AE x in lborel. P x* **shows** *AE x in lborel. P* $(a{+}x)$
proof −
from *assms* **obtain** N **where** P: $\bigwedge x.\ x \in$ *space lborel* $- N \Longrightarrow P\ x$ **and** *null*: $N \in$ *null-sets lborel*
using *AE-E3* **by** *blast*
hence $\{y.\ a{+}y \in N\} \in$ *null-sets lborel*
using *null-sets-translation*[*of* N $-a$, *simplified*] **by** (*simp add*: *add.commute*)
moreover have $\bigwedge y.\ y \in$ *space lborel* $- \{y.\ a{+}y \in N\} \Longrightarrow P\ (a{+}y)$ **using** P **by** *force*
ultimately show *AE y in lborel. P* $(a{+}y)$

by (*smt* (*verit*, *del-insts*) *Diff-iff eventually-ae-filter mem-Collect-eq subsetI*)
qed

lemma *set-AE-translation*:
assumes *AE x∈S in lborel. P x* **shows** *AE x ∈ plus* (−*a*) ‘ *S in lborel. P* (*a*+*x*)
proof −
have *AE x in lborel. a*+*x* ∈ *S* ⟶ *P* (*a*+*x*) **using** *assms* **by** (*rule AE-translation*)
moreover have ⋀*x. a*+*x* ∈ *S* ⟷ *x* ∈ *plus* (−*a*) ‘ *S* **by** *force*
ultimately show *?thesis* **by** *simp*
qed

lemma *AE-scale-measure-iff*:
assumes *r* > *0*
shows (*AE x in* (*scale-measure r M*). *P x*) ⟷ (*AE x in M*. *P x*)
unfolding *ae-filter-def null-sets-def*
apply (*rewrite space-scale-measure*, *simp*)
using *assms* **by** (*smt* (*verit*) *Collect-cong not-gr-zero*)

lemma *nn-set-integral-cong2*:
assumes *AE x∈A in M*. *f x* = *g x*
shows (∫⁺*x*∈*A*. *f x* ∂*M*) = (∫⁺*x*∈*A*. *g x* ∂*M*)
proof −
{ fix *x*
assume *x* ∈ *space M*
have (*x* ∈ *A* ⟶ *f x* = *g x*) ⟶ *f x* ∗ *indicator A x* = *g x* ∗ *indicator A x* **by** *force* **}**
hence *AE x in M*. (*x* ∈ *A* ⟶ *f x* = *g x*) ⟶ *f x* ∗ *indicator A x* = *g x* ∗ *indicator A x*
by (*rule AE-I2*)
hence *AE x in M*. *f x* ∗ *indicator A x* = *g x* ∗ *indicator A x* **using** *assms AE-mp* **by** *auto*
thus *?thesis* **by** (*rule nn-integral-cong-AE*)
qed

lemma *set-lebesgue-integral-cong-AE2*:
assumes [*measurable*]: *A* ∈ *sets M set-borel-measurable M A f set-borel-measurable M A g*
assumes *AE x* ∈ *A in M*. *f x* = *g x*
shows (*LINT x*:*A*|*M*. *f x*) = (*LINT x*:*A*|*M*. *g x*)
proof −
let *?fA* = λ*x*. *indicator A x* ∗$_R$ *f x* **and** *?gA* = λ*x*. *indicator A x* ∗$_R$ *g x*
have *?fA* ∈ *borel-measurable M ?gA* ∈ *borel-measurable M*
using *assms* **unfolding** *set-borel-measurable-def* **by** *simp-all*
moreover have *AE x* ∈ *A in M*. *?fA x* = *?gA x* **using** *assms* **by** *simp*
ultimately have (*LINT x*:*A*|*M*. *?fA x*) = (*LINT x*:*A*|*M*. *?gA x*)
by (*intro set-lebesgue-integral-cong-AE*; *simp*)
moreover have (*LINT x*:*A*|*M*. *f x*) = (*LINT x*:*A*|*M*. *?fA x*) (*LINT x*:*A*|*M*. *g x*) = (*LINT x*:*A*|*M*. *?gA x*)
unfolding *set-lebesgue-integral-def*

by (*metis indicator-scaleR-eq-if*)+
ultimately show *?thesis* **by** *simp*
qed

proposition *set-nn-integral-eq-set-integral*:
assumes *AE x∈A in M. 0 ≤ f x set-integrable M A f*
shows $(\int^+ x{\in}A.\ f\ x\ \partial M) = (\int x{\in}A.\ f\ x\ \partial M)$
proof −
have $(\int^+ x{\in}A.\ f\ x\ \partial M) = \int^+ x.\ ennreal\ (f\ x * indicator\ A\ x)\ \partial M$
using *nn-integral-set-ennreal* **by** *blast*
also have $\ldots = \int x.\ f\ x * indicator\ A\ x\ \partial M$
using *assms* **unfolding** *set-integrable-def*
by (*rewrite nn-integral-eq-integral*; *force simp add*: *mult.commute*)
also have $\ldots = (\int x{\in}A.\ f\ x\ \partial M)$ **unfolding** *set-lebesgue-integral-def* **by** (*simp add*: *mult.commute*)
finally show *?thesis* **.**
qed

proposition *nn-integral-disjoint-family-on-finite*:
assumes [*measurable*]: $f \in borel\text{-}measurable\ M$ $\bigwedge n{::}nat.\ n \in S \Longrightarrow B\ n \in sets\ M$
and *disjoint-family-on B S finite S*
shows $(\int^+ x \in (\bigcup n{\in}S.\ B\ n).\ f\ x\ \partial M) = (\sum n{\in}S.\ (\int^+ x \in B\ n.\ f\ x\ \partial M))$
proof −
let *?A* = $\lambda n{::}nat.$ *if n ∈ S then B n else* {}
have $\bigwedge n{::}nat.$ *?A n ∈ sets M* **by** *simp*
moreover have *disjoint-family ?A*
unfolding *disjoint-family-on-def*
proof −
{ fix *m n* :: *nat*
assume *m ≠ n*
hence (*if m ∈ S then B m else* {}) ∩ (*if n ∈ S then B n else* {}) = {}
apply *simp*
using *assms* **unfolding** *disjoint-family-on-def* **by** *blast* **}**
thus $\forall\, m{::}nat{\in}UNIV.\ \forall\, n{::}nat{\in}UNIV.\ m \neq n \longrightarrow$
(*if m ∈ S then B m else* {}) ∩ (*if n ∈ S then B n else* {}) = {}
by *blast*
qed
ultimately have *set-nn-integral M* (⋃ (*range ?A*)) *f* = $(\sum n.$ *set-nn-integral M* (*?A n*) *f*)
by (*rewrite nn-integral-disjoint-family*; *simp*)
moreover have *set-nn-integral M* (⋃ (*range ?A*)) *f* = $(\int^+ x \in (\bigcup n{\in}S.\ B\ n).$ *f x ∂M*)
proof −
have ⋃ (*range ?A*) = $(\bigcup n{\in}S.\ B\ n)$ **by** *simp*
thus *?thesis* **by** *simp*
qed
moreover have $(\sum n.$ *set-nn-integral M* (*?A n*) *f*) = $(\sum n{\in}S.$ *set-nn-integral M* (*B n*) *f*)

by (*rewrite suminf-finite*[*of S*]; *simp add*: *assms*)
ultimately show *?thesis* **by** *simp*
qed

lemma *nn-integral-distr-set*:
assumes $T \in$ *measurable M M'* **and** $f \in$ *borel-measurable* (*distr M M' T*)
and $A \in$ *sets M'* **and** $\bigwedge x.\ x \in$ *space M* $\Longrightarrow T\ x \in A$
shows $integral^N$ (*distr M M' T*) $f =$ *set-nn-integral* (*distr M M' T*) *A f*
proof −
have $integral^N$ (*distr M M' T*) $f = (\int^+ x \in (space\ M').\ f\ x\ \partial(distr\ M\ M'\ T))$
by (*rewrite nn-set-integral-space*[*THEN sym*], *simp*)
also have ... $= (\int^+ x \in A.\ f\ x\ \partial(distr\ M\ M'\ T))$
proof −
have [*simp*]: *sym-diff* (*space M'*) *A* = *space M'* − *A*
using *assms* **by** (*metis Diff-mono sets.sets-into-space sup.orderE*)
show *?thesis*
apply (*rule nn-integral-null-delta*; *simp add*: *assms*)
unfolding *null-sets-def* **using** *assms*
apply (*simp*, *rewrite emeasure-distr*; *simp*)
unfolding *vimage-def* **using** *emeasure-empty*
by (*metis* (*no-types*, *lifting*) *Diff-disjoint disjoint-iff-not-equal mem-Collect-eq*)
qed
finally show *?thesis* **.**
qed

lemma *measure-eqI-Ioc*:
fixes *M N* :: *real measure*
assumes *sets*: *sets M* = *sets borel sets N* = *borel*
assumes *fin*: $\bigwedge a\ b.\ a \leq b \Longrightarrow$ *emeasure M* $\{a<..b\} < \infty$
assumes *eq*: $\bigwedge a\ b.\ a \leq b \Longrightarrow$ *emeasure M* $\{a<..b\}$ = *emeasure N* $\{a<..b\}$
shows *M* = *N*
proof (*rule measure-eqI-generator-eq-countable*)
let *?Ioc* = $\lambda(a{::}real, b{::}real).\ \{a<..b\}$ **let** *?E* = *range ?Ioc*
show *Int-stable ?E* **using** *Int-stable-def Int-greaterThanAtMost* **by** *force*
show *?E* ⊆ *Pow UNIV sets M* = *sigma-sets UNIV ?E sets N* = *sigma-sets UNIV ?E*
unfolding *sets* **by** (*auto simp add*: *borel-sigma-sets-Ioc*)
show $\bigwedge I.\ I \in$ *?E* $\Longrightarrow$ *emeasure M I* = *emeasure N I*
proof −
fix *I* **assume** *I* ∈ *?E*
then obtain *a b* **where** *I* = $\{a<..b\}$ **by** *auto*
thus *emeasure M I* = *emeasure N I* **by** (*smt* (*verit*, *best*) *eq greaterThanAtMost-empty*)
qed
show *?Ioc* ‘ (*Rats* × *Rats*) ⊆ *?E* $(\bigcup i \in (Rats \times Rats).\ ?Ioc\ i) = UNIV$
using *Rats-no-bot-less Rats-no-top-le* **by** *auto*
show *countable* (*?Ioc* ‘ (*Rats* × *Rats*)) **using** *countable-rat* **by** *blast*
show $\bigwedge I.\ I \in$ *?Ioc* ‘ (*Rats* × *Rats*) $\Longrightarrow$ *emeasure M I* $\neq \infty$

proof −
fix I **assume** $I \in$ *?Ioc* ‘ $(Rats \times Rats)$
then obtain a b **where** $(a,b) \in (Rats \times Rats)$ $I = \{a<..b\}$ **by** *blast*
thus *emeasure M I* $\neq \infty$ **by** (*smt* (*verit*, *best*) *Ioc-inj fin order.strict-implies-not-eq*)
qed
qed

lemma (**in** *finite-measure*) *distributed-measure*:
assumes *distributed M N X f*
and $\bigwedge x.\ x \in space\ N \Longrightarrow f\ x \geq 0$
and $A \in sets\ N$
shows *measure M* $(X -‘ A \cap space\ M) = (\int x.\ indicator\ A\ x * f\ x\ \partial N)$
proof −
have [*simp*]: $(\lambda x.\ indicat\text{-}real\ A\ x * f\ x) \in borel\text{-}measurable\ N$
using *assms* **apply** (*measurable*; *simp?*)
using *distributed-real-measurable assms* **by** *force*
have *emeasure M* $(X -‘ A \cap space\ M) = (\int^+ x{\in}A.\ ennreal\ (f\ x)\ \partial N)$
by (*rule distributed-emeasure*; *simp add*: *assms*)
moreover have *enn2real* $(\int^+ x{\in}A.\ ennreal\ (f\ x)\ \partial N) = \int x.\ indicator\ A\ x * f\ x\ \partial N$
apply (*rewrite enn2real-nn-integral-eq-integral*
[**where** $f{=}\lambda x.\ ennreal\ (indicator\ A\ x * f\ x)$, *THEN sym*]; (*simp add*: *assms*)*?*)
using *distributed-emeasure assms*
by (*smt* (*verit*) *emeasure-finite indicator-mult-ennreal mult.commute*
nn-integral-cong top.not-eq-extremum)
ultimately show *?thesis* **using** *measure-def* **by** *metis*
qed

lemma *set-integrable-const*[*simp*]:
$A \in sets\ M \Longrightarrow emeasure\ M\ A < \infty \Longrightarrow set\text{-}integrable\ M\ A\ (\lambda\text{-}.\ c)$
using *has-bochner-integral-indicator* **unfolding** *set-integrable-def* **by** *simp*

lemma *set-integral-const*[*simp*]:
$A \in sets\ M \Longrightarrow emeasure\ M\ A < \infty \Longrightarrow set\text{-}lebesgue\text{-}integral\ M\ A\ (\lambda\text{-}.\ c) = measure\ M\ A *_R c$
unfolding *set-lebesgue-integral-def* **using** *has-bochner-integral-indicator* **by** *force*

lemma *set-integral-empty-0*[*simp*]: *set-lebesgue-integral M* $\{\}$ $f = 0$
unfolding *set-lebesgue-integral-def* **by** *simp*

lemma *set-integral-nonneg*[*simp*]:
fixes $f :: 'a \Rightarrow real$ **and** $A :: 'a\ set$
shows $(\bigwedge x.\ x \in A \Longrightarrow 0 \leq f\ x) \Longrightarrow 0 \leq set\text{-}lebesgue\text{-}integral\ M\ A\ f$
unfolding *set-lebesgue-integral-def* **by** (*simp add*: *indicator-times-eq-if*(*1*))

lemma
fixes $f :: 'a \Rightarrow 'b{::}\{banach, second\text{-}countable\text{-}topology\}$ **and** $w :: 'a \Rightarrow real$
assumes $A \in sets\ M$ *set-borel-measurable M A f*

```
    ⋀i. set-borel-measurable M A (s i) set-integrable M A w
  assumes lim: AE x∈A in M. (λi. s i x) ⟶ f x
  assumes bound: ⋀i::nat. AE x∈A in M. norm (s i x) ≤ w x
  shows set-integrable-dominated-convergence: set-integrable M A f
    and set-integrable-dominated-convergence2: ⋀i. set-integrable M A (s i)
    and set-integral-dominated-convergence:
    (λi. set-lebesgue-integral M A (s i)) ⟶ set-lebesgue-integral M A f
proof −
  have (λx. indicator A x *R f x) ∈ borel-measurable M and
    ⋀i. (λx. indicator A x *R s i x) ∈ borel-measurable M and
    integrable M (λx. indicator A x *R w x)
    using assms unfolding set-borel-measurable-def set-integrable-def by simp-all
  moreover have AE x in M. (λi. indicator A x *R s i x) ⟶ indicator A x
*R f x
  proof −
    obtain N where N-null: N ∈ null-sets M and
      si-f: ⋀x. x ∈ space M − N ⟹ x ∈ A ⟶ (λi. s i x) ⟶ f x
      using lim AE-E3 by (smt (verit))
    hence ⋀x. x ∈ space M − N ⟹ (λi. indicator A x *R s i x) ⟶ indicator
A x *R f x
    proof −
      fix x assume asm: x ∈ space M − N
      thus (λi. indicator A x *R s i x) ⟶ indicator A x *R f x
      proof (cases ‹x ∈ A›)
        case True
        with si-f asm show ?thesis by simp
      next
        case False
        thus ?thesis by simp
      qed
    qed
    thus ?thesis by (smt (verit) AE-I' DiffI N-null mem-Collect-eq subsetI)
  qed
  moreover have ⋀i. AE x in M. norm (indicator A x *R s i x) ≤ indicator A x
*R w x
  proof −
    fix i
    from bound obtain N where N-null: N ∈ null-sets M and
      ⋀x. x ∈ space M − N ⟹ x ∈ A ⟶ norm (s i x) ≤ w x
      using AE-E3 by (smt (verit))
    hence ⋀x. x ∈ space M − N ⟹ norm (indicator A x *R s i x) ≤ indicator
A x *R w x
      by (simp add: indicator-scaleR-eq-if)
    with N-null show AE x in M. norm (indicator A x *R s i x) ≤ indicator A x
*R w x
      by (smt (verit) DiffI eventually-ae-filter mem-Collect-eq subsetI)
  qed
  ultimately show set-integrable M A f ⋀i. set-integrable M A (s i)
    (λi. set-lebesgue-integral M A (s i)) ⟶ set-lebesgue-integral M A f
```

unfolding *set-integrable-def set-lebesgue-integral-def*
by (*rule integrable-dominated-convergence*, *rule integrable-dominated-convergence2*, *rule integral-dominated-convergence*)
qed

lemma *absolutely-integrable-on-iff-set-integrable*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow real$
assumes $f \in borel\text{-}measurable\ lborel$
and $S \in sets\ lborel$
shows $set\text{-}integrable\ lborel\ S\ f \longleftrightarrow f\ absolutely\text{-}integrable\text{-}on\ S$
unfolding *set-integrable-def* **apply** (*simp*, *rewrite integrable-completion*[*THEN sym*])
apply *measurable* **using** *assms* **by** *simp-all*

corollary *integrable-on-iff-set-integrable-nonneg*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow real$
assumes $\bigwedge x.\ x \in S \Longrightarrow f\ x \geq 0\ f \in borel\text{-}measurable\ lborel$
and $S \in sets\ lborel$
shows $set\text{-}integrable\ lborel\ S\ f \longleftrightarrow f\ integrable\text{-}on\ S$
using *absolutely-integrable-on-iff-set-integrable assms*
by (*metis absolutely-integrable-on-iff-nonneg*)

lemma *integrable-on-iff-set-integrable-nonneg-AE*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow real$
assumes $AE\ x{\in}S\ in\ lborel.\ f\ x \geq 0\ f \in borel\text{-}measurable\ lborel$
and $S \in sets\ lborel$
shows $set\text{-}integrable\ lborel\ S\ f \longleftrightarrow f\ integrable\text{-}on\ S$
proof −
from *assms* **obtain** N **where** *nonneg*: $\bigwedge x.\ x \in S - N \Longrightarrow f\ x \geq 0$ **and** *null*: $N \in null\text{-}sets\ lborel$
by (*smt* (*verit*, *ccfv-threshold*) *AE-E3 Diff-iff UNIV-I space-borel space-lborel*)
let $?g = \lambda x.\ if\ x \in N\ then\ 0\ else\ f\ x$
have [*simp*]: $negligible\ N$ **using** *null negligible-iff-null-sets null-sets-completionI* **by** *blast*
have $N \in sets\ lborel$ **using** *null* **by** *auto*
hence [*simp*]: $?g \in borel\text{-}measurable\ borel$ **using** *assms* **by** *force*
have $set\text{-}integrable\ lborel\ S\ f \longleftrightarrow set\text{-}integrable\ lborel\ S\ ?g$
proof −
have $AE\ x{\in}S\ in\ lborel.\ f\ x = ?g\ x$ **by** (*rule AE-I'*[*of N*], *simp-all add*: *null*, *blast*)
thus *?thesis* **using** *assms* **by** (*intro set-integrable-cong-AE*[*of f - ?g S*]; *simp*)
qed
also have $\ldots \longleftrightarrow ?g\ integrable\text{-}on\ S$
using *assms* **by** (*intro integrable-on-iff-set-integrable-nonneg*; *simp add*: *nonneg*)
also have $\ldots \longleftrightarrow f\ integrable\text{-}on\ S$ **by** (*rule integrable-spike-cong*[*of N*]; *simp*)
finally show *?thesis* **.**
qed

lemma *set-borel-integral-eq-integral-nonneg*:

fixes $f :: 'a::euclidean\text{-}space \Rightarrow real$
assumes $\bigwedge x.\ x \in S \Longrightarrow f\ x \geq 0\ f \in borel\text{-}measurable\ borel\ S \in sets\ borel$
shows $(LINT\ x : S \mid lborel.\ f\ x) = integral\ S\ f$
— Note that $0 = 0$ holds when the integral diverges.
proof (*cases* ‹*set-integrable lborel S f*›)
case *True*
thus *?thesis* **using** *set-borel-integral-eq-integral* **by** *force*
next
case *False*
hence $(LINT\ x : S \mid lborel.\ f\ x) = 0$
unfolding *set-lebesgue-integral-def set-integrable-def*
by (*rewrite not-integrable-integral-eq*; *simp*)
moreover have $integral\ S\ f = 0$
apply (*rule not-integrable-integral*)
using *False assms* **by** (*rewrite* **in** *asm integrable-on-iff-set-integrable-nonneg*; *simp*)
ultimately show *?thesis* **..**
qed

lemma *set-borel-integral-eq-integral-nonneg-AE*:
fixes $f :: 'a::euclidean\text{-}space \Rightarrow real$
assumes $AE\ x{\in}S\ in\ lborel.\ f\ x \geq 0\ f \in borel\text{-}measurable\ borel\ S \in sets\ borel$
shows $(LINT\ x : S \mid lborel.\ f\ x) = integral\ S\ f$
— Note that $0 = 0$ holds when the integral diverges.
proof (*cases* ‹*set-integrable lborel S f*›)
case *True*
thus *?thesis* **using** *set-borel-integral-eq-integral* **by** *force*
next
case *False*
hence $(LINT\ x : S \mid lborel.\ f\ x) = 0$
unfolding *set-lebesgue-integral-def set-integrable-def*
by (*rewrite not-integrable-integral-eq*; *simp*)
moreover have $integral\ S\ f = 0$
apply (*rule not-integrable-integral*)
using *False assms* **by** (*rewrite* **in** *asm integrable-on-iff-set-integrable-nonneg-AE*; *simp*)
ultimately show *?thesis* **..**
qed

2.1 Set Lebesgue Integrability on Affine Transformation

lemma *set-integrable-Icc-affine-pos-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b\ c\ t :: real$
assumes $c > 0$
shows *set-integrable lborel* $\{(a-t)/c..(b-t)/c\}$ $(\lambda x.\ f\ (t + c{*}x))$
$\longleftrightarrow$ *set-integrable lborel* $\{a..b\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Icc-affine-pos-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Icc-shift*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b\ t :: real$
shows *set-integrable lborel* $\{a-t..b-t\}$ $(\lambda x.\ f\ (t+x)) \longleftrightarrow$ *set-integrable lborel* $\{a..b\}$ f
using *set-integrable-Icc-affine-pos-iff*[**where** *c=1*] **by** *simp*

lemma *set-integrable-Ici-affine-pos-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ c\ t :: real$
assumes $c > 0$
shows *set-integrable lborel* $\{(a-t)/c..\}$ $(\lambda x.\ f\ (t + c*x))$
$\longleftrightarrow$ *set-integrable lborel* $\{a..\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Ici-affine-pos-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Ici-shift*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ t :: real$
shows *set-integrable lborel* $\{a-t..\}$ $(\lambda x.\ f\ (t+x)) \longleftrightarrow$ *set-integrable lborel* $\{a..\}$ f
using *set-integrable-Ici-affine-pos-iff*[**where** *c=1*] **by** *simp*

lemma *set-integrable-Iic-affine-pos-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $b\ c\ t :: real$
assumes $c > 0$
shows *set-integrable lborel* $\{..(b-t)/c\}$ $(\lambda x.\ f\ (t + c*x))$
$\longleftrightarrow$ *set-integrable lborel* $\{..b\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Iic-affine-pos-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Iic-shift*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $b\ t :: real$
shows *set-integrable lborel* $\{..b-t\}$ $(\lambda x.\ f\ (t+x)) \longleftrightarrow$ *set-integrable lborel* $\{..b\}$ f
using *set-integrable-Iic-affine-pos-iff*[**where** *c=1*] **by** *simp*

lemma *set-integrable-Icc-affine-neg-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b\ c\ t :: real$
assumes $c < 0$
shows *set-integrable lborel* $\{(b-t)/c..(a-t)/c\}$ $(\lambda x.\ f\ (t + c*x))$
$\longleftrightarrow$ *set-integrable lborel* $\{a..b\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Icc-affine-neg-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Icc-reverse*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b\ t :: real$
shows *set-integrable lborel* $\{t-b..t-a\}$ $(\lambda x.\ f\ (t-x)) \longleftrightarrow$ *set-integrable lborel* $\{a..b\}$ f
using *set-integrable-Icc-affine-neg-iff*[**where** *c=−1*] **by** *simp*

lemma *set-integrable-Ici-affine-neg-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $b\ c\ t :: real$
assumes $c < 0$
shows *set-integrable lborel* $\{(b-t)/c..\}$ $(\lambda x.\ f\ (t + c*x))$
$\longleftrightarrow$ *set-integrable lborel* $\{..b\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Ici-affine-neg-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Ici-reverse*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $b\ t :: real$
shows *set-integrable lborel* $\{t-b..\}$ $(\lambda x.\ f\ (t-x)) \longleftrightarrow$ *set-integrable lborel* $\{..b\}$ f
using *set-integrable-Ici-affine-neg-iff*[**where** $c=-1$] **by** *simp*

lemma *set-integrable-Iic-affine-neg-iff*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ c\ t :: real$
assumes $c < 0$
shows *set-integrable lborel* $\{..(a-t)/c\}$ $(\lambda x.\ f\ (t + c*x))$
$\longleftrightarrow$ *set-integrable lborel* $\{a..\}$ f
unfolding *set-integrable-def* **using** *assms*
apply (*rewrite indicator-Iic-affine-neg-inverse*, *simp*)
by (*rule lborel-integrable-real-affine-iff*) *simp*

corollary *set-integrable-Iic-reverse*:
fixes $f :: real \Rightarrow 'a::\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ t :: real$
shows *set-integrable lborel* $\{..t-a\}$ $(\lambda x.\ f\ (t-x)) \longleftrightarrow$ *set-integrable lborel* $\{a..\}$ f
using *set-integrable-Iic-affine-neg-iff*[**where** $c=-1$] **by** *simp*

2.2 Set Lebesgue Integral on Affine Transformation

lemma *lborel-set-integral-Icc-affine-pos*:
fixes $f :: real \Rightarrow 'a :: \{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b\ c :: real$
assumes $c > 0$
shows $(\int x{\in}\{a..b\}.\ f\ x\ \partial lborel) = c *_R (\int x{\in}\{(a-t)/c..(b-t)/c\}.\ f\ (t + c*x)\ \partial lborel)$
unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Icc-affine-pos-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** $c=c$] **by** *force*

corollary *lborel-set-integral-Icc-shift*:
fixes $f :: real \Rightarrow 'a :: \{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b :: real$
shows $(\int x{\in}\{a..b\}.\ f\ x\ \partial lborel) = (\int x{\in}\{a-t..b-t\}.\ f\ (t+x)\ \partial lborel)$
using *lborel-set-integral-Icc-affine-pos*[**where** $c=1$] **by** *simp*

lemma *lborel-set-integral-Ici-affine-pos*:
fixes $f :: real \Rightarrow 'a :: \{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ c :: real$
assumes $c > 0$
shows $(\int x{\in}\{a..\}.\ f\ x\ \partial lborel) = c *_R (\int x{\in}\{(a-t)/c..\}.\ f\ (t + c*x)\ \partial lborel)$

unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Ici-affine-pos-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** *c=c*] **by** *force*

corollary *lborel-set-integral-Ici-shift*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *a*::*real*
shows (∫ *x*∈{*a*..}. *f x* ∂*lborel*) = (∫ *x*∈{*a*−*t*..}. *f* (*t*+*x*) ∂*lborel*)
using *lborel-set-integral-Ici-affine-pos*[**where** *c=1*] **by** *simp*

lemma *lborel-set-integral-Iic-affine-pos*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *b c* :: *real*
assumes *c* > *0*
shows (∫ *x*∈{..*b*}. *f x* ∂*lborel*) = *c* ∗*R* (∫ *x*∈{..(*b*−*t*)/*c*}. *f* (*t* + *c*∗*x*) ∂*lborel*)
unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Iic-affine-pos-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** *c=c*] **by** *force*

corollary *lborel-set-integral-Iic-shift*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *b*::*real*
shows (∫ *x*∈{..*b*}. *f x* ∂*lborel*) = (∫ *x*∈{..*b*−*t*}. *f* (*t*+*x*) ∂*lborel*)
using *lborel-set-integral-Iic-affine-pos*[**where** *c=1*] **by** *simp*

lemma *lborel-set-integral-Icc-affine-neg*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *a b c* :: *real*
assumes *c* < *0*
shows (∫ *x*∈{*a*..*b*}. *f x* ∂*lborel*) = −*c* ∗*R* (∫ *x*∈{(*b*−*t*)/*c*..(*a*−*t*)/*c*}. *f* (*t* + *c*∗*x*) ∂*lborel*)
unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Icc-affine-neg-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** *c=c*] **by** *force*

corollary *lborel-set-integral-Icc-reverse*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *a b* :: *real*
shows (∫ *x*∈{*a*..*b*}. *f x* ∂*lborel*) = (∫ *x*∈{*t*−*b*..*t*−*a*}. *f* (*t*−*x*) ∂*lborel*)
using *lborel-set-integral-Icc-affine-neg*[**where** *c=−1*] **by** *simp*

lemma *lborel-set-integral-Ici-affine-neg*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *b c* :: *real*
assumes *c* < *0*
shows (∫ *x*∈{..*b*}. *f x* ∂*lborel*) = −*c* ∗*R* (∫ *x*∈{(*b*−*t*)/*c*..}. *f* (*t* + *c*∗*x*) ∂*lborel*)
unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Ici-affine-neg-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** *c=c*] **by** *force*

corollary *lborel-set-integral-Ici-reverse*:
fixes *f* :: *real* ⇒ *'a* :: {*banach*, *second-countable-topology*} **and** *b*::*real*
shows (∫ *x*∈{..*b*}. *f x* ∂*lborel*) = (∫ *x*∈{*t*−*b*..}. *f* (*t*−*x*) ∂*lborel*)
using *lborel-set-integral-Ici-affine-neg*[**where** *c=−1*] **by** *simp*

lemma *lborel-set-integral-Iic-affine-neg*:
fixes $f :: real \Rightarrow 'a :: \{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ c :: real$
assumes $c < 0$
shows $(\int x{\in}\{a..\}.\ f\ x\ \partial lborel) = -c *_R (\int x{\in}\{..(a-t)/c\}.\ f\ (t + c{*}x)\ \partial lborel)$
unfolding *set-lebesgue-integral-def* **using** *assms*
apply (*rewrite indicator-Iic-affine-neg-inverse*, *simp*)
using *lborel-integral-real-affine*[**where** $c{=}c$] **by** *force*

corollary *lborel-set-integral-Iic-reverse*:
fixes $f :: real \Rightarrow 'a :: \{banach, second\text{-}countable\text{-}topology\}$ **and** $a{::}real$
shows $(\int x{\in}\{a..\}.\ f\ x\ \partial lborel) = (\int x{\in}\{..t-a\}.\ f\ (t-x)\ \partial lborel)$
using *lborel-set-integral-Iic-affine-neg*[**where** $c{=}{-}1$] **by** *simp*

lemma *set-integrable-Ici-equiv-aux*:
fixes $f :: real \Rightarrow 'a{::}\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b :: real$
assumes $\bigwedge c\ d.\ set\text{-}integrable\ lborel\ \{c..d\}\ f\ a \le b$
shows *set-integrable lborel* $\{a..\}\ f \longleftrightarrow$ *set-integrable lborel* $\{b..\}\ f$
proof
assume *set-integrable lborel* $\{a..\}\ f$
thus *set-integrable lborel* $\{b..\}\ f$ **by** (*rule set-integrable-subset*; *simp add*: *assms*)
next
assume *set-integrable lborel* $\{b..\}\ f$
moreover have *set-integrable lborel* $\{a..b\}\ f$ **using** *assms* **by** *blast*
moreover have $\{a..\} = \{a..b\} \cup \{b..\}$ **using** *assms* **by** *auto*
ultimately show *set-integrable lborel* $\{a..\}\ f$ **using** *set-integrable-Un* **by** *force*
qed

corollary *set-integrable-Ici-equiv*:
fixes $f :: real \Rightarrow 'a{::}\{banach, second\text{-}countable\text{-}topology\}$ **and** $a\ b :: real$
assumes $\bigwedge c\ d.\ set\text{-}integrable\ lborel\ \{c..d\}\ f$
shows *set-integrable lborel* $\{a..\}\ f \longleftrightarrow$ *set-integrable lborel* $\{b..\}\ f$
using *set-integrable-Ici-equiv-aux assms* **by** (*smt* (*verit*))

lemma *set-integrable-Iic-equiv*:
fixes $f :: real \Rightarrow real$ **and** $a\ b :: real$
assumes $\bigwedge c\ d.\ set\text{-}integrable\ lborel\ \{c..d\}\ f$
shows *set-integrable lborel* $\{..a\}\ f \longleftrightarrow$ *set-integrable lborel* $\{..b\}\ f$ (**is** *?LHS* $\longleftrightarrow$ *?RHS*)
proof −
have *?LHS* $\longleftrightarrow$ *set-integrable lborel* $\{-a..\}\ (\lambda x.\ f\ (-x))$
using *set-integrable-Ici-reverse*[**where** $t{=}0$] **by** *force*
also have ... $\longleftrightarrow$ *set-integrable lborel* $\{-b..\}\ (\lambda x.\ f\ (-x))$
proof −
have $\bigwedge c\ d.\ set\text{-}integrable\ lborel\ \{c..d\}\ (\lambda x.\ f\ (-x))$
apply (*rewrite at* {¤..-} *minus-minus*[*THEN sym*])
apply (*rewrite at* {-..¤} *minus-minus*[*THEN sym*])
using *assms set-integrable-Icc-reverse*[**where** $t{=}0$] **by** *force*
thus *?thesis* **by** (*rule set-integrable-Ici-equiv*)
qed

also have ... $\longleftrightarrow$ *?RHS* **using** *set-integrable-Ici-reverse*[**where** *t=0*] **by** *force*
finally show *?thesis* **.**
qed

2.3 Alternative Integral Test

lemma *nn-integral-suminf-Ico-real-nat*:
fixes *a*::*real* **and** *f* :: *real* ⇒ *ennreal*
assumes *f* ∈ *borel-measurable lborel*
shows $(\int^+ x{\in}\{a..\}.\ f\ x\ \partial lborel) = (\sum k.\ \int^+ x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ x\ \partial lborel)$
apply (*rewrite Ico-real-nat-union*[*THEN sym*])
using *Ico-real-nat-disjoint assms* **by** (*intro nn-integral-disjoint-family*; *simp*)

lemma *set-integrable-iff-bounded*:
fixes *f* :: *'a* ⇒ *'b*::{*banach*, *second-countable-topology*}
assumes *A* ∈ *sets M*
shows *set-integrable M A f* $\longleftrightarrow$ *set-borel-measurable M A f* ∧ $(\int^+ x{\in}A.\ norm\ (f\ x)\ \partial M) < \infty$
unfolding *set-integrable-def set-borel-measurable-def* **using** *integrable-iff-bounded*
by (*smt* (*verit*, *ccfv-threshold*) *indicator-mult-ennreal indicator-pos-le*
mult.commute nn-integral-cong norm-scaleR)

theorem *set-integrable-iff-summable*:
fixes *a*::*real* **and** *f* :: *real* ⇒ *real*
assumes *antimono-on* {*a*..} *f* ⋀*x*. *a* ≤ *x* ⟹ *f x* ≥ *0 f* ∈ *borel-measurable lborel*
shows *set-integrable lborel* {*a*..} *f* $\longleftrightarrow$ *summable* (λ*k*. *f* (*a*+*k*))
proof
assume *asm*: *set-integrable lborel* {*a*..} *f*
have [*measurable*]: (λ*x*. *ennreal* (*f x*)) ∈ *borel-measurable lborel* **using** *assms* **by** *simp*
have ∀ *k*≥*0*. *norm* (*f* (*a*+(*k*+*1*::*nat*))) ≤ $(\int x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ x\ \partial lborel)$
proof −
{ fix *k*::*nat*
have *norm* (*f* (*a*+(*k*+*1*::*nat*))) = *f* (*a*+*k*+*1*)
using *assms* **by** (*smt* (*verit*) *of-nat-0-le-iff of-nat-1 of-nat-add real-norm-def*)
also have ... = $(\int x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ (a{+}k{+}1)\ \partial lborel)$
unfolding *set-lebesgue-integral-def* **by** *simp*
also have ... ≤ $(\int x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ x\ \partial lborel)$
apply (*rule set-integral-mono*, *simp*)
apply (*rule set-integrable-restrict-space*[*of lborel* {*a*..}], *simp add*: *asm*)
apply (*rewrite sets-restrict-space*, *force*)
using *assms* **unfolding** *mono-on-def monotone-on-def* **by** *simp*
finally have *norm* (*f* (*a*+(*k*+*1*::*nat*))) ≤ $(\int x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ x\ \partial lborel)$
. }
thus *?thesis* **by** *simp*
qed
moreover have *summable* (λ*k*. $\int x{\in}\{a{+}k..{<}a{+}k{+}1\}.\ f\ x\ \partial lborel$)
proof −

```
    have (∫⁺x∈{a..}. ennreal (f x) ∂lborel) ≠ ∞
      using asm unfolding set-integrable-def apply simp
      by (smt (verit) indicator-mult-ennreal infinity-ennreal-def mult.commute
          nn-integral-cong real-integrable-def)
    thus ?thesis
      apply (rewrite in asm nn-integral-suminf-Ico-real-nat, simp)
      apply (rule summable-suminf-not-top)
      using assms apply (intro set-integral-nonneg, force)
      apply (rewrite set-nn-integral-eq-set-integral[THEN sym], simp add: assms)
      by (rule set-integrable-subset[of lborel {a..}], simp-all add: asm) force
  qed
  ultimately have summable (λk. f (a+(k+1::nat)))
    using summable-comparison-test by (smt (verit, del-insts))
  thus summable (λk. f (a+k)) using summable-iff-shift by blast
next
  assume asm: summable (λk. f (a+k))
  hence (∫⁺x∈{a..}. ennreal |f x| ∂lborel) < ∞
  proof −
    have (∫⁺x∈{a..}. ennreal |f x| ∂lborel) = (∫⁺x∈{a..}. ennreal (f x) ∂lborel)
     using assms by (metis abs-of-nonneg atLeast-iff indicator-simps(2) mult-eq-0-iff)
    also have ... = (∑k. ∫⁺x∈{a+k..<a+k+1}. ennreal (f x) ∂lborel)
      using assms by (rewrite nn-integral-suminf-Ico-real-nat; simp)
    also have ... ≤ (∑k. ∫⁺x∈{a+k..<a+k+1}. ennreal (f (a+k)) ∂lborel)
    proof −
      have ⋀(k::nat) x. x∈{a+k..<a+k+1} ⟹ f x ≤ f (a+k)
        using assms unfolding monotone-on-def by auto
      thus ?thesis
        apply (intro suminf-le, simp-all)
        by (rule nn-integral-mono)
           (metis (no-types, opaque-lifting) atLeastLessThan-iff dual-order.refl en-
nreal-leI
           indicator-simps(2) mult-eq-0-iff mult-mono zero-le)
    qed
    also have ... = (∑k. ennreal (f (a+k)))
      apply (rule suminf-cong)
      by (rewrite nn-integral-cmult-indicator; simp)
    also have ... < ∞
      unfolding infinity-ennreal-def apply (rewrite less-top[THEN sym])
      using asm assms by (smt (verit) of-nat-0-le-iff suminf-cong suminf-ennreal2
top-neq-ennreal)
    finally show ?thesis .
  qed
  moreover have set-borel-measurable lborel {a..} f
    using assms unfolding set-borel-measurable-def by simp
  ultimately show set-integrable lborel {a..} f by (rewrite set-integrable-iff-bounded)
auto
qed
```

2.4 Interchange of Differentiation and Lebesgue Integration

definition *measurable-extension* :: *'a measure* ⇒ *'b measure* ⇒ (*'a* ⇒ *'b*) ⇒ *'a* ⇒ *'b* **where**
measurable-extension M N f =
(*SOME g*. *g* ∈ *M* →$_M$ *N* ∧ (∃ *S*∈(*null-sets M*). {*x* ∈ *space M*. *f x* ≠ *g x*} ⊆ *S*))
— The term *measurable-extension* is proposed by Reynald Affeldt.
— This function is used to make an almost-everywhere-defined function measurable.

lemma
fixes *f g*
assumes *g* ∈ *M* →$_M$ *N S* ∈ *null-sets M* {*x* ∈ *space M*. *f x* ≠ *g x*} ⊆ *S*
shows *measurable-extensionI*: *AE x in M*. *f x* = *measurable-extension M N f x*
and
measurable-extensionI2: *AE x in M*. *g x* = *measurable-extension M N f x* **and**
measurable-extension-measurable: *measurable-extension M N f* ∈ *measurable M N*
proof −
let *?G* = λ*g*. *g* ∈ *M* →$_M$ *N* **and** *?S* = λ*g*. ∃ *S*∈*null-sets M*. {*x* ∈ *space M*. *f x* ≠ *g x*} ⊆ *S*
show *AE x in M*. *f x* = *measurable-extension M N f x*
unfolding *measurable-extension-def*
apply (*rule someI2*[*of* λ*g*. *?G g* ∧ *?S g g*])
using *assms* **apply** *blast*
using *AE-I'* **by** *auto*
moreover have *AE x in M*. *g x* = *f x*
using *assms* **by** (*smt* (*verit*, *best*) *AE-I' Collect-cong*)
ultimately show *AE x in M*. *g x* = *measurable-extension M N f x* **by** *force*
show *measurable-extension M N f* ∈ *measurable M N*
unfolding *measurable-extension-def*
apply (*rule conjE*[*of ?G g ?S g*])
using *assms* **apply** *auto*[*1*]
using *someI-ex*[*of* λ*g*. *?G g* ∧ *?S g*] **by** *auto*
qed

corollary *measurable-measurable-extension-AE*:
fixes *f*
assumes *f* ∈ *M* →$_M$ *N*
shows *AE x in M*. *f x* = *measurable-extension M N f x*
by (*rule measurable-extensionI*[**where** *g*=*f* **and** *S*={}]; *simp add*: *assms*)

definition *borel-measurable-extension* ::
'a measure ⇒ (*'a* ⇒ *'b*::*topological-space*) ⇒ *'a* ⇒ *'b* **where**
borel-measurable-extension M f = *measurable-extension M borel f*

lemma
fixes *f g*
assumes *g* ∈ *borel-measurable M S* ∈ *null-sets M* {*x* ∈ *space M*. *f x* ≠ *g x*} ⊆ *S*

shows *borel-measurable-extensionI*: *AE x in M. f x = borel-measurable-extension M f x* **and**
borel-measurable-extensionI2: *AE x in M. g x = borel-measurable-extension M f x* **and**
borel-measurable-extension-measurable: *borel-measurable-extension M f ∈ borel-measurable M*
unfolding *borel-measurable-extension-def* **using** *assms*
apply −
using *measurable-extensionI* **apply** *blast*
using *measurable-extensionI2* **apply** *blast*
using *measurable-extension-measurable* **by** *blast*

corollary *borel-measurable-measurable-extension-AE*:
fixes *f*
assumes *f ∈ borel-measurable M*
shows *AE x in M. f x = borel-measurable-extension M f x*
using *assms measurable-measurable-extension-AE* **unfolding** *borel-measurable-extension-def* **by** *auto*

definition *set-borel-measurable-extension* ::
'a measure ⇒ 'a set ⇒ ('a ⇒ 'b::topological-space) ⇒ 'a ⇒ 'b
where *set-borel-measurable-extension M A f = borel-measurable-extension (restrict-space M A) f*

lemma
fixes *f g :: 'a ⇒ 'b::real-normed-vector* **and** *A*
assumes *A ∈ sets M set-borel-measurable M A g S ∈ null-sets M {x ∈ A. f x ≠ g x} ⊆ S*
shows *set-borel-measurable-extensionI*:
AE x∈A in M. f x = set-borel-measurable-extension M A f x **and**
set-borel-measurable-extensionI2:
AE x∈A in M. g x = set-borel-measurable-extension M A f x **and**
set-borel-measurable-extension-measurable:
set-borel-measurable M A (set-borel-measurable-extension M A f)
proof −
have *g ∈ borel-measurable (restrict-space M A)*
using *assms* **by** (*rewrite set-borel-measurable-restrict-space-iff*; *simp*)
moreover have *S ∩ A ∈ null-sets (restrict-space M A)*
using *assms null-sets-restrict-space* **by** (*metis Int-lower2 null-set-Int2*)
moreover have *{x ∈ space (restrict-space M A). f x ≠ g x} ⊆ S ∩ A*
using *assms* **by** (*rewrite space-restrict-space2*; *simp*)
ultimately show *AE x∈A in M. f x = set-borel-measurable-extension M A f x* **and**
AE x∈A in M. g x = set-borel-measurable-extension M A f x **and**
set-borel-measurable M A (set-borel-measurable-extension M A f)
unfolding *set-borel-measurable-extension-def* **using** *assms*
apply −
apply (*rewrite AE-restrict-space-iff*[*THEN sym*], *simp*)
apply (*rule borel-measurable-extensionI*[*of g - S ∩ A*]; *simp*)

apply (*rewrite AE-restrict-space-iff*[*THEN sym*], *simp*)
apply (*rule borel-measurable-extensionI2*[*of g - S ∩ A*]; *simp*)
apply (*rewrite set-borel-measurable-restrict-space-iff*[*THEN sym*], *simp*)
by (*rule borel-measurable-extension-measurable*[*of g - S ∩ A*]; *simp*)
qed

corollary *set-borel-measurable-measurable-extension-AE*:
fixes *f*::*'a* ⇒ *'b*::*real-normed-vector* **and** *A*
assumes *set-borel-measurable M A f A* ∈ *sets M*
shows *AE x*∈*A in M*. *f x* = *set-borel-measurable-extension M A f x*
using *set-borel-measurable-restrict-space-iff*
borel-measurable-measurable-extension-AE AE-restrict-space-iff
unfolding *set-borel-measurable-extension-def*
by (*smt* (*verit*) *AE-cong sets.Int-space-eq2 assms*)

proposition *interchange-deriv-LINT-general*:
fixes *a b* :: *real* **and** *f* :: *real* ⇒ *'a* ⇒ *real* **and** *g* :: *'a* ⇒ *real*
assumes *f-integ*: ⋀*r*. *r*∈{*a*<..<*b*} ⟹ *integrable M* (*f r*) **and**
f-diff: *AE x in M*. (λ*r*. *f r x*) *differentiable-on* {*a*<..<*b*} **and**
Df-bound: *AE x in M*. ∀ *r*∈{*a*<..<*b*}. |*deriv* (λ*r*. *f r x*) *r*| ≤ *g x integrable M g*
shows ⋀*r*. *r*∈{*a*<..<*b*} ⟹ ((λ*r*. ∫ *x*. *f r x* ∂*M*) *has-real-derivative*
∫ *x*. *borel-measurable-extension M* (λ*x*. *deriv* (λ*r*. *f r x*) *r*) *x* ∂*M*) (*at r*)
proof −

Preparation

have *f-msr*: ⋀*r*. *r*∈{*a*<..<*b*} ⟹ *f r* ∈ *borel-measurable M* **using** *f-integ* **by** *auto*
from *f-diff* **obtain** *N1* **where** *N1-null*: *N1* ∈ *null-sets M* **and**
⋀*x*. *x* ∈ *space M* − *N1* ⟹ (λ*s*. *f s x*) *differentiable-on* {*a*<..<*b*}
by (*smt* (*verit*) *AE-E3*)
hence *f-diffN1*: ⋀*x*. *x* ∈ *space M* − *N1* ⟹ (λ*s*. *f s x*) *differentiable-on* {*a*<..<*b*}
by (*meson Diff-iff sets.sets-into-space subset-eq*)
from *Df-bound* **obtain** *N2* **where** *N2-null*: *N2* ∈ *null-sets M* **and**
⋀*x*. *x* ∈ *space M* − *N2* ⟹ ∀ *r*∈{*a*<..<*b*}. |*deriv* (λ*s*. *f s x*) *r*| ≤ *g x*
by (*smt* (*verit*) *AE-E3*)
hence *Df-boundN2*:⋀*x*. *x* ∈ *space M* − *N2* ⟹ ∀ *r*∈{*a*<..<*b*}. |*deriv* (λ*s*. *f s x*) *r*| ≤ *g x*
by (*meson Diff-iff sets.sets-into-space subset-eq*)
define *N* **where** *N* ≡ *N1* ∪ *N2*
let *?CN* = *space M* − *N*
have *N-null*: *N* ∈ *null-sets M* **and** *N-msr*: *N* ∈ *sets M*
unfolding *N-def* **using** *N1-null N2-null* **by** *auto*
have *f-diffCN*: ⋀*x*. *x*∈*?CN* ⟹ (λ*s*. *f s x*) *differentiable-on* {*a*<..<*b*}
unfolding *N-def* **using** *f-diffN1* **by** *simp*
define *Df* :: *real* ⇒ *'a* ⇒ *real* **where**
Df r x ≡ *indicator* ({*a*<..<*b*}×*?CN*) (*r*,*x*) ∗ *deriv* (λ*s*. *f s x*) *r* **for** *r x*
have *Df-boundCN*: ⋀*x*. *x*∈*?CN* ⟹ ∀ *r*∈{*a*<..<*b*}. |*Df r x*| ≤ *g x*
unfolding *Df-def N-def* **using** *Df-boundN2* **by** *simp*

Main Part of the Proof

fix r **assume** *r-ab*: $r \in \{a<..<b\}$
then obtain e **where** *e-pos*: $e > 0$ **and** *ball-ab*: $ball\ r\ e \subseteq \{a<..<b\}$
by (*meson openE open-greaterThanLessThan*)
have $\bigwedge d::nat \Rightarrow real.\ [\![\forall i.\ d\ i \in UNIV - \{0\};\ d \longrightarrow 0]\!] \Longrightarrow$
$((\lambda h.\ ((\int x.\ f\ (r+h)\ x\ \partial M) - \int x.\ f\ r\ x\ \partial M)\ /\ h) \circ d) \longrightarrow$
$\int x.\ borel\text{-}measurable\text{-}extension\ M\ (\lambda y.\ deriv\ (\lambda s.\ f\ s\ y)\ r)\ x\ \partial M$
proof −
fix $d::nat \Rightarrow real$ **assume** *d-neq0*: $\forall i.\ d\ i \in UNIV - \{0\}$ **and** *d-to0*: $d \longrightarrow 0$
then obtain m **where** $\forall i \geq m.\ |d\ i - 0| < e$ **using** *LIMSEQ-def e-pos dist-real-def* **by** *metis*
hence *rd-ab*: $\bigwedge n.\ r + d\ (n+m) \in \{a<..<b\}$ **using** *dist-real-def ball-ab* **by** (*simp add*: *subset-eq*)
hence *fd-msr*: $\bigwedge n.\ (\lambda x.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m)) \in borel\text{-}measurable\ M$
using *r-ab* **by** (*measurable*; (*intro f-msr*)*?*; *simp*)
hence *limf-msr*: $(\lambda x.\ lim\ (\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m))) \in borel\text{-}measurable\ M$
by *measurable*
moreover have *limf-Df*: $\bigwedge x.\ x \in ?CN \Longrightarrow (\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m)) \longrightarrow Df\ r\ x$
proof −
fix x **assume** *x-CN*: $x \in ?CN$
hence $(\lambda s.\ f\ s\ x)$ *field-differentiable* $(at\ r)$
using *f-diffCN r-ab*
by (*metis at-within-open differentiable-on-eq-field-differentiable-real open-greaterThanLessThan*)
hence $((\lambda h.\ (f\ (r+h)\ x - f\ r\ x)\ /\ h) \longrightarrow Df\ r\ x)\ (at\ 0)$
apply (*rewrite* **in** *asm DERIV-deriv-iff-field-differentiable*[*THEN sym*])
unfolding *Df-def* **using** *r-ab x-CN* **by** (*simp add*: *DERIV-def*)
hence $(\lambda i.\ (f\ (r + d\ i)\ x - f\ r\ x)\ /\ d\ i) \longrightarrow Df\ r\ x$
apply (*rewrite* **in** *asm tendsto-at-iff-sequentially*)
apply (*rule allE'*[**where** x=d], *simp*)
unfolding *comp-def* **using** *d-neq0 d-to0* **by** *simp*
thus $(\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m)) \longrightarrow Df\ r\ x$
by (*rule LIMSEQ-ignore-initial-segment*[**where** k=m])
qed
ultimately have *Df-eq*:
$\bigwedge x.\ Df\ r\ x = indicator\ ?CN\ x * lim\ (\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m))$
proof −
fix x
show $Df\ r\ x = indicator\ ?CN\ x * lim\ (\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m))$
proof (*cases* ‹$x \in ?CN$›)
case *True*
hence $lim\ (\lambda n.\ (f\ (r + d\ (n+m))\ x - f\ r\ x)\ /\ d\ (n+m)) = Df\ r\ x$
by (*intro limI*, *rule limf-Df*)
thus *?thesis* **using** *True* **by** *simp*
next

```
      case False
      thus ?thesis unfolding Df-def by simp
    qed
  qed
  hence Df-msr: Df r ∈ borel-measurable M
    apply (rewrite in λx. ⌑ Df-eq)
    apply (measurable; (rule limf-msr)?)
    using N-null unfolding null-sets-def by force
  have ((λh. ((∫ x. f (r+h) x ∂M) − ∫ x. f r x ∂M) / h) ∘ d) ⟶
    ∫ x. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ∂M
  proof −
    have (λn. ∫ x. (f (r + d (n+m)) x − f r x) / d (n+m) ∂M) ⟶
      ∫ x. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ∂M
    proof − — by Lebesgue's Dominated Convergence Theorem
      have AE x in M. (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ⟶
        lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m))
       using limf-Df Df-eq N-null by (smt (verit) DiffI AE-I' limI mem-Collect-eq
subset-eq)
      moreover have ⋀n. AE x in M. norm ((f (r + d (n+m)) x − f r x) / d
(n+m)) ≤ g x
      proof −
        fix n
        { fix x assume x-CN: x∈?CN
          let ?I = {r..(r + d (n+m))} ∪ {(r + d (n+m))..r}
          have f-diffI: (λs. f s x) differentiable-on ?I
           apply (rule differentiable-on-subset[where t={a<..<b}], rule f-diffCN,
rule x-CN)
            using r-ab rd-ab[of n] by (rewrite Un-subset-iff, auto)
         hence continuous-on ?I (λs. f s x) (λs. f s x) differentiable-on interior ?I
           apply −
           using differentiable-imp-continuous-on apply blast
           by (metis differentiable-on-subset interior-subset)
         then obtain t where t-01: t∈{0<..<1} and
           f-MVT: f (r + d (n+m)) x − f r x = d (n+m) * deriv (λs. f s x) (r
+ t * (d (n+m)))
           by (rule MVT-order-free)
         hence 0 < t t < 1 by simp-all
         hence rtd-ab: r + t * (d (n+m)) ∈ {a<..<b}
           using r-ab rd-ab[of n]
        by simp (smt (verit, ccfv-threshold) mult-less-cancel-left mult-less-cancel-right2)
         have d (n+m) * deriv (λs. f s x) (r + t * (d (n+m))) =
           d (n+m) * Df (r + t * (d (n+m))) x
         proof −
           have r + t * (d (n+m)) ∈ {a<..<b}
             using r-ab rd-ab[of n] t-01
             by (smt (verit) ball-eq-greaterThanLessThan dist-real-def
                greaterThanLessThan-eq-iff greaterThanLessThan-eq-ball mem-ball
                 mult-le-cancel-right1 mult-minus-right mult-pos-neg)
           thus ?thesis unfolding Df-def using x-CN by simp
```

```
          qed
          with f-MVT have (f (r + d (n+m)) x − f r x) / d (n+m) = Df (r +
t ∗ (d (n+m))) x
            using d-neq0 by simp
          moreover have |Df (r + t ∗ (d (n+m))) x| ≤ g x using Df-boundCN
x-CN rtd-ab by simp
          ultimately have |(f (r + d (n+m)) x − f r x) / d (n+m)| ≤ g x by
simp }
        thus AE x in M. norm ((f (r + d (n+m)) x − f r x) / d (n+m)) ≤ g x
          unfolding real-norm-def using AE-I′ N-null
          by (smt (verit, ccfv-threshold) Diff-iff mem-Collect-eq subsetI)
      qed
      ultimately show ((λn. ∫ x. (f (r + d (n+m)) x − f r x) / d (n+m) ∂M)
⟶
        ∫ x. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ∂M)
        using limf-msr fd-msr Df-bound
        by (intro integral-dominated-convergence[where w=g], simp-all)
    qed
    moreover have ⋀n. ((∫ x. f (r + d (n+m)) x ∂M) − ∫ x. f r x ∂M) / d
(n+m) =
      ∫ x. (f (r + d (n+m)) x − f r x) / d (n+m) ∂M
      using d-neq0 apply simp
      by (rewrite Bochner-Integration.integral-diff;
          (rule f-integ | simp); (rule rd-ab | rule r-ab))
    ultimately show ?thesis
      unfolding comp-def using d-neq0
      apply −
      by (rule LIMSEQ-offset[where k=m]) simp
  qed
  moreover have (∫ x. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ∂M)
=
    ∫ x. borel-measurable-extension M (λy. deriv (λs. f s y) r) x ∂M
  proof −
    have (∫ x. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) ∂M) = ∫ x. Df
r x ∂M
    proof −
      have AE x in M. lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) = Df
r x
      proof −
        { fix x assume x-CN: x∈?CN
          hence lim (λn. (f (r + d (n+m)) x − f r x) / d (n+m)) = Df r x by
(simp add: Df-eq) }
        thus ?thesis using AE-I′ N-null by (smt (verit, del-insts) DiffI mem-Collect-eq
subsetI)
      qed
      thus ?thesis using limf-msr Df-msr by (intro integral-cong-AE; simp)
    qed
    also have ... = ∫ x. borel-measurable-extension M (λy. deriv (λs. f s y) r)
x ∂M
```

```
    proof −
      have AE x in M. Df r x = borel-measurable-extension M (λy. deriv (λs. f s
y) r) x and
          borel-measurable-extension M (λy. deriv (λs. f s y) r) ∈ borel-measurable
M
       proof −
        have {x ∈ space M. deriv (λs. f s x) r ≠ Df r x} ⊆ N
        proof −
          { fix x assume x∈?CN
            hence deriv (λs. f s x) r = Df r x unfolding Df-def using r-ab by
simp }
          thus ?thesis by blast
        qed
        thus AE x in M. Df r x = borel-measurable-extension M (λy. deriv (λs. f
s y) r) x and
          borel-measurable-extension M (λy. deriv (λs. f s y) r) ∈ borel-measurable
M
          using Df-msr N-null
           apply −
           apply (rule borel-measurable-extensionI2[where S=N]; simp)
          by (rule borel-measurable-extension-measurable[where g=Df r]; simp)
       qed
       thus ?thesis using Df-msr by (intro integral-cong-AE; simp)
     qed
     finally show ?thesis .
   qed
    ultimately show ((λh. ((∫ x. f (r+h) x ∂M) − ∫ x. f r x ∂M) / h) ∘ d)
⟶
     ∫ x. borel-measurable-extension M (λy. deriv (λs. f s y) r) x ∂M
     using tendsto-cong-limit by simp
  qed
  thus ((λs. ∫ x. f s x ∂M) has-real-derivative
    ∫ x. borel-measurable-extension M (λy. deriv (λs. f s y) r) x ∂M) (at r)
    by (rewrite DERIV-def, rewrite tendsto-at-iff-sequentially) simp
qed

proposition interchange-deriv-LINT:
  fixes a b :: real and f :: real ⇒ 'a ⇒ real and g :: 'a ⇒ real
  assumes ⋀r. r∈{a<..<b} ⟹ integrable M (f r) and
    AE x in M. (λr. f r x) differentiable-on {a<..<b} and
    ⋀r. r∈{a<..<b} ⟹ (λx. (deriv (λr. f r x) r)) ∈ borel-measurable M and
    AE x in M. ∀r∈{a<..<b}. |deriv (λr. f r x) r| ≤ g x integrable M g
  shows ⋀r. r∈{a<..<b} ⟹ ((λr. ∫ x. f r x ∂M) has-real-derivative
    ∫ x. deriv (λr. f r x) r ∂M) (at r)
proof −
  fix r assume r-ab: r∈{a<..<b}
  hence Df-msr: (λx. deriv (λs. f s x) r) ∈ borel-measurable M using assms by
simp
  have ((λs. ∫ x. f s x ∂M) has-real-derivative
```

$\int x.$ *borel-measurable-extension* M $(\lambda y.$ *deriv* $(\lambda s.$ f s $y)$ $r)$ x $\partial M)$ $(at$ $r)$
using *assms r-ab* **by** (*intro interchange-deriv-LINT-general*; *simp*)
moreover have ($\int x.$ *borel-measurable-extension* M $(\lambda y.$ *deriv* $(\lambda s.$ f s $y)$ $r)$ x $\partial M) =$
$\int x.$ *deriv* $(\lambda s.$ f s $x)$ r ∂M
apply (*rule integral-cong-AE*)
apply (*rule borel-measurable-extension-measurable*
[**where** $g{=}\lambda y.$ *deriv* $(\lambda s.$ f s $y)$ r **and** $S{=}\{\}$], *simp-all add*: *Df-msr*)
using *borel-measurable-measurable-extension-AE Df-msr* **by** (*smt* (*verit*) *AE-cong*)
ultimately show ($(\lambda r.$ $\int x.$ f r x $\partial M)$ *has-real-derivative* $\int x.$ *deriv* $(\lambda r.$ f r $x)$ r $\partial M)$ $(at$ $r)$
by *simp*
qed

proposition *interchange-deriv-LINT-set-general*:
fixes a b :: *real* **and** f :: *real* $\Rightarrow$ $'a$ $\Rightarrow$ *real* **and** g :: $'a$ $\Rightarrow$ *real* **and** A :: $'a$ *set*
assumes *A-msr*: $A \in$ *sets* M **and**
f-integ: $\bigwedge r.$ $r{\in}\{a{<..<}b\} \Longrightarrow$ *set-integrable* M A $(f$ $r)$ **and**
f-diff: *AE* $x{\in}A$ *in* $M.$ $(\lambda r.$ f r $x)$ *differentiable-on* $\{a{<..<}b\}$ **and**
Df-bound: *AE* $x{\in}A$ *in* $M.$ $\forall$ $r{\in}\{a{<..<}b\}.$ $|$*deriv* $(\lambda r.$ f r $x)$ $r| \leq g$ x *set-integrable* M A g
shows $\bigwedge r.$ $r{\in}\{a{<..<}b\} \Longrightarrow$ $((\lambda r.$ $\int x{\in}A.$ f r x $\partial M)$ *has-real-derivative*
($\int x{\in}A.$ *set-borel-measurable-extension* M A $(\lambda x.$ *deriv* $(\lambda r.$ f r $x)$ $r)$ x $\partial M))$ $(at$ $r)$
proof $-$
let *?M-A* = *restrict-space* M A
have $\bigwedge r.$ $r{\in}\{a{<..<}b\} \Longrightarrow$ *integrable* *?M-A* $(f$ $r)$
using *A-msr f-integ set-integrable-restrict-space-iff* **by** *auto*
moreover have *AE* x *in* *?M-A*. $(\lambda r.$ f r $x)$ *differentiable-on* $\{a{<..<}b\}$
using *AE-restrict-space-iff A-msr f-diff* **by** (*metis sets.Int-space-eq2*)
moreover have *AE* x *in* *?M-A*. $\forall$ $r{\in}\{a{<..<}b\}.$ $|$*deriv* $(\lambda r.$ f r $x)$ $r| \leq g$ x **and**
integrable *?M-A* g
using *A-msr Df-bound set-integrable-restrict-space-iff*
apply $-$
by (*simp add*: *AE-restrict-space-iff*, *auto*)
ultimately have $\bigwedge r.$ $r{\in}\{a{<..<}b\} \Longrightarrow$ $((\lambda r.$ *integral*^{L} *?M-A* $(f$ $r))$ *has-real-derivative*
integral^L *?M-A* (*borel-measurable-extension ?M-A* $(\lambda x.$ *deriv* $(\lambda r.$ f r $x)$ $r))$) $(at$ $r)$
by (*rule interchange-deriv-LINT-general*[**where** $M{=}$*restrict-space* M A]) *auto*
thus $\bigwedge r.$ $r{\in}\{a{<..<}b\} \Longrightarrow$ $((\lambda r.$ $\int x{\in}A.$ f r x $\partial M)$ *has-real-derivative*
($\int x{\in}A.$ *set-borel-measurable-extension* M A $(\lambda x.$ *deriv* $(\lambda r.$ f r $x)$ $r)$ x $\partial M))$ $(at$ $r)$
unfolding *set-borel-measurable-extension-def* **using** *assms*
by (*rewrite set-lebesgue-integral-restrict-space*, *simp*)+
qed

proposition *interchange-deriv-LINT-set*:
fixes a b :: *real* **and** f :: *real* $\Rightarrow$ $'a$ $\Rightarrow$ *real* **and** g :: $'a$ $\Rightarrow$ *real* **and** A :: $'a$ *set*
assumes $A \in$ *sets* M **and**

$\bigwedge r.\ r{\in}\{a{<}..{<}b\} \Longrightarrow$ *set-integrable M A* $(f\ r)$ **and**
$AE\ x{\in}A$ *in* $M.\ (\lambda r.\ f\ r\ x)$ *differentiable-on* $\{a{<}..{<}b\}$ **and**
$\bigwedge r.\ r{\in}\{a{<}..{<}b\} \Longrightarrow$ *set-borel-measurable M A* $(\lambda x.\ (deriv\ (\lambda r.\ f\ r\ x)\ r))$ **and**
$AE\ x{\in}A$ *in* $M.\ \forall r{\in}\{a{<}..{<}b\}.\ |deriv\ (\lambda r.\ f\ r\ x)\ r| \leq g\ x$ *set-integrable M A g*
shows $\bigwedge r.\ r{\in}\{a{<}..{<}b\} \Longrightarrow ((\lambda r.\ \int x{\in}A.\ f\ r\ x\ \partial M)$ *has-real-derivative*
$(\int x{\in}A.\ deriv\ (\lambda r.\ f\ r\ x)\ r\ \partial M))\ (at\ r)$
proof −
fix *r* **assume** *r-ab*: $r{\in}\{a{<}..{<}b\}$
hence *Df-msr*: *set-borel-measurable M A* $(\lambda x.\ deriv\ (\lambda s.\ f\ s\ x)\ r)$ **using** *assms* **by** *simp*
have $((\lambda s.\ \int x{\in}A.\ f\ s\ x\ \partial M)$ *has-real-derivative*
$(\int x{\in}A.$ *set-borel-measurable-extension M A* $(\lambda y.\ deriv\ (\lambda s.\ f\ s\ y)\ r)\ x\ \partial M))$ $(at\ r)$
using *assms r-ab* **by** (*intro interchange-deriv-LINT-set-general*; *simp*)
moreover have $(\int x{\in}A.$ *set-borel-measurable-extension M A* $(\lambda y.\ deriv\ (\lambda s.\ f\ s\ y)\ r)\ x\ \partial M) =$
$(\int x{\in}A.\ deriv\ (\lambda s.\ f\ s\ x)\ r\ \partial M)$
apply (*rule set-lebesgue-integral-cong-AE2*, *simp add*: *assms*)
apply (*rule set-borel-measurable-extension-measurable*
[**where** $g{=}\lambda y.\ deriv\ (\lambda s.\ f\ s\ y)\ r$ **and** $S{=}\{\}$], *simp-all add*: *Df-msr assms*)
using *set-borel-measurable-measurable-extension-AE Df-msr assms* **by** (*smt* (*verit*) *AE-cong*)
ultimately show
$((\lambda r.\ \int x{\in}A.\ f\ r\ x\ \partial M)$ *has-real-derivative* $(\int x{\in}A.\ deriv\ (\lambda r.\ f\ r\ x)\ r\ \partial M))\ (at\ r)$
by *simp*
qed

3 Additional Lemmas for the *HOL−Probability* Library

lemma (**in** *finite-borel-measure*)
fixes $F :: real \Rightarrow real$
assumes *nondecF* : $\bigwedge\ x\ y.\ x \leq y \Longrightarrow F\ x \leq F\ y$ **and**
right-cont-F : $\bigwedge a.$ *continuous* $(at\text{-}right\ a)\ F$ **and**
lim-F-at-bot : $(F \longrightarrow 0)$ *at-bot* **and**
lim-F-at-top : $(F \longrightarrow m)$ *at-top* **and**
$m : 0 \leq m$
shows *emeasure-interval-measure-Ioi*: *emeasure* $(interval\text{-}measure\ F)\ \{x{<}..\} = m - F\ x$
and *measure-interval-measure-Ioi*: *measure* $(interval\text{-}measure\ F)\ \{x{<}..\} = m - F\ x$
proof −
interpret *F-FM*: *finite-measure interval-measure F*
using *finite-borel-measure.axioms*(*1*) *finite-borel-measure-interval-measure lim-F-at-bot lim-F-at-top m nondecF right-cont-F* **by** *blast*
have $UNIV = \{..x\} \cup \{x{<}..\}$ **by** *auto*
moreover have $\{..x\} \cap \{x{<}..\} = \{\}$ **by** *auto*

ultimately have *emeasure* (*interval-measure F*) *UNIV* =
emeasure (*interval-measure F*) {..*x*} + *emeasure* (*interval-measure F*) {*x*<..}
by (*simp add*: *plus-emeasure*)
moreover have *emeasure* (*interval-measure F*) *UNIV* = *m*
using *assms interval-measure-UNIV* **by** *presburger*
ultimately show ⋆: *emeasure* (*interval-measure F*) {*x*<..} = *m* − *F x*
using *assms emeasure-interval-measure-Iic*
by (*metis ennreal-add-diff-cancel-left ennreal-minus measure-interval-measure-Iic measure-nonneg top-neq-ennreal*)
hence *ennreal* (*measure* (*interval-measure F*) {*x*<..}) = *m* − *F x*
using *emeasure-eq-measure* **by** (*metis emeasure-eq-ennreal-measure top-neq-ennreal*)
moreover have ⋀*x*. *F x* ≤ *m*
using *lim-F-at-top nondecF* **by** (*intro mono-at-top-le*[**where** *f*=*F*]; *simp add*: *mono-def*)
ultimately show *measure* (*interval-measure F*) {*x*<..} = *m* − *F x*
using *ennreal-inj F-FM.emeasure-eq-measure* **by** *force*
qed

lemma (**in** *prob-space*) *cond-prob-nonneg*[*simp*]: *cond-prob M P Q* ≥ *0*
by (*auto simp*: *cond-prob-def*)

lemma (**in** *prob-space*) *cond-prob-whole-1*: *cond-prob M P P* = *1* **if** *prob* {*ω* ∈ *space M*. *P ω*} ≠ *0*
unfolding *cond-prob-def* **using** *that* **by** *simp*

lemma (**in** *prob-space*) *cond-prob-0-null*: *cond-prob M P Q* = *0* **if** *prob* {*ω* ∈ *space M*. *Q ω*} = *0*
unfolding *cond-prob-def* **using** *that* **by** *simp*

lemma (**in** *prob-space*) *cond-prob-AE-prob*:
assumes {*ω* ∈ *space M*. *P ω*} ∈ *events* {*ω* ∈ *space M*. *Q ω*} ∈ *events*
and *AE ω in M*. *Q ω*
shows *cond-prob M P Q* = *prob* {*ω* ∈ *space M*. *P ω*}
proof −
let *?setP* = {*ω* ∈ *space M*. *P ω*}
let *?setQ* = {*ω* ∈ *space M*. *Q ω*}
have [*simp*]: *prob ?setQ* = *1* **using** *assms prob-Collect-eq-1* **by** *simp*
hence *cond-prob M P Q* = *prob* (*?setP* ∩ *?setQ*)
unfolding *cond-prob-def* **by** (*simp add*: *Collect-conj-eq2*)
also have ... = *prob ?setP*
proof (*rule antisym*)
show *prob* (*?setP* ∩ *?setQ*) ≤ *prob ?setP*
using *assms finite-measure-mono inf-sup-ord*(*1*) **by** *blast*
next
show *prob ?setP* ≤ *prob* (*?setP* ∩ *?setQ*)
proof −
have *prob* (*?setP* ∩ *?setQ*) = *prob ?setP* + *prob ?setQ* − *prob* (*?setP* ∪ *?setQ*)
using *assms* **by** (*smt* (*verit*) *finite-measure-Diff′ finite-measure-Union′*

sup-commute)
also have ... = *prob ?setP* + (*1* − *prob* (*?setP* ∪ *?setQ*)) **by** *simp*
also have ... ≥ *prob ?setP* **by** *simp*
finally show *?thesis* **.**
qed
qed
finally show *?thesis* **.**
qed

3.1 More Properties of *cdf*'s

context *finite-borel-measure*
begin

lemma *cdf-diff-eq2*:
assumes $x \leq y$
shows *cdf M y* − *cdf M x* = *measure M* {*x*<..*y*}
proof (*cases* ‹*x* = *y*›)
case *True*
thus *?thesis* **by** *force*
next
case *False*
hence $x < y$ **using** *assms* **by** *simp*
thus *?thesis* **by** (*rule cdf-diff-eq*)
qed

end

context *prob-space*
begin

lemma *cdf-distr-measurable* [*measurable*]:
assumes [*measurable*]: *random-variable borel X*
shows *cdf* (*distr M borel X*) ∈ *borel-measurable borel*
proof (*rule borel-measurable-mono*)
show *mono* (*cdf* (*distr M borel X*))
unfolding *mono-def*
using *finite-borel-measure.cdf-nondecreasing*
by (*simp add*: *real-distribution.finite-borel-measure-M*)
qed

lemma *cdf-distr-P*:
assumes *random-variable borel X*
shows *cdf* (*distr M borel X*) *x* = $\mathcal{P}(\omega$ *in M*. *X* $\omega \leq x)$
unfolding *cdf-def* **apply** (*rewrite measure-distr*; (*simp add*: *assms*)*?*)
unfolding *vimage-def* **by** (*rule arg-cong*[**where** *f*=*prob*], *force*)

lemma *cdf-continuous-distr-P-lt*:
assumes *random-variable borel X isCont* (*cdf* (*distr M borel X*)) *x*

shows *cdf* (*distr M borel X*) *x* = $\mathcal{P}$(*ω in M. X ω* < *x*)
proof −
have $\mathcal{P}$(*ω in M. X ω* < *x*) = *measure* (*distr M borel X*) {..<*x*}
apply (*rewrite measure-distr*, *simp-all add*: *assms*)
unfolding *vimage-def* **by** *simp* (*smt* (*verit*) *Collect-cong Int-def mem-Collect-eq*)
also have ... = *measure* (*distr M borel X*) ({..<*x*} ∪ {*x*})
apply (*rewrite finite-measure.measure-zero-union*, *simp-all add*: *assms finite-measure-distr*)
using *finite-borel-measure.isCont-cdf real-distribution.finite-borel-measure-M assms* **by** *blast*
also have ... = *measure* (*distr M borel X*) {..*x*} **by** (*metis ivl-disj-un-singleton*(*2*))
also have ... = *cdf* (*distr M borel X*) *x* **unfolding** *cdf-def* **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *cdf-distr-diff-P*:
assumes *x* ≤ *y*
and *random-variable borel X*
shows *cdf* (*distr M borel X*) *y* − *cdf* (*distr M borel X*) *x* = $\mathcal{P}$(*ω in M. x* < *X ω* ∧ *X ω* ≤ *y*)
proof −
interpret *distrX-FBM*: *finite-borel-measure distr M borel X*
using *real-distribution.finite-borel-measure-M real-distribution-distr assms* **by** *simp*
have *cdf* (*distr M borel X*) *y* − *cdf* (*distr M borel X*) *x* = *measure* (*distr M borel X*) {*x*<..*y*}
by (*rewrite distrX-FBM.cdf-diff-eq2*; *simp add*: *assms*)
also have ... = $\mathcal{P}$(*ω in M. x* < *X ω* ∧ *X ω* ≤ *y*)
apply (*rewrite measure-distr*; (*simp add*: *assms*)*?*)
unfolding *vimage-def* **by** (*rule arg-cong*[**where** *f*=*prob*], *force*)
finally show *?thesis* **.**
qed

lemma *cdf-distr-max*:
fixes *c*::*real*
assumes [*measurable*]: *random-variable borel X*
shows *cdf* (*distr M borel* (*λx. max* (*X x*) *c*)) *u* = *cdf* (*distr M borel X*) *u* ∗ *indicator* {*c*..} *u*
proof (*cases* ‹*c* ≤ *u*›)
case *True*
thus *?thesis*
unfolding *cdf-def*
apply (*rewrite measure-distr*; *simp?*)+
by (*smt* (*verit*) *Collect-cong atMost-iff vimage-def*)
next
case *False*
thus *?thesis*
unfolding *cdf-def*
apply (*rewrite measure-distr*; *simp?*)+
by (*smt* (*verit*, *best*) *Int-emptyI atMost-iff measure-empty vimage-eq*)

qed

lemma *cdf-distr-min*:
fixes *c*::*real*
assumes [*measurable*]: *random-variable borel X*
shows *cdf* (*distr M borel* (λ*x*. *min* (*X x*) *c*)) *u* =
1 − (*1* − *cdf* (*distr M borel X*) *u*) ∗ *indicator* {..<*c*} *u*
proof (*cases* ‹*c* ≤ *u*›)
case *True*
thus *?thesis*
unfolding *cdf-def*
using *real-distribution.finite-borel-measure-M real-distribution-distr*
apply (*rewrite measure-distr*; *simp?*)
by (*smt* (*verit*, *del-insts*) *Int-absorb1 atMost-iff prob-space subset-vimage-iff*)
next
case *False*
thus *?thesis*
unfolding *cdf-def*
using *real-distribution.finite-borel-measure-M real-distribution-distr*
apply (*rewrite measure-distr*; *simp?*)+
using *prob-space-axioms assms*
by (*smt* (*verit*) *Collect-cong Int-def atMost-iff prob-space prob-space.cdf-distr-P vimage-eq*)
qed

lemma *cdf-distr-floor-P*:
fixes *X* :: *'a* ⇒ *real*
assumes [*measurable*]: *random-variable borel X*
shows *cdf* (*distr M borel* (λ*x*. ⌊*X x*⌋)) *u* = *P*(*x in M*. *X x* < ⌊*u*⌋ + *1*)
unfolding *cdf-def*
apply (*rewrite measure-distr*; *simp?*)
apply (*rule arg-cong*[**where** *f*=*prob*])
unfolding *vimage-def* **using** *floor-le-iff le-floor-iff* **by** *blast*

lemma *expectation-nonneg-tail*:
assumes [*measurable*]: *random-variable borel X*
and *X-nonneg*: ⋀*x*. *x* ∈ *space M* ⟹ *X x* ≥ *0*
defines *F u* ≡ *cdf* (*distr M borel X*) *u*
shows (∫⁺*x*. *ennreal* (*X x*) ∂*M*) = (∫⁺*u*∈{*0*..}. *ennreal* (*1* − *F u*) ∂*lborel*)
proof −
let *?distrX* = *distr M borel X*
have (∫⁺*x*. *ennreal* (*X x*) ∂*M*) = (∫⁺*u*. *ennreal u* ∂*?distrX*)
by (*rewrite nn-integral-distr*; *simp*)
also have ... = (∫⁺*u*∈{*0*..}. *ennreal u* ∂*?distrX*)
by (*rule nn-integral-distr-set*; *simp add*: *X-nonneg*)
also have ... = (∫⁺*u*∈{*0*..}. (∫⁺*v*∈{*0*..}. *indicator* {..<*u*} *v* ∂*lborel*) ∂*?distrX*)
proof −
have ⋀*u*::*real*. *u*∈{*0*..} ⟹ *ennreal u* = (∫⁺*v*∈{*0*..}. *indicator* {..<*u*} *v* ∂*lborel*)
apply (*rewrite indicator-inter-arith*[*THEN sym*])

```
    apply (rewrite nn-integral-indicator, measurable, simp)
    by (metis atLeastLessThan-def diff-zero emeasure-lborel-Ico inf.commute)
  thus ?thesis by (metis (no-types, lifting) indicator-eq-0-iff mult-eq-0-iff)
 qed
 also have ... = (∫⁺v∈{0..}. (∫⁺u∈{0..}. indicator {..<u} v ∂?distrX) ∂lborel)
 proof −
  have (∫⁺u∈{0..}. (∫⁺v∈{0..}. indicator {..<u} v ∂lborel) ∂?distrX) =
    ∫⁺u. (∫⁺v. indicator {..<u} v * indicator {0..} v * indicator {0..} u ∂lborel)
∂?distrX
    by (rewrite nn-integral-multc; simp)
  also have ... =
      ∫⁺v. (∫⁺u. indicator {..<u} v * indicator {0..} v * indicator {0..} u
∂?distrX) ∂lborel
    apply (rewrite pair-sigma-finite.Fubini'; simp?)
    using pair-sigma-finite.intro assms
      prob-space-distr prob-space-imp-sigma-finite sigma-finite-lborel
     apply blast
    by measurable auto
    also have ... = (∫⁺v∈{0..}. (∫⁺u∈{0..}. indicator {..<u} v ∂?distrX)
∂lborel)
    apply (rewrite nn-integral-multc[THEN sym]; measurable; simp?)
    apply (rule nn-integral-cong)+
    using mult.assoc mult.commute by metis
  finally show ?thesis by simp
 qed
 also have ... = (∫⁺v∈{0..}. (∫⁺u. indicator {v<..} u ∂?distrX) ∂lborel)
  apply (rule nn-integral-cong)
 apply (rewrite nn-integral-multc[THEN sym], measurable; (simp del: nn-integral-indicator)?)+
  apply (rule nn-integral-cong)
  using lessThan-iff greaterThan-iff atLeast-iff indicator-simps
  by (smt (verit, del-insts) mult-1 mult-eq-0-iff)
 also have ... = (∫⁺v∈{0..}. ennreal (1 − F v) ∂lborel)
  apply (rule nn-integral-cong, simp)
  apply (rewrite emeasure-distr; simp?)
  apply (rewrite vimage-compl-atMost[THEN sym])
  unfolding F-def cdf-def
  apply (rewrite measure-distr; simp?)
  apply (rewrite prob-compl[THEN sym], simp)
 by (metis (no-types, lifting) Diff-Compl Diff-Diff-Int Int-commute emeasure-eq-measure)
 finally show ?thesis .
qed

lemma expectation-nonpos-tail:
 assumes [measurable]: random-variable borel X
  and X-nonpos: ⋀x. x ∈ space M ⟹ X x ≤ 0
 defines F u ≡ cdf (distr M borel X) u
 shows (∫⁺x. ennreal (− X x) ∂M) = (∫⁺u∈{..0}. ennreal (F u) ∂lborel)
proof −
 let ?distrX = distr M borel X
```

have $(\int^+ x.\ ennreal\ (-\ X\ x)\ \partial M) = (\int^+ u.\ ennreal\ (-u)\ \partial ?distrX)$
by (*rewrite nn-integral-distr*; *simp*)
also have $\ldots = (\int^+ u{\in}\{..0\}.\ ennreal\ (-u)\ \partial ?distrX)$
proof −
have [*simp*]: $\{..0::real\} \cup \{0<..\} = UNIV$ **by** *force*
have $(\int^+ u.\ ennreal\ (-u)\ \partial ?distrX) =$
$(\int^+ u{\in}\{..0\}.\ ennreal\ (-u)\ \partial ?distrX) + (\int^+ u{\in}\{0<..\}.\ ennreal\ (-u)\ \partial ?distrX)$
by (*rewrite nn-integral-disjoint-pair*[*THEN sym*], *simp-all*, *force*)
also have $\ldots = (\int^+ u{\in}\{..0\}.\ ennreal\ (-u)\ \partial ?distrX)$
apply (*rewrite nn-integral-zero'*[*of* $\lambda u.\ ennreal\ (-u) * indicator\ \{0<..\}\ u$]; *simp?*)
unfolding *indicator-def* **using** *always-eventually ennreal-lt-0* **by** *force*
finally show *?thesis* **.**
qed
also have $\ldots = (\int^+ u{\in}\{..0\}.\ (\int^+ v{\in}\{..0\}.\ indicator\ \{u..\}\ v\ \partial lborel)\ \partial ?distrX)$
proof −
have $\bigwedge u{::}real.\ u{\in}\{..0\} \Longrightarrow ennreal\ (-u) = (\int^+ v{\in}\{..0\}.\ indicator\ \{u..\}\ v\ \partial lborel)$
by (*rewrite indicator-inter-arith*[*THEN sym*]) *simp*
thus *?thesis* **by** (*metis* (*no-types*, *lifting*) *indicator-eq-0-iff mult-eq-0-iff*)
qed
also have $\ldots = (\int^+ v{\in}\{..0\}.\ (\int^+ u{\in}\{..0\}.\ indicator\ \{u..\}\ v\ \partial ?distrX)\ \partial lborel)$
proof −
have $(\int^+ u{\in}\{..0\}.\ (\int^+ v{\in}\{..0\}.\ indicator\ \{u..\}\ v\ \partial lborel)\ \partial ?distrX) =$
$\int^+ u.\ (\int^+ v.\ indicator\ \{u..\}\ v * indicator\ \{..0\}\ v * indicator\ \{..0\}\ u\ \partial lborel)\ \partial ?distrX$
by (*rewrite nn-integral-multc*; *simp*)
also have $\ldots =$
$\int^+ v.\ (\int^+ u.\ indicator\ \{u..\}\ v * indicator\ \{..0\}\ v * indicator\ \{..0\}\ u\ \partial ?distrX)\ \partial lborel$
apply (*rewrite pair-sigma-finite.Fubini'*; *simp?*)
using *pair-sigma-finite.intro assms*
prob-space-distr prob-space-imp-sigma-finite sigma-finite-lborel
apply *blast*
by *measurable auto*
also have $\ldots = (\int^+ v{\in}\{..0\}.\ (\int^+ u{\in}\{..0\}.\ indicator\ \{u..\}\ v\ \partial ?distrX)\ \partial lborel)$
apply (*rewrite nn-integral-multc*[*THEN sym*]; *measurable*; *simp?*)
apply (*rule nn-integral-cong*)+
using *mult.assoc mult.commute* **by** *metis*
finally show *?thesis* **by** *simp*
qed
also have $\ldots = (\int^+ v{\in}\{..0\}.\ (\int^+ u.\ indicator\ \{..v\}\ u\ \partial ?distrX)\ \partial lborel)$
apply (*rule nn-integral-cong*)
apply (*rewrite nn-integral-multc*[*THEN sym*], *measurable*; (*simp del*: *nn-integral-indicator*) *?*)+
apply (*rule nn-integral-cong*)
using *atMost-iff atLeast-iff indicator-simps* **by** (*smt* (*verit*, *del-insts*) *mult-1 mult-eq-0-iff*)
also have $\ldots = (\int^+ v{\in}\{..0\}.\ ennreal\ (F\ v)\ \partial lborel)$
apply (*rule nn-integral-cong*, *simp*)

```
    apply (rewrite emeasure-distr; simp?)
    unfolding F-def cdf-def
    by (rewrite measure-distr; simp add: emeasure-eq-measure)
  finally show ?thesis .
qed

proposition expectation-tail:
  assumes [measurable]: integrable M X
  defines F u ≡ cdf (distr M borel X) u
  shows expectation X = (LBINT u:{0..}. 1 − F u) − (LBINT u:{..0}. F u)
proof −
  have expectation X = expectation (λx. max (X x) 0) + expectation (λx. min (X
x) 0)
    using integrable-max integrable-min
    apply (rewrite Bochner-Integration.integral-add[THEN sym], measurable)
    by (rule Bochner-Integration.integral-cong; simp)
  also have ... = expectation (λx. max (X x) 0) − expectation (λx. − min (X x)
0) by force
  also have ... = (LBINT u:{0..}. 1 − F u) − (LBINT u:{..0}. F u)
  proof −
    have expectation (λx. max (X x) 0) = (LBINT u:{0..}. 1 − F u)
    proof −
      have expectation (λx. max (X x) 0) = enn2real (∫⁺x. ennreal (max (X x)
0) ∂M)
        by (rule integral-eq-nn-integral; simp)
      also have ... = enn2real (∫⁺u∈{0..}. ennreal (1 − F u) ∂lborel)
        apply (rewrite expectation-nonneg-tail; simp?)
        apply (rewrite cdf-distr-max, simp)
        unfolding F-def
      by (metis (opaque-lifting) indicator-simps mult.commute mult-1 mult-eq-0-iff)
       also have ... = enn2real (∫⁺u. ennreal ((1 − F u) ∗ indicator {0..} u)
∂lborel)
        by (simp add: indicator-mult-ennreal mult.commute)
      also have ... = (LBINT u:{0..}. 1 − F u)
        apply (rewrite integral-eq-nn-integral[THEN sym], simp add: F-def)
        unfolding F-def using real-distribution.cdf-bounded-prob apply force
      unfolding set-lebesgue-integral-def by (rule Bochner-Integration.integral-cong;
simp)
    finally show ?thesis .
    qed
    moreover have expectation (λx. − min (X x) 0) = (LBINT u:{..0}. F u)
    proof −
      have expectation (λx. − min (X x) 0) = enn2real (∫⁺x. ennreal (− min (X
x) 0) ∂M)
        by (rule integral-eq-nn-integral; simp)
      also have ... = enn2real (∫⁺u∈{..0}. ennreal (F u) ∂lborel)
      proof −
        let ?distrminX = distr M borel (λx. min (X x) 0)
        have [simp]: sym-diff {..0} {..<0} = {0::real} by force
```

```
    have enn2real (∫⁺x. ennreal (− min (X x) 0) ∂M) =
   enn2real (∫⁺u∈{..0}. ennreal (cdf ?distrminX u) ∂lborel)
      by (rewrite expectation-nonpos-tail; simp)
    also have ... = enn2real (∫⁺u∈{..<0}. ennreal (cdf ?distrminX u) ∂lborel)
      by (rewrite nn-integral-null-delta, auto)
     also have ... = enn2real (∫⁺u∈{..<0}. ennreal (F u) ∂lborel)
      apply (rewrite cdf-distr-min, simp)
      apply (rule arg-cong[where f=enn2real], rule nn-integral-cong)
    unfolding F-def by (smt (verit) indicator-simps mult-cancel-left1 mult-eq-0-iff)
     also have ... = enn2real (∫⁺u∈{..0}. ennreal (F u) ∂lborel)
      by (rewrite nn-integral-null-delta, auto simp add: sup-commute)
     finally show ?thesis .
   qed
   also have ... = enn2real (∫⁺u. ennreal (F u * indicator {..0} u) ∂lborel)
     using mult.commute indicator-mult-ennreal by metis
   also have ... = (LBINT u:{..0}. F u)
     apply (rewrite integral-eq-nn-integral[THEN sym], simp add: F-def)
     unfolding F-def
    using finite-borel-measure.cdf-nonneg real-distribution.finite-borel-measure-M
apply simp
    unfolding set-lebesgue-integral-def by (rule Bochner-Integration.integral-cong;
simp)
    finally show ?thesis .
  qed
  ultimately show ?thesis by simp
 qed
 finally show ?thesis .
qed

proposition distributed-deriv-cdf:
 assumes [measurable]: random-variable borel X
 defines F u ≡ cdf (distr M borel X) u
 assumes finite S ⋀x. x ∉ S ⟹ (F has-real-derivative f x) (at x)
   and continuous-on UNIV F f ∈ borel-measurable lborel
 shows distributed M lborel X f
proof −
 have FBM: finite-borel-measure (distr M borel X)
   using real-distribution.finite-borel-measure-M real-distribution-distr assms by
simp
 then interpret distrX-FBM: finite-borel-measure distr M borel X .
 have FBMl: finite-borel-measure (distr M lborel X) using FBM distr-borel-lborel
by (smt (verit))
 then interpret distrlX-FBM: finite-borel-measure distr M lborel X .
 have [simp]: (λx. ennreal (f x)) ∈ borel-measurable borel using assms by simp
 moreover have distr M lborel X = density lborel f
 proof −
  have ⋀a b. a ≤ b ⟹ emeasure (distr M lborel X) {a<..b} < ⊤
    using distrlX-FBM.emeasure-real less-top-ennreal by blast
  moreover have ⋀a b. a ≤ b ⟹
```

```
    emeasure (distr M lborel X) {a<..b} = emeasure (density lborel f) {a<..b}
  proof −
    fix a b :: real assume a ≤ b
    hence [simp]: sym-diff {a<..b} {a..b} = {a} by force
     have emeasure (density lborel f) {a<..b} = (∫⁺x∈{a<..b}. ennreal (f x)
∂lborel)
      by (rewrite emeasure-density; simp)
  also have ... = (∫⁺x∈{a..b}. ennreal (f x) ∂lborel) by (rewrite nn-integral-null-delta,
auto)
    also have ... = ∫⁺x. ennreal (indicat-real {a..b} x * f x) ∂lborel
      by (metis indicator-mult-ennreal mult.commute)
    also have ... = ennreal (F b − F a)
    proof −
      define g where g x = (if x ∈ S then 0 else f x) for x :: real
      have [simp]: ⋀x. g x ≥ 0
        unfolding g-def
        apply (split if-split, auto)
        apply (rule mono-on-imp-deriv-nonneg[of UNIV F], auto)
       unfolding F-def mono-on-def using distrX-FBM.cdf-nondecreasing apply
blast
        using assms unfolding F-def by force
      have (∫⁺x. ennreal (indicat-real {a..b} x * f x) ∂lborel)
        = ∫⁺x. ennreal (indicat-real {a..b} x * g x) ∂lborel
        apply (rule nn-integral-cong-AE)
        apply (rule AE-mp[where P= λx. x ∉ S])
        using assms finite-imp-null-set-lborel AE-not-in apply blast
        unfolding g-def by simp
      also have ... = ennreal (F b − F a)
        apply (rewrite nn-integral-has-integral-lebesgue, simp)
          apply (rule fundamental-theorem-of-calculus-strong[of S], auto simp: ‹a
≤ b› g-def assms)
        using has-real-derivative-iff-has-vector-derivative assms apply presburger
        using assms continuous-on-subset subsetI by fastforce
      finally show ?thesis .
    qed
    also have ... = emeasure (distr M lborel X) {a <.. b}
      apply (rewrite distrlX-FBM.emeasure-Ioc, simp add: ‹a ≤ b›)
      unfolding F-def cdf-def
      apply (rewrite ennreal-minus[THEN sym], simp)+
      by (metis distr-borel-lborel)
    finally show emeasure (distr M lborel X) {a<..b} = emeasure (density lborel
f) {a<..b}
      by simp
  qed
  ultimately show ?thesis by (intro measure-eqI-Ioc; simp)
 qed
 ultimately show ?thesis unfolding distributed-def by auto
qed
```

end

Define the conditional probability space. This is obtained by rescaling the restricted space of a probability space.

3.2 Conditional Probability Space

lemma *restrict-prob-space*:
assumes *measure-space* Ω A μ $a \in A$
and $0 < \mu\ a$ $\mu\ a < \infty$
shows *prob-space* (*scale-measure* (*1 / μ a*) (*restrict-space* (*measure-of* Ω A μ) *a*))
proof
let *?M = measure-of* Ω A μ
let *?Ma = restrict-space ?M a*
let *?rMa = scale-measure* (*1 / μ a*) *?Ma*
have *emeasure ?rMa* (*space ?rMa*) = *1 / μ a ∗ emeasure ?Ma* (*space ?rMa*) **by** *simp*
also have ... = *1 / μ a ∗ emeasure ?M* (*space ?rMa*)
using *assms*
apply (*rewrite emeasure-restrict-space*)
apply (*simp add*: *measure-space-def sigma-algebra.sets-measure-of-eq*)
by (*simp add*: *space-restrict-space space-scale-measure*) *simp*
also have ... = *1 / μ a ∗ emeasure ?M* (*space ?Ma*) **by** (*rewrite space-scale-measure*) *simp*
also have ... = *1 / μ a ∗ emeasure ?M a*
using *assms*
apply (*rewrite space-restrict-space2*)
by (*simp add*: *measure-space-closed*)+
also have ... = *1*
using *assms measure-space-def*
apply (*rewrite emeasure-measure-of-sigma*, *blast*+)
by (*simp add*: *ennreal-divide-times*)
finally show *emeasure ?rMa* (*space ?rMa*) = *1* **.**
qed

definition *cond-prob-space* :: *'a measure ⇒ 'a set ⇒ 'a measure* (**infix** ‹⌊› *200*)
where $M \lfloor A \equiv$ *scale-measure* (*1 / emeasure M A*) (*restrict-space M A*)

context *prob-space*
begin

lemma *cond-prob-space-whole*[*simp*]: $M \lfloor$ *space* $M = M$
unfolding *cond-prob-space-def* **using** *emeasure-space-1* **by** *simp*

lemma *cond-prob-space-correct*:
assumes $A \in$ *events prob* $A > 0$
shows *prob-space* ($M \lfloor A$)
unfolding *cond-prob-space-def*

apply (*subst*(*2*) *measure-of-of-measure*[*of M*, *THEN sym*])
using *assms*
by (*intro restrict-prob-space*; (*simp add*: *measure-space*)*?*; *simp-all add*: *emeasure-eq-measure*)

lemma *space-cond-prob-space*:
assumes $A \in$ *events*
shows *space* $(M\lfloor A) = A$
unfolding *cond-prob-space-def* **using** *assms* **by** (*simp add*: *space-scale-measure*)

lemma *sets-cond-prob-space*: *sets* $(M\lfloor A) = (\cap)\ A$ ‘ *events*
unfolding *cond-prob-space-def* **by** (*metis sets-restrict-space sets-scale-measure*)

lemma *measure-cond-prob-space*:
assumes $A \in$ *events* $B \in$ *events*
and *prob* $A > 0$
shows *measure* $(M\lfloor A)\ (B \cap A) =$ *prob* $(B \cap A)$ / *prob* A
proof −
have *1* / *emeasure* $M\ A =$ *ennreal* $(1$ / *prob* $A)$
using *assms* **by** (*smt* (*verit*) *divide-ennreal emeasure-eq-measure ennreal-1*)
hence *measure* $(M\lfloor A)\ (B \cap A) = (1$ / *prob* $A) *$ *measure* (*restrict-space* $M\ A$) $(B \cap A)$
unfolding *cond-prob-space-def* **using** *measure-scale-measure* **by** *force*
also have $\ldots = (1$ / *prob* $A) *$ *prob* $(B \cap A)$
using *measure-restrict-space assms* **by** (*metis inf.cobounded2 sets.Int-space-eq2*)
also have $\ldots =$ *prob* $(B \cap A)$ / *prob* A **by** *simp*
finally show *?thesis* **.**
qed

corollary *measure-cond-prob-space-subset*:
assumes $A \in$ *events* $B \in$ *events* $B \subseteq A$
and *prob* $A > 0$
shows *measure* $(M\lfloor A)\ B =$ *prob* B / *prob* A
proof −
have $B = B \cap A$ **using** *assms* **by** *auto*
moreover have *measure* $(M\lfloor A)\ (B \cap A) =$ *prob* $(B \cap A)$ / *prob* A
using *assms measure-cond-prob-space* **by** *simp*
ultimately show *?thesis* **by** *auto*
qed

lemma *cond-cond-prob-space*:
assumes $A \in$ *events* $B \in$ *events* $B \subseteq A$ *prob* $B > 0$
shows $(M\lfloor A)\lfloor B = M\lfloor B$
proof (*rule measure-eqI*)
have *pA-pos*[*simp*]: *prob* $A > 0$ **using** *assms* **by** (*smt* (*verit*, *ccfv-SIG*) *finite-measure-mono*)
interpret *MA-PS*: *prob-space* $M\lfloor A$ **using** *cond-prob-space-correct assms* **by** *simp*
interpret *MB-PS*: *prob-space* $M\lfloor B$ **using** *cond-prob-space-correct assms* **by** *simp*
have *1* / *emeasure* $M\ A =$ *ennreal* $(1$ / *prob* $A)$

using *pA-pos* **by** (*smt* (*verit*, *ccfv-SIG*) *divide-ennreal emeasure-eq-measure ennreal-1*)
hence [*simp*]: $0 < MA\text{-}PS.prob\ B$
using *assms pA-pos*
by (*metis divide-eq-0-iff measure-cond-prob-space-subset zero-less-measure-iff*)
have [*simp*]: $B \in MA\text{-}PS.events$
using *assms* **by** (*rewrite sets-cond-prob-space*, *unfold image-def*) *blast*
have [*simp*]: *finite-measure* $((M\lfloor A)\lfloor B)$
by (*rule prob-space.finite-measure*, *rule prob-space.cond-prob-space-correct*,
simp-all add: *MA-PS.prob-space-axioms*)
show *sets-MAB*: *sets* $((M\lfloor A)\lfloor B) = sets\ (M\lfloor B)$
apply (*rewrite prob-space.sets-cond-prob-space*)
using *MA-PS.prob-space-axioms* **apply** *presburger*
apply (*rewrite sets-cond-prob-space*, *unfold image-def*)+
using *assms* **by** *blast*
show $\bigwedge C.\ C \in sets\ ((M\lfloor A)\lfloor B) \Longrightarrow emeasure\ ((M\lfloor A)\lfloor B)\ C = emeasure\ (M\lfloor B)\ C$
proof −
fix C **assume** $C \in sets\ ((M\lfloor A)\lfloor B)$
hence $C \in sets\ (M\lfloor B)$ **using** *sets-MAB* **by** *simp*
from *this* **obtain** D **where** $D \in events\ C = B \cap D$
by (*rewrite* **in** *asm sets-cond-prob-space*, *auto*)
hence [*simp*]: $C \in events$ **and** [*simp*]: $C \subseteq B$ **and** [*simp*]: $C \subseteq A$ **using** *assms* **by** *auto*
hence [*simp*]: $C \in MA\text{-}PS.events$
using *assms* **by** (*rewrite sets-cond-prob-space*, *unfold image-def*) *blast*
show *emeasure* $((M\lfloor A)\lfloor B)\ C = emeasure\ (M\lfloor B)\ C$
apply (*rewrite finite-measure.emeasure-eq-measure*, *simp*)+
apply (*rewrite ennreal-inj*, *simp-all*)
apply (*rewrite prob-space.measure-cond-prob-space-subset*,
simp-all add: *assms prob-space-axioms MA-PS.prob-space-axioms*)+
using *pA-pos* **by** *fastforce*
qed
qed

lemma *cond-prob-space-prob*:
assumes[*measurable*]: *Measurable.pred M P Measurable.pred M Q*
and $\mathcal{P}(x\ in\ M.\ Q\ x) > 0$
shows *measure* $(M \lfloor \{x \in space\ M.\ Q\ x\})\ \{x \in space\ M.\ P\ x \wedge Q\ x\} = \mathcal{P}(x\ in\ M.\ P\ x \mid Q\ x)$
proof −
let *?SetP* $= \{x \in space\ M.\ P\ x\}$
let *?SetQ* $= \{x \in space\ M.\ Q\ x\}$
have *measure* $(M\lfloor ?SetQ)\ \{x \in space\ M.\ P\ x \wedge Q\ x\} = measure\ (M\lfloor ?SetQ)\ (?SetP \cap ?SetQ)$
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong Int-def mem-Collect-eq*)
also have $\ldots$ = *prob* $(?SetP \cap ?SetQ)$ / *prob ?SetQ*
using *assms* **by** (*rewrite measure-cond-prob-space*; *simp?*)
also have $\ldots = \mathcal{P}(x\ in\ M.\ P\ x \mid Q\ x)$

unfolding *cond-prob-def assms* **by** (*smt* (*verit*) *Collect-cong Int-def mem-Collect-eq*)
finally show *?thesis* **.**
qed

lemma *cond-prob-space-cond-prob*:
assumes [*measurable*]: *Measurable.pred M P Measurable.pred M Q*
and $\mathcal{P}(x \text{ in } M.\ Q\ x) > 0$
shows $\mathcal{P}(x \text{ in } M.\ P\ x \mid Q\ x) = \mathcal{P}(x \text{ in } (M \lfloor \{x \in \text{space } M.\ Q\ x\}).\ P\ x)$
proof −
let *?setQ* $= \{x \in \text{space } M.\ Q\ x\}$
have $\mathcal{P}(x \text{ in } M.\ P\ x \mid Q\ x) = \text{measure } (M \lfloor ?setQ)\ \{x \in \text{space } M.\ P\ x \wedge Q\ x\}$
using *cond-prob-space-prob assms* **by** *simp*
also have $\ldots = \mathcal{P}(x \text{ in } (M \lfloor ?setQ).\ P\ x)$
proof −
have $\{x \in \text{space } M.\ P\ x \wedge Q\ x\} = \{x \in \text{space } (M \lfloor ?setQ).\ P\ x\}$
using *space-cond-prob-space assms* **by** *force*
thus *?thesis* **by** *simp*
qed
finally show *?thesis* **.**
qed

lemma *cond-prob-neg*:
assumes[*measurable*]: *Measurable.pred M P Measurable.pred M Q*
and $\mathcal{P}(x \text{ in } M.\ Q\ x) > 0$
shows $\mathcal{P}(x \text{ in } M.\ \neg\ P\ x \mid Q\ x) = 1 - \mathcal{P}(x \text{ in } M.\ P\ x \mid Q\ x)$
proof −
let *?setP* $= \{x \in \text{space } M.\ P\ x\}$
let *?setQ* $= \{x \in \text{space } M.\ Q\ x\}$
interpret *setQ-PS*: *prob-space M* $\lfloor$ *?setQ* **using** *cond-prob-space-correct assms* **by** *simp*
have [*simp*]: $\{x \in \text{space } (M \lfloor ?setQ).\ P\ x\} \in$ *setQ-PS.events*
proof −
have $\{x \in \text{space } (M \lfloor ?setQ).\ P\ x\} = ?setQ \cap ?setP$ **using** *space-cond-prob-space* **by** *force*
thus *?thesis* **using** *sets-cond-prob-space* **by** *simp*
qed
have $\mathcal{P}(x \text{ in } M.\ \neg\ P\ x \mid Q\ x) = \mathcal{P}(x \text{ in } M \lfloor ?setQ.\ \neg\ P\ x)$
by (*rewrite cond-prob-space-cond-prob*; *simp add*: *assms*)
also have $\ldots = 1 - \mathcal{P}(x \text{ in } M \lfloor ?setQ.\ P\ x)$ **by** (*rewrite setQ-PS.prob-neg*, *simp-all add*: *assms*)
also have $\ldots = 1 - \mathcal{P}(x \text{ in } M.\ P\ x \mid Q\ x)$
by (*rewrite cond-prob-space-cond-prob*; *simp add*: *assms*)
finally show *?thesis* **.**
qed

lemma *random-variable-cond-prob-space*:
assumes $A \in$ *events prob* $A > 0$
and [*measurable*]: *random-variable borel X*
shows $X \in$ *borel-measurable* $(M \lfloor A)$

proof (*rule borel-measurableI*)
fix $S :: 'b$ *set*
assume [*measurable*]: *open S*
show $X -` S \cap$ *space* $(M \restriction A) \in$ *sets* $(M \restriction A)$
apply (*rewrite space-cond-prob-space*, *simp add*: *assms*)
apply (*rewrite sets-cond-prob-space*, *simp add*: *image-def*)
apply (*rule bexI*[*of* - $X -` S \cap$ *space M*]; *measurable*)
using *sets.Int-space-eq2 Int-commute assms* **by** *auto*
qed

lemma *AE-cond-prob-space-iff*:
assumes $A \in$ *events prob* $A > 0$
shows (*AE x in* $M \restriction A$. $P\ x$) $\longleftrightarrow$ (*AE x in M*. $x \in A \longrightarrow P\ x$)
proof −
have [*simp*]: *1 / emeasure M A* > 0
using *assms divide-ennreal emeasure-eq-measure ennreal-1*
by (*smt* (*verit*) *divide-pos-pos ennreal-eq-0-iff not-gr-zero*)
show *?thesis*
unfolding *cond-prob-space-def*
apply (*rewrite AE-scale-measure-iff*, *simp*)
by (*rewrite AE-restrict-space-iff*; *simp add*: *assms*)
qed

lemma *integral-cond-prob-space-nn*:
assumes $A \in$ *events prob* $A > 0$
and [*measurable*]: *random-variable borel X*
and *nonneg*: *AE x in M*. $x \in A \longrightarrow 0 \leq X\ x$
shows $integral^L\ (M \restriction A)\ X =$ *expectation* $(\lambda x.$ *indicator A* $x * X\ x)$ */ prob A*
proof −
have [*simp*]: $X \in$ *borel-measurable* $(M \restriction A)$
by (*rule random-variable-cond-prob-space*; (*simp add*: *assms*))
have [*simp*]: *AE x in* $(M \restriction A)$. $0 \leq X\ x$
by (*rewrite AE-cond-prob-space-iff*; *simp add*: *assms*)
have [*simp*]: *random-variable borel* $(\lambda x.$ *indicat-real A* $x * X\ x)$
using *borel-measurable-indicator assms* **by** *force*
have [*simp*]: *AE x in M*. $0 \leq$ *indicat-real A* $x * X\ x$ **using** *nonneg* **by** *fastforce*
have $integral^L\ (M \restriction A)\ X =$ *enn2real* $(\int^+ x.$ *ennreal* $(X\ x)\ \partial(M \restriction A))$
by (*rewrite integral-eq-nn-integral*; *simp*)
also have ... = *enn2real* (*1 / prob A* $* \int^+ x.$ *ennreal* $(X\ x)\ \partial($*restrict-space M A*$))$
unfolding *cond-prob-space-def*
apply (*rewrite nn-integral-scale-measure*, *simp add*: *measurable-restrict-space1*)
using *divide-ennreal emeasure-eq-measure ennreal-1 assms* **by** (*smt* (*verit*))
also have ... = *enn2real* (*1 / prob A* $* (\int^+ x.$ *ennreal* (*indicator A x* $* X\ x)$ $\partial M))$
apply (*rewrite nn-integral-restrict-space*, *simp add*: *assms*)
by (*metis indicator-mult-ennreal mult.commute*)
also have ... = $integral^L$ *M* $(\lambda x.$ *indicator A* $x * X\ x)$ */ prob A*
apply (*rewrite integral-eq-nn-integral*; *simp?*)

by (*simp add*: *divide-nonneg-pos enn2real-mult*)
finally show *?thesis* **by** *simp*
qed

end

Define the complementary cumulative distribution function, also known as tail distribution. The theory presented below is a slight modification of the subsection "Properties of cdf's" in the theory *Distribution-Functions*.

3.3 Complementary Cumulative Distribution Function

definition *ccdf* :: *real measure* ⇒ *real* ⇒ *real*
where *ccdf M* ≡ λ*x*. *measure M* {*x*<..}
— complementary cumulative distribution function (tail distribution)

lemma *ccdf-def2*: *ccdf M x* = *measure M* {*x*<..}
by (*simp add*: *ccdf-def*)

context *finite-borel-measure*
begin

lemma *add-cdf-ccdf*: *cdf M x* + *ccdf M x* = *measure M* (*space M*)
proof −
have {..*x*} ∪ {*x*<..} = *UNIV* **by** *auto*
moreover have {..*x*} ∩ {*x*<..} = {} **by** *auto*
ultimately have *emeasure M* {..*x*} + *emeasure M* {*x*<..} = *emeasure M UNIV*
using *plus-emeasure M-is-borel atMost-borel greaterThan-borel* **by** *metis*
hence *measure M* {..*x*} + *measure M* {*x*<..} = *measure M UNIV*
using *finite-emeasure-space emeasure-eq-measure ennreal-inj*
by (*smt* (*verit*, *ccfv-SIG*) *ennreal-plus measure-nonneg*)
thus *?thesis* **unfolding** *cdf-def ccdf-def* **using** *borel-UNIV* **by** *simp*
qed

lemma *ccdf-cdf*: *ccdf M* = (λ*x*. *measure M* (*space M*) − *cdf M x*)
by (*rule ext*) (*smt* (*verit*) *add-cdf-ccdf*)

lemma *cdf-ccdf*: *cdf M* = (λ*x*. *measure M* (*space M*) − *ccdf M x*)
by (*rule ext*) (*smt* (*verit*) *add-cdf-ccdf*)

lemma *isCont-cdf-ccdf*: *isCont* (*cdf M*) *x* ⟷ *isCont* (*ccdf M*) *x*
proof
show *isCont* (*cdf M*) *x* ⟹ *isCont* (*ccdf M*) *x* **by** (*rewrite ccdf-cdf*) *auto*
next
show *isCont* (*ccdf M*) *x* ⟹ *isCont* (*cdf M*) *x* **by** (*rewrite cdf-ccdf*) *auto*
qed

lemma *isCont-ccdf*: *isCont* (*ccdf M*) *x* ⟷ *measure M* {*x*} = *0*
using *isCont-cdf-ccdf isCont-cdf* **by** *simp*

lemma *continuous-cdf-ccdf*:
shows *continuous F* (*cdf M*) ⟷ *continuous F* (*ccdf M*)
(**is** *?LHS* ⟷ *?RHS*)
proof
assume *?LHS*
thus *?RHS* **using** *continuous-diff continuous-const* **by** (*rewrite ccdf-cdf*) *blast*
next
assume *?RHS*
thus *?LHS* **using** *continuous-diff continuous-const* **by** (*rewrite cdf-ccdf*) *blast*
qed

lemma *has-real-derivative-cdf-ccdf*:
(*cdf M has-real-derivative D*) *F* ⟷ (*ccdf M has-real-derivative* −*D*) *F*
proof
assume (*cdf M has-real-derivative D*) *F*
thus (*ccdf M has-real-derivative* −*D*) *F*
using *ccdf-cdf DERIV-const Deriv.field-differentiable-diff* **by** *fastforce*
next
assume (*ccdf M has-real-derivative* −*D*) *F*
thus (*cdf M has-real-derivative D*) *F*
using *cdf-ccdf DERIV-const Deriv.field-differentiable-diff* **by** *fastforce*
qed

lemma *differentiable-cdf-ccdf*: (*cdf M differentiable F*) ⟷ (*ccdf M differentiable F*)
unfolding *differentiable-def*
apply (*rewrite has-real-derivative-iff*[*THEN sym*])+
apply (*rewrite has-real-derivative-cdf-ccdf*)
by (*metis verit-minus-simplify*(4))

lemma *deriv-cdf-ccdf*:
assumes *cdf M differentiable at x*
shows *deriv* (*cdf M*) *x* = − *deriv* (*ccdf M*) *x*
using *has-real-derivative-cdf-ccdf differentiable-cdf-ccdf assms*
by (*simp add*: *DERIV-deriv-iff-real-differentiable DERIV-imp-deriv*)

lemma *ccdf-diff-eq2*:
assumes *x* ≤ *y*
shows *ccdf M x* − *ccdf M y* = *measure M* {*x*<..*y*}
proof −
have *ccdf M x* − *ccdf M y* = *cdf M y* − *cdf M x* **using** *add-cdf-ccdf* **by** (*smt* (*verit*))
also have ... = *measure M* {*x*<..*y*} **using** *cdf-diff-eq2 assms* **by** *simp*
finally show *?thesis* **.**
qed

lemma *ccdf-nonincreasing*: *x* ≤ *y* ⟹ *ccdf M x* ≥ *ccdf M y*
using *add-cdf-ccdf cdf-nondecreasing* **by** (*smt* (*verit*))

lemma *ccdf-nonneg*: *ccdf M x* $\geq$ *0*
using *add-cdf-ccdf cdf-bounded* **by** (*smt* (*verit*))

lemma *ccdf-bounded*: *ccdf M x* $\leq$ *measure M* (*space M*)
using *add-cdf-ccdf cdf-nonneg* **by** (*smt* (*verit*))

lemma *ccdf-lim-at-top*: (*ccdf M* $\longrightarrow$ *0*) *at-top*
proof −
have ((*λx*. *measure M* (*space M*) − *cdf M x*) $\longrightarrow$ *measure M* (*space M*) − *measure M* (*space M*)) *at-top*
apply (*intro tendsto-intros*)
by (*rule cdf-lim-at-top*)
thus *?thesis*
by (*rewrite ccdf-cdf*) *simp*
qed

lemma *ccdf-lim-at-bot*: (*ccdf M* $\longrightarrow$ *measure M* (*space M*)) *at-bot*
proof −
have ((*λx*. *measure M* (*space M*) − *cdf M x*) $\longrightarrow$ *measure M* (*space M*) − *0*) *at-bot*
apply (*intro tendsto-intros*)
by (*rule cdf-lim-at-bot*)
thus *?thesis*
by (*rewrite ccdf-cdf*) *simp*
qed

lemma *ccdf-is-right-cont*: *continuous* (*at-right a*) (*ccdf M*)
proof −
have *continuous* (*at-right a*) (*λx*. *measure M* (*space M*) − *cdf M x*)
apply (*intro continuous-intros*)
by (*rule cdf-is-right-cont*)
thus *?thesis* **by** (*rewrite ccdf-cdf*) *simp*
qed

end

context *prob-space*
begin

lemma *ccdf-distr-measurable* [*measurable*]:
assumes [*measurable*]: *random-variable borel X*
shows *ccdf* (*distr M borel X*) $\in$ *borel-measurable borel*
using *real-distribution.finite-borel-measure-M* **by** (*rewrite finite-borel-measure.ccdf-cdf*; *simp*)

lemma *ccdf-distr-P*:
assumes *random-variable borel X*
shows *ccdf* (*distr M borel X*) *x* = $\mathcal{P}$(*ω in M*. *X ω* > *x*)

unfolding *ccdf-def* **apply** (*rewrite measure-distr*; (*simp add*: *assms*)*?*)
unfolding *vimage-def* **by** (*rule arg-cong*[**where** *f*=*prob*]) *force*

lemma *ccdf-continuous-distr-P-ge*:
assumes *random-variable borel X isCont* (*ccdf* (*distr M borel X*)) *x*
shows *ccdf* (*distr M borel X*) *x* = $\mathcal{P}$(*ω in M*. *X ω* ≥ *x*)
proof −
have *ccdf* (*distr M borel X*) *x* = *measure* (*distr M borel X*) {*x*<..} **unfolding** *ccdf-def* **by** *simp*
also have ... = *measure* (*distr M borel X*) ({*x*<..} ∪ {*x*})
apply (*rewrite finite-measure.measure-zero-union*, *simp-all add*: *assms finite-measure-distr*)
using *finite-borel-measure.isCont-ccdf real-distribution.finite-borel-measure-M assms* **by** *blast*
also have ... = *measure* (*distr M borel X*) {*x*..} **by** (*metis Un-commute ivl-disj-un-singleton*(*1*))
also have ... = $\mathcal{P}$(*ω in M*. *X ω* ≥ *x*)
apply (*rewrite measure-distr*, *simp-all add*: *assms*)
unfolding *vimage-def* **by** *simp* (*smt* (*verit*) *Collect-cong Int-def mem-Collect-eq*)
finally show *?thesis* **.**
qed

lemma *ccdf-distr-diff-P*:
assumes *x* ≤ *y*
and *random-variable borel X*
shows *ccdf* (*distr M borel X*) *x* − *ccdf* (*distr M borel X*) *y* = $\mathcal{P}$(*ω in M*. *x* < *X ω* ∧ *X ω* ≤ *y*)
proof −
interpret *distrX-FBM*: *finite-borel-measure distr M borel X*
using *real-distribution.finite-borel-measure-M real-distribution-distr assms* **by** *simp*
have *ccdf* (*distr M borel X*) *x* − *ccdf* (*distr M borel X*) *y* = *measure* (*distr M borel X*) {*x*<..*y*}
by (*rewrite distrX-FBM.ccdf-diff-eq2*; *simp add*: *assms*)
also have ... = $\mathcal{P}$(*ω in M*. *x* < *X ω* ∧ *X ω* ≤ *y*)
apply (*rewrite measure-distr*; (*simp add*: *assms*)*?*)
unfolding *vimage-def* **by** (*rule arg-cong*[**where** *f*=*prob*], *force*)
finally show *?thesis* **.**
qed

end

context *real-distribution*
begin

lemma *ccdf-bounded-prob*: ⋀*x*. *ccdf M x* ≤ *1*
by (*subst prob-space*[*THEN sym*], *rule ccdf-bounded*)

lemma *ccdf-lim-at-bot-prob*: (*ccdf M* ⟶ *1*) *at-bot*
by (*subst prob-space*[*THEN sym*], *rule ccdf-lim-at-bot*)

end

Introduce the hazard rate. This notion will be used to define the force of mortality.

3.4 Hazard Rate

context *prob-space*
begin

definition *hazard-rate* :: (*'a* ⇒ *real*) ⇒ *real* ⇒ *real*
  where *hazard-rate X t* ≡
    Lim (*at-right 0*) (λ*dt*. *P*(*x in M*. *t* < *X x* ∧ *X x* ≤ *t* + *dt* | *X x* > *t*) / *dt*)
    — Here, *X* is supposed to be a random variable.

lemma *hazard-rate-0-ccdf-0*:
  assumes [*measurable*]: *random-variable borel X*
    and *ccdf* (*distr M borel X*) *t* = *0*
  shows *hazard-rate X t* = *0*
  — Note that division by *0* is calculated as *0* in Isabelle/HOL.
proof −
  have ⋀*dt*. *P*(*x in M*. *t* < *X x* ∧ *X x* ≤ *t* + *dt* | *X x* > *t*) = *0*
    unfolding *cond-prob-def* **using** *ccdf-distr-P assms* **by** *simp*
  hence *hazard-rate X t* = *Lim* (*at-right 0*) (λ*dt*::*real*. *0*)
    unfolding *hazard-rate-def* **by** (*rewrite Lim-cong*; *simp*)
  also have ... = *0* **by** (*rule tendsto-Lim*; *simp*)
  finally show *?thesis* **.**
qed

lemma *hazard-rate-deriv-cdf*:
  assumes [*measurable*]: *random-variable borel X*
    and (*cdf* (*distr M borel X*)) *differentiable at t*
  shows *hazard-rate X t* = *deriv* (*cdf* (*distr M borel X*)) *t* / *ccdf* (*distr M borel X*) *t*
proof (*cases* ‹*ccdf* (*distr M borel X*) *t* = *0*›)
  case *True*
  with *hazard-rate-0-ccdf-0* **show** *?thesis* **by** *simp*
next
  case *False*
  let *?F* = *cdf* (*distr M borel X*)
  have ∀ *F dt in at-right 0*. *P*(*x in M*. *t* < *X x* ∧ *X x* ≤ *t* + *dt* | *X x* > *t*) / *dt* =
    (*?F* (*t* + *dt*) − *?F t*) / *dt* / *ccdf* (*distr M borel X*) *t*
    apply (*rule eventually-at-rightI*[**where** *b*=*1*]; *simp*)
    unfolding *cond-prob-def*
    apply (*rewrite cdf-distr-diff-P*; *simp*)
    apply (*rewrite ccdf-distr-P*[*THEN sym*], *simp*)
    by (*smt* (*verit*) *Collect-cong mult.commute*)
  moreover have ((λ*dt*. (*?F* (*t* + *dt*) − *?F t*) / *dt* / *ccdf* (*distr M borel X*) *t*)
⟶

```
    deriv ?F t / ccdf (distr M borel X) t) (at-right 0)
    apply (rule tendsto-intros, simp-all add: False)
    apply (rule Lim-at-imp-Lim-at-within)
    using DERIV-deriv-iff-real-differentiable assms DERIV-def by blast
  ultimately show ?thesis
    unfolding hazard-rate-def using tendsto-cong by (intro tendsto-Lim; force)
qed

lemma deriv-cdf-hazard-rate:
  assumes [measurable]: random-variable borel X
    and (cdf (distr M borel X)) differentiable at t
  shows deriv (cdf (distr M borel X)) t = ccdf (distr M borel X) t * hazard-rate
X t
proof −
  interpret distrX-FBM: finite-borel-measure distr M borel X
    using real-distribution.finite-borel-measure-M real-distribution-distr assms by
simp
  show ?thesis
  proof (cases ‹ccdf (distr M borel X) t = 0›)
    case True
    hence cdf (distr M borel X) t = 1
      using distrX-FBM.cdf-ccdf
        by simp (metis assms(1) distrX-FBM.borel-UNIV prob-space.prob-space
prob-space-distr)
    moreover obtain D where (cdf (distr M borel X) has-real-derivative D) (at
t)
      using assms real-differentiable-def by atomize-elim blast
    ultimately have (cdf (distr M borel X) has-real-derivative 0) (at t)
      using assms
    by (smt (verit) DERIV-local-max real-distribution.cdf-bounded-prob real-distribution-distr)
    thus ?thesis using True by (simp add: DERIV-imp-deriv)
  next
    case False
    thus ?thesis using hazard-rate-deriv-cdf assms by simp
  qed
qed

lemma hazard-rate-deriv-ccdf:
  assumes [measurable]: random-variable borel X
    and (ccdf (distr M borel X)) differentiable at t
  shows hazard-rate X t = − deriv (ccdf (distr M borel X)) t / ccdf (distr M borel
X) t
proof −
  interpret distrX-FBM: finite-borel-measure distr M borel X
    using real-distribution.finite-borel-measure-M real-distribution-distr assms by
simp
  show ?thesis
   using hazard-rate-deriv-cdf distrX-FBM.deriv-cdf-ccdf assms distrX-FBM.differentiable-cdf-ccdf
    by presburger
```

qed

lemma *deriv-ccdf-hazard-rate*:
assumes [*measurable*]: *random-variable borel X*
and (*ccdf* (*distr M borel X*)) *differentiable at t*
shows *deriv* (*ccdf* (*distr M borel X*)) *t* = − *ccdf* (*distr M borel X*) *t* ∗ *hazard-rate X t*
proof −
interpret *distrX-FBM*: *finite-borel-measure distr M borel X*
using *real-distribution.finite-borel-measure-M real-distribution-distr assms* **by** *simp*
show *?thesis*
using *deriv-cdf-hazard-rate distrX-FBM.deriv-cdf-ccdf assms distrX-FBM.differentiable-cdf-ccdf*
by *simp*
qed

lemma *hazard-rate-deriv-ln-ccdf*:
assumes [*measurable*]: *random-variable borel X*
and (*ccdf* (*distr M borel X*)) *differentiable at t*
and *ccdf* (*distr M borel X*) *t* ≠ *0*
shows *hazard-rate X t* = − *deriv* (λ*t*. *ln* (*ccdf* (*distr M borel X*) *t*)) *t*
proof −
interpret *distrX-FBM*: *finite-borel-measure distr M borel X*
using *real-distribution.finite-borel-measure-M real-distribution-distr assms* **by** *simp*
let *?srvl* = *ccdf* (*distr M borel X*)
have *?srvl t* > *0* **using** *distrX-FBM.ccdf-nonneg assms* **by** (*smt* (*verit*))
moreover have (*?srvl has-real-derivative* (*deriv ?srvl t*)) (*at t*)
using *DERIV-deriv-iff-real-differentiable assms* **by** *blast*
ultimately have ((λ*t*. *ln* (*?srvl t*)) *has-real-derivative* *1* / *?srvl t* ∗ *deriv ?srvl t*) (*at t*)
by (*rule derivative-intros*)
hence *deriv* (λ*t*. *ln* (*?srvl t*)) *t* = *deriv ?srvl t* / *?srvl t* **by** (*simp add*: *DERIV-imp-deriv*)
also have ... = − *hazard-rate X t* **using** *hazard-rate-deriv-ccdf assms* **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *hazard-rate-has-integral*:
assumes [*measurable*]: *random-variable borel X*
and *t* ≤ *u*
and (*ccdf* (*distr M borel X*)) *piecewise-differentiable-on* {*t*<..<*u*}
and *continuous-on* {*t*..*u*} (*ccdf* (*distr M borel X*))
and ⋀*s*. *s* ∈ {*t*..*u*} ⟹ *ccdf* (*distr M borel X*) *s* ≠ *0*
shows
(*hazard-rate X has-integral ln* (*ccdf* (*distr M borel X*) *t* / *ccdf* (*distr M borel X*) *u*)) {*t*..*u*}
proof −
interpret *distrX-FBM*: *finite-borel-measure distr M borel X*

```
    using real-distribution.finite-borel-measure-M real-distribution-distr assms by
simp
  let ?srvl = ccdf (distr M borel X)
  have [simp]: ⋀s. t ≤ s ∧ s ≤ u ⟹ ?srvl s > 0
    using distrX-FBM.ccdf-nonneg assms by (smt (verit) atLeastAtMost-iff)
  have (deriv (λs. − ln (?srvl s)) has-integral − ln (?srvl u) − − ln (?srvl t))
{t..u}
  proof −
    have continuous-on {t..u} (λs. − ln (?srvl s))
      by (rule continuous-intros, rule continuous-on-ln, auto simp add: assms)
    moreover hence (λs. − ln (?srvl s)) piecewise-differentiable-on {t<..<u}
    proof −
      have ?srvl ‘ {t<..<u} ⊆ {0<..}
      proof −
        { fix s assume s ∈ {t<..<u}
          hence ?srvl s ≠ 0 using assms by simp
          moreover have ?srvl s ≥ 0 using distrX-FBM.ccdf-nonneg by simp
          ultimately have ?srvl s > 0 by simp }
        thus ?thesis by auto
      qed
      hence (λr. − ln r) ∘ ?srvl piecewise-differentiable-on {t<..<u}
        apply (intro differentiable-on-piecewise-compose, simp add: assms)
        apply (rule derivative-intros)
        apply (rule differentiable-on-subset[of ln {0<..}], simp-all)
        apply (rewrite differentiable-on-eq-field-differentiable-real, auto)
      unfolding field-differentiable-def using DERIV-ln by (metis has-field-derivative-at-within)
      thus ?thesis unfolding comp-def by simp
    qed
     ultimately show ?thesis by (intro FTC-real-deriv-has-integral; simp add:
assms)
  qed
  hence ln: (deriv (λs. − ln (?srvl s)) has-integral ln (?srvl t / ?srvl u)) {t..u}
    by simp (rewrite ln-div; force simp: assms)
  thus ((hazard-rate X) has-integral ln (?srvl t / ?srvl u)) {t..u}
  proof −
    from assms obtain S0 where finS0: finite S0 and
      diffS0: ⋀s. s ∈ {t<..<u} − S0 ⟹ ?srvl differentiable at s within {t<..<u}
      unfolding piecewise-differentiable-on-def by blast
    from this obtain S where finite S and ⋀s. s ∈ {t..u} − S ⟹ ?srvl differ-
entiable at s
    proof (atomize-elim)
      let ?S = S0 ∪ {t, u}
      have finite ?S using finS0 by simp
     moreover have ∀ s. s ∈ {t..u} − ?S ⟶ ccdf (distr M borel X) differentiable
at s
      proof −
        { fix s assume s-in: s ∈ {t..u} − ?S
          hence ?srvl differentiable at s within {t<..<u} using diffS0 by simp
          hence ?srvl differentiable at s
```

```
        using s-in by (rewrite at-within-open[THEN sym], simp-all) }
      thus ?thesis by blast
    qed
    ultimately show
     ∃ S. finite S ∧ (∀ s. s ∈ {t..u} − S ⟶ ccdf (distr M borel X) differentiable
at s)
      by blast
  qed
  thus ?thesis
    apply (rewrite has-integral-spike-finite-eq [of S - deriv (λs. − ln (?srvl s))],
simp)
     apply (rewrite hazard-rate-deriv-ln-ccdf, simp-all add: assms)
     apply (rewrite deriv-minus, simp-all)
     apply (rewrite in asm differentiable-eq-field-differentiable-real)
      apply (rewrite comp-def[THEN sym], rule field-differentiable-compose[of
?srvl], simp-all)
    unfolding field-differentiable-def apply (rule exI, rule DERIV-ln, simp)
    using ln by simp
 qed
qed

corollary hazard-rate-integrable:
 assumes [measurable]: random-variable borel X
   and t ≤ u
   and (ccdf (distr M borel X)) piecewise-differentiable-on {t<..<u}
   and continuous-on {t..u} (ccdf (distr M borel X))
   and ⋀s. s ∈ {t..u} ⟹ ccdf (distr M borel X) s ≠ 0
 shows hazard-rate X integrable-on {t..u}
 using has-integral-integrable hazard-rate-has-integral assms by blast

lemma ccdf-exp-cumulative-hazard:
 assumes [measurable]: random-variable borel X
   and t ≤ u
   and (ccdf (distr M borel X)) piecewise-differentiable-on {t<..<u}
   and continuous-on {t..u} (ccdf (distr M borel X))
   and ⋀s. s ∈ {t..u} ⟹ ccdf (distr M borel X) s ≠ 0
 shows ccdf (distr M borel X) u / ccdf (distr M borel X) t =
   exp (− integral {t..u} (hazard-rate X))
proof −
 interpret distrX-FBM: finite-borel-measure distr M borel X
   using real-distribution.finite-borel-measure-M real-distribution-distr assms by
simp
 let ?srvl = ccdf (distr M borel X)
 have [simp]: ⋀s. t ≤ s ∧ s ≤ u ⟹ ?srvl s > 0
   using distrX-FBM.ccdf-nonneg assms by (smt (verit) atLeastAtMost-iff)
 have integral {t..u} (hazard-rate X) = ln (?srvl t / ?srvl u)
   using hazard-rate-has-integral has-integral-integrable-integral assms by auto
 also have ... = − ln (?srvl u / ?srvl t) using ln-div assms by simp
 finally have − integral {t..u} (hazard-rate X) = ln (?srvl u / ?srvl t) by simp
```

```
  thus ?thesis using assms by simp
qed

lemma hazard-rate-density-ccdf:
  assumes distributed M lborel X f
    and ⋀s. f s ≥ 0 t < u continuous-on {t..u} f
  shows hazard-rate X t = f t / ccdf (distr M borel X) t
proof (cases ‹ccdf (distr M borel X) t = 0›)
  case True
  thus ?thesis
    apply (rewrite hazard-rate-0-ccdf-0, simp-all)
    using assms(1) distributed-measurable by force
next
  case False
  have [simp]: t ≤ u using assms by simp
  have [measurable]: random-variable borel X
    using assms distributed-measurable measurable-lborel1 by blast
  have [measurable]: f ∈ borel-measurable lborel
    using assms distributed-real-measurable by metis
  have [simp]: integrable lborel f
  proof −
    have prob (X −‘ UNIV ∩ space M) = LINT x|lborel. indicat-real UNIV x * f x
      by (rule distributed-measure; simp add: assms)
    thus ?thesis
      using prob-space not-integrable-integral-eq by fastforce
  qed
  have ((λdt. (LBINT s:{t..t+dt}. f s) / dt) ⟶ f t) (at-right 0)
  proof −
    have ⋀dt. (∫⁺ x. ennreal (indicat-real {t..t+dt} x * f x) ∂lborel) < ∞
    proof −
      fix dt :: real
      have (∫⁺ x. ennreal (indicat-real {t..t+dt} x * f x) ∂lborel) =
        set-nn-integral lborel {t..t+dt} f
        by (metis indicator-mult-ennreal mult.commute)
      moreover have emeasure M (X −‘ {t..t+dt} ∩ space M) = set-nn-integral
lborel {t..t+dt} f
        by (rule distributed-emeasure; simp add: assms)
      moreover have emeasure M (X −‘ {t..t+dt} ∩ space M) < ∞
      using emeasure-eq-measure ennreal-less-top infinity-ennreal-def by presburger
      ultimately show (∫⁺ x. ennreal (indicat-real {t..t+dt} x * f x) ∂lborel) <
∞ by simp
    qed
    hence ⋀dt. (LBINT s:{t..t+dt}. f s) = integral {t..t+dt} f
      apply (intro set-borel-integral-eq-integral)
      unfolding set-integrable-def
      apply (rule integrableI-nonneg; simp)
      by (rule AE-I2, simp add: assms)
    moreover have ((λdt. (integral {t..t+dt} f) / dt) ⟶ f t) (at-right 0)
    proof −
```

have ((λx. *integral* {*t*..*x*} *f*) *has-real-derivative f t*) (*at t within* {*t*..*u*})
by (*rule integral-has-real-derivative*; *simp add*: *assms*)
moreover have (*at t within* {*t*..*u*}) = (*at* (*t*+*0*) *within* (+)*t* ‘ {*0*..*u*−*t*}) **by** *simp*
ultimately have
((λx. *integral* {*t*..*x*} *f*) ∘ (+)*t has-real-derivative f t*) (*at 0 within* {*0*..*u*−*t*})
by (*metis DERIV-at-within-shift-lemma*)
hence ((λ*dt*. (*integral* {*t*..*t*+*dt*} *f*) / *dt*) ⟶ *f t*) (*at 0 within* {*0*..*u*−*t*})
using *has-field-derivative-iff* **by** *force*
thus *?thesis* **using** *at-within-Icc-at-right assms* **by** (*smt* (*verit*))
qed
ultimately show *?thesis* **by** *simp*
qed
moreover have ⋀*dt*. *dt* > *0* ⟹ *P*(*x in M*. *X x* ∈ {*t* <.. *t*+*dt*}) = (*LBINT s*:{*t*..*t*+*dt*}. *f s*)
proof −
fix *dt* :: *real* **assume** *dt* > *0*
hence [*simp*]: *sym-diff* {*t*<..*t* + *dt*} {*t*..*t* + *dt*} = {*t*} **by** *force*
have *prob* (*X* −‘ {*t*<..*t*+*dt*} ∩ *space M*) = ∫ *s*. *indicator* {*t*<..*t*+*dt*} *s* ∗ *f s* ∂*lborel*
by (*rule distributed-measure*; *simp add*: *assms*)
hence *P*(*x in M*. *X x* ∈ {*t* <.. *t*+*dt*}) = (*LBINT s*:{*t*<..*t*+*dt*}. *f s*)
unfolding *set-lebesgue-integral-def vimage-def Int-def* **by** *simp* (*smt* (*verit*) *Collect-cong*)
moreover have (*LBINT s*:{*t*<..*t*+*dt*}. *f s*) = (*LBINT s*:{*t*..*t*+*dt*}. *f s*)
by (*rule set-integral-null-delta*; *force*)
ultimately show *P*(*x in M*. *X x* ∈ {*t* <.. *t*+*dt*}) = (*LBINT s*:{*t*..*t*+*dt*}. *f s*) **by** *simp*
qed
ultimately have ((λ*dt*. *P*(*x in M*. *t* < *X x* ∧ *X x* ≤ *t* + *dt*) / *dt*) ⟶ *f t*) (*at-right 0*)
by *simp* (*smt* (*verit*) *Lim-cong-within greaterThan-iff*)
hence ((λ*dt*. *P*(*x in M*. *t* < *X x* ∧ *X x* ≤ *t* + *dt* | *X x* > *t*) / *dt*) ⟶
f t / *ccdf* (*distr M borel X*) *t*) (*at-right 0*)
unfolding *cond-prob-def*
apply (*rewrite ccdf-distr-P*[*THEN sym*]; *simp*)
apply (*rewrite mult.commute*, *rewrite divide-divide-eq-left*[*THEN sym*])
by (*rule tendsto-intros*; (*simp add*: *False*)*?*) (*smt* (*verit*) *Collect-cong Lim-cong-within*)
thus *?thesis* **unfolding** *hazard-rate-def* **by** (*intro tendsto-Lim*; *simp*)
qed

end

end
theory *Interest*
imports *Preliminaries*
begin

4 Theory of Interest

locale *interest* =
  fixes i :: *real* — i stands for an interest rate.
  assumes *v-futr-pos*: $1 + i > 0$ — Assume that the future value is positive.
begin

definition *i-nom* :: *nat* ⇒ *real* (‹$i^{-}› [0] 200)
  where $i^{m} ≡ m * ((1+i).^(1/m) - 1)$ — nominal interest rate

definition *i-force* :: *real* (‹$δ› 200)
  where $δ ≡ ln (1+i)$ — force of interest

definition *d-nom* :: *nat* ⇒ *real* (‹$d^{-}› [0] 200)
  where $d^{m} ≡ i^{m} / (1 + i^{m}/m)$ — discount rate

abbreviation *d-nom-yr* :: *real* (‹$d› 200)
  where $d ≡ d^{1}$ — Post-fix *yr* stands for "year".

definition *v-pres* :: *real* (‹$v› 200)
  where $v ≡ 1 / (1+i)$ — present value factor

definition *ann* :: *nat* ⇒ *nat* ⇒ *real* (‹$a^{-}'--› [0,101] 200)
  where $a^{m}-n ≡ ∑k<n*m. v.^((k+1::nat)/m) / m$
  — present value of an immediate annuity

abbreviation *ann-yr* :: *nat* ⇒ *real* (‹$a'--› [101] 200)
  where $a-n ≡ a^{1}-n$

definition *acc* :: *nat* ⇒ *nat* ⇒ *real* (‹$s^{-}'--› [0,101] 200)
  where $s^{m}-n ≡ ∑k<n*m. (1+i).^((k::nat)/m) / m$
  — future value of an immediate annuity
  — The name *acc* stands for "accumulation".

abbreviation *acc-yr* :: *nat* ⇒ *real* (‹$s'--› 200)
  where $s-n ≡ s^{1}-n$

definition *ann-due* :: *nat* ⇒ *nat* ⇒ *real* (‹$a''''^{-}'--› [0,101] 200)
  where $a''^{m}-n ≡ ∑k<n*m. v.^((k::nat)/m) / m$
  — present value of an annuity-due

abbreviation *ann-due-yr* :: *nat* ⇒ *real* (‹$a'''''--› [101] 200)
  where $a''-n ≡ a''^{1}-n$

definition *acc-due* :: *nat* ⇒ *nat* ⇒ *real* (‹$s''''^{-}'--› [0,101] 200)
  where $s''^{m}-n ≡ ∑k<n*m. (1+i).^((k+1::nat)/m) / m$
  — future value of an annuity-due

abbreviation *acc-due-yr* :: *nat* ⇒ *real* (‹$s'''''--› [101] 200)

where $s''\text{-}n \equiv \$s''\hat{}\{1\}\text{-}n$

definition *ann-cont* :: *real* ⇒ *real* (‹$a'''--› [*101*] *200*)
where $a'\text{-}n \equiv \textit{integral}\ \{0..n\}\ (\lambda t{::}\textit{real}.\ \$v.\hat{}t)$
— present value of a continuous annuity

definition *acc-cont* :: *real* ⇒ *real* (‹$s'''--› [*101*] *200*)
where $s'\text{-}n \equiv \textit{integral}\ \{0..n\}\ (\lambda t{::}\textit{real}.\ (1{+}i).\hat{}t)$
— future value of a continuous annuity

definition *perp* :: *nat* ⇒ *real* (‹$a^{-}'-∞› [*0*] *200*)
where $a\hat{}\{m\}\text{-}\infty \equiv 1\ /\ \$i\hat{}\{m\}$
— present value of a perpetual annuity

abbreviation *perp-yr* :: *real* (‹$a'-∞› *200*)
where $a\text{-}\infty \equiv \$a\hat{}\{1\}\text{-}\infty$

definition *perp-due* :: *nat* ⇒ *real* (‹$a''''^{-}'-∞› [*0*] *200*)
where $a''\hat{}\{m\}\text{-}\infty \equiv 1\ /\ \$d\hat{}\{m\}$
— present value of a perpetual annuity-due

abbreviation *perp-due-yr* :: *real* (‹$a'''''-∞› *200*)
where $a''\text{-}\infty \equiv \$a''\hat{}\{1\}\text{-}\infty$

definition *ann-incr* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *real* (‹$'(I^{-}a')^{-}'--› [*0*,*0*,*101*] *200*)
where $(I\hat{}\{l\}a)\hat{}\{m\}\text{-}n \equiv \sum k{<}n{*}m.\ \$v.\hat{}((k{+}1{::}\textit{nat})/m) * \lceil l{*}(k{+}1{::}\textit{nat})/m\rceil\ /\ (l{*}m)$
— present value of an increasing annuity
— This is my original definition.
— Here, l represents the number of increments per unit time.

abbreviation *ann-incr-lvl* :: *nat* ⇒ *nat* ⇒ *real* (‹$'(Ia')^{-}'--› [*0*,*101*] *200*)
where $(Ia)\hat{}\{m\}\text{-}n \equiv \$(I\hat{}\{1\}a)\hat{}\{m\}\text{-}n$
— The post-fix *lvl* stands for "level".

abbreviation *ann-incr-yr* :: *nat* ⇒ *real* (‹$'(Ia')'--› [*101*] *200*)
where $(Ia)\text{-}n \equiv \$(Ia)\hat{}\{1\}\text{-}n$

definition *acc-incr* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *real* (‹$'(I^{-}s')^{-}'--› [*0*,*0*,*101*] *200*)
where $(I\hat{}\{l\}s)\hat{}\{m\}\text{-}n \equiv \sum k{<}n{*}m.\ (1{+}i).\hat{}(n{-}(k{+}1{::}\textit{nat})/m) * \lceil l{*}(k{+}1{::}\textit{nat})/m\rceil\ /\ (l{*}m)$
— future value of an increasing annuity

abbreviation *acc-incr-lvl* :: *nat* ⇒ *nat* ⇒ *real* (‹$'(Is')^{-}'--› [*0*,*101*] *200*)
where $(Is)\hat{}\{m\}\text{-}n \equiv \$(I\hat{}\{1\}s)\hat{}\{m\}\text{-}n$

abbreviation *acc-incr-yr* :: *nat* ⇒ *real* (‹$'(Is')'--› [*101*] *200*)
where $(Is)\text{-}n \equiv \$(Is)\hat{}\{1\}\text{-}n$

definition *ann-due-incr* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *real* (‹$′(*I*^{-}*a*′′′′)^{-}′-› [*0*,*0*,*101*] *200*)
where $(*I*^{*l*}*a*′′)^{*m*}-*n* ≡ ∑ *k*<*n*∗*m*. $*v*.^((*k*::*nat*)/*m*) ∗ ⌈*l*∗(*k*+*1*::*nat*)/*m*⌉ / (*l*∗*m*)

abbreviation *ann-due-incr-lvl* :: *nat* ⇒ *nat* ⇒ *real* (‹$′(*Ia*′′′′)^{-}′-› [*0*,*101*] *200*)
where $(*Ia*′′)^{*m*}-*n* ≡ $(*I*^{*1*}*a*′′)^{*m*}-*n*

abbreviation *ann-due-incr-yr* :: *nat* ⇒ *real* (‹$′(*Ia*′′′′)′-› [*101*] *200*)
where $(*Ia*′′)-*n* ≡ $(*Ia*′′)^{*1*}-*n*

definition *acc-due-incr* :: *nat* ⇒ *nat* ⇒ *nat* ⇒ *real* (‹$′(*I*^{-}*s*′′′′)^{-}′-› [*0*,*0*,*101*] *200*)
where $(*I*^{*l*}*s*′′)^{*m*}-*n* ≡ ∑ *k*<*n*∗*m*. (*1*+*i*).^(*n*−(*k*::*nat*)/*m*) ∗ ⌈*l*∗(*k*+*1*::*nat*)/*m*⌉ / (*l*∗*m*)

abbreviation *acc-due-incr-lvl* :: *nat* ⇒ *nat* ⇒ *real* (‹$′(*Is*′′′′)^{-}′-› [*0*,*101*] *200*)
where $(*Is*′′)^{*m*}-*n* ≡ $(*I*^{*1*}*s*′′)^{*m*}-*n*

abbreviation *acc-due-incr-yr* :: *nat* ⇒ *real* (‹$′(*Is*′′′′)′-› [*101*] *200*)
where $(*Is*′′)-*n* ≡ $(*Is*′′)^{*1*}-*n*

definition *perp-incr* :: *nat* ⇒ *nat* ⇒ *real* (‹$′(*I*^{-}*a*′)^{-}′-∞› [*0*,*0*] *200*)
where $(*I*^{*l*}*a*)^{*m*}-∞ ≡ *lim* (λ*n*. $(*I*^{*l*}*a*)^{*m*}-*n*)

abbreviation *perp-incr-lvl* :: *nat* ⇒ *real* (‹$′(*Ia*′)^{-}′-∞› [*0*] *200*)
where $(*Ia*)^{*m*}-∞ ≡ $(*I*^{*1*}*a*)^{*m*}-∞

abbreviation *perp-incr-yr* :: *real* (‹$′(*Ia*′)′-∞› *200*)
where $(*Ia*)-∞ ≡ $(*Ia*)^{*1*}-∞

definition *perp-due-incr* :: *nat* ⇒ *nat* ⇒ *real* (‹$′(*I*^{-}*a*′′′′)^{-}′-∞› [*0*,*0*] *200*)
where $(*I*^{*l*}*a*′′)^{*m*}-∞ ≡ *lim* (λ*n*. $(*I*^{*l*}*a*′′)^{*m*}-*n*)

abbreviation *perp-due-incr-lvl* :: *nat* ⇒ *real* (‹$′(*Ia*′′′′)^{-}′-∞› [*0*] *200*)
where $(*Ia*′′)^{*m*}-∞ ≡ $(*I*^{*1*}*a*′′)^{*m*}-∞

abbreviation *perp-due-incr-yr* :: *real* (‹$′(*Ia*′′′′)′-∞› *200*)
where $(*Ia*′′)-∞ ≡ $(*Ia*′′)^{*1*}-∞

lemma *v-futr-m-pos*: *1* + $*i*^{*m*}/*m* > *0* **if** *m* ≠ *0* **for** *m*::*nat*
using *v-futr-pos i-nom-def* **by** *force*

lemma *i-nom-1*[*simp*]: $*i*^{*1*} = *i*
using *v-futr-pos i-nom-def* **by** *force*

lemma *i-nom-eff*: (*1* + $*i*^{*m*}/*m*)^*m* = *1* + *i* **if** *m* ≠ *0* **for** *m*::*nat*

unfolding *i-nom-def* **using** *less-imp-neq v-futr-pos that*
apply (*simp*, *subst powr-realpow*[*THEN sym*], *simp*)
by (*subst powr-powr*, *simp*)

lemma *i-nom-i*: *1* + $*i*^{*m*}/*m* = (*1*+*i*).^(*1*/*m*) **if** *m* ≠ *0* **for** *m*::*nat*
unfolding *i-nom-def* **by** (*simp add*: *that*)

lemma *i-nom-0-iff-i-0*: $*i*^{*m*} = *0* ⟷ *i* = *0* **if** *m* ≠ *0* **for** *m*::*nat*
proof
assume $*i*^{*m*} = *0*
hence ⋆: (*1*+*i*).^(*1*/*m*) = (*1*+*i*).^*0*
unfolding *i-nom-def* **using** *v-futr-pos that* **by** *simp*
show *i* = *0*
proof (*rule ccontr*)
assume *i* ≠ *0*
hence *1*/*m* = *0* **using** *powr-inj* ⋆ *v-futr-pos* **by** (*smt* (*verit*))
thus *False* **using** *that* **by** *simp*
qed
next
assume *i* = *0*
thus $*i*^{*m*} = *0*
unfolding *i-nom-def* **by** *simp*
qed

lemma *i-nom-pos-iff-i-pos*: $*i*^{*m*} > *0* ⟷ *i* > *0* **if** *m* ≠ *0* **for** *m*::*nat*
proof
assume $*i*^{*m*} > *0*
hence ⋆: (*1*+*i*).^(*1*/*m*) > *1*.^(*1*/*m*)
unfolding *i-nom-def* **using** *v-futr-pos that* **by** (*simp add*: *zero-less-mult-iff*)
thus *i* > *0*
using *powr-less-cancel2*[*of 1*/*m 1 1*+*i*] *v-futr-pos that* **by** *simp*
next
assume *i* > *0*
hence (*1*+*i*).^(*1*/*m*) > *1*.^(*1*/*m*)
using *powr-less-mono2 v-futr-pos that* **by** *simp*
thus $*i*^{*m*} > *0*
unfolding *i-nom-def* **using** *that* **by** (*simp add*: *zero-less-mult-iff*)
qed

lemma *e-delta*: *exp* ($*δ*) = *1* + *i*
unfolding *i-force-def* **by** (*simp add*: *v-futr-pos*)

lemma *delta-0-iff-i-0*: $*δ* = *0* ⟷ *i* = *0*
proof
assume $*δ* = *0*
thus *i* = *0*
using *e-delta* **by** *auto*
next
assume *i* = *0*

thus $\$\delta = 0$
unfolding *i-force-def* **by** *simp*
qed

lemma *lim-i-nom*: $(\lambda m.\ \$i\hat{}\{m\}) \longrightarrow \$\delta$
proof −
let *?f* = $\lambda h.\ ((1+i).\hat{}h - 1)\ /\ h$
have *D1ipwr*: *DERIV* $(\lambda h.\ (1+i).\hat{}h)\ 0 :> \$\delta$
unfolding *i-force-def*
using *has-real-derivative-powr2*[*OF v-futr-pos*, **where** *x=0*] *v-futr-pos* **by** *simp*
hence *limf*: (*?f* $\longrightarrow \$\delta$) (*at 0*)
unfolding *DERIV-def* **using** *v-futr-pos* **by** *auto*
hence $(\lambda m.\ \$i\hat{}\{Suc\ m\}) \longrightarrow \$\delta$
unfolding *i-nom-def* **using** *tendsto-at-iff-sequentially*[*of ?f* $\$\delta$ *0* $\mathbb{R}$, *THEN iffD1*]
apply *simp*
apply (*drule-tac x=*$\lambda m.\ 1\ /\ Suc\ m$ **in** *spec*, *simp*, *drule mp*)
subgoal using *lim-1-over-n LIMSEQ-Suc* **by** *force*
by (*simp add*: *o-def mult.commute*)
thus *?thesis*
by (*simp add*: *LIMSEQ-imp-Suc*)
qed

lemma *d-nom-0-iff-i-0*: $\$d\hat{}\{m\} = 0 \longleftrightarrow i = 0$ **if** $m \neq 0$ **for** *m*::*nat*
proof −
have $\$d\hat{}\{m\} = 0 \longleftrightarrow \$i\hat{}\{m\} = 0$
unfolding *d-nom-def* **using** *v-futr-m-pos* **by** (*smt* (*verit*) *divide-eq-0-iff of-nat-0*)
thus *?thesis*
using *i-nom-0-iff-i-0 that* **by** *auto*
qed

lemma *d-nom-pos-iff-i-pos*: $\$d\hat{}\{m\} > 0 \longleftrightarrow i > 0$ **if** $m \neq 0$ **for** *m*::*nat*
proof −
have $\$d\hat{}\{m\} > 0 \longleftrightarrow \$i\hat{}\{m\} > 0$
unfolding *d-nom-def* **using** *zero-less-divide-iff i-nom-pos-iff-i-pos v-futr-m-pos that* **by** (*smt* (*verit*))
thus *?thesis*
using *i-nom-pos-iff-i-pos that* **by** *auto*
qed

lemma *d-nom-i-nom*: $1 - \$d\hat{}\{m\}/m = 1\ /\ (1 + \$i\hat{}\{m\}/m)$ **if** $m \neq 0$ **for** *m*::*nat*
proof −
have $1 - \$d\hat{}\{m\}/m = 1 - (\$i\hat{}\{m\}/m)\ /\ (1 + \$i\hat{}\{m\}/m)$
by (*simp add*: *d-nom-def*)
also have $\ldots = 1\ /\ (1 + \$i\hat{}\{m\}/m)$
using *v-futr-m-pos*
by (*smt* (*verit*, *ccfv-SIG*) *add-divide-distrib that div-self*)
finally show *?thesis* **.**
qed

lemma *lim-d-nom*: $(\lambda m.\ \$d\hat{}\{m\}) \longrightarrow \$\delta$
proof −
  have $(\lambda m.\ \$i\hat{}\{m\}/m) \longrightarrow 0$
    using *lim-i-nom tendsto-divide-0 tendsto-of-nat* **by** *blast*
  hence $(\lambda m.\ 1 + \$i\hat{}\{m\}/m) \longrightarrow 1$
    by (*metis add.right-neutral tendsto-add-const-iff*)
  thus *?thesis*
    unfolding *d-nom-def* **using** *lim-i-nom tendsto-divide div-by-1* **by** *fastforce*
qed

lemma *v-pos*: $\$v > 0$
  unfolding *v-pres-def* **using** *v-futr-pos* **by** *auto*

lemma *v-1-iff-i-0*: $\$v = 1 \longleftrightarrow i = 0$
proof
  assume $\$v = 1$
  thus $i = 0$
    unfolding *v-pres-def* **by** *simp*
next
  assume $i = 0$
  thus $\$v = 1$
    unfolding *v-pres-def* **by** *simp*
qed

lemma *v-lt-1-iff-i-pos*: $\$v < 1 \longleftrightarrow i > 0$
proof
  assume $\$v < 1$
  thus $i > 0$
    unfolding *v-pres-def* **by** (*simp add*: *v-futr-pos*)
next
  assume $i > 0$
  thus $\$v < 1$
    unfolding *v-pres-def* **by** (*simp add*: *v-futr-pos*)
qed

lemma *v-i-nom*: $\$v = (1 + \$i\hat{}\{m\}/m).\hat{}-m$ **if** $m \neq 0$ **for** m::*nat*
proof −
  have $\$v = (1 + i).\hat{}-1$
   unfolding *v-pres-def* **using** *v-futr-pos powr-real-def that* **by** (*simp add*: *powr-neg-one*)
  also have $\ldots = ((1 + \$i\hat{}\{m\}/m)\hat{}m).\hat{}-1$
    using *i-nom-eff that* **by** *presburger*
  also have $\ldots = (1 + \$i\hat{}\{m\}/m).\hat{}-m$
    using *i-nom-eff i-nom-i powr-powr that* **by** *fastforce*
  finally show *?thesis* **.**
qed

lemma *i-v*: $1 + i = \$v.\hat{}-1$
  unfolding *v-pres-def powr-real-def* **using** *v-futr-pos powr-neg-one* **by** *simp*

lemma *i-v-powr*: $(1 + i).\hat{}a = \$v.\hat{}-a$ **for** *a*::*real*
by (*subst i-v*, *subst powr-powr*, *simp*)

lemma *v-delta*: *ln* ($v) = − δ
unfolding *i-force-def v-pres-def* **using** *v-futr-pos* **by** (*simp add*: *ln-div*)

lemma *is-derive-vpow*: *DERIV* (λt. $v.\hat{}t) *t* :> − δ ∗ $v.\hat{}t
using *v-delta has-real-derivative-powr2 v-pos* **by** (*metis mult.commute*)

lemma *d-nom-v*: $d^{m} = *m* ∗ (*1* − $v.^(*1*/*m*)) **if** *m* ≠ *0* **for** *m*::*nat*
proof −
have $d^{m} = *m* ∗ (*1* − *1* / (*1* + $i^{m}/*m*))
using *d-nom-i-nom*[*THEN sym*] *that* **by** *force*
also have ... = *m* ∗ (*1* − *1* / (*1* + *i*).^(*1*/*m*))
using *i-nom-i that powr-minus-divide* **by** *simp*
also have ... = *m* ∗ (*1* − $v.^(*1*/*m*))
using *v-pres-def v-futr-pos powr-divide* **by** *simp*
finally show *?thesis* **.**
qed

lemma *d-nom-i-nom-v*: $d^{m} = $i^{m} ∗ $v.^(*1*/*m*) **if** *m* ≠*0* **for** *m*::*nat*
unfolding *d-nom-def v-pres-def* **using** *i-nom-i powr-divide v-futr-pos that* **by** *auto*

lemma *a-calc*: $a^{m}-*n* = (*1* − $v^*n*) / $i^{m} **if** *m* ≠ *0* *i* ≠ *0* **for** *n m* ::*nat*
proof −
have ⋀*l*::*nat*. *l*/*m* = (*1*/*m*) ∗ *l* **by** *simp*
hence ⋆: ⋀*l*::*nat*. $v.^(*l*/*m*) = ($v.^(*1*/*m*))^*l*
using *powr-powr powr-realpow v-pos* **by** (*metis powr-gt-zero*)
hence $a^{m}-*n* = (∑ *k*<*n*∗*m*. ($v.^(*1*/*m*))^(*k*+*1*::*nat*) / *m*)
unfolding *ann-def* **by** *presburger*
also have ... = $v.^(*1*/*m*) ∗ (∑ *k*<*n*∗*m*. ($v.^(*1*/*m*))^*k*) / *m*
by (*simp*, *subst sum-divide-distrib*[*THEN sym*], *subst sum-distrib-left*[*THEN sym*], *simp*)
also have ... = $v.^(*1*/*m*) ∗ ((($v.^(*1*/*m*))^(*n*∗*m*) − *1*) / ($v.^(*1*/*m*) − *1*)) / *m*
apply (*subst geometric-sum*[*of* $v.^(*1*/*m*) *n*∗*m*]; *simp?*)
using *powr-zero-eq-one*[*of* $v] *v-pos v-1-iff-i-0 powr-inj that*
by (*smt* (*verit*, *del-insts*) *divide-eq-0-iff of-nat-eq-0-iff*)
also have ... = (($v.^(*1*/*m*))^(*n*∗*m*) − *1*) / (*m* ∗ ($v.^(*1*/*m*) − *1*) / $v.^(*1*/*m*))
by (*simp add*: *field-simps*)
also have ... = ($v^*n* − *1*) / (*m* ∗ (*1* − *1* / $v.^(*1*/*m*)))
apply (*subst* ⋆[*of n*∗*m*::*nat*, *THEN sym*], *simp only*: *of-nat-simps*)
apply (*subst nonzero-mult-div-cancel-right*[**where** *'a*=*real*, *of m n*], *simp add*: *that*)
apply (*subst powr-realpow*[*OF v-pos*])
apply (*subst times-divide-eq-right*[*of* - - $v.^(*1*/*m*), *THEN sym*])
using *v-pos* **by** (*subst diff-divide-distrib*[*of* - - $v.^(*1*/*m*)], *simp*)

```
  also have ... = (1 - $v^n) / (m * (1 / $v.^(1/m) - 1))
    using minus-divide-divide by (smt (verit) mult-minus-right)
  also have ... = (1 - $v^n) / $i^{m}
    unfolding i-nom-def v-pres-def using v-futr-pos powr-divide by auto
  finally show ?thesis .
qed

lemma a-calc-i-0: $a^{m}-n = n if m ≠ 0 i = 0 for n m :: nat
  unfolding ann-def v-pres-def using that by simp

lemma s-calc-i-0: $s^{m}-n = n if m ≠ 0 i = 0 for n m :: nat
  unfolding acc-def using that by simp

lemma s-a: $s^{m}-n = (1+i)^n * $a^{m}-n if m ≠ 0 for n m :: nat
proof -
  have (1+i)^n * $a^{m}-n = (∑ k<n*m. (1+i)^n * ($v.^((k+1::nat)/m) / m))
    unfolding ann-def using sum-distrib-left by blast
  also have ... = (∑ k<n*m. (1+i).^((n*m - Suc k)/m) / m)
  proof -
   have ⋀k::nat. k < n*m ⟹ (1+i)^n * ($v.^((k+1::nat)/m) / m) = (1+i).^((n*m
- Suc k)/m) / m
     unfolding v-pres-def
     by (simp add: v-futr-pos inverse-powr diff-divide-eq-iff that flip: powr-realpow
powr-add)
    thus ?thesis by (meson lessThan-iff sum.cong)
  qed
  also have ... = (∑ k<n*m. (1+i).^(k/m) / m)
    apply (subst atLeast0LessThan[THEN sym])+
   by (subst sum.atLeastLessThan-rev[THEN sym, of - n*m 0, simplified add-0-right],
simp)
  also have ... = $s^{m}-n
    unfolding acc-def by simp
  finally show ?thesis ..
qed

lemma s-calc: $s^{m}-n = ((1+i)^n - 1) / $i^{m} if m ≠ 0 i ≠ 0 for n m ::
nat
  using that v-futr-pos
  apply (subst s-a, simp, subst a-calc; simp?)
  apply (rule disjI2)
  apply (subst right-diff-distrib, simp)
  apply (rule left-right-inverse-power)
  unfolding v-pres-def by auto

lemma a''-a: $a''^{m}-n = (1+i).^(1/m) * $a^{m}-n if m ≠ 0 for m::nat
  unfolding ann-def ann-due-def
  apply (subst sum-distrib-left, subst times-divide-eq-right, simp)
 by (subst i-v, subst powr-powr, subst powr-add[THEN sym], simp, subst add-divide-distrib,
simp)
```

lemma *a-a''*: $a^{m}-n = $v.^(1/m) * $a''^{m}-n **if** $m \neq 0$ **for** *m*::*nat*
unfolding *ann-def ann-due-def*
apply (*subst sum-distrib-left*, *subst times-divide-eq-right*, *simp*)
by (*subst powr-add*[*THEN sym*], *subst add-divide-distrib*, *simp*)

lemma *a''-calc-i-0*: $a''^{m}-n = n **if** $m \neq 0$ $i = 0$ **for** *n m* :: *nat*
unfolding *ann-due-def v-pres-def* **using** *that* **by** *simp*

lemma *s''-calc-i-0*: $s''^{m}-n = n **if** $m \neq 0$ $i = 0$ **for** *n m* :: *nat*
unfolding *acc-due-def* **using** *that* **by** *simp*

lemma *a''-calc*: $a''^{m}-n = (1 - $v^n) / $d^{m} **if** $m \neq 0$ $i \neq 0$ **for** *n m* :: *nat*
proof −
have $a''^{m}-n = (1+i).^(1/m) * ((1 - $v^n) / $i^{m})
using *a''-a a-calc times-divide-eq-right that* **by** *simp*
also have ... = (1 - $v^n) / ($v.^(1/m) * $i^{m})
by (*subst i-v*, *subst powr-powr*, *simp*, *subst powr-minus-divide*, *simp*)
also have ... = (1 - $v^n) / $d^{m}
using *d-nom-i-nom-v that* **by** *simp*
finally show *?thesis* **.**
qed

lemma *s''-s*: $s''^{m}-n = (1+i).^(1/m) * $s^{m}-n **if** $m \neq 0$ **for** *m*::*nat*
unfolding *acc-def acc-due-def*
apply (*subst sum-distrib-left*, *subst times-divide-eq-right*)
by (*subst powr-add*[*THEN sym*], *simp*, *subst add-divide-distrib*, *simp*)

lemma *s-s''*: $s^{m}-n = $v.^(1/m) * $s''^{m}-n **if** $m \neq 0$ **for** *m*::*nat*
unfolding *acc-def acc-due-def v-pres-def* **using** *v-futr-pos*
apply (*subst sum-distrib-left*, *subst times-divide-eq-right*, *simp*)
by (*subst inverse-powr*, *simp*, *subst powr-add*[*THEN sym*], *subst add-divide-distrib*, *simp*)

lemma *s''-calc*: $s''^{m}-n = ((1+i)^n - 1) / $d^{m} **if** $m \neq 0$ $i \neq 0$ **for** *n m* :: *nat*
proof −
have $s''^{m}-n = (1+i).^(1/m) * ((1+i)^n - 1) / $i^{m}
using *s''-s s-calc times-divide-eq-right that* **by** *simp*
also have ... = ((1+i)^n - 1) / ($v.^(1/m) * $i^{m})
by (*subst i-v*, *subst powr-powr*, *simp*, *subst powr-minus-divide*, *simp*)
also have ... = ((1+i)^n - 1) / $d^{m}
using *d-nom-i-nom-v that* **by** *simp*
finally show *?thesis* **.**
qed

lemma *s''-a''*: $s''^{m}-n = (1+i)^n * $a''^{m}-n **if** $m \neq 0$ **for** *m*::*nat*
using *that s''-s a''-a s-a* **by** *simp*

lemma *a′-calc*: $*a′-n* = (*1* − $*v*.^*n*) / $*δ* **if** *i* ≠ *0 n* ≥ *0* **for** *n*::*real*
unfolding *ann-cont-def*
apply (*rule integral-unique*)
using *has-integral-powr2-from-0*[*OF v-pos - that*(*2*)] *v-delta v-1-iff-i-0 that*
by (*smt* (*verit*) *minus-divide-divide*)

lemma *a′-calc-i-0*: $*a′-n* = *n* **if** *i* = *0 n* ≥ *0* **for** *n*::*real*
unfolding *ann-cont-def*
apply (*subst iffD2*[*OF v-1-iff-i-0*], *simp add*: *that*)
by (*simp add*: *integral-cong that*)

lemma *s′-calc*: $*s′-n* = ((*1*+*i*).^*n* − *1*) / $*δ* **if** *i* ≠ *0 n* ≥ *0* **for** *n*::*real*
unfolding *acc-cont-def*
apply (*rule integral-unique*)
using *has-integral-powr2-from-0*[*OF v-futr-pos - that*(*2*)] *i-force-def that*
by *simp*

lemma *s′-calc-i-0*: $*s′-n* = *n* **if** *i* = *0 n* ≥ *0* **for** *n*::*real*
unfolding *acc-cont-def*
apply (*subst* ‹*i* = *0*›, *simp*)
by (*simp add*: *integral-cong that*)

lemma *s′-a′*: $*s′-n* = (*1*+*i*).^*n* ∗ $*a′-n* **if** *n* ≥ *0* **for** *n*::*real*
proof −
have (*1*+*i*).^*n* ∗ $*a′-n* = *integral* {*0*..*n*} (λ*t*. (*1*+*i*).^(*n*−*t*))
unfolding *ann-cont-def*
using *integrable-on-powr2-from-0-general*[*of* $*v n*] *v-pos v-futr-pos that*
apply (*subst integral-mult*, *simp*)
apply (*rule integral-cong*)
unfolding *v-pres-def* **using** *inverse-powr powr-add*[*THEN sym*] **by** (*smt* (*verit*))
also have ... = $*s′-n*
unfolding *acc-cont-def* **using** *v-futr-pos that*
apply (*subst has-integral-interval-reverse*[*of 0 n*, *simplified*, *THEN integral-unique*];
simp?)
by (*rule continuous-on-powr*; *auto*)
finally show *?thesis* **..**
qed

lemma *lim-m-a*: (λ*m*. $*a*^{*m*}-*n*) ⟶ $*a′-n* **for** *n*::*nat*
proof (*rule LIMSEQ-imp-Suc*)
show (λ*m*. $*a*^{*Suc m*}-*n*) ⟶ $*a′-n*
proof (*cases i* = *0*)
case *True*
show *?thesis*
using *a-calc-i-0 a′-calc-i-0 True* **by** *simp*
next
case *False*
show *?thesis*

using *False v-pos delta-0-iff-i-0*
apply (*subst a-calc*; *simp?*)
apply (*subst a′-calc*; *simp?*)
apply (*subst powr-realpow*, *simp*)
apply (*rule tendsto-divide*; *simp?*)
by (*rule LIMSEQ-Suc*[*OF lim-i-nom*])
qed
qed

lemma *lim-m-a″*: $(\lambda m.\ \$a''\hat{}\{m\}\text{-}n) \longrightarrow \$a'\text{-}n$ **for** *n*::*nat*
proof (*rule LIMSEQ-imp-Suc*)
show $(\lambda m.\ \$a''\hat{}\{Suc\ m\}\text{-}n) \longrightarrow \$a'\text{-}n$
proof (*cases i = 0*)
case *True*
show *?thesis*
using *a″-calc-i-0 a′-calc-i-0 True* **by** *simp*
next
case *False*
show *?thesis*
using *False v-pos delta-0-iff-i-0*
apply (*subst a″-calc*; *simp?*)
apply (*subst a′-calc*; *simp?*)
apply (*subst powr-realpow*, *simp*)
apply (*rule tendsto-divide*; *simp?*)
by (*rule LIMSEQ-Suc*[*OF lim-d-nom*])
qed
qed

lemma *lim-m-s*: $(\lambda m.\ \$s\hat{}\{m\}\text{-}n) \longrightarrow \$s'\text{-}n$ **for** *n*::*nat*
proof (*rule LIMSEQ-imp-Suc*)
show $(\lambda m.\ \$s\hat{}\{Suc\ m\}\text{-}n) \longrightarrow \$s'\text{-}n$
proof (*cases i = 0*)
case *True*
show *?thesis*
using *s-calc-i-0 s′-calc-i-0 True* **by** *simp*
next
case *False*
show *?thesis*
using *False v-futr-pos delta-0-iff-i-0*
apply (*subst s-calc*; *simp?*)
apply (*subst s′-calc*; *simp?*)
apply (*subst powr-realpow*, *simp*)
apply (*rule tendsto-divide*; *simp?*)
by (*rule LIMSEQ-Suc*[*OF lim-i-nom*])
qed
qed

lemma *lim-m-s″*: $(\lambda m.\ \$s''\hat{}\{m\}\text{-}n) \longrightarrow \$s'\text{-}n$ **for** *n*::*nat*
proof (*rule LIMSEQ-imp-Suc*)

```
  show (λm. $s''^{Suc m}-n) ⟶ $s'-n
  proof (cases i = 0)
    case True
    show ?thesis
      using s''-calc-i-0 s'-calc-i-0 True by simp
  next
    case False
    show ?thesis
      using False v-futr-pos delta-0-iff-i-0
      apply (subst s''-calc; simp?)
      apply (subst s'-calc; simp?)
      apply (subst powr-realpow, simp)
      apply (rule tendsto-divide; simp?)
      by (rule LIMSEQ-Suc[OF lim-d-nom])
  qed
qed

lemma lim-n-a: (λn. $a^{m}-n) ⟶ $a^{m}-∞ if m ≠ 0 i > 0 for m::nat
proof −
  have $i^{m} ≠ 0 using i-nom-pos-iff-i-pos that by (smt (verit))
  moreover have (λn. $v^n) ⟶ 0
    using LIMSEQ-realpow-zero[of $v] v-pos v-lt-1-iff-i-pos that by simp
  ultimately show ?thesis
    using that apply (subst a-calc; simp?)
    unfolding perp-def apply (rule tendsto-divide; simp?)
    using tendsto-diff[where a=1 and b=0] by auto
qed

lemma lim-n-a'': (λn. $a''^{m}-n) ⟶ $a''^{m}-∞ if m ≠ 0 i > 0 for m::nat
proof −
  have $d^{m} ≠ 0 using d-nom-pos-iff-i-pos that by (smt (verit))
  moreover have (λn. $v^n) ⟶ 0
    using LIMSEQ-realpow-zero[of $v] v-pos v-lt-1-iff-i-pos that by simp
  ultimately show ?thesis
    using that apply (subst a''-calc; simp?)
    unfolding perp-due-def apply (rule tendsto-divide; simp?)
    using tendsto-diff[where a=1 and b=0] by auto
qed

lemma Ilsm-Ilam: $(I^{l}s)^{m}-n = (1+i)^n * $(I^{l}a)^{m}-n
  if l ≠ 0 m ≠ 0 for l n m :: nat
  unfolding acc-incr-def ann-incr-def v-pres-def using v-futr-pos powr-realpow
  apply (subst inverse-powr, simp)
  apply (subst sum-distrib-left)
 by (subst minus-real-def, subst powr-add, subst times-divide-eq-right, subst mult.assoc,
simp)

lemma Iam-calc: $(Ia)^{m}-n = (∑ j<n. (j+1)/m * (∑ k=j*m..<(j+1)*m. $v.^((k+1)/m)))
  if m ≠ 0 for n m :: nat
```

proof −
let *?I* = {..<*n*}
let *?A* = λ*j*. {*j*∗*m*..<(*j*+*1*)∗*m*}
let *?g* = λ*k*. $*v*.^((*k*+*1*::*nat*)/*m*) ∗ ⌈(*k*+*1*::*nat*)/*m*⌉ / *m*
have $(*Ia*)^{*m*}-*n* = (∑*j*<*n*. ∑ *k*=*j*∗*m*..<(*j*+*1*)∗*m*. $*v*.^((*k*+*1*)/*m*) ∗ ⌈(*k*+*1*)/*m*⌉ / *m*)
unfolding *ann-incr-def* **using** *seq-part-multiple that*
apply (*simp only*: *mult-1*)
by (*subst sum.UNION-disjoint*[*of ?I ?A ?g*, *THEN sym*]; *simp*)
also have ... = (∑*j*<*n*. (*j*+*1*)/*m* ∗ (∑ *k*=*j*∗*m*..<(*j*+*1*)∗*m*. $*v*.^((*k*+*1*)/*m*)))
proof −
{ fix *j k*
assume *j*∗*m* ≤ *k* ∧ *k* < (*j*+*1*)∗*m*
hence *j*∗*m* < *k*+*1* ∧ *k*+*1* ≤ (*j*+*1*)∗*m* **by** *force*
hence *j* < (*k*+*1*)/*m* ∧ (*k*+*1*)/*m* ≤ *j*+*1*
using *pos-less-divide-eq pos-divide-le-eq of-nat-less-iff of-nat-le-iff that*
by (*smt* (*verit*) *of-nat-le-0-iff of-nat-mult*)
hence ⌈(*k*+*1*)/*m*⌉ = *j*+*1*
by (*simp add*: *ceiling-unique*) **}**
hence ⋀*j k*. *j*∗*m* ≤ *k* ∧ *k* < (*j*+*1*)∗*m* ⟹ ⌈(*k*+*1*)/*m*⌉ = *j*+*1*
by (*metis* (*no-types*) *of-nat-1 of-nat-add*)
with *v-pos* **show** *?thesis*
apply −
apply (*rule sum.cong*, *simp*)
apply (*subst sum-distrib-left*, *rule sum.cong*; *simp*)
by (*smt* (*verit*, *ccfv-SIG*) *of-int-1 of-int-diff of-int-of-nat-eq*)
qed
finally show *?thesis* **.**
qed

lemma *Ism-calc*: $(*Is*)^{*m*}-*n* = (∑*j*<*n*. (*j*+*1*)/*m* ∗ (∑ *k*=*j*∗*m*..<(*j*+*1*)∗*m*. (*1*+*i*).^(*n*−(*k*+*1*)/*m*)))
if *m* ≠ *0* **for** *n m* :: *nat*
using *v-pos that*
apply (*subst Ilsm-Ilam*; *simp*)
apply (*subst Iam-calc*[*simplified*]; *simp?*)
apply ((*subst sum-distrib-left*, *rule sum.cong*; *simp*))+
unfolding *v-pres-def* **using** *v-futr-pos*
apply (*subst inverse-powr*; *simp*)
apply (*subst powr-realpow*[*THEN sym*], *simp*)
by (*subst powr-add*[*THEN sym*]; *simp*)

lemma *Imam-calc-aux*: $(*I*^{*m*}*a*)^{*m*}-*n* = (∑*k*<*n*∗*m*. $*v*.^((*k*+*1*)/*m*) ∗ (*k*+*1*) / *m*^*2*)
if *m* ≠ *0* **for** *m*::*nat*
unfolding *ann-incr-def power-def*
apply (*rule sum.cong*, *simp*)
apply (*subst of-nat-mult*)
using *v-pos that*
apply (*subst nonzero-mult-div-cancel-left*, *simp*)

by (*subst ceiling-of-nat*; *simp*)

lemma *Imam-calc*:
$(I^{m}a)^{m}-n = ($v.^(1/m) * (1 - (n*m+1)*$v^n + n*m*$v.^(n+1/m)))
/ (m*(1-$v.^(1/m)))^2
if $i \neq 0$ $m \neq 0$ **for** n m :: *nat*
proof −
have ⋆: $v.^(1/m) > 0$ **using** *v-pos* **by** *force*
hence $(I^{m}a)^{m}-n = (\sum k<n*m.\ (k+1)*($v.^(1/m))^(k+1)) / m^2$
using *that*
apply (*subst Imam-calc-aux*, *simp*)
apply (*subst sum-divide-distrib*[*THEN sym*], *simp*)
apply (*rule sum.cong*; *simp*)
using *powr-realpow*[*THEN sym*] *powr-powr* **by** (*simp add*: *add-divide-distrib powr-add*)
also have ... = $v.^(1/m) * (\sum k<n*m.\ (k+1)*($v.^(1/m))^k) / m^2$
by (*subst sum-distrib-left*, *simp add*: *that*, *rule sum.cong*; *simp*)
also have ... = $v.^(1/m) *
((1 - (n*m+1)*($v.^(1/m))^(n*m) + n*m*($v.^(1/m))^(n*m+1)) / (1 -
$v.^(1/m))^2) / m^2
using *v-pos v-1-iff-i-0 that* **by** (*subst geometric-increasing-sum*; *simp?*)
also have ... = ($v.^(1/m) * (1 - (n*m+1)*$v^n + n*m*$v.^(n+1/m))) /
(m*(1-$v.^(1/m)))^2
using ⋆
apply (*subst powr-realpow*[*of* $v.^(1/m)$, *THEN sym*], *simp*)+
apply (*subst powr-powr*)+
apply (*subst times-divide-eq-right*[*THEN sym*], *subst divide-divide-eq-left*)
apply (*subst power-mult-distrib*)
using *powr-eq-one-iff-gen v-pos v-1-iff-i-0* **apply** (*simp add*: *field-simps*)
by ((*subst powr-realpow*, *simp*)+, *simp*)
finally show *?thesis* **.**
qed

lemma *Imam-calc-i-0*: $(I^{m}a)^{m}-n = (n*m+1)*n / (2*m)$ **if** $i = 0$ $m \neq 0$
for n m :: *nat*
proof −
have $(I^{m}a)^{m}-n = (\sum k<n*m.\ $v.^((k+1)/m) * (k+1) / m^2)$
by (*subst Imam-calc-aux*, *simp-all add*: *that*)
also have ... = $(\sum k<n*m.\ k+1) / m^2$
apply (*subst v-1-iff-i-0*[*THEN iffD2*], *simp-all add*: *that*)
by (*subst sum-divide-distrib*[*THEN sym*], *simp*)
also have ... = $(n*m*(n*m+1)\ div\ 2) / m^2$
apply (*subst Suc-eq-plus1*[*THEN sym*], *subst sum-bounds-lt-plus1*[*of id*, *simplified*])
by (*subst Sum-Icc-nat*, *simp*)
also have ... = $(n*m+1)*n / (2*m)$
apply (*subst real-of-nat-div*, *simp*)
using *that* **by** (*subst power2-eq-square*, *simp add*: *field-simps*)
finally show *?thesis* **.**

qed

lemma *Imsm-calc*:
$(I^{m}s)^{m}-n = ((1+i).^(n+1/m) - (n*m+1)*(1+i).^(1/m) + n*m) / (m*((1+i).^(1/m)-1))^2$
if $i \neq 0$ $m \neq 0$ **for** n m :: *nat*
proof −
have $(I^{m}a)^{m}-n =$
$($v^n * ((1+i).^(n+1/m) - (n*m+1)*(1+i).^(1/m) + n*m)) / (m*((1+i).^(1/m)-1))^2$
proof −
have $(I^{m}a)^{m}-n =$
$($v.^(1/m) * (1 - (n*m+1)*$v^n + n*m*$v.^(n+1/m))) / (m*(1-$v.^(1/m)))^2$
using *that* **by** (*subst Imam-calc*; *simp*)
also have ... $= (1 - (n*m+1)*$v^n + n*m*$v.^(n+1/m)) / ($v.^(1/m)*(m*($v.^(-1/m)-1))^2)$
apply (*subgoal-tac* $v.^(-1/m) = 1 / $v.^(1/m)$, *erule ssubst*)
apply ((*subst power2-eq-square*)+, *simp add*: *field-simps that*)
by (*simp add*: *powr-minus-divide*)
also have ... =
$($v.^(n+1/m) * ($v.^(-n-1/m) - (n*m+1)*$v.^(-1/m) + n*m)) / ($v.^(1/m)*(m*($v.^(-1/m)-1))^2)$
apply (*subgoal-tac* $v.^(-n-1/m) = 1 / $v.^(n+1/m) $v.^(-1/m) = $v^n / $v.^(n+1/m)$)
apply ((*erule ssubst*)+, *simp-all add*: *field-simps*)
using *v-pos*
apply (*simp add*: *powr-diff*[*THEN sym*] *powr-realpow*[*THEN sym*])
by (*smt* (*verit*) *powr-minus-divide*)
also have ... =
$($v^n * ($v.^(-n-1/m) - (n*m+1)*$v.^(-1/m) + n*m)) / ((m*($v.^(-1/m)-1))^2)$
apply (*subst powr-add*[*of* - *n* 1/*m*])
using *v-pos powr-realpow* **by** *simp*
also have ... =
$($v^n * ((1+i).^(n+1/m) - (n*m+1)*(1+i).^(1/m) + n*m)) / ((m*((1+i).^(1/m)-1))^2)$
apply (*subgoal-tac* $-n-1/m = -(n+1/m)$ $-1/m = -(1/m)$, (*erule ssubst*)+)
apply (*subst i-v-powr*[*THEN sym*])+
by *simp-all*
finally show *?thesis* **.**
qed
thus *?thesis*
apply −
using *that v-futr-pos*
apply (*subst Ilsm-Ilam*, *simp*)
apply (*erule ssubst*, *simp*)
apply (*rule disjI2*)
by (*subst power-mult-distrib*[*THEN sym*], *simp add*: *v-pres-def*)
qed

lemma *Imsm-calc-i-0*: $(I^{m}s)^{m}-n = (n*m+1)*n / (2*m)$ **if** $i = 0$ $m \neq 0$
for n m :: *nat*
using *that*

apply (*subst Ilsm-Ilam*, *simp*)
by (*subst Imam-calc-i-0*; *simp*)

lemma *Ila''m-Ilam*: $(I^{l}a'')^{m}-n = (1+i).^(1/m) * $(I^{l}a)^{m}-n
if $l \neq 0$ $m \neq 0$ **for** *l m n* :: *nat*
unfolding *ann-incr-def ann-due-incr-def* **using** *that*
apply (*subst i-v*, *subst powr-powr*, *simp*)
apply (*subst sum-distrib-left*)
apply (*rule sum.cong*; *simp*)
apply (*rule disjI2*)
by (*smt* (*verit*) *add-divide-distrib powr-add*)

lemma *Ia''m-calc*: $(Ia'')^{m}-n = (\sum j<n.\ (j+1)/m * (\sum k=j*m..<(j+1)*m.\ $v.^(k/m)))
if $m \neq 0$ **for** *n m* :: *nat*
using *that*
apply (*subst Ila''m-Ilam*; *simp del*: *One-nat-def*)
apply (*subst Iam-calc*; *simp*)
apply (*subst sum-distrib-left*)
apply (*rule sum.cong*; *simp*)
apply (*subst sum-distrib-left*)+
apply (*rule sum.cong*; *simp*)
apply (*subst i-v-powr*)
using *powr-add*[*of* v, *THEN sym*] **by** (*simp add*: *field-simps*)

lemma *Ima''m-calc-aux*: $(I^{m}a'')^{m}-n = (\sum k<n*m.\ $v.^(k/m) * (k+1) / m^2)
if $m \neq 0$ **for** *m*::*nat*
using *that*
apply (*subst Ila''m-Ilam*, *simp*)
apply (*subst Imam-calc-aux*, *simp*)
apply (*subst sum-distrib-left*)
apply (*rule sum.cong*; *simp*)
using *powr-add*[*of* v, *THEN sym*] *i-v-powr* **by** (*simp add*: *field-simps*)

lemma *Ima''m-calc*: $(I^{m}a'')^{m}-n = (1 - (n*m+1)*$v^n + n*m*$v.^(n+1/m)) / (m*(1-$v.^(1/m)))^2
if $i \neq 0$ $m \neq 0$ **for** *n m* :: *nat*
using *that v-pos*
apply (*subst Ila''m-Ilam*, *simp*)
apply (*subst Imam-calc*; *simp*)
by (*simp add*: *powr-divide v-pres-def*)

lemma *Ils''m-Ilsm*: $(I^{l}s'')^{m}-n = (1+i).^(1/m) * $(I^{l}s)^{m}-n
if $l \neq 0$ $m \neq 0$ **for** *l m n* :: *nat*
unfolding *acc-incr-def acc-due-incr-def* **using** *that*
apply (*subst sum-distrib-left*)
apply (*rule sum.cong*; *simp*)
apply (*rule disjI2*)

by (*subst powr-add*[*THEN sym*], *subst add-divide-distrib*, *simp*)

lemma *Ims''m-calc*:
$(I^{m}s'')^{m}-n =
((1+i).^(1/m) * ((1+i).^(n+1/m) - (n*m+1)*(1+i).^(1/m) + n*m) /
(m*((1+i).^(1/m)-1))^2
if $i \neq 0$ $m \neq 0$ **for** n m :: *nat*
using *that* **by** (*simp add*: *Ils''m-Ilsm Imsm-calc*)

lemma *lim-Imam*: $(\lambda n.\ \$(I^{m}a)^{m}-n) \longrightarrow 1 / (\$i^{m}*\$d^{m})$ **if** $m \neq 0$ $i > 0$ **for** m::*nat*
proof −
have $(\lambda n.\ \$(I^{m}a)^{m}-n) =$
$(\lambda n.\ \$v.^(1/m) * (1 - (n*m+1)*\$v^n + n*m*\$v.^(n+1/m)) / (m*(1-\$v.^(1/m)))^2)$
using *that* **by** (*subst Imam-calc*; *simp*)
moreover have $(\lambda n.\ \$v.^(1/m) * (1 - (n*m+1)*\$v^n + n*m*\$v.^(n+1/m))$
$/ (m*(1-\$v.^(1/m)))^2)$
$\longrightarrow 1 / (\$i^{m}*\$d^{m})$
proof −
have $\star$: $|\$v| < 1$
using *v-lt-1-iff-i-pos v-pos that* **by** *force*
hence $(\lambda n.\ (n*m+1)*\$v^n) \longrightarrow 0$
apply (*subst tendsto-cong*[*of* - $(\lambda n.\ n*m*\$v^n + \$v^n)$])
apply (*rule always-eventually*, *rule allI*)
apply (*simp add*: *distrib-right*)
apply (*subgoal-tac* $0 = 0 + 0$, *erule ssubst*, *intro tendsto-intros*; *simp*)
apply (*subst mult.commute*, *subst mult.assoc*)
apply (*subgoal-tac* $0 =$ *real* $m * 0$, *erule ssubst*, *intro tendsto-intros*; *simp?*)
by (*rule powser-times-n-limit-0*; *simp*)
moreover have $(\lambda n.\ n*m*\$v.^(n+1/m)) \longrightarrow 0$
apply (*subst tendsto-cong*[*of* - $(\lambda n.\ (m*\$v.^(1/m))*(n*\$v^n))$])
apply (*rule always-eventually*, *rule allI*)
apply (*subst powr-add*, *subst powr-realpow*; *simp add*: *v-pos*)
apply (*subgoal-tac* $0 = m*\$v.^(1/m) * 0$, *erule ssubst*, *intro tendsto-intros*;
simp?)
by (*rule powser-times-n-limit-0*, *simp add*: $\star$)
ultimately have $(\lambda n.\ \$v.^(1/m) * (1 - (n*m+1)*\$v^n + n*m*\$v.^(n+1/m))$
$/ (m*(1-\$v.^(1/m)))^2)$
$\longrightarrow \$v.^(1/m) * (1 - 0 + 0)/ (m*(1-\$v.^(1/m)))^2$
using *v-lt-1-iff-i-pos v-pos that* **by** (*intro tendsto-intros*; *simp*)
thus *?thesis*
unfolding *i-nom-def* **using** *v-pos that*
apply (*subst i-v-powr*, *subst powr-minus-divide*, *subst d-nom-v*; *simp*)
by (*subst*(*asm*)(*2*) *power2-eq-square*, *simp add*: *field-simps*)
qed
ultimately show *?thesis* **by** *simp*
qed

lemma *perp-incr-calc*: $\$(I^{m}a)^{m}-\infty = 1 / (\$i^{m}*\$d^{m})$ **if** $m \neq 0$ $i >$

0 **for** *m*::*nat*
unfolding *perp-incr-def* **by** (*rule limI*, *rule lim-Imam*; *simp add*: *that*)

lemma *lim-Ima''m*: (λ*n*. $(*I*^{*m*}*a''*)^{*m*}-*n*) ⟶ *1* / ($*d*^{*m*})^*2* **if** *m* ≠ *0* *i* > *0* **for** *m*::*nat*
unfolding *perp-due-incr-def* **using** *that*
apply (*subst Ila''m-Ilam*, *simp*, *subst mult.commute*, *subst i-v-powr*, *subst powr-minus-divide*)
apply (*subgoal-tac 1*/($*d*^{*m*})^*2* = (*1*/($*i*^{*m*}*$*d*^{*m*}))*(*1*/$*v*.^(*1*/*m*)), *erule ssubst*)
apply (*intro tendsto-intros*, *simp add*: *lim-Imam*)
by (*subst power2-eq-square*, *subst*(*1*) *d-nom-i-nom-v*; *simp add*: *field-simps that*)

lemma *perp-due-incr-calc*: $(*I*^{*m*}*a''*)^{*m*}-∞ = *1* / ($*d*^{*m*})^*2* **if** *m* ≠ *0* *i* > *0* **for** *m*::*nat*
unfolding *perp-due-incr-def* **by** (*rule limI*, *rule lim-Ima''m*; *simp add*: *that*)

end

end
theory *Survival-Model*
imports *HOL−Library.Rewrite HOL−Library.Extended-Nonnegative-Real HOL−Library.Extended-Real HOL−Probability.Probability Preliminaries*
begin

5 Survival Model

The survival model is built on the probability space $\mathfrak{M}$. Additionally, the random variable X : *space* $\mathfrak{M}$ → $\mathbb{R}$ is introduced, which represents the age at death.

locale *prob-space-actuary* = *MM-PS*: *prob-space* $\mathfrak{M}$ **for** $\mathfrak{M}$
— Since the letter M may be used as a commutation function, adopt the letter $\mathfrak{M}$ instead as a notation for the measure space.

locale *survival-model* = *prob-space-actuary* +
fixes *X* :: *'a* ⇒ *real*
assumes *X-RV*[*simp*]: *MM-PS.random-variable* (*borel* :: *real measure*) *X*
and *X-pos-AE*[*simp*]: *AE* ξ *in* $\mathfrak{M}$. *X* ξ > *0*
begin

5.1 General Theory of Survival Model

interpretation *distrX-RD*: *real-distribution distr* $\mathfrak{M}$ *borel X*
using *MM-PS.real-distribution-distr* **by** *simp*

lemma *X-le-event*[*simp*]: {ξ ∈ *space* $\mathfrak{M}$. *X* ξ ≤ *x*} ∈ *MM-PS.events*
by *measurable simp*

lemma *X-gt-event*[*simp*]: {ξ ∈ *space* $\mathfrak{M}$. *X* ξ > *x*} ∈ *MM-PS.events*

by *measurable simp*

lemma *X-compl-le-gt*: *space* $\mathfrak{M}$ − {ξ ∈ *space* $\mathfrak{M}$. *X* ξ ≤ *x*} = {ξ ∈ *space* $\mathfrak{M}$. *X* ξ > *x*} **for** *x*::*real*
proof −
have *space* $\mathfrak{M}$ − {ξ ∈ *space* $\mathfrak{M}$. *X* ξ ≤ *x*} = *space* $\mathfrak{M}$ − (*X* −‘ {..*x*}) **by** *blast*
also have ... = (*X* −‘ {*x*<..}) ∩ *space* $\mathfrak{M}$ **using** *vimage-compl-atMost* **by** *fastforce*
also have ... = {ξ ∈ *space* $\mathfrak{M}$. *X* ξ > *x*} **by** *blast*
finally show *?thesis* **.**
qed

lemma *X-compl-gt-le*: *space* $\mathfrak{M}$ − {ξ ∈ *space* $\mathfrak{M}$. *X* ξ > *x*} = {ξ ∈ *space* $\mathfrak{M}$. *X* ξ ≤ *x*} **for** *x*::*real*
using *X-compl-le-gt* **by** *blast*

5.1.1 Introduction of Survival Function for X

Note that *ccdf* (*distr* $\mathfrak{M}$ *borel X*) is the survival (distributive) function for *X*.

lemma *ccdfX-0-1*: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0* = *1*
apply (*rewrite MM-PS.ccdf-distr-P*, *simp*)
using *X-pos-AE MM-PS.prob-space*
using *MM-PS.prob-Collect-eq-1 X-gt-event* **by** *presburger*

lemma *ccdfX-unborn-1*: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *1* **if** *x* ≤ *0*
proof (*rule antisym*)
show *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* ≤ *1* **using** *MM-PS.ccdf-distr-P* **by** *simp*
show *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* ≥ *1*
proof −
have *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* ≥ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0*
using *finite-borel-measure.ccdf-nonincreasing distrX-RD.finite-borel-measure-M that* **by** *simp*
also have *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0* = *1* **using** *ccdfX-0-1 that* **by** *simp*
finally show *?thesis* **.**
qed
qed

definition *death-pt* :: *ereal* (‹$\$\psi$›)
where $\$\psi$ ≡ *Inf* (*ereal* ‘ {*x* ∈ ℝ. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0*})
— This is my original notation, which is used to develop life insurance mathematics rigorously.

lemma *psi-nonneg*: $\$\psi$ ≥ *0*
unfolding *death-pt-def*
proof (*rule Inf-greatest*)
fix *x*′::*ereal*
assume *x*′ ∈ *ereal* ‘ {*x* ∈ ℝ. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0*}
then obtain *x*::*real* **where** *x*′ = *ereal x* **and** *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0* **by**

blast
hence *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0* > *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* **using** *ccdfX-0-1 X-pos-AE* **by** *simp*
hence $x \geq 0$
using *mono-invE finite-borel-measure.ccdf-nonincreasing distrX-RD.finite-borel-measure-M X-RV*
by (*smt*(*verit*))
thus $x' \geq 0$ **using** ‹$x' = ereal\ x$› **by** *simp*
qed

lemma *ccdfX-beyond-0*: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0* **if** $x > \$\psi$ **for** *x*::*real*
proof −
have *ereal* ‘ {*y* ∈ ℝ. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *y* = *0*} ≠ {} **using** *death-pt-def that* **by** *force*
hence ∃ *y*′∈(*ereal* ‘ {*y* ∈ ℝ. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *y* = *0*}). *y*′ < *ereal x*
using *that* **unfolding** *death-pt-def* **by** (*rule cInf-lessD*)
then obtain *y*′
where *y*′ ∈ (*ereal* ‘ {*y* ∈ ℝ. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *y* = *0*}) **and** *y*′ < *ereal x* **by** *blast*
then obtain *y*::*real*
where *y*′ = *ereal y* **and** *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *y* = *0* **and** *ereal y* < *ereal x* **by** *blast*
hence *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *y* = *0* **and** *y* < *x* **by** *simp-all*
hence *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* ≤ *0*
using *finite-borel-measure.ccdf-nonincreasing distrX-RD.finite-borel-measure-M X-RV*
by (*metis order-less-le*)
thus *?thesis* **using** *finite-borel-measure.ccdf-nonneg distrX-RD.finite-borel-measure-M X-RV* **by** (*smt* (*verit*))
qed

lemma *ccdfX-psi-0*: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) (*real-of-ereal* $\$\psi$) = *0* **if** $\$\psi < \infty$
proof −
have $|\$\psi| \neq \infty$ **using** *that psi-nonneg* **by** *simp*
then obtain *u*::*real* **where** $\$\psi = ereal\ u$ **using** *ereal-real*′ **by** *blast*
hence *real-of-ereal* $\$\psi = u$ **by** *simp*
moreover have *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *u* = *0*
proof −
have ⋀*x*::*real*. *x* ≠ *u* ⟹ *x* ∈ {*u*<..} ⟹ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0*
by (*rule ccdfX-beyond-0*, *simp add*: ‹$\$\psi = ereal\ u$›)
hence (*ccdf* (*distr* $\mathfrak{M}$ *borel X*) ⟶ *0*) (*at-right u*)
apply −
by (*rule iffD2*[*OF Lim-cong-within*[**where** *?g*=(λ*x*.*0*)]], *simp-all*+)
moreover have (*ccdf* (*distr* $\mathfrak{M}$ *borel X*) ⟶ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *u*) (*at-right u*)
using *finite-borel-measure.ccdf-is-right-cont distrX-RD.finite-borel-measure-M continuous-within X-RV* **by** *blast*
ultimately show *?thesis* **using** *tendsto-unique trivial-limit-at-right-real* **by** *blast*

qed
ultimately show *?thesis* **by** *simp*
qed

lemma *ccdfX-0-equiv*: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0* $\longleftrightarrow$ *x* $\geq$ $\$\psi$ **for** *x*::*real*
proof
assume *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0*
thus *ereal x* $\geq$ $\$\psi$ **unfolding** *death-pt-def* **by** (*simp add*: *INF-lower*)
next
assume $\$\psi \leq$ *ereal x*
hence $\$\psi$ = *ereal x* $\vee$ $\$\psi <$ *ereal x* **unfolding** *less-eq-ereal-def* **by** *auto*
thus *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0*
proof
assume $\star$: $\$\psi$ = *ereal x*
hence $\$\psi < \infty$ **by** *simp*
moreover have *real-of-ereal* $\$\psi$ = *x* **using** $\star$ **by** *simp*
ultimately show *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0* **using** *ccdfX-psi-0* **by** *simp*
next
assume $\$\psi <$ *ereal x*
thus *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x* = *0* **by** (*rule ccdfX-beyond-0*)
qed
qed

lemma *psi-pos*[*simp*]: $\$\psi >$ *0*
proof (*rule not-le-imp-less*, *rule notI*)
show $\$\psi \leq$ (*0*::*ereal*) $\Longrightarrow$ *False*
proof −
assume $\$\psi \leq$ (*0*::*ereal*)
hence *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0* = *0* **using** *ccdfX-0-equiv* **by** (*simp add*: *zero-ereal-def*)
moreover have *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *0* = *1* **using** *ccdfX-0-1* **by** *simp*
ultimately show *False* **by** *simp*
qed
qed

corollary *psi-pos'*[*simp*]: $\$\psi >$ *ereal 0*
using *psi-pos zero-ereal-def* **by** *presburger*

5.1.2 Introdution of Future-Lifetime Random Variable $T(x)$

definition *alive* :: *real* ⇒ *'a set*
where *alive x* ≡ {$\xi \in$ *space* $\mathfrak{M}$. *X* $\xi >$ *x*}

lemma *alive-event*[*simp*]: *alive x* ∈ *MM-PS.events* **for** *x*::*real*
unfolding *alive-def* **by** *simp*

lemma *X-alivex-measurable*[*measurable*, *simp*]: *X* ∈ *borel-measurable* ($\mathfrak{M}$ ⌊ *alive x*) **for** *x*::*real*
unfolding *cond-prob-space-def* **by** (*measurable*; *simp add*: *measurable-restrict-space1*)

definition *futr-life* :: *real* ⇒ (*'a* ⇒ *real*) (‹*T*›)
where *T x* ≡ (λξ. *X* ξ − *x*)
— Note that $T(x)$: *space* $\mathfrak{M}$ → $\mathbb{R}$ represents the time until death of a person aged x.

lemma *T0-eq-X*[*simp*]: *T 0* = *X*
unfolding *futr-life-def* **by** *simp*

lemma *Tx-measurable*[*measurable*, *simp*]: *T x* ∈ *borel-measurable* $\mathfrak{M}$ **for** *x*::*real*
unfolding *futr-life-def* **by** (*simp add*: *borel-measurable-diff*)

lemma *Tx-alivex-measurable*[*measurable*, *simp*]: *T x* ∈ *borel-measurable* ($\mathfrak{M}$ ⌊ *alive x*) **for** *x*::*real*
unfolding *futr-life-def* **by** (*simp add*: *borel-measurable-diff*)

lemma *alive-T*: *alive x* = {ξ ∈ *space* $\mathfrak{M}$. *T x* ξ > *0*} **for** *x*::*real*
unfolding *alive-def futr-life-def* **by** *force*

lemma *alivex-Tx-pos*[*simp*]: *0* < *T x* ξ **if** ξ ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*) **for** *x*::*real*
using *MM-PS.space-cond-prob-space alive-T that* **by** *simp*

lemma *PT0-eq-PX-lborel*: $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *T 0* ξ ∈ *A* | *T 0* ξ > *0*) = $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X* ξ ∈ *A*)
if *A* ∈ *sets lborel* **for** *A* :: *real set*
apply (*rewrite MM-PS.cond-prob-AE-prob*, *simp-all*)
using *that X-RV measurable-lborel1 predE pred-sets2* **by** *blast*

5.1.3 Actuarial Notations on the Survival Model

definition *survive* :: *real* ⇒ *real* ⇒ *real* (‹$*p*'-{-&-}› [*0*,*0*] *200*)
where $*p*-{*t*&*x*} ≡ *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *t*
— the probability that a person aged x will survive for t years
— Note that the function $*p*-{·&*x*} is the survival function on ($\mathfrak{M}$ ⌊ *alive x*) for the random variable $T(x)$.
— The parameter t is usually nonnegative, but theoretically it can be negative.

abbreviation *survive-1* :: *real* ⇒ *real* (‹$*p*'--› [*101*] *200*)
where $*p*-*x* ≡ $*p*-{*1*&*x*}

definition *die* :: *real* ⇒ *real* ⇒ *real* (‹$*q*'-{-&-}› [*0*,*0*] *200*)
where $*q*-{*t*&*x*} ≡ *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *t*
— the probability that a person aged x will die within t years
— Note that the function $*q*-{·&*x*} is the cumulative distributive function on ($\mathfrak{M}$ ⌊ *alive x*) for the random variable $T(x)$.
— The parameter t is usually nonnegative, but theoretically it can be negative.

abbreviation *die-1* :: *real* ⇒ *real* (‹$*q*'--› [*101*] *200*)
where $*q*-*x* ≡ $*q*-{*1*&*x*}

definition *die-defer* :: *real* ⇒ *real* ⇒ *real* ⇒ *real* (‹$*q*′-{-|-&-}› [*0*,*0*,*0*] *200*)
where $*q*-{*f*|*t*&*x*} = |$*q*-{*f*+*t*&*x*} − $*q*-{*f*&*x*}|
— the probability that a person aged x will die within t years, deferred f years
— The parameters f and t are usually nonnegative, but theoretically they can be negative.

abbreviation *die-defer-1* :: *real* ⇒ *real* ⇒ *real* (‹$*q*′-{-|&-}› [*0*,*0*] *200*)
where $*q*-{*f*|&*x*} ≡ $*q*-{*f*|*1*&*x*}

definition *life-expect* :: *real* ⇒ *real* (‹$*e*‘∘′--› [*101*] *200*)
where $*e*‘∘-*x* ≡ *integral*^{L} (𝔐 ⌊ *alive x*) (*T x*)
— complete life expectation
— Note that $*e*‘∘-*x* is calculated as *0* when *nn-integral* (𝔐 ⌊ *alive x*) (*T x*) = ∞.

definition *temp-life-expect* :: *real* ⇒ *real* ⇒*real* (‹$*e*‘∘′-{-:-}› [*0*,*0*] *200*)
where $*e*‘∘-{*x*:*n*} ≡ *integral*^{L} (𝔐 ⌊ *alive x*) (λξ. *min* (*T x* ξ) *n*)
— temporary complete life expectation

definition *curt-life-expect* :: *real* ⇒ *real* (‹$*e*′--› [*101*] *200*)
where $*e*-*x* ≡ *integral*^{L} (𝔐 ⌊ *alive x*) (λξ. ⌊*T x* ξ⌋)
— curtate life expectation
— Note that $*e*-*x* is calculated as *0* when *nn-integral* (𝔐 ⌊ *alive x*) ⌊*T x*⌋ = ∞.

definition *temp-curt-life-expect* :: *real* ⇒ *real* ⇒ *real* (‹$*e*′-{-:-}› [*0*,*0*] *200*)
where $*e*-{*x*:*n*} ≡ *integral*^{L} (𝔐 ⌊ *alive x*) (λξ. ⌊*min* (*T x* ξ) *n*⌋)
— temporary curtate life expectation
— In the definition n can be a real number, but in practice n is usually a natural number.

5.1.4 Properties of Survival Function for $T(x)$

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: *x* < $ψ
begin

lemma *PXx-pos*[*simp*]: *P*(ξ *in* 𝔐. *X* ξ > *x*) > *0*
proof −
have *P*(ξ *in* 𝔐. *X* ξ > *x*) = *ccdf* (*distr* 𝔐 *borel X*) *x*
unfolding *alive-def* **using** *MM-PS.ccdf-distr-P* **by** *simp*
also have ... > *0*
using *ccdfX-0-equiv distrX-RD.ccdf-nonneg x-lt-psi* **by** (*smt* (*verit*) *linorder-not-le*)
finally show *?thesis* **.**
qed

lemma *PTx-pos*[*simp*]: *P*(ξ *in* 𝔐. *T x* ξ > *0*) > *0*
apply (*rewrite alive-T*[*THEN sym*])

unfolding *alive-def* **by** *simp*

interpretation *alivex-PS*: *prob-space* $\mathfrak{M} \restriction$ *alive x*
by (*rule MM-PS.cond-prob-space-correct*, *simp-all add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*) **by** *simp*

lemma *ccdfTx-cond-prob*:
ccdf (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* = $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *T x* ξ > *t* | *T x* ξ > *0*) **for** *t*::*real*
apply (*rewrite alivex-PS.ccdf-distr-P*, *simp*)
unfolding *alive-def*
apply (*rewrite MM-PS.cond-prob-space-cond-prob*[*THEN sym*], *simp-all add*: *pred-def*)
unfolding *futr-life-def* **by** *simp*

lemma *ccdfTx-0-1*: *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *0* = *1*
apply (*rewrite ccdfTx-cond-prob*)
unfolding *futr-life-def cond-prob-def*
by (*smt* (*verit*, *best*) *Collect-cong PXx-pos divide-eq-1-iff*)

lemma *ccdfTx-nonpos-1*: *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* = *1* **if** *t* ≤ *0* **for** *t* :: *real*
using *antisym ccdfTx-0-1 that*
by (*metis distrTx-RD.ccdf-bounded-prob distrTx-RD.ccdf-nonincreasing*)

lemma *ccdfTx-0-equiv*: *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* = *0* ⟷ *x*+*t* ≥ $\$\psi$ **for** *t*::*real*
proof −
have *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* =
$\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* ξ > *x*+*t* ∧ *X* ξ > *x*) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* ξ > *x*)
apply (*rewrite ccdfTx-cond-prob*)
unfolding *cond-prob-def futr-life-def* **by** (*smt* (*verit*) *Collect-cong*)
hence *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* = *0* ⟷
$\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* ξ > *x*+*t* ∧ *X* ξ > *x*) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* ξ > *x*) = *0*
by *simp*
also have ... ⟷ $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* ξ > *x*+*t* ∧ *X* ξ > *x*) = *0*
using *x-lt-psi PXx-pos* **by** (*smt* (*verit*) *divide-eq-0-iff*)
also have ... ⟷ *x*+*t* ≥ $\$\psi$
using *ccdfX-0-equiv MM-PS.ccdf-distr-P*
by (*smt* (*verit*) *Collect-cong X-RV le-ereal-le linorder-not-le x-lt-psi*)
finally show *?thesis* **.**
qed

lemma *ccdfTx-continuous-on-nonpos*[*simp*]:
continuous-on {..*0*} (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)))
by (*metis atMost-iff ccdfTx-nonpos-1 continuous-on-cong continuous-on-const*)

lemma *ccdfTx-differentiable-on-nonpos*[*simp*]:
(*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) *differentiable-on* {*..0*}
by (*rewrite differentiable-on-cong*[**where** *f*=λ-. *1*]; *simp add*: *ccdfTx-nonpos-1*)

lemma *ccdfTx-has-real-derivative-0-at-neg*:
(*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative 0*) (*at t*) **if** *t* < *0* **for** *t*::*real*
apply (*rewrite has-real-derivative-iff-has-vector-derivative*)
apply (*rule has-vector-derivative-transform-within-open*[*of* λ-. *1* - - {*..<0*}])
using *ccdfTx-nonpos-1 that* **by** *simp-all*

lemma *ccdfTx-integrable-Icc*:
set-integrable lborel {*a..b*} (*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) **for** *a b* :: *real*
proof −
have (∫ ⁺*t*. *ennreal* (*indicat-real* {*a..b*} *t* ∗ *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t*) ∂*lborel*)
< ⊤
proof −
have (∫ ⁺*t*. *ennreal* (*indicat-real* {*a..b*} *t* ∗ *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t*) ∂*lborel*)
≤ (∫ ⁺*t*. *ennreal* (*indicat-real* {*a..b*} *t*) ∂*lborel*)
apply (*rule nn-integral-mono*)
using *distrTx-RD.ccdf-bounded*
by (*simp add*: *distrTx-RD.ccdf-bounded-prob indicator-times-eq-if*(*1*))
also have ... = *nn-integral lborel* (*indicator* {*a..b*}) **by** (*meson ennreal-indicator*)
also have ... = *emeasure lborel* {*a..b*} **by** (*rewrite nn-integral-indicator*; *simp*)
also have ... < ⊤
using *emeasure-lborel-Icc-eq ennreal-less-top infinity-ennreal-def* **by** *presburger*
finally show *?thesis* **.**
qed
thus *?thesis*
unfolding *set-integrable-def*
apply (*intro integrableI-nonneg*, *simp-all*)
using *distrTx-RD.ccdf-nonneg* **by** (*intro always-eventually*) *auto*
qed

corollary *ccdfTx-integrable-on-Icc*:
ccdf (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *integrable-on* {*a..b*} **for** *a b* :: *real*
using *set-borel-integral-eq-integral ccdfTx-integrable-Icc* **by** *force*

lemma *ccdfTx-PX*:
ccdf (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t* = 𝒫(ξ *in* 𝔐. *X* ξ > *x*+*t*) / 𝒫(ξ *in* 𝔐. *X* ξ > *x*)
if *t* ≥ *0* **for** *t*::*real*
apply (*rewrite ccdfTx-cond-prob*)
unfolding *cond-prob-def futr-life-def PXx-pos* **by** (*smt* (*verit*) *Collect-cong that*)

lemma *ccdfTx-ccdfX*: *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t* =
ccdf (*distr* 𝔐 *borel X*) (*x* + *t*) / *ccdf* (*distr* 𝔐 *borel X*) *x*

if $t \geq 0$ **for** t::*real*
using *ccdfTx-PX that MM-PS.ccdf-distr-P X-RV* **by** *presburger*

lemma *ccdfT0-eq-ccdfX*: *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive 0*) *borel* (*T 0*)) = *ccdf* (*distr* $\mathfrak{M}$ *borel X*)
proof
fix *x*
show *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive 0*) *borel* (*T 0*)) *x* = *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *x*
proof (*cases* ‹$x \geq 0$›)
case *True*
thus *?thesis*
using *survival-model.ccdfTx-ccdfX*[**where** *x=0*] *ccdfX-0-1 survival-model-axioms*
by *fastforce*
next
case *False*
hence $x \leq 0$ **by** *simp*
thus *?thesis*
apply (*rewrite ccdfX-unborn-1*, *simp*)
by (*rewrite survival-model.ccdfTx-nonpos-1*; *simp add*: *survival-model-axioms*)
qed
qed

lemma *continuous-ccdfX-ccdfTx*:
continuous (*at* (*x*+*t*) *within* {*x*..}) (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) ⟷
continuous (*at t within* {*0*..}) (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
if $t \geq 0$ **for** *t*::*real*
proof −
let *?srvl* = *ccdf* (*distr* $\mathfrak{M}$ *borel X*)
have [*simp*]: $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* $\xi > x) \neq 0$ **using** *PXx-pos* **by** *force*
have ⋆: $\bigwedge u$. $u \geq 0 \Longrightarrow$ *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *u* =
?srvl (*x* + *u*) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* $\xi > x)$
using *survive-def MM-PS.ccdf-distr-P ccdfTx-PX that* **by** *simp*
have *continuous* (*at t within* {*0*..}) (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) ⟷
continuous (*at t within* {*0*..}) (λ*u*. *?srvl* (*x*+*u*) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *x* < *X* $\xi)$)
proof −
have $\forall_F$ *u in at t within* {*0*..}. *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *u* =
?srvl (*x*+*u*) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* $\xi > x)$
using ⋆ **by** (*rewrite eventually-at-topological*, *simp-all*) *blast*
thus *?thesis*
by (*intro continuous-at-within-cong*, *simp-all add*: ⋆ *that*)
qed
also have ... ⟷ *continuous* (*at t within* {*0*..}) (λ*u*. *?srvl* (*x*+*u*))
by (*rewrite at* - = ¤ *continuous-cdivide-iff*[*of* $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *X* $\xi > x)$], *simp-all*)
also have ... ⟷ *continuous* (*at* (*x*+*t*) *within* {*x*..}) *?srvl*
proof
let *?subx* = λ*v*. *v*−*x*
assume *LHS*: *continuous* (*at t within* {*0*..}) (λ*u*. *?srvl* (*x*+*u*))
hence *continuous* (*at* (*?subx* (*x*+*t*)) *within ?subx* ‘ {*x*..}) (λ*u*. *?srvl* (*x*+*u*))
proof −

have *?subx* ‘ $\{x..\} = \{0..\}$
by (*metis* (*no-types*, *lifting*) *add.commute add-uminus-conv-diff diff-self image-add-atLeast image-cong*)
thus *?thesis* **using** *LHS* **by** *simp*
qed
moreover have *continuous* (*at* $(x+t)$ *within* $\{x..\}$) *?subx* **by** (*simp add*: *continuous-diff*)
ultimately have *continuous* (*at* $(x+t)$ *within* $\{x..\}$) $(\lambda u.$ *?srvl* $(x + ($*?subx* $u)))$
using *continuous-within-compose2* **by** *force*
thus *continuous* (*at* $(x+t)$ *within* $\{x..\}$) *?srvl* **by** *simp*
next
assume *RHS*: *continuous* (*at* $(x+t)$ *within* $\{x..\}$) *?srvl*
hence *continuous* (*at* ((*plus* x) t) *within* (*plus* x) ‘ $\{0..\}$) *?srvl* **by** *simp*
moreover have *continuous* (*at* t *within* $\{0..\}$) (*plus* x) **by** (*simp add*: *continuous-add*)
ultimately show *continuous* (*at* t *within* $\{0..\}$) $(\lambda u.$ *?srvl* $(x+u))$
using *continuous-within-compose2* **by** *force*
qed
finally show *?thesis* **by** *simp*
qed

lemma *isCont-ccdfX-ccdfTx*:
isCont (*ccdf* (*distr* $\mathfrak{M}$ *borel* X)) $(x+t) \longleftrightarrow$
isCont (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive* x) *borel* (T x))) t
if $t > 0$ **for** t::*real*
proof −
have *isCont* (*ccdf* (*distr* $\mathfrak{M}$ *borel* X)) $(x+t) \longleftrightarrow$
continuous (*at* $(x+t)$ *within* $\{x<..\}$) (*ccdf* (*distr* $\mathfrak{M}$ *borel* X))
by (*smt* (*verit*) *at-within-open greaterThan-iff open-greaterThan that*)
also have $\ldots \longleftrightarrow$ *continuous* (*at* $(x+t)$ *within* $\{x..\}$) (*ccdf* (*distr* $\mathfrak{M}$ *borel* X))
by (*meson Ioi-le-Ico calculation continuous-within-subset top-greatest*)
also have $\ldots \longleftrightarrow$ *continuous* (*at* t *within* $\{0..\}$) (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive* x) *borel* (T x)))
using *that continuous-ccdfX-ccdfTx* **by** *force*
also have $\ldots \longleftrightarrow$ *continuous* (*at* t *within* $\{0<..\}$) (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive* x) *borel* (T x)))
by (*metis Ioi-le-Ico at-within-open continuous-at-imp-continuous-at-within continuous-within-subset greaterThan-iff open-greaterThan that*)
also have $\ldots \longleftrightarrow$ *isCont* (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive* x) *borel* (T x))) t
by (*metis at-within-open greaterThan-iff open-greaterThan that*)
finally show *?thesis* **.**
qed

lemma *has-real-derivative-ccdfX-ccdfTx*:
((*ccdf* (*distr* $\mathfrak{M}$ *borel* X)) *has-real-derivative* D) (*at* $(x+t)$) $\longleftrightarrow$
((*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive* x) *borel* (T x))) *has-real-derivative* (D / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. X ξ $> x)))$ (*at* t)
if $t > 0$ **for** t D :: *real*

```
proof −
  have ((ccdf (distr (𝔐 ⌊ alive x) borel (T x))) has-real-derivative
    (D / 𝒫(ξ in 𝔐. X ξ > x))) (at t) ⟷
    ((λt. (ccdf (distr 𝔐 borel X)) (x+t) / 𝒫(ξ in 𝔐. X ξ > x)) has-real-derivative
    (D / 𝒫(ξ in 𝔐. X ξ > x))) (at t)
  proof −
    let ?d = t/2
    { fix u::real assume dist u t < ?d
      hence u > 0 by (smt (verit) dist-real-def dist-triangle-half-r)
      hence ccdf (distr (𝔐 ⌊ alive x) borel (T x)) u =
        ccdf (distr 𝔐 borel X) (x + u) / MM-PS.prob {ξ::'a ∈ space 𝔐. x < X ξ}
        using survive-def MM-PS.ccdf-distr-P ccdfTx-PX that by simp }
    moreover have ?d > 0 using that by simp
    ultimately show ?thesis
      apply −
      apply (rule DERIV-cong-ev, simp)
       apply (rewrite eventually-nhds-metric, blast)
      by simp
  qed
  also have ... ⟷ ((λt. (ccdf (distr 𝔐 borel X)) (x+t)) has-real-derivative D)
(at t)
    using PXx-pos by (rewrite DERIV-cdivide-iff[of 𝒫(ξ in 𝔐. X ξ > x), THEN
sym]; force)
  also have ... ⟷ (ccdf (distr 𝔐 borel X) has-real-derivative D) (at (x+t))
    by (simp add: DERIV-shift add.commute)
  finally show ?thesis by simp
qed

lemma differentiable-ccdfX-ccdfTx:
  (ccdf (distr 𝔐 borel X)) differentiable at (x+t) ⟷
  (ccdf (distr (𝔐 ⌊ alive x) borel (T x))) differentiable at t
  if t > 0 for t::real
  apply (rewrite differentiable-eq-field-differentiable-real)+
  unfolding field-differentiable-def using has-real-derivative-ccdfX-ccdfTx that
  by (smt (verit, del-insts) PXx-pos nonzero-mult-div-cancel-left)
```

5.1.5 Properties of $\$p\text{-}\{t\&x\}$

```
lemma p-0-1: $p-{0&x} = 1
  unfolding survive-def using ccdfTx-0-1 by simp

lemma p-nonneg[simp]: $p-{t&x} ≥ 0 for t::real
  unfolding survive-def using distrTx-RD.ccdf-nonneg by simp

lemma p-le-1[simp]: $p-{t&x} ≤ 1 for t::real
  unfolding survive-def using distrTx-RD.ccdf-bounded-prob by auto

lemma p-0-equiv: $p-{t&x} = 0 ⟷ x+t ≥ $ψ for t::real
  unfolding survive-def by (rule ccdfTx-0-equiv)
```

lemma *p-PTx*: $\$p\text{-}\{t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi > t \mid T\ x\ \xi > 0)$ **for** *t*::*real*
unfolding *survive-def* **using** *ccdfTx-cond-prob* **by** *simp*

lemma *p-PX*: $\$p\text{-}\{t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x + t) \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x)$ **if** $t \geq 0$ **for** *t*::*real*
unfolding *survive-def* **using** *ccdfTx-PX that* **by** *simp*

lemma *p-mult*: $\$p\text{-}\{t{+}t' \ \&\ x\} = \$p\text{-}\{t\&x\} * \$p\text{-}\{t' \ \&\ x{+}t\}$
if $t \geq 0\ t' \geq 0\ x{+}t < \$\psi$ **for** *t t'* :: *real*
proof −
have $\$p\text{-}\{t{+}t' \ \&\ x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x{+}t{+}t') \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x)$
apply (*rewrite p-PX*; (*simp add*: *that*)*?*)
by (*rule disjI2*, *smt* (*verit*, *best*) *Collect-cong*)
also have ... $= (\mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x{+}t{+}t') \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x{+}t)) *$
$(\mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x{+}t) \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x))$
using *that survival-model.PXx-pos survival-model-axioms* **by** *fastforce*
also have ... $= \$p\text{-}\{t\&x\} * \$p\text{-}\{t' \ \&\ x{+}t\}$
apply (*rewrite p-PX*, *simp add*: *that*)
by (*rewrite survival-model.p-PX*, *simp-all add*: *that survival-model-axioms*)
finally show *?thesis* **.**
qed

lemma *p-PTx-ge-ccdf-isCont*: $\$p\text{-}\{t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi \geq t \mid T\ x\ \xi > 0)$
if *isCont* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) (*x*+*t*) *t* > *0* **for** *t*::*real*
unfolding *survive-def* **using** *that isCont-ccdfX-ccdfTx*
apply (*rewrite alivex-PS.ccdf-continuous-distr-P-ge*, *simp-all*)
by (*rewrite MM-PS.cond-prob-space-cond-prob*, *simp-all add*: *alive-T*)

end

5.1.6 Properties of Survival Function for X

lemma *ccdfX-continuous-unborn*[*simp*]: *continuous-on* {..*0*} (*ccdf* (*distr* $\mathfrak{M}$ *borel X*))
using *ccdfTx-continuous-on-nonpos* **by** (*metis ccdfT0-eq-ccdfX psi-pos'*)

lemma *ccdfX-differentiable-unborn*[*simp*]: (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable-on* {..*0*}
using *ccdfTx-differentiable-on-nonpos* **by** (*metis ccdfT0-eq-ccdfX psi-pos'*)

lemma *ccdfX-has-real-derivative-0-unborn*:
(*ccdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative 0*) (*at x*) **if** *x* < *0* **for** *x*::*real*
using *ccdfTx-has-real-derivative-0-at-neg* **by** (*metis ccdfT0-eq-ccdfX psi-pos' that*)

lemma *ccdfX-integrable-Icc*:
set-integrable lborel {*a*..*b*} (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) **for** *a b* :: *real*
using *ccdfTx-integrable-Icc* **by** (*metis ccdfT0-eq-ccdfX psi-pos'*)

corollary *ccdfX-integrable-on-Icc*:
ccdf (*distr* $\mathfrak{M}$ *borel* X) *integrable-on* $\{a..b\}$ **for** a b :: *real*
using *set-borel-integral-eq-integral ccdfX-integrable-Icc* **by** *force*

lemma *ccdfX-p*: *ccdf* (*distr* $\mathfrak{M}$ *borel* X) x = \$*p*-$\{x\&0\}$ **for** x::*real*
by (*metis ccdfT0-eq-ccdfX survive-def psi-pos'*)

5.1.7 Introduction of Cumulative Distributive Function for X

lemma *cdfX-0-0*: *cdf* (*distr* $\mathfrak{M}$ *borel* X) *0* = *0*
using *ccdfX-0-1 distrX-RD.ccdf-cdf distrX-RD.prob-space* **by** *fastforce*

lemma *cdfX-unborn-0*: *cdf* (*distr* $\mathfrak{M}$ *borel* X) x = *0* **if** $x \leq 0$
using *ccdfX-unborn-1 cdfX-0-0 distrX-RD.cdf-ccdf that* **by** *fastforce*

lemma *cdfX-beyond-1*: *cdf* (*distr* $\mathfrak{M}$ *borel* X) x = *1* **if** $x >$ \$$\psi$ **for** x::*real*
using *ccdfX-beyond-0 distrX-RD.cdf-ccdf that distrX-RD.prob-space* **by** *force*

lemma *cdfX-psi-1*: *cdf* (*distr* $\mathfrak{M}$ *borel* X) (*real-of-ereal* \$$\psi$) = *1* **if** \$$\psi < \infty$
using *ccdfX-psi-0 distrX-RD.cdf-ccdf distrX-RD.prob-space that* **by** *fastforce*

lemma *cdfX-1-equiv*: *cdf* (*distr* $\mathfrak{M}$ *borel* X) x = *1* $\longleftrightarrow$ $x \geq$ \$$\psi$ **for** x::*real*
using *ccdfX-0-equiv distrX-RD.cdf-ccdf distrX-RD.prob-space* **by** *force*

5.1.8 Properties of Cumulative Distributive Function for $T(x)$

context
fixes x::*real*
assumes *x-lt-psi*[*simp*]: $x <$ \$$\psi$
begin

interpretation *alivex-PS*: *prob-space* $\mathfrak{M} \downharpoonright$ *alive x*
by (*rule MM-PS.cond-prob-space-correct*, *simp-all add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M} \downharpoonright$ *alive x*) *borel* (*T x*) **by** *simp*

lemma *cdfTx-cond-prob*:
cdf (*distr* ($\mathfrak{M} \downharpoonright$ *alive x*) *borel* (*T x*)) *t* = $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. *T x* $\xi \leq t \mid$ *T x* $\xi > 0)$ **for** *t*::*real*
apply (*rewrite distrTx-RD.cdf-ccdf*, *rewrite distrTx-RD.prob-space*)
apply (*rewrite ccdfTx-cond-prob*, *simp*)
by (*rewrite not-less*[*THEN sym*], *rewrite MM-PS.cond-prob-neg*; *simp*)

lemma *cdfTx-0-0*: *cdf* (*distr* ($\mathfrak{M} \downharpoonright$ *alive x*) *borel* (*T x*)) *0* = *0*
using *ccdfTx-0-1 distrTx-RD.cdf-ccdf distrTx-RD.prob-space* **by** *force*

lemma *cdfTx-nonpos-0*: *cdf* (*distr* ($\mathfrak{M} \downharpoonright$ *alive x*) *borel* (*T x*)) *t* = *0* **if** $t \leq 0$ **for** *t* :: *real*
using *ccdfTx-nonpos-1 distrTx-RD.cdf-ccdf distrTx-RD.prob-space that* **by** *force*

lemma *cdfTx-1-equiv*: *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t* = *1* ⟷ *x*+*t* ≥ $*ψ*
for *t*::*real*
using *ccdfTx-0-equiv distrTx-RD.cdf-ccdf distrTx-RD.prob-space* **by** *force*

lemma *cdfTx-continuous-on-nonpos*[*simp*]:
continuous-on {..*0*} (*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)))
by (*rewrite continuous-on-cong*[**where** *g*=λ*t*. *0*]) (*simp-all add*: *cdfTx-nonpos-0*)+

lemma *cdfTx-differentiable-on-nonpos*[*simp*]:
(*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) *differentiable-on* {..*0*}
by (*rewrite differentiable-on-cong*[**where** *f*=λ*t*. *0*]; *simp add*: *cdfTx-nonpos-0*)

lemma *cdfTx-has-real-derivative-0-at-neg*:
(*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative 0*) (*at t*) **if** *t* < *0* **for** *t*::*real*
apply (*rewrite has-real-derivative-iff-has-vector-derivative*)
apply (*rule has-vector-derivative-transform-within-open*[*of* λ-. *0* - - {..<*0*}])
using *cdfTx-nonpos-0 that* **by** *simp-all*

lemma *cdfTx-integrable-Icc*:
set-integrable lborel {*a*..*b*} (*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) **for** *a b* :: *real*
proof −
have *set-integrable lborel* {*a*..*b*} (λ-. *1*::*real*)
unfolding *set-integrable-def*
using *emeasure-compact-finite* **by** (*simp*, *intro integrable-real-indicator*; *force*)
thus *?thesis*
apply (*rewrite distrTx-RD.cdf-ccdf*, *rewrite distrTx-RD.prob-space*)
using *ccdfTx-integrable-Icc* **by** (*rewrite set-integral-diff*; *simp*)
qed

corollary *cdfTx-integrable-on-Icc*:
cdf (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *integrable-on* {*a*..*b*} **for** *a b* :: *real*
using *cdfTx-integrable-Icc set-borel-integral-eq-integral* **by** *force*

lemma *cdfTx-PX*:
cdf (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *t* = 𝒫(ξ *in* 𝔐. *x* < *X* ξ ∧ *X* ξ ≤ *x*+*t*) / 𝒫(ξ *in* 𝔐. *X* ξ > *x*)
for *t*::*real*
apply (*rewrite cdfTx-cond-prob*)
unfolding *cond-prob-def futr-life-def PXx-pos* **by** (*smt* (*verit*) *Collect-cong*)

lemma *cdfT0-eq-cdfX*: *cdf* (*distr* (𝔐 ⌊ *alive 0*) *borel* (*T 0*)) = *cdf* (*distr* 𝔐 *borel* *X*)
proof
interpret *alive0-PS*: *prob-space* 𝔐 ⌊ *alive 0*
apply (*rule MM-PS.cond-prob-space-correct*, *simp*)
using *PXx-pos alive-def psi-pos′* **by** *presburger*
interpret *distrT0-RD*: *real-distribution distr* (𝔐 ⌊ *alive 0*) *borel* (*T 0*) **by** *simp*

show $\bigwedge x.$ *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive* 0) *borel* (*T* 0)) *x* = *cdf* (*distr* $\mathfrak{M}$ *borel X*) *x*
using *ccdfT0-eq-ccdfX distrX-RD.ccdf-cdf distrT0-RD.ccdf-cdf*
by (*smt* (*verit*, *best*) *distrT0-RD.prob-space distrX-RD.prob-space psi-pos'*)
qed

lemma *continuous-cdfX-cdfTx*:
continuous (*at* (*x*+*t*) *within* {*x*..}) (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) ⟷
continuous (*at t within* {0..}) (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
if *t* ≥ 0 **for** *t*::*real*
proof −
have *continuous* (*at* (*x*+*t*) *within* {*x*..}) (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) ⟷
continuous (*at* (*x*+*t*) *within* {*x*..}) (*ccdf* (*distr* $\mathfrak{M}$ *borel X*))
by (*rule distrX-RD.continuous-cdf-ccdf*)
also have ... ⟷ *continuous* (*at t within* {0..}) (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
using *continuous-ccdfX-ccdfTx that* **by** *simp*
also have ... ⟷ *continuous* (*at t within* {0..}) (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
using *distrTx-RD.continuous-cdf-ccdf* **by** *simp*
finally show *?thesis* .
qed

lemma *isCont-cdfX-cdfTx*:
isCont (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) (*x*+*t*) ⟷
isCont (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *t*
if *t* > 0 **for** *t*::*real*
apply (*rewrite distrX-RD.isCont-cdf-ccdf*)
apply (*rewrite isCont-ccdfX-ccdfTx*, *simp-all add*: *that*)
by (*rule distrTx-RD.isCont-cdf-ccdf*[*THEN sym*])

lemma *has-real-derivative-cdfX-cdfTx*:
((*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *has-real-derivative D*) (*at* (*x*+*t*)) ⟷
((*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *has-real-derivative* (*D* / $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X* ξ > *x*))) (*at t*)
if *t* > 0 **for** *t D* :: *real*
proof −
have ((*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *has-real-derivative D*) (*at* (*x*+*t*)) ⟷
(*ccdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative* −*D*) (*at* (*x*+*t*))
using *distrX-RD.has-real-derivative-cdf-ccdf* **by** *force*
also have ... ⟷
((*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *has-real-derivative* (−*D* / $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X* ξ > *x*))) (*at t*)
using *has-real-derivative-ccdfX-ccdfTx that* **by** *simp*
also have ... ⟷
((*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *has-real-derivative* (*D* / $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X* ξ > *x*))) (*at t*)
by (*simp add*: *distrTx-RD.has-real-derivative-cdf-ccdf*)
finally show *?thesis* .
qed

lemma *differentiable-cdfX-cdfTx*:
(*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable at* (*x*+*t*) $\longleftrightarrow$
(*cdf* (*distr* ($\mathfrak{M}$ $\restriction$ *alive x*) *borel* (*T x*))) *differentiable at t*
if *t* > *0* **for** *t*::*real*
apply (*rewrite differentiable-eq-field-differentiable-real*)+
unfolding *field-differentiable-def* **using** *has-real-derivative-cdfX-cdfTx that*
by (*meson differentiable-ccdfX-ccdfTx distrTx-RD.finite-borel-measure-axioms distrX-RD.finite-borel-measure-axioms finite-borel-measure.differentiable-cdf-ccdf real-differentiable-def x-lt-psi*)

5.1.9 Properties of $q-{t&x}

lemma *q-nonpos-0*: $q-{t&x} = *0* **if** *t* ≤ *0* **for** *t*::*real*
unfolding *die-def* **using** *that cdfTx-nonpos-0* **by** *simp*

corollary *q-0-0*: $q-{*0*&x} = *0*
using *q-nonpos-0* **by** *simp*

lemma *q-nonneg*[*simp*]: $q-{t&x} ≥ *0* **for** *t*::*real*
unfolding *die-def* **using** *distrTx-RD.cdf-nonneg* **by** *simp*

lemma *q-le-1*[*simp*]: $q-{t&x} ≤ *1* **for** *t*::*real*
unfolding *die-def* **using** *distrTx-RD.cdf-bounded-prob* **by** *force*

lemma *q-1-equiv*: $q-{t&x} = *1* $\longleftrightarrow$ *x*+*t* ≥ $ψ **for** *t*::*real*
unfolding *die-def* **using** *cdfTx-1-equiv* **by** *simp*

lemma *q-PTx*: $q-{t&x} = $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *T x ξ* ≤ *t* | *T x ξ* > *0*) **for** *t*::*real*
unfolding *die-def* **using** *cdfTx-cond-prob* **by** *simp*

lemma *q-PX*: $q-{t&x} = $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *x* < *X ξ* ∧ *X ξ* ≤ *x* + *t*) / $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X ξ* > *x*)
unfolding *die-def* **using** *cdfTx-PX* **by** *simp*

lemma *q-defer-0-q*[*simp*]: $q-{*0*|t&x} = $q-{t&x} **for** *t*::*real*
unfolding *die-defer-def* **using** *q-0-0* **by** *simp*

lemma *q-defer-0-0*: $q-{f|*0*&x} = *0* **for** *f*::*real*
unfolding *die-defer-def* **by** *simp*

lemma *q-defer-nonneg*[*simp*]: $q-{f|t&x} ≥ *0* **for** *f t* :: *real*
unfolding *die-defer-def* **by** *simp*

lemma *q-defer-q*: $q-{f|t&x} = $q-{f+t & x} − $q-{f&x} **if** *t* ≥ *0* **for** *f t* :: *real*
unfolding *die-defer-def die-def* **using** *distrTx-RD.cdf-nondecreasing that* **by** *simp*

corollary *q-defer-le-1*[*simp*]: $q-{f|t&x} ≤ *1* **if** *t* ≥ *0* **for** *f t* :: *real*

by (*smt* (*verit*, *ccfv-SIG*) *q-defer-q q-le-1 q-nonneg that*)

lemma *q-defer-PTx*: $\$q\text{-}\{f|t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t \mid T\ x\ \xi > 0)$
if $t \geq 0$ **for** $f\ t :: real$
proof −
have $\$q\text{-}\{f|t\&x\} = \$q\text{-}\{f{+}t\ \&\ x\} - \$q\text{-}\{f\&x\}$ **using** *q-defer-q that* **by** *simp*
also have $\ldots = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi \leq f + t \mid T\ x\ \xi > 0) - \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi \leq f \mid T\ x\ \xi > 0)$
using *q-PTx* **by** *simp*
also have $\ldots = \mathcal{P}(\xi \text{ in } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f + t) - \mathcal{P}(\xi \text{ in } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f)$
using *MM-PS.cond-prob-space-cond-prob alive-T* **by** *simp*
also have $\ldots = \mathcal{P}(\xi \text{ in } (\mathfrak{M} \restriction \text{alive } x).\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t)$
proof −
have $\{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f + t\} - \{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f\} =$
$\{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t\}$
using *that* **by** *force*
hence *alivex-PS.prob*
$(\{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f + t\} - \{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f\}) =$
$\mathcal{P}(\xi \text{ in } (\mathfrak{M} \restriction \text{alive } x).\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t)$
by *simp*
moreover have $\{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f\} \subseteq \{\xi \in \text{space } (\mathfrak{M} \restriction \text{alive } x).\ T\ x\ \xi \leq f + t\}$
using *that* **by** *force*
ultimately show *?thesis* **by** (*rewrite alivex-PS.finite-measure-Diff*[*THEN sym*]; *simp*)
qed
also have $\ldots = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t \mid T\ x\ \xi > 0)$
using *MM-PS.cond-prob-space-cond-prob alive-T* **by** *simp*
finally show *?thesis* **.**
qed

lemma *q-defer-PX*: $\$q\text{-}\{f|t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ x + f < X\ \xi \wedge X\ \xi \leq x + f + t) \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ X\ \xi > x)$
if $f \geq 0\ t \geq 0$ **for** $f\ t :: real$
proof −
have $\$q\text{-}\{f|t\&x\} = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t \wedge T\ x\ \xi > 0) \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi > 0)$
apply (*rewrite q-defer-PTx*; (*simp add*: *that*)*?*)
unfolding *cond-prob-def* **by** *simp*
also have $\ldots = \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t) \ / \ \mathcal{P}(\xi \text{ in } \mathfrak{M}.\ T\ x\ \xi > 0)$
proof −
have $\bigwedge \xi.\ \xi \in space\ \mathfrak{M} \Longrightarrow f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t \wedge T\ x\ \xi > 0 \longleftrightarrow f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t$
using *that* **by** *auto*

hence $\{\xi \in \mathit{space}\ \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t \wedge T\ x\ \xi > 0\} =$
$\{\xi \in \mathit{space}\ \mathfrak{M}.\ f < T\ x\ \xi \wedge T\ x\ \xi \leq f + t\}$ **by** *blast*
thus *?thesis* **by** *simp*
qed
also have $\ldots = \mathcal{P}(\xi\ \mathit{in}\ \mathfrak{M}.\ x + f < X\ \xi \wedge X\ \xi \leq x + f + t)\ /\ \mathcal{P}(\xi\ \mathit{in}\ \mathfrak{M}.\ X\ \xi > x)$
unfolding *futr-life-def* **by** (*smt* (*verit*) *Collect-cong*)
finally show *?thesis* **.**
qed

lemma *q-defer-old-0*: $\$q\text{-}\{f|t\&x\} = 0$ **if** $x+f \geq \$\psi\ t \geq 0$ **for** $f\ t :: \mathit{real}$
proof −
have $\$q\text{-}\{f|t\&x\} = \$q\text{-}\{f+t\ \&\ x\} - \$q\text{-}\{f\&x\}$ **using** *q-defer-q that* **by** *simp*
moreover have $\$q\text{-}\{f+t\ \&\ x\} = 1$ **using** *q-1-equiv that le-ereal-le* **by** *auto*
moreover have $\$q\text{-}\{f\&x\} = 1$ **using** *q-1-equiv that* **by** *simp*
ultimately show *?thesis* **by** *simp*
qed

end

5.1.10 Properties of Cumulative Distributive Function for X

lemma *cdfX-continuous-unborn*[*simp*]: *continuous-on* $\{..0\}$ (*cdf* (*distr* $\mathfrak{M}$ *borel X*))
using *cdfTx-continuous-on-nonpos* **by** (*metis cdfT0-eq-cdfX psi-pos'*)

lemma *cdfX-differentiable-unborn*[*simp*]: (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable-on* $\{..0\}$
using *cdfTx-differentiable-on-nonpos* **by** (*metis cdfT0-eq-cdfX psi-pos'*)

lemma *cdfX-has-real-derivative-0-unborn*:
(*cdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative 0*) (*at x*) **if** $x < 0$ **for** *x*::*real*
using *cdfTx-has-real-derivative-0-at-neg* **by** (*metis cdfT0-eq-cdfX psi-pos' that*)

lemma *cdfX-integrable-Icc*:
set-integrable lborel $\{a..b\}$ (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) **for** *a b* :: *real*
using *cdfTx-integrable-Icc* **by** (*metis cdfT0-eq-cdfX psi-pos'*)

corollary *cdfX-integrable-on-Icc*:
cdf (*distr* $\mathfrak{M}$ *borel X*) *integrable-on* $\{a..b\}$ **for** *a b* :: *real*
using *cdfX-integrable-Icc set-borel-integral-eq-integral* **by** *force*

lemma *cdfX-q*: *cdf* (*distr* $\mathfrak{M}$ *borel X*) $x = \$q\text{-}\{x\&0\}$ **if** $x \geq 0$ **for** *x*::*real*
by (*metis cdfT0-eq-cdfX die-def psi-pos'*)

5.1.11 Relations between $\$p\text{-}\{t\&x\}$ and $\$q\text{-}\{t\&x\}$

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: $x < \$\psi$
begin

interpretation *alivex-PS*: *prob-space* $\mathfrak{M} \restriction$ *alive x*
by (*rule MM-PS.cond-prob-space-correct*, *simp-all add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*) **by** *simp*

lemma *p-q-1*: $p-{t&x} + $q-{t&x} = *1* **for** *t*::*real*
unfolding *survive-def die-def* **using** *distrTx-RD.add-cdf-ccdf*
by (*smt* (*verit*) *distrTx-RD.prob-space x-lt-psi*)

lemma *q-defer-p*: $q-{f|t&x} = $p-{f&x} − $p-{f+t & x} **if** $t \geq 0$ **for** *f t* :: *real*
using *q-defer-q p-q-1 that x-lt-psi* **by** (*smt* (*verit*))

lemma *q-defer-p-q-defer*: $p-{f&x} ∗ $q-{f′|t & x+f} = $q-{f+f′|t & x}
if $x+f < \$\psi$ $f \geq 0$ $f' \geq 0$ $t \geq 0$ **for** *f f′ t* :: *real*
proof −
have $p-{f&x} ∗ $q-{f′|t & x+f} =
$\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $X\ \xi > x+f) / \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $X\ \xi > x) ∗$
$\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $x+f+f' < X\ \xi \wedge X\ \xi \leq x+f+f'+t) / \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $X\ \xi > x+f)$
apply (*rewrite p-PX*, (*simp-all add*: *that*)[*2*])
by (*rewrite survival-model.q-defer-PX*, *simp-all add*: *that survival-model-axioms*)
also have ... = $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $x+f+f' < X\ \xi \wedge X\ \xi \leq x+f+f'+t) / \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $X\ \xi > x)$
using *survival-model.PXx-pos*[*of* $\mathfrak{M}$ *X x+f*] *nonzero-mult-div-cancel-left that*
by (*smt* (*verit*, *ccfv-SIG*) *survival-model-axioms times-divide-eq-left times-divide-eq-right*)
also have ... = $q-{f+f′|t & x}
by (*rewrite q-defer-PX*; *simp add*: *that group-cancel.add1*)
finally show *?thesis* **.**
qed

lemma *q-defer-pq*: $q-{f|t&x} = $p-{f&x} ∗ $q-{t & x+f}
if $x+f < \$\psi$ $t \geq 0$ $f \geq 0$ **for** *f t* :: *real*
using *q-defer-p-q-defer*[**where** *f′*=*0*] *that*
by (*simp add*: *survival-model.q-defer-0-q survival-model-axioms*)

5.1.12 Properties of Life Expectation

lemma *e-nonneg*: $e`○-x \geq 0$
unfolding *life-expect-def*
by (*rule Bochner-Integration.integral-nonneg*, *simp add*: *less-eq-real-def*)

lemma *e-P*: $e`○-x =$
MM-PS.expectation (λξ. *indicator* (*alive x*) ξ ∗ *T x* ξ) / $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi > 0)$
unfolding *life-expect-def*
by (*rewrite MM-PS.integral-cond-prob-space-nn*, *auto simp add*: *alive-T*)

proposition *nn-integral-T-p*:
($\int^+\xi$. *ennreal* (*T x* ξ) ∂($\mathfrak{M} \restriction$ *alive x*)) = ($\int^+ t \in \{0..\}$. *ennreal* ($p-{t&x}) ∂*lborel*)

apply (*rewrite alivex-PS.expectation-nonneg-tail*, *simp-all add*: *less-imp-le*)
apply (*rule nn-integral-cong*)
unfolding *survive-def* **using** *distrTx-RD.prob-space distrTx-RD.ccdf-cdf* **by** *presburger*

lemma *nn-integral-T-pos*: $(\int^+\xi.\ ennreal\ (T\ x\ \xi)\ \partial(\mathfrak{M} \restriction alive\ x)) > 0$
proof −
let *?f* = $\lambda t.\ -\ ccdf\ (distr\ (\mathfrak{M} \restriction alive\ x)\ borel\ (T\ x))\ t$
have $\bigwedge t\ u.\ t \le u \Longrightarrow$ *?f t* ≤ *?f u* **using** *distrTx-RD.ccdf-nonincreasing* **by** *simp*
moreover have *continuous* (*at-right 0*) *?f*
using *distrTx-RD.ccdf-is-right-cont* **by** (*intro continuous-intros*)
ultimately have $\forall\ e{>}0.\ \exists\ d{>}0.\ ?f\ (0 + d) - ?f\ 0 < e$
using *continuous-at-right-real-increasing* **by** *simp*
hence $\exists\ d{>}0.\ ?f\ (0 + d) - ?f\ 0 < 1/2$ **by** (*smt* (*verit*, *del-insts*) *field-sum-of-halves*)
from *this* **obtain** *d* **where** *d-pos*: $d > 0$ **and** $\$p\text{-}\{d\&x\} \ge 1/2$
using *p-0-1* **unfolding** *survive-def* **by** *auto*
hence $\bigwedge t.\ t{\in}\{0..d\} \Longrightarrow \$p\text{-}\{t\&x\} \ge 1/2$
unfolding *survive-def* **using** *distrTx-RD.ccdf-nonincreasing* **by** *force*
hence $(\int^+ t{\in}\{0..d\}.\ ennreal\ (\$p\text{-}\{t\&x\})\ \partial lborel) \ge (\int^+ t{\in}\{0..d\}.\ ennreal\ (1/2)\ \partial lborel)$
apply (*intro nn-set-integral-mono*, *simp-all*)
unfolding *survive-def* **using** *Tx-alivex-measurable* **apply** *force*
by (*rule AE-I2*) (*smt* (*verit*) *ennreal-half ennreal-leI half-bounded-equal*)
moreover have $(\int^+ t{\in}\{0..\}.\ ennreal\ (\$p\text{-}\{t\&x\})\ \partial lborel) \ge (\int^+ t{\in}\{0..d\}.\ ennreal\ (\$p\text{-}\{t\&x\})\ \partial lborel)$
by (*rule nn-set-integral-set-mono*) *simp*
moreover have $(\int^+ t{\in}\{0..d\}.\ ennreal\ (1/2)\ \partial lborel) > 0$
apply (*rewrite nn-integral-cmult-indicator*, *simp-all*)
using *d-pos emeasure-lborel-Icc ennreal-zero-less-mult-iff* **by** *fastforce*
ultimately show *?thesis* **using** *nn-integral-T-p* **by** *simp*
qed

lemma *e-pos-Tx*: $\$e`\circ\text{-}x > 0$ **if** *integrable* $(\mathfrak{M} \restriction alive\ x)\ (T\ x)$
unfolding *life-expect-def*
apply (*rewrite integral-eq-nn-integral*, *simp-all*)
apply (*smt* (*verit*, *ccfv-SIG*) *AE-I2 alivex-Tx-pos*)
using *nn-integral-T-pos that*
by (*smt* (*verit*) *AE-I2 alivex-Tx-pos enn2real-ennreal ennreal-less-zero-iff nn-integral-cong nn-integral-eq-integral*)

proposition *e-LBINT-p*: $\$e`\circ\text{-}x = (LBINT\ t{:}\{0..\}.\ \$p\text{-}\{t\&x\})$
— Note that $0 = 0$ holds when the integral diverges.
unfolding *life-expect-def* **apply** (*rewrite integral-eq-nn-integral*, *simp-all add*: *less-imp-le*)
unfolding *set-lebesgue-integral-def* **apply** (*rewrite integral-eq-nn-integral*, *simp-all*)
apply (*measurable*, *simp add*: *survive-def*)
by (*rewrite nn-integral-T-p*) (*simp add*: *indicator-mult-ennreal mult.commute*)

corollary *e-integral-p*: $\$e`\circ\text{-}x = integral\ \{0..\}\ (\lambda t.\ \$p\text{-}\{t\&x\})$

— Note that *0 = 0* holds when the integral diverges.
proof −
have $*e*‘∘-*x* = (*LBINT t*:{*0*..}. $*p*-{*t*&*x*}) **using** *e-LBINT-p* **by** *simp*
also have ... = *integral* {*0*..} (λ*t*. $*p*-{*t*&*x*})
apply (*rule set-borel-integral-eq-integral-nonneg*, *simp-all*)
unfolding *survive-def* **by** *simp*
finally show *?thesis* **.**
qed

lemma *e-pos*: $*e*‘∘-*x* > *0* **if** *set-integrable lborel* {*0*..} (λ*t*. $*p*-{*t*&*x*})
proof −
have (∫ +*t*∈{*0*..}. *ennreal* ($*p*-{*t*&*x*}) *∂lborel*) = *ennreal* (∫ *t*∈{*0*..}. $*p*-{*t*&*x*} *∂lborel*)
by (*intro set-nn-integral-eq-set-integral*; *simp add*: *that*)
also have ... < ∞ **using** *that* **by** *simp*
finally have (∫ +ξ. *ennreal* (*T x* ξ) *∂*(𝔐 ⌊ *alive x*)) < ∞ **using** *nn-integral-T-p* **by** *simp*
hence *integrable* (𝔐 ⌊ *alive x*) (*T x*)
by (*smt* (*verit*) *alivex-Tx-pos integrableI-bounded nn-integral-cong real-norm-def survival-model.Tx-alivex-measurable survival-model-axioms*)
thus *?thesis* **by** (*rule e-pos-Tx*)
qed

corollary *e-pos'*: $*e*‘∘-*x* > *0* **if** (λ*t*. $*p*-{*t*&*x*}) *integrable-on* {*0*..}
apply (*rule e-pos*)
using *that* **apply** (*rewrite integrable-on-iff-set-integrable-nonneg*; *simp*)
unfolding *survive-def* **by** *simp*

lemma *e-LBINT-p-Icc*: $*e*‘∘-*x* = (*LBINT t*:{*0*..*n*}. $*p*-{*t*&*x*}) **if** *x*+*n* ≥ $ψ **for** *n*::*real*
proof −
have [*simp*]: {*0*..*n*} ∩ {*n*<..} = {} **using** *ivl-disj-int-one*(*7*) **by** *blast*
have [*simp*]: {*0*..*n*} ∪ {*n*<..} = {*0*..}
by (*smt* (*verit*) *ereal-less-le ivl-disj-un-one*(*7*) *leD that x-lt-psi*)
have [*simp*]: ⋀*t*. *n* < *t* ⟹ *0* ≤ *t* **using** *that x-lt-psi* **by** (*smt* (*verit*) *ereal-less-le leD*)
have [*simp*]: ⋀*t*. *n* < *t* ⟹ $ψ ≤ *ereal* (*x*+*t*) **using** *that* **by** (*simp add*: *le-ereal-le*)
have *gt-n-0*: *has-bochner-integral lborel* (λ*t*. *indicat-real* {*n*<..} *t* ∗ $*p*-{*t*&*x*}) *0*
apply (*rewrite has-bochner-integral-cong*[**where** *N*=*lborel* **and** *g*=λ*t.0* **and** *y*=*0*], *simp-all*)
using *p-0-equiv that x-lt-psi*
apply (*smt* (*verit*, *ccfv-SIG*) *greaterThan-iff indicator-simps le-ereal-le linorder-not-le*)
by (*rule has-bochner-integral-zero*)
hence *gt-n*: *set-integrable lborel* {*n*<..} (λ*t*. $*p*-{*t*&*x*})
unfolding *set-integrable-def* **using** *integrable.simps* **by** *auto*
moreover have *le-n*: *set-integrable lborel* {*0*..*n*} (λ*t*. $*p*-{*t*&*x*})
unfolding *survive-def* **by** (*intro ccdfTx-integrable-Icc*) *simp*
ultimately have *set-integrable lborel* ({*0*..*n*} ∪ {*n*<..}) (λ*t*. $*p*-{*t*&*x*})
using *set-integrable-Un* **by** *force*

hence *set-integrable lborel* {*0*..} (λ*t*. $*p*-{*t*&*x*}) **by** *force*
thus *?thesis*
apply (*rewrite e-LBINT-p*, *simp*)
apply (*rewrite set-integral-Un*[*of* {*0*..*n*} {*n*<..}, *simplified*], *simp-all add*: *gt-n le-n*)
unfolding *set-lebesgue-integral-def* **using** *gt-n-0 has-bochner-integral-integral-eq* **by** *fastforce*
qed

lemma *e-integral-p-Icc*: $*e*‘∘-*x* = *integral* {*0*..*n*} (λ*t*. $*p*-{*t*&*x*}) **if** *x*+*n* ≥ $*ψ* **for** *n*::*real*
using *that* **apply** (*rewrite e-LBINT-p-Icc*, *simp-all*)
using *ccdfTx-integrable-Icc* **unfolding** *survive-def*
by (*rewrite set-borel-integral-eq-integral*; *simp*)

lemma *temp-e-le-n*: $*e*‘∘-{*x*:*n*} ≤ *n* **if** *n* ≥ *0* **for** *n*::*real*
proof −
have *nni-n*: (∫ $^+$-. *ennreal n* ∂(𝔐 ⌊ *alive x*)) = *ennreal n*
by (*rewrite nn-integral-const*, *rewrite alivex-PS.emeasure-space-1*) *simp*
hence *hbi-n*: *has-bochner-integral* (𝔐 ⌊ *alive x*) (λ-. *n*) *n*
by (*intro has-bochner-integral-nn-integral*; *simp add*: *that*)
hence *integrable* (𝔐 ⌊ *alive x*) (λ-. *n*) **by** *simp*
moreover have *integrable* (𝔐 ⌊ *alive x*) (λξ. *min* (*T x* ξ) *n*)
proof −
have (∫ $^+$ξ. *ennreal* (*norm* (*min* (*T x* ξ) *n*)) ∂(𝔐 ⌊ *alive x*)) ≤ ∫ $^+$-. *ennreal n* ∂(𝔐 ⌊ *alive x*)
apply (*rule nn-integral-mono*, *rule ennreal-leI*)
apply (*rewrite real-norm-def*, *rewrite abs-of-nonneg*; *simp add*: *that*)
by (*smt* (*verit*) *alivex-Tx-pos*)
also have ... < ∞ **using** *nni-n* **by** *simp*
finally have (∫ $^+$ξ. *ennreal* (*norm* (*min* (*T x* ξ) *n*)) ∂(𝔐 ⌊ *alive x*)) < ∞ **.**
thus *?thesis* **by** (*intro integrableI-bounded*; *simp*)
qed
ultimately have $*e*‘∘-{*x*:*n*} ≤ *integral*^{L} (𝔐 ⌊ *alive x*) (λ-. *n*)
unfolding *temp-life-expect-def* **by** (*intro integral-mono*; *simp*)
also have ... = *n* **using** *hbi-n has-bochner-integral-iff* **by** *blast*
finally show *?thesis* **.**
qed

lemma *temp-e-P*: $*e*‘∘-{*x*:*n*} =
MM-PS.expectation (λξ. *indicator* (*alive x*) ξ ∗ *min* (*T x* ξ) *n*) / *P*(ξ *in* 𝔐. *T x* ξ > *0*)
if *n* ≥ *0* **for** *n*::*real*
unfolding *temp-life-expect-def*
by (*rewrite MM-PS.integral-cond-prob-space-nn*; *simp add*: *alive-T that*)

lemma *temp-e-LBINT-p*: $*e*‘∘-{*x*:*n*} = (*LBINT t*:{*0*..*n*}. $*p*-{*t*&*x*}) **if** *n* ≥ *0* **for** *n*::*real*
proof −

let *?minTxn* = $\lambda\xi$. *min* (*T x* ξ) *n*
let *?F* = *cdf* (*distr* ($\mathfrak{M}$ $\downharpoonright$ *alive x*) *borel* (*T x*))
let *?Fn* = *cdf* (*distr* ($\mathfrak{M}$ $\downharpoonright$ *alive x*) *borel ?minTxn*)
interpret *distrTxn-RD*: *real-distribution distr* ($\mathfrak{M}$ $\downharpoonright$ *alive x*) *borel ?minTxn* **by** (*simp add*: *that*)
have [*simp*]: $\bigwedge\xi$. $\xi \in$ *space* ($\mathfrak{M}$ $\downharpoonright$ *alive x*) $\Longrightarrow$ *0* $\leq$ *T x* ξ **by** (*smt* (*verit*) *alivex-Tx-pos*)
have ($\int^+\xi$. *ennreal* (*min* (*T x* ξ) *n*) ∂($\mathfrak{M}$ $\downharpoonright$ *alive x*)) = ($\int^+ t\in\{0..\}$. *ennreal* (*1* − *?Fn t*) *∂lborel*)
by (*rewrite alivex-PS.expectation-nonneg-tail*; *simp add*: *that*)
also have ... = ($\int^+ t\in\{0..\}$. (*ennreal* (*1* − *?F t*) ∗ *indicator* {..<*n*} *t*) *∂lborel*)
apply (*rule nn-integral-cong*)
by (*rewrite alivex-PS.cdf-distr-min*; *simp add*: *indicator-mult-ennreal mult.commute*)
also have ... = ($\int^+ t\in\{0..<n\}$. *ennreal* (*1* − *?F t*) *∂lborel*)
apply (*rule nn-integral-cong*) **using** *nn-integral-set-ennreal*
by (*smt* (*verit*, *best*) *Int-def atLeastLessThan-def ennreal-mult-right-cong indicator-simps mem-Collect-eq mult.commute mult-1*)
also have ... = ($\int^+ t\in\{0..n\}$. *ennreal* (*1* − *?F t*) *∂lborel*)
proof −
have *sym-diff* {*0*..<*n*} {*0*..*n*} = {*n*} **using** *that* **by** *force*
thus *?thesis* **by** (*intro nn-integral-null-delta*; *force*)
qed
also have ... = *ennreal* (*LBINT t*:{*0*..*n*}. $p-{*t*&*x*})
proof −
have *set-integrable lborel* {*0*..*n*} (λt. $p-{*t*&*x*})
unfolding *survive-def* **by** (*intro ccdfTx-integrable-Icc*) *simp*
thus *?thesis*
unfolding *set-lebesgue-integral-def* **unfolding** *set-integrable-def*
apply (*rewrite nn-integral-eq-integral*[*THEN sym*]; *simp*)
apply (*rule nn-integral-cong*, *simp*)
unfolding *survive-def* **using** *distrTx-RD.ccdf-cdf distrTx-RD.prob-space nn-integral-set-ennreal*
by (*simp add*: *indicator-mult-ennreal mult.commute*)
qed
finally have ($\int^+\xi$. *ennreal* (*min* (*T x* ξ) *n*) ∂($\mathfrak{M}$ $\downharpoonright$ *alive x*)) =
ennreal (*LBINT t*:{*0*..*n*}. $p-{*t*&*x*}) .
thus *?thesis*
unfolding *temp-life-expect-def* **by** (*rewrite integral-eq-nn-integral*; *simp add*: *that*)
qed

lemma *temp-e-integral-p*: $e`○-{*x*:*n*} = *integral* {*0*..*n*} (λt. $p-{*t*&*x*}) **if** *n* $\geq$ *0* **for** *n*::*real*
using *that* **apply** (*rewrite temp-e-LBINT-p*, *simp-all*)
using *ccdfTx-integrable-Icc* **unfolding** *survive-def*
by (*rewrite set-borel-integral-eq-integral*; *simp*)

lemma *e-eq-temp*: $e`○-*x* = $e`○-{*x*:*n*} **if** *n* $\geq$ *0* *x*+*n* $\geq$ ψ **for** *n*::*real*
using *that e-LBINT-p-Icc temp-e-LBINT-p* **by** *simp*

lemma *curt-e-P*: $*e-x* =
MM-PS.expectation (λξ. *indicator* (*alive x*) ξ ∗ ⌊*T x* ξ⌋) / *P*(ξ *in* 𝔐. *T x* ξ > *0*)
unfolding *curt-life-expect-def*
apply (*rewrite MM-PS.integral-cond-prob-space-nn*; *simp add*: *alive-T*)
by (*metis* (*no-types*, *lifting*) *Bochner-Integration.integral-cong indicator-simps of-int-0 of-int-1*)

lemma *curt-e-sum-P*: $*e-x* = (∑ *k*. *P*(ξ *in* 𝔐. *T x* ξ ≥ *k* + *1* | *T x* ξ > *0*))
if *summable* (λ*k*. *P*(ξ *in* 𝔐. *T x* ξ ≥ *k* + *1* | *T x* ξ > *0*))
proof −
let *?F-flrTx* = *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (λξ. ⌊*T x* ξ⌋))
have [*simp*]: ⋀ξ. ξ ∈ *space* (𝔐 ⌊ *alive x*) ⟹ *0* ≤ *T x* ξ **by** (*smt* (*verit*) *alivex-Tx-pos*)
have *integral*^{N} (𝔐 ⌊ *alive x*) (λξ. *ennreal* ⌊*T x* ξ⌋) =
(∫$^+$*t*∈{*0*..}. *ennreal* (*1* − *?F-flrTx t*) ∂*lborel*)
by (*rewrite alivex-PS.expectation-nonneg-tail*; *simp*)
also have ... = (∫$^+$*t*∈{*0*::*real*..}. *ennreal* *P*(ξ *in* 𝔐. *T x* ξ ≥ ⌊*t*⌋ + *1* | *T x* ξ > *0*) ∂*lborel*)
proof −
{ fix *t*::*real* **assume** *t* ≥ *0*
hence *1* − *?F-flrTx t* = *P*(ξ *in* 𝔐. *T x* ξ ≥ *real-of-int* ⌊*t*⌋ + *1* | *T x* ξ > *0*)
proof −
have *1* − *?F-flrTx t* = *1* − *P*(ξ *in* (𝔐 ⌊ *alive x*). *T x* ξ < *real-of-int* ⌊*t*⌋ + *1*)
by (*rewrite alivex-PS.cdf-distr-floor-P*; *simp*)
also have ... = *1* − *P*(ξ *in* 𝔐. *T x* ξ < *real-of-int* ⌊*t*⌋ + *1* | *T x* ξ > *0*)
using *alive-T* **by** (*rewrite MM-PS.cond-prob-space-cond-prob*; *simp*)
also have ... = *P*(ξ *in* 𝔐. *T x* ξ ≥ *real-of-int* ⌊*t*⌋ + *1* | *T x* ξ > *0*)
by (*rewrite not-le*[*THEN sym*], *rewrite MM-PS.cond-prob-neg*; *simp*)
finally show *?thesis* **.**
qed }
thus *?thesis*
apply −
by (*rule nn-set-integral-cong2*, *rule AE-I2*) *simp*
qed
also have ... = (∑ *k*. ∫$^+$*t*∈{*k*..<*k*+*1*}. *ennreal* *P*(ξ *in* 𝔐. *T x* ξ ≥ ⌊*t*⌋ + *1* | *T x* ξ > *0*) ∂*lborel*)
apply (*rewrite nn-integral-disjoint-family*[*THEN sym*]; *simp*)
apply (*rewrite add.commute*, *rule Ico-nat-disjoint*)
by (*rewrite Ico-nat-union*[*THEN sym*], *simp add*: *add.commute*)
also have ... = (∑ *k*. ∫$^+$*t*∈{*k*..<*k*+*1*::*nat*}. *ennreal* *P*(ξ *in* 𝔐. *T x* ξ ≥ *k* + *1* | *T x* ξ > *0*) ∂*lborel*)
proof −
{ fix *k*::*nat* **and** *t*::*real*
assume *real k* ≤ *t* **and** *t* < *1* + *real k*
hence *real-of-int* ⌊*t*⌋ = *real k*
by (*metis add.commute floor-eq2 of-int-of-nat-eq*)

hence $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq$ *real-of-int* $\lfloor t \rfloor + 1 \mid T\ x\ \xi > 0) =$
$\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq 1 +$ *real* $k \mid T\ x\ \xi > 0)$
by (*simp add*: *add.commute*) **}**
thus *?thesis*
apply −
apply (*rule suminf-cong*, *rule nn-set-integral-cong2*, *rule AE-I2*)
by (*rule impI*) *simp*
qed
also have ... = $(\sum k.$ *ennreal* $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$
by (*rewrite nn-integral-cmult-indicator*; *simp add*: *add.commute*)
also have ... = *ennreal* $(\sum k.\ \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$
by (*rewrite suminf-ennreal2*; *simp add*: *that*)
finally have $integral^N\ (\mathfrak{M} \downharpoonright alive\ x)\ (\lambda\xi.\ ennreal\ \lfloor T\ x\ \xi \rfloor) =$
ennreal $(\sum k.\ \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$ **.**
hence $integral^L\ (\mathfrak{M} \downharpoonright alive\ x)\ (\lambda\xi.\ \lfloor T\ x\ \xi \rfloor) = (\sum k.\ \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$
apply (*rewrite integral-eq-nn-integral*; *simp*)
apply (*rewrite enn2real-ennreal*; *simp add*: *add.commute*)
apply (*rule suminf-nonneg*; *simp?*)
by (*rewrite add.commute*, *simp add*: *that*)
thus *?thesis* **unfolding** *curt-life-expect-def* **by** (*simp add*: *add.commute*)
qed

lemma *curt-e-sum-P-finite*: $\$e\text{-}x = (\sum k{<}n.\ \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$
if $x{+}n{+}1 > \$\psi$ **for** *n*::*nat*
proof −
from *that* **have** *psi-fin*: $\$\psi < \infty$ **by** *force*
let *?P* = $\lambda k{::}nat.\ \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0)$
let *?P-fin* = $\lambda k{::}nat.$ *if* $k{\in}\{..{<}n\}$ *then ?P k else 0*
have $\bigwedge k.$ *?P k = ?P-fin k*
proof −
fix *k*
show *?P k = ?P-fin k*
proof (*cases* ‹$k \in \{..{<}n\}$›)
case *True*
thus *?thesis* **by** *simp*
next
case *False*
hence $\neg\ k < n$ **by** *simp*
hence $x + k + 1 >$ *real-of-ereal* $\$\psi$
using *that psi-nonneg real-of-ereal-ord-simps*(*4*) **by** *fastforce*
hence $\{\xi \in$ *space* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \wedge T\ x\ \xi > 0\} \subseteq \{\xi \in$ *space* $\mathfrak{M}$. $X\ \xi >$ *real-of-ereal* $\$\psi\}$
unfolding *futr-life-def* **using** *that less-ereal-le of-nat-1 of-nat-add* **by** *force*
hence $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \wedge T\ x\ \xi > 0) \leq \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $X\ \xi >$ *real-of-ereal* $\$\psi)$
by (*intro MM-PS.finite-measure-mono*, *simp-all*)
also have ... = *0* **using** *MM-PS.ccdf-distr-P X-RV ccdfX-psi-0 psi-fin* **by**

presburger
    finally have $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \wedge T\ x\ \xi > 0) = 0$ **using** *measure-le-0-iff*
by *blast*
     hence *?P k = 0* **unfolding** *cond-prob-def* **by** (*simp add*: *add.commute*)
     thus *?thesis* **by** *simp*
   qed
  qed
  moreover have *?P-fin sums* ($\sum k{<}n$. *?P k*) **using** *sums-If-finite-set* **by** *force*
  ultimately have $\star$: *?P sums* ($\sum k{<}n$. *?P k*) **using** *sums-cong* **by** *simp*
  moreover hence *summable ?P* **using** *sums-summable* **by** *blast*
  ultimately have *?P sums* \$*e-x* **using** *curt-e-sum-P* **by** *force*
  hence \$*e-x* = ($\sum k{<}n$. *?P k*) **by** (*rewrite sums-unique2*[*of ?P*]; *simp add*: $\star$)
  thus *?thesis* **by** (*simp add*: *add.commute*)
qed

lemma *curt-e-sum-p*: \$*e-x* = ($\sum k$. \$*p-*{*k+1&x*})
  if *summable* (λk. \$*p-*{*k+1&x*}) $\bigwedge k{::}nat$. *isCont* (λt. \$*p-*{*t&x*}) (*k+1*)
proof −
  have $\bigwedge k{::}nat$. \$*p-*{*k+1&x*} = $\mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0)$
    apply (*rewrite p-PTx-ge-ccdf-isCont*, *simp-all*)
    using *that(2) isCont-ccdfX-ccdfTx* **unfolding** *survive-def* **by** *simp*
  thus *?thesis* **using** *that p-PTx-ge-ccdf-isCont curt-e-sum-P* **by** *presburger*
qed

lemma *curt-e-rec*: \$*e-x* = \$*p-x* ∗ (*1* + \$*e-*(*x+1*))
  if *summable* (λk. \$*p-*{*k+1&x*}) $\bigwedge k{::}nat$. *isCont* (λt. \$*p-*{*t&x*}) (*real k + 1*) *x+1* < \$$\psi$
proof −
  have *px-neq-0*[*simp*]: \$*p-x* ≠ *0* **using** *p-0-equiv that* **by** *auto*
  have (λk. \$*p-*{*k+1&x*}) *sums* \$*e-x*
    using *that* **apply** (*rewrite curt-e-sum-p*, *simp-all add*: *add.commute*)
    by (*rule summable-sums*, *simp add*: *that*)
  hence (λk. \$*p-x* ∗ \$*p-*{*k&x+1*}) *sums* \$*e-x*
    apply (*rewrite sums-cong*[**where** *g*=λk. \$*p-*{*k+1&x*}]; *simp?*)
    using *p-mult* **by** (*smt* (*verit*) *of-nat-0-le-iff that(3) x-lt-psi*)
  hence (λk. \$*p-*{*k&x+1*}) *sums* (\$*e-x* / \$*p-x*)
   using *sums-mult-D that* **by** (*smt* (*verit*, *best*) *linorder-not-le p-0-equiv sums-cong*
x-lt-psi)
  hence *p-e-p*: (λk. \$*p-*{*Suc k & x+1*}) *sums* (\$*e-x* / \$*p-x* − \$*p-*{*0&x+1*})
    using *sums-split-initial-segment*[**where** *n=1*] **by** *force*
  moreover have (λk. \$*p-*{*Suc k & x+1*}) *sums* \$*e-*(*x+1*)
  proof −
    have [*simp*]: *summable* ($\lambda k{::}nat$. \$*p-*{*real k + 1 & x + 1*})
      apply (*intro sums-summable*[**where** *l*=\$*e-x* / \$*p-x* − \$*p-*{*0&x+1*}])
      using *p-e-p* **by** (*simp add*: *add.commute*)
    have [*simp*]: $\bigwedge k{::}nat$. *isCont* (λt. \$*p-*{*t&x+1*}) (*real k + 1*)
    proof −
      fix *k*::*nat*
      have *isCont* (λt. \$*p-x* ∗ \$*p-*{*t−1&x+1*}) (*real k + 2*)

```
    proof −
      let ?S={real k + 1 <..< real k + 3}
      have open ?S by simp
      moreover have real k + 2 ∈ ?S by simp
      moreover have ⋀t. t ∈ ?S ⟹ $p-x ∗ $p-{t−1&x+1} = $p-{t&x}
        using p-mult
        by (smt (verit, del-insts) greaterThanLessThan-iff of-nat-0-le-iff that(3)
x-lt-psi)
      ultimately show ?thesis
        apply (rewrite isCont-cong[where g=λt. $p-{t&x}])
        apply (rewrite eventually-nhds, blast)
        using that by (smt (verit) of-nat-1 of-nat-add)
    qed
    hence isCont (λt. $p-x ∗ $p-{t−1&x+1} / $p-x) (real k + 2)
      by (intro isCont-divide[where g=λt. $p-x], auto)
    hence isCont ((λt. $p-{t−1&x+1}) ∘ (λt. t+1)) (real k + 1)
      by simp (rule continuous-at-compose, simp-all add: add.commute)
    thus isCont (λt. $p-{t&x+1}) (real k + 1) unfolding comp-def by simp
  qed
  show ?thesis
    apply (rewrite survival-model.curt-e-sum-p; simp add: survival-model-axioms
that)
    using summable-sums by (rewrite add.commute) force
 qed
 ultimately have $e-x / $p-x − $p-{0&x+1} = $e-(x+1) by (rule sums-unique2)
 thus ?thesis
  using p-0-1 that
  by (smt (verit) px-neq-0 divide-mult-cancel mult.commute mult-cancel-left2
p-mult that(3))
qed

lemma curt-e-sum-p-finite: $e-x = (∑ k<n. $p-{k+1&x})
  if ⋀k::nat. k < n ⟹ isCont (λt. $p-{t&x}) (real k + 1) x+n+1 > $ψ for
n::nat
proof −
  have ⋀k::nat. k < n ⟹ $p-{k+1&x} = 𝒫(ξ in 𝔐. T x ξ ≥ k + 1 | T x ξ >
0)
   apply (rewrite p-PTx-ge-ccdf-isCont, simp-all)
  using that isCont-ccdfX-ccdfTx unfolding survive-def by (smt (verit) of-nat-0-le-iff
x-lt-psi)
  thus ?thesis using that p-PTx-ge-ccdf-isCont curt-e-sum-P-finite by simp
qed

lemma temp-curt-e-P: $e-{x:n} =
  MM-PS.expectation (λξ. indicator (alive x) ξ ∗ ⌊min (T x ξ) n⌋) / 𝒫(ξ in 𝔐. T
x ξ > 0)
  if n ≥ 0 for n::real
  unfolding temp-curt-life-expect-def
  apply (rewrite MM-PS.integral-cond-prob-space-nn; simp add: alive-T that)
```

apply (*rule disjI2*, *rule Bochner-Integration.integral-cong*; *simp*)
using *indicator-simps of-int-0 of-int-1* **by** (*smt* (*verit*))

lemma *temp-curt-e-sum-P*: $*e*-{*x*:*n*} = (∑ *k*<*n*. *P*(*ξ in* 𝔐. *T x ξ* ≥ *k* + *1* | *T x ξ* > *0*)) **for** *n*::*nat*
proof −
let *?F-flrminTx* = *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (λ*ξ*. ⌊*min* (*T x ξ*) *n*⌋))
have [*simp*]: ⋀*ξ*. *ξ* ∈ *space* (𝔐 ⌊ *alive x*) ⟹ *0* ≤ *T x ξ* **by** (*smt* (*verit*) *alivex-Tx-pos*)
have *integral*^{N} (𝔐 ⌊ *alive x*) (λ*ξ*. *ennreal* ⌊*min* (*T x ξ*) *n*⌋) =
(∫$^+$*t*∈{*0*..}. *ennreal* (*1* − *?F-flrminTx t*) *∂lborel*)
by (*rewrite alivex-PS.expectation-nonneg-tail*; *simp*)
also have ... = (∫$^+$*t*∈{*0*::*real*..}. *ennreal*
((*1* − *P*(*ξ in* (𝔐 ⌊ *alive x*). *T x ξ* < ⌊*t*⌋ + *1*)) ∗ *of-bool* (⌊*t*⌋ + *1* ≤ *n*)) *∂lborel*)
proof −
have ⋀*t*. *t* ≥ *0* ⟹ *ennreal* (*1* − *?F-flrminTx t*) =
ennreal ((*1* − *P*(*ξ in* (𝔐 ⌊ *alive x*). *T x ξ* < ⌊*t*⌋ + *1*)) ∗ *of-bool* (⌊*t*⌋ + *1* ≤ *n*))
proof −
fix *t*::*real* **assume** *t* ≥ *0*
thus *ennreal* (*1* − *?F-flrminTx t*) =
ennreal ((*1* − *P*(*ξ in* (𝔐 ⌊ *alive x*). *T x ξ* < ⌊*t*⌋ + *1*)) ∗ *of-bool* (⌊*t*⌋ + *1* ≤ *n*))
proof (*cases* ‹⌊*t*⌋ + *1* ≤ *n*›)
case *True*
thus *?thesis*
apply (*rewrite alivex-PS.cdf-distr-floor-P*; *simp*)
using *min-less-iff-disj*
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong*
floor-eq floor-less-cancel floor-of-nat of-int-floor-le)
next
case *False*
thus *?thesis*
apply (*rewrite alivex-PS.cdf-distr-floor-P*; *simp*)
using *min-less-iff-disj*
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong MM-PS.space-cond-prob-space alive-T alive-event*
alivex-PS.prob-space mem-Collect-eq of-int-of-nat-eq of-nat-less-of-int-iff)
qed
qed
thus *?thesis*
by (*intro nn-set-integral-cong2*, *intro AE-I2*) *auto*
qed
also have ... = (∫$^+$*t*∈{*0*..<*n*}. *ennreal* (*1* − *P*(*ξ in* (𝔐 ⌊ *alive x*). *T x ξ* < ⌊*t*⌋ + *1*)) *∂lborel*)
proof −
{ **fix** *t*::*real*
have (⌊*t*⌋ + *1* ≤ *n*) = (*t* < *n*) **by** *linarith*
hence ⋀*r*::*real*.

```
      ennreal (r * of-bool (⌊t⌋ + 1 ≤ n)) * indicator {0..} t = ennreal r * indicator
{0..<n} t
       unfolding atLeastLessThan-def by (simp add: indicator-def) }
    thus ?thesis by simp
  qed
  also have ... = (∫⁺t∈{0..<n}. ennreal 𝒫(ξ in (𝔐 ⌊ alive x). T x ξ ≥ ⌊t⌋ +
1) ∂lborel)
    by (rewrite alivex-PS.prob-neg[THEN sym]; simp add: not-less)
  also have ... = (∑k<n. ∫⁺t∈{k..<k+1}. ennreal 𝒫(ξ in (𝔐 ⌊ alive x). T x ξ
≥ ⌊t⌋ + 1) ∂lborel)
    apply (rewrite  Ico-nat-union-finite[of n, THEN sym])
    apply (rewrite nn-integral-disjoint-family-on-finite; simp add: add.commute)
    apply (rule disjoint-family-on-mono[of - {0..}]; simp)
    by (rewrite add.commute, rule Ico-nat-disjoint)
  also have ... = (∑k<n. ennreal 𝒫(ξ in 𝔐. T x ξ ≥ k + 1 | T x ξ > 0))
  proof −
    { fix k::nat
      assume k < n
      hence (∫⁺t∈{k..<(1 + real k)}.
        ennreal 𝒫(ξ in (𝔐 ⌊ alive x). T x ξ ≥ real-of-int ⌊t⌋ + 1) ∂lborel) =
        𝒫(ξ in 𝔐. T x ξ ≥ 1 + real k | T x ξ > 0) (is ?LHS = ?RHS)
      proof −
        have ?LHS = (∫⁺t∈{k..<(1 + real k)}. ennreal 𝒫(ξ in (𝔐 ⌊ alive x). T x
ξ ≥ k + 1) ∂lborel)
        proof −
          { fix t::real
            assume k ≤ t t < 1 + real k
            hence real-of-int ⌊t⌋ = real k by (smt (verit) floor-eq2 of-int-of-nat-eq)
            hence 𝒫(ξ in (𝔐 ⌊ alive x). T x ξ ≥ real-of-int ⌊t⌋ + 1) =
              𝒫(ξ in (𝔐 ⌊ alive x). T x ξ ≥ 1 + real k)
              by (simp add: add.commute) }
          thus ?thesis by (intro nn-set-integral-cong2, intro AE-I2) auto
        qed
        also have ... = ennreal 𝒫(ξ in (𝔐 ⌊ alive x). T x ξ ≥ k + 1)
          by (rewrite nn-integral-cmult-indicator; simp)
        also have ... = ?RHS
          by (rewrite MM-PS.cond-prob-space-cond-prob, simp-all add: alive-T)
        finally show ?thesis .
      qed }
    thus ?thesis by (intro sum.cong) auto
  qed
  also have ... = ennreal (∑k<n. 𝒫(ξ in 𝔐. T x ξ ≥ k + 1 | T x ξ > 0)) by
simp
  finally have integral^N (𝔐 ⌊ alive x) (λξ. ennreal ⌊min (T x ξ) n⌋) =
    ennreal (∑k<n. 𝒫(ξ in 𝔐. T x ξ ≥ k + 1 | T x ξ > 0)) .
  hence integral^L (𝔐 ⌊ alive x) (λξ. ⌊min (T x ξ) n⌋) =
    (∑k<n. 𝒫(ξ in 𝔐. T x ξ ≥ k + 1 | T x ξ > 0))
    apply (intro nn-integral-eq-integrable[THEN iffD1, THEN conjunct2]; simp)
    using MM-PS.cond-prob-nonneg by (meson sum-nonneg)
```

thus *?thesis* **unfolding** *temp-curt-life-expect-def* **by** *simp*
qed

corollary *curt-e-eq-temp*: $\$e\text{-}x = \$e\text{-}\{x{:}n\}$ **if** $x{+}n{+}1 > \$\psi$ **for** *n*::*nat*
using *curt-e-sum-P-finite temp-curt-e-sum-P that* **by** *simp*

lemma *temp-curt-e-sum-p*: $\$e\text{-}\{x{:}n\} = (\sum k{<}n.\ \$p\text{-}\{k{+}1\&x\})$
if $\bigwedge k{::}nat.\ k < n \Longrightarrow isCont\ (\lambda t.\ \$p\text{-}\{t\&x\})\ (real\ k + 1)$ **for** *n*::*nat*
proof −
have $\bigwedge k{::}nat.\ k < n \Longrightarrow \$p\text{-}\{k{+}1\&x\} = \mathcal{P}(\xi\ in\ \mathfrak{M}.\ T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0)$
apply (*rewrite p-PTx-ge-ccdf-isCont*, *simp-all*)
using *that isCont-ccdfX-ccdfTx* **unfolding** *survive-def* **by** (*smt* (*verit*) *of-nat-0-le-iff x-lt-psi*)
thus *?thesis*
apply (*rewrite temp-curt-e-sum-P*)
by (*rule sum.cong*) *simp-all*
qed

lemma *temp-curt-e-rec*: $\$e\text{-}\{x{:}n\} = \$p\text{-}x * (1 + \$e\text{-}\{x{+}1{:}n{-}1\})$
if $\bigwedge k{::}nat.\ k < n \Longrightarrow isCont\ (\lambda t.\ \$p\text{-}\{t\&x\})\ (real\ k + 1)\ x{+}1 < \$\psi\ n \neq 0$ **for** *n*::*nat*
proof −
have [*simp*]: $\$p\text{-}x \neq 0$ **using** *p-0-equiv that* **by** *auto*
have [*simp*]: $\bigwedge k.\ k < n{-}1 \Longrightarrow isCont\ (\lambda t.\ \$p\text{-}\{t\&x{+}1\})\ (real\ k + 1)$
proof −
fix *k*::*nat* **assume** *k-n*: $k < n{-}1$
have $isCont\ (\lambda t.\ \$p\text{-}x * \$p\text{-}\{t{-}1\&x{+}1\})\ (real\ k + 2)$
proof −
let $?S{=}\{real\ k + 1 <..< real\ k + 3\}$
have *open ?S* **by** *simp*
moreover have $real\ k + 2 \in ?S$ **by** *simp*
moreover have $\bigwedge t.\ t \in ?S \Longrightarrow \$p\text{-}x * \$p\text{-}\{t{-}1\&x{+}1\} = \$p\text{-}\{t\&x\}$
using *p-mult*
by (*smt* (*verit*, *del-insts*) *greaterThanLessThan-iff of-nat-0-le-iff that*(*2*) *x-lt-psi*)
ultimately show *?thesis*
apply (*rewrite isCont-cong*[**where** $g{=}\lambda t.\ \$p\text{-}\{t\&x\}$])
apply (*rewrite eventually-nhds*, *blast*)
using *that k-n* **by** (*smt* (*verit*) *less-diff-conv of-nat-1 of-nat-add*)
qed
hence $isCont\ (\lambda t.\ \$p\text{-}x * \$p\text{-}\{t{-}1\&x{+}1\} / \$p\text{-}x)\ (real\ k + 2)$
by (*intro isCont-divide*[**where** $g{=}\lambda t.\ \$p\text{-}x$], *auto*)
hence $isCont\ ((\lambda t.\ \$p\text{-}\{t{-}1\&x{+}1\}) \circ (\lambda t.\ t{+}1))\ (real\ k + 1)$
by *simp* (*rule continuous-at-compose*, *simp-all add*: *add.commute*)
thus $isCont\ (\lambda t.\ \$p\text{-}\{t\&x{+}1\})\ (real\ k + 1)$ **unfolding** *comp-def* **by** *simp*
qed
have $\$p\text{-}x * (1 + \$e\text{-}\{x{+}1{:}n{-}1\}) = \$p\text{-}x * (1 + (\sum k{<}(n{-}1).\ \$p\text{-}\{k{+}1\&x{+}1\}))$
by (*rewrite survival-model.temp-curt-e-sum-p*; *simp add*: *survival-model-axioms*

that)
also have ... = $p-x + (∑ k<(n−1). $p-x ∗ $p-{k+1&x+1})
apply (*rewrite distrib-left*, *simp*)
by (*rewrite vector-space-over-itself.scale-sum-right*, *simp*)
also have ... = $p-x + (∑ k<(n−1). $p-{k+2&x})
apply (*rewrite sum.cong*, *simp-all add*: *add.commute*)
using *p-mult* **by** (*smt* (*verit*) *of-nat-0-le-iff that*(2) *x-lt-psi*)
also have ... = (∑ k<Suc(n−1). $p-{k+1&x})
apply (*rewrite lessThan-atLeast0*)+
by (*rewrite sum.atLeast0-lessThan-Suc-shift*) *simp*
also have ... = $e-{x:n} **using** *that* **by** (*rewrite temp-curt-e-sum-p*; *simp*)
finally show *?thesis* **by** *simp*
qed

end

lemma *p-set-integrable-shift*:
set-integrable lborel {0..} (λt. $p-{t&0}) ⟷ *set-integrable lborel* {0..} (λt. $p-{t&x})
if *x* < $ψ **for** *x*::*real*
proof −
have *set-integrable lborel* {0..} (λt. $p-{t&0}) ⟷ *set-integrable lborel* {x..} (λt. $p-{t&0})
by (*rule set-integrable-Ici-equiv*)
(*metis* (*no-types*, *lifting*) *ccdfX-integrable-Icc ccdfX-p set-integrable-cong*)
also have ... ⟷ *set-integrable lborel* {0..} (λt. $p-{x+t&0})
using *set-integrable-Ici-shift*[*of x x*] **by** *force*
also have ... ⟷ *set-integrable lborel* {0..} (λt. $p-{x+t&0} / $p-{x&0})
using *that p-0-equiv* **by** (*rewrite set-integrable-mult-divide-iff*; *simp*)
also have ... ⟷ *set-integrable lborel* {0..} (λt. $p-{t&x})
by (*rule set-integrable-cong*; *simp*) (*simp add*: *ccdfTx-ccdfX ccdfX-p survive-def that*)
finally show *?thesis* **.**
qed

lemma *e-p-e*: $e‘○-x = $e‘○-{x:n} + $p-{n&x} ∗ $e‘○-(x+n)
if *set-integrable lborel* {0..} (λt. $p-{t&x}) *n* ≥ *0 x*+*n* < $ψ **for** *x n* :: *real*
proof −
have [*simp*]: *ereal x* < $ψ **using** *that* **by** (*simp add*: *ereal-less-le*)
hence $e‘○-x = (*LBINT t*:{0..}. $p-{t&x}) **by** (*simp add*: *e-LBINT-p*)
also have ... = (*LBINT t*:{0..n}. $p-{t&x}) + (*LBINT t*:{n..}. $p-{t&x})
proof −
have *AE t in lborel*. ¬ (*t*∈{0..n} ∧ *t*∈{n..}) **using** *AE-lborel-singleton* **by** *force*
moreover have {0..} = {0..n} ∪ {n..} **using** *that* **by** *auto*
moreover have *set-integrable lborel* {0..n} (λt. $p-{t&x})
using *that*
by (*metis Icc-subset-Ici-iff atLeastAtMost-borel order.refl set-integrable-subset sets-lborel*)
moreover have *set-integrable lborel* {n..} (λt. $p-{t&x})

```
      using that by (metis atLeast-borel atLeast-subset-iff set-integrable-subset
sets-lborel)
    ultimately show ?thesis
      using set-integral-Un-AE
    by (metis (no-types, lifting) AE-cong atLeastAtMost-borel atLeast-borel sets-lborel)
  qed
  also have ... = (LBINT t:{0..n}. $p-{t&x}) + $p-{n&x} * (LBINT t:{0..}.
$p-{t & x+n})
  proof −
    have (LBINT t:{n..}. $p-{t&x}) = (LBINT t:{0..}. $p-{n+t & x})
      using lborel-set-integral-Ici-shift[of n - n, simplified] by force
    also have ... = (LBINT t:{0..}. $p-{n&x} * $p-{t & x+n})
      apply (rule set-lebesgue-integral-cong; simp)
      using that p-mult by force
    finally show ?thesis by simp
  qed
  also have ... = $e`○-{x:n} + $p-{n&x} * $e`○-(x+n)
    apply (rewrite temp-e-LBINT-p, (simp-all add: that)[2])
    by (rewrite e-LBINT-p; simp add: that)
  finally show ?thesis .
qed

proposition x-ex-mono: x + $e`○-x ≤ y + $e`○-y if x ≤ y y < $ψ for x y :: real
proof −
  have x-lt-psi[simp]: ereal x < $ψ using that ereal-less-le by simp
  show ?thesis
  proof (cases ‹set-integrable lborel {0..} (λt. $p-{t&x})›)
    case True
     hence $e`○-x = $e`○-{x:y−x} + $p-{y−x&x}*$e`○-y by (rewrite e-p-e[of x
y−x]; simp add: that)
    also have ... ≤ y − x + $e`○-y
    proof −
      have $e`○-{x:y−x} ≤ y − x using that by (intro temp-e-le-n; simp)
      moreover have $p-{y−x&x}*$e`○-y ≤ $e`○-y
        using p-le-1 x-lt-psi that
        by (smt (verit, ccfv-threshold) e-nonneg mult-less-cancel-right1)
      ultimately show ?thesis by simp
    qed
    finally show ?thesis by simp
  next
    case False
    hence $e`○-x = 0
      using e-LBINT-p not-integrable-integral-eq
      unfolding set-integrable-def set-lebesgue-integral-def
      by simp
    moreover have $e`○-y = 0
    proof −
      have ¬ set-integrable lborel {0..} (λt. $p-{t&y})
        using that False
```

```
      apply (rewrite p-set-integrable-shift[THEN sym], simp)
      by (rewrite p-set-integrable-shift[of x]; simp)
    thus ?thesis
      using e-LBINT-p not-integrable-integral-eq that
      unfolding set-integrable-def set-lebesgue-integral-def
      by simp
  qed
  ultimately show ?thesis using that by simp
 qed
qed

proposition x-ex-const-equiv: x + $e`○-x = y + $e`○-y ⟷ $q-{y−x&x} = 0
 if set-integrable lborel {0..} (λt. $p-{t&0}) x ≤ y y < $ψ for x y :: real
proof −
 have ey: set-integrable lborel {0..} (λt. $p-{t&y}) using that p-set-integrable-shift
by blast
 have x-lt-psi[simp]: ereal x < $ψ using that ereal-less-le by simp
 hence ex: set-integrable lborel {0..} (λt. $p-{t&x}) using that p-set-integrable-shift
by blast
 show ?thesis
 proof
  assume const: x + $e`○-x = y + $e`○-y
  hence 0 = y − x − $e`○-x + $e`○-y by simp
  also have ... = y − x − $e`○-{x:y−x} − $p-{y−x&x}*$e`○-y + $e`○-y
    using e-p-e[of x y−x] ex that by simp
  also have ... = (y − x − $e`○-{x:y−x}) + (1 − $p-{y−x&x})*$e`○-y
    by (simp add: left-diff-distrib)
  finally have 0 = (y − x − $e`○-{x:y−x}) + (1 − $p-{y−x&x})*$e`○-y .
  moreover have y − x − $e`○-{x:y−x} ≥ 0 using temp-e-le-n that by simp
  ultimately have (1 − $p-{y−x&x})*$e`○-y = 0
      by (smt (verit, ccfv-threshold) e-nonneg mult-nonneg-nonneg p-le-1 that
x-lt-psi)
  moreover have $e`○-y > 0 using that e-pos ey by simp
  ultimately have 1 − $p-{y−x&x} = 0 by simp
  thus $q-{y−x&x} = 0 by (smt (verit) p-q-1 x-lt-psi)
 next
  interpret alivex-PS: prob-space 𝔐 ⌊ alive x
    by (rule MM-PS.cond-prob-space-correct, simp-all add: alive-def)
  interpret distrTx-RD: real-distribution distr (𝔐 ⌊ alive x) borel (T x) by simp
  assume $q-{y−x&x} = 0
  hence p1: $p-{y−x&x} = 1 using p-q-1 by (metis add.right-neutral x-lt-psi)
  hence ⋀t. t∈{0..y−x} ⟹ $p-{t&x} = 1
    unfolding survive-def using distrTx-RD.ccdf-nonincreasing
    by simp (smt (verit) distrTx-RD.ccdf-bounded-prob)
  hence $e`○-{x:y−x} = y − x
    using that apply (rewrite temp-e-LBINT-p; simp)
    by (rewrite set-lebesgue-integral-cong[where g=λ-. 1]; simp)
  moreover have $e`○-x = $e`○-{x:y−x} + $p-{y−x&x}*$e`○-y
    by (rewrite e-p-e[of x y−x]; simp add: that ex)
```

ultimately show $x + \$e`\circ`\text{-}x = y + \$e`\circ`\text{-}y$ **using** *p1* **by** *simp*
qed
qed

end

5.2 Piecewise Differentiable Survival Function

locale *smooth-survival-function* = *survival-model* +
assumes *ccdfX-piecewise-differentiable*[*simp*]:
(*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) *piecewise-differentiable-on UNIV*
begin

interpretation *distrX-RD*: *real-distribution distr* $\mathfrak{M}$ *borel X*
using *MM-PS.real-distribution-distr* **by** *simp*

5.2.1 Properties of Survival Function for X

lemma *ccdfX-continuous*[*simp*]: *continuous-on UNIV* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*))
using *ccdfX-piecewise-differentiable piecewise-differentiable-on-imp-continuous-on*
by *fastforce*

corollary *ccdfX-borel-measurable*[*measurable*]: *ccdf* (*distr* $\mathfrak{M}$ *borel X*) $\in$ *borel-measurable borel*
by (*rule borel-measurable-continuous-onI*) *simp*

lemma *ccdfX-nondifferentiable-finite-set*[*simp*]:
finite {*x*. $\neg$ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*}
proof −
obtain *S* **where**
fin: *finite S* **and** $\bigwedge x.\ x \notin S \Longrightarrow$ (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable at x*
using *ccdfX-piecewise-differentiable* **unfolding** *piecewise-differentiable-on-def*
by *blast*
hence {*x*. $\neg$ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*} $\subseteq$ *S* **by** *blast*
thus *?thesis* **using** *finite-subset fin* **by** *blast*
qed

lemma *ccdfX-differentiable-open-set*: *open* {*x*. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*}
using *ccdfX-nondifferentiable-finite-set finite-imp-closed*
by (*metis* (*mono-tags*, *lifting*) *Collect-cong open-Collect-neg*)

lemma *ccdfX-differentiable-borel-set*[*measurable*, *simp*]:
{*x*. *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*} $\in$ *sets borel*
using *ccdfX-differentiable-open-set* **by** *simp*

lemma *ccdfX-differentiable-AE*:
AE x in lborel. (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable at x*
apply (*rule AE-I'*[*of* {*x*. $\neg$ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*}], *simp-all*)
using *ccdfX-nondifferentiable-finite-set* **by** (*simp add*: *finite-imp-null-set-lborel*)

lemma *deriv-ccdfX-measurable*[*measurable*]: *deriv* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) ∈ *borel-measurable borel*
proof −
have *set-borel-measurable borel UNIV* (*deriv* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)))
by (*rule piecewise-differentiable-on-deriv-measurable-real*; *simp*)
thus *?thesis* **unfolding** *set-borel-measurable-def* **by** *simp*
qed

5.2.2 Properties of Cumulative Distributive Function for X

lemma *cdfX-piecewise-differentiable*[*simp*]:
(*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *piecewise-differentiable-on UNIV*
by (*rewrite distrX-RD.cdf-ccdf*) (*rule piecewise-differentiable-diff*; *simp*)

lemma *cdfX-continuous*[*simp*]: *continuous-on UNIV* (*cdf* (*distr* $\mathfrak{M}$ *borel X*))
using *cdfX-piecewise-differentiable piecewise-differentiable-on-imp-continuous-on*
by *fastforce*

corollary *cdfX-borel-measurable*[*measurable*]: *cdf* (*distr* $\mathfrak{M}$ *borel X*) ∈ *borel-measurable borel*
by (*rule borel-measurable-continuous-onI*) *simp*

lemma *cdfX-nondifferentiable-finite-set*[*simp*]:
finite {*x*. ¬ *cdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*}
using *distrX-RD.differentiable-cdf-ccdf ccdfX-nondifferentiable-finite-set* **by** *simp*

lemma *cdfX-differentiable-open-set*: *open* {*x*. *cdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*}
using *distrX-RD.differentiable-cdf-ccdf ccdfX-differentiable-open-set* **by** *simp*

lemma *cdfX-differentiable-borel-set*[*measurable*, *simp*]:
{*x*. *cdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x*} ∈ *sets borel*
using *cdfX-differentiable-open-set* **by** *simp*

lemma *cdfX-differentiable-AE*:
AE x in lborel. (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *differentiable at x*
using *ccdfX-differentiable-AE distrX-RD.differentiable-cdf-ccdf AE-iffI* **by** *simp*

lemma *deriv-cdfX-measurable*[*measurable*]: *deriv* (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) ∈ *borel-measurable borel*
proof −
have *set-borel-measurable borel UNIV* (*deriv* (*cdf* (*distr* $\mathfrak{M}$ *borel X*)))
by (*rule piecewise-differentiable-on-deriv-measurable-real*; *simp*)
thus *?thesis* **unfolding** *set-borel-measurable-def* **by** *simp*
qed

5.2.3 Introduction of Probability Density Functions of X and $T(x)$

definition *pdfX* :: *real* ⇒ *real*
  where *pdfX x* ≡ *if cdf* (*distr* 𝔐 *borel X*) *differentiable at x*
    then deriv (*cdf* (*distr* 𝔐 *borel X*)) *x else 0*
    — This function is defined to be always nonnegative for future application.

definition *pdfT* :: *real* ⇒ *real* ⇒ *real*
  where *pdfT x t* ≡ *if cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*
    then deriv (*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) *t else 0*
    — This function is defined to be always nonnegative for future application.

lemma *pdfX-measurable*[*measurable*]: *pdfX* ∈ *borel-measurable borel*
proof −
  let *?cdfX* = *cdf* (*distr* 𝔐 *borel X*)
  have *countable* {*x*. ¬ *?cdfX differentiable at x*}
    using *cdfX-nondifferentiable-finite-set uncountable-infinite* **by** *force*
  thus *?thesis*
    unfolding *pdfX-def*
    apply −
    by (*rule measurable-discrete-difference*
        [**where** *X*={*x*. ¬ *?cdfX differentiable at x*} **and** *f*=*deriv ?cdfX*]; *simp*)
qed

lemma *distributed-pdfX*: *distributed* 𝔐 *lborel X pdfX*
proof −
  let *?cdfX* = *cdf* (*distr* 𝔐 *borel X*)
  obtain *S* **where** *fin*: *finite S* **and** *diff*: ⋀*t*. *t* ∉ *S* ⟹ *?cdfX differentiable at t*
    using *cdfX-piecewise-differentiable* **unfolding** *piecewise-differentiable-on-def*
by *blast*
  { fix *t*::*real* **assume** *t-notin*: *t* ∉ *S*
    have *?cdfX differentiable at t* **using** *diff t-notin* **by** *simp*
    hence (*?cdfX has-real-derivative pdfX t*) (*at t*)
      unfolding *pdfX-def* **using** *DERIV-deriv-iff-real-differentiable* **by** *auto* **}**
  thus *?thesis*
    by (*intro MM-PS.distributed-deriv-cdf*[**where** *S*=*S*]; *simp add*: *fin*)
qed

lemma *pdfT0-X*: *pdfT 0* = *pdfX*
  unfolding *pdfT-def pdfX-def* **using** *cdfT0-eq-cdfX psi-pos′* **by** *fastforce*

5.2.4 Properties of Survival Function for $T(x)$

context
  fixes *x*::*real*
  assumes *x-lt-psi*[*simp*]: *x* < $*ψ*
begin

interpretation *alivex-PS*: *prob-space* 𝔐 ⌊ *alive x*

by (*rule MM-PS.cond-prob-space-correct*; *simp add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*) **by** *simp*

lemma *ccdfTx-continuous-on-nonneg*[*simp*]:
continuous-on {*0*..} (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
apply (*rewrite continuous-on-eq-continuous-within*, *auto*)
apply (*rewrite continuous-ccdfX-ccdfTx*[*THEN sym*], *auto*)
by (*metis UNIV-I ccdfX-continuous continuous-at-imp-continuous-at-within continuous-on-eq-continuous-within*)

lemma *ccdfTx-continuous*[*simp*]: *continuous-on UNIV* (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
proof −
have [*simp*]: {*..0*::*real*} ∪ {*0*..} = *UNIV* **by** *auto*
have *continuous-on* {*..0*} (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
by (*rule ccdfTx-continuous-on-nonpos*) *simp*
moreover have *continuous-on* {*0*..} (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) **by** *simp*
ultimately have *continuous-on* ({*..0*} ∪ {*0*..}) (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
by (*intro continuous-on-closed-Un*) *simp-all*
thus *?thesis* **by** *simp*
qed

corollary *ccdfTx-borel-measurable*[*measurable*]:
ccdf (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) ∈ *borel-measurable borel*
by (*rule borel-measurable-continuous-onI*) *simp*

lemma *ccdfTx-nondifferentiable-finite-set*[*simp*]:
finite {*t*. ¬ *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}
proof −
let *?P* = λ*t*. *ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*
have {*t*. *t* < *0* ∧ ¬ *?P t*} = {}
proof (*rule equals0I*)
fix *t* **assume** *asm*: *t* ∈ {*t*. *t* < *0* ∧ ¬ *?P t*}
hence *?P t* **using** *ccdfTx-has-real-derivative-0-at-neg real-differentiable-def x-lt-psi* **by** *blast*
with *asm* **show** *False* **by** *simp*
qed
hence {*t*. ¬ *?P t*} ⊆ *insert 0* {*t*. *t* > *0* ∧ ¬ *?P t*} **by** *force*
moreover have *finite* {*t*. *t* > *0* ∧ ¬ *?P t*}
proof −
have {*t*. ¬ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at* (*x*+*t*)} =
plus (−*x*) ‘ {*s*. ¬ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at s*}
by *force*
hence *finite* {*t*. ¬ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at* (*x*+*t*)}
using *ccdfX-nondifferentiable-finite-set* **by** *simp*

thus *?thesis*
using *finite-subset differentiable-ccdfX-ccdfTx*
by (*metis* (*no-types*, *lifting*) *mem-Collect-eq subsetI x-lt-psi*)
qed
ultimately show *?thesis* **using** *finite-subset* **by** *auto*
qed

lemma *ccdfTx-differentiable-open-set*:
open {*t*. *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}
using *ccdfTx-nondifferentiable-finite-set finite-imp-closed*
by (*metis* (*mono-tags*, *lifting*) *Collect-cong open-Collect-neg*)

lemma *ccdfTx-differentiable-borel-set*[*measurable*, *simp*]:
{*t*. *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*} ∈ *sets borel*
using *ccdfTx-differentiable-open-set* **by** *simp*

lemma *ccdfTx-differentiable-AE*:
AE t in lborel. (*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) *differentiable at t*
apply (*rule AE-I'*[*of* {*t*. ¬ *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}]; *simp?*)
using *ccdfTx-nondifferentiable-finite-set* **by** (*simp add*: *finite-imp-null-set-lborel*)

lemma *ccdfTx-piecewise-differentiable*[*simp*]:
(*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) *piecewise-differentiable-on UNIV*
proof −
have ∀ *t*∈*UNIV*−{*t*. ¬ *ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}.
ccdf (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*
by *auto*
thus *?thesis*
unfolding *piecewise-differentiable-on-def*
using *ccdfTx-continuous ccdfTx-nondifferentiable-finite-set* **by** *blast*
qed

lemma *deriv-ccdfTx-measurable*[*measurable*]:
deriv (*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))) ∈ *borel-measurable borel*
proof −
have *set-borel-measurable borel UNIV* (*deriv* (*ccdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))))
by (*rule piecewise-differentiable-on-deriv-measurable-real*; *simp*)
thus *?thesis* **unfolding** *set-borel-measurable-def* **by** *simp*
qed

5.2.5 Properties of Cumulative Distributive Function for $T(x)$

lemma *cdfTx-continuous*[*simp*]:
continuous-on UNIV (*cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)))
using *distrTx-RD.cdf-ccdf ccdfTx-continuous* **by** (*simp add*: *continuous-on-eq-continuous-within*)

corollary *cdfTx-borel-measurable*[*measurable*]:

cdf (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) ∈ *borel-measurable borel*
by (*rule borel-measurable-continuous-onI*) *simp*

lemma *cdfTx-nondifferentiable-finite-set*[*simp*]:
finite {*t*. ¬ *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}
using *distrTx-RD.differentiable-cdf-ccdf* **by** *simp*

lemma *cdfTx-differentiable-open-set*:
open {*t*. *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*}
using *distrTx-RD.differentiable-cdf-ccdf ccdfTx-differentiable-open-set* **by** *simp*

lemma *cdfTx-differentiable-borel-set*[*measurable*, *simp*]:
{*t*. *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*} ∈ *sets borel*
using *cdfTx-differentiable-open-set* **by** *simp*

lemma *cdfTx-differentiable-AE*:
AE t in lborel. (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *differentiable at t*
by (*rewrite distrTx-RD.differentiable-cdf-ccdf*, *simp add*: *ccdfTx-differentiable-AE*)

lemma *cdfTx-piecewise-differentiable*[*simp*]:
(*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *piecewise-differentiable-on UNIV*
using *piecewise-differentiable-diff piecewise-differentiable-const ccdfTx-piecewise-differentiable*
by (*rewrite distrTx-RD.cdf-ccdf*) *blast*

lemma *deriv-cdfTx-measurable*[*measurable*]:
deriv (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) ∈ *borel-measurable borel*
proof −
have *set-borel-measurable borel UNIV* (*deriv* (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))))
by (*rule piecewise-differentiable-on-deriv-measurable-real*; *simp*)
thus *?thesis* **unfolding** *set-borel-measurable-def* **by** *simp*
qed

5.2.6 Properties of Probability Density Function of $T(x)$

lemma *pdfTx-nonneg*: *pdfT x t* ≥ *0* **for** *t*::*real*
proof −
fix *t*
show *pdfT x t* ≥ *0*
proof (*cases* ‹*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*›)
case *True*
have *mono-on UNIV* (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)))
unfolding *mono-on-def* **using** *distrTx-RD.cdf-nondecreasing* **by** *blast*
moreover have (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative pdfT x t*) (*at t*)
unfolding *pdfT-def* **using** *True DERIV-deriv-iff-real-differentiable* **by** *presburger*
ultimately show *?thesis* **by** (*rule mono-on-imp-deriv-nonneg*) *simp*
next

```
    case False
    thus ?thesis unfolding pdfT-def by simp
  qed
qed

lemma pdfTx-neg-0: pdfT x t = 0 if t < 0 for t::real
proof −
  let ?e = −t/2
  have (cdf (distr (𝔐 ⌊ alive x) borel (T x)) has-real-derivative 0) (at t)
    apply (rewrite DERIV-cong-ev[of t t - λ-. 0 0 0], simp-all add: that)
    apply (rewrite eventually-nhds)
    apply (rule exI[of - ball t ?e], auto simp add: that)
    by (rule cdfTx-nonpos-0; simp add: dist-real-def)
  thus ?thesis unfolding pdfT-def by (meson DERIV-imp-deriv)
qed

lemma pdfTx-0-0: pdfT x 0 = 0
proof (cases ‹cdf (distr (𝔐 ⌊ alive x) borel (T x)) differentiable at 0›)
  case True
  let ?cdfx = cdf (distr (𝔐 ⌊ alive x) borel (T x))
  have (?cdfx has-real-derivative 0) (at 0)
  proof −
    from True obtain c where cdfx-deriv: (?cdfx has-real-derivative c) (at 0)
      unfolding differentiable-def using has-real-derivative by blast
    hence (?cdfx has-real-derivative c) (at 0 within {..0})
      by (rule has-field-derivative-at-within)
    moreover have (?cdfx has-real-derivative 0) (at 0 within {..0})
    proof −
      have ∀F t in at 0 within {..0}. ?cdfx t = 0
      proof −
        { fix t assume t ∈ {..0::real} t ≠ 0 dist t 0 < 1
          hence ?cdfx t = 0 using cdfTx-nonpos-0 x-lt-psi by blast }
        hence ∃ d>0::real. ∀ t∈{..0}. t≠0 ∧ dist t 0 < d ⟶ ?cdfx t = 0 by (smt
(verit))
        thus ?thesis by (rewrite eventually-at) simp
      qed
      moreover have ?cdfx 0 = 0
      proof −
        have continuous (at 0 within {..0}) ?cdfx
          using True differentiable-imp-continuous-within differentiable-subset by
blast
        thus ?thesis by (simp add: cdfTx-nonpos-0)
      qed
      ultimately show ?thesis
        by (rewrite has-field-derivative-cong-eventually[of - λ-. 0::real 0 {..0} 0];
simp)
    qed
    ultimately have c = 0
      using has-real-derivative-iff-has-vector-derivative
```

```
    apply (intro vector-derivative-unique-within[of 0 {..0} ?cdfx c 0]; blast?)
    by (rewrite at-within-eq-bot-iff)
      (metis closure-lessThan islimpt-in-closure limpt-of-closure
        trivial-limit-at-left-real trivial-limit-within)
   thus (?cdfx has-real-derivative 0) (at 0) using cdfx-deriv by simp
 qed
 thus ?thesis unfolding pdfT-def by (meson DERIV-imp-deriv)
next
 case False
 thus ?thesis unfolding pdfT-def by simp
qed

lemma pdfTx-nonpos-0: pdfT x t = 0 if t ≤ 0 for t::real
 using pdfTx-neg-0 pdfTx-0-0 that by force

lemma pdfTx-beyond-0: pdfT x t = 0 if x+t ≥ $ψ for t::real
proof (cases ‹t ≤ 0›)
 case True
 thus ?thesis using pdfTx-nonpos-0 by simp
next
 let ?cdfTx = cdf (distr (𝔐 ⌊ alive x) borel (T x))
 case False
 hence t-pos: t > 0 by simp
 thus ?thesis
 proof −
  have (?cdfTx has-real-derivative 0) (at-right t)
   apply (rewrite has-field-derivative-cong-eventually[where g=λ-. 1], simp-all)
    apply (rewrite eventually-at-right-field)
   using that cdfTx-1-equiv
   by (intro exI[of -t+1], auto simp add: le-ereal-less less-eq-ereal-def)
  thus ?thesis unfolding pdfT-def
  by (meson has-real-derivative-iff-has-vector-derivative has-vector-derivative-at-within
      DERIV-deriv-iff-real-differentiable trivial-limit-at-right-real
      vector-derivative-unique-within)
 qed
qed

lemma pdfTx-pdfX: pdfT x t = pdfX (x+t) / 𝒫(ξ in 𝔐. X ξ > x) if t > 0 for
t::real
proof (cases ‹cdf (distr 𝔐 borel X) differentiable at (x+t)›)
 case True
 let ?cdfX = cdf (distr 𝔐 borel X)
 let ?cdfTx = cdf (distr (𝔐 ⌊ alive x) borel (T x))
 have [simp]: ?cdfTx differentiable at t using differentiable-cdfX-cdfTx that True
by simp
 have pdfT x t = deriv ?cdfTx t unfolding pdfT-def using that differentiable-cdfX-cdfTx
by simp
 hence (?cdfTx has-field-derivative (pdfT x t)) (at t)
  using True DERIV-deriv-iff-real-differentiable by simp
```

moreover have $\bigwedge u.$ *dist u t* $< t \Longrightarrow$
?cdfX $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - (1$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1) =$ *?cdfTx u*
proof −
fix *u*::*real*
assume *dist u t* $< t$
hence [*simp*]: $u > 0$ **using** *dist-real-def* **by** *fastforce*
have *?cdfX* $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - (1$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1) =$
$(1 - \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x+u))$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - (1$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1)$
using *MM-PS.ccdf-distr-P X-RV distrX-RD.cdf-ccdf distrX-RD.prob-space* **by** *presburger*
also have $\ldots = 1 - \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x)$
by (*simp add*: *diff-divide-distrib*)
also have $\ldots =$ *?cdfTx u*
apply (*rewrite ccdfTx-PX*[*THEN sym*], *simp-all add*: *less-eq-real-def*)
using *distrTx-RD.cdf-ccdf distrTx-RD.prob-space* **by** *presburger*
finally show *?cdfX* $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - (1$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1) =$ *?cdfTx u* **.**
qed
ultimately have $((\lambda u.$ *?cdfX* $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - (1$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1))$
has-field-derivative $(pdfT\ x\ t))$ $(at\ t)$
apply −
by (*rule has-field-derivative-transform-within*[**where** *d*=*t*]; *simp add*: *that*)
hence $((\lambda u.$ *?cdfX* $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x))$ *has-field-derivative* $(pdfT\ x\ t))$ $(at\ t)$
unfolding *has-field-derivative-def*
using *has-derivative-add-const*[**where** *c*=1 / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) - 1$] **by** *force*
hence $((\lambda u.$ *?cdfX* $(x+u)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) * \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x))$ *has-field-derivative*
pdfT x t $* \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x))$ $(at\ t)$
using *DERIV-cmult-right*[**where** *c*=$\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x)$] **by** *force*
hence $((\lambda u.$ *?cdfX* $(x+u))$ *has-field-derivative pdfT x t* $* \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x))$ $(at\ t)$
unfolding *has-field-derivative-def* **using** *has-derivative-transform PXx-pos x-lt-psi*
by (*smt* (*verit*) *Collect-cong UNIV-I nonzero-mult-div-cancel-right times-divide-eq-left*)
hence (*?cdfX has-field-derivative pdfT x t* $* \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x))$ $(at\ (x+t))$
using *DERIV-at-within-shift* **by** *force*
moreover have (*?cdfX has-field-derivative deriv ?cdfX* $(x+t))$ $(at\ (x+t))$
using *True DERIV-deriv-iff-real-differentiable* **by** *blast*
ultimately have *pdfT x t* $* \mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x) =$ *deriv ?cdfX* $(x+t)$
by (*simp add*: *DERIV-imp-deriv*)
thus *pdfT x t* = *pdfX* $(x+t)$ / $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $X\ \xi > x)$
unfolding *pdfX-def* **using** *True*
by *simp* (*metis PXx-pos nonzero-mult-div-cancel-right x-lt-psi zero-less-measure-iff*)
next
case *False*

hence [*simp*]: ¬ *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*)) *differentiable at t*
using *differentiable-cdfX-cdfTx that* **by** *simp*
thus *pdfT x t* = *pdfX* (*x*+*t*) / 𝒫(ξ *in* 𝔐. *X* ξ > *x*) **unfolding** *pdfT-def pdfX-def*
using *False* **by** *simp*
qed

lemma *pdfTx-measurable*[*measurable*]: *pdfT x* ∈ *borel-measurable borel*
proof −
let *?cdfTx* = *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))
have *countable* {*x*. ¬ *?cdfTx differentiable at x*}
using *cdfX-nondifferentiable-finite-set uncountable-infinite* **by** *force*
thus *?thesis*
unfolding *pdfT-def*
apply −
by (*rule measurable-discrete-difference*
[**where** *X*={*x*. ¬ *?cdfTx differentiable at x*} **and** *f*=*deriv ?cdfTx*]; *simp*)
qed

lemma *distributed-pdfTx*: *distributed* (𝔐 ⌊ *alive x*) *lborel* (*T x*) (*pdfT x*)
proof −
let *?cdfTx* = *cdf* (*distr* (𝔐 ⌊ *alive x*) *borel* (*T x*))
obtain *S* **where** *fin*: *finite S* **and** *diff*: ⋀*t*. *t* ∉ *S* ⟹ *?cdfTx differentiable at t*
using *cdfTx-piecewise-differentiable* **unfolding** *piecewise-differentiable-on-def*
by *blast*
{ fix *t*::*real* **assume** *t-notin*: *t* ∉ *S*
have *?cdfTx differentiable at t* **using** *diff t-notin* **by** *simp*
hence (*?cdfTx has-real-derivative pdfT x t*) (*at t*)
unfolding *pdfT-def* **using** *DERIV-deriv-iff-real-differentiable* **by** *force* **}**
thus *?thesis*
by (*intro alivex-PS.distributed-deriv-cdf*[**where** *S*=*S*]; *simp add*: *fin*)
qed

lemma *nn-integral-pdfTx-1*: (∫⁺*s*. *pdfT x s* ∂*lborel*) = *1*
proof −
have (∫⁺*s*. *pdfT x s* ∂*lborel*) = *emeasure* (*density lborel* (*pdfT x*)) *UNIV*
by (*rewrite emeasure-density*) *simp-all*
also have ... = *emeasure* (*distr* (𝔐 ⌊ *alive x*) *lborel* (*T x*)) *UNIV*
using *distributed-pdfTx* **unfolding** *distributed-def* **by** *simp*
also have ... = *1*
by (*metis distrTx-RD.emeasure-space-1 distrTx-RD.space-eq-univ distr-cong sets-lborel*)
finally show *?thesis* **.**
qed

corollary *has-bochner-integral-pdfTx-1*: *has-bochner-integral lborel* (*pdfT x*) *1*
by (*rule has-bochner-integral-nn-integral*; *simp add*: *pdfTx-nonneg nn-integral-pdfTx-1*)

corollary *LBINT-pdfTx-1*: (*LBINT s*. *pdfT x s*) = *1*
using *has-bochner-integral-pdfTx-1* **by** (*simp add*: *has-bochner-integral-integral-eq*)

corollary *pdfTx-has-integral-1*: (*pdfT x has-integral 1*) *UNIV*
by (*rule nn-integral-has-integral*; *simp add*: *pdfTx-nonneg nn-integral-pdfTx-1*)

lemma *set-nn-integral-pdfTx-1*: ($\int^+ s{\in}\{0..\}$. *pdfT x s* ∂*lborel*) = *1*
apply (*rewrite nn-integral-pdfTx-1*[*THEN sym*])
apply (*rule nn-integral-cong*)
using *pdfTx-nonpos-0*
by (*metis atLeast-iff ennreal-0 indicator-simps*(*1*) *linorder-le-cases*
mult.commute mult-1 mult-zero-left)

corollary *has-bochner-integral-pdfTx-1-nonpos*:
has-bochner-integral lborel (λ*s*. *pdfT x s* ∗ *indicator* {*0*..} *s*) *1*
apply (*rule has-bochner-integral-nn-integral*, *simp-all*)
using *pdfTx-nonneg* **apply** *simp*
using *set-nn-integral-pdfTx-1* **by** (*simp add*: *nn-integral-set-ennreal*)

corollary *set-LBINT-pdfTx-1*: (*LBINT s*:{*0*..}. *pdfT x s*) = *1*
unfolding *set-lebesgue-integral-def*
using *has-bochner-integral-pdfTx-1-nonpos has-bochner-integral-integral-eq*
apply (*simp*, *rewrite mult.commute*)
by (*smt* (*verit*) *Bochner-Integration.integral-cong has-bochner-integral-integral-eq*)

corollary *pdfTx-has-integral-1-nonpos*: (*pdfT x has-integral 1*) {*0*..}
proof −
have *set-integrable lebesgue* {*0*..} (*pdfT x*)
unfolding *set-integrable-def*
apply (*rewrite integrable-completion*, *simp-all*)
using *has-bochner-integral-pdfTx-1-nonpos* **by** (*rewrite mult.commute*, *rule integrable.intros*)
moreover have (*LINT s*:{*0*..}|*lebesgue*. *pdfT x s*) = *1*
using *set-LBINT-pdfTx-1* **unfolding** *set-lebesgue-integral-def*
by (*rewrite integral-completion*; *simp*)
ultimately show *?thesis* **using** *has-integral-set-lebesgue* **by** *force*
qed

lemma *set-nn-integral-pdfTx-PTx*: ($\int^+ s{\in}A$. *pdfT x s* ∂*lborel*) = $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *T x* ξ ∈ *A* | *T x* ξ > *0*)
if *A* ∈ *sets lborel* **for** *A* :: *real set*
proof −
have [*simp*]: *A* ∈ *sets borel* **using** *that* **by** *simp*
have ($\int^+ s{\in}A$. *pdfT x s* ∂*lborel*) = *emeasure* (*density lborel* (*pdfT x*)) *A*
using *that* **by** (*rewrite emeasure-density*; *force*)
also have ... = *emeasure* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *lborel* (*T x*)) *A*
using *distributed-pdfTx* **unfolding** *distributed-def* **by** *simp*
also have ... = *ennreal* $\mathcal{P}$(ξ *in* ($\mathfrak{M}$ ⌊ *alive x*). *T x* ξ ∈ *A*)
apply (*rewrite emeasure-distr*, *simp-all*)
apply (*rewrite finite-measure.emeasure-eq-measure*, *simp*)
by (*smt* (*verit*) *Collect-cong vimage-eq Int-def*)

also have ... = *ennreal* $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $T\ x\ \xi \in A \mid T\ x\ \xi > 0)$
unfolding *alive-def*
apply (*rewrite MM-PS.cond-prob-space-cond-prob*[*THEN sym*], *simp-all add*: *pred-def futr-life-def*)
using *borel-measurable-diff X-RV that* **by** *measurable*
finally show *?thesis* **.**
qed

lemma *pdfTx-set-integrable*: *set-integrable lborel A* (*pdfT x*) **if** $A \in$ *sets lborel*
unfolding *set-integrable-def*
using *that pdfTx-nonneg* **apply** (*intro integrableI-nonneg*, *simp-all*)
apply (*rewrite mult.commute*)
using *set-nn-integral-pdfTx-PTx that*
by (*metis* (*no-types*, *lifting*) *ennreal-indicator ennreal-less-top ennreal-mult' nn-integral-cong*)

lemma *set-integral-pdfTx-PTx*: (*LBINT s:A. pdfT x s*) = $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $T\ x\ \xi \in A \mid T\ x\ \xi > 0)$
if $A \in$ *sets lborel* **for** A :: *real set*
unfolding *set-lebesgue-integral-def*
apply (*rewrite integral-eq-nn-integral*)
using *that* **apply** (*simp-all add*: *pdfTx-nonneg*)
apply (*rewrite indicator-mult-ennreal*[*THEN sym*], *rewrite mult.commute*)
by (*rewrite set-nn-integral-pdfTx-PTx*; *simp*)

lemma *pdfTx-has-integral-PTx*: (*pdfT x has-integral* $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $T\ x\ \xi \in A \mid T\ x\ \xi > 0))\ A$
if $A \in$ *sets lborel* **for** A :: *real set*
proof −
have $((\lambda s.$ *pdfT x s* $*$ *indicat-real A s*$)$ *has-integral* $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $T\ x\ \xi \in A \mid T\ x\ \xi > 0))$ *UNIV*
using *that pdfTx-nonneg* **apply** (*intro nn-integral-has-integral*, *simp-all*)
using *set-nn-integral-pdfTx-PTx that* **by** (*simp add*: *nn-integral-set-ennreal*)
thus *?thesis*
by (*smt* (*verit*) *has-integral-cong has-integral-restrict-UNIV indicator-eq-0-iff indicator-simps*(*1*) *mult-cancel-left2 mult-eq-0-iff*)
qed

corollary *pdfTx-has-integral-PTx-Icc*:
(*pdfT x has-integral* $\mathcal{P}(\xi$ *in* $\mathfrak{M}.$ $a \le T\ x\ \xi \wedge T\ x\ \xi \le b \mid T\ x\ \xi > 0))$ $\{a..b\}$ **for** $a\ b$ :: *real*
using *pdfTx-has-integral-PTx*[*of* $\{a..b\}$] **by** *simp*

corollary *pdfTx-integrable-on-Icc*: *pdfT x integrable-on* $\{a..b\}$ **for** $a\ b$ :: *real*
using *pdfTx-has-integral-PTx-Icc* **by** *auto*

end

5.2.7 Properties of Probability Density Function of X

lemma *pdfX-nonneg*: *pdfX x* $\geq$ *0* **for** *x*::*real*
using *pdfTx-nonneg pdfT0-X psi-pos′* **by** (*smt* (*verit*))

lemma *pdfX-nonpos-0*: *pdfX x* = *0* **if** *x* $\leq$ *0* **for** *x*::*real*
using *pdfTx-nonpos-0 pdfT0-X psi-pos′ that* **by** (*smt* (*verit*))

lemma *pdfX-beyond-0*: *pdfX x* = *0* **if** *x* $\geq$ \$$\psi$ **for** *x*::*real*
using *pdfTx-beyond-0 pdfT0-X psi-pos′ that* **by** (*smt* (*verit*))

lemma *nn-integral-pdfX-1*: *integral*^{N} *lborel pdfX* = *1*
using *nn-integral-pdfTx-1 pdfT0-X psi-pos′* **by** *metis*

corollary *has-bochner-integral-pdfX-1*: *has-bochner-integral lborel pdfX 1*
by (*rule has-bochner-integral-nn-integral*; *simp add*: *pdfX-nonneg nn-integral-pdfX-1*)

corollary *LBINT-pdfX-1*: (*LBINT s. pdfX s*) = *1*
using *has-bochner-integral-pdfX-1* **by** (*simp add*: *has-bochner-integral-integral-eq*)

corollary *pdfX-has-integral-1*: (*pdfX has-integral 1*) *UNIV*
by (*rule nn-integral-has-integral*; *simp add*: *pdfX-nonneg nn-integral-pdfX-1*)

lemma *set-nn-integral-pdfX-PX*: *set-nn-integral lborel A pdfX* = $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *X ξ* $\in$ *A*)
if *A* $\in$ *sets lborel* **for** *A* :: *real set*
using *PT0-eq-PX-lborel that*
by (*rewrite pdfT0-X*[*THEN sym*], *rewrite set-nn-integral-pdfTx-PTx*; *simp*)

lemma *pdfX-set-integrable*: *set-integrable lborel A pdfX* **if** *A* $\in$ *sets lborel* **for** *A* :: *real set*
using *pdfTx-set-integrable pdfT0-X psi-pos′ that* **by** (*smt* (*verit*))

lemma *set-integral-pdfX-PX*: (*LBINT s*:*A. pdfX s*) = $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *X ξ* $\in$ *A*)
if *A* $\in$ *sets lborel* **for** *A* :: *real set*
using *PT0-eq-PX-lborel that* **by** (*rewrite pdfT0-X*[*THEN sym*], *rewrite set-integral-pdfTx-PTx*; *simp*)

lemma *pdfX-has-integral-PX*: (*pdfX has-integral* $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *X ξ* $\in$ *A*)) *A*
if *A* $\in$ *sets lborel* **for** *A* :: *real set*
using *that* **apply** (*rewrite pdfT0-X*[*THEN sym*], *rewrite PT0-eq-PX-lborel*[*THEN sym*], *simp*)
by (*rule pdfTx-has-integral-PTx*; *simp*)

corollary *pdfX-has-integral-PX-Icc*: (*pdfX has-integral* $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *a* $\leq$ *X ξ* $\wedge$ *X ξ* $\leq$ *b*)) {*a*..*b*}
for *a b* :: *real*
using *pdfX-has-integral-PX*[*of* {*a*..*b*}] **by** *simp*

corollary *pdfX-integrable-on-Icc*: *pdfX integrable-on* {*a*..*b*} **for** *a b* :: *real*

using *pdfX-has-integral-PX-Icc* **by** *auto*

5.2.8 Relations between Life Expectation and Probability Density Function

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: $x < \$\psi$
begin

interpretation *alivex-PS*: *prob-space* $\mathfrak{M} \restriction$ *alive x*
by (*rule MM-PS.cond-prob-space-correct*; *simp add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*) **by** *simp*

proposition *nn-integral-T-pdfT*:
($\int^+\xi$. *ennreal* (*g* (*T x* ξ)) ∂($\mathfrak{M} \restriction$ *alive x*)) = ($\int^+ s \in \{0..\}$. *ennreal* (*pdfT x s* $*$ *g s*) ∂*lborel*)
if *g* $\in$ *borel-measurable lborel* **for** *g* :: *real* $\Rightarrow$ *real*
proof −
have ($\int^+\xi$. *ennreal* (*g* (*T x* ξ)) ∂($\mathfrak{M} \restriction$ *alive x*)) = $\int^+ s$. *ennreal* (*pdfT x s*) $*$ *ennreal* (*g s*) ∂*lborel*
proof −
have *distributed* ($\mathfrak{M} \restriction$ *alive x*) *lborel* (*T x*) (λs. *ennreal* (*pdfT x s*))
by (*intro distributed-pdfTx*) *simp*
moreover have (λs. *ennreal* (*g s*)) $\in$ *borel-measurable borel* **using** *that* **by** *measurable*
ultimately show *?thesis* **by** (*rewrite distributed-nn-integral*; *simp*)
qed
also have ... = $\int^+ s$. *ennreal* (*pdfT x s* $*$ *g s*) ∂*lborel* **using** *ennreal-mult' pdfTx-nonneg* **by** *force*
also have ... = ($\int^+ s \in \{0..\}$. *ennreal* (*pdfT x s* $*$ *g s*) ∂*lborel*)
apply (*rule nn-integral-cong*, *simp*)
by (*metis atLeast-iff ennreal-0 indicator-simps linorder-not-le mult-1 mult-commute-abs mult-zero-left pdfTx-neg-0 x-lt-psi*)
finally show *?thesis* **.**
qed

lemma *expectation-LBINT-pdfT-nonneg*:
alivex-PS.expectation ($\lambda\xi$. *g* (*T x* ξ)) = (*LBINT s*:{0..}. *pdfT x s* $*$ *g s*)
if $\bigwedge s$. *s* $\geq$ *0* $\Longrightarrow$ *g s* $\geq$ *0* *g* $\in$ *borel-measurable lborel* **for** *g* :: *real* $\Rightarrow$ *real*
— Note that *0* = *0* holds when the integral diverges.
using *that* **apply** (*rewrite integral-eq-nn-integral*, *simp*)
apply (*rule AE-I2*, *metis alivex-Tx-pos less-imp-le*)
unfolding *set-lebesgue-integral-def* **apply** (*rewrite integral-eq-nn-integral*, *simp-all*)
apply (*rule AE-I2*,
metis indicator-pos-le pdfTx-nonpos-0 x-lt-psi zero-le-mult-iff zero-le-square pdfTx-nonneg)

by (*rewrite nn-integral-T-pdfT*) (*simp-all add*: *indicator-mult-ennreal mult.commute*)

corollary *expectation-integral-pdfT-nonneg*:
alivex-PS.expectation (λξ. *g* (*T x* ξ)) = *integral* {*0*..} (λ*s*. *pdfT x s* ∗ *g s*)
if ⋀*s*. *s* ≥ *0* ⟹ *g s* ≥ *0 g* ∈ *borel-measurable lborel* **for** *g* :: *real* ⇒ *real*
— Note that *0* = *0* holds when the integral diverges.
proof −
have *alivex-PS.expectation* (λξ. *g* (*T x* ξ)) = (*LBINT s*:{*0*..}. *pdfT x s* ∗ *g s*)
using *expectation-LBINT-pdfT-nonneg that* **by** *simp*
also have ... = *integral* {*0*..} (λ*s*. *pdfT x s* ∗ *g s*)
using *that pdfTx-nonneg* **by** (*intro set-borel-integral-eq-integral-nonneg*; *simp*)
finally show *?thesis* **.**
qed

proposition *expectation-LBINT-pdfT*:
alivex-PS.expectation (λξ. *g* (*T x* ξ)) = (*LBINT s*:{*0*..}. *pdfT x s* ∗ *g s*)
if *set-integrable lborel* {*0*..} (λ*s*. *pdfT x s* ∗ *g s*) *g* ∈ *borel-measurable lborel*
for *g* :: *real* ⇒ *real*
proof −
have *pg-fin*: (∫ +ξ. *ennreal* (*g* (*T x* ξ)) ∂(𝔐 ⌊ *alive x*)) ≠ ∞
using *that* **apply** (*rewrite nn-integral-T-pdfT*, *simp*)
using *that* **unfolding** *set-integrable-def* **apply** (*rewrite* **in** *asm real-integrable-def*, *simp*)
by (*simp add*: *indicator-mult-ennreal mult.commute*)
moreover have *mg-fin*: (∫ +ξ. *ennreal* (− *g* (*T x* ξ)) ∂(𝔐 ⌊ *alive x*)) ≠ ∞
using *that* **apply** (*rewrite nn-integral-T-pdfT*[*of* λ*s*. − *g s*], *simp*)
using *that* **unfolding** *set-integrable-def* **apply** (*rewrite* **in** *asm real-integrable-def*, *simp*)
by (*simp add*: *indicator-mult-ennreal mult.commute*)
ultimately have [*simp*]: *integrable* (𝔐 ⌊ *alive x*) (λξ. *g* (*T x* ξ))
using *that* **by** (*rewrite real-integrable-def*; *simp*)
have (∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ *max 0* (*g s*)) ∂*lborel*) =
(∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ *g s*) ∂*lborel*)
using *SPMF.ennreal-max-0 ennreal-mult′ pdfTx-nonneg x-lt-psi* **by** *presburger*
also have ... < ∞
using *pg-fin nn-integral-T-pdfT that top.not-eq-extremum* **by** *auto*
finally have (∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ *max 0* (*g s*)) ∂*lborel*) < ∞ **.**
hence [*simp*]: *set-integrable lborel* {*0*..} (λ*s*. *pdfT x s* ∗ *max 0* (*g s*))
unfolding *set-integrable-def* **using** *that* **apply** (*intro integrableI-nonneg*, *simp-all*)
using *pdfTx-nonneg* **apply** (*intro AE-I2*, *force*)
by (*metis* (*no-types*, *lifting*) *indicator-mult-ennreal mult.commute nn-integral-cong*)
have (∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ *max 0* (− *g s*)) ∂*lborel*) =
(∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ − *g s*) ∂*lborel*)
using *SPMF.ennreal-max-0 ennreal-mult′ pdfTx-nonneg x-lt-psi* **by** *presburger*
also have ... < ∞
using *mg-fin nn-integral-T-pdfT*[*of* λ*s*. − *g s*] *that top.not-eq-extremum* **by** *force*
finally have (∫ +*s*∈{*0*..}. *ennreal* (*pdfT x s* ∗ *max 0* (− *g s*)) ∂*lborel*) < ∞ **.**
hence [*simp*]: *set-integrable lborel* {*0*..} (λ*s*. *pdfT x s* ∗ *max 0* (− *g s*))

unfolding *set-integrable-def* **using** *that* **apply** (*intro integrableI-nonneg*, *simp-all*)
using *pdfTx-nonneg* **apply** (*intro AE-I2*, *force*)
by (*metis* (*no-types*, *lifting*) *indicator-mult-ennreal mult.commute nn-integral-cong*)
have *alivex-PS.expectation* (λξ. *g* (*T x* ξ)) =
alivex-PS.expectation (λξ. *max 0* (*g* (*T x* ξ))) − *alivex-PS.expectation* (λξ. *max 0* (− *g* (*T x* ξ)))
by (*rewrite Bochner-Integration.integral-cong*
[**where** *N*=𝔐 ⌊ *alive x* **and** *g*=λξ. *max 0* (*g* (*T x* ξ)) − *max 0* (− *g* (*T x* ξ))]; *simp*)
moreover have *alivex-PS.expectation* (λξ. *max 0* (*g* (*T x* ξ))) =
(*LBINT s*:{*0*..}. *pdfT x s* ∗ *max 0* (*g s*))
using *that* **by** (*rewrite expectation-LBINT-pdfT-nonneg*[**where** *g*=λs. *max 0* (*g s*)]) *simp-all*
moreover have *alivex-PS.expectation* (λξ. *max 0* (− *g* (*T x* ξ))) =
(*LBINT s*:{*0*..}. *pdfT x s* ∗ *max 0* (− *g s*))
using *that* **by** (*rewrite expectation-LBINT-pdfT-nonneg*[**where** *g*=λs. *max 0* (− *g s*)]) *simp-all*
ultimately have *alivex-PS.expectation* (λξ. *g* (*T x* ξ)) =
(*LBINT s*:{*0*..}. *pdfT x s* ∗ *max 0* (*g s*)) − (*LBINT s*:{*0*..}. *pdfT x s* ∗ *max 0* (− *g s*)) **by** *simp*
also have ... = (*LBINT s*:{*0*..}. (*pdfT x s* ∗ *max 0* (*g s*) − *pdfT x s* ∗ *max 0* (− *g s*)))
by (*rewrite set-integral-diff*; *simp*)
also have ... = (*LBINT s*:{*0*..}. *pdfT x s* ∗ (*max 0* (*g s*) − *max 0* (− *g s*)))
by (*simp add*: *right-diff-distrib*)
also have ... = (*LBINT s*:{*0*..}. *pdfT x s* ∗ *g s*)
using *minus-max-eq-min*
by (*metis* (*no-types*, *opaque-lifting*) *diff-zero max-def min-def minus-diff-eq*)
finally show *?thesis* **.**
qed

corollary *expectation-integral-pdfT*:
alivex-PS.expectation (λξ. *g* (*T x* ξ)) = *integral* {*0*..} (λs. *pdfT x s* ∗ *g s*)
if (λs. *pdfT x s* ∗ *g s*) *absolutely-integrable-on* {*0*..} *g* ∈ *borel-measurable lborel*
for *g* :: *real* ⇒ *real*
proof −
have [*simp*]: *set-integrable lborel* {*0*..} (λs. *pdfT x s* ∗ *g s*)
using *that* **by** (*rewrite absolutely-integrable-on-iff-set-integrable*; *simp*)
show *?thesis*
apply (*rewrite set-borel-integral-eq-integral*(*2*)[*THEN sym*], *simp*)
using *that* **by** (*rewrite expectation-LBINT-pdfT*; *simp*)
qed

corollary *e-LBINT-pdfT*: $*e*‘∘-*x* = (*LBINT s*:{*0*..}. *pdfT x s* ∗ *s*)
— Note that *0* = *0* holds when the life expectation diverges.
unfolding *life-expect-def* **using** *expectation-LBINT-pdfT-nonneg* **by** *force*

corollary *e-integral-pdfT*: $*e*‘∘-*x* = *integral* {*0*..} (λs. *pdfT x s* ∗ *s*)
— Note that *0* = *0* holds when the life expectation diverges.

unfolding *life-expect-def* **using** *expectation-integral-pdfT-nonneg* **by** *force*

end

corollary *e-LBINT-pdfX*: $\$e`\circ$-*0* = (*LBINT x*:{*0*..}. *pdfX x* ∗ *x*)
— Note that *0* = *0* holds when the life expectation diverges.
using *e-LBINT-pdfT pdfT0-X psi-pos′* **by** *presburger*

corollary *e-integral-pdfX*: $\$e`\circ$-*0* = *integral* {*0*..} (λ*x*. *pdfX x* ∗ *x*)
— Note that *0* = *0* holds when the life expectation diverges.
using *e-integral-pdfT pdfT0-X psi-pos′* **by** *presburger*

5.2.9 Introduction of Force of Mortality

definition *force-mortal* :: *real* ⇒ *real* (‹$\mu′$--› [*101*] *200*)
where $\$\mu$-*x* ≡ *MM-PS.hazard-rate X x*

lemma *mu-pdfX*: $\$\mu$-*x* = *pdfX x* / *ccdf* (*distr* 𝔑 *borel X*) *x*
if (*cdf* (*distr* 𝔑 *borel X*)) *differentiable at x* **for** *x*::*real*
unfolding *force-mortal-def pdfX-def*
by (*rewrite MM-PS.hazard-rate-deriv-cdf*, *simp-all add*: *that*)

lemma *mu-unborn-0*: $\$\mu$-*x* = *0* **if** *x* < *0* **for** *x*::*real*
apply (*rewrite mu-pdfX*)
using *cdfX-has-real-derivative-0-unborn real-differentiable-def that* **apply** *blast*
using *pdfX-nonpos-0 that* **by** *auto*

lemma *mu-beyond-0*: $\$\mu$-*x* = *0* **if** *x* ≥ $\$\psi$ **for** *x*::*real*
— Note that division by *0* is defined as *0* in Isabelle/HOL.
unfolding *force-mortal-def* **using** *MM-PS.hazard-rate-0-ccdf-0 ccdfX-0-equiv that*
by *simp*

lemma *mu-nonneg-differentiable*: $\$\mu$-*x* ≥ *0*
if (*cdf* (*distr* 𝔑 *borel X*)) *differentiable at x* **for** *x*::*real*
apply (*rewrite mu-pdfX*, *simp add*: *that*)
using *pdfX-nonneg distrX-RD.ccdf-nonneg* **by** *simp*

lemma *mu-nonneg-AE*: *AE x in lborel*. $\$\mu$-*x* ≥ *0*
using *cdfX-differentiable-AE mu-nonneg-differentiable* **by** *auto*

lemma *mu-measurable*[*measurable*]: (λ*x*. $\$\mu$-*x*) ∈ *borel-measurable borel*
proof −
obtain *S* **where**
finite S **and** ⋀*x*. *x* ∉ *S* ⟹ (*cdf* (*distr* 𝔑 *borel X*)) *differentiable at x*
using *cdfX-piecewise-differentiable* **unfolding** *piecewise-differentiable-on-def*
by *blast*
thus *?thesis*
apply (*rewrite measurable-discrete-difference*
[**where** *f*=λ*x*. *pdfX x* / *ccdf* (*distr* 𝔑 *borel X*) *x* **and** *X*=*S*], *simp-all*)

by (*simp-all add*: *MM-PS.ccdf-distr-measurable borel-measurable-divide countable-finite mu-pdfX*)
qed

lemma *mu-deriv-ccdf*: $\$\mu$-*x* = − *deriv* (*ccdf* (*distr* 𝔐 *borel X*)) *x* / *ccdf* (*distr* 𝔐 *borel X*) *x*
if (*ccdf* (*distr* 𝔐 *borel X*)) *differentiable at x x* < $\$\psi$ **for** *x*::*real*
unfolding *force-mortal-def*
by (*rewrite MM-PS.hazard-rate-deriv-ccdf*; *simp add*: *that*)

lemma *mu-deriv-ln*: $\$\mu$-*x* = − *deriv* (λ*x*. *ln* (*ccdf* (*distr* 𝔐 *borel X*) *x*)) *x*
if (*ccdf* (*distr* 𝔐 *borel X*)) *differentiable at x x* < $\$\psi$ **for** *x*::*real*
unfolding *force-mortal-def*
apply (*rewrite MM-PS.hazard-rate-deriv-ln-ccdf*, *simp-all add*: *that*)
using *ccdfX-0-equiv that* **by** *force*

lemma *p-exp-integral-mu*: $\$p$-{*t*&*x*} = *exp* (− *integral* {*x*..*x*+*t*} (λ*y*. $\$\mu$-*y*))
if *x* ≥ *0 t* ≥ *0 x*+*t* < $\$\psi$ **for** *x t* :: *real*
proof −
have [*simp*]: *x* < $\$\psi$ **using** *that* **by** (*simp add*: *ereal-less-le*)
have $\$p$-{*t*&*x*} = (*ccdf* (*distr* 𝔐 *borel X*) (*x*+*t*)) / (*ccdf* (*distr* 𝔐 *borel X*) *x*)
apply (*rewrite p-PX*, *simp-all add*: *that*)
by (*rewrite MM-PS.ccdf-distr-P*, *simp*)+ *simp*
also have ... = *exp* (− *integral* {*x*..*x*+*t*} (*MM-PS.hazard-rate X*))
apply (*rule MM-PS.ccdf-exp-cumulative-hazard*, *simp-all add*: *that*)
using *ccdfX-piecewise-differentiable piecewise-differentiable-on-subset* **apply** *blast*
using *ccdfX-continuous continuous-on-subset* **apply** *blast*
using *ccdfX-0-equiv that ereal-less-le linorder-not-le* **by** *force*
also have ... = *exp* (− *integral* {*x*..*x*+*t*} (λ*y*. $\$\mu$-*y*)) **unfolding** *force-mortal-def* **by** *simp*
finally show *?thesis* **.**
qed

corollary *ccdfX-exp-integral-mu*: *ccdf* (*distr* 𝔐 *borel X*) *x* = *exp* (− *integral* {*0*..*x*} (λ*y*. $\$\mu$-*y*))
if *0* ≤ *x* ∧ *x* < $\$\psi$ **for** *x*::*real*
by (*rewrite p-exp-integral-mu*[**where** *t*=*x* **and** *x*=*0*, *simplified*, *THEN sym*]; *simp add*: *that ccdfX-p*)

5.2.10 Properties of Force of Mortality

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: *x* < $\$\psi$
begin

interpretation *alivex-PS*: *prob-space* 𝔐 ⌊ *alive x*
by (*rule MM-PS.cond-prob-space-correct*; *simp add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*) **by** *simp*

lemma *hazard-rate-Tx-mu*: *alivex-PS.hazard-rate* (*T x*) *t* = μ-(*x*+*t*)
if *t* ≥ *0* *x*+*t* < ψ **for** *t*::*real*
proof −
have [*simp*]: $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *X ξ* > *x*) > *0* **by** *force*
moreover have [*simp*]: $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *X ξ* > *x* + *t*) > *0* **using** *that* **by** *force*
moreover have [*simp*]: {*ξ* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X ξ* > *x* + *t*} = {*ξ* ∈ *space* $\mathfrak{M}$. *X ξ* > *x* + *t*}
unfolding *alive-def* **using** *that* **by** (*rewrite MM-PS.space-cond-prob-space*, *simp-all*, *force*)
hence [*simp*]: {*ξ* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X ξ* > *x* + *t*} ∈ *MM-PS.events* **by** *simp*
ultimately have ⋆[*simp*]: $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ *alive x*). *X ξ* > *x* + *t*) > *0*
unfolding *alive-def*
apply (*rewrite MM-PS.cond-prob-space-cond-prob*[*THEN sym*], (*simp-all add*: *pred-def*)[*3*])
unfolding *cond-prob-def* **by** (*smt* (*verit*) *Collect-cong divide-pos-pos*)
have *alivex-PS.hazard-rate* (*T x*) *t* =
Lim (*at-right 0*) (*λdt*. $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ *alive x*). *t* < *T x ξ* ∧ *T x ξ* ≤ *t* + *dt* | *T x ξ* > *t*) / *dt*)
unfolding *alivex-PS.hazard-rate-def* **by** *simp*
also have ... = *Lim* (*at-right 0*)
(*λdt*. $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ *alive x*). *x* + *t* < *X ξ* ∧ *X ξ* ≤ *x* + *t* + *dt* | *X ξ* > *x* + *t*) / *dt*)
apply (*rule Lim-cong*, *rule eventually-at-rightI*[*of 0 1*], *simp-all*)
unfolding *futr-life-def cond-prob-def* **using** *Collect-cong* **by** (*smt* (*verit*))
also have ... =
Lim (*at-right 0*) (*λdt*. $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *x* + *t* < *X ξ* ∧ *X ξ* ≤ *x* + *t* + *dt* | *X ξ* > *x* + *t*) / *dt*)
proof −
have ⋀*dt*. $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ *alive x*). *x* + *t* < *X ξ* ∧ *X ξ* ≤ *x* + *t* + *dt* | *X ξ* > *x* + *t*) =
$\mathcal{P}$(*ξ in* $\mathfrak{M}$. *x* + *t* < *X ξ* ∧ *X ξ* ≤ *x* + *t* + *dt* | *X ξ* > *x* + *t*)
proof −
fix *dt*
let *?rngX* = *λξ*. *x* + *t* < *X ξ* ∧ *X ξ* ≤ *x* + *t* + *dt*
have $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ *alive x*). *?rngX ξ* | *X ξ* > *x* + *t*) =
$\mathcal{P}$(*ξ in* (($\mathfrak{M}$ ⌊ *alive x*) ⌊ {*η* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X η* > *x* + *t*}). *?rngX ξ*)
using ⋆ **by** (*rewrite alivex-PS.cond-prob-space-cond-prob*) *simp-all*
also have ... = $\mathcal{P}$(*ξ in* ($\mathfrak{M}$ ⌊ {*η* ∈ *space* $\mathfrak{M}$. *X η* > *x* + *t*}). *?rngX ξ*)
proof −
have ($\mathfrak{M}$ ⌊ *alive x*) ⌊ {*η* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X η* > *x* + *t*} =
$\mathfrak{M}$ ⌊ {*η* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X η* > *x* + *t*}
apply (*rewrite MM-PS.cond-cond-prob-space*, *simp-all*)
unfolding *alive-def* **using** *that* **by** *force*
also have ... = $\mathfrak{M}$ ⌊ {*η* ∈ *space* $\mathfrak{M}$. *X η* > *x* + *t*} **by** *simp*
finally have ($\mathfrak{M}$ ⌊ *alive x*) ⌊ {*η* ∈ *space* ($\mathfrak{M}$ ⌊ *alive x*). *X η* > *x* + *t*} =

$\mathfrak{M} \restriction \{\eta \in space\ \mathfrak{M}.\ X\ \eta > x + t\}$.
thus *?thesis* **by** *simp*
qed
also have $\ldots = \mathcal{P}(\xi\ in\ \mathfrak{M}.\ x + t < X\ \xi \wedge X\ \xi \leq x + t + dt \mid X\ \xi > x + t)$
by (*rewrite MM-PS.cond-prob-space-cond-prob*, *simp-all add*: *pred-def sets.sets-Collect-conj*)
finally show $\mathcal{P}(\xi\ in\ (\mathfrak{M} \restriction alive\ x).\ ?rngX\ \xi \mid X\ \xi > x + t) =$
$\mathcal{P}(\xi\ in\ \mathfrak{M}.\ x + t < X\ \xi \wedge X\ \xi \leq x + t + dt \mid X\ \xi > x + t)$.
qed
thus *?thesis*
apply −
by (*rule Lim-cong*, *rule eventually-at-rightI*[*of 0 1*]; *simp*)
qed
also have $\ldots = \$\mu\text{-}(x{+}t)$ **unfolding** *force-mortal-def MM-PS.hazard-rate-def*
by *simp*
finally show *?thesis* .
qed

lemma *pdfTx-p-mu*: *pdfT x t* = $\$p\text{-}\{t\&x\} * \$\mu\text{-}(x{+}t)$
if (*cdf* (*distr* ($\mathfrak{M} \restriction alive\ x$) *borel* (*T x*))) *differentiable at t t* > *0* **for** *t*::*real*
proof (*cases* ‹$x{+}t < \$\psi$›)
case *True*
hence [*simp*]: $t \geq 0$ **and** (*ccdf* (*distr* ($\mathfrak{M} \restriction alive\ x$) *borel* (*T x*))) $t \neq 0$
using *that p-0-equiv* **unfolding** *survive-def* **by** *auto*
have *deriv* (*cdf* (*distr* ($\mathfrak{M} \restriction alive\ x$) *borel* (*T x*))) *t* =
ccdf (*distr* ($\mathfrak{M} \restriction alive\ x$) *borel* (*T x*)) *t* ∗ *alivex-PS.hazard-rate* (*T x*) *t*
by (*rule alivex-PS.deriv-cdf-hazard-rate*; *simp add*: *that*)
thus *?thesis* **unfolding** *survive-def pdfT-def* **using** *hazard-rate-Tx-mu True that*
by *simp*
next
case *False*
hence $x{+}t \geq \$\psi$ **by** *simp*
thus *?thesis* **using** *pdfTx-beyond-0 mu-beyond-0* **by** *simp*
qed

lemma *deriv-t-p-mu*: *deriv* ($\lambda s.\ \$p\text{-}\{s\&x\}$) *t* = − $\$p\text{-}\{t\&x\} * \$\mu\text{-}(x{+}t)$
if ($\lambda s.\ \$p\text{-}\{s\&x\}$) *differentiable at t t* > *0* **for** *t*::*real*
proof −
let *?cdfTx* = *cdf* (*distr* ($\mathfrak{M} \restriction alive\ x$) *borel* (*T x*))
have *diff*: *?cdfTx differentiable at t*
using *that distrTx-RD.differentiable-cdf-ccdf* **unfolding** *survive-def* **by** *blast*
hence *deriv ?cdfTx t* = $\$p\text{-}\{t\&x\} * \$\mu\text{-}(x{+}t)$ **using** *that pdfTx-p-mu pdfT-def*
by *simp*
moreover have *deriv ?cdfTx t* = − *deriv* ($\lambda s.\ \$p\text{-}\{s\&x\}$) *t*
using *that diff distrTx-RD.deriv-cdf-ccdf* **unfolding** *survive-def* **by** *presburger*
ultimately show *?thesis* **by** *simp*
qed

lemma *pdfTx-p-mu-AE*: *AE s in lborel*. *s* > *0* ⟶ *pdfT x s* = $\$p\text{-}\{s\&x\} * \$\mu\text{-}(x{+}s)$

proof −
let *?cdfX* = *cdf* (*distr* $\mathfrak{M}$ *borel X*)
let *?cdfTx* = *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))
from *cdfX-differentiable-AE* **obtain** *N*
where *diff*: $\bigwedge r$. *r* ∈ *space lborel* − *N* ⟹ *?cdfX differentiable at r*
and *null*: *N* ∈ *null-sets lborel*
using *AE-E3* **by** *blast*
let *?N′* = {*s*. *x*+*s* ∈ *N*}
have *?N′* ∈ *null-sets lborel*
using *null-sets-translation*[*of N* −*x*, *simplified*] *null* **by** (*simp add*: *add.commute*)
moreover have {*s* ∈ *space lborel*. ¬ (*s* > *0* ⟶ *pdfT x s* = $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*))} ⊆ *?N′*
proof (*rewrite Compl-subset-Compl-iff*[*THEN sym*], *safe*)
fix *s*::*real*
assume *s* ∈ *space lborel* **and** *x*+*s* ∉ *N* **and** *s* > *0*
thus *pdfT x s* = $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)
apply (*intro pdfTx-p-mu*, *simp-all*)
by (*rewrite differentiable-cdfX-cdfTx*[*THEN sym*]; *simp add*: *diff*)
qed
ultimately show *?thesis* **using** *AE-I′*[*of ?N′*] **by** *simp*
qed

lemma *LBINT-p-mu-q-defer*: (*LBINT s*:{*f*<..*f*+*t*}. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)) = $*q*-{*f*|*t*&*x*}
if *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
proof −
have (*LBINT s*:{*f*<..*f*+*t*}. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)) = (*LBINT s*:{*f*<..*f*+*t*}. *pdfT x s*)
apply (*rule set-lebesgue-integral-cong-AE*; *simp*)
apply (*simp add*: *survive-def*)
using *pdfTx-p-mu-AE* **apply** (*rule AE-mp*)
using *that* **by** (*intro always-eventually*; *simp add*: *ereal-less-le*)
also have ... = *enn2real* (∫⁺*s*∈{*f*<..*f*+*t*}. *ennreal* (*pdfT x s*) *∂lborel*)
proof −
have (∫⁺*s*∈{*f*<..*f*+*t*}. *ennreal* (*pdfT x s*) *∂lborel*) < ⊤
proof −
have (∫⁺*s*∈{*f*<..*f*+*t*}. *ennreal* (*pdfT x s*) *∂lborel*) ≤ ∫⁺*s*. *ennreal* (*pdfT x s*) *∂lborel*
by (*smt* (*verit*) *indicator-simps le-zero-eq linorder-le-cases mult.right-neutral mult-zero-right nn-integral-mono*)
also have ... < ⊤ **using** *nn-integral-pdfTx-1* **by** *simp*
finally show *?thesis* **.**
qed
thus *?thesis*
unfolding *set-lebesgue-integral-def*
by (*rewrite enn2real-nn-integral-eq-integral*[**where** *g*=*λs*. *pdfT x s* ∗ *indicator* {*f*<..*f*+*t*} *s*])
(*simp-all add*: *pdfTx-nonneg mult.commute ennreal-indicator ennreal-mult′*)
qed
also have ... = *measure* (*density lborel* (*pdfT x*)) {*f*<..*f*+*t*}

unfolding *measure-def* **by** (*rewrite emeasure-density*; *simp*)
also have ... = *measure* (*distr* (𝔐 ⌊ *alive x*) *lborel* (*T x*)) {*f*<..*f*+*t*}
using *distributed-pdfTx* **unfolding** *distributed-def* **by** *simp*
also have ... =
cdf (*distr* (𝔐 ⌊ *alive x*) *lborel* (*T x*)) (*f*+*t*) − *cdf* (*distr* (𝔐 ⌊ *alive x*) *lborel* (*T x*)) *f*
using *that finite-borel-measure.cdf-diff-eq2*
by (*smt* (*verit*) *distrTx-RD.finite-borel-measure-axioms distr-cong sets-lborel*)
also have ... = $*q*-{*f*|*t*&*x*}
using *q-defer-q die-def that* **by** (*metis distr-cong sets-lborel x-lt-psi*)
finally show *?thesis* **.**
qed

corollary *LBINT-p-mu-q*: (*LBINT s*:{*0*<..*t*}. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)) = $*q*-{*t*&*x*}
if *t* ≥ *0* **for** *t*::*real*
using *LBINT-p-mu-q-defer that* **by** *force*

lemma *set-integrable-p-mu*: *set-integrable lborel* {*f*<..*f*+*t*} (λ*s*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*))
if *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
proof −
have (λ*s*. $*p*-{*s*&*x*}) ∈ *borel-measurable borel* **unfolding** *survive-def* **by** *simp*
moreover have *AE s in lborel*. *f* < *s* ∧ *s* ≤ *f* + *t* ⟶ $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) = *pdfT x s*
using *pdfTx-p-mu-AE* **apply** (*rule AE-mp*)
using *that* **by** (*intro always-eventually*; *simp add*: *ereal-less-le*)
moreover have *set-integrable lborel* {*f*<..*f*+*t*} (*pdfT x*) **using** *pdfTx-set-integrable* **by** *simp*
ultimately show *?thesis* **by** (*rewrite set-integrable-cong-AE*[**where** *g*=*pdfT x*]; *simp*)
qed

lemma *p-mu-has-integral-q-defer-Ioc*:
((λ*s*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)) *has-integral* $*q*-{*f*|*t*&*x*}) {*f*<..*f*+*t*}
if *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
apply (*rewrite LBINT-p-mu-q-defer*[*THEN sym*], *simp-all add*: *that*)
apply (*rewrite set-borel-integral-eq-integral*, *simp add*: *set-integrable-p-mu that*)
by (*rewrite has-integral-integral*[*THEN sym*];
simp add: *set-borel-integral-eq-integral set-integrable-p-mu that*)

lemma *p-mu-has-integral-q-defer-Icc*:
((λ*s*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)) *has-integral* $*q*-{*f*|*t*&*x*}) {*f*..*f*+*t*} **if** *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
proof −
have *negligible* {*f*} **by** *simp*
hence [*simp*]: *negligible* ({*f*..*f*+*t*} − {*f*<..*f*+*t*})
by (*smt* (*verit*) *Diff-iff atLeastAtMost-iff greaterThanAtMost-iff insertI1*
negligible-sing negligible-subset subsetI)
have [*simp*]: *negligible* ({*f*<..*f*+*t*} − {*f*..*f*+*t*}) **by** (*simp add*: *subset-eq*)
show *?thesis*

```
    apply (rewrite has-integral-spike-set-eq[where T={f<..f+t}])
     apply (rule negligible-subset[of {f..f+t} - {f<..f+t}], simp, blast)
    apply (rule negligible-subset[of {f<..f+t} - {f..f+t}], simp, blast)
    using p-mu-has-integral-q-defer-Ioc that by simp
qed

corollary p-mu-has-integral-q-Icc:
  ((λs. $p-{s&x} * $μ-(x+s)) has-integral $q-{t&x}) {0..t} if t ≥ 0 for t::real
  using p-mu-has-integral-q-defer-Icc[where f=0] that by simp

corollary p-mu-integrable-on-Icc:
  (λs. $p-{s&x} * $μ-(x+s)) integrable-on {0..t} if t ≥ 0 for t::real
  using p-mu-has-integral-q-Icc that by auto

lemma e-ennreal-p-mu: (∫⁺ξ. ennreal (T x ξ) ∂(𝔐 ⌊ alive x)) =
  (∫⁺s∈{0..}. ennreal ($p-{s&x} * $μ-(x+s) * s) ∂lborel)
proof −
  have [simp]: sym-diff {0..} {0<..} = {0::real} by force
  have (∫⁺ξ. ennreal (T x ξ) ∂(𝔐 ⌊ alive x)) = (∫⁺s∈{0..}. ennreal (pdfT x s *
s) ∂lborel)
    by (rewrite nn-integral-T-pdfT[where g=λs. s]; simp)
  also have ... = (∫⁺s∈{0<..}. ennreal (pdfT x s * s) ∂lborel)
    by (rewrite nn-integral-null-delta; force)
  also have ... = (∫⁺s∈{0<..}. ennreal ($p-{s&x} * $μ-(x+s) * s) ∂lborel)
    apply (rule nn-integral-cong-AE)
    using pdfTx-p-mu-AE apply (rule AE-mp, intro AE-I2) by force
  also have ... = (∫⁺s∈{0..}. ennreal ($p-{s&x} * $μ-(x+s) * s) ∂lborel)
    by (rewrite nn-integral-null-delta[THEN sym]; force)
  finally show ?thesis .
qed

lemma e-LBINT-p-mu: $e‘○-x = (LBINT s:{0..}. $p-{s&x} * $μ-(x+s) * s)
  — Note that 0 = 0 holds when the life expectation diverges.
proof −
  let ?f = λs. $p-{s&x} * $μ-(x+s) * s
  have [simp]: (λs. ?f s * indicat-real {0..} s) ∈ borel-measurable borel
    by measurable (simp-all add: survive-def)
  hence [simp]: (λs. indicat-real {0..} s * ?f s) ∈ borel-measurable borel
    by (meson measurable-cong mult.commute)
  have [simp]: AE s in lborel. ?f s * indicat-real {0..} s ≥ 0
  proof −
    have AE s in lborel. pdfT x s * s * indicat-real {0..} s ≥ 0
      using pdfTx-nonneg pdfTx-nonpos-0 x-lt-psi
     by (intro AE-I2) (smt (verit, del-insts) indicator-pos-le mult-eq-0-iff mult-nonneg-nonneg)
    thus ?thesis
      apply (rule AE-mp)
      using pdfTx-p-mu-AE apply (rule AE-mp)
     by (rule AE-I2) (metis atLeast-iff indicator-simps(2) mult-eq-0-iff order-le-less)
  qed
```

```
  hence [simp]: AE s in lborel. indicat-real {0..} s * ?f s ≥ 0
    by (metis (no-types, lifting) AE-cong mult.commute)
  show ?thesis
  proof (cases ‹integrable (𝔐 ⌊ alive x) (T x)›)
    case True
    hence ennreal ($e‘∘-x) = ∫⁺ξ. ennreal (T x ξ) ∂(𝔐 ⌊ alive x)
      unfolding life-expect-def apply (rewrite nn-integral-eq-integral, simp-all)
      by (smt (verit) AE-I2 alivex-Tx-pos)
    also have ... = ∫⁺s. ennreal (?f s * indicat-real {0..} s) ∂lborel
      by (simp add: nn-integral-set-ennreal e-ennreal-p-mu)
    also have ... = ennreal (LBINT s:{0..}. ?f s)
    proof −
      have integrable lborel (λs. ?f s * indicat-real {0..} s)
      proof −
        have (∫⁺s. ennreal (?f s * indicat-real {0..} s) ∂lborel) < ∞
          by (metis calculation ennreal-less-top infinity-ennreal-def)
        thus ?thesis by (intro integrableI-nonneg; simp)
      qed
      thus ?thesis
        unfolding set-lebesgue-integral-def
        by (rewrite nn-integral-eq-integral, simp-all) (meson mult.commute)
    qed
    finally have ennreal ($e‘∘-x) = ennreal (LBINT s:{0..}. ?f s) .
    moreover have (LBINT s:{0..}. ?f s) ≥ 0
      unfolding set-lebesgue-integral-def by (rule integral-nonneg-AE) simp
    ultimately show ?thesis using e-nonneg by simp
  next
    case False
   hence $e‘∘-x = 0 unfolding life-expect-def using not-integrable-integral-eq by
force
    also have ... = (LBINT s:{0..}. ?f s)
    proof −
      have ∞ = ∫⁺ξ. ennreal (T x ξ) ∂(𝔐 ⌊ alive x)
        using nn-integral-nonneg-infinite False
      by (smt (verit) AE-cong Tx-alivex-measurable alivex-PS.AE-prob-1 alivex-PS.prob-space
          alivex-Tx-pos nn-integral-cong)
     hence 0 = enn2real (∫⁺s∈{0..}. ennreal (?f s) ∂lborel) using e-ennreal-p-mu
by simp
      also have ... = (LBINT s:{0..}. ?f s)
        unfolding set-lebesgue-integral-def apply (rewrite integral-eq-nn-integral,
simp-all)
        by (simp add: indicator-mult-ennreal mult.commute)
      finally show ?thesis by simp
    qed
    finally show ?thesis .
  qed
qed

lemma e-integral-p-mu: $e‘∘-x = integral {0..} (λs. $p-{s&x} * $μ-(x+s) * s)
```

— Note that *0 = 0* holds when the life expectation diverges.
proof −
have (*LBINT s*:{*0*..}. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) ∗ *s*) = *integral* {*0*..} (*λs*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) ∗ *s*)
proof −
have *AE s in lborel*. *s* ≥ *0* ⟶ $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) ∗ *s* ≥ *0*
proof −
have *AE s in lborel*. $*μ*-(*x*+*s*) ≥ *0* **by** (*rule AE-translation*, *rule mu-nonneg-AE*)
with *p-nonneg* **show** *?thesis* **by** *force*
qed
moreover have (*λs*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) ∗ *s*) ∈ *borel-measurable borel*
unfolding *survive-def* **by** *simp*
ultimately show *?thesis* **by** (*intro set-borel-integral-eq-integral-nonneg-AE*; *simp*)
qed
thus *?thesis* **using** *e-LBINT-p-mu* **by** *simp*
qed

end

lemma *p-has-real-derivative-x-ccdfX*:
((*λy*. $*p*-{*t*&*y*}) *has-real-derivative*
((*deriv svl* (*x*+*t*) ∗ *svl x* − *svl* (*x*+*t*) ∗ *deriv svl x*) / (*svl x*)²)) (*at x*)
if *svl* ≡ *ccdf* (*distr* 𝔑 *borel X*) *svl differentiable at x svl differentiable at* (*x*+*t*)
t ≥ *0 x* < $*ψ* **for** *x t* :: *real*
proof −
have *open* {*y*. *svl differentiable at y*} **using** *ccdfX-differentiable-open-set that* **by** *simp*
with *that* **obtain** *e0* **where** *e0-pos*: *e0* > *0* **and** *ball-e0*: *ball x e0* ⊆ {*y*. *svl differentiable at y*}
using *open-contains-ball* **by** *blast*
define *e* **where** *e* ≡ *if* $*ψ* < ∞ *then min e0* (*real-of-ereal* $*ψ* − *x*) *else e0*
have *e* > *0* ∧ *ball x e* ⊆ {*y*. *svl y* ≠ *0* ∧ *svl differentiable at y*}
proof (*cases* ‹$*ψ* < ∞›)
case *True*
hence *e* > *0*
proof −
have *real-of-ereal* $*ψ* − *x* > *0* **using** *not-inftyI True that* **by** *force*
with *e0-pos* **show** *?thesis* **unfolding** *e-def* **using** *True* **by** *simp*
qed
moreover have *ball x e* ⊆ {*y*. *svl y* ≠ *0*}
proof −
have *e* ≤ *real-of-ereal* $*ψ* − *x* **unfolding** *e-def* **using** *True* **by** *simp*
hence *ball x e* ⊆ {..< *real-of-ereal* $*ψ*} **unfolding** *ball-def dist-real-def* **by** *force*
thus *?thesis* **using** *that ccdfX-0-equiv*
using *True ereal-less-real-iff psi-nonneg* **by** *auto*
qed
moreover have *ball x e* ⊆ {*y*. *svl differentiable at y*}

proof −
have $e \leq e0$ **unfolding** *e-def* **using** *True* **by** *simp*
hence *ball x e* $\subseteq$ *ball x e0* **by** *force*
with *ball-e0* **show** *?thesis* **by** *simp*
qed
ultimately show *?thesis* **by** *blast*
next
case *False*
hence *ball x e* $\subseteq$ $\{y.\ svl\ y \neq 0\}$ **using** *ccdfX-0-equiv that* **by** *simp*
with *False* **show** *?thesis* **unfolding** *e-def* **using** *e0-pos ball-e0* **by** *force*
qed
hence *e-pos*: $e > 0$ **and** *ball-e*: *ball x e* $\subseteq$ $\{y.\ svl\ y \neq 0 \wedge svl$ *differentiable at* $y\}$
by *auto*
hence *ball-svl*: $\bigwedge y.\ y \in ball\ x\ e \Longrightarrow svl\ y \neq 0 \wedge svl$ *field-differentiable at* y
using *differentiable-eq-field-differentiable-real* **by** *blast*
hence $\bigwedge y.\ y \in ball\ x\ e \Longrightarrow$ \$*p*-$\{t\&y\} = svl\ (y+t)\ /\ svl\ y$
unfolding *survive-def* **using** *that ccdfX-0-equiv* **by** (*rewrite ccdfTx-ccdfX*, *simp-all*) *force*
moreover from *ball-svl* **have** $((\lambda y.\ svl\ (y+t)\ /\ svl\ y)$ *has-real-derivative*
$((deriv\ svl\ (x+t) * svl\ x - svl\ (x+t) * deriv\ svl\ x)\ /\ (svl\ x)^2))\ (at\ x)$
apply (*rewrite power2-eq-square*)
apply (*rule DERIV-divide*)
using *DERIV-deriv-iff-real-differentiable DERIV-shift that* **apply** *blast*
using *that DERIV-deriv-iff-real-differentiable* **apply** *simp*
by (*simp add*: *e-pos*)
ultimately show *?thesis*
using *e-pos*
apply (*rewrite has-field-derivative-cong-eventually*[**where** $g=\lambda y.\ svl\ (y+t)\ /\ svl\ y$], *simp-all*)
by (*smt* (*verit*) *dist-commute eventually-at*)
qed

lemma *p-has-real-derivative-x-p-mu*:
$((\lambda y.$ \$*p*-$\{t\&y\})$ *has-real-derivative* \$*p*-$\{t\&x\} * (\$\mu$-$x - \$\mu$-$(x+t)))\ (at\ x)$
if *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at* $(x+t)$
$t \geq 0\ x < \$\psi$ **for** $x\ t$ :: *real*
proof (*cases* ‹$x+t < \$\psi$›)
case *True*
let *?svl* = *ccdf* (*distr* $\mathfrak{M}$ *borel X*)
have [*simp*]: *?svl x* $\neq$ *0* **using** *that ccdfX-0-equiv* **by** (*smt* (*verit*) *le-ereal-le linorder-not-le*)
have [*simp*]: *?svl field-differentiable at* $(x+t)$
using *that differentiable-eq-field-differentiable-real* **by** *simp*
have $((\lambda y.$ \$*p*-$\{t\&y\})$ *has-real-derivative*
$((deriv\ ?svl\ (x+t) * ?svl\ x - ?svl\ (x+t) * deriv\ ?svl\ x)\ /\ (?svl\ x)^2))\ (at\ x)$
using *p-has-real-derivative-x-ccdfX that* **by** *simp*
moreover have $(deriv\ ?svl\ (x+t) * ?svl\ x - ?svl\ (x+t) * deriv\ ?svl\ x)\ /\ (?svl\ x)^2 =$

```
    $p-{t&x} * ($μ-x - $μ-(x+t)) (is ?LHS = ?RHS)
  proof -
    have deriv ?svl (x+t) = deriv (λy. ?svl (y+t)) x
      using that by (metis DERIV-deriv-iff-real-differentiable DERIV-imp-deriv
DERIV-shift)
    hence ?LHS = (deriv (λy. ?svl (y+t)) x * ?svl x - ?svl (x+t) * deriv ?svl x)
/ (?svl x)^2
      by simp
    also have ... = (deriv (λy. ?svl (y+t)) x - ?svl (x+t) * deriv ?svl x / ?svl
x) / ?svl x
      by (simp add: add-divide-eq-if-simps(4) power2-eq-square)
    also have ... = (- ?svl (x+t) * $μ-(x+t) + ?svl (x+t) * $μ-x) / ?svl x
    proof -
      have deriv (λy. ?svl (y+t)) x = - ?svl (x+t) * $μ-(x+t)
      apply (rewrite add.commute, rewrite deriv-shift[THEN sym], rewrite add.commute,
simp)
        using add.commute that by (metis MM-PS.deriv-ccdf-hazard-rate X-RV
force-mortal-def)
      moreover have - ?svl (x+t) * deriv ?svl x / ?svl x = ?svl (x+t) * $μ-x
        using that by (simp add: MM-PS.deriv-ccdf-hazard-rate force-mortal-def)
      ultimately show ?thesis by simp
    qed
    also have ... = ?svl (x+t) * ($μ-x - $μ-(x+t)) / ?svl x by (simp add:
mult-diff-mult)
    also have ... = ?RHS unfolding survive-def using ccdfTx-ccdfX that by
simp
    ultimately show ?thesis by simp
  qed
  ultimately show ?thesis by simp
next
  case False
  hence ptx-0: $p-{t&x} = 0 using p-0-equiv that by simp
  moreover have ((λy. $p-{t&y}) has-real-derivative 0) (at x)
  proof -
    have ⋀y. x < y ⟹ y < $ψ ⟹ $p-{t&y} = 0
    using False p-0-equiv that by (smt (verit, ccfv-SIG) ereal-less-le linorder-not-le)
    hence ∀_F x in at-right x. $p-{t&x} = 0
      apply (rewrite eventually-at-right-field)
    using that by (meson ereal-dense2 ereal-le-less less-eq-ereal-def less-ereal.simps)
    hence ((λy. $p-{t&y}) has-real-derivative 0) (at-right x)
      using ptx-0 by (rewrite has-field-derivative-cong-eventually[where g=λ-. 0];
simp)
    thus ?thesis
      using vector-derivative-unique-within p-has-real-derivative-x-ccdfX that
    by (metis has-field-derivative-at-within has-real-derivative-iff-has-vector-derivative
        trivial-limit-at-right-real)
  qed
  ultimately show ?thesis by simp
qed
```

corollary *deriv-x-p-mu*: *deriv* (λy. \$*p*-{*t*&*y*}) *x* = \$*p*-{*t*&*x*} * (\$$\mu$-*x* − \$$\mu$-(*x*+*t*))
if *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x ccdf* (*distr* $\mathfrak{M}$ *borel X*) *differentiable at* (*x*+*t*)
t ≥ *0 x* < \$$\psi$ **for** *x t* :: *real*
using *that p-has-real-derivative-x-p-mu DERIV-imp-deriv* **by** *blast*

lemma *e-has-derivative-mu-e*: ((λ*x*. \$*e*‘∘-*x*) *has-real-derivative* (\$$\mu$-*x* * \$*e*‘∘-*x* − *1*)) (*at x*)
if ⋀*x*. *x*∈{*a*<..<*b*} ⟹ *set-integrable lborel* {*x*..} (*ccdf* (*distr* $\mathfrak{M}$ *borel X*))
ccdf (*distr* $\mathfrak{M}$ *borel X*) *differentiable at x x*∈{*a*<..<*b*} *b* ≤ \$$\psi$
for *a b x* :: *real*
proof −
let *?svl* = *ccdf* (*distr* $\mathfrak{M}$ *borel X*)
have *x-lt-psi*[*simp*]: *x* < \$$\psi$ **using** *that ereal-le-less* **by** *simp*
hence *svlx-neq0*[*simp*]: *?svl x* ≠ *0* **by** (*simp add*: *ccdfX-0-equiv linorder-not-le*)
define *d* ::*real* **where** *d* ≡ *min* (*b*−*x*) (*x*−*a*)
have *d-pos*: *d* > *0* **unfolding** *d-def* **using** *that ereal-less-real-iff* **by** *force*
have *e-svl*: ⋀*y*. *y* < \$$\psi$ ⟹ \$*e*‘∘-*y* = (*LBINT t*:{*0*..}. *?svl* (*y*+*t*)) / *?svl y*
apply (*rewrite e-LBINT-p*, *simp*)
apply (*rewrite set-integral-divide-zero*[*THEN sym*])
apply (*rule set-lebesgue-integral-cong*, *simp-all*)
unfolding *survive-def* **using** *ccdfTx-ccdfX* **by** *force*
have ((λ*y*. *LBINT t*:{*0*..}. *?svl* (*y*+*t*)) *has-real-derivative* (− *?svl x*)) (*at x*)
proof −
{ fix *y* **assume** *dist y x* < *d*
hence *y-ab*: *y* ∈ {*a*<..<*b*} **unfolding** *d-def dist-real-def* **by** *force*
hence *set-integrable lborel* {*y*..} *?svl* **using** *that* **by** *simp*
hence *set-integrable lborel* (*einterval y* ∞) *?svl*
by (*rewrite set-integrable-discrete-difference*[**where** *X*={*y*}]; *simp*) *force*
moreover have ⋀*u*. ((λ*u*. *u*−*y*) *has-real-derivative* (*1*−*0*)) (*at u*)
by (*rule derivative-intros*) *auto*
moreover have ⋀*u*. *y* < *u* ⟹ *isCont* (λ*t*. *?svl* (*y*+*t*)) (*u*−*y*)
apply (*rewrite add.commute*, *rewrite isCont-shift*, *simp*)
using *ccdfX-continuous continuous-on-eq-continuous-within* **by** *blast*
moreover have ((*ereal* ∘ (λ*u*. *u*−*y*) ∘ *real-of-ereal*) ⟶ *0*) (*at-right* (*ereal y*))
by (*smt* (*verit*, *ccfv-SIG*) *LIM-zero Lim-cong-within ereal-tendsto-simps1*(*2*) *ereal-tendsto-simps2*(*1*) *tendsto-ident-at zero-ereal-def*)
moreover have ((*ereal* ∘ (λ*u*. *u*−*y*) ∘ *real-of-ereal*) ⟶ ∞) (*at-left* ∞)
proof −
have ⋀*r u*. *r*+*y*+*1* ≤ *u* ⟹ *r* < *u*−*y* **by** *auto*
hence ⋀*r*. ∀$_F$ *u in at-top*. *r* < *u* − *y* **by** (*rule eventually-at-top-linorderI*)
thus *?thesis* **by** (*rewrite ereal-tendsto-simps*, *rewrite tendsto-PInfty*) *simp*
qed
ultimately have (*LBINT t*=*0*..∞. *?svl* (*y*+*t*)) = (*LBINT u*=*y*..∞. *?svl* (*y*+(*u*−*y*)) * *1*)
using *distrX-RD.ccdf-nonneg* **by** (*intro interval-integral-substitution-nonneg*(*2*); *simp*)

moreover have (*LBINT t*:{*0*..}. *?svl* (*y*+*t*)) = (*LBINT t*=*0*..∞. *?svl* (*y*+*t*))
unfolding *interval-lebesgue-integral-def einterval-def* **apply** *simp*
by (*rule set-integral-discrete-difference*[**where** *X*={*0*}]; *force*)
moreover have (*LBINT u*=*y*..∞. *?svl* (*y*+(*u*−*y*))∗*1*) = (*LBINT u*:{*y*..}. *?svl u*)
unfolding *interval-lebesgue-integral-def einterval-def* **apply** *simp*
by (*rule set-integral-discrete-difference*[**where** *X*={*y*}]; *force*)
ultimately have (*LBINT t*:{*0*..}. *?svl* (*y*+*t*)) = (*LBINT u*:{*y*..}. *?svl u*) **by** *simp* **}**
hence ∀ *F y in nhds x*. (*LBINT t*:{*0*..}. *?svl* (*y*+*t*)) = (*LBINT u*:{*y*..}. *?svl u*)
using *d-pos* **by** (*rewrite eventually-nhds-metric*) *auto*
moreover have ((λ*y*. *LBINT u*:{*y*..}. *?svl u*) *has-real-derivative* (− *?svl x*)) (*at x*)
proof −
have ((λ*y*. *integral* {*y*..*b*} *?svl*) *has-real-derivative* (− *?svl x*)) (*at x within* {*a*..*b*})
using *that* **apply** (*intro integral-has-real-derivative*′; *simp*)
using *ccdfX-continuous continuous-on-subset* **by** *blast*
hence ((λ*y*. *integral* {*y*..*b*} *?svl*) *has-real-derivative* (− *?svl x*)) (*at x*)
using *that* **apply** (*rewrite at-within-open*[**where** *S*={*a*<..<*b*}, *THEN sym*], *simp-all*)
by (*rule DERIV-subset*[**where** *s*={*a*..*b*}]) *auto*
moreover have ∀ *F y in nhds x*. (*LBINT u*:{*y*..*b*}. *?svl u*) = *integral* {*y*..*b*} *?svl*
apply (*rewrite eventually-nhds-metric*)
using *d-pos* **by** (*metis ccdfX-integrable-Icc set-borel-integral-eq-integral*(*2*))
ultimately have ((λ*y*. *LBINT u*:{*y*..*b*}. *?svl u*) *has-real-derivative* (− *?svl x*)) (*at x*)
by (*rewrite DERIV-cong-ev*; *simp*)
hence ((λ*y*. (*LBINT u*:{*y*..*b*}. *?svl u*) + (*LBINT u*:{*b*<..}. *?svl u*)) *has-real-derivative* (− *?svl x*)) (*at x*)
by (*rewrite to* − *?svl x* + *0 add-0-right*[*THEN sym*], *rule DERIV-add*; *simp*)
moreover have ∀ *F y in nhds x*.
(*LBINT u*:{*y*..}. *?svl u*) = (*LBINT u*:{*y*..*b*}. *?svl u*) + (*LBINT u*:{*b*<..}. *?svl u*)
proof −
{ fix *y* **assume** *dist y x* < *d*
hence *y-ab*: *y* ∈ {*a*<..<*b*} **unfolding** *d-def dist-real-def* **by** *force*
hence *set-integrable lborel* {*y*..} *?svl* **using** *that* **by** *simp*
hence *set-integrable lborel* {*y*..*b*} *?svl set-integrable lborel* {*b*<..} *?svl*
apply (*rule set-integrable-subset*, *simp-all*)+
using *y-ab* **by** *force*
moreover have {*y*..*b*} ∩ {*b*<..} = {} {*y*..} = {*y*..*b*} ∪ {*b*<..} **using** *y-ab* **by** *force*+
ultimately have
(*LBINT u*:{*y*..}. *?svl u*) = (*LBINT u*:{*y*..*b*}. *?svl u*) + (*LBINT u*:{*b*<..}. *?svl u*)
using *set-integral-Un* **by** *simp* **}**
thus *?thesis* **using** *d-pos* **by** (*rewrite eventually-nhds-metric*) *blast*

```
    qed
    ultimately show ?thesis by (rewrite has-field-derivative-cong-ev; simp)
   qed
   ultimately show ?thesis by (rewrite DERIV-cong-ev; simp)
  qed
  moreover have (?svl has-real-derivative (deriv ?svl x)) (at x)
   using that DERIV-deriv-iff-real-differentiable by blast
  ultimately have ((λy. (LBINT t:{0..}. ?svl (y+t)) / ?svl y) has-real-derivative
   ((− ?svl x) * ?svl x − (LBINT t:{0..}. ?svl (x+t)) * deriv ?svl x) / (?svl x *
?svl x)) (at x)
   by (rule DERIV-divide) simp
  moreover have eventually (λy. (LBINT t:{0..}. ?svl (y+t)) / ?svl y = $e‘○-y)
(at x)
  proof −
   { fix y assume dist y x < d y ≠ x
    hence y < $ψ
     unfolding dist-real-def d-def using that ereal-le-less by fastforce
    hence $e‘○-y = (LBINT t:{0..}. ?svl (y+t)) / ?svl y by (rule e-svl) }
   thus ?thesis
    apply (rewrite eventually-at-filter, rewrite eventually-nhds-metric)
    using d-pos that by metis
  qed
  ultimately have ((λy. $e‘○-y) has-real-derivative
   ((− ?svl x) * ?svl x − (LBINT t:{0..}. ?svl (x+t)) * deriv ?svl x) / (?svl x *
?svl x)) (at x)
   using e-svl by (rewrite has-field-derivative-cong-eventually[THEN sym]; simp)
  moreover have
   ((− ?svl x) * ?svl x − (LBINT t:{0..}. ?svl (x+t)) * deriv ?svl x) / (?svl x *
?svl x) =
    $μ-x * $e‘○-x − 1 (is ?LHS = ?RHS)
  proof −
   have ?LHS = −1 + (LBINT t:{0..}. ?svl (x+t)) / ?svl x * (− deriv ?svl x /
?svl x)
    by simp (smt (verit) svlx-neq0 add-divide-distrib divide-eq-minus-1-iff
      mult-minus-left real-divide-square-eq)
   also have ... = −1 + $e‘○-x * $μ-x using e-svl mu-deriv-ccdf that by force
   also have ... = ?RHS by simp
   finally show ?thesis .
  qed
  ultimately show ?thesis by simp
qed

corollary e-has-derivative-mu-e′: ((λx. $e‘○-x) has-real-derivative ($μ-x * $e‘○-x
− 1)) (at x)
  if ⋀x. x∈{a<..<b} ⟹ ccdf (distr 𝔐 borel X) integrable-on {x..}
   ccdf (distr 𝔐 borel X) differentiable at x x∈{a<..<b} b ≤ $ψ
  for a b x :: real
  using that apply (intro e-has-derivative-mu-e[where a=a], simp-all)
  using distrX-RD.ccdf-nonneg by (rewrite integrable-on-iff-set-integrable-nonneg;
```

simp)

5.2.11 Properties of Curtate Life Expectation

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: $x < \$\psi$
begin

lemma *isCont-p-nat*: *isCont* ($\lambda t.$ \$*p*-{*t*&*x*}) (*k* + (*1*::*real*)) **for** *k*::*nat*
proof −
fix *k*::*nat*
have *continuous-on* {*0*<..} ($\lambda t.$ \$*p*-{*t*&*x*})
unfolding *survive-def*
using *ccdfTx-continuous-on-nonneg continuous-on-subset Ioi-le-Ico x-lt-psi* **by** *blast*
hence $\forall\, t{\in}\{0{<}..\}.$ *isCont* ($\lambda t.$ \$*p*-{*t*&*x*}) *t*
using *continuous-on-eq-continuous-at open-greaterThan* **by** *blast*
thus *isCont* ($\lambda t.$ \$*p*-{*t*&*x*}) (*real k*+*1*) **by** *force*
qed

lemma *curt-e-sum-p-smooth*: \$*e*-*x* = ($\sum k.$ \$*p*-{*k*+*1*&*x*}) **if** *summable* ($\lambda k.$ \$*p*-{*k*+*1*&*x*})
using *curt-e-sum-p isCont-p-nat that* **by** *simp*

lemma *curt-e-rec-smooth*: \$*e*-*x* = \$*p*-*x* ∗ (*1* + \$*e*-(*x*+*1*)) **if** *summable* ($\lambda k.$ \$*p*-{*k*+*1*&*x*}) *x*+*1* < \$$\psi$
using *curt-e-rec isCont-p-nat that* **by** *simp*

lemma *curt-e-sum-p-finite-smooth*: \$*e*-*x* = ($\sum k{<}n.$ \$*p*-{*k*+*1*&*x*}) **if** *x*+*n*+*1* > \$$\psi$ **for** *n*::*nat*
using *curt-e-sum-p-finite isCont-p-nat that* **by** *simp*

lemma *temp-curt-e-sum-p-smooth*: \$*e*-{*x*:*n*} = ($\sum k{<}n.$ \$*p*-{*k*+*1*&*x*}) **for** *n*::*nat*
using *temp-curt-e-sum-p isCont-p-nat* **by** *simp*

lemma *temp-curt-e-rec-smooth*: \$*e*-{*x*:*n*} = \$*p*-*x* ∗ (*1* + \$*e*-{*x*+*1*:*n*−*1*})
if *x*+*1* < \$$\psi$ *n* ≠ *0* **for** *n*::*nat*
using *temp-curt-e-rec isCont-p-nat that* **by** *simp*

end

end

5.3 Limited Survival Function

locale *limited-survival-function* = *survival-model* +
assumes *psi-limited*[*simp*]: \$$\psi < \infty$
begin

definition *ult-age* :: *nat* (‹\$$\omega$›)

where $\$\omega \equiv LEAST\ x{::}nat.\ ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ x = 0$
— the conventional notation for ultimate age

lemma *ccdfX-ceil-psi-0*: $ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ \lceil real\text{-}of\text{-}ereal\ \$\psi \rceil = 0$
proof −
have $real\text{-}of\text{-}ereal\ \$\psi \leq \lceil real\text{-}of\text{-}ereal\ \$\psi \rceil$ **by** *simp*
thus *?thesis* **using** *ccdfX-0-equiv psi-limited ccdfX-psi-0 le-ereal-le* **by** *presburger*
qed

lemma *ccdfX-omega-0*: $ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ \$\omega = 0$
unfolding *ult-age-def*
proof (*rule LeastI-ex*)
have $\lceil real\text{-}of\text{-}ereal\ \$\psi \rceil \geq 0$ **using** *psi-nonneg real-of-ereal-pos* **by** *fastforce*
thus $\exists\, x{::}nat.\ ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ (real\ x) = 0$
using *ccdfX-ceil-psi-0* **by** (*metis of-int-of-nat-eq zero-le-imp-eq-int*)
qed

corollary *psi-le-omega*: $\$\psi \leq \$\omega$
using *ccdfX-0-equiv ccdfX-omega-0* **by** *blast*

corollary *omega-pos*: $\$\omega > 0$
using *psi-le-omega order.strict-iff-not* **by** *fastforce*

lemma *omega-ceil-psi*: $\$\omega = \lceil real\text{-}of\text{-}ereal\ \$\psi \rceil$
proof (*rule antisym*)
let *?cpsi* $= \lceil real\text{-}of\text{-}ereal\ \$\psi \rceil$
have $\star$: *?cpsi* ≥ 0 **using** *psi-nonneg real-of-ereal-pos* **by** *fastforce*
moreover have $ccdf\ (distr\ \mathfrak{M}\ borel\ X)$ *?cpsi* $= 0$ **by** (*rule ccdfX-ceil-psi-0*)
ultimately have $\$\omega \leq nat$ *?cpsi*
unfolding *ult-age-def* **using** *wellorder-Least-lemma of-nat-nat* **by** (*smt* (*verit*))
thus *int* $\$\omega \leq$ *?cpsi* **using** *le-nat-iff* $\star$ **by** *blast*
next
show $\lceil real\text{-}of\text{-}ereal\ \$\psi \rceil \leq int\ \$\omega$
using *psi-le-omega* **by** (*simp add*: *ceiling-le-iff real-of-ereal-ord-simps*(*2*))
qed

lemma *ccdfX-0-equiv-nat*: $ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ x = 0 \longleftrightarrow x \geq \$\omega$ **for** $x{::}nat$
proof
assume $ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ (real\ x) = 0$
thus $x \geq \$\omega$ **unfolding** *ult-age-def* **using** *wellorder-Least-lemma* **by** *fastforce*
next
assume $x \geq \$\omega$
hence $ereal\ (real\ x) \geq \$\psi$ **using** *psi-le-omega le-ereal-le of-nat-mono* **by** *blast*
thus $ccdf\ (distr\ \mathfrak{M}\ borel\ X)\ (real\ x) = 0$ **using** *ccdfX-0-equiv* **by** *simp*
qed

lemma *psi-le-iff-omega-le*: $\$\psi \leq x \longleftrightarrow \$\omega \leq x$ **for** $x{::}nat$
using *ccdfX-0-equiv ccdfX-0-equiv-nat* **by** *presburger*

context
  fixes *x*::*nat*
  assumes *x-lt-omega*[*simp*]: $x < \$\omega$
begin

lemma *x-lt-psi*[*simp*]: $x < \$\psi$
  using *x-lt-omega psi-le-iff-omega-le* **by** (*meson linorder-not-less*)

lemma *p-0-1-nat*: $\$p\text{-}\{0\&x\} = 1$
  using *p-0-1* **by** *simp*

lemma *p-0-equiv-nat*: $\$p\text{-}\{t\&x\} = 0 \longleftrightarrow x+t \geq \$\omega$ **for** *t*::*nat*
  using *psi-le-iff-omega-le p-0-equiv* **by** (*metis of-nat-add x-lt-psi*)

lemma *q-0-0-nat*: $\$q\text{-}\{0\&x\} = 0$
  using *p-q-1 p-0-1-nat* **by** (*smt* (*verit*) *x-lt-psi*)

lemma *q-1-equiv-nat*: $\$q\text{-}\{t\&x\} = 1 \longleftrightarrow x+t \geq \$\omega$ **for** *t*::*nat*
  using *p-q-1 p-0-equiv-nat* **by** (*smt* (*verit*) *x-lt-psi*)

lemma *q-defer-old-0-nat*: $\$q\text{-}\{f|t\&x\} = 0$ **if** $\$\omega \leq x+f$ **for** *f t* :: *nat*
  using *q-defer-old-0 psi-le-iff-omega-le that* **by** (*metis of-nat-0-le-iff of-nat-add x-lt-psi*)

lemma *curt-e-sum-P-finite-nat*: $\$e\text{-}x = (\sum k{<}n.\ \mathcal{P}(\xi\ in\ \mathfrak{M}.\ T\ x\ \xi \geq k + 1 \mid T\ x\ \xi > 0))$
  if $x+n \geq \$\omega$ **for** *n*::*nat*
  apply (*rule curt-e-sum-P-finite*, *simp*)
  using *that psi-le-iff-omega-le* **by** (*smt* (*verit*) *le-ereal-less of-nat-add*)

lemma *curt-e-sum-p-finite-nat*: $\$e\text{-}x = (\sum k{<}n.\ \$p\text{-}\{k+1\&x\})$
  if $\bigwedge k$::*nat*. $k < n \Longrightarrow$ *isCont* ($\lambda t.\ \$p\text{-}\{t\&x\}$) (*real k* + *1*) $x+n \geq \$\omega$ **for** *n*::*nat*
  apply (*rule curt-e-sum-p-finite*, *simp-all add*: *that*)
  using *that psi-le-iff-omega-le* **by** (*smt* (*verit*) *le-ereal-less of-nat-add*)

end

lemma *q-omega-1*: $\$q\text{-}(\$\omega{-}1) = 1$
  using *q-1-equiv-nat*
  by (*metis diff-less order.refl le-diff-conv of-nat-1 omega-pos zero-less-one*)

end

end
theory *Life-Table*
  imports *Survival-Model*
begin

6 Life Table

Define a life table axiomatically.

locale *life-table* =
  fixes *l* :: *real* ⇒ *real* (‹$*l*′--› [*101*] *200*)
  assumes *l-0-pos*: *0* < *l 0*
    and *l-neg-nil*: ⋀*x*. *x* ≤ *0* ⟹ *l x* = *l 0*
    and *l-PInfty-0*: (*l* ⟶ *0*) *at-top*
    and *l-antimono*: *antimono l*
    and *l-right-continuous*: ⋀*x*. *continuous* (*at-right x*) *l*
begin

6.1 Basic Properties of Life Table

lemma *l-0-neq-0*[*simp*]: $*l-0* ≠ *0*
  using *l-0-pos* **by** *simp*

lemma *l-nonneg*[*simp*]: $*l-x* ≥ *0* **for** *x*::*real*
  using *l-antimono l-PInfty-0* **by** (*rule antimono-at-top-le*)

lemma *l-bounded*[*simp*]: $*l-x* ≤ $*l-0* **for** *x*::*real*
  using *l-neg-nil l-antimono* **by** (*smt* (*verit*) *antimonoD*)

lemma *l-measurable*[*measurable*, *simp*]: *l* ∈ *borel-measurable borel*
  by (*rule borel-measurable-antimono*, *rule l-antimono*)

lemma *l-left-continuous-nonpos*: *continuous* (*at-left x*) *l* **if** *x* ≤ *0* **for** *x*::*real*
proof −
  have $*l-x* = $*l-0* **using** *l-neg-nil that* **by** *blast*
  moreover have *continuous* (*at-left x*) (λ-. $*l-0*) **by** *simp*
  moreover have *eventually* (λ*y*. $*l-y* = $*l-0*) (*at-left x*)
    apply (*rule eventually-at-leftI*[*of x−1*], *simp-all*)
    using *that l-neg-nil* **by** (*smt* (*verit*))
  ultimately show *?thesis* **by** (*rewrite continuous-at-within-cong*[**where** *g*=λ-. $*l-0*]; *simp*)
qed

lemma *l-integrable-Icc*: *set-integrable lborel* {*a*..*b*} *l* **for** *a b* :: *real*
  unfolding *set-integrable-def*
  apply (*rule integrableI-bounded-set*[**where** *A*={*a*..*b*} **and** *B*=$*l-0*], *simp-all*)
  using *emeasure-compact-finite* **by** *auto*

corollary *l-integrable-on-Icc*: *l integrable-on* {*a*..*b*} **for** *a b* :: *real*
  using *l-integrable-Icc* **by** (*rewrite integrable-on-iff-set-integrable-nonneg*[*THEN sym*]; *simp*)

lemma *l-integrable-Icc-shift*: *set-integrable lborel* {*a*..*b*} (λ*t*. $*l-*(*x*+*t*)) **for** *a b x* :: *real*
  using *set-integrable-Icc-shift l-integrable-Icc* **by** (*metis* (*full-types*) *add-diff-cancel-right*′)

corollary *l-integrable-on-Icc-shift*: $(\lambda t.\ \$l\text{-}(x+t))$ *integrable-on* $\{a..b\}$ **for** *x a b* :: *real*
using *l-integrable-Icc-shift* **by** (*rewrite integrable-on-iff-set-integrable-nonneg*[*THEN sym*]; *simp*)

lemma *l-normal-antimono*: *antimono* $(\lambda x.\ \$l\text{-}x\ /\ \$l\text{-}0)$
using *divide-le-cancel l-0-pos l-antimono* **unfolding** *antimono-def* **by** *fastforce*

lemma *compl-l-normal-right-continuous*: *continuous* (*at-right x*) $(\lambda x.\ 1 - \$l\text{-}x\ /\ \$l\text{-}0)$ **for** *x*::*real*
using *l-0-pos l-right-continuous* **by** (*intro continuous-intros*; *simp*)

lemma *compl-l-normal-NInfty-0*: $((\lambda x.\ 1 - \$l\text{-}x\ /\ \$l\text{-}0) \longrightarrow 0)$ *at-bot*
apply (*rewrite tendsto-cong*[**where** *g*=λ-. *0*], *simp-all*)
by (*smt* (*verit*) *div-self eventually-at-bot-linorder l-0-pos l-neg-nil*)

lemma *compl-l-normal-PInfty-1*: $((\lambda x.\ 1 - \$l\text{-}x\ /\ \$l\text{-}0) \longrightarrow 1)$ *at-top*
using *l-0-pos l-PInfty-0* **by** (*intro tendsto-eq-intros*) *simp-all+*

lemma *compl-l-real-distribution*: *real-distribution* (*interval-measure* $(\lambda x.\ 1 - \$l\text{-}x\ /\ \$l\text{-}0)$)
using *l-normal-antimono compl-l-normal-right-continuous*
compl-l-normal-NInfty-0 compl-l-normal-PInfty-1
by (*intro real-distribution-interval-measure*; *simp add*: *antimono-def*)

definition *total* :: *real* ⇒ *real* (‹$T'--› [*101*] *200*) **where** $\$T\text{-}x \equiv LBINT\ y{:}\{x..\}.\ \$l\text{-}y$
— the number of lives older than the ones aged x
— The parameter x must be nonnegative.

lemma *T-nonneg*[*simp*]: $\$T\text{-}x \geq 0$ **for** *x*::*real*
unfolding *total-def* **by** *simp*

definition *total-finite* ≡ *set-integrable lborel* $\{0..\}$ *l*

lemma *total-finite-iff-set-integrable-Ici*:
total-finite ⟷ *set-integrable lborel* $\{x..\}$ *l* **for** *x*::*real*
unfolding *total-finite-def* **using** *set-integrable-Ici-equiv l-integrable-Icc* **by** *blast*

lemma *total-finite-iff-integrable-on-Ici*: *total-finite* ⟷ *l integrable-on* $\{x..\}$ **for** *x*::*real*
using *total-finite-iff-set-integrable-Ici integrable-on-iff-set-integrable-nonneg l-nonneg*
by (*metis atLeast-borel l-measurable measurable-lborel2 sets-lborel*)

lemma *total-finite-iff-summable*: *total-finite* ⟷ *summable* $(\lambda k.\ \$l\text{-}(x+k))$ **for** *x*::*real*
apply (*rewrite total-finite-iff-set-integrable-Ici*)
apply (*rule set-integrable-iff-summable*[*of x*, *simplified*], *simp-all*)
using *l-antimono* **unfolding** *antimono-def monotone-on-def* **by** *simp*

lemma *T-tendsto-0*: $((\lambda x.\ \$T\text{-}x) \longrightarrow 0)$ *at-top* **if** *total-finite*
proof −
have $\bigwedge x.\ x \geq 0 \Longrightarrow \$T\text{-}x = \$T\text{-}0 - (LBINT\ y{:}\{0..x\}.\ \$l\text{-}y)$
proof −
fix *x*::*real* **assume** *asm*: $x \geq 0$
let $?A = \{x..\}$ **and** $?B = \{0..x\}$
have $\{0..\} = ?A \cup ?B$ **using** *asm* **by** *auto*
thus $\$T\text{-}x = \$T\text{-}0 - (LBINT\ y{:}\{0..x\}.\ \$l\text{-}y)$
unfolding *total-def* **apply** (*rewrite eq-diff-eq*)
using *that total-finite-iff-set-integrable-Ici l-integrable-Icc*
apply (*rewrite set-integral-Un-AE*[*THEN sym*], *simp-all*)
using *AE-lborel-singleton add-0 asm le-add-same-cancel2 le-numeral-extra*(*3*)
by *force*
qed
hence $\forall_F\ x\ in\ at\text{-}top.\ \$T\text{-}x = \$T\text{-}0 - (LBINT\ y{:}\{0..x\}.\ \$l\text{-}y)$
by (*rule eventually-at-top-linorderI*[*of 0*])
moreover have $((\lambda x.\ LBINT\ y{:}\{0..x\}.\ \$l\text{-}y) \longrightarrow \$T\text{-}0)$ *at-top*
using *that* **unfolding** *total-def total-finite-def*
by (*intro tendsto-set-lebesgue-integral-at-top*; *simp*)
ultimately show *?thesis*
apply (*rewrite tendsto-cong*, *simp-all*)
using *LIM-zero-iff'* **by** *force*
qed

definition *lives* :: *real* ⇒ *real* ⇒ *real* (‹$L'-{-&-}› [*0*,*0*] *200*)
where $\$L\text{-}\{n\&x\} \equiv LBINT\ y{:}\{x..x+n\}.\ \$l\text{-}y$
— the number of lives between ages x and $x+n$
— The parameter x must be nonnegative.
— The parameter n is usually nonnegative, but theoretically it can be negative.

abbreviation *lives-1* :: *real* ⇒ *real* (‹$L'--› [*101*] *200*)
where $\$L\text{-}x \equiv \$L\text{-}\{1\&x\}$

lemma *l-has-integral-L*: (*l has-integral* $\$L\text{-}\{n\&x\}$) $\{x..x+n\}$ **for** *x n* :: *real*
unfolding *lives-def* **by** (*rule has-integral-set-integral-real*) (*rule l-integrable-Icc*)

lemma *L-neg-0*[*simp*]: $\$L\text{-}\{n\&x\} = 0$ **if** $n < 0$ **for** *x n* :: *real*
unfolding *lives-def* **using** *that* **by** (*rewrite to* {} *atLeastatMost-empty*; *simp*)

lemma *L-nonneg*[*simp*]: $\$L\text{-}\{n\&x\} \geq 0$ **for** *x n* :: *real*
unfolding *lives-def* **by** *simp*

lemma *L-T*: $\$L\text{-}\{n\&x\} = \$T\text{-}x - \$T\text{-}(x+n)$ **if** *total-finite* $n \geq 0$ **for** *x n* :: *real*
proof −
have $\{x..x+n\} \cup \{x+n..\} = \{x..\}$ **using** *that* **by** *force*
moreover have
$(LBINT\ y{:}\{x..x+n\} \cup \{x+n..\}.\ \$l\text{-}y) = (LBINT\ y{:}\{x..x+n\}.\ \$l\text{-}y) + (LBINT\ y{:}\{x+n..\}.\ \$l\text{-}y)$

proof −
have *AE y in lborel.* ¬ (*y* ∈ {*x*..*x*+*n*} ∧ *y* ∈ {*x*+*n*..}) **by** (*rule AE-I'*[**where** *N*={*x*+*n*}]; *force*)
moreover have *set-integrable lborel* {*x*..*x*+*n*} *l* **by** (*rule l-integrable-Icc*)
moreover have *set-integrable lborel* {*x*+*n*..} *l*
using *that total-finite-iff-set-integrable-Ici* **by** *simp*
ultimately show *?thesis* **by** (*intro set-integral-Un-AE*; *simp*)
qed
ultimately show *?thesis* **unfolding** *total-def lives-def* **by** *simp*
qed

lemma *L-sums-T*: (λ*k*. $*L*-(*x*+*k*)) *sums* $*T*-*x* **if** *total-finite* **for** *x*::*real*
proof −
have (λ*k*::*nat*. $*T*-(*x*+*k*)) ⟶ *0*
using *T-tendsto-0*
apply (*rule filterlim-compose*[**where** *f*=λ*k*::*nat*. *x*+*k* **and** *g*=*total*], *simp add*: *that*)
using *filterlim-real-sequentially filterlim-tendsto-add-at-top* **by** *blast*
hence (λ*k*. $*T*-(*x*+*k*) − $*T*-(*x* + *Suc k*)) *sums* $*T*-*x*
by (*simp*) (*rule telescope-sums'*[*of* λ*k*. $*T*-(*x*+*k*) *0*, *simplified*])
thus *?thesis* **using** *that L-T* **by** (*rewrite sums-cong*, *simp-all*) (*smt* (*verit*))
qed

definition *death* :: *real* ⇒ *real* ⇒ *real* (‹$*d*'-{-&-}› [*0*,*0*] *200*)
where $*d*-{*t*&*x*} ≡ *max 0* ($*l*-*x* − $*l*-(*x*+*t*))
— the number of deaths between ages *x* and *x*+*t*
— The parameter *t* is usually nonnegative, but theoretically it can be negative.

abbreviation *death1* :: *real* ⇒ *real* (‹$*d*'--› [*101*] *200*)
where $*d*-*x* ≡ $*d*-{*1*&*x*}

lemma *death-def-nonneg*: $*d*-{*t*&*x*} = $*l*-*x* − $*l*-(*x*+*t*) **if** *t* ≥ *0* **for** *t x* :: *real*
using *that l-antimono* **unfolding** *death-def antimono-def* **by** *simp*

lemma *d-nonpos-0*: $*d*-{*t*&*x*} = *0* **if** *t* ≤ *0* **for** *t x* :: *real*
using *that l-antimono* **unfolding** *death-def antimono-def* **by** *simp*

corollary *d-0-0*: $*d*-{*0*&*x*} = *0* **for** *x*::*real*
using *d-nonpos-0* **by** *simp*

lemma *d-nonneg*[*simp*]: $*d*-{*t*&*x*} ≥ *0* **for** *t x* :: *real*
unfolding *death-def* **by** *simp*

lemma *dx-l*: $*d*-*x* = $*l*-*x* − $*l*-(*x*+*1*) **for** *x*::*real*
using *death-def-nonneg* **by** *simp*

lemma *sum-dx-l*: (∑ *k*<*n*. $*d*-(*x*+*k*)) = $*l*-*x* − $*l*-(*x*+*n*) **for** *x*::*real* **and** *n*::*nat*
proof (*induction n*)
case *0*

thus *?case* **by** *simp*
next
case (*Suc n*)
thus *?case*
using *dx-l*
by (*metis Set-Interval.comm-monoid-add-class.sum.lessThan-Suc add-diff-cancel-left' diff-diff-eq2 of-nat-Suc*)
qed

corollary *d-sums-l*: (λ*k*. $*d*-(*x*+*k*)) *sums* $*l*-*x* **for** *x*::*real*
unfolding *sums-def*
apply (*rewrite sum-dx-l*)
apply (*rule tendsto-diff*[**where** *b*=*0*, *simplified*], *simp*)
using *l-PInfty-0 filterlim-compose filterlim-real-sequentially filterlim-tendsto-add-at-top tendsto-const* **by** *blast*

lemma *add-d*: $*d*-{*t*&*x*} + $*d*-{*t'* & *x*+*t*} = $*d*-{*t*+*t'* & *x*} **if** *t* ≥ *0* *t'* ≥ *0* **for** *t* *t'* :: *real*
using *death-def-nonneg that* **by** (*smt* (*verit*))

definition *die-central* :: *real* ⇒ *real* ⇒ *real* (‹$*m'*-{-&-}› [*0*,*0*] *200*)
where $*m*-{*n*&*x*} ≡ $*d*-{*n*&*x*} / $*L*-{*n*&*x*}
— central death rate

abbreviation *die-central-1* :: *real* ⇒ *real* (‹$*m'*--› [*101*] *200*)
where $*m*-*x* ≡ $*m*-{*1*&*x*}

6.2 Construction of Survival Model from Life Table

definition *life-table-measure* :: *real measure* (‹𝔐›)
where 𝔐 ≡ *interval-measure* (λ*x*. *1* − $*l*-*x* / $*l*-*0*)

lemma *prob-space-actuary-MM*: *prob-space-actuary* 𝔐
unfolding *life-table-measure-def* **using** *compl-l-real-distribution real-distribution-def*
by (*intro prob-space-actuary.intro*) *force*

definition *survival-model-X* :: *real* ⇒ *real* (‹*X*›) **where** *X* ≡ λ*x*. *x*

lemma *survival-model-MM-X*: *survival-model* 𝔐 *X*
proof −
let *?F* = λ*x*. *1* − $*l*-*x* / $*l*-*0*
show *survival-model* 𝔐 *X*
unfolding *life-table-measure-def survival-model-X-def*
proof (*rule survival-model.intro*)
show *prob-space-actuary* (*interval-measure ?F*)
using *prob-space-actuary-MM* **unfolding** *life-table-measure-def* **by** *simp*
show *survival-model-axioms* (*interval-measure ?F*) (λ*x*. *x*)
proof −
have [*simp*]: {ξ::*real*. ξ ≤ *0*} = {..*0*} **by** *blast*

```
    have measure (interval-measure (λx. 1 − $l-x / $l-0)) {..0} = 0
    using l-normal-antimono compl-l-normal-right-continuous compl-l-normal-NInfty-0
      by (rewrite measure-interval-measure-Iic, simp-all add: antimono-def)
    hence emeasure (interval-measure (λx. 1 − $l-x / $l-0)) {..0} = ennreal 0
      apply (rewrite finite-measure.emeasure-eq-measure, simp-all)
      using compl-l-real-distribution prob-space-def real-distribution-def by blast
    thus ?thesis
      apply (intro survival-model-axioms.intro, simp)
      apply (rewrite AE-iff-null, simp)
      by (rewrite not-less) auto
  qed
 qed
qed

end

sublocale life-table ⊆ survival-model 𝔐 X
 by (rule survival-model-MM-X)

context life-table
begin

interpretation distrX-RD: real-distribution distr 𝔐 borel X
 using MM-PS.real-distribution-distr by simp
```

6.2.1 Relations between Life Table and Survival Function for X

```
lemma ccdfX-l-normal: ccdf (distr 𝔐 borel X) = (λx. $l-x / $l-0)
proof (rule ext)
 let ?F = λx. 1 − $l-x / $l-0
 interpret F-FBM: finite-borel-measure interval-measure ?F
  using compl-l-real-distribution real-distribution.finite-borel-measure-M by blast
 show ⋀x. ccdf (distr 𝔐 borel X) x = $l-x / $l-0
  unfolding ccdf-def life-table-measure-def survival-model-X-def
  apply (rewrite measure-distr, simp-all)
  using l-normal-antimono compl-l-normal-right-continuous
    compl-l-normal-NInfty-0 compl-l-normal-PInfty-1
  by (rewrite F-FBM.measure-interval-measure-Ioi; simp add: antimono-def)
qed

corollary deriv-ccdfX-l: deriv (ccdf (distr 𝔐 borel X)) x = deriv l x / $l-0
 if l differentiable at x for x::real
 using differentiable-eq-field-differentiable-real that
 by (rewrite ccdfX-l-normal, rewrite deriv-cdivide-right; simp)

notation death-pt (‹$ψ›)

lemma l-0-equiv: $l-x = 0 ⟷ x ≥ $ψ for x::real
 using ccdfX-l-normal ccdfX-0-equiv by simp
```

lemma *d-old-0*: $\$d\text{-}\{t\&x\} = 0$ **if** $x \geq \$\psi\ t \geq 0$ **for** $x\ t :: real$
unfolding *death-def* **using** *l-0-equiv that* **by** (*smt* (*verit*) *le-ereal-le*)

lemma *d-l-equiv*: $\$d\text{-}\{t\&x\} = \$l\text{-}x \longleftrightarrow x+t \geq \$\psi$ **if** $t \geq 0$ **for** $x\ t :: real$
using *death-def-nonneg l-0-equiv that* **by** *simp*

lemma *continuous-ccdfX-l*: *continuous F* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$ *continuous F l*
for *F* :: *real filter*
proof −
have *continuous F* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$ *continuous F* ($\lambda x.\ \$l\text{-}x\ /\ \$l\text{-}0$)
using *ccdfX-l-normal* **by** *simp*
also have ... $\longleftrightarrow$ *continuous F l* **using** *continuous-cdivide-iff l-0-neq-0* **by** *blast*
finally show *?thesis* **.**
qed

lemma *has-real-derivative-ccdfX-l*:
(*ccdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative D*) (*at x*) $\longleftrightarrow$
(*l has-real-derivative* $\$l\text{-}0 * D$) (*at x*)
for *D x* :: *real*
proof −
have (*ccdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative D*) (*at x*) $\longleftrightarrow$
(($\lambda x.\ \$l\text{-}x\ /\ \$l\text{-}0$) *has-real-derivative D*) (*at x*)
by (*rule has-field-derivative-cong-eventually*; *simp add*: *ccdfX-l-normal*)
also have ... $\longleftrightarrow$ (($\lambda x.\ \$l\text{-}0 * (\$l\text{-}x\ /\ \$l\text{-}0)$) *has-real-derivative* $\$l\text{-}0 * D$) (*at x*)
by (*rule DERIV-cmult-iff*, *simp*)
also have ... $\longleftrightarrow$ (*l has-real-derivative* $\$l\text{-}0 * D$) (*at x*) **by** *simp*
finally show *?thesis* **.**
qed

corollary *differentiable-ccdfX-l*:
ccdf (*distr* $\mathfrak{M}$ *borel X*) *differentiable* (*at x*) $\longleftrightarrow$ *l differentiable* (*at x*)
for *D x* :: *real*
using *has-real-derivative-ccdfX-l*
by (*metis l-0-neq-0 mult.commute nonzero-divide-eq-eq real-differentiable-def*)

lemma *PX-l-normal*: $\mathcal{P}(\xi\ in\ \mathfrak{M}.\ X\ \xi > x) = \$l\text{-}x\ /\ \$l\text{-}0$ **for** *x*::*real*
using *MM-PS.ccdf-distr-P ccdfX-l-normal X-RV* **by** (*metis* (*mono-tags*, *lifting*) *Collect-cong*)

lemma *set-integrable-ccdfX-l*:
set-integrable lborel A (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$ *set-integrable lborel A l*
if $A \in sets\ lborel$ **for** *A* :: *real set*
proof −
have *set-integrable lborel A* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$
set-integrable lborel A ($\lambda x.\ \$l\text{-}x\ /\ \$l\text{-}0$)
by (*rule set-integrable-cong*; *simp add*: *ccdfX-l-normal*)
also have ... $\longleftrightarrow$ *set-integrable lborel A l* **by** *simp*

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

corollary *integrable-ccdfX-l*: *integrable lborel* (*ccdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$ *integrable lborel l*
using *set-integrable-ccdfX-l*[**where** *A=UNIV*] **by** (*simp add*: *set-integrable-def*)

lemma *integrable-on-ccdfX-l*:
ccdf (*distr* $\mathfrak{M}$ *borel X*) *integrable-on A* $\longleftrightarrow$ *l integrable-on A* **for** *A* :: *real set*
proof −
have *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *integrable-on A* $\longleftrightarrow$ (λx. $l-x / $l-0) *integrable-on A*
by (*rule integrable-cong*) (*simp add*: *ccdfX-l-normal*)
also have ... $\longleftrightarrow$ *l integrable-on A*
using *integrable-on-cdivide-iff*[*of* $l-0 *l*] **by** *simp*
finally show *?thesis* **.**
qed

6.2.2 Relations between Life Table and Cumulative Distributive Function for X

lemma *cdfX-l-normal*: *cdf* (*distr* $\mathfrak{M}$ *borel X*) = (λx. *1* − $l-x / $l-0) **for** *x*::*real*
using *ccdfX-l-normal distrX-RD.cdf-ccdf distrX-RD.prob-space* **by** *presburger*

lemma *deriv-cdfX-l*: *deriv* (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) *x* = − *deriv l x* / $l-0
if *l differentiable at x* **for** *x*::*real*
using *distrX-RD.cdf-ccdf differentiable-eq-field-differentiable-real that differentiable-ccdfX-l*
deriv-diff deriv-ccdfX-l that **by** *simp*

lemma *continuous-cdfX-l*: *continuous F* (*cdf* (*distr* $\mathfrak{M}$ *borel X*)) $\longleftrightarrow$ *continuous F l*
for *F* :: *real filter*
using *distrX-RD.continuous-cdf-ccdf continuous-ccdfX-l* **by** *simp*

lemma *has-real-derivative-cdfX-l*:
(*cdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative D*) (*at x*) $\longleftrightarrow$
(*l has-real-derivative* − ($l-0 ∗ *D*)) (*at x*)
for *D x* :: *real*
using *distrX-RD.has-real-derivative-cdf-ccdf has-real-derivative-ccdfX-l* **by** *simp*

lemma *differentiable-cdfX-l*:
cdf (*distr* $\mathfrak{M}$ *borel X*) *differentiable* (*at x*) $\longleftrightarrow$ *l differentiable* (*at x*) **for** *D x* :: *real*
using *differentiable-eq-field-differentiable-real distrX-RD.differentiable-cdf-ccdf differentiable-ccdfX-l* **by** *simp*

lemma *PX-compl-l-normal*: $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X* $\xi \leq x$) = *1* − $l-x / $l-0 **for** *x*::*real*
using *PX-l-normal* **by** (*metis MM-PS.prob-compl X-compl-gt-le X-gt-event*)

6.2.3 Relations between Life Table and Survival Function for $T(x)$

context
  fixes *x*::*real*
  assumes *x-lt-psi*[*simp*]: $x < \$\psi$
begin

notation *futr-life* (‹ *T* ›)

interpretation *alivex-PS*: *prob-space* $\mathfrak{M} \restriction$ *alive x*
  by (*rule MM-PS.cond-prob-space-correct*, *simp-all add*: *alive-def*)

interpretation *distrTx-RD*: *real-distribution distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*) **by** *simp*

lemma *lx-neq-0*[*simp*]: $\$l\text{-}x \neq 0$
  using *l-0-equiv x-lt-psi linorder-not-less* **by** *blast*

corollary *lx-pos*[*simp*]: $\$l\text{-}x > 0$
  using *lx-neq-0 l-nonneg* **by** (*smt* (*verit*))

lemma *ccdfTx-l-normal*: *ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *t* = $\$l\text{-}(x+t)$ / $\$l\text{-}x$
  if $t \geq 0$ **for** *t*::*real*
  using *ccdfTx-PX PX-l-normal l-0-neq-0 that* **by** *simp*

lemma *deriv-ccdfTx-l*:
  deriv (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*))) *t* = *deriv* ($\lambda t.$ $\$l\text{-}(x+t)$ / $\$l\text{-}x$) *t*
  if $t > 0$ *l differentiable at* ($x+t$) **for** *t*::*real*
proof −
  have $\forall_F$ *s in nhds t. ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *s* = $\$l\text{-}(x+s)$ / $\$l\text{-}x$
    apply (*rewrite eventually-nhds-metric*)
    using *that ccdfTx-l-normal dist-real-def* **by** (*intro exI*[*of* - *t*]) *auto*
  thus *?thesis* **by** (*rule deriv-cong-ev*) *simp*
qed

lemma *continuous-at-within-ccdfTx-l*:
  continuous (*at t within* {*0*..}) (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*))) ⟷
    continuous (*at* ($x+t$) *within* {*x*..}) *l*
  if $t \geq 0$ **for** *t*::*real*
  using *continuous-ccdfX-ccdfTx that continuous-ccdfX-l* **by** *force*

lemma *isCont-ccdfTx-l*:
  isCont (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*))) *t* ⟷ *isCont l* ($x+t$) **if** $t > 0$ **for** *t*::*real*
  using *that continuous-ccdfX-l isCont-ccdfX-ccdfTx* **by** *force*

lemma *has-real-derivative-ccdfTx-l*:
  (*ccdf* (*distr* ($\mathfrak{M} \restriction$ *alive x*) *borel* (*T x*)) *has-real-derivative D*) (*at t*) ⟷
    (*l has-real-derivative* $\$l\text{-}x * D$) (*at* ($x+t$))

if $t > 0$ **for** $t\ D$ :: *real*
proof −
have (*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative D*) (*at t*) ⟷
(*ccdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative*
($l-x / $l-0 * D / $\mathcal{P}$(ξ *in* $\mathfrak{M}$. *X ξ* > *x*))) (*at t*)
using *PX-l-normal* **by** *force*
also have ... = (*ccdf* (*distr* $\mathfrak{M}$ *borel X*) *has-real-derivative* ($l-x / $l-0 * D)) (*at* (*x*+*t*))
using *has-real-derivative-ccdfX-ccdfTx that* **by** *simp*
also have ... = (*l has-real-derivative* ($l-x * D)) (*at* (*x*+*t*))
using *has-real-derivative-ccdfX-l* **by** *simp*
finally show *?thesis* **.**
qed

lemma *differentiable-ccdfTx-l*:
ccdf (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t* ⟷ *l differentiable* (*at* (*x*+*t*))
if $t > 0$ **for** *t*::*real*
using *differentiable-ccdfX-ccdfTx differentiable-ccdfX-l that* **by** *force*

lemma *PTx-l-normal*: $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *T x ξ* > *t* | *T x ξ* > 0) = $l-(*x*+*t*) / $l-x **if** $t \geq 0$ **for** *t*::*real*
using *ccdfTx-l-normal that* **by** (*simp add*: *ccdfTx-cond-prob*)

6.2.4 Relations between Life Table and Cumulative Distributive Function for $T(x)$

lemma *cdfTx-compl-l-normal*: *cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *t* = 1 − $l-(*x*+*t*) / $l-x
if $t \geq 0$ **for** *t*::*real*
using *distrTx-RD.cdf-ccdf ccdfTx-l-normal that distrTx-RD.prob-space* **by** *auto*

lemma *deriv-cdfTx-l*:
deriv (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *t* = − *deriv* (λ*t*. $l-(*x*+*t*) / $l-x) *t*
if $t > 0$ *l differentiable at* (*x*+*t*) **for** *t*::*real*
using *deriv-ccdfTx-l differentiable-cdfX-cdfTx differentiable-cdfX-l distrTx-RD.deriv-cdf-ccdf that* **by** *fastforce*

lemma *continuous-at-within-cdfTx-l*:
continuous (*at t within* {0..}) (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) ⟷
continuous (*at* (*x*+*t*) *within* {*x*..}) *l*
if $t \geq 0$ **for** *t*::*real*
using *that continuous-cdfX-l continuous-cdfX-cdfTx* **by** *force*

lemma *isCont-cdfTx-l*:
isCont (*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*))) *t* ⟷ *isCont l* (*x*+*t*) **if** $t > 0$ **for** *t*::*real*
using *that continuous-cdfX-l isCont-cdfX-cdfTx* **by** *force*

lemma *has-real-derivative-cdfTx-l*:
(*cdf* (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *has-real-derivative D*) (*at t*) ⟷
(*l has-real-derivative* − $*l-x* ∗ *D*) (*at* (*x*+*t*))
if *t* > *0* **for** *t D* :: *real*
using *has-real-derivative-ccdfTx-l that distrTx-RD.has-real-derivative-cdf-ccdf* **by** *auto*

lemma *differentiable-cdfTx-l*:
cdf (*distr* ($\mathfrak{M}$ ⌊ *alive x*) *borel* (*T x*)) *differentiable at t* ⟷ *l differentiable* (*at* (*x*+*t*))
if *t* > *0* **for** *t*::*real*
using *differentiable-cdfX-l that differentiable-cdfX-cdfTx* **by** *auto*

lemma *PTx-compl-l-normal*: $\mathcal{P}$(*ξ in* $\mathfrak{M}$. *T x ξ* ≤ *t* | *T x ξ* > *0*) = *1*− $*l*-(*x*+*t*) / $*l-x*
if *t* ≥ *0* **for** *t*::*real*
using *cdfTx-compl-l-normal that* **by** (*simp add*: *cdfTx-cond-prob*)

6.2.5 Life Table and Actuarial Notations

notation *survive* (‹$*p*′-{-&-}› [*0*,*0*] *200*)
notation *survive-1* (‹$*p*′--› [*101*] *200*)
notation *die* (‹$*q*′-{-&-}› [*0*,*0*] *200*)
notation *die-1* (‹$*q*′--› [*101*] *200*)
notation *die-defer* (‹$*q*′-{-|-&-}› [*0*,*0*,*0*] *200*)
notation *die-defer-1* (‹$*q*′-{-|&-}› [*0*,*0*] *200*)
notation *life-expect* (‹$*e*‘∘′--› [*101*] *200*)
notation *temp-life-expect* (‹$*e*‘∘′-{-:-}› [*0*,*0*] *200*)
notation *curt-life-expect* (‹$*e*′--› [*101*] *200*)
notation *temp-curt-life-expect* (‹$*e*′-{-:-}› [*0*,*0*] *200*)

lemma *p-l*: $*p*-{*t*&*x*} = $*l*-(*x*+*t*) / $*l-x* **if** *t* ≥ *0* **for** *t*::*real*
unfolding *survive-def* **using** *ccdfTx-l-normal that* **by** *simp*

corollary *p-1-l*: $*p-x* = $*l*-(*x*+*1*) / $*l-x*
using *p-l* **by** *simp*

lemma *isCont-p-l*: *isCont* (λ*s*. $*p*-{*s*&*x*}) *t* ⟷ *isCont l* (*x*+*t*) **if** *t* > *0* **for** *t*::*real*
proof −
have ∀ *F s in nhds t*. $*p*-{*s*&*x*} = $*l*-(*x*+*s*) / $*l-x*
apply (*rewrite eventually-nhds-metric*)
apply (*rule exI*[*of* - *t*], *auto simp add*: *that*)
by (*rewrite p-l*; *simp add*: *dist-real-def*)
hence *isCont* (λ*s*. $*p*-{*s*&*x*}) *t* ⟷ *isCont* (λ*s*. $*l*-(*x*+*s*) / $*l-x*) *t* **by** (*rule isCont-cong*)
also have ... ⟷ *isCont* (λ*s*. $*l*-(*x*+*s*)) *t* **using** *continuous-cdivide-iff lx-neq-0* **by** *metis*
also have ... ⟷ *isCont l* (*x*+*t*) **using** *isCont-shift* **by** (*force simp add*: *add.commute*)

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

lemma *total-finite-iff-p-set-integrable-Ici*:
total-finite $\longleftrightarrow$ *set-integrable lborel* $\{0..\}$ (λt. \$*p*-$\{t\&x\}$)
apply (*rewrite set-integrable-cong-AE*[**where** *g*=λt. \$*l*-$(x+t)$ / \$*l*-*x*]; *simp*)
using *survive-def* **apply** *simp*
using *p-l* **apply** (*intro AE-I2*, *simp*)
by (*metis l-integrable-Icc-shift set-integrable-Ici-equiv set-integrable-Ici-shift total-finite-iff-set-integrable-Ici*)

lemma *p-PTx-ge-l-isCont*: \$*p*-$\{t\&x\} = \mathcal{P}(\xi$ *in* $\mathfrak{M}$. $T\ x\ \xi \geq t \mid T\ x\ \xi > 0)$
if *isCont l* $(x+t)$ $t > 0$ **for** *t*::*real*
using *p-PTx-ge-ccdf-isCont that continuous-ccdfX-l* **by** *force*

lemma *q-defer-l*: \$*q*-$\{f|t\&x\}$ = (\$*l*-$(x+f)$ − \$*l*-$(x+f+t)$) / \$*l*-*x* **if** $f \geq 0$ $t \geq 0$ **for** *f t* :: *real*
apply (*rewrite q-defer-p*, *simp-all add*: *that*)
using *that* **by** (*rewrite p-l*, *simp*)+ (*smt* (*verit*) *diff-divide-distrib*)

corollary *q-defer-d-l*: \$*q*-$\{f|t\&x\}$ = \$*d*-$\{t$ & $x+f\}$ / \$*l*-*x* **if** $f \geq 0$ $t \geq 0$ **for** *f t* :: *real*
using *q-defer-l that death-def-nonneg* **by** *simp*

corollary *q-defer-1-d-l*: \$*q*-$\{f|\&x\}$ = \$*d*-$(x+f)$ / \$*l*-*x* **if** $f \geq 0$ **for** *f*::*real*
using *q-defer-d-l that* **by** *simp*

lemma *q-d-l*: \$*q*-$\{t\&x\}$ = \$*d*-$\{t\&x\}$ / \$*l*-*x* **for** *t*::*real*
proof (*cases* ‹$t \geq 0$›)
case *True*
thus *?thesis* **using** *q-defer-d-l*[*of 0*] **by** *simp*
next
case *False*
thus *?thesis* **using** *q-nonpos-0 d-nonpos-0* **by** *simp*
qed

corollary *q-1-d-l*: \$*q*-*x* = \$*d*-*x* / \$*l*-*x*
using *q-d-l* **by** *simp*

lemma *LBINT-p-l*: (*LBINT t*:*A*. \$*p*-$\{t\&x\}$) = (*LBINT t*:*A*. \$*l*-$(x+t)$) / \$*l*-*x*
if $A \subseteq \{0..\}$ $A \in$ *sets lborel* **for** *A* :: *real set*
— Note that $0 = 0$ holds when the integral diverges.
proof −
have [*simp*]: $\bigwedge t.\ t \in A \Longrightarrow$ \$*p*-$\{t\&x\}$ = \$*l*-$(x+t)$ / \$*l*-*x* **using** *p-l that* **by** *blast*
hence (*LBINT t*:*A*. \$*p*-$\{t\&x\}$) = (*LBINT t*:*A*. \$*l*-$(x+t)$ / \$*l*-*x*)
using *that* **by** (*rewrite set-lebesgue-integral-cong*[**where** *g*=λt. \$*l*-$(x+t)$ / \$*l*-*x*]; *simp*)
also have ... = (*LBINT t*:*A*. \$*l*-$(x+t)$) / \$*l*-*x* **by** (*rewrite set-integral-divide-zero*) *simp*

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

corollary *e-LBINT-l*: $*e*‘∘-*x* = (*LBINT t*:{*0*..}. $*l*-(*x*+*t*)) / $*l*-*x*
— Note that *0* = *0* holds when the integral diverges.
by (*simp add*: *e-LBINT-p LBINT-p-l*)

corollary *e-LBINT-l-Icc*: $*e*‘∘-*x* = (*LBINT t*:{*0*..*n*}. $*l*-(*x*+*t*)) / $*l*-*x* **if** *x*+*n* ≥ $*ψ* **for** *n*::*real*
using *e-LBINT-p-Icc* **by** (*rewrite LBINT-p-l*[*THEN sym*]; *simp add*: *that*)

lemma *temp-e-LBINT-l*: $*e*‘∘-{*x*:*n*} = (*LBINT t*:{*0*..*n*}. $*l*-(*x*+*t*)) / $*l*-*x* **if** *n* ≥ *0* **for** *n*::*real*
using *temp-e-LBINT-p* **by** (*rewrite LBINT-p-l*[*THEN sym*]; *simp add*: *that*)

lemma *integral-p-l*: *integral A* (λ*t*. $*p*-{*t*&*x*}) = (*integral A* (λ*t*. $*l*-(*x*+*t*))) / $*l*-*x*
if *A* ⊆ {*0*..} *A* ∈ *sets lborel* **for** *A* :: *real set*
— Note that *0* = *0* holds when the integral diverges.
using *that* **apply** (*rewrite set-borel-integral-eq-integral-nonneg*[*THEN sym*], *simp-all*)
apply (*simp add*: *survive-def*)
apply (*rewrite set-borel-integral-eq-integral-nonneg*[*THEN sym*], *simp-all*)
by (*rule LBINT-p-l*; *simp*)

corollary *e-integral-l*: $*e*‘∘-*x* = *integral* {*0*..} (λ*t*. $*l*-(*x*+*t*)) / $*l*-*x*
— Note that *0* = *0* holds when the integral diverges.
by (*simp add*: *e-integral-p integral-p-l*)

corollary *e-integral-l-Icc*:
$*e*‘∘-*x* = *integral* {*0*..*n*} (λ*t*. $*l*-(*x*+*t*)) / $*l*-*x* **if** *x*+*n* ≥ $*ψ* **for** *n*::*real*
using *e-integral-p-Icc* **by** (*rewrite integral-p-l*[*THEN sym*]; *simp add*: *that*)

lemma *e-pos-total-finite*: $*e*‘∘-*x* > *0* **if** *total-finite*
using *e-pos total-finite-iff-p-set-integrable-Ici that* **by** *simp*

lemma *temp-e-integral-l*:
$*e*‘∘-{*x*:*n*} = *integral* {*0*..*n*} (λ*t*. $*l*-(*x*+*t*)) / $*l*-*x* **if** *n* ≥ *0* **for** *n*::*real*
using *temp-e-integral-p* **by** (*rewrite integral-p-l*[*THEN sym*]; *simp add*: *that*)

lemma *curt-e-sum-l*: $*e*-*x* = (∑ *k*. $*l*-(*x*+*k*+*1*)) / $*l*-*x* **if** *total-finite* ⋀*k*::*nat*. *is-Cont l* (*x*+*k*+*1*)
proof −
have *summable* (λ*k*. $*l*-(*x*+(*k*+*1*::*nat*)))
using *that total-finite-iff-summable* **by** (*rewrite summable-iff-shift*[*of* λ*k*. $*l*-(*x*+*k*) *1*]) *simp*
moreover hence *summable* (λ*k*. $*p*-{*k*+*1*&*x*}) **by** (*rewrite p-l*, *simp-all add*: *add.commute*)
moreover have ⋀*k*::*nat*. *isCont* (λ*t*. $*p*-{*t*&*x*}) (*k*+*1*)
using *isCont-p-l that* **by** (*simp add*: *add.assoc*)
ultimately show *?thesis*

apply (*rewrite curt-e-sum-p*, *simp-all*)
apply (*rewrite p-l*, *simp*)
by (*rewrite suminf-divide*) (*simp add*: *add.commute*, *simp add*: *add.assoc*)
qed

lemma *curt-e-sum-l-finite*: $*e-x* = (∑ k<n. $*l-*(*x*+*k*+*1*)) / $*l-x*
if ⋀*k*::*nat*. *k* < *n* ⟹ *isCont l* (*x*+*k*+*1*) *x*+*n*+*1* > $*ψ* **for** *n*::*nat*
apply (*rewrite curt-e-sum-p-finite*[*of x n*], *simp-all add*: *that*)
using *isCont-p-l that* **apply** (*simp add*: *add.assoc*)
apply (*rewrite sum-divide-distrib*, *rule sum.cong*, *simp*)
using *p-l* **by** (*smt* (*verit*) *of-nat-0-le-iff*)

lemma *temp-curt-e-sum-p*: $*e-*{*x*:*n*} = (∑ k<n. $*l-*(*x*+*k*+*1*)) / $*l-x*
if ⋀*k*::*nat*. *k* < *n* ⟹ *isCont l* (*x*+*k*+*1*) **for** *n*::*nat*
apply (*rewrite temp-curt-e-sum-p*[*of x n*], *simp-all add*: *that*)
using *isCont-p-l that* **apply** (*simp add*: *add.assoc*)
apply (*rewrite sum-divide-distrib*, *rule sum.cong*, *simp*)
using *p-l* **by** (*smt* (*verit*) *of-nat-0-le-iff*)

lemma *e-T-l*: $*e‘∘-x* = $*T-x* / $*l-x*
unfolding *total-def*
apply (*rewrite e-LBINT-l*, *simp-all*)
by (*metis add-cancel-left-left diff-add-cancel lborel-set-integral-Ici-shift*)

lemma *temp-e-L-l*: $*e‘∘-*{*x*:*n*} = $*L-*{*n*&*x*} / $*l-x* **if** *n* ≥ *0* **for** *n*::*real*
unfolding *lives-def* **using** *that*
apply (*rewrite temp-e-LBINT-l*, *simp-all*)
using *diff-self add-diff-cancel-left′ lborel-set-integral-Icc-shift* **by** *metis*

lemma *m-q-e*: $*m-*{*n*&*x*} = $*q-*{*n*&*x*} / $*e‘∘-*{*x*:*n*} **if** *n* ≥ *0* **for** *n*::*real*
proof −
have $*m-*{*n*&*x*} = ($*d-*{*n*&*x*} / $*l-x*) / ($*L-*{*n*&*x*} / $*l-x*) **unfolding** *die-central-def*
by *simp*
thus *?thesis* **using** *q-d-l temp-e-L-l that* **by** *simp*
qed

end

lemma *l-p*: $*l-x* / $*l-0* = $*p-*{*x*&*0*} **for** *x*::*real*
using *ccdfX-l-normal ccdfX-p* **by** *force*

lemma *e-p-e-total-finite*: $*e‘∘-x* = $*e‘∘-*{*x*:*n*} + $*p-*{*n*&*x*} ∗ $*e‘∘-*(*x*+*n*)
if *total-finite n* ≥ *0 x*+*n* < $*ψ* **for** *x n* :: *real*
using *e-p-e that total-finite-iff-p-set-integrable-Ici* **by** (*smt* (*verit*) *ereal-less-le*)

proposition *x-ex-const-equiv-total-finite*: *x* + $*e‘∘-x* = *y* + $*e‘∘-y* ⟷ $*q-*{*y*−*x*&*x*} = *0*
if *total-finite x* ≤ *y y* < $*ψ* **for** *x y* :: *real*
using *x-ex-const-equiv that total-finite-iff-p-set-integrable-Ici p-set-integrable-shift*

by *blast*

corollary *x-ex-const-iff-l-const*: $x + \$e`\circ\text{-}x = y + \$e`\circ\text{-}y \longleftrightarrow \$l\text{-}x = \$l\text{-}y$
if *total-finite* $x \leq y$ $y < \$\psi$ **for** x y :: *real*
using *x-ex-const-equiv-total-finite that*
by (*smt* (*verit*, *ccfv-threshold*) *divide-cancel-right ereal-less-le l-0-equiv life-table.death-def-nonneg life-table.q-d-l life-table-axioms q-1-equiv*)

end

6.3 Piecewise Differentiable Life Table

locale *smooth-life-table* = *life-table* +
assumes *l-piecewise-differentiable*[*simp*]: *l piecewise-differentiable-on UNIV*
begin

lemma *smooth-survival-function-MM-X*: *smooth-survival-function* $\mathfrak{M}$ *X*
proof (*rule smooth-survival-function.intro*)
show *survival-model* $\mathfrak{M}$ *X* **by** (*rule survival-model-axioms*)
show *smooth-survival-function-axioms* $\mathfrak{M}$ *X*
proof
show *ccdf* (*distr* $\mathfrak{M}$ *borel X*) *piecewise-differentiable-on UNIV*
apply (*rewrite ccdfX-l-normal*)
apply (*rewrite divide-inverse*, *rewrite mult.commute*)
using *l-piecewise-differentiable piecewise-differentiable-scaleR*[*of l*] **by** *simp*
qed
qed

end

sublocale *smooth-life-table* ⊆ *smooth-survival-function* $\mathfrak{M}$ *X*
by (*rule smooth-survival-function-MM-X*)

context *smooth-life-table*
begin

notation *force-mortal* (‹μ'--› [*101*] *200*)

lemma *l-continuous*[*simp*]: *continuous-on UNIV l*
using *l-piecewise-differentiable piecewise-differentiable-on-imp-continuous-on* **by** *fastforce*

lemma *l-nondifferentiable-finite-set*[*simp*]: *finite* {*x*. ¬ *l differentiable at x*}
using *differentiable-ccdfX-l ccdfX-nondifferentiable-finite-set* **by** *simp*

lemma *l-differentiable-borel-set*[*measurable*, *simp*]: {*x*. *l differentiable at x*} ∈ *sets borel*
using *differentiable-ccdfX-l ccdfX-differentiable-borel-set* **by** *simp*

lemma *l-differentiable-AE*: *AE x in lborel. l differentiable at x*
using *differentiable-ccdfX-l ccdfX-differentiable-AE* **by** *simp*

lemma *deriv-l-measurable*[*measurable*]: *deriv l* ∈ *borel-measurable borel*
proof −
let *?S* = {*x*. ¬ *l differentiable at x*}
have ⋀*x*. *x* ∉ *?S* ⟹ $*l-0* ∗ *deriv* (*ccdf* (*distr* 𝔐 *borel X*)) *x* = *deriv l x*
using *deriv-ccdfX-l* **by** *simp*
thus *?thesis*
apply −
by (*rule measurable-discrete-difference*
[**where** *X*=*?S* **and** *f*=λ*x*. $*l-0* ∗ *deriv* (*ccdf* (*distr* 𝔐 *borel X*)) *x*])
(*simp-all add*: *countable-finite*)
qed

lemma *pdfX-l-normal*:
pdfX x = (*if l differentiable at x then* − *deriv l x* / $*l-0 else 0*) **for** *x*::*real*
unfolding *pdfX-def*
using *differentiable-eq-field-differentiable-real differentiable-cdfX-l deriv-cdfX-l* **by** *simp*

lemma *mu-deriv-l*: $*μ-x* = − *deriv l x* / $*l-x* **if** *l differentiable at x* **for** *x*::*real*
using *mu-pdfX that ccdfX-l-normal that pdfX-l-normal* **by** (*simp add*: *differentiable-cdfX-l*)

lemma *mu-nonneg-differentiable-l*: $*μ-x* ≥ *0* **if** *l differentiable at x* **for** *x*::*real*
using *differentiable-cdfX-l mu-nonneg-differentiable that* **by** *simp*

lemma *mu-deriv-ln-l*:
$*μ-x* = − *deriv* (λ*x*. *ln* ($*l-x*)) *x* **if** *l differentiable at x x* < $*ψ* **for** *x*::*real*
proof −
have ∀ *F x in nhds x*. *ln* ($*l-x* / $*l-0*) = *ln* ($*l-x*) − *ln* ($*l-0*)
proof (*cases* ‹$*ψ* < ∞›)
case *True*
thus *?thesis*
apply (*rewrite eventually-nhds-metric*)
apply (*intro exI*[*of* - *real-of-ereal* $*ψ* − *x*], *auto*)
using *that True not-inftyI* **apply** *fastforce*
apply (*rewrite ln-div*, *simp-all*)
using *lx-pos dist-real-def not-inftyI that*(*2*) **by** *fastforce*
next
case *False*
hence ⋀*x*. $*l-x* > *0* **using** *l-0-equiv* **by** *force*
thus *?thesis*
by (*simp add*: *ln-divide-pos*)
qed
hence *deriv* (λ*x*. *ln* ($*l-x* / $*l-0*)) *x* = *deriv* (λ*x*. *ln* ($*l-x*)) *x*
apply (*rewrite deriv-cong-ev*[*of* - λ*x*. *ln* ($*l-x*) − *ln* ($*l-0*)], *simp-all*)
apply (*rewrite deriv-diff*, *simp-all*)

unfolding *field-differentiable-def* **using** *that*
by (*metis DERIV-ln-divide-chain lx-pos real-differentiable-def*)
thus *?thesis* **using** *ccdfX-l-normal mu-deriv-ln that differentiable-ccdfX-l* **by** *force*
qed

lemma *deriv-l-shift*: *deriv l* (*x*+*t*) = *deriv* (λ*t*. $*l*-(*x*+*t*)) *t*
if *l differentiable at* (*x*+*t*) **for** *x t* :: *real*
using *deriv-shift differentiable-eq-field-differentiable-real that* **by** *simp*

context
fixes *x*::*real*
assumes *x-lt-psi*[*simp*]: *x* < $ψ
begin

lemma *p-mu-l*: $*p*-{*t*&*x*} ∗ $*μ*-(*x*+*t*) = − *deriv l* (*x*+*t*) / $*l*-*x*
if *l differentiable at* (*x*+*t*) *t* > *0 x*+*t* < $ψ **for** *t*::*real*
using *p-l mu-deriv-l that* **by** *simp*

lemma *p-mu-l-AE*: *AE s in lborel*. *0* < *s* ∧ *x*+*s* < $ψ ⟶ $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)
= − *deriv l* (*x*+*s*) / $*l*-*x*
proof −
have *AE s in lborel*. *l differentiable at* (*x*+*s*)
apply (*rule AE-borel-affine*[*of 1* λ*u*. *l differentiable at u x*, *simplified*])
unfolding *pred-def* **using** *l-differentiable-AE* **by** *simp-all*
moreover have *AE s in lborel*.
l differentiable at (*x*+*s*) ⟶ *0* < *s* ∧ *x*+*s* < $ψ ⟶ $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*) =
− *deriv l* (*x*+*s*) / $*l*-*x*
using *p-mu-l* **by** (*intro AE-I2*) *simp*
ultimately show *?thesis* **by** (*rule AE-mp*)
qed

lemma *LBINT-l-mu-q*: (*LBINT s*:{*f*<..*f*+*t*}. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*)) / $*l*-*x* = $*q*-{*f*|*t*&*x*}
if *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
proof −
have ⋀*s*. *s*∈{*f*<..*f*+*t*} ⟹ $*p*-{*s*&*x*} = $*l*-(*x*+*s*) / $*l*-*x* **using** *p-l that* **by** *simp*
hence $*q*-{*f*|*t*&*x*} = (*LBINT s*:{*f*<..*f*+*t*}. $*l*-(*x*+*s*) / $*l*-*x* ∗ $*μ*-(*x*+*s*))
using *LBINT-p-mu-q-defer*
by (*smt* (*verit*) *greaterThanAtMost-borel set-lebesgue-integral-cong sets-lborel*
that x-lt-psi)
also have ... = (*LBINT s*:{*f*<..*f*+*t*}. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*)) / $*l*-*x*
using *set-integral-divide-zero* **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *set-integrable-l-mu*: *set-integrable lborel* {*f*<..*f*+*t*} (λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*))
if *t* ≥ *0 f* ≥ *0* **for** *t f* :: *real*
proof −
have *set-integrable lborel* {*f*<..*f*+*t*} (λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) / $*l*-*x*)
using *p-l set-integrable-p-mu that*

by (*rewrite set-integrable-cong*[**where** *f′*=λ*s*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)]) *simp-all*+
thus *?thesis* **by** *simp*
qed

lemma *l-mu-has-integral-q-defer*:
((λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) / $*l*-*x*) *has-integral* $*q*-{*f*|*t*&*x*}) {*f*..*f*+*t*}
if *t* ≥ *0* *f* ≥ *0* **for** *t f* :: *real*
using *p-l that p-mu-has-integral-q-defer-Icc*
by (*rewrite has-integral-cong*[*of* - - λ*s*. $*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*)]; *simp*)

corollary *l-mu-has-integral-q*:
((λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) / $*l*-*x*) *has-integral* $*q*-{*t*&*x*}) {*0*..*t*} **if** *t* ≥ *0* **for** *t*::*real*
using *l-mu-has-integral-q-defer*[**where** *f*=*0*] *that* **by** *simp*

lemma *l-mu-has-integral-d*:
((λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*)) *has-integral* $*d*-{*t* & *x*+*f*}) {*f*..*f*+*t*}
if *t* ≥ *0* *f* ≥ *0* **for** *t f* :: *real*
proof −
have ((λ*s*. $*l*-*x* ∗ ($*p*-{*s*&*x*} ∗ $*μ*-(*x*+*s*))) *has-integral* $*l*-*x* ∗ $*q*-{*f*|*t*&*x*}) {*f*..*f*+*t*}
apply (*rule has-integral-mult-right*)
by (*rule p-mu-has-integral-q-defer-Icc*; *simp add*: *that*)
thus *?thesis*
using *that* **apply** (*rewrite* **in** *asm q-defer-d-l*, *simp-all*)
apply (*rewrite has-integral-cong*[**where** *g*=λ*s*. $*l*-*x* ∗ ($*p*-{*s*&*x*} ∗ $*μ*-(*x* + *s*))])
by (*rewrite p-l*; *simp*)
qed

corollary *l-mu-has-integral-d-1*:
((λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*)) *has-integral* $*d*-(*x*+*f*)) {*f*..*f*+*1*} **if** *t* ≥ *0* *f* ≥ *0* **for** *t*
f :: *real*
using *l-mu-has-integral-d*[**where** *t*=*1*] *that* **by** *simp*

lemma *e-LBINT-l*: $*e*ʻ∘-*x* = (*LBINT* *s*:{*0*..}. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) ∗ *s*) / $*l*-*x*
— Note that *0* = *0* holds when the life expectation diverges.
proof −
have ⋀*s*. *s*∈{*0*..} ⟹ $*p*-{*s*&*x*} = $*l*-(*x*+*s*) / $*l*-*x* **using** *p-l* **by** *simp*
hence $*e*ʻ∘-*x* = (*LBINT* *s*:{*0*..}. $*l*-(*x*+*s*) / $*l*-*x* ∗ $*μ*-(*x*+*s*) ∗ *s*)
using *e-LBINT-p-mu*
by (*smt* (*verit*) *atLeast-borel set-lebesgue-integral-cong sets-lborel x-lt-psi*)
also have ... = (*LBINT* *s*:{*0*..}. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) ∗ *s*) / $*l*-*x*
using *set-integral-divide-zero* **by** *simp*
finally show *?thesis* **.**
qed

lemma *e-integral-l*: $*e*ʻ∘-*x* = *integral* {*0*..} (λ*s*. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) ∗ *s*) / $*l*-*x*
— Note that *0* = *0* holds when the life expectation diverges.
proof −
have *AE s in lborel*. $*μ*-(*x*+*s*) ≥ *0* **by** (*rule AE-translation*, *rule mu-nonneg-AE*)
hence (*LBINT* *s*:{*0*..}. $*l*-(*x*+*s*) ∗ $*μ*-(*x*+*s*) ∗ *s*) = *integral* {*0*..} (λ*s*. $*l*-(*x*+*s*)

$* \$\mu\text{-}(x+s) * s)$
  by (*intro set-borel-integral-eq-integral-nonneg-AE*; *force*)
 thus *?thesis* **using** *e-LBINT-l* **by** *simp*
qed

lemma *m-LBINT-p-mu*: $\$m\text{-}\{n\&x\} = (LBINT\ t{:}\{0<..n\}.\ \$p\text{-}\{t\&x\} * \$\mu\text{-}(x+t)) / (LBINT\ t{:}\{0..n\}.\ \$p\text{-}\{t\&x\})$
 if $n \geq 0$ **for** *n*::*real*
 using *that*
 apply (*rewrite m-q-e*, *simp-all*)
 apply (*rewrite LBINT-p-mu-q*[*simplified*], *simp-all*)
 by (*rewrite temp-e-LBINT-p*; *simp*)

lemma *m-integral-p-mu*:
 $\$m\text{-}\{n\&x\} = integral\ \{0..n\}\ (\lambda t.\ \$p\text{-}\{t\&x\} * \$\mu\text{-}(x+t)) / integral\ \{0..n\}\ (\lambda t.\ \$p\text{-}\{t\&x\})$
 if $n \geq 0$ **for** *n*::*real*
 using *that*
 apply (*rewrite m-q-e*, *simp-all*)
 apply (*rewrite integral-unique*[*OF p-mu-has-integral-q-Icc*])
  apply *simp-all*[*2*]
 by (*rewrite temp-e-integral-p*; *simp*)

end

lemma *deriv-x-p-mu-l*: $deriv\ (\lambda y.\ \$p\text{-}\{t\&y\})\ x = \$p\text{-}\{t\&x\} * (\$\mu\text{-}x - \$\mu\text{-}(x+t))$
 if *l differentiable at x l differentiable at* $(x+t)$ $t \geq 0$ $x < \$\psi$ **for** *x t* :: *real*
 using *deriv-x-p-mu that differentiable-ccdfX-l* **by** *blast*

lemma *e-has-derivative-mu-e-l*: $((\lambda x.\ \$e\text{`}\circ\text{-}x)\ has\text{-}real\text{-}derivative\ (\$\mu\text{-}x * \$e\text{`}\circ\text{-}x - 1))\ (at\ x)$
 if *total-finite l differentiable at x* $x{\in}\{a<..<b\}$ $b \leq \$\psi$ **for** *a b x* :: *real*
 using *total-finite-iff-set-integrable-Ici that*
  e-has-derivative-mu-e differentiable-ccdfX-l set-integrable-ccdfX-l
 by *force*

corollary *e-has-derivative-mu-e-l'*: $((\lambda x.\ \$e\text{`}\circ\text{-}x)\ has\text{-}real\text{-}derivative\ (\$\mu\text{-}x * \$e\text{`}\circ\text{-}x - 1))\ (at\ x)$
 if *total-finite l differentiable at x* $x{\in}\{a<..<b\}$ $b \leq \$\psi$ **for** *a b x* :: *real*
 using *that* **by** (*intro e-has-derivative-mu-e-l*[**where** *a*=*a*]; *simp*)

context
 fixes *x*::*real*
 assumes *x-lt-psi*[*simp*]: $x < \$\psi$
begin

lemma *curt-e-sum-l-smooth*: $\$e\text{-}x = (\sum k.\ \$l\text{-}(x+k+1)) / \$l\text{-}x$ **if** *total-finite*
proof −
 have [*simp*]: *summable* $(\lambda k.\ \$l\text{-}(x+k+1))$

using *total-finite-iff-summable*[*of* $x+1$] *that*
by (*metis* (*no-types*, *lifting*) *add.commute add.left-commute summable-def sums-cong*)
hence *summable* (λk. \$*p*-{$k+1$&x}) **by** (*rewrite p-l*; *simp add*: *add.assoc*)
hence \$*e*-x = ($\sum k$. \$*p*-{$k+1$&x}) **using** *curt-e-sum-p-smooth* **by** *simp*
also have ... = ($\sum k$. \$*l*-($x+k+1$) / \$*l*-x) **by** (*rewrite p-l*; *simp add*: *add.assoc*)
also have ... = ($\sum k$. \$*l*-($x+k+1$)) / \$*l*-x **by** (*rewrite suminf-divide*; *simp*)
finally show *?thesis* **.**
qed

lemma *curt-e-sum-l-finite-smooth*: \$*e*-x = ($\sum k<n$. \$*l*-($x+k+1$)) / \$*l*-x **if** $x+n+1 > \$\psi$ **for** n::*nat*
apply (*rewrite curt-e-sum-p-finite-smooth*[*of* x n], *simp-all add*: *that*)
apply (*rewrite p-l*, *simp-all*)
by (*smt* (*verit*) *sum.cong sum-divide-distrib*)

lemma *temp-curt-e-sum-l-smooth*: \$*e*-{x:n} = ($\sum k<n$. \$*l*-($x+k+1$)) / \$*l*-x **for** n::*nat*
apply (*rewrite temp-curt-e-sum-p-smooth*[*of* x n], *simp*)
apply (*rewrite p-l*, *simp-all*)
by (*smt* (*verit*) *sum.cong sum-divide-distrib*)

end

end

6.4 Interpolations

context *life-table*
begin

definition *linear-interpolation* ≡
$\forall (x::nat)(t::real).\ 0 \le t \wedge t \le 1 \longrightarrow \$l\text{-}(x+t) = (1-t)*\$l\text{-}x + t*\$l\text{-}(x+1)$

lemma *linear-l*: \$*l*-($x+t$) = ($1-t$)*\$*l*-x + t*\$*l*-($x+1$)
if *linear-interpolation* $0 \le t$ $t \le 1$ **for** x::*nat* **and** t::*real*
using *that* **unfolding** *linear-interpolation-def* **by** (*metis of-nat-1 of-nat-add*)

lemma *linear-l-d*: \$*l*-($x+t$) = \$*l*-x − t*\$*d*-x
if *linear-interpolation* $0 \le t$ $t \le 1$ **for** x::*nat* **and** t::*real*
using *death-def-nonneg that* **unfolding** *linear-interpolation-def*
by (*smt* (*verit*) *distrib-left left-diff-distrib'*)

lemma *linear-p-q*: \$*p*-{t&x} = 1 − t*\$*q*-x
if *linear-interpolation* $0 \le t$ $t \le 1$ $x < \$\psi$ **for** x::*nat* **and** t::*real*
using *that*
apply (*rewrite p-l*, *simp-all*)
apply (*rewrite q-d-l*, *simp-all*)
using *divide-self*[*of* \$*l*-(*real* x)] *linear-l-d*
by (*smt* (*verit*, *ccfv-SIG*) *add-divide-distrib lx-neq-0*)

lemma *linear-q*: $q-\{t\&x\} = t*\$q-x$
if *linear-interpolation* $0 \leq t$ $t \leq 1$ $x < \$\psi$ **for** x::*nat* **and** t::*real*
using *that linear-p-q p-q-1* **by** (*smt* (*verit*))

lemma *linear-L-l-d*: $L-x = \$l-x - \$d-x / 2$ **if** *linear-interpolation* **for** x::*nat*
proof −
have $L-(real\ x) = (LBINT\ t{:}\{0..1\}.\ \$l-(real\ x + t))$
unfolding *lives-def* **using** *lborel-set-integral-Icc-shift*[*of real x real x + 1 l real x*]
by (*simp add*: *add.commute*)
also have ... $= (LBINT\ t{:}\{0..1\}.\ \$l-(real\ x) - t*\$d-(real\ x))$
using *linear-l-d that* **by** (*intro set-lebesgue-integral-cong*; *simp*)
also have ... $= \$l-(real\ x) - \$d-(real\ x) / 2$
proof −
have $(LBINT\ t{:}\{0::real..1\}.\ t) = 1/2$
unfolding *set-lebesgue-integral-def* **using** *integral-power*[*of 0 1 1*] **by** (*simp add*: *mult.commute*)
hence $(LBINT\ t{:}\{0..1\}.\ t*\$d-(real\ x)) = \$d-(real\ x) / 2$ **by** *auto*
moreover have *set-integrable lborel* $\{0..1\}$ $(\lambda t.\ t*\$d-(real\ x))$
apply (*rule set-integrable-mult-left*[**where** *f*=*id* **and** *a*=$\$d-(real\ x)$, *simplified*])
unfolding *set-integrable-def* **using** *integrable-power*[*of 0 1 1*] **by** (*simp add*: *mult.commute*)
moreover have $(LBINT\ t{:}\{0::real..1\}.\ \$l-(real\ x)) = \$l-(real\ x)$
unfolding *set-lebesgue-integral-def* **by** *simp*
ultimately show *?thesis* **using** *set-integrable-def* **by** (*rewrite set-integral-diff*; *force*)
qed
finally show *?thesis* **.**
qed

lemma *linear-L-l-d'*: $L-x = \$l-(x+1) + \$d-x / 2$ **if** *linear-interpolation* **for** x::*nat*
proof −
have $L-(real\ x) = \$l-(real\ x) - \$d-(real\ x) + \$d-(real\ x) / 2$ **using** *that linear-L-l-d* **by** *simp*
also have ... $= \$l-(real\ (x+1)) + \$d-(real\ x) / 2$ **using** *dx-l* **by** (*smt* (*verit*) *of-nat-1 of-nat-add*)
finally show *?thesis* **.**
qed

lemma *linear-l-continuous*: *continuous-on UNIV l* **if** *linear-interpolation*
unfolding *continuous-on-def*
proof
fix u::*real*
show $l \longrightarrow u \rightarrow \$l-u$
proof (*cases* ‹$u \leq 0$›)
case *True*
hence $(l \longrightarrow \$l-u)$ (*at-left u*) **using** *l-left-continuous-nonpos continuous-within* **by** *auto*

```
    thus ?thesis
      apply (rule filterlim-split-at-real)
      using l-right-continuous continuous-within by auto
  next
    case False
    hence u-pos: u > 0 by simp
    thus ?thesis
    proof (cases ‹u = real-of-int ⌊u⌋›)
      case True
      from this u-pos obtain x::nat where ux: u = Suc x
        by (metis gr0-implies-Suc of-int-0-less-iff of-int-of-nat-eq pos-int-cases)
     have ((λt. (1−t)*$l-(real x) + t*$l-(real x + 1)) ⟶⟶ $l-(real x + 1)) (at-left
1)
        apply (rewrite in (- ⟶⟶ ⌑) - add-0[THEN sym], rule tendsto-add)
         apply (simp add: LIM-zero-iff' tendsto-mult-left-zero)
         by (rewrite in (- ⟶⟶ ⌑) - mult-1[THEN sym], rule tendsto-mult-right)
simp
      hence ((λt. $l-(real x + t)) ⟶⟶ $l-(real x + 1)) (at-left 1)
        apply (rewrite tendsto-cong; simp)
        apply (rule eventually-at-leftI[of 0]; simp)
        using that by (rewrite linear-l; simp add: add.commute)
      moreover have ((λt. $l-(real x + t)) ⟶⟶ $l-(real x + 1)) (at-right 1)
          using l-right-continuous apply (rule continuous-within-tendsto-compose,
simp-all)
         apply (rule eventually-at-right-less)
        by (rule tendsto-intros, simp-all)
      ultimately show ?thesis
        apply (rewrite ux)+
        apply (rewrite filterlim-shift-iff[where d=x, THEN sym])
        by (rule filterlim-split-at-real; simp add: comp-def add.commute)
    next
      case False
      let ?x = nat ⌊u⌋
      let ?t = u − real ?x
      let ?e = min ?t (1 − ?t)
      from False have ?t > 0 using u-pos by linarith
      moreover have ?t < 1 by linarith
      ultimately have e-pos: ?e > 0 by simp
      hence
      ∀ F v in nhds u. $l-v = (1 − (v − real ?x))*$l-(real ?x) + (v − real ?x)*$l-(real
?x + 1)
      proof −
        { fix v::real assume vu-e: dist v u < ?e
          hence v − real ?x ≥ 0 using dist-real-def by force
          moreover have v − real ?x ≤ 1 using dist-real-def vu-e by force
            ultimately have $l-v = (1 − (v − real ?x))*$l-(real ?x) + (v − real
?x)*$l-(real ?x + 1)
          using linear-l that by (smt (verit, ccfv-threshold) linear-interpolation-def)
}
```

```
      thus ?thesis using eventually-nhds-metric e-pos by blast
    qed
    moreover have
      isCont (λv. (1 − (v − real ?x))∗$l-(real ?x) + (v − real ?x)∗$l-(real ?x +
1)) u
      by (rule continuous-intros)+
    ultimately have isCont l u using isCont-cong by force
    thus ?thesis by (simp add: isContD)
  qed
 qed
qed

lemma linear-l-sums-T-l: (λk. $l-(x + Suc k)) sums ($T-x − $l-x / 2)
 if linear-interpolation total-finite for x::nat
proof −
 have ⋀k::nat. $l-(real (x + Suc k)) = $L-(real (x+k)) − $d-(real (x+k)) / 2
  using linear-L-l-d′ that by (smt (verit) Suc-eq-plus1 add-Suc-right)
 moreover have (λk::nat. $L-(real (x+k))) sums $T-x using L-sums-T that by
simp
 moreover have (λk::nat. $d-(real (x+k)) / 2) sums ($l-(real x) / 2)
  using sums-divide d-sums-l by auto
 ultimately show ?thesis
  apply (rewrite sums-cong, simp)
  by (rule sums-diff; simp)
qed

corollary linear-T-suminf-l: $T-x = (∑ k. $l-(x+k+1)) + $l-x / 2
 if linear-interpolation total-finite for x::nat
 using linear-l-sums-T-l that sums-unique
  by (metis (no-types, lifting) diff-diff-eq2 nat-arith.add1 right-minus-eq semir-
ing-norm(174) suminf-cong)

lemma linear-mx-q: $m-x = $q-x / (1 − $q-x / 2) if linear-interpolation x < $ψ
for x::nat
proof −
 have [simp]: $l-(real x) ≠ 0 using that by simp
 have $m-(real x) = $d-(real x) / ($l-(real x) − $d-(real x) / 2)
  unfolding die-central-def using linear-L-l-d that by simp
 also have ... = ($d-(real x) / $l-(real x)) / (($l-(real x) − $d-(real x) / 2) /
$l-(real x))
  by simp
 also have ... = ($d-(real x) / $l-(real x)) / (1 − ($d-(real x) / $l-(real x)) / 2)
  by (rewrite diff-divide-distrib) simp
 also have ... = $q-(real x) / (1 − $q-(real x) / 2) using that q-d-l by simp
 finally show ?thesis .
qed

lemma linear-e-curt-e: $e‘∘-x = $e-x + 1/2
 if linear-interpolation total-finite x < $ψ for x::nat
```

proof −
have $e‘∘-(*real x*) = ((∑ *k*::*nat*. $*l*-(*real* (*x*+*k*+*1*))) + $*l*-(*real x*) / *2*) / $*l*-(*real x*)
using *e-T-l linear-T-suminf-l that* **by** *simp*
also have ... = (∑ *k*::*nat*. $*l*-(*real* (*x*+*k*+*1*))) / $*l*-(*real x*) + ($*l*-(*real x*) / *2*) / $*l*-(*real x*)
using *add-divide-distrib* **by** *blast*
also have ... = $*e*-(*real x*) + *1*/*2*
using *that* **apply** (*rewrite curt-e-sum-l*, *simp-all*)
using *linear-l-continuous* **by** (*rule continuous-on-interior*, *simp-all add*: *that add.commute*)
finally show *?thesis* **.**
qed

end

context *smooth-life-table*
begin

lemma *linear-l-has-derivative-at-frac*:
((λ*s*. $*l*-(*x*+*s*)) *has-real-derivative* − $*d*-*x*) (*at t*)
if *linear-interpolation 0* < *t t* < *1* **for** *x*::*nat* **and** *t*::*real*
proof −
let *?x* = *real x*
have ((λ*s*. $*l*-*?x* − *s*∗$*d*-*?x*) *has-real-derivative* (*0* − $*d*-*?x*)) (*at t*)
apply (*rule derivative-intros*, *simp*)
apply (*rule DERIV-cmult-right*[*of id 1*, *simplified*])
by (*metis DERIV-ident eq-id-iff*)
moreover have ∀ *F s in nhds t*. $*l*-(*?x* + *s*) = $*l*-*?x* − *s*∗$*d*-*?x*
proof −
let *?r* = *min t* (*1*−*t*)
have *?r* > *0* **using** *that* **by** *simp*
moreover
{ fix *s* **assume** *dist s t* < *?r*
hence $*l*-(*?x* + *s*) = $*l*-*?x* − *s*∗$*d*-*?x* **using** *linear-l-d that dist-real-def* **by** *force* **}**
ultimately show *?thesis* **using** *eventually-nhds-metric that* **by** *blast*
qed
ultimately show *?thesis* **by** (*rewrite DERIV-cong-ev*; *simp*)
qed

lemma *linear-l-has-derivative-at-frac′*:
(*l has-real-derivative* − $*d*-*x*) (*at y*)
if *linear-interpolation x* < *y y* < *x*+*1* **for** *x*::*nat* **and** *y*::*real*
apply (*rewrite DERIV-at-within-shift*[**where** *x*=*y* − *real x* **and** *z*=*real x* **and** *S*=*UNIV*, *simplified*])
using *linear-l-has-derivative-at-frac that* **by** *simp*

lemma *linear-l-differentiable-on-frac*:

```
  l differentiable-on {x<..<x+1} if linear-interpolation for x::nat
proof −
  { fix y::real assume y ∈ {real x <..< real (x+1)}
    hence l differentiable at y
      using linear-l-has-derivative-at-frac' that real-differentiable-def by auto }
  thus ?thesis unfolding differentiable-on-def by (metis differentiable-at-withinI)
qed

lemma linear-l-has-right-derivative-at-nat:
  (l has-real-derivative − $d-x) (at-right x) if linear-interpolation for x::nat
proof −
  let ?x = real x
  have [simp]: plus ?x ‘ {0<..} = {?x<..}
    unfolding image-def greaterThan-def apply simp
    by (metis Groups.ab-semigroup-add-class.add.commute
        add-minus-cancel neg-less-iff-less real-0-less-add-iff)
  have ((λs. $l-(?x + s)) has-real-derivative − $d-?x) (at-right 0)
   apply (rewrite has-field-derivative-cong-eventually[where g=λs. $l-?x − s∗$d-?x])
    using linear-l-d that apply (intro eventually-at-rightI[of 0 1], simp-all)
    apply (rule has-field-derivative-at-within)
    apply (rewrite diff-0[of $d-?x, THEN sym])
    apply (rule DERIV-diff, simp)
    apply (rule DERIV-cmult-right[of id 1, simplified])
    by (metis DERIV-ident eq-id-iff)
  thus ?thesis
     by (rewrite DERIV-at-within-shift[where z=?x and x=0 and S={0<..},
simplified]) simp
qed

lemma linear-l-has-left-derivative-at-nat:
   (l has-real-derivative − $d-(real x − 1)) (at-left x) if linear-interpolation for
x::nat
proof (cases x)
  case 0
  hence (l has-real-derivative 0) (at-left (real x))
    apply (rewrite has-field-derivative-cong-eventually[where g=λ-. $l-0]; simp)
    apply (rule eventually-at-leftI[of −1]; simp)
    using l-neg-nil less-eq-real-def by blast
  moreover have $d-(real x − 1) = 0 using 0 dx-l l-neg-nil less-eq-real-def by
fastforce
  ultimately show ?thesis by auto
next
  case (Suc y)
  let ?x = real x and ?y = real y
  have [simp]: ?y + 1 = ?x using Suc by simp
  have [simp]: plus ?y ‘ {..<1} = {..<?x}
    using Suc unfolding image-def lessThan-def apply simp
   by (metis (no-types, opaque-lifting) Groups.ab-semigroup-add-class.add.commute
       add-less-cancel-right diff-add-cancel)
```

```
  have $l-?x = $l-real y − $d-real y using Suc by (simp add: dx-l)
  moreover have  ∀ F s in at-left 1. $l-(?y + s) = $l-?y − s*$d-?y
    apply (rule eventually-at-leftI[of 0], simp-all)
    using Suc linear-l-d that by simp
  moreover have ((λs. $l-?y − s*$d-?y) has-real-derivative − $d-?y) (at-left 1)
    apply (rule has-field-derivative-at-within)
    apply (rewrite diff-0[of $d-?y, THEN sym])
    apply (rule DERIV-diff, simp)
    apply (rule DERIV-cmult-right[of id 1, simplified])
    by (metis DERIV-ident eq-id-iff)
  ultimately have ((λs. $l-(?y + s)) has-real-derivative − $d-?y) (at-left 1)
    by (rewrite has-field-derivative-cong-eventually[where g=λs. $l-?y − s*$d-?y];
simp)
  thus ?thesis
    apply (rewrite DERIV-at-within-shift[where S={..<1} and z=?y and x=1,
simplified])
    using Suc by simp
qed

lemma linear-l-has-derivative-at-nat-iff-d:
  (l has-real-derivative − $d-x) (at x) ⟷ $d-x = $d-(real x − 1)
  if linear-interpolation for x::nat
proof −
  let ?x = real x
  have (l has-real-derivative − $d-?x) (at ?x) ⟷
    (l has-real-derivative − $d-?x) (at-right ?x) ∧
    (l has-real-derivative − $d-?x) (at-left ?x)
    using has-real-derivative-at-split by auto
  also have ... ⟷ $d-?x = $d-(?x − 1) (is ?LHS = ?RHS)
  proof
    assume ?LHS
    hence (l has-real-derivative − $d-?x) (at-left ?x) by simp
    moreover have (l has-real-derivative − $d-(?x − 1)) (at-left ?x)
      using that linear-l-has-left-derivative-at-nat by simp
    ultimately show ?RHS
    using has-real-derivative-iff-has-vector-derivative vector-derivative-unique-within
       trivial-limit-at-left-real by (smt (verit, ccfv-SIG))
  next
    assume ?RHS
    thus ?LHS
     using that linear-l-has-right-derivative-at-nat linear-l-has-left-derivative-at-nat
by metis
  qed
  finally show ?thesis .
qed

lemma linear-l-differentiable-at-nat-iff-d:
  l differentiable at x ⟷ $d-x = $d-(real x − 1)
  if linear-interpolation for x::nat
```

proof
let *?x* = *real x*
assume *l differentiable at x*
from *this* **obtain** *D* **where** *DERIV-l*: (*l has-real-derivative D*) (*at ?x*)
using *real-differentiable-def* **by** *blast*
hence (*l has-real-derivative D*) (*at-right ?x*)
using *has-field-derivative-at-within* **by** *blast*
moreover have *at ?x within* {*real x<..*} ≠ ⊥ **by** *simp*
moreover have (*l has-real-derivative* − $*d-?x*) (*at-right ?x*)
using *linear-l-has-right-derivative-at-nat that* **by** *simp*
ultimately have *D* = − $*d-?x*
using *that has-real-derivative-iff-has-vector-derivative vector-derivative-unique-within*
by *blast*
thus $*d-?x* = $*d-*(*?x* − *1*)
using *linear-l-has-derivative-at-nat-iff-d that DERIV-l* **by** *blast*
next
assume $*d-*(*real x*) = $*d-*(*real x* − *1*)
thus *l differentiable at* (*real x*)
using *linear-l-has-derivative-at-nat-iff-d that real-differentiable-def* **by** *blast*
qed

lemma *linear-l-limited*: $*ψ* < ∞ **if** *linear-interpolation*
proof −
let *?ND* = {*y*. ¬ *l differentiable at y*}
obtain *xn*::*nat* **where** *xn-def*: *Max ?ND* < *real xn* **using** *reals-Archimedean2*
by *blast*
hence *xn-dif*: ⋀*x*::*nat*. *x* ≥ *xn* ⟹ *l differentiable at* (*real x*)
proof −
fix *x*::*nat* **assume** *x* ≥ *xn*
with *xn-def* **have** *real x* > *Max ?ND* **by** *simp*
hence *real x* ∉ *?ND* **using** *notI Max.coboundedI l-nondifferentiable-finite-set*
leD **by** *blast*
thus *l differentiable at* (*real x*) **by** *simp*
qed
hence *d-const*: ⋀*x*::*nat*. *x* ≥ *xn* ⟹ $*d-*(*real x*) = $*d-*(*real xn*)
proof −
fix *x*::*nat* **assume** *x* ≥ *xn*
moreover have
⋀*y*::*nat*. *xn* ≤ *y* ⟹ $*d-*(*real y*) = $*d-*(*real xn*) ⟹ $*d-*(*real* (*Suc y*)) = $*d-*(*real*
xn)
using *linear-l-differentiable-at-nat-iff-d that*
by (*smt* (*verit*, *best*) *of-nat-Suc of-nat-le-iff xn-dif*)
ultimately show $*d-*(*real x*) = $*d-*(*real xn*)
using *nat-induct-at-least*[**where** *P*=λ*x*::*nat*. $*d-*(*real x*) = $*d-*(*real xn*)] **by**
simp
qed
have ¬ $*d-*(*real xn*) > *0*
proof (*rule notI*)
assume $*d-*(*real xn*) > *0*

```
    from this obtain N::nat where N-def: N * $d-(real xn) > $l-(real xn)
      using reals-Archimedean3 by blast
    hence $l-(real (xn+N)) < 0
    proof −
     have $l-(real (xn+N)) = $l-(real xn) − (∑ k<N. $d-(real xn + real k)) using
sum-dx-l by simp
      also have ... = $l-(real xn) − (∑ k<N. $d-(real xn))
        using d-const by (metis le-add1 of-nat-add)
      also have ... = $l-(real xn) − N * $d-(real xn) by simp
      also have ... < 0 using N-def by simp
      finally show ?thesis .
    qed
    thus False by (smt (verit, ccfv-SIG) l-nonneg)
  qed
  hence dxn0: $d-(real xn) = 0 by (smt (verit) d-nonneg)
  hence ⋀x::nat. x ≥ xn ⟹ $l-(real x) = $l-(real xn)
  proof −
    fix x::nat
    assume xn ≤ x
    moreover have
      ⋀y::nat. xn ≤ y ⟹ $l-(real y) = $l-(real xn) ⟹ $l-(real (Suc y)) = $l-(real
xn)
      by (smt (verit, ccfv-threshold) dxn0 d-const dx-l of-nat-Suc)
    ultimately show $l-(real x) = $l-(real xn)
      using nat-induct-at-least[where P=λx::nat. $l-(real x) = $l-(real xn)] by
simp
  qed
  hence (λx::nat. $l-(real x)) ⟶ $l-(real xn)
    using eventually-sequentially by (intro tendsto-eventually) blast
  moreover have (λx::nat. $l-(real x)) ⟶ 0
    using l-PInfty-0 by (simp add: filterlim-compose filterlim-real-sequentially)
  ultimately have $l-(real xn) = 0 by (simp add: LIMSEQ-unique)
  thus ?thesis by force
qed

lemma linear-mu-q: $μ-(x+t) = $q-x / (1 − t*$q-x)
  if linear-interpolation l differentiable at (x+t) 0 < t t < 1 x+t < $ψ
  for x::nat and t::real
proof −
  have [simp]: ereal x < $ψ using that by (simp add: ereal-less-le)
  have [simp]: $l-(real x) ≠ 0 by simp
  have [simp]: l field-differentiable at (real x + t)
    using differentiable-eq-field-differentiable-real that by simp
  define d where d ≡ min t (1−t)
  have d-pos: d > 0 unfolding d-def using that by simp
  have $p-{t & real x} ≠ 0 using that by (simp add: ereal-less-le p-0-equiv)
  moreover have (λs. $p-{s & real x}) differentiable at t
  proof −
    have (λs. $l-(real x + s) / $l-(real x)) field-differentiable at t
```

using *that* **apply** (*intro derivative-intros*)
apply (*rewrite add.commute*, *rewrite field-differentiable-shift*[*THEN sym*])
by (*rewrite add.commute*) *simp-all*
thus *?thesis*
apply (*rewrite differentiable-eq-field-differentiable-real*)
apply (*rule field-differentiable-transform-within*[**where** *d=d*], *simp-all add*: *d-pos*)
apply (*rewrite p-l*, *simp-all*) **unfolding** *d-def* **using** *dist-real-def* **by** *auto*
qed
ultimately have $\$\mu$-(*real x + t*) = − *deriv* (*λs.* $\$p$-{*s* & *real x*}) *t* / $\$p$-{*t* & *real x*}
using *that deriv-t-p-mu* **by** *simp*
also have ... = $\$q$-(*real x*) / (*1* − *t*∗$\$q$-(*real x*))
proof −
have ⋀*s. dist s t* < *d* ⟹ $\$p$-{*s* & *real x*} = *1* − *s*∗$\$q$-(*real x*)
proof −
fix *s* **assume** *dist s t* < *d*
hence *0* ≤ *s s* ≤ *1* **unfolding** *d-def* **using** *that dist-real-def* **by** *auto*
thus $\$p$-{*s* & *real x*} = *1* − *s*∗$\$q$-(*real x*) **by** (*intro linear-p-q*; *simp add*: *that*)
qed
hence *deriv* (*λs.* $\$p$-{*s* & *real x*}) *t* = *deriv* (*λs. 1* − *s*∗$\$q$-(*real x*)) *t*
using *d-pos* **apply** (*intro deriv-cong-ev*, *simp-all*)
by (*rewrite eventually-nhds-metric*) *auto*
also have ... = − $\$q$-(*real x*)
apply (*rewrite deriv-diff*, *simp-all*)
by (*rule derivative-intros*) *auto*
finally have *deriv* (*λs.* $\$p$-{*s* & *real x*}) *t* = − $\$q$-(*real x*) **.**
thus *?thesis* **using** *linear-p-q that* **by** *simp*
qed
finally show *?thesis* **.**
qed

definition *exponential-interpolation* ≡
∀(*x*::*nat*)(*t*::*real*). *x+1* < $\$\psi$ ⟶ *0* ≤ *t* ∧ *t* < *1* ⟶ $\$\mu$-(*x*+*t*) = $\$\mu$-*x*
— Without *x+1* < $\$\psi$, the smooth life table could not be limited.

lemma *exponential-mu*: $\$\mu$-(*x*+*t*) = $\$\mu$-*x*
if *exponential-interpolation x+1* < $\$\psi$ *0* ≤ *t t* < *1* **for** *x*::*nat* **and** *t*::*real*
using *that* **unfolding** *exponential-interpolation-def* **by** *simp*

corollary *exponential-mu'*: $\$\mu$-*y* = $\$\mu$-*x*
if *exponential-interpolation x* ≤ *y y* < *x+1 x+1* < $\$\psi$ **for** *x*::*nat* **and** *y*::*real*
proof −
let *?t* = *y* − *real x*
have *0* ≤ *?t* **and** *?t* < *1* **using** *that* **by** *simp-all*
moreover have $\$\mu$-*y* = $\$\mu$-(*real x* + *?t*) **by** *simp*
ultimately show *?thesis* **using** *exponential-mu that* **by** *presburger*
qed

lemma *exponential-integral-mu*: *integral* $\{x..<x+t\}$ $(\lambda y.\ \$\mu\text{-}y) = \$\mu\text{-}x * t$
if *exponential-interpolation* $x+1 < \$\psi$ $0 \le t$ $t \le 1$ **for** x::*nat* **and** t::*real*
proof −
have *integral* $\{real\ x\ ..<\ real\ x + t\}$ $(\lambda y.\ \$\mu\text{-}y) =$
integral $\{real\ x\ ..<\ real\ x + t\}$ $(\lambda y.\ \$\mu\text{-}(real\ x))$
using *exponential-mu′ that* **by** (*intro integral-cong*; *simp*)
also have ... = *integral* $\{real\ x\ ..\ real\ x + t\}$ $(\lambda y.\ \$\mu\text{-}(real\ x))$
apply (*rule integral-subset-negligible*, *force*)
using *that* **by** (*rewrite Icc-minus-Ico*; *simp*)
also have ... = $\$\mu\text{-}x * t$ **using** *that* **by** (*rewrite integral-const-real*) *simp*
finally show *integral* $\{real\ x\ ..<\ real\ x + t\}$ $(\lambda y.\ \$\mu\text{-}y) = \$\mu\text{-}(real\ x) * t$ **.**
qed

lemma *exponential-p-mu*: $\$p\text{-}x = exp\ (-\$\mu\text{-}x)$ **if** *exponential-interpolation* $x+1 <$ $\$\psi$ **for** x::*nat*
proof −
have $\$p\text{-}x = exp\ (-\ integral\ \{real\ x\ ..<\ real\ x + 1\}\ (\lambda y.\ \$\mu\text{-}y))$
using *that* **apply** (*rewrite p-exp-integral-mu*; *simp add*: *add.commute*)
apply (*rule integral-subset-negligible*[*THEN sym*], *force*)
by (*rewrite Icc-minus-Ico*; *simp*)
also have ... = *exp* $(-\ \$\mu\text{-}(real\ x))$ **using** *that* **by** (*rewrite exponential-integral-mu*; *simp*)
finally show *?thesis* **.**
qed

corollary *exponential-mu-p*: $\$\mu\text{-}x = -\ ln\ (\$p\text{-}x)$ **if** *exponential-interpolation* $x+1$ $< \$\psi$ **for** x::*nat*
using *exponential-p-mu that* **by** *simp*

corollary *exponential-mu-xt-p*: $\$\mu\text{-}(x+t) = -\ ln\ (\$p\text{-}x)$
if *exponential-interpolation* $x+1 < \$\psi$ $0 \le t$ $t < 1$ **for** x::*nat* **and** t::*real*
using *that exponential-mu exponential-mu-p* **by** *presburger*

corollary *exponential-q-mu*: $\$q\text{-}x = 1 - exp\ (-\$\mu\text{-}x)$
if *exponential-interpolation* $x+1 < \$\psi$ **for** x::*nat*
using *exponential-p-mu that p-q-1*
by (*smt* (*verit*, *ccfv-SIG*) *ereal-less-le not-add-less1 of-nat-less-imp-less*)

lemma *exponential-p*: $\$p\text{-}\{t\&x\} = (\$p\text{-}x).\hat{}t$
if *exponential-interpolation* $x+1 < \$\psi$ $0 \le t$ $t \le 1$ **for** x::*nat* **and** t::*real*
proof −
have [*simp*]: $real\ x + t < \$\psi$ **using** *that ereal-less-le* **by** *auto*
have $\$p\text{-}\{t\ \&\ real\ x\} = exp\ (-\ integral\ \{real\ x\ ..<\ real\ x + t\}\ (\lambda y.\ \$\mu\text{-}y))$
using *that* **apply** (*rewrite p-exp-integral-mu*, *simp-all*)
apply (*rule integral-subset-negligible*[*THEN sym*], *force*)
by (*rewrite Icc-minus-Ico*; *simp*)
also have ... = *exp* $(-\ \$\mu\text{-}(real\ x) * t)$
using *that* **by** (*rewrite exponential-integral-mu*; *simp*)
also have ... = $(\$p\text{-}(real\ x)).\hat{}t$

using *exponential-p-mu that*
by (*smt* (*verit*) *exp-not-eq-zero exponential-mu-p mult.commute powr-def*)
finally show *?thesis* **.**
qed

lemma *exponential-q*: $\$q\text{-}\{t\&x\} = 1 - (1 - \$q\text{-}x).\hat{}t$
if *exponential-interpolation* $x{+}1 < \$\psi$ $0 \leq t$ $t \leq 1$ **for** *x*::*nat* **and** *t*::*real*
proof −
have $\$q\text{-}\{t$ & *real x*$\} = 1 - \$p\text{-}\{t$ & *real x*$\}$
using *p-q-1 that* **by** (*smt* (*verit*) *ereal-less-le le-add1 of-nat-mono*)
also have ... $= 1 - (\$p\text{-}(real\ x)).\hat{}t$ **using** *that* **by** (*rewrite exponential-p*; *simp*)
also have ... $= 1 - (1 - \$q\text{-}(real\ x)).\hat{}t$
using *p-q-1 that* **by** (*smt* (*verit*) *ereal-less-le not-add-less1 of-nat-less-imp-less*)
finally show *?thesis* **.**
qed

lemma *exponential-l-p*: $\$l\text{-}(x{+}t) = \$l\text{-}x * (\$p\text{-}x).\hat{}t$
if *exponential-interpolation* $x{+}1 < \$\psi$ $0 \leq t$ $t \leq 1$ **for** *x*::*nat* **and** *t*::*real*
proof −
have *ereal* (*real x*) $< \$\psi$ **using** *that ereal-less-le* **by** *auto*
hence $\$l\text{-}(real\ x + t) = \$l\text{-}x * \$p\text{-}\{t$ & *real x*$\}$ **using** *that* **by** (*rewrite p-l*; *simp*)
also have ... $= \$l\text{-}(real\ x) * (\$p\text{-}(real\ x)).\hat{}t$ **using** *that* **by** (*rewrite exponential-p*; *simp*)
finally show *?thesis* **.**
qed

lemma *exponential-l-has-derivative-at-frac*:
$((\lambda s.\ \$l\text{-}(x{+}s))$ *has-real-derivative* $(-\ \$l\text{-}x * \$\mu\text{-}x * (\$p\text{-}x).\hat{}t))$ $(at\ t)$
if *exponential-interpolation* $x{+}1 < \$\psi$ $0 < t$ $t < 1$ **for** *x*::*nat* **and** *t*::*real*
proof −
let *?x* = *real x*
have $((\lambda s.\ (\$p\text{-}?x).\hat{}s)$ *has-real-derivative* $(-\ \$\mu\text{-}?x * (\$p\text{-}?x).\hat{}t))$ $(at\ t)$
using *that exponential-p-mu has-real-derivative-powr2*
by (*metis Groups.ab-semigroup-mult-class.mult.commute exp-gt-zero ln-exp*)
hence $((\lambda s.\ \$l\text{-}?x * (\$p\text{-}?x).\hat{}s)$ *has-real-derivative* $(-\ \$l\text{-}?x * \$\mu\text{-}?x * (\$p\text{-}?x).\hat{}t))$ $(at\ t)$
by (*rewrite minus-mult-commute*, *rewrite mult.assoc*) (*rule DERIV-cmult*)
moreover have $\forall_F\ s$ *in nhds t*. $\$l\text{-}(?x + s) = \$l\text{-}?x * (\$p\text{-}?x).\hat{}s$
proof −
let *?r* = *min t* (*1*−*t*)
have *?r* > *0* **using** *that* **by** *simp*
moreover have $\bigwedge s.\ dist\ s\ t < ?r \Longrightarrow \$l\text{-}(?x + s) = \$l\text{-}?x * (\$p\text{-}?x).\hat{}s$
using *dist-real-def that exponential-l-p* **by** *force*
ultimately show *?thesis* **using** *eventually-nhds-metric* **by** *blast*
qed
ultimately show *?thesis* **by** (*rewrite DERIV-cong-ev*[**where** $g{=}\lambda s.\ \$l\text{-}?x * (\$p\text{-}?x).\hat{}s$]; *simp*)
qed

lemma *exponential-l-has-derivative-at-frac'*:
(*l has-real-derivative* (− $*l*-*x* ∗ $*μ*-*x* ∗ ($*p*-*x*).^(*y*−*x*))) (*at y*)
if *exponential-interpolation x*+*1* < $*ψ x* < *y y* < *x*+*1* **for** *x*::*nat* **and** *y*::*real*
apply (*rewrite DERIV-at-within-shift*[**where** *x*=*y* − *real x* **and** *z*=*real x* **and** *S*=*UNIV*, *simplified*])
using *exponential-l-has-derivative-at-frac that* **by** *simp*

lemma *exponential-l-differentiable-on-frac*:
l differentiable-on {*x*<..<*x*+*1*} **if** *exponential-interpolation x*+*1* < $*ψ* **for** *x*::*nat*
proof −
{ fix *y*::*real* **assume** *y* ∈ {*real x* <..< *real* (*x*+*1*)}
hence *l differentiable at y*
using *exponential-l-has-derivative-at-frac' that real-differentiable-def* **by** *auto*
}
thus *?thesis* **unfolding** *differentiable-on-def* **by** (*metis differentiable-at-withinI*)
qed

lemma *exponential-l-has-right-derivative-at-nat*:
(*l has-real-derivative* (− $*l*-*x* ∗ $*μ*-*x*)) (*at-right x*)
if *exponential-interpolation x*+*1* < $*ψ* **for** *x*::*nat*
proof −
let *?x* = *real x*
have [*simp*]: *plus ?x* ' {*0*<..} = {*?x*<..}
unfolding *image-def greaterThan-def* **apply** *simp*
by (*metis Groups.ab-semigroup-add-class.add.commute add-minus-cancel neg-less-iff-less real-0-less-add-iff*)
have [*simp*]: $*p*-*x* > *0* **using** *exponential-p-mu that* **by** *auto*
hence [*simp*]: $*p*-*x* ≠ *0* **by** *force*
have ((*λs*. $*l*-(*?x* + *s*)) *has-real-derivative* (− $*l*-*?x* ∗ $*μ*-*?x*)) (*at-right 0*)
apply (*rewrite has-field-derivative-cong-eventually*[**where** *g*=*λs*. $*l*-*?x* ∗ ($*p*-*?x*).^*s*])
using *exponential-l-p that* **apply** (*intro eventually-at-rightI*[*of 0 1*]; *simp*)
using *powr-zero-eq-one* **apply** *simp*
apply (*rewrite minus-mult-commute*, *rule DERIV-cmult*)
apply (*rule has-field-derivative-at-within*)
using *that* **apply** (*rewrite exponential-mu-p*; *simp*)
using *has-real-derivative-powr2*[*of* $*p*-*x 0*] *powr-zero-eq-one* **by** *force*
thus *?thesis*
by (*rewrite DERIV-at-within-shift*[**where** *z*=*?x* **and** *x*=*0* **and** *S*={*0*<..}, *simplified*]) *simp*
qed

lemma *exponential-l-has-left-derivative-at-nat*:
(*l has-real-derivative* (− $*l*-*x* ∗ $*μ*-(*real x* − *1*))) (*at-left x*)
if *exponential-interpolation x* < $*ψ* **for** *x*::*nat*
proof (*cases x*)
case *0*
hence (*l has-real-derivative 0*) (*at-left* (*real x*))
apply (*rewrite has-field-derivative-cong-eventually*[**where** *g*=*λ*-. $*l*-*0*]; *simp*)
apply (*rule eventually-at-leftI*[*of* −*1*]; *simp*)

```
    using l-neg-nil less-eq-real-def by blast
  moreover have − $l-x * $μ-(real x −1) = 0 using mu-unborn-0 0 by simp
  ultimately show ?thesis by auto
next
  let ?x = real x
  case (Suc y)
  let ?y = real y
  have [simp]: ?y + 1 = ?x using Suc by simp
  hence [simp]: ereal ?y < $ψ using that by (smt (verit) ereal-less-le)
  have [simp]: $p-?y > 0 using Suc exponential-p-mu that by auto
  have [simp]: plus ?y ' {..<1} = {..<?x}
    using Suc unfolding image-def lessThan-def apply simp
   by (metis (no-types, opaque-lifting) Groups.ab-semigroup-add-class.add.commute
       add-less-cancel-right diff-add-cancel)
  have ((λt. ($p-?y).^t) has-real-derivative ($p-?y * −$μ-?y)) (at-left 1)
    apply (rule DERIV-subset[where s=UNIV]; simp)
    using that apply (rewrite exponential-mu-p; simp add: add.commute[of 1 ?y])
    by (rule has-real-derivative-powr2[of $p-?y 1, simplified])
  hence ((λt. $l-?y * ($p-?y).^t) has-real-derivative ($l-?y * $p-?y * −$μ-?y))
(at-left 1)
    by (metis DERIV-cmult mult-ac(1))
  moreover have $l-?y * $p-?y = $l-?x using Suc p-1-l by simp
  ultimately have ((λt. $l-?y * ($p-?y).^t) has-real-derivative (− $l-?x * $μ-?y))
(at-left 1)
    by simp
  moreover have ∀F t in at-left 1. $l-(?y + t) = $l-?y * ($p-?y).^t
    apply (rule eventually-at-leftI[of 0]; simp)
    by (rewrite exponential-l-p; simp add: that add.commute[of 1 ?y])
  ultimately have ((λt. $l-(?y + t)) has-real-derivative (− $l-?x * $μ-?y)) (at-left
1)
   by (rewrite has-field-derivative-cong-eventually[where g=λt. $l-?y * ($p-?y).^t];
       simp add: p-1-l)
  thus ?thesis
    apply (rewrite DERIV-at-within-shift[where S={..<1} and z=?y and x=1,
simplified])
    using Suc by simp
qed

lemma exponential-l-has-derivative-at-nat-iff-mu:
  (l has-real-derivative (− $l-x * $μ-x)) (at x) ⟷ $μ-x = $μ-(real x − 1)
  if exponential-interpolation x+1 < $ψ for x::nat
proof −
  let ?x = real x
  have [simp]: ?x < $ψ using that by (simp add: ereal-less-le)
  hence [simp]: $l-?x ≠ 0 by simp
  have (l has-real-derivative (− $l-?x * $μ-?x)) (at ?x) ⟷
    (l has-real-derivative (− $l-?x * $μ-?x)) (at-right ?x) ∧
    (l has-real-derivative (− $l-?x * $μ-?x)) (at-left ?x)
    using has-real-derivative-at-split by auto
```

also have ... $\longleftrightarrow$ $-$ \$*l*-*?x* $*$ \$$\mu$-*?x* $=$ $-$ \$*l*-*?x* $*$ \$$\mu$-(*?x* $-$ *1*) (**is** *?LHS* $\longleftrightarrow$ *?RHS*)
proof
assume *?LHS*
hence (*l has-real-derivative* ($-$ \$*l*-*?x* $*$ \$$\mu$-*?x*)) (*at-left ?x*) **by** *simp*
moreover have (*l has-real-derivative* ($-$ \$*l*-*?x* $*$ \$$\mu$-(*?x* $-$ *1*))) (*at-left ?x*)
using *that exponential-l-has-left-derivative-at-nat* **by** *force*
ultimately show *?RHS*
using *has-real-derivative-iff-has-vector-derivative vector-derivative-unique-within trivial-limit-at-left-real* **by** *blast*
next
assume *?RHS*
thus *?LHS*
using *that exponential-l-has-right-derivative-at-nat exponential-l-has-left-derivative-at-nat* **by** *force*
qed
also have ... $\longleftrightarrow$ \$$\mu$-*?x* $=$ \$$\mu$-(*?x* $-$ *1*) **by** *simp*
finally show *?thesis* **.**
qed

lemma *exponential-l-differentiable-at-nat-iff-mu*:
l differentiable at x $\longleftrightarrow$ \$$\mu$-*x* $=$ \$$\mu$-(*real x* $-$ *1*)
if *exponential-interpolation x+1* $<$ \$$\psi$ **for** *x*::*nat*
proof
let *?x* $=$ *real x*
assume *l differentiable at ?x*
from *this* **obtain** *D* **where** *DERIV-l*: (*l has-real-derivative D*) (*at ?x*)
using *real-differentiable-def* **by** *blast*
hence (*l has-real-derivative D*) (*at-right ?x*)
using *has-field-derivative-at-within* **by** *blast*
moreover have *at ?x within* {*real x*<..} $\neq$ $\bot$ **by** *simp*
moreover have (*l has-real-derivative* ($-$ \$*l*-*real x* $*$ \$$\mu$-*real x*)) (*at-right ?x*)
using *exponential-l-has-right-derivative-at-nat that* **by** *simp*
ultimately have *D* $=$ $-$ \$*l*-*?x* $*$ \$$\mu$-*?x*
using *that has-real-derivative-iff-has-vector-derivative vector-derivative-unique-within*
by *blast*
thus \$$\mu$-*?x* $=$ \$$\mu$-(*?x* $-$ *1*)
using *exponential-l-has-derivative-at-nat-iff-mu that DERIV-l* **by** *blast*
next
assume \$$\mu$-(*real x*) $=$ \$$\mu$-(*real x* $-$ *1*)
thus *l differentiable at* (*real x*)
using *exponential-l-has-derivative-at-nat-iff-mu that real-differentiable-def* **by** *blast*
qed

lemma *exponential-L-d-mu*: \$*L*-*x* $=$ \$*d*-*x* / \$$\mu$-*x*
if *exponential-interpolation* \$$\mu$-*x* $\neq$ *0 x+1* $<$ \$$\psi$ **for** *x*::*nat*
proof $-$
have [*simp*]: *ereal* (*real x*) $<$ \$$\psi$ **using** *that ereal-less-le* **by** *simp*

```
  have [simp]: $l-(real x) ≠ 0 by simp
  have p-pos[simp]: $p-(real x) > 0 using that by (simp add: exponential-p-mu)
  have [simp]: $p-(real x) ≠ 1 using that exponential-p-mu by simp
  have $L-(real x) = (LBINT t:{0..1}. $l-(real x + t))
    unfolding lives-def by (rewrite lborel-set-integral-Icc-shift[where t=x]) simp
  also have ... = integral {0..1} (λt. $l-(real x) * ($p-(real x)).^t)
    apply (rule integral-cong)
    using that by (rewrite exponential-l-p; simp)
  also have ... = $l-(real x) * integral {0..1} (λt. ($p-(real x)).^t)
    using integral-mult-right by blast
  also have ... = $l-(real x) * ($q-(real x) / - ln ($p-(real x)))
  proof -
   have integral {0..1} (λt. ($p-(real x)).^t) = (($p-(real x)).^1 - 1) / ln ($p-(real
x))
     apply (rule integral-unique)
     by (intro has-integral-powr2-from-0; simp)
    also have ... = $q-(real x) / - ln ($p-(real x))
     using p-pos apply (rewrite powr-one, linarith)
     using that by (rewrite in - - ⌑ p-q-1[of x 1, THEN sym]; simp)
    finally show ?thesis by simp
  qed
 also have ... = $d-(real x) / $μ-(real x) using that exponential-mu-p by (rewrite
q-d-l; simp)
  finally show ?thesis .
qed

lemma exponential-mx-mu: $m-x = $μ-x if exponential-interpolation x+1 < $ψ
for x::nat
proof (cases ‹$μ-(real x) = 0›)
  have lx-neq0: $l-(real x) ≠ 0 using ereal-less-le that by simp
  case True
  hence $q-(real x) = 0 using exponential-q-mu that by simp
  hence $d-(real x) = 0
    using q-1-d-l that by (metis lx-neq0 d-old-0 divide-eq-0-iff linorder-not-less
zero-le-one)
  hence $m-(real x) = 0 unfolding die-central-def by simp
  also have ... = $μ-(real x) using True by simp
  finally show ?thesis .
next
  case False
  thus ?thesis unfolding die-central-def using exponential-L-d-mu that
   by (smt (verit) divide-eq-0-iff divide-mult-cancel exp-eq-one-iff exponential-q-mu
       linorder-not-less mu-beyond-0 nonzero-mult-div-cancel-left q-1-d-l)
qed

lemma exponential-d-mu-sums-T: (λk. $d-(x+k) / $μ-(x+k)) sums $T-x
  if exponential-interpolation total-finite ⋀k::nat. $μ-(x+k) ≠ 0 for x::nat
```

proof −
have ¬ $\$\psi < \infty$
proof
assume $\$\psi < \infty$
from *this* **obtain** *y*::*nat* **where** $\$\psi < ereal\ (real\ y)$ **using** *less-PInf-Ex-of-nat*
by *fastforce*
hence *xy*: $\$\psi < ereal\ (real\ (x+y))$ **by** (*simp add*: *less-ereal-le*)
hence $\$\mu\text{-}(real\ (x+y)) \neq 0$ **using** *that* **by** *simp*
moreover have $\$\mu\text{-}(real\ (x+y)) = 0$ **using** *xy mu-beyond-0* **by** *simp*
ultimately show *False* **..**
qed
hence $\$\psi = \infty$ **by** *simp*
moreover hence $\bigwedge k$::*nat*. $\$d\text{-}(real\ (x+k)) / \$\mu\text{-}(real\ (x+k)) = \$L\text{-}(real\ (x+k))$
using *that* **by** (*rewrite exponential-L-d-mu*; *simp*)
ultimately show *?thesis*
apply (*rewrite sums-cong*; *simp*)
by (*rule L-sums-T*, *simp add*: *that*)
qed

lemma *exponential-e-d-l-mu*: $(\lambda k.\ \$d\text{-}(x+k) / (\$l\text{-}x * \$\mu\text{-}(x+k)))$ *sums* $\$e\text{`}\circ\text{-}x$
if *exponential-interpolation total-finite* $\bigwedge k$::*nat*. $\$\mu\text{-}(x+k) \neq 0$ **for** *x*::*nat*
proof −
let *?x* = *real x*
have ¬ *ereal ?x* ≥ $\$\psi$ **using** *that mu-beyond-0* **by** (*metis add-cancel-right-right*)
hence [*simp*]: *ereal ?x* < $\$\psi$ **by** *simp*
have $(\lambda k.\ \$d\text{-}(real\ (x+k)) / \$\mu\text{-}(real\ (x+k)) / \$l\text{-}?x)$ *sums* $(\$T\text{-}?x / \$l\text{-}?x)$
using *sums-divide exponential-d-mu-sums-T that* **by** *force*
thus *?thesis* **by** (*rewrite e-T-l*; *simp add*: *mult.commute*)
qed

end

6.5 Limited Life Table

locale *limited-life-table* = *life-table* +
assumes *l-limited*: $\exists x$::*real*. $\$l\text{-}x = 0$
begin

lemma *limited-survival-function-MM-X*: *limited-survival-function* $\mathfrak{M}$ *X*
proof (*rule limited-survival-function.intro*)
show *survival-model* $\mathfrak{M}$ *X* **by** (*rule survival-model-MM-X*)
show *limited-survival-function-axioms* $\mathfrak{M}$ *X*
unfolding *limited-survival-function-axioms-def* **using** *l-limited death-pt-def* **by**
fastforce
qed

end

sublocale *limited-life-table* ⊆ *limited-survival-function* $\mathfrak{M}$ *X*

by (*rule limited-survival-function-MM-X*)

context *limited-life-table*
begin

notation *ult-age* (‹$\omega›)

lemma *l-omega-0*: $*l*-$\omega = 0$
using *ccdfX-l-normal ccdfX-omega-0* **by** *simp*

lemma *l-0-equiv-nat*: $*l*-x = 0 \longleftrightarrow x \geq \omega$ **for** *x*::*nat*
using *ccdfX-l-normal ccdfX-0-equiv-nat* **by** *simp*

lemma *d-l-equiv-nat*: $*d*-\{t\&x\} = *l*-x \longleftrightarrow x+t \geq \omega$ **if** $t \geq 0$ **for** $x\ t$ :: *nat*
by (*metis d-l-equiv of-nat-0-le-iff of-nat-add psi-le-iff-omega-le*)

corollary *d-1-omega-l*: $*d*-(\omega-1) = *l*-(\omega-1)$
using *d-l-equiv-nat*[*of 1* $\omega-1$] *omega-pos* **by** *simp*

lemma *limited-life-table-imp-total-finite*: *total-finite*
proof −
have $\{0..\} = \{0\ ..\ real\ \omega\} \cup \{real\ \omega <..\}$ **by** *force*
moreover have *set-integrable lborel* $\{0\ ..\ real\ \omega\}$ *l* **by** (*rule l-integrable-Icc*)
moreover have *set-integrable lborel* $\{real\ \omega <..\}$ *l*
apply (*rewrite set-integrable-cong*[**where** $f'=\lambda$-. *0*], *simp-all*)
using *l-0-equiv-nat* **apply** (*meson l-0-equiv le-ereal-le order-le-less*)
unfolding *set-integrable-def* **by** *simp*
ultimately have *set-integrable lborel* $\{0..\}$ *l*
using *set-integrable-Un*
by (*smt* (*verit*, *del-insts*) *Ici-subset-Ioi-iff add-mono-thms-linordered-field*(*1*)
atLeast-borel l-0-pos set-integrable-subset sets-lborel total-finite-iff-set-integrable-Ici)
thus *?thesis* **unfolding** *total-finite-def* **by** *simp*
qed

context
fixes *x*::*nat*
assumes *x-lt-omega*[*simp*]: $x < \omega$
begin

lemma *curt-e-sum-l-finite-nat*: $*e*-x = (\sum k<n.\ *l*-(x+k+1)) / *l*-x$
if $\bigwedge k$::*nat*. $k < n \Longrightarrow$ *isCont l* $(x+k+1)$ $x+n \geq \omega$ **for** *n*::*nat*
apply (*rewrite curt-e-sum-l-finite*[*of x n*], *simp*)
using *that le-ereal-less psi-le-omega* **apply** (*metis of-nat-1 of-nat-add*, *force*)
by (*simp add*: *add.commute*)

end

end

end
theory *Examples*
imports *Life-Table*
begin

7 Examples

The following lemma is a verification of the solution to the multiple choice question No. 3 of Exam LTAM Spring 2022 by Society of Actuaries.

context *smooth-survival-function*
begin

lemma *SoA-LTAM-2022-Spring-MCQ-No3*:
assumes $\bigwedge x$::*real*. $0 \leq x \Longrightarrow x \leq 100 \Longrightarrow$ *ccdf* (*distr* $\mathfrak{M}$ *borel X*) $x = (1 - 0.01{*}x).\hat{}0.5$
shows $|1000{*}\$\mu\text{-}25 - 6.7| < 0.05$
proof −
let *?svl* = *ccdf* (*distr* $\mathfrak{M}$ *borel X*)
have [*simp*]: *ereal 25* < $\$\psi$
apply (*rewrite not-le*[*THEN sym*])
using *assms* **by** (*rewrite ccdfX-0-equiv*[*THEN sym*]) *simp*
have ⋆: ((λ*x*. *ln* (*1* − *x*/*100*)) *has-real-derivative* (−*1*/*75*)) (*at 25*)
proof −
have ((λ*x*. (*1* − *x*/*100*)) *has-real-derivative* (*0* − *1*/*100*)) (*at 25*)
apply (*intro derivative-intros*)
by (*rule DERIV-cdivide*) *simp*
hence ((λ*x*. *ln* (*1* − *x*/*100*)) *has-real-derivative* (*1* / (*1* − *25*/*100*) ∗ (−*1*/*100*))) (*at 25*)
by (*intro derivative-intros*) *auto*
thus *?thesis* **by** *simp*
qed
have *exp* ∘ (λ*x*::*real*. *0.5* ∗ *ln* (*1* − *0.01*∗*x*)) *field-differentiable at 25*
apply (*rule field-differentiable-compose*, *simp-all*)
apply (*rule derivative-intros*, *simp-all*)
using ⋆ *field-differentiable-def* **apply** *blast*
using *field-differentiable-within-exp* **by** *blast*
hence *?svl differentiable at 25*
apply (*rewrite differentiable-eq-field-differentiable-real*)
by (*rule field-differentiable-transform-within*[**where** *d*=*1*])
(*simp-all add*: *powr-def dist-real-def assms*)
hence $\$\mu\text{-}25 = -$ *deriv* (λ*x*. *ln* (*?svl x*)) *25* **by** (*rule mu-deriv-ln*; *simp*)
also have ... = *0.005* / (*1* − *0.01*∗*25*)
proof −
have ∀ *F x in nhds 25*. *ln* (*?svl x*) ∗ *2* = *ln* (*1* − *x*/*100*)
proof −
{ fix *x*::*real* **assume** *dist x 25* < *1*
hence *asm*: *0* < *x x* < *100* **using** *dist-real-def* **by** *auto*
hence *ln* (*?svl x*) ∗ *2* = *ln* ((*1* − *0.01*∗*x*).^*0.5*) ∗ *2* **using** *assms* **by** (*smt*

(*verit*))
also have ... = (*0.5* ∗ *ln* (*1* − *0.01*∗*x*)) ∗ *2* **using** *asm* **by** (*rewrite ln-powr*) *auto*
finally have *ln* (*?svl x*) ∗ *2* = *ln* (*1* − *x*/*100*) **by** *simp* **}**
thus *?thesis* **by** (*rewrite eventually-nhds-metric*) (*smt* (*verit*, *del-insts*))
qed
hence ((λ*x*. *ln* (*?svl x*)) *has-real-derivative* (−*0.005* / (*1* − *0.01*∗*25*))) (*at 25*)
using ⋆ **apply** (*rewrite DERIV-cong-ev*[**where** *g*=λ*x*. *0.5* ∗ *ln* (*1* − *0.01*∗*x*)], *simp-all*)
by (*rule derivative-eq-intros*) *auto*
thus *?thesis* **using** *DERIV-imp-deriv* **by** *force*
qed
finally show *?thesis* **by** *simp*
qed

end

The following lemma is a verification of the solution to the problem No. 2. (1)-1 of Life Insurance Mathematics 2016 by the Institute of Actuaries of Japan, slightly modified; see the remark below.

context *smooth-life-table*
begin

lemma *IAJ-Life-Insurance-Math-2016-2-1-1*:
fixes *a b* :: *real*
assumes −*1* < *a a* < *0 0* < *b* −*b*/*a* ≤ $ψ **and**
total-finite **and**
⋀*x*. *0* < *x* ⟹ *x* < −*b*/*a* ⟹ *l differentiable at x* **and**
⋀*x*. *0* ≤ *x* ⟹ *x* < −*b*/*a* ⟹ $*e*‘∘-*x* = *a*∗*x* + *b*
shows ⋀*x*. *0* ≤ *x* ⟹ *x* < −*b*/*a* ⟹ $*l*-*x* = $*l*-*0* ∗ (*b* / (*a*∗*x* + *b*)).^((*a*+*1*)/*a*)
proof −
fix *x* **assume** *asm-x*: *0* ≤ *x x* < −*b*/*a*
hence *x-lt-psi*[*simp*]: *ereal x* < $ψ **using** *assms ereal-le-less* **by** *presburger*
from *asm-x* **have** *axb-pos*[*simp*]: *a*∗*x* + *b* > *0*
using *assms* **by** (*smt* (*verit*, *ccfv-threshold*) *mult.commute neg-less-divide-eq*)
have *mu*: ⋀*t*. *t*∈{*0*<..<−*b*/*a*} ⟹ $μ-*t* = (*a* + *1*) / (*a*∗*t* + *b*)
proof −
fix *t* **assume** *asm-t*: *t*∈{*0*<..<−*b*/*a*}
with *assms* **have** ((λ*u*. $*e*‘∘-*u*) *has-real-derivative* ($μ-*t* ∗ $*e*‘∘-*t* − *1*)) (*at t*)
by (*intro e-has-derivative-mu-e-l′*[**where** *a*=*0*]; *simp*)
moreover have ((λ*u*. $*e*‘∘-*u*) *has-real-derivative a*) (*at t*)
proof −
let *?d* = *min t* (−*b*/*a* − *t*)
have *?d* > *0* **using** *assms asm-t* **by** *simp*
moreover have ⋀*u*. *dist u t* < *?d* ⟹ $*e*‘∘-*u* = *a*∗*u* + *b* **using** *assms dist-real-def* **by** *auto*
ultimately have ∀$_F$ *u in nhds t*. $*e*‘∘-*u* = *a*∗*u* + *b* **by** (*rewrite eventually-nhds-metric*) *blast*
thus *?thesis*

```
    using assms apply (rewrite DERIV-cong-ev[where g=λt. a*t + b], simp-all)
      by (intro derivative-eq-intros) auto
  qed
  ultimately have $μ-t * $e‘∘-t - 1 = a using DERIV-unique by blast
  moreover have $e‘∘-t = a*t + b using assms asm-t by simp
  ultimately show $μ-t = (a + 1) / (a*t + b)
    using assms by (smt (verit) mult-eq-0-iff nonzero-mult-div-cancel-right)
qed
hence $p-{x&0} = (b / (a*x + b)).^((a+1)/a)
proof −
  have integ-mu: integral {0..x} (λt. $μ-t) = (a + 1) / a * ln ((a*x + b) / b)
  proof −
    have 0: negligible ({0..x} - {0<..x})
      by (simp add: Icc-minus-Ioc asm-x(1))
    from integral-subset-negligible [OF - this]
    have integral {0..x} (λt. $μ-t) = integral {0<..x} (λt. $μ-t)
      by (metis greaterThanAtMost-subseteq-atLeastAtMost-iff order.refl)
    also have ... = integral {0<..x} (λt. ((a + 1) / a) * (a / (a*t + b)))
      using asm-x assms by (intro integral-cong) (rewrite mu; simp)
    also have ... = (a + 1) / a * integral {0<..x} (λt. a / (a*t + b))
      by (metis integral-mult-right)
    also have ... = (a + 1) / a * ln ((a*x + b) / b)
    proof −
      have integral {0<..x} (λt. a / (a*t + b)) = integral {0..x} (λt. a / (a*t +
b))
        using integral-subset-negligible [OF - 0] by force
      also have ... = ln (a*x + b) − ln (a*0 + b)
        apply (rule integral-unique)
        using assms asm-x apply (intro inverse-fun-has-integral-ln, simp-all)
        using axb-pos assms apply (smt (verit) mult-less-cancel-left)
        apply (intro continuous-intros)
        by (intro derivative-eq-intros) auto
      also have ... = ln ((a*x + b) / b) using assms by (simp add: ln-divide-pos)
      finally have integral {0<..x} (λt. a / (a*t + b)) = ln ((a*x + b) / b) .
      thus ?thesis by simp
    qed
    finally show ?thesis .
  qed
  thus ?thesis
    apply (rewrite p-exp-integral-mu, simp-all add: asm-x)
    unfolding powr-def using assms
    by simp (smt (verit) axb-pos ln-div
        nonzero-minus-divide-divide nonzero-minus-divide-right right-diff-distrib')
qed
thus $l-x = $l-0 * (b / (a*x + b)).^((a+1)/a)
  using assms asm-x apply (rewrite in asm p-l, simp-all)
  by (metis divide-mult-cancel l-0-neq-0 mult.commute)
qed
```

REMARK. The original problem lacks the following hypotheses: (i) $0 < b$,

(ii) $-b/a \leq \$\psi$, (iii) $\forall\, x.\ 0 < x < -b/a \longrightarrow l$ *differentiable at* x, (iv) $\forall\, x.\ 0 \leq x < -b/a \longrightarrow l$ *integrable-on* $\{x..\}$. Moreover, the hypothesis $\forall\, x.\ 0 \leq x < -b/a$ is originally $\forall\, x.\ 0 \leq x \leq -b/a$.

end

end