

# On the Formalization of Martingales

Ata Keskin

November 17, 2023

## Abstract

In the scope of this project, we present a formalization of martingales in arbitrary Banach spaces using Isabelle/HOL [2].

The current formalization of conditional expectation in the Isabelle library is limited to real-valued functions. To overcome this limitation, we extend the construction of conditional expectation to general Banach spaces, employing an approach similar to the one described in [1]. We use measure theoretic arguments to construct the conditional expectation using suitable limits of simple functions.

Subsequently, we define stochastic processes and introduce the concepts of adapted, progressively measurable and predictable processes using suitable locale definitions<sup>1</sup>. We show the relation

$$\text{adapted} \supseteq \text{progressive} \supseteq \text{predictable}$$

Furthermore, we show that progressive measurability and adaptedness are equivalent when the indexing set is discrete. We pay special attention to predictable processes in discrete-time, showing that  $(X_n)_{n \in \mathbb{N}}$  is predictable if and only if  $(X_{n+1})_{n \in \mathbb{N}}$  is adapted.

Moving forward, we rigorously define martingales, submartingales, and supermartingales, presenting their first consequences and corollaries<sup>2</sup>. Discrete-time martingales are given special attention in the formalization. In every step of our formalization, we make extensive use of the powerful locale system of Isabelle.

The formalization further contributes by generalizing concepts in Bochner integration by extending their application from the real numbers to arbitrary Banach spaces equipped with a second-countable topology. Induction schemes for integrable simple functions on Banach spaces are introduced, accommodating various scenarios with or without a real vector ordering<sup>3</sup>. Specifically, we formalize a powerful result called the “Averaging Theorem”[4] which allows us to show that densities are unique in Banach spaces.

In depth information on the formalization and the proofs of the individual theorems can be found in [3].

---

<sup>1</sup>Martingale.Stochastic\_Process in [2]

<sup>2</sup>Martingale.Martingale in [2]

<sup>3</sup>Martingale.Bochner\_Integration\_Addendum in [2]

## Contents

<b>1</b>	<b>Supplementary Lemmas for Measure Spaces</b>	<b>4</b>
1.1	$\sigma$ -Algebra Generated by a Family of Functions . . . . .	4
<b>2</b>	<b>Supplementary Lemmas for Elementary Metric Spaces</b>	<b>5</b>
2.1	Diameter Lemma . . . . .	5
<b>3</b>	<b>Supplementary Lemmas for Bochner Integration</b>	<b>7</b>
3.1	Integrable Simple Functions . . . . .	7
3.2	Totally Ordered Banach Spaces . . . . .	16
3.3	Integrability and Measurability of the Diameter . . . . .	19
3.4	Auxiliary Lemmas for Set Integrals . . . . .	20
3.5	Averaging Theorem . . . . .	21
<b>4</b>	<b>Conditional Expectation in Banach Spaces</b>	<b>27</b>
4.1	Existence . . . . .	31
4.2	Properties . . . . .	41
4.3	Linearly Ordered Banach Spaces . . . . .	47
4.4	Probability Spaces . . . . .	51
<b>5</b>	<b>Filtered Measure Spaces</b>	<b>57</b>
5.1	Filtered Measure . . . . .	57
5.2	$\sigma$ -Finite Filtered Measure . . . . .	58
5.3	Finite Filtered Measure . . . . .	58
5.4	Constant Filtration . . . . .	58
<b>6</b>	<b>Stochastic Processes</b>	<b>59</b>
6.1	Stochastic Process . . . . .	59
6.1.1	Natural Filtration . . . . .	61
6.2	Adapted Process . . . . .	63
6.3	Progressively Measurable Process . . . . .	66
6.4	Predictable Process . . . . .	69
<b>7</b>	<b>Martingales</b>	<b>78</b>
7.1	Additional Locale Definitions . . . . .	78
7.2	Martingale . . . . .	80
7.3	Submartingale . . . . .	81
7.4	Supermartingale . . . . .	81
7.5	Martingale Lemmas . . . . .	81
7.6	Submartingale Lemmas . . . . .	84
7.7	Supermartingale Lemmas . . . . .	88
7.8	Discrete Time Martingales . . . . .	91
7.9	Discrete Time Submartingales . . . . .	93
7.10	Discrete Time Supermartingales . . . . .	95



```

theory Measure-Space-Supplement
  imports HOL-Analysis.Measure-Space
begin

```

## 1 Supplementary Lemmas for Measure Spaces

### 1.1 $\sigma$ -Algebra Generated by a Family of Functions

**definition** *family-vimage-algebra* :: '*a* set  $\Rightarrow$  ('*a*  $\Rightarrow$  '*b*) set  $\Rightarrow$  '*b* measure  $\Rightarrow$  '*a* measure **where**  
*family-vimage-algebra*  $\Omega$  *S* *M*  $\equiv$  *sigma*  $\Omega$  ( $\bigcup f \in S. \{f - 'A \cap \Omega \mid A. A \in M\}$ )

For singleton *S*, i.e.  $S = \{f\}$  for some *f*, the definition simplifies to that of *vimage-algebra*.

**lemma** *family-vimage-algebra-singleton*: *family-vimage-algebra*  $\Omega$   $\{f\}$  *M* = *vimage-algebra*  $\Omega$  *f* *M* **unfolding** *family-vimage-algebra-def* *vimage-algebra-def* **by** *simp*

**lemma**

**shows** *sets-family-vimage-algebra*: *sets* (*family-vimage-algebra*  $\Omega$  *S* *M*) = *sigma-sets*  $\Omega$  ( $\bigcup f \in S. \{f - 'A \cap \Omega \mid A. A \in M\}$ )  
**and** *space-family-vimage-algebra*[*simp*]: *space* (*family-vimage-algebra*  $\Omega$  *S* *M*) =  $\Omega$   
**by** (*auto simp add: family-vimage-algebra-def sets-measure-of-conv space-measure-of-conv*)

**lemma** *measurable-family-vimage-algebra*:

**assumes**  $f \in S$   $f \in \Omega \rightarrow \text{space } M$   
**shows**  $f \in \text{family-vimage-algebra } \Omega$  *S* *M*  $\rightarrow_M M$   
**using** *assms* **by** (*intro measurableI, auto simp add: sets-family-vimage-algebra*)

**lemma** *measurable-family-vimage-algebra-singleton*:

**assumes**  $f \in \Omega \rightarrow \text{space } M$   
**shows**  $f \in \text{family-vimage-algebra } \Omega$   $\{f\}$  *M*  $\rightarrow_M M$   
**using** *assms* *measurable-family-vimage-algebra* **by** *blast*

A collection of functions are measurable with respect to some  $\sigma$ -algebra *N*, if and only if the  $\sigma$ -algebra they generate is contained in *N*.

**lemma** *measurable-family-iff-sets*:

**shows** ( $S \subseteq N \rightarrow_M M$ )  $\longleftrightarrow$   $S \subseteq \text{space } N \rightarrow \text{space } M \wedge \text{family-vimage-algebra} (\text{space } N) S M \subseteq N$

**proof** (*standard, goal-cases*)

**case** 1

**hence** *subset*:  $S \subseteq \text{space } N \rightarrow \text{space } M$  **using** *measurable-space* **by** *fast*

**have**  $\{f - 'A \cap \text{space } N \mid A. A \in M\} \subseteq N$  **if**  $f \in S$  **for** *f* **using** *measurable-iff-sets*[*unfolded family-vimage-algebra-singleton*[*symmetric*], *of f*] 1 *subset that*  
**by** (*fastforce simp add: sets-family-vimage-algebra*)

**then show** ?*case* **unfolding** *sets-family-vimage-algebra* **using** *sets.sigma-algebra-axioms*  
**by** (*simp add: subset, intro sigma-algebra.sigma-sets-subset, blast+*)

```

next
  case 2
  hence subset:  $S \subseteq \text{space } N \rightarrow \text{space } M$  by simp
  show ?case
  proof (standard, goal-cases)
    case (1 x)
    have family-vimage-algebra ( $\text{space } N$ )  $\{x\}$   $M \subseteq N$  by (metis (no-types, lifting)
1 2 sets-family-vimage-algebra SUP-le-iff sigma-sets-le-sets-iff singletonD)
    thus ?case using measurable-iff-sets[unfolded family-vimage-algebra-singleton[symmetric]]
subset[THEN subsetD, OF 1] by fast
  qed
qed

lemma family-vimage-algebra-diff:
  shows family-vimage-algebra  $\Omega$   $S$   $M = \text{sigma } \Omega$  (sets (family-vimage-algebra  $\Omega$ 
( $S - I$ )  $M$ )  $\cup$  family-vimage-algebra  $\Omega$  ( $S \cap I$ )  $M$ )
  using sets.space-closed space-measure-of-conv
  unfolding family-vimage-algebra-def sets-family-vimage-algebra
  by (intro sigma-eqI, blast, fastforce)
    (intro sigma-sets-eqI, blast, simp add: sets-measure-of-conv split: if-splits,
meson Diff-subset Sup-subset-mono in-mono inf-sup-ord(1) sigma-sets-subseteq
subset-image-iff, fastforce+)

end
theory Elementary-Metric-Spaces-Supplement
  imports HOL-Analysis.Elementary-Metric-Spaces
begin

```

## 2 Supplementary Lemmas for Elementary Metric Spaces

### 2.1 Diameter Lemma

```

lemma diameter-comp-strict-mono:
  fixes  $s :: \text{nat} \Rightarrow 'a :: \text{metric-space}$ 
  assumes strict-mono  $r$  bounded  $\{s\ i \mid i. r\ n \leq i\}$ 
  shows diameter  $\{s\ (r\ i) \mid i. n \leq i\} \leq \text{diameter } \{s\ i \mid i. r\ n \leq i\}$ 
proof (rule diameter-subset)
  show  $\{s\ (r\ i) \mid i. n \leq i\} \subseteq \{s\ i \mid i. r\ n \leq i\}$  using assms(1) monotoneD
strict-mono-mono by fastforce
qed (intro assms(2))

lemma diameter-bounded-bound':
  fixes  $S :: 'a :: \text{metric-space}$  set
  assumes  $S$ : bdd-above (case-prod dist ' ( $S \times S$ ))  $x \in S\ y \in S$ 
  shows dist  $x\ y \leq \text{diameter } S$ 
proof -
  have  $(x,y) \in S \times S$  using  $S$  by auto
  then have dist  $x\ y \leq (\text{SUP } (x,y) \in S \times S. \text{dist } x\ y)$  by (rule cSUP-upper2[OF

```

```

assms(1)]) simp
  with  $\langle x \in S \rangle$  show ?thesis by (auto simp: diameter-def)
qed

```

```

lemma bounded-imp-dist-bounded:
  assumes bounded (range s)
  shows bounded (( $\lambda(i, j). \text{dist } (s\ i) (s\ j)$ ) ' ( $\{n..\} \times \{n..\}$ ))
  using bounded-dist-comp[OF bounded-fst bounded-snd, OF bounded-Times(1,1)[OF
assms(1,1)]] by (rule bounded-subset, force)

```

A sequence is Cauchy, if and only if it is bounded and it's diameter tends to zero. The diameter is well-defined only if the sequence is bounded.

```

lemma cauchy-iff-diameter-tends-to-zero-and-bounded:
  fixes s :: nat  $\Rightarrow$  'a :: metric-space
  shows Cauchy s  $\longleftrightarrow$  (( $\lambda n. \text{diameter } \{s\ i \mid i. i \geq n\}$ )  $\longrightarrow$  0  $\wedge$  bounded (range s))
proof -
  have {s i | i. N  $\leq$  i}  $\neq$  {} for N by blast
  hence diameter-SUP: diameter {s i | i. N  $\leq$  i} = (SUP (i, j)  $\in$  {N..}  $\times$  {N..}.
dist (s i) (s j)) for N unfolding diameter-def by (auto intro!: arg-cong[of - - Sup])
  show ?thesis
  proof (intro iffI)
    assume asm: Cauchy s
    have  $\exists N. \forall n \geq N. \text{norm } (\text{diameter } \{s\ i \mid i. n \leq i\}) < e$  if e-pos: e > 0 for e
    proof -
      obtain N where dist-less: dist (s n) (s m) < (1/2) * e if n  $\geq$  N m  $\geq$  N
      for n m using asm e-pos by (metis Cauchy-def mult-pos-pos zero-less-divide-iff
zero-less-numeral zero-less-one)
      {
        fix r assume r  $\geq$  N
        hence dist (s n) (s m) < (1/2) * e if n  $\geq$  r m  $\geq$  r for n m using dist-less
        that by simp
        hence (SUP (i, j)  $\in$  {r..}  $\times$  {r..}. dist (s i) (s j))  $\leq$  (1/2) * e by (intro
cSup-least) fastforce+
        also have ... < e using e-pos by simp
        finally have diameter {s i | i. r  $\leq$  i} < e using diameter-SUP by presburger
      }
      moreover have diameter {s i | i. r  $\leq$  i}  $\geq$  0 for r unfolding diameter-SUP
      using bounded-imp-dist-bounded[OF cauchy-imp-bounded, THEN bounded-imp-bdd-above,
OF asm] by (intro cSup-upper2, auto)
      ultimately show ?thesis by auto
    qed
    thus ( $\lambda n. \text{diameter } \{s\ i \mid i. n \leq i\}$ )  $\longrightarrow$  0  $\wedge$  bounded (range s) using
cauchy-imp-bounded[OF asm] by (simp add: LIMSEQ-iff)
  next
    assume asm: ( $\lambda n. \text{diameter } \{s\ i \mid i. n \leq i\}$ )  $\longrightarrow$  0  $\wedge$  bounded (range s)
    have  $\exists N. \forall n \geq N. \forall m \geq N. \text{dist } (s\ n) (s\ m) < e$  if e-pos: e > 0 for e
    proof -
      obtain N where diam-less: diameter {s i | i. r  $\leq$  i} < e if r  $\geq$  N for r

```

```

using LIMSEQ-D asm(1) e-pos by fastforce
{
  fix n m assume  $n \geq N$   $m \geq N$ 
  hence  $\text{dist } (s\ n) (s\ m) < e$  using cSUP-lessD[OF bounded-imp-dist-bounded[THEN
bounded-imp-bdd-above], OF - diam-less[unfolded diameter-SUP]] asm by auto
}
thus ?thesis by blast
qed
then show Cauchy s by (simp add: Cauchy-def)
qed
qed
end

```

```

theory Bochner-Integration-Supplement
imports HOL-Analysis.Bochner-Integration HOL-Analysis.Set-Integral Elementary-Metric-Spaces-Supplement
begin

```

### 3 Supplementary Lemmas for Bochner Integration

#### 3.1 Integrable Simple Functions

We restate some basic results concerning Bochner-integrable functions.

```

lemma integrable-implies-simple-function-sequence:
  fixes  $f :: 'a \Rightarrow 'b :: \{\text{banach, second-countable-topology}\}$ 
  assumes integrable M f
  obtains s where  $\bigwedge i. \text{simple-function } M (s\ i)$ 
    and  $\bigwedge i. \text{emeasure } M \{y \in \text{space } M. s\ i\ y \neq 0\} \neq \infty$ 
    and  $\bigwedge x. x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x$ 
    and  $\bigwedge x\ i. x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$ 
proof -
  have  $f: f \in \text{borel-measurable } M (\int^+ x. \text{norm } (f\ x) \partial M) < \infty$  using assms
unfolding integrable-iff-bounded by auto
  obtain s where  $s: \bigwedge i. \text{simple-function } M (s\ i) \bigwedge x. x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x$ 
     $\bigwedge i\ x. x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$  using
    borel-measurable-implies-sequence-metric[OF f(1)] unfolding norm-conv-dist by
    metis
  {
    fix i
    have  $(\int^+ x. \text{norm } (s\ i\ x) \partial M) \leq (\int^+ x. \text{ennreal } (2 * \text{norm } (f\ x)) \partial M)$  using
    s by (intro nn-integral-mono, auto)
    also have  $\dots < \infty$  using f by (simp add: nn-integral-cmult ennreal-mult-less-top ennreal-mult)
    finally have sbi: Bochner-Integration.simple-bochner-integrable M (s i) using
    s by (intro simple-bochner-integrableI-bounded) auto
    hence  $\text{emeasure } M \{y \in \text{space } M. s\ i\ y \neq 0\} \neq \infty$  by (auto intro: inte-

```

```

grableI-simple-bochner-integrable simple-bochner-integrable.cases)
}
thus ?thesis using that s by blast
qed

```

Simple functions can be represented by sums of indicator functions.

```

lemma simple-function-indicator-representation:
  fixes f :: 'a  $\Rightarrow$  'b :: {second-countable-topology, banach}
  assumes simple-function M f x  $\in$  space M
  shows f x = ( $\sum$  y  $\in$  f ' space M. indicator (f - ' {y}  $\cap$  space M) x *R y)
  (is ?l = ?r)
proof -
  have ?r = ( $\sum$  y  $\in$  f ' space M.
    (if y = f x then indicator (f - ' {y}  $\cap$  space M) x *R y else 0)) by (auto intro!:
sum.cong)
  also have ... = indicator (f - ' {f x}  $\cap$  space M) x *R f x using assms by (auto
dest: simple-functionD)
  also have ... = f x using assms by (auto simp: indicator-def)
  finally show ?thesis by auto
qed

```

```

lemma simple-function-indicator-representation-AE:
  fixes f :: 'a  $\Rightarrow$  'b :: {second-countable-topology, banach}
  assumes simple-function M f
  shows AE x in M. f x = ( $\sum$  y  $\in$  f ' space M. indicator (f - ' {y}  $\cap$  space M) x
*_R y)
  by (metis (mono-tags, lifting) AE-I2 simple-function-indicator-representation
assms)

```

```

lemmas simple-function-scaleR[intro] = simple-function-compose2[where h=(*_R)]

```

```

lemmas integrable-simple-function = simple-bochner-integrable.intros[THEN has-bochner-integral-simple-boch]
THEN integrable.intros]

```

Induction rule for simple integrable functions.

```

lemma integrable-simple-function-induct[consumes 2, case-names cong indicator
add, induct set: simple-function]:
  fixes f :: 'a  $\Rightarrow$  'b :: {second-countable-topology, banach}
  assumes f: simple-function M f emeasure M {y  $\in$  space M. f y  $\neq$  0}  $\neq$   $\infty$ 
  assumes cong:  $\bigwedge$  f g. simple-function M f  $\implies$  emeasure M {y  $\in$  space M. f y  $\neq$ 
0}  $\neq$   $\infty$ 
 $\implies$  simple-function M g  $\implies$  emeasure M {y  $\in$  space M. g y  $\neq$ 
0}  $\neq$   $\infty$ 
 $\implies$  ( $\bigwedge$  x. x  $\in$  space M  $\implies$  f x = g x)  $\implies$  P f  $\implies$  P g
  assumes indicator:  $\bigwedge$  A y. A  $\in$  sets M  $\implies$  emeasure M A <  $\infty \implies$  P ( $\lambda$ x.
indicator A x *_R y)
  assumes add:  $\bigwedge$  f g. simple-function M f  $\implies$  emeasure M {y  $\in$  space M. f y  $\neq$ 
0}  $\neq$   $\infty \implies$ 
    simple-function M g  $\implies$  emeasure M {y  $\in$  space M. g y  $\neq$  0}  $\neq$ 
 $\infty \implies$ 

```



$(\bigwedge z. z \in \text{space } M \implies \text{norm } (f z + g z) = \text{norm } (f z) + \text{norm } (g z)) \implies$   
 $P f \implies P g \implies P (\lambda x. f x + g x)$   
**shows**  $P f$   
**proof** –  
**let**  $?f = \lambda x. (\sum y \in f - \{y\} \cap \text{space } M. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y)$   
**have**  $f\text{-ae-eq}: f x = ?f x$  **if**  $x \in \text{space } M$  **for**  $x$  **using**  $\text{simple-function-indicator-representation}[OF f(1) \text{ that}]$ .  
**moreover have**  $\text{emeasure } M \{y \in \text{space } M. ?f y \neq 0\} \neq \infty$  **by**  $(\text{metis } (\text{no-types}, \text{lifting}) \text{ Collect-cong calculation } f(2))$   
**moreover have**  $P (\lambda x. \sum y \in S. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y)$   
 $\text{simple-function } M (\lambda x. \sum y \in S. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y)$   
 $\text{emeasure } M \{y \in \text{space } M. (\sum x \in S. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x) \neq 0\} \neq \infty$   
**if**  $S \subseteq f - \{x\} \cap \text{space } M$  **for**  $S$  **using**  $\text{simple-functionD}(1)[OF \text{ assms}(1), \text{THEN rev-finite-subset, OF that}]$  **that**  
**proof**  $(\text{induction rule: finite-induct})$   
**case empty**  
**{**  
**case 1**  
**then show**  $?case$  **using**  $\text{indicator}[of \{ \}]$  **by force**  
**next**  
**case 2**  
**then show**  $?case$  **by force**  
**next**  
**case 3**  
**then show**  $?case$  **by force**  
**}**  
**next**  
**case**  $(\text{insert } x F)$   
**have**  $(f - \{x\} \cap \text{space } M) \subseteq \{y \in \text{space } M. f y \neq 0\}$  **if**  $x \neq 0$  **using**  $\text{that}$  **by blast**  
**moreover have**  $\{y \in \text{space } M. f y \neq 0\} = \text{space } M - (f - \{0\} \cap \text{space } M)$   
**by blast**  
**moreover have**  $\text{space } M - (f - \{0\} \cap \text{space } M) \in \text{sets } M$  **using**  $\text{simple-functionD}(2)[OF f(1)]$  **by blast**  
**ultimately have**  $\text{fin-0}: \text{emeasure } M (f - \{x\} \cap \text{space } M) < \infty$  **if**  $x \neq 0$  **using**  $\text{that}$  **by**  $(\text{metis } \text{emeasure-mono } f(2) \text{ infinity-ennreal-def top.not-eq-extremum top-unique})$   
**hence**  $\text{fin-1}: \text{emeasure } M \{y \in \text{space } M. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x \neq 0\} \neq \infty$  **if**  $x \neq 0$  **by**  $(\text{metis } (\text{mono-tags}, \text{lifting}) \text{ emeasure-mono } f(1) \text{ indicator-simps}(2) \text{ linorder-not-less mem-Collect-eq scaleR-eq-0-iff simple-functionD}(2) \text{ subsetI that})$   
  
**have**  $*$ :  $(\sum y \in \text{insert } x F. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y) = (\sum y \in F. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y) + \text{indicat-real } (f - \{x\} \cap \text{space } M) x *_R x$  **for**  $x$  **by**  $(\text{metis } (\text{no-types}, \text{lifting}) \text{ Diff-empty Diff-insert0 add commute insert.hyps}(1) \text{ insert.hyps}(2) \text{ sum.insert-remove})$

```

have **:  $\{y \in \text{space } M. (\sum x \in \text{insert } x \ F. \text{indicat-real } (f - \{x\} \cap \text{space } M) \ y$ 
 $*_R \ x) \neq 0\} \subseteq \{y \in \text{space } M. (\sum x \in F. \text{indicat-real } (f - \{x\} \cap \text{space } M) \ y *_R \ x)$ 
 $\neq 0\} \cup \{y \in \text{space } M. \text{indicat-real } (f - \{x\} \cap \text{space } M) \ y *_R \ x \neq 0\}$  unfolding
* by fastforce
{
  case 1
  hence  $x: x \in f \text{ ' space } M$  and  $F: F \subseteq f \text{ ' space } M$  by auto
  show ?case
  proof (cases  $x = 0$ )
    case True
    then show ?thesis unfolding * using insert(3)[OF F] by simp
  next
    case False
    have norm-argument:  $\text{norm } ((\sum y \in F. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ z$ 
 $*_R \ y) + \text{indicat-real } (f - \{x\} \cap \text{space } M) \ z *_R \ x) = \text{norm } (\sum y \in F. \text{indicat-real}$ 
 $(f - \{y\} \cap \text{space } M) \ z *_R \ y) + \text{norm } (\text{indicat-real } (f - \{x\} \cap \text{space } M) \ z *_R \ x)$ 
if  $z: z \in \text{space } M$  for  $z$ 
    proof (cases  $f \ z = x$ )
      case True
      have  $\text{indicat-real } (f - \{y\} \cap \text{space } M) \ z *_R \ y = 0$  if  $y \in F$  for  $y$  using
True insert(2) z that 1 unfolding indicator-def by force
      hence  $(\sum y \in F. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ z *_R \ y) = 0$  by (meson
sum.neutral)
      then show ?thesis by force
    next
      case False
      then show ?thesis by force
    qed
    show ?thesis using False simple-functionD(2)[OF f(1)] insert(3,5)[OF F]
simple-function-scaleR fin-0 fin-1 by (subst *, subst add, subst simple-function-sum)
(blast intro: norm-argument indicator)+
    qed
  next
    case 2
    hence  $x: x \in f \text{ ' space } M$  and  $F: F \subseteq f \text{ ' space } M$  by auto
    show ?case
    proof (cases  $x = 0$ )
      case True
      then show ?thesis unfolding * using insert(4)[OF F] by simp
    next
      case False
      then show ?thesis unfolding * using insert(4)[OF F] simple-functionD(2)[OF
f(1)] by fast
    qed
  next
    case 3
    hence  $x: x \in f \text{ ' space } M$  and  $F: F \subseteq f \text{ ' space } M$  by auto
    show ?case
    proof (cases  $x = 0$ )

```

```

case True
then show ?thesis unfolding * using insert(5)[OF F] by simp
next
case False
have emeasure M {y ∈ space M. (∑ x ∈ insert x F. indicat-real (f - ' {x} ∩ space M) y *R x) ≠ 0} ≤ emeasure M ({y ∈ space M. (∑ x ∈ F. indicat-real (f - ' {x} ∩ space M) y *R x) ≠ 0} ∪ {y ∈ space M. indicat-real (f - ' {x} ∩ space M) y *R x ≠ 0})
using ** simple-functionD(2)[OF insert(4)[OF F]] simple-functionD(2)[OF f(1)] by (intro emeasure-mono, force+)
also have ... ≤ emeasure M {y ∈ space M. (∑ x ∈ F. indicat-real (f - ' {x} ∩ space M) y *R x) ≠ 0} + emeasure M {y ∈ space M. indicat-real (f - ' {x} ∩ space M) y *R x ≠ 0}
using simple-functionD(2)[OF insert(4)[OF F]] simple-functionD(2)[OF f(1)] by (intro emeasure-subadditive, force+)
also have ... < ∞ using insert(5)[OF F] fin-1[OF False] by (simp add: less-top)
finally show ?thesis by simp
qed
}
qed
moreover have simple-function M (λx. ∑ y ∈ f ' space M. indicat-real (f - ' {y} ∩ space M) x *R y) using calculation by blast
moreover have P (λx. ∑ y ∈ f ' space M. indicat-real (f - ' {y} ∩ space M) x *R y) using calculation by blast
ultimately show ?thesis by (intro cong[OF - - f(1,2)], blast, presburger+)
qed

```

Induction rule for non-negative simple integrable functions

**lemma** *integrable-simple-function-induct-nn[consumes 3, case-names cong indicator add, induct set: simple-function]:*

**fixes** *f :: 'a ⇒ 'b :: {second-countable-topology, banach, linorder-topology, ordered-real-vector}*

**assumes** *f: simple-function M f emeasure M {y ∈ space M. f y ≠ 0} ≠ ∞ ∧ x. x ∈ space M ⇒ f x ≥ 0*

**assumes** *cong: ∧ f g. simple-function M f ⇒ emeasure M {y ∈ space M. f y ≠ 0} ≠ ∞ ⇒ (∧ x. x ∈ space M ⇒ f x ≥ 0) ⇒ simple-function M g ⇒ emeasure M {y ∈ space M. g y ≠ 0} ≠ ∞ ⇒ (∧ x. x ∈ space M ⇒ g x ≥ 0) ⇒ (∧ x. x ∈ space M ⇒ f x = g x) ⇒ P f ⇒ P g*

**assumes** *indicator: ∧ A y. y ≥ 0 ⇒ A ∈ sets M ⇒ emeasure M A < ∞ ⇒ P (λx. indicator A x \*<sub>R</sub> y)*

**assumes** *add: ∧ f g. (∧ x. x ∈ space M ⇒ f x ≥ 0) ⇒ simple-function M f ⇒ emeasure M {y ∈ space M. f y ≠ 0} ≠ ∞ ⇒*

*(∧ x. x ∈ space M ⇒ g x ≥ 0) ⇒ simple-function M g ⇒ emeasure M {y ∈ space M. g y ≠ 0} ≠ ∞ ⇒*

*(∧ z. z ∈ space M ⇒ norm (f z + g z) = norm (f z) + norm (g z)) ⇒*

*P f ⇒ P g ⇒ P (λx. f x + g x)*

**shows** *P f*

**proof**–

**let**  $?f = \lambda x. (\sum_{y \in f^{-1} \text{ space } M}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y)$   
**have**  $f\text{-ae-eq}: f \ x = ?f \ x$  **if**  $x \in \text{space } M$  **for**  $x$  **using** *simple-function-indicator-representation*[*OF*  $f(1)$  *that*] .  
**moreover have**  $\text{emeasure } M \ \{y \in \text{space } M. \ ?f \ y \neq 0\} \neq \infty$  **by** (*metis* (*no-types*, *lifting*) *Collect-cong calculation*  $f(2)$ )  
**moreover have**  $P \ (\lambda x. \sum_{y \in S}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y)$   
 $\text{simple-function } M \ (\lambda x. \sum_{y \in S}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y)$   
 $\text{emeasure } M \ \{y \in \text{space } M. \ (\sum_{x \in S}. \text{indicat-real } (f - \{x\} \cap \text{space } M) \ y \ *_R \ x) \neq 0\} \neq \infty$   
 $\bigwedge x. x \in \text{space } M \implies 0 \leq (\sum_{y \in S}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y)$   
**if**  $S \subseteq f^{-1} \text{ space } M$  **for**  $S$  **using** *simple-functionD*(1)[*OF* *assms*(1), *THEN* *rev-finite-subset*, *OF* *that*] *that*  
**proof** (*induction rule: finite-subset-induct'*)  
**case** *empty*  
{  
**case** 1  
**then show**  $?case$  **using** *indicator*[*of* 0 { $\}$ ] **by** *force*  
**next**  
**case** 2  
**then show**  $?case$  **by** *force*  
**next**  
**case** 3  
**then show**  $?case$  **by** *force*  
**next**  
**case** 4  
**then show**  $?case$  **by** *force*  
}  
**next**  
**case** (*insert*  $x \ F$ )  
**have**  $(f - \{x\} \cap \text{space } M) \subseteq \{y \in \text{space } M. \ f \ y \neq 0\}$  **if**  $x \neq 0$  **using** *that* **by** *blast*  
**moreover have**  $\{y \in \text{space } M. \ f \ y \neq 0\} = \text{space } M - (f - \{0\} \cap \text{space } M)$   
**by** *blast*  
**moreover have**  $\text{space } M - (f - \{0\} \cap \text{space } M) \in \text{sets } M$  **using** *simple-functionD*(2)[*OF*  $f(1)$ ] **by** *blast*  
**ultimately have**  $\text{fin-0}: \text{emeasure } M \ (f - \{x\} \cap \text{space } M) < \infty$  **if**  $x \neq 0$   
**using** *that* **by** (*metis* *emeasure-mono*  $f(2)$  *infinity-ennreal-def* *top.not-eq-extremum* *top-unique*)  
**hence**  $\text{fin-1}: \text{emeasure } M \ \{y \in \text{space } M. \ \text{indicat-real } (f - \{x\} \cap \text{space } M) \ y \ *_R \ x \neq 0\} \neq \infty$  **if**  $x \neq 0$  **by** (*metis* (*mono-tags*, *lifting*) *emeasure-mono*  $f(1)$  *indicator-simps*(2) *linorder-not-less* *mem-Collect-eq* *scaleR-eq-0-iff* *simple-functionD*(2) *subsetI* *that*)

**have** *nonneg-x*:  $x \geq 0$  **using** *insert*  $f$  **by** *blast*  
**have**  $*$ :  $(\sum_{y \in \text{insert } x \ F}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y) =$   
 $(\sum_{y \in F}. \text{indicat-real } (f - \{y\} \cap \text{space } M) \ x \ *_R \ y) + \text{indicat-real } (f - \{x\} \cap$

```

space M) xa *R x for xa by (metis (no-types, lifting) add commute insert.hyps(1)
insert.hyps(4) sum.insert)
  have **: {y ∈ space M. (∑ x ∈ insert x F. indicat-real (f - ' {x} ∩ space M) y
*_R x) ≠ 0} ⊆ {y ∈ space M. (∑ x ∈ F. indicat-real (f - ' {x} ∩ space M) y *_R x)
≠ 0} ∪ {y ∈ space M. indicat-real (f - ' {x} ∩ space M) y *_R x ≠ 0} unfolding
* by fastforce
  {
    case 1
    show ?case
    proof (cases x = 0)
      case True
      then show ?thesis unfolding * using insert by simp
    next
      case False
      have norm-argument: norm ((∑ y ∈ F. indicat-real (f - ' {y} ∩ space M) z
*_R y) + indicat-real (f - ' {x} ∩ space M) z *_R x) = norm (∑ y ∈ F. indicat-real
(f - ' {y} ∩ space M) z *_R y) + norm (indicat-real (f - ' {x} ∩ space M) z *_R x)
if z: z ∈ space M for z
      proof (cases f z = x)
        case True
        have indicat-real (f - ' {y} ∩ space M) z *_R y = 0 if y ∈ F for y using
True insert z that 1 unfolding indicator-def by force
        hence (∑ y ∈ F. indicat-real (f - ' {y} ∩ space M) z *_R y) = 0 by (meson
sum.neutral)
        thus ?thesis by force
      qed (force)
      show ?thesis using False fin-0 fin-1 f norm-argument by (subst *, subst add,
presburger add: insert, intro insert, intro insert, fastforce simp add: indicator-def
intro!: insert(2) f(3), auto intro!: indicator insert nonneg-x)
    qed
  next
    case 2
    show ?case
    proof (cases x = 0)
      case True
      then show ?thesis unfolding * using insert by simp
    next
      case False
      then show ?thesis unfolding * using insert simple-functionD(2)[OF f(1)]
by fast
    qed
  next
    case 3
    show ?case
    proof (cases x = 0)
      case True
      then show ?thesis unfolding * using insert by simp
    next
      case False

```

```

      have emeasure M {y ∈ space M. (∑ x ∈ insert x F. indicat-real (f - ' {x}
      ∩ space M) y *R x) ≠ 0} ≤ emeasure M ({y ∈ space M. (∑ x ∈ F. indicat-real (f
      - ' {x} ∩ space M) y *R x) ≠ 0} ∪ {y ∈ space M. indicat-real (f - ' {x} ∩ space
      M) y *R x ≠ 0})
      using ** simple-functionD(2)[OF insert(6)] simple-functionD(2)[OF f(1)]
      insert.IH(2) by (intro emeasure-mono, blast, simp)
      also have ... ≤ emeasure M {y ∈ space M. (∑ x ∈ F. indicat-real (f - ' {x}
      ∩ space M) y *R x) ≠ 0} + emeasure M {y ∈ space M. indicat-real (f - ' {x} ∩
      space M) y *R x ≠ 0}
      using simple-functionD(2)[OF insert(6)] simple-functionD(2)[OF f(1)]
    by (intro emeasure-subadditive, force+)
      also have ... < ∞ using insert(7) fin-1[OF False] by (simp add: less-top)
      finally show ?thesis by simp
    qed
  next
    case (4 ξ)
    thus ?case using insert nonneg-x f(3) by (auto simp add: scaleR-nonneg-nonneg
    intro: sum-nonneg)
  }
  qed
  moreover have simple-function M (λx. ∑ y ∈ f ' space M. indicat-real (f - ' {y}
  ∩ space M) x *R y) using calculation by blast
  moreover have P (λx. ∑ y ∈ f ' space M. indicat-real (f - ' {y} ∩ space M) x
  *R y) using calculation by blast
  moreover have ∧x. x ∈ space M ⇒ 0 ≤ f x using f(3) by simp
  ultimately show ?thesis by (intro cong[OF - - - f(1,2)], blast, blast, fast)
presburger+
qed

lemma finite-nn-integral-imp-ae-finite:
  fixes f :: 'a ⇒ ennreal
  assumes f ∈ borel-measurable M (∫+x. f x ∂M) < ∞
  shows AE x in M. f x < ∞
proof (rule ccontr, goal-cases)
  case 1
  let ?A = space M ∩ {x. f x = ∞}
  have *: emeasure M ?A > 0 using 1 assms(1) by (metis (mono-tags, lifting)
  assms(2) eventually-mono infinity-ennreal-def nn-integral-noteq-infinite top.not-eq-extremum)
  have (∫+x. f x * indicator ?A x ∂M) = (∫+x. ∞ * indicator ?A x ∂M) by
  (metis (mono-tags, lifting) indicator-inter-arith indicator-simps(2) mem-Collect-eq
  mult-eq-0-iff)
  also have ... = ∞ * emeasure M ?A using assms(1) by (intro nn-integral-cmult-indicator,
  simp)
  also have ... = ∞ using * by fastforce
  finally have (∫+x. f x * indicator ?A x ∂M) = ∞ .
  moreover have (∫+x. f x * indicator ?A x ∂M) ≤ (∫+x. f x ∂M) by (intro
  nn-integral-mono, simp add: indicator-def)
  ultimately show ?case using assms(2) by simp
qed

```

Convergence in L1-Norm implies existence of a subsequence which convergences almost everywhere. This lemma is easier to use than the existing one in *HOL-Analysis.Bochner-Integration*

**lemma** *cauchy-L1-AE-cauchy-subseq*:

**fixes**  $s :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{banach}, \text{second-countable-topology}\}$   
**assumes**  $[\text{measurable}] : \bigwedge n. \text{integrable } M (s \ n)$   
**and**  $\bigwedge e. e > 0 \implies \exists N. \forall i \geq N. \forall j \geq N. \text{LINT } x | M. \text{norm } (s \ i \ x - s \ j \ x) < e$   
**obtains**  $r$  **where** *strict-mono*  $r$  *AE*  $x$  *in*  $M$ . *Cauchy*  $(\lambda i. s \ (r \ i) \ x)$   
**proof** –  
**have**  $\exists r. \forall n. (\forall i \geq r \ n. \forall j \geq r \ n. \text{LINT } x | M. \text{norm } (s \ i \ x - s \ j \ x) < (1 / 2) \wedge n) \wedge (r \ (\text{Suc } n) > r \ n)$   
**proof** (*intro dependent-nat-choice, goal-cases*)  
**case** 1  
**then show** ?case **using** *assms*(2) **by** *presburger*  
**next**  
**case** (2  $x \ n$ )  
**obtain**  $N$  **where** \*:  $\text{LINT } x | M. \text{norm } (s \ i \ x - s \ j \ x) < (1 / 2) \wedge \text{Suc } n$  **if**  $i \geq N \ j \geq N$  **for**  $i \ j$  **using** *assms*(2)[*of*  $(1 / 2) \wedge \text{Suc } n$ ] **by** *auto*  
{  
**fix**  $i \ j$  **assume**  $i \geq \max N \ (\text{Suc } x) \ j \geq \max N \ (\text{Suc } x)$   
**hence**  $\text{integral}^L M (\lambda x. \text{norm } (s \ i \ x - s \ j \ x)) < (1 / 2) \wedge \text{Suc } n$  **using** \* **by** *fastforce*  
}  
**then show** ?case **by** *fastforce*  
**qed**  
**then obtain**  $r$  **where** *strict-mono*: *strict-mono*  $r$  **and**  $\forall i \geq r \ n. \forall j \geq r \ n. \text{LINT } x | M. \text{norm } (s \ i \ x - s \ j \ x) < (1 / 2) \wedge n$  **for**  $n$  **using** *strict-mono-Suc-iff* **by** *blast*  
**hence** *r-is*:  $\text{LINT } x | M. \text{norm } (s \ (r \ (\text{Suc } n)) \ x - s \ (r \ n) \ x) < (1 / 2) \wedge n$  **for**  $n$  **by** (*simp add: strict-mono-leD*)

**define**  $g$  **where**  $g = (\lambda n \ x. (\sum i \leq n. \text{ennreal } (\text{norm } (s \ (r \ (\text{Suc } i)) \ x - s \ (r \ i) \ x))))$

**define**  $g'$  **where**  $g' = (\lambda x. \sum i. \text{ennreal } (\text{norm } (s \ (r \ (\text{Suc } i)) \ x - s \ (r \ i) \ x)))$

**have** *integrable-g*:  $(\int^+ x. g \ n \ x \ \partial M) < 2$  **for**  $n$

**proof** –

**have**  $(\int^+ x. g \ n \ x \ \partial M) = (\int^+ x. (\sum i \leq n. \text{ennreal } (\text{norm } (s \ (r \ (\text{Suc } i)) \ x - s \ (r \ i) \ x))) \ \partial M)$  **using** *g-def* **by** *simp*

**also have** ... =  $(\sum i \leq n. (\int^+ x. \text{ennreal } (\text{norm } (s \ (r \ (\text{Suc } i)) \ x - s \ (r \ i) \ x)) \ \partial M))$  **by** (*intro nn-integral-sum, simp*)

**also have** ... =  $(\sum i \leq n. \text{LINT } x | M. \text{norm } (s \ (r \ (\text{Suc } i)) \ x - s \ (r \ i) \ x))$  **unfolding** *dist-norm* **using** *assms*(1) **by** (*subst nn-integral-eq-integral[OF integrable-norm], auto*)

**also have** ... <  $\text{ennreal } (\sum i \leq n. (1 / 2) \wedge i)$  **by** (*intro ennreal-lessI[OF sum-pos sum-strict-mono[OF finite-atMost - r-is]], auto*)

**also have** ...  $\leq \text{ennreal } 2$  **unfolding** *sum-gp0*[*of*  $1 / 2 \ n$ ] **by** (*intro ennreal-leI, auto*)

**finally show**  $(\int^+ x. g \ n \ x \ \partial M) < 2$  **by** *simp*

**qed**

**have** *integrable-g'*:  $(\int^+ x. g' x \partial M) \leq 2$   
**proof** –  
**have** *incseq*  $(\lambda n. g n x)$  **for**  $x$  **by**  $(\text{intro } \text{incseq-SucI}, \text{auto simp add: } g\text{-def } \text{ennreal-leI})$   
**hence** *convergent*  $(\lambda n. g n x)$  **for**  $x$  **unfolding** *convergent-def* **using** *LIM-SEQ-incseq-SUP* **by** *fast*  
**hence**  $(\lambda n. g n x) \longrightarrow g' x$  **for**  $x$  **unfolding** *g-def g'-def* **by**  $(\text{intro } \text{summable-iff-convergent}'[\text{THEN } \text{iffD2}, \text{THEN } \text{summable-LIMSEQ}], \text{blast})$   
**hence**  $(\int^+ x. g' x \partial M) = (\int^+ x. \liminf (\lambda n. g n x) \partial M)$  **by**  $(\text{metis } \text{lim-imp-Liminf } \text{trivial-limit-sequentially})$   
**also have**  $\dots \leq \liminf (\lambda n. \int^+ x. g n x \partial M)$  **by**  $(\text{intro } \text{nn-integral-liminf}, \text{simp add: } g\text{-def})$   
**also have**  $\dots \leq \liminf (\lambda n. 2)$  **using** *integrable-g* **by**  $(\text{intro } \text{Liminf-mono})$   $(\text{simp add: } \text{order-le-less})$   
**also have**  $\dots = 2$  **using** *sequentially-bot tendsto-iff-Liminf-eq-Limsup* **by** *blast*  
**finally show** *?thesis* .  
**qed**  
**hence** *AE*  $x$  *in*  $M. g' x < \infty$  **by**  $(\text{intro } \text{finite-nn-integral-imp-ae-finite})$   $(\text{auto simp add: } \text{order-le-less-trans } g'\text{-def})$   
**moreover have** *summable*  $(\lambda i. \text{norm } (s (r (Suc i)) x - s (r i) x))$  **if**  $g' x \neq \infty$  **for**  $x$  **using** *that* **unfolding** *g'-def* **by**  $(\text{intro } \text{summable-suminf-not-top})$  *fastforce* +  
  
**ultimately have** *ae-summable*: *AE*  $x$  *in*  $M. \text{summable } (\lambda i. s (r (Suc i)) x - s (r i) x)$  **using** *summable-norm-cancel* **unfolding** *dist-norm* **by** *force*  
  
{  
**fix**  $x$  **assume** *summable*  $(\lambda i. s (r (Suc i)) x - s (r i) x)$   
**hence**  $(\lambda n. \sum_{i < n}. s (r (Suc i)) x - s (r i) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x)$  **using** *summable-LIMSEQ* **by** *blast*  
**moreover have**  $(\lambda n. (\sum_{i < n}. s (r (Suc i)) x - s (r i) x)) = (\lambda n. s (r n) x - s (r 0) x)$  **using** *sum-lessThan-telescope* **by** *fastforce*  
**ultimately have**  $(\lambda n. s (r n) x - s (r 0) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x)$  **by** *argo*  
**hence**  $(\lambda n. s (r n) x - s (r 0) x + s (r 0) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x) + s (r 0) x$  **by**  $(\text{intro } \text{isCont-tendsto-compose}[\text{of } - \lambda z. z + s (r 0) x], \text{auto})$   
**hence** *Cauchy*  $(\lambda n. s (r n) x)$  **by**  $(\text{simp add: } \text{LIMSEQ-imp-Cauchy})$   
}  
**hence** *AE*  $x$  *in*  $M. \text{Cauchy } (\lambda i. s (r i) x)$  **using** *ae-summable* **by** *fast*  
**thus** *?thesis* **by**  $(\text{rule } \text{that}[\text{OF } \text{strict-mono}(1)])$   
**qed**

### 3.2 Totally Ordered Banach Spaces

**lemma** *integrable-max* $[\text{simp}, \text{intro}]$ :

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}, \text{linorder-topology}\}$   
**assumes** *fg* $[\text{measurable}]$ : *integrable*  $M f$  *integrable*  $M g$   
**shows** *integrable*  $M (\lambda x. \max (f x) (g x))$   
**proof**  $(\text{rule } \text{Bochner-Integration.integrable-bound})$



```

{
  fix x y :: 'b
  have norm (max x y) ≤ max (norm x) (norm y) by linarith
  also have ... ≤ norm x + norm y by simp
  finally have norm (max x y) ≤ norm (norm x + norm y) by simp
}
thus AE x in M. norm (max (f x) (g x)) ≤ norm (norm (f x) + norm (g x)) by
simp
qed (auto intro: Bochner-Integration.integrable-add[OF integrable-norm[OF fg(1)]
integrable-norm[OF fg(2)]])

```

```

lemma integrable-min[simp, intro]:
  fixes f :: 'a ⇒ 'b :: {second-countable-topology, banach, linorder-topology}
  assumes [measurable]: integrable M f integrable M g
  shows integrable M (λx. min (f x) (g x))
proof -
  have norm (min (f x) (g x)) ≤ norm (f x) ∨ norm (min (f x) (g x)) ≤ norm (g
x) for x by linarith
  thus ?thesis by (intro integrable-bound[OF integrable-max[OF integrable-norm(1,1),
OF assms]], auto)
qed

```

Restatement of *integral-nonneg-AE* for functions taking values in a Banach space.

```

lemma integral-nonneg-AE-banach:
  fixes f :: 'a ⇒ 'b :: {second-countable-topology, banach, linorder-topology, or-
dered-real-vector}
  assumes [measurable]: f ∈ borel-measurable M and nonneg: AE x in M. 0 ≤ f x
  shows 0 ≤ integralL M f
proof cases
  assume integrable: integrable M f
  hence max: (λx. max 0 (f x)) ∈ borel-measurable M ∧ x. 0 ≤ max 0 (f x)
integrable M (λx. max 0 (f x)) by auto
  hence 0 ≤ integralL M (λx. max 0 (f x))
  proof -
    obtain s where *: ∧i. simple-function M (s i)
      ∧i. emeasure M {y ∈ space M. s i y ≠ 0} ≠ ∞
      ∧x. x ∈ space M ⇒ (λi. s i x) ⟶ f x
      ∧x i. x ∈ space M ⇒ norm (s i x) ≤ 2 * norm (f x) using
integrable-implies-simple-function-sequence[OF integrable] by blast
    have simple: ∧i. simple-function M (λx. max 0 (s i x)) using * by fast
    have ∧i. {y ∈ space M. max 0 (s i y) ≠ 0} ⊆ {y ∈ space M. s i y ≠ 0}
  unfolding max-def by force
  moreover have ∧i. {y ∈ space M. s i y ≠ 0} ∈ sets M using * by measurable
  ultimately have ∧i. emeasure M {y ∈ space M. max 0 (s i y) ≠ 0} ≤
emeasure M {y ∈ space M. s i y ≠ 0} by (rule emeasure-mono)
  hence **: ∧i. emeasure M {y ∈ space M. max 0 (s i y) ≠ 0} ≠ ∞ using *(2)
by (auto intro: order.strict-trans1 simp add: top.not-eq-extremum)
  have ∧x. x ∈ space M ⇒ (λi. max 0 (s i x)) ⟶ max 0 (f x) using *(3)

```

*tendsto-max* **by** *blast*  
**moreover** **have**  $\bigwedge x i. x \in \text{space } M \implies \text{norm } (\max 0 (s i x)) \leq \text{norm } (2 *_R f x)$  **using** *\*(4) unfolding max-def by auto*  
**ultimately** **have** *tendsto*:  $(\lambda i. \text{integral}^L M (\lambda x. \max 0 (s i x))) \longrightarrow \text{integral}^L M (\lambda x. \max 0 (f x))$   
**using** *borel-measurable-simple-function simple integrable by (intro integral-dominated-convergence[OF max(1) - integrable-norm[OF integrable-scaleR-right], of - 2 f], blast+)*  
**{**  
**fix**  $h :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assume** *simple-function*  $M h \text{ emeasure } M \{y \in \text{space } M. h y \neq 0\} \neq \infty \bigwedge x. x \in \text{space } M \longrightarrow h x \geq 0$   
**hence**  $*$ :  $\text{integral}^L M h \geq 0$   
**proof** (*induct rule: integrable-simple-function-induct-nn*)  
**case** (*cong f g*)  
**then show** *?case* **using** *Bochner-Integration.integral-cong* **by** *force*  
**next**  
**case** (*indicator A y*)  
**hence**  $A \neq \{\}$   $\implies y \geq 0$  **using** *sets.sets-into-space* **by** *fastforce*  
**then show** *?case* **using** *indicator* **by** (*cases A = \{\}*, *auto simp add: scaleR-nonneg-nonneg*)  
**next**  
**case** (*add f g*)  
**then show** *?case* **by** (*simp add: integrable-simple-function*)  
**qed**  
**}**  
**thus** *?thesis* **using** *LIMSEQ-le-const[OF tendsto, of 0] \*\* simple* **by** *fastforce*  
**qed**  
**also** **have**  $\dots = \text{integral}^L M f$  **using** *nonneg* **by** (*auto intro: integral-cong-AE*)  
**finally** **show** *?thesis* .  
**qed** (*simp add: not-integrable-integral-eq*)

**lemma** *integral-mono-AE-banach*:

**fixes**  $f g :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$  *AE*  $x$  *in*  $M. f x \leq g x$   
**shows**  $\text{integral}^L M f \leq \text{integral}^L M g$   
**using** *integral-nonneg-AE-banach* [*of*  $\lambda x. g x - f x$ ] *assms* *Bochner-Integration.integral-diff* [*OF assms(1,2)*] **by** *force*

**lemma** *integral-mono-banach*:

**fixes**  $f g :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   $\bigwedge x. x \in \text{space } M \implies f x \leq g x$   
**shows**  $\text{integral}^L M f \leq \text{integral}^L M g$   
**using** *integral-mono-AE-banach* *assms* **by** *blast*

### 3.3 Integrability and Measurability of the Diameter

**context**

**fixes**  $s :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$  **and**  $M$

**assumes**  $\text{bounded}: \bigwedge x. x \in \text{space } M \implies \text{bounded } (\text{range } (\lambda i. s \ i \ x))$

**begin**

**lemma** *borel-measurable-diameter*:

**assumes**  $[\text{measurable}]: \bigwedge i. (s \ i) \in \text{borel-measurable } M$

**shows**  $(\lambda x. \text{diameter } \{s \ i \ x \mid i. n \leq i\}) \in \text{borel-measurable } M$

**proof** –

**have**  $\{s \ i \ x \mid i. N \leq i\} \neq \{\}$  **for**  $x \ N$  **by** *blast*

**hence** *diameter-SUP*:  $\text{diameter } \{s \ i \ x \mid i. N \leq i\} = (\text{SUP } (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i \ x) (s \ j \ x))$  **for**  $x \ N$  **unfolding** *diameter-def* **by**  $(\text{auto intro!}: \text{arg-cong}[\text{of} - \text{Sup}])$

**have** *case-prod dist* ‘ $(\{s \ i \ x \mid i. n \leq i\} \times \{s \ i \ x \mid i. n \leq i\}) = ((\lambda(i, j). \text{dist } (s \ i \ x) (s \ j \ x)) ' (\{n..\} \times \{n..\}))$ ’ **for**  $x$  **by** *fast*

**hence** \*:  $(\lambda x. \text{diameter } \{s \ i \ x \mid i. n \leq i\}) = (\lambda x. \text{Sup } ((\lambda(i, j). \text{dist } (s \ i \ x) (s \ j \ x)) ' (\{n..\} \times \{n..\})))$  **using** *diameter-SUP* **by**  $(\text{simp add: case-prod-beta})$

**have** *bounded*  $((\lambda(i, j). \text{dist } (s \ i \ x) (s \ j \ x)) ' (\{n..\} \times \{n..\}))$  **if**  $x \in \text{space } M$  **for**  $x$  **by**  $(\text{rule bounded-imp-dist-bounded}[\text{OF bounded, OF that}])$

**hence** *bdd*: *bdd-above*  $((\lambda(i, j). \text{dist } (s \ i \ x) (s \ j \ x)) ' (\{n..\} \times \{n..\}))$  **if**  $x \in \text{space } M$  **for**  $x$  **using** *that bounded-imp-bdd-above* **by** *presburger*

**have** *fst*  $p \in \text{borel-measurable } M$  **and**  $p \in \text{borel-measurable } M$  **if**  $p \in s ' \{n..\} \times s ' \{n..\}$  **for**  $p$  **using** *that* **by** *fastforce+*

**hence**  $(\lambda x. \text{fst } p \ x - \text{snd } p \ x) \in \text{borel-measurable } M$  **if**  $p \in s ' \{n..\} \times s ' \{n..\}$  **for**  $p$  **using** *that borel-measurable-diff* **by** *simp*

**hence**  $(\lambda x. \text{case } p \text{ of } (f, g) \Rightarrow \text{dist } (f \ x) (g \ x)) \in \text{borel-measurable } M$  **if**  $p \in s ' \{n..\} \times s ' \{n..\}$  **for**  $p$  **unfolding** *dist-norm* **using** *that* **by** *measurable*

**moreover** **have** *countable*  $(s ' \{n..\} \times s ' \{n..\})$  **by**  $(\text{intro countable-SIGMA countable-image, auto})$

**ultimately show** *?thesis* **unfolding** \* **by**  $(\text{auto intro!}: \text{borel-measurable-cSUP bdd})$

**qed**

**lemma** *integrable-bound-diameter*:

**fixes**  $f :: 'a \Rightarrow \text{real}$

**assumes** *integrable*  $M \ f$

**and**  $[\text{measurable}]: \bigwedge i. (s \ i) \in \text{borel-measurable } M$

**and**  $\bigwedge x \ i. x \in \text{space } M \implies \text{norm } (s \ i \ x) \leq f \ x$

**shows** *integrable*  $M \ (\lambda x. \text{diameter } \{s \ i \ x \mid i. n \leq i\})$

**proof** –

**have**  $\{s \ i \ x \mid i. N \leq i\} \neq \{\}$  **for**  $x \ N$  **by** *blast*

**hence** *diameter-SUP*:  $\text{diameter } \{s \ i \ x \mid i. N \leq i\} = (\text{SUP } (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i \ x) (s \ j \ x))$  **for**  $x \ N$  **unfolding** *diameter-def* **by**  $(\text{auto intro!}: \text{arg-cong}[\text{of} - \text{Sup}])$

{

**fix**  $x$  **assume**  $x: x \in \text{space } M$

```

    let ?S = (λ(i, j). dist (s i x) (s j x)) ‘ ({n..} × {n..})
    have case-prod dist ‘ ({s i x | i. n ≤ i} × {s i x | i. n ≤ i}) = (λ(i, j). dist (s i
x) (s j x)) ‘ ({n..} × {n..}) by fast
    hence *: diameter {s i x | i. n ≤ i} = Sup ?S using diameter-SUP by (simp
add: case-prod-beta')

    have bounded ?S by (rule bounded-imp-dist-bounded[OF bounded[OF x]])
    hence Sup-S-nonneg: 0 ≤ Sup ?S by (auto intro!: cSup-upper2 x bounded-imp-bdd-above)

    have dist (s i x) (s j x) ≤ 2 * f x for i j by (intro dist-triangle2[THEN
order-trans, of - 0]) (metis norm-conv-dist assms(3) x add-mono mult-2)
    hence ∀ c ∈ ?S. c ≤ 2 * f x by force
    hence Sup ?S ≤ 2 * f x by (intro cSup-least, auto)
    hence norm (Sup ?S) ≤ 2 * norm (f x) using Sup-S-nonneg by auto
    also have ... = norm (2 *R f x) by simp
    finally have norm (diameter {s i x | i. n ≤ i}) ≤ norm (2 *R f x) unfolding
* .
}
    hence AE x in M. norm (diameter {s i x | i. n ≤ i}) ≤ norm (2 *R f x) by blast
    thus integrable M (λx. diameter {s i x | i. n ≤ i}) using borel-measurable-diameter
by (intro Bochner-Integration.integrable-bound[OF assms(1)[THEN integrable-scaleR-right[of
2]]], measurable)
qed
end

```

### 3.4 Auxiliary Lemmas for Set Integrals

**lemma** *set-integral-scaleR-left*:

```

assumes A ∈ sets M c ≠ 0 ⇒ integrable M f
shows LINT t:A|M. f t *R c = (LINT t:A|M. f t) *R c
unfolding set-lebesgue-integral-def
using integrable-mult-indicator[OF assms]
by (subst integral-scaleR-left[symmetric], auto)

```

**lemma** *nn-set-integral-eq-set-integral*:

```

assumes [measurable]: integrable M f
and AE x ∈ A in M. 0 ≤ f x A ∈ sets M
shows (∫+ x ∈ A. f x ∂M) = (∫ x ∈ A. f x ∂M)

```

**proof** –

```

    have (∫+ x. indicator A x *R f x ∂M) = (∫ x ∈ A. f x ∂M)
    unfolding set-lebesgue-integral-def using assms(2) by (intro nn-integral-eq-integral[of
- λx. indicat-real A x *R f x], blast intro: assms integrable-mult-indicator, fastforce)
    moreover have (∫+ x. indicator A x *R f x ∂M) = (∫+ x ∈ A. f x ∂M) by (metis
ennreal-0 indicator-simps(1) indicator-simps(2) mult.commute mult-1 mult-zero-left
real-scaleR-def)

```

ultimately show ?thesis by argo

qed

**lemma** *set-integral-restrict-space*:

```

fixes f :: 'a  $\Rightarrow$  'b::{banach, second-countable-topology}
assumes  $\Omega \cap \text{space } M \in \text{sets } M$ 
shows set-lebesgue-integral (restrict-space M  $\Omega$ ) A f = set-lebesgue-integral M A
( $\lambda x. \text{indicator } \Omega x *_{\mathbb{R}} f x$ )
unfolding set-lebesgue-integral-def
by (subst integral-restrict-space, auto intro!: integrable-mult-indicator assms simp:
mult.commute)

lemma set-integral-const:
fixes c :: 'b::{banach, second-countable-topology}
assumes A  $\in \text{sets } M$  emeasure M A  $\neq \infty$ 
shows set-lebesgue-integral M A ( $\lambda \cdot. c$ ) = measure M A  $*_{\mathbb{R}} c$ 
unfolding set-lebesgue-integral-def
using assms by (metis has-bochner-integral-indicator has-bochner-integral-integral-eq
infinity-enreal-def less-top)

lemma set-integral-mono-banach:
fixes f g :: 'a  $\Rightarrow$  'b :: {second-countable-topology, banach, linorder-topology, or-
dered-real-vector}
assumes set-integrable M A f set-integrable M A g
 $\bigwedge x. x \in A \implies f x \leq g x$ 
shows (LINT x:A|M. f x)  $\leq$  (LINT x:A|M. g x)
using assms unfolding set-integrable-def set-lebesgue-integral-def
by (auto intro: integral-mono-banach split: split-indicator)

lemma set-integral-mono-AE-banach:
fixes f g :: 'a  $\Rightarrow$  'b :: {second-countable-topology, banach, linorder-topology, or-
dered-real-vector}
assumes set-integrable M A f set-integrable M A g AE x $\in$ A in M. f x  $\leq$  g x
shows set-lebesgue-integral M A f  $\leq$  set-lebesgue-integral M A g using assms
unfolding set-lebesgue-integral-def by (auto simp add: set-integrable-def intro!:
integral-mono-AE-banach[of M  $\lambda x. \text{indicator } A x *_{\mathbb{R}} f x$   $\lambda x. \text{indicator } A x *_{\mathbb{R}} g x$ ],
simp add: indicator-def)

```

### 3.5 Averaging Theorem

We aim to lift results from the real case to arbitrary Banach spaces. Our fundamental tool in this regard will be the averaging theorem. The proof of this theorem is due to Serge Lang (Real and Functional Analysis). The theorem allows us to make statements about a functions value almost everywhere, depending on the value its integral takes on various sets of the measure space.

Before we introduce and prove the averaging theorem, we will first show the following lemma which is crucial for our proof. While not stated exactly in this manner, our proof makes use of the characterization of second-countable topological spaces given in the book General Topology by Ryszard Engelking (Theorem 4.1.15).

**lemma** *balls-countable-basis*:

**obtains**  $D :: 'a :: \{\text{metric-space, second-countable-topology}\}$  *set*  
**where** *topological-basis* (*case-prod ball* ' ( $D \times (\mathbb{Q} \cap \{0 < ..\})$ )))  
**and** *countable*  $D$   
**and**  $D \neq \{\}$   
**proof** –  
**obtain**  $D :: 'a$  *set* **where** *dense-subset: countable*  $D$   $D \neq \{\}$   $\llbracket \text{open } U; U \neq \{\} \rrbracket$   
 $\implies \exists y \in D. y \in U$  **for**  $U$  **using** *countable-dense-exists* **by** *blast*  
**have** *topological-basis* (*case-prod ball* ' ( $D \times (\mathbb{Q} \cap \{0 < ..\})$ )))  
**proof** (*intro topological-basis-iff* [*THEN iffD2*], *fast*, *clarify*)  
**fix**  $U$  **and**  $x :: 'a$  **assume** *asm: open*  $U$   $x \in U$   
**obtain**  $e$  **where**  $e: e > 0$  *ball*  $x \in U$  **using** *asm openE* **by** *blast*  
**obtain**  $y$  **where**  $y: y \in D$   $y \in \text{ball } x (e / 3)$  **using** *dense-subset(3)* [*OF open-ball,*  
*of x e / 3*] *centre-in-ball* [*THEN iffD2, OF divide-pos-pos* [*OF e(1), of 3*]] **by** *force*  
**obtain**  $r$  **where**  $r: r \in \mathbb{Q} \cap \{e/3 < .. < e/2\}$  **unfolding** *Rats-def* **using** *of-rat-dense* [*OF*  
*divide-strict-left-mono* [*OF - e(1)*], *of 2 3*] **by** *auto*  
  
**have**  $*$ :  $x \in \text{ball } y r$  **using**  $r$   $y$  **by** (*simp add: dist-commute*)  
**hence**  $\text{ball } y r \subseteq U$  **using**  $r$  **by** (*intro order-trans* [*OF - e(2)*], *simp*, *metric*)  
**moreover** **have**  $\text{ball } y r \in (\text{case-prod ball ' } (D \times (\mathbb{Q} \cap \{0 < ..\})))$  **using**  $y(1)$   
*r by force*  
**ultimately show**  $\exists B' \in (\text{case-prod ball ' } (D \times (\mathbb{Q} \cap \{0 < ..\}))). x \in B' \wedge B' \subseteq$   
 $U$  **using**  $*$  **by** *meson*  
**qed**  
**thus** *?thesis* **using** *that dense-subset* **by** *blast*  
**qed**

**context** *sigma-finite-measure*  
**begin**

To show statements concerning  $\sigma$ -finite measure spaces, one usually shows the statement for finite measure spaces and uses a limiting argument to show it for the  $\sigma$ -finite case. The following induction scheme formalizes this.

**lemma** *sigma-finite-measure-induct* [*case-names finite-measure, consumes 0*]:

**assumes**  $\bigwedge (N :: 'a \text{ measure}) \Omega. \text{finite-measure } N$   
 $\implies N = \text{restrict-space } M \ \Omega$   
 $\implies \Omega \in \text{sets } M$   
 $\implies \text{emeasure } N \ \Omega \neq \infty$   
 $\implies \text{emeasure } N \ \Omega \neq 0$   
 $\implies \text{almost-everywhere } N \ Q$   
**and** [*measurable*]: *Measurable.pred*  $M \ Q$   
**shows** *almost-everywhere*  $M \ Q$   
**proof** –  
**have**  $*$ : *almost-everywhere*  $N \ Q$  **if** *finite-measure*  $N$   $N = \text{restrict-space } M \ \Omega$   $\Omega \in \text{sets } M$   $\text{emeasure } N \ \Omega \neq \infty$  **for**  $N \ \Omega$  **using** *that* **by** (*cases*  $\text{emeasure } N \ \Omega = 0$ , *auto intro: emeasure-0-AE assms(1)*)  
  
**obtain**  $A :: \text{nat} \Rightarrow 'a$  *set* **where**  $A$ : *range*  $A \subseteq \text{sets } M$   $(\bigcup i. A \ i) = \text{space } M$  **and**  
*emeasure-finite: emeasure*  $M \ (A \ i) \neq \infty$  **for**  $i$  **using** *sigma-finite* **by** *metis*

```

note  $A(1)[\text{measurable}]$ 
have  $\text{space-restr} : \text{space } (\text{restrict-space } M (A i)) = A i$  for  $i$  unfolding  $\text{space-restrict-space}$ 
by  $\text{simp}$ 
{
  fix  $i$ 
  have  $*$ :  $\{x \in A i \cap \text{space } M. Q x\} = \{x \in \text{space } M. Q x\} \cap (A i)$  by  $\text{fast}$ 
  have  $\text{Measurable.pred } (\text{restrict-space } M (A i)) Q$  using  $A$  by  $(\text{intro measurableI},$ 
 $\text{auto simp add: space-restr intro!: sets-restrict-space-iff[THEN iffD2], measurable},$ 
 $\text{auto})$ 
}
note  $\text{this}[\text{measurable}]$ 
{
  fix  $i$ 
  have  $\text{finite-measure } (\text{restrict-space } M (A i))$  using  $\text{emeasure-finite}$  by  $(\text{intro}$ 
 $\text{finite-measureI}, \text{subst space-restr}, \text{subst emeasure-restrict-space}, \text{auto})$ 
  hence  $\text{emeasure } (\text{restrict-space } M (A i)) \{x \in A i. \neg Q x\} = 0$  using  $\text{emea-}$ 
 $\text{sure-finite}$  by  $(\text{intro AE-iff-measurable[THEN iffD1, OF - - *]}, \text{measurable}, \text{subst}$ 
 $\text{space-restr[symmetric]}, \text{intro sets.top}, \text{auto simp add: emeasure-restrict-space})$ 
  hence  $\text{emeasure } M \{x \in A i. \neg Q x\} = 0$  by  $(\text{subst emeasure-restrict-space[symmetric]},$ 
 $\text{auto})$ 
}
hence  $\text{emeasure } M (\bigcup i. \{x \in A i. \neg Q x\}) = 0$  by  $(\text{intro emeasure-UN-eq-0},$ 
 $\text{auto})$ 
moreover have  $(\bigcup i. \{x \in A i. \neg Q x\}) = \{x \in \text{space } M. \neg Q x\}$  using  $A$  by
 $\text{auto}$ 
ultimately show  $?thesis$  by  $(\text{intro AE-iff-measurable[THEN iffD2]}, \text{auto})$ 
qed

```

The Averaging Theorem allows us to make statements concerning how a function behaves almost everywhere, depending on its behaviour on average.

**lemma** *averaging-theorem*:

```

fixes  $f :: \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$ 
assumes  $[\text{measurable}]: \text{integrable } M f$ 
and  $\text{closed}: \text{closed } S$ 
and  $\bigwedge A. A \in \text{sets } M \implies \text{measure } M A > 0 \implies (1 / \text{measure } M A) *_R$ 
 $\text{set-lebesgue-integral } M A f \in S$ 
shows  $\text{AE } x \text{ in } M. f x \in S$ 
proof  $(\text{induct rule: sigma-finite-measure-induct})$ 
case  $(\text{finite-measure } N \Omega)$ 

```

**interpret**  $\text{finite-measure } N$  **by**  $(\text{rule finite-measure})$

```

have  $\text{integrable}[\text{measurable}]: \text{integrable } N f$  using  $\text{assms finite-measure}$  by  $(\text{auto}$ 
 $\text{simp: integrable-restrict-space integrable-mult-indicator})$ 
have  $\text{average}: (1 / \text{Sigma-Algebra.measure } N A) *_R \text{set-lebesgue-integral } N A f$ 
 $\in S$  if  $A \in \text{sets } N$   $\text{measure } N A > 0$  for  $A$ 
proof  $-$ 
have  $*$ :  $A \in \text{sets } M$  using  $\text{that}$  by  $(\text{simp add: sets-restrict-space-iff finite-measure})$ 
have  $A = A \cap \Omega$  by  $(\text{metis finite-measure}(2,3) \text{inf.orderE sets.sets-into-space})$ 

```

space-restrict-space that(1))

hence  $\text{set-lebesgue-integral } N \ A \ f = \text{set-lebesgue-integral } M \ A \ f$  **unfolding**  
*finite-measure* **by** (*subst set-integral-restrict-space*, *auto simp add: finite-measure*  
*set-lebesgue-integral-def indicator-inter-arith[symmetric]*)

moreover **have**  $\text{measure } N \ A = \text{measure } M \ A$  **using** *that* **by** (*auto intro!:*  
*measure-restrict-space simp add: finite-measure sets-restrict-space-iff*)

ultimately **show** *?thesis* **using** *that \* assms(3)* **by** *presburger*

**qed**

**obtain**  $D :: 'b \text{ set}$  **where** *balls-basis: topological-basis (case-prod ball ' (D × (Q*  
 $\cap \{0 < ..\}))$ ) **and** *countable-D: countable D* **using** *balls-countable-basis* **by** *blast*

**have** *countable-balls: countable (case-prod ball ' (D × (Q*  
 $\cap \{0 < ..\}))$ ) **using**  
*countable-rat countable-D* **by** *blast*

**obtain**  $B$  **where** *B-balls: B ⊆ case-prod ball ' (D × (Q*  
 $\cap \{0 < ..\})) \cup B = -S$   
**using** *topological-basis[THEN iffD1, OF balls-basis] open-Compl[OF assms(2)]* **by**  
*meson*

hence *countable-B: countable B* **using** *countable-balls countable-subset* **by** *fast*

**define**  $b$  **where**  $b = \text{from-nat-into } (B \cup \{\{\}\})$

**have**  $B \cup \{\{\}\} \neq \{\}$  **by** *simp*

**have** *range-b: range b = B ∪ { { } }* **using** *countable-B* **by** (*auto simp add: b-def*  
*intro!:* *range-from-nat-into*)

**have** *open-b: open (b i)* **for**  $i$  **unfolding** *b-def* **using** *B-balls open-ball from-nat-into[of*  
 $B \cup \{\{\}\}$   $i]$  **by** *force*

**have** *Union-range-b:  $\bigcup (\text{range } b) = -S$*  **using** *B-balls range-b* **by** *simp*

{

**fix**  $v \ r$  **assume** *ball-in-Compl: ball v r ⊆ -S*

**define**  $A$  **where**  $A = f - ' \text{ball } v \ r \cap \text{space } N$

**have** *dist-less: dist (f x) v < r* **if**  $x \in A$  **for**  $x$  **using** *that* **unfolding** *A-def*  
*vimage-def* **by** (*simp add: dist-commute*)

**hence** *AE-less: AE x ∈ A in N. norm (f x - v) < r* **by** (*auto simp add:*  
*dist-norm*)

**have**  $*$ :  $A \in \text{sets } N$  **unfolding** *A-def* **by** *simp*

**have** *emeasure N A = 0*

**proof** -

{

**assume** *asm: emeasure N A > 0*

**hence** *measure-pos: measure N A > 0* **unfolding** *emeasure-eq-measure* **by**  
*simp*

**hence**  $(1 / \text{measure } N \ A) *_{\mathbb{R}} \text{set-lebesgue-integral } N \ A \ f - v = (1 / \text{measure } N$   
 $A) *_{\mathbb{R}} \text{set-lebesgue-integral } N \ A \ (\lambda x. f \ x - v)$  **using** *integrable integrable-const \** **by**  
*(subst set-integral-diff(2), auto simp add: set-integrable-def set-integral-const[OF \*]*  
*algebra-simps intro!:* *integrable-mult-indicator*)

**moreover** **have**  $\text{norm } (\int x \in A. (f \ x - v) \partial N) \leq (\int x \in A. \text{norm } (f \ x$   
 $- v) \partial N)$  **using**  $*$  **by** (*auto intro!:* *integral-norm-bound[of N λx. indicator A x*  
 $*_{\mathbb{R}} (f \ x - v), \text{ THEN order-trans}]$  *integrable-mult-indicator integrable simp add:*  
*set-lebesgue-integral-def*)



**ultimately have**  $\text{norm } ((1 / \text{measure } N \ A) *_{\mathcal{R}} \text{set-lebesgue-integral } N \ A \ f - v) \leq \text{set-lebesgue-integral } N \ A \ (\lambda x. \text{norm } (f \ x - v)) / \text{measure } N \ A$  **using** *asm*  
**by** (*auto intro: divide-right-mono*)  
**also have**  $\dots < \text{set-lebesgue-integral } N \ A \ (\lambda x. r) / \text{measure } N \ A$   
**unfolding** *set-lebesgue-integral-def*  
**using** *asm \* integrable integrable-const AE-less measure-pos*  
**by** (*intro divide-strict-right-mono integral-less-AE[of - - A] integrable-mult-indicator*)  
*(fastforce simp add: dist-less dist-norm indicator-def)+*  
**also have**  $\dots = r$  **using** *measure-pos* **by** (*simp add: set-integral-const*)  
**finally have**  $\text{dist } ((1 / \text{measure } N \ A) *_{\mathcal{R}} \text{set-lebesgue-integral } N \ A \ f) \ v < r$   
**by** (*subst dist-norm*)  
**hence** *False* **using** *average[OF \* measure-pos]* **by** (*metis ComplD dist-commute in-mono mem-ball ball-in-Compl*)  
**}**  
**thus** *?thesis* **by** *fastforce*  
**qed**  
**}**  
**note**  $*$  **=** *this*  
**{**  
**fix**  $b'$  **assume**  $b' \in B$   
**hence** *ball-subset-Compl*:  $b' \subseteq -S$  **and** *ball-radius-pos*:  $\exists v \in D. \exists r > 0. b' = \text{ball } v \ r$  **using** *B-balls* **by** (*blast, fast*)  
**}**  
**note**  $**$  **=** *this*  
**hence**  $\text{emeasure } N \ (f - ' b \ i \cap \text{space } N) = 0$  **for**  $i$  **by** (*cases b i = {}, simp*)  
*(metis UnE singletonD \* range-b[THEN eq-refl, THEN range-subsetD])*  
**hence**  $\text{emeasure } N \ (\bigcup i. f - ' b \ i \cap \text{space } N) = 0$  **using** *open-b* **by** (*intro emeasure-UN-eq-0 fastforce+*)  
**moreover have**  $(\bigcup i. f - ' b \ i \cap \text{space } N) = f - ' (\bigcup (\text{range } b)) \cap \text{space } N$  **by** *blast*  
**ultimately have**  $\text{emeasure } N \ (f - ' (-S) \cap \text{space } N) = 0$  **using** *Union-range-b*  
**by** *argo*  
**hence**  $\text{AE } x \text{ in } N. f \ x \notin -S$  **using** *open-Compl[OF assms(2)]* **by** (*intro AE-iff-measurable[THEN iffD2], auto*)  
**thus** *?case* **by** *force*  
**qed** (*simp add: pred-sets2[OF borel-closed] assms(2)*)

**lemma** *density-zero*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$   
**assumes** *integrable M f*  
**and** *density-0*:  $\bigwedge A. A \in \text{sets } M \implies \text{set-lebesgue-integral } M \ A \ f = 0$   
**shows**  $\text{AE } x \text{ in } M. f \ x = 0$   
**using** *averaging-theorem[OF assms(1), of {0}] assms(2)*  
**by** (*simp add: scaleR-nonneg-nonneg*)

The following lemma shows that densities are unique in Banach spaces.

**lemma** *density-unique-banach*:

**fixes**  $f \ f' :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$   
**assumes** *integrable M f integrable M f'*

**and** *density-eq*:  $\bigwedge A. A \in \text{sets } M \implies \text{set-lebesgue-integral } M A f = \text{set-lebesgue-integral } M A f'$   
**shows**  $AE\ x\ \text{in } M. f\ x = f'\ x$   
**proof** –  
{  
  **fix**  $A$  **assume**  $asm: A \in \text{sets } M$   
  **hence**  $LINT\ x|M. \text{indicat-real } A\ x *_R (f\ x - f'\ x) = 0$  **using** *density-eq*  
*assms(1,2)* **by** (*simp add: set-lebesgue-integral-def algebra-simps Bochner-Integration.integral-diff[OF integrable-mult-indicator(1,1)]*)  
}  
**thus** *?thesis* **using** *density-zero[OF Bochner-Integration.integrable-diff[OF assms(1,2)]]*  
**by** (*simp add: set-lebesgue-integral-def*)  
**qed**

**lemma** *density-nonneg*:  
**fixes**  $f :: - \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$   
  **and**  $\bigwedge A. A \in \text{sets } M \implies \text{set-lebesgue-integral } M A f \geq 0$   
**shows**  $AE\ x\ \text{in } M. f\ x \geq 0$   
**using** *averaging-theorem[OF assms(1), of \{0..\}, OF closed-atLeast] assms(2)*  
**by** (*simp add: scaleR-nonneg-nonneg*)

**corollary** *integral-nonneg-eq-0-iff-AE-banach*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes**  $f[\text{measurable}]: \text{integrable } M f$  **and** *nonneg*:  $AE\ x\ \text{in } M. 0 \leq f\ x$   
**shows**  $\text{integral}^L M f = 0 \longleftrightarrow (AE\ x\ \text{in } M. f\ x = 0)$

**proof**

**assume**  $*$ :  $\text{integral}^L M f = 0$   
{  
  **fix**  $A$  **assume**  $asm: A \in \text{sets } M$   
  **have**  $0 \leq \text{integral}^L M (\lambda x. \text{indicator } A\ x *_R f\ x)$  **using** *nonneg* **by** (*subst integral-zero[of M, symmetric], intro integral-mono-AE-banach integrable-mult-indicator asm f integrable-zero, auto simp add: indicator-def*)  
  **moreover** **have**  $\dots \leq \text{integral}^L M f$  **using** *nonneg* **by** (*intro integral-mono-AE-banach integrable-mult-indicator asm f, auto simp add: indicator-def*)  
  **ultimately** **have**  $\text{set-lebesgue-integral } M A f = 0$  **unfolding** *set-lebesgue-integral-def*  
**using**  $*$  **by** *force*  
}  
**thus**  $AE\ x\ \text{in } M. f\ x = 0$  **by** (*intro density-zero f, blast*)  
**qed** (*auto simp add: integral-eq-zero-AE*)

**corollary** *integral-eq-mono-AE-eq-AE*:

**fixes**  $f\ g :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   $\text{integral}^L M f = \text{integral}^L M g$   $AE\ x\ \text{in } M. f\ x \leq g\ x$   
**shows**  $AE\ x\ \text{in } M. f\ x = g\ x$   
**proof** –

```

define  $h$  where  $h = (\lambda x. g\ x - f\ x)$ 
  have  $AE\ x\ in\ M. h\ x = 0$  unfolding  $h\text{-def}$  using  $assms$  by (subst integ-
ral-nonneg-eq-0-iff-AE-banach[symmetric]) auto
  then show ?thesis unfolding  $h\text{-def}$  by auto
qed

end

end

```

```

theory Conditional-Expectation-Banach
  imports HOL-Probability.Conditional-Expectation HOL-Probability.Independent-Family
  Bochner-Integration-Supplement
begin

```

## 4 Conditional Expectation in Banach Spaces

Before we can talk about 'the' conditional expectation, we must define what it means for a function to have a conditional expectation.

**definition**  $has\text{-}cond\text{-}exp :: 'a\ measure \Rightarrow 'a\ measure \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b :: \{real\text{-}normed\text{-}vector, second\text{-}countable\text{-}topology\}) \Rightarrow bool$  **where**

$$has\text{-}cond\text{-}exp\ M\ F\ f\ g = ((\forall A \in sets\ F. (\int x \in A. f\ x\ \partial M) = (\int x \in A. g\ x\ \partial M))$$

$$\wedge integrable\ M\ f$$

$$\wedge integrable\ M\ g$$

$$\wedge g \in borel\text{-}measurable\ F)$$

This predicate precisely characterizes what it means for a function  $f$  to have a conditional expectation  $g$ , with respect to the measure  $M$  and the sub- $\sigma$ -algebra  $F$ .

**lemma**  $has\text{-}cond\text{-}expI'$ :

**assumes**  $\bigwedge A. A \in sets\ F \implies (\int x \in A. f\ x\ \partial M) = (\int x \in A. g\ x\ \partial M)$

$integrable\ M\ f$

$integrable\ M\ g$

$g \in borel\text{-}measurable\ F$

**shows**  $has\text{-}cond\text{-}exp\ M\ F\ f\ g$

**using**  $assms$  **unfolding**  $has\text{-}cond\text{-}exp\text{-}def$  **by** *simp*

**lemma**  $has\text{-}cond\text{-}expD$ :

**assumes**  $has\text{-}cond\text{-}exp\ M\ F\ f\ g$

**shows**  $\bigwedge A. A \in sets\ F \implies (\int x \in A. f\ x\ \partial M) = (\int x \in A. g\ x\ \partial M)$

$integrable\ M\ f$

$integrable\ M\ g$

$g \in borel\text{-}measurable\ F$

**using**  $assms$  **unfolding**  $has\text{-}cond\text{-}exp\text{-}def$  **by** *simp+*

Now we can use Hilberts  $\epsilon$ -operator to define the conditional expectation,

if it exists.

**definition** *cond-exp* :: 'a measure  $\Rightarrow$  'a measure  $\Rightarrow$  ('a  $\Rightarrow$  'b)  $\Rightarrow$  ('a  $\Rightarrow$  'b::{banach, second-countable-topology}) **where**  
*cond-exp* *M F f* = (if  $\exists g. \text{has-cond-exp } M F f g$  then (SOME *g. has-cond-exp M F f g*) else ( $\lambda\cdot. 0$ ))

**lemma** *borel-measurable-cond-exp*[*measurable*]: *cond-exp M F f*  $\in$  *borel-measurable F*

**by** (metis *cond-exp-def someI has-cond-exp-def borel-measurable-const*)

**lemma** *integrable-cond-exp*[*intro*]: *integrable M (cond-exp M F f)*

**by** (metis *cond-exp-def has-cond-expD(3) integrable-zero someI*)

**lemma** *set-integrable-cond-exp*[*intro*]:

**assumes** *A*  $\in$  *sets M*

**shows** *set-integrable M A (cond-exp M F f)* **using** *integrable-mult-indicator*[*OF assms integrable-cond-exp, of F f*] **by** (auto simp add: *set-integrable-def intro!*: *integrable-mult-indicator*[*OF assms integrable-cond-exp*])

**lemma** *has-cond-exp-self*:

**assumes** *integrable M f*

**shows** *has-cond-exp M (vimage-algebra (space M) f borel) f f*

**using** *assms* **by** (auto *intro!*: *has-cond-expI' measurable-vimage-algebra1*)

**lemma** *has-cond-exp-sets-cong*:

**assumes** *sets F = sets G*

**shows** *has-cond-exp M F = has-cond-exp M G*

**using** *assms* **unfolding** *has-cond-exp-def* **by** *force*

**lemma** *cond-exp-sets-cong*:

**assumes** *sets F = sets G*

**shows** *AE x in M. cond-exp M F f x = cond-exp M G f x*

**by** (*intro AE-I2, simp add: cond-exp-def has-cond-exp-sets-cong*[*OF assms, of M*])

**context** *sigma-finite-subalgebra*

**begin**

**lemma** *borel-measurable-cond-exp'*[*measurable*]: *cond-exp M F f*  $\in$  *borel-measurable M*

**by** (metis *cond-exp-def someI has-cond-exp-def borel-measurable-const subalg measurable-from-subalg*)

**lemma** *cond-exp-null*:

**assumes**  $\nexists g. \text{has-cond-exp } M F f g$

**shows** *cond-exp M F f* = ( $\lambda\cdot. 0$ )

**unfolding** *cond-exp-def* **using** *assms* **by** *arg0*

We state the tower property of the conditional expectation in terms of the

predicate *has-cond-exp*.

**lemma** *has-cond-exp-nested-subalg*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *subalgebra*  $G F$  *has-cond-exp*  $M F f h$  *has-cond-exp*  $M G f h'$   
**shows** *has-cond-exp*  $M F h' h$   
**by** (*intro has-cond-expI'*) (*metis assms has-cond-expD in-mono subalgebra-def*)+

The following lemma shows that the conditional expectation is unique as an element of L1, given that it exists.

**lemma** *has-cond-exp-charact*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *has-cond-exp*  $M F f g$   
**shows** *has-cond-exp*  $M F f$  (*cond-exp*  $M F f$ )  
 $AE\ x\ in\ M. \text{cond-exp}\ M F f\ x = g\ x$

**proof** –

**show** *cond-exp*: *has-cond-exp*  $M F f$  (*cond-exp*  $M F f$ ) **using** *assms someI cond-exp-def* **by** *metis*

**let**  $?MF = \text{restr-to-subalg}\ M F$

**interpret** *sigma-finite-measure*  $?MF$  **by** (*rule sigma-fin-subalg*)

{

**fix**  $A$  **assume**  $A \in \text{sets}\ ?MF$

**then have** [*measurable*]:  $A \in \text{sets}\ F$  **using** *sets-restr-to-subalg[OF subalg]* **by**

*simp*

**have**  $(\int x \in A. g\ x\ \partial ?MF) = (\int x \in A. g\ x\ \partial M)$  **using** *assms subalg* **by** (*auto simp add: integral-subalgebra2 set-lebesgue-integral-def dest!: has-cond-expD*)

**also have**  $\dots = (\int x \in A. \text{cond-exp}\ M F f\ x\ \partial M)$  **using** *assms cond-exp* **by** (*simp add: has-cond-exp-def*)

**also have**  $\dots = (\int x \in A. \text{cond-exp}\ M F f\ x\ \partial ?MF)$  **using** *subalg* **by** (*auto simp add: integral-subalgebra2 set-lebesgue-integral-def*)

**finally have**  $(\int x \in A. g\ x\ \partial ?MF) = (\int x \in A. \text{cond-exp}\ M F f\ x\ \partial ?MF)$  **by** *simp*

}

**hence**  $AE\ x\ in\ ?MF. \text{cond-exp}\ M F f\ x = g\ x$  **using** *cond-exp assms subalg* **by** (*intro density-unique-banach, auto dest: has-cond-expD intro!: integrable-in-subalg*)

**then show**  $AE\ x\ in\ M. \text{cond-exp}\ M F f\ x = g\ x$  **using** *AE-restr-to-subalg[OF subalg]* **by** *simp*

**qed**

**corollary** *cond-exp-charact*:

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

**assumes**  $\bigwedge A. A \in \text{sets}\ F \implies (\int x \in A. f\ x\ \partial M) = (\int x \in A. g\ x\ \partial M)$

*integrable*  $M f$

*integrable*  $M g$

$g \in \text{borel-measurable}\ F$

**shows**  $AE\ x\ in\ M. \text{cond-exp}\ M F f\ x = g\ x$

**by** (*intro has-cond-exp-charact has-cond-expI' assms*) *auto*

Identity on F-measurable functions:

If an integrable function  $f$  is already  $F$ -measurable, then  $\text{cond-exp } M F f = f$   $\mathbb{T}$ -a.e. This is a corollary of the lemma on the characterization of  $\text{cond-exp}$ .

**corollary**  $\text{cond-exp-F-meas}[\text{intro}, \text{simp}]$ :  
**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes**  $\text{integrable } M f$   
 $f \in \text{borel-measurable } F$   
**shows**  $\text{AE } x \text{ in } M. \text{ cond-exp } M F f x = f x$   
**by** (rule  $\text{cond-exp-charact}$ , auto intro:  $\text{assms}$ )

Congruence

**lemma**  $\text{has-cond-exp-cong}$ :  
**assumes**  $\text{integrable } M f \wedge x. x \in \text{space } M \implies f x = g x$   $\text{has-cond-exp } M F g h$   
**shows**  $\text{has-cond-exp } M F f h$   
**proof** (intro  $\text{has-cond-expI}'[OF - \text{assms}(1)]$ )  
**fix**  $A$  **assume**  $\text{asm}: A \in \text{sets } F$   
**hence**  $\text{set-lebesgue-integral } M A f = \text{set-lebesgue-integral } M A g$  **by** (intro  $\text{set-lebesgue-integral-cong}$ )  
(meson  $\text{assms}(2)$  subalg in-mono subalgebra-def sets.sets-into-space subalgebra-def subsetD)+  
**thus**  $\text{set-lebesgue-integral } M A f = \text{set-lebesgue-integral } M A h$  **using**  $\text{asm assms}(3)$   
**by** (simp add:  $\text{has-cond-exp-def}$ )  
**qed** (auto simp add:  $\text{has-cond-expD}[OF \text{assms}(3)]$ )

**lemma**  $\text{cond-exp-cong}$ :  
**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes**  $\text{integrable } M f \text{ integrable } M g \wedge x. x \in \text{space } M \implies f x = g x$   
**shows**  $\text{AE } x \text{ in } M. \text{ cond-exp } M F f x = \text{cond-exp } M F g x$   
**proof** (cases  $\exists h. \text{has-cond-exp } M F f h$ )  
**case**  $\text{True}$   
**then obtain**  $h$  **where**  $h: \text{has-cond-exp } M F f h \text{ has-cond-exp } M F g h$  **using**  
 $\text{has-cond-exp-cong assms}$  **by**  $\text{metis}$   
**show**  $?thesis$  **using**  $h[\text{THEN } \text{has-cond-exp-charact}(2)]$  **by**  $\text{fastforce}$   
**next**  
**case**  $\text{False}$   
**moreover have**  $\nexists h. \text{has-cond-exp } M F g h$  **using**  $\text{False has-cond-exp-cong assms}$   
**by**  $\text{auto}$   
**ultimately show**  $?thesis$  **unfolding**  $\text{cond-exp-def}$  **by**  $\text{auto}$   
**qed**

**lemma**  $\text{has-cond-exp-cong-AE}$ :  
**assumes**  $\text{integrable } M f \text{ AE } x \text{ in } M. f x = g x$   $\text{has-cond-exp } M F g h$   
**shows**  $\text{has-cond-exp } M F f h$   
**using**  $\text{assms}(1,2)$  subalg subalgebra-def subset-iff  
**by** (intro  $\text{has-cond-expI}'$ , subst  $\text{set-lebesgue-integral-cong-AE}[OF - \text{assms}(1)[\text{THEN } \text{borel-measurable-integrable}]]$   $\text{borel-measurable-integrable}(1)[OF \text{has-cond-expD}(2)[OF \text{assms}(3)]]$ )  
(fast intro:  $\text{has-cond-expD}[OF \text{assms}(3)]$   $\text{integrable-cong-AE-imp}[OF - - \text{AE-symmetric}])$ +

**lemma**  $\text{has-cond-exp-cong-AE}'$ :  
**assumes**  $h \in \text{borel-measurable } F \text{ AE } x \text{ in } M. h x = h' x$   $\text{has-cond-exp } M F f h'$

**shows** *has-cond-exp*  $M F f h$   
**using** *assms*(1, 2) *subalg* *subalgebra-def* *subset-iff*  
**using** *AE-restr-to-subalg2*[*OF* *subalg* *assms*(2)] *measurable-from-subalg*  
**by** (*intro* *has-cond-expI'*, *subst* *set-lebesgue-integral-cong-AE*[*OF* - *measurable-from-subalg*(1,1)](*OF* *subalg*), *OF* - *assms*(1) *has-cond-expD*(4)[*OF* *assms*(3)])  
*(fast intro: has-cond-expD*[*OF* *assms*(3)] *integrable-cong-AE-imp*[*OF* - - *AE-symmetric*])+

**lemma** *cond-exp-cong-AE*:  
**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$  *AE*  $x$  *in*  $M$ .  $f x = g x$   
**shows** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F f x = \text{cond-exp } M F g x$   
**proof** (*cases*  $\exists h$ . *has-cond-exp*  $M F f h$ )  
**case** *True*  
**then obtain**  $h$  **where**  $h$ : *has-cond-exp*  $M F f h$  *has-cond-exp*  $M F g h$  **using**  
*has-cond-exp-cong-AE* *assms* **by** (*metis* (*mono-tags*, *lifting*) *eventually-mono*)  
**show** *?thesis* **using**  $h$ [*THEN* *has-cond-exp-charact*(2)] **by** *fastforce*  
**next**  
**case** *False*  
**moreover have**  $\nexists h$ . *has-cond-exp*  $M F g h$  **using** *False* *has-cond-exp-cong-AE*  
*assms* **by** *auto*  
**ultimately show** *?thesis* **unfolding** *cond-exp-def* **by** *auto*  
**qed**

The conditional expectation operator on the reals, *real-cond-exp*, satisfies the conditions of the conditional expectation as we have defined it.

**lemma** *has-cond-exp-real*:  
**fixes**  $f :: 'a \Rightarrow \text{real}$   
**assumes** *integrable*  $M f$   
**shows** *has-cond-exp*  $M F f$  (*real-cond-exp*  $M F f$ )  
**by** (*intro* *has-cond-expI'*, *auto* *intro!*: *real-cond-exp-intA* *assms*)

**lemma** *cond-exp-real[intro]*:  
**fixes**  $f :: 'a \Rightarrow \text{real}$   
**assumes** *integrable*  $M f$   
**shows** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F f x = \text{real-cond-exp } M F f x$   
**using** *has-cond-exp-charact* *has-cond-exp-real* *assms* **by** *blast*

**lemma** *cond-exp-cmult*:  
**fixes**  $f :: 'a \Rightarrow \text{real}$   
**assumes** *integrable*  $M f$   
**shows** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F (\lambda x. c * f x) x = c * \text{cond-exp } M F f x$   
**using** *real-cond-exp-cmult*[*OF* *assms*(1), *of*  $c$ ] *assms*(1)[*THEN* *cond-exp-real*]  
*assms*(1)[*THEN* *integrable-mult-right*, *THEN* *cond-exp-real*, *of*  $c$ ] **by** *fastforce*

## 4.1 Existence

Showing the existence is a bit involved. Specifically, what we aim to show is that *has-cond-exp*  $M F f$  (*cond-exp*  $M F f$ ) holds for any Bochner-integrable  $f$ . We will employ the standard machinery of measure theory. First, we will

prove existence for indicator functions. Then we will extend our proof by linearity to simple functions. Finally we use a limiting argument to show that the conditional expectation exists for all Bochner-integrable functions.

Indicator functions

**lemma** *has-cond-exp-indicator*:

**assumes**  $A \in \text{sets } M \text{ emeasure } M \ A < \infty$

**shows**  $\text{has-cond-exp } M \ F \ (\lambda x. \text{indicat-real } A \ x \ *_R \ y) \ (\lambda x. \text{real-cond-exp } M \ F \ (\text{indicator } A) \ x \ *_R \ y)$

**proof** (*intro has-cond-expI'*, *goal-cases*)

**case** (*1 B*)

**have**  $\int_{x \in B. (\text{indicat-real } A \ x \ *_R \ y) \ \partial M} = (\int_{x \in B. \text{indicat-real } A \ x \ \partial M} \ *_R \ y)$  **using** *assms* **by** (*intro set-integral-scaleR-left*, *meson 1 in-mono subalg subalgebra-def*, *blast*)

**also have**  $\dots = (\int_{x \in B. \text{real-cond-exp } M \ F \ (\text{indicator } A) \ x \ \partial M} \ *_R \ y)$  **using** *1 assms* **by** (*subst real-cond-exp-intA*, *auto*)

**also have**  $\dots = \int_{x \in B. (\text{real-cond-exp } M \ F \ (\text{indicator } A) \ x \ *_R \ y) \ \partial M}$  **using** *assms* **by** (*intro set-integral-scaleR-left[symmetric]*, *meson 1 in-mono subalg subalgebra-def*, *blast*)

**finally show** *?case* .

**next**

**case** *2*

**show** *?case* **using** *integrable-scaleR-left integrable-real-indicator assms* **by** *blast*

**next**

**case** *3*

**show** *?case* **using** *assms* **by** (*intro integrable-scaleR-left*, *intro real-cond-exp-int*, *blast+*)

**next**

**case** *4*

**show** *?case* **by** (*intro borel-measurable-scaleR*, *intro Conditional-Expectation.borel-measurable-cond-exp*, *simp*)

**qed**

**lemma** *cond-exp-indicator[intro]*:

**fixes**  $y :: 'b :: \{\text{second-countable-topology}, \text{banach}\}$

**assumes** [*measurable*]:  $A \in \text{sets } M \text{ emeasure } M \ A < \infty$

**shows**  $\text{AE } x \text{ in } M. \text{cond-exp } M \ F \ (\lambda x. \text{indicat-real } A \ x \ *_R \ y) \ x = \text{cond-exp } M \ F \ (\text{indicator } A) \ x \ *_R \ y$

**proof** –

**have**  $\text{AE } x \text{ in } M. \text{cond-exp } M \ F \ (\lambda x. \text{indicat-real } A \ x \ *_R \ y) \ x = \text{real-cond-exp } M \ F \ (\text{indicator } A) \ x \ *_R \ y$  **using** *has-cond-exp-indicator[OF assms]* *has-cond-exp-charact* **by** *blast*

**thus** *?thesis* **using** *cond-exp-real[OF integrable-real-indicator, OF assms]* **by** *fast-force*

**qed**

Addition

**lemma** *has-cond-exp-add*:

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



```

assumes has-cond-exp  $M$   $F$   $f$   $f'$  has-cond-exp  $M$   $F$   $g$   $g'$ 
shows has-cond-exp  $M$   $F$   $(\lambda x. f\ x + g\ x)$   $(\lambda x. f'\ x + g'\ x)$ 
proof (intro has-cond-expI', goal-cases)
  case (1  $A$ )
    have  $\int_{x \in A}. (f\ x + g\ x) \partial M = (\int_{x \in A}. f\ x\ \partial M) + (\int_{x \in A}. g\ x\ \partial M)$  using
assms[THEN has-cond-expD(2)] subalg 1 by (intro set-integral-add(2), auto simp
add: subalgebra-def set-integrable-def intro: integrable-mult-indicator)
    also have  $\dots = (\int_{x \in A}. f'\ x\ \partial M) + (\int_{x \in A}. g'\ x\ \partial M)$  using assms[THEN
has-cond-expD(1)[OF - 1]] by argo
    also have  $\dots = \int_{x \in A}. (f'\ x + g'\ x) \partial M$  using assms[THEN has-cond-expD(3)]
subalg 1 by (intro set-integral-add(2)[symmetric], auto simp add: subalgebra-def
set-integrable-def intro: integrable-mult-indicator)
    finally show ?case .
  next
    case 2
    show ?case by (metis Bochner-Integration.integrable-add assms has-cond-expD(2))
  next
    case 3
    show ?case by (metis Bochner-Integration.integrable-add assms has-cond-expD(3))
  next
    case 4
    show ?case using assms borel-measurable-add has-cond-expD(4) by blast
qed

```

**lemma** *has-cond-exp-scaleR-right*:

```

fixes  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$ 
assumes has-cond-exp  $M$   $F$   $f$   $f'$ 
shows has-cond-exp  $M$   $F$   $(\lambda x. c *_{\mathbb{R}} f\ x)$   $(\lambda x. c *_{\mathbb{R}} f'\ x)$ 
using has-cond-expD[OF assms] by (intro has-cond-expI', auto)

```

**lemma** *cond-exp-scaleR-right*:

```

fixes  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$ 
assumes integrable  $M$   $f$ 
shows  $\forall x \in M. \text{cond-exp } M\ F\ (\lambda x. c *_{\mathbb{R}} f\ x)\ x = c *_{\mathbb{R}} \text{cond-exp } M\ F\ f\ x$ 
proof (cases  $\exists f'. \text{has-cond-exp } M\ F\ f\ f'$ )
  case True
    then show ?thesis using assms has-cond-exp-charact has-cond-exp-scaleR-right
by metis
  next
    case False
    show ?thesis
    proof (cases  $c = 0$ )
      case True
        then show ?thesis by simp
    next
      case c-nonzero: False
      have  $\nexists f'. \text{has-cond-exp } M\ F\ (\lambda x. c *_{\mathbb{R}} f\ x)\ f'$ 
      proof (standard, goal-cases)
        case 1

```

```

    then obtain f' where f': has-cond-exp M F (λx. c *R f x) f' by blast
    have has-cond-exp M F f (λx. inverse c *R f' x) using has-cond-expD[OF
f'] divideR-right[OF c-nonzero] assms by (intro has-cond-expI', auto)
    then show ?case using False by blast
qed
then show ?thesis using cond-exp-null[OF False] cond-exp-null by force
qed
qed

```

**lemma** *cond-exp-uminus*:

```

  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes integrable M f
  shows AE x in M. cond-exp M F (λx. - f x) x = - cond-exp M F f x
  using cond-exp-scaleR-right[OF assms, of -1] by force

```

Together with the induction scheme *integrable-simple-function-induct*, we can show that the conditional expectation of an integrable simple function exists.

**corollary** *has-cond-exp-simple*:

```

  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes simple-function M f emeasure M {y ∈ space M. f y ≠ 0} ≠ ∞
  shows has-cond-exp M F f (cond-exp M F f)
  using assms

```

**proof** (*induction rule: integrable-simple-function-induct*)

```

  case (cong f g)

```

```

    then show ?case using has-cond-exp-cong by (metis (no-types, opaque-lifting)
Bochner-Integration.integrable-cong has-cond-expD(2) has-cond-exp-charact(1))

```

```

next

```

```

  case (indicator A y)

```

```

    then show ?case using has-cond-exp-charact[OF has-cond-exp-indicator] by fast

```

```

next

```

```

  case (add u v)

```

```

    then show ?case using has-cond-exp-add has-cond-exp-charact(1) by blast

```

```

qed

```

Now comes the most difficult part. Given a convergent sequence of integrable simple functions  $s$ , we must show that the sequence  $\lambda n. \text{cond-exp } M F (s \ n)$  is also convergent. Furthermore, we must show that this limit satisfies the properties of a conditional expectation. Unfortunately, we will only be able to show that this sequence converges in the L1-norm. Luckily, this is enough to show that the operator  $\text{cond-exp } M F$  preserves limits as a function from L1 to L1.

In anticipation of this result, we show that the conditional expectation operator is a contraction for simple functions. We first reformulate the lemma *real-cond-exp-abs*, which shows the statement for real-valued functions, using our definitions. Then we show the statement for simple functions via induction.

**lemma** *cond-exp-contraction-real*:

**fixes**  $f :: 'a \Rightarrow \text{real}$   
**assumes** *integrable*[*measurable*]: *integrable*  $M f$   
**shows**  $AE\ x\ in\ M. \text{norm}(\text{cond-exp } M F f\ x) \leq \text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$   
**proof** –  
**have** *int*: *integrable*  $M (\lambda x. \text{norm}(f\ x))$  **using** *assms* **by** *blast*  
**have** \*:  $AE\ x\ in\ M. 0 \leq \text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$  **using** *cond-exp-real*[*THEN* *AE-symmetric*, *OF integrable-norm*[*OF integrable*]] *real-cond-exp-ge-c*[*OF integrable-norm*[*OF integrable*], *of 0*] *norm-ge-zero* **by** *fastforce*  
**have** \*\*:  $A \in \text{sets } F \implies \int x \in A. |f\ x| \partial M = \int x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x \partial M$  **for**  $A$  **unfolding** *real-norm-def* **using** *assms* *integrable-abs* *real-cond-exp-intA* **by** *blast*  
  
**have** *norm-int*:  $A \in \text{sets } F \implies (\int x \in A. |f\ x| \partial M) = (\int^+ x \in A. |f\ x| \partial M)$  **for**  $A$  **using** *assms* **by** (*intro nn-set-integral-eq-set-integral*[*symmetric*], *blast*, *fastforce*) (*meson subalg subalgebra-def subsetD*)  
  
**have**  $AE\ x\ in\ M. \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x \geq 0$  **using** *int real-cond-exp-ge-c* **by** *force*  
**hence** *cond-exp-norm-int*:  $A \in \text{sets } F \implies (\int x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x \partial M) = (\int^+ x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x \partial M)$  **for**  $A$  **using** *assms* **by** (*intro nn-set-integral-eq-set-integral*[*symmetric*], *blast*, *fastforce*) (*meson subalg subalgebra-def subsetD*)  
  
**have**  $A \in \text{sets } F \implies \int^+ x \in A. |f\ x| \partial M = \int^+ x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x \partial M$  **for**  $A$  **using** \*\* *norm-int cond-exp-norm-int* **by** (*auto simp add: nn-integral-set-ennreal*)  
**moreover** **have**  $(\lambda x. \text{ennreal } |f\ x|) \in \text{borel-measurable } M$  **by** *measurable*  
**moreover** **have**  $(\lambda x. \text{ennreal } (\text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x)) \in \text{borel-measurable } F$  **by** *measurable*  
**ultimately** **have**  $AE\ x\ in\ M. \text{nn-cond-exp } M F (\lambda x. \text{ennreal } |f\ x|)\ x = \text{real-cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$  **by** (*intro nn-cond-exp-charact*[*THEN AE-symmetric*], *auto*)  
**hence**  $AE\ x\ in\ M. \text{nn-cond-exp } M F (\lambda x. \text{ennreal } |f\ x|)\ x \leq \text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$  **using** *cond-exp-real*[*OF int*] **by** *force*  
**moreover** **have**  $AE\ x\ in\ M. |\text{real-cond-exp } M F f\ x| = \text{norm}(\text{cond-exp } M F f\ x)$  **unfolding** *real-norm-def* **using** *cond-exp-real*[*OF assms*] \* **by** *force*  
**ultimately** **have**  $AE\ x\ in\ M. \text{ennreal}(\text{norm}(\text{cond-exp } M F f\ x)) \leq \text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$  **using** *real-cond-exp-abs*[*OF assms*][*THEN borel-measurable-integrable*] **by** *fastforce*  
**hence**  $AE\ x\ in\ M. \text{enn2real}(\text{ennreal}(\text{norm}(\text{cond-exp } M F f\ x))) \leq \text{enn2real}(\text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x)$  **using** *ennreal-le-iff2* **by** *force*  
**thus** *?thesis* **using** \* **by** *fastforce*  
**qed**

**lemma** *cond-exp-contraction-simple*:

**fixes**  $f :: 'a \Rightarrow 'b::\{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *simple-function*  $M f$  *emeasure*  $M \{y \in \text{space } M. f\ y \neq 0\} \neq \infty$   
**shows**  $AE\ x\ in\ M. \text{norm}(\text{cond-exp } M F f\ x) \leq \text{cond-exp } M F (\lambda x. \text{norm}(f\ x))\ x$

**using** *assms*  
**proof** (*induction rule: integrable-simple-function-induct*)  
**case** (*cong f g*)  
**hence** *ae: AE x in M. f x = g x by blast*  
**hence** *AE x in M. cond-exp M F f x = cond-exp M F g x using cong has-cond-exp-simple*  
**by** (*subst cond-exp-cong-AE*) (*auto intro!: has-cond-expD(2)*)  
**hence** *AE x in M. norm (cond-exp M F f x) = norm (cond-exp M F g x) by*  
*force*  
**moreover have** *AE x in M. cond-exp M F (λx. norm (f x)) x = cond-exp M F*  
*(λx. norm (g x)) x using ae cong has-cond-exp-simple by (subst cond-exp-cong-AE)*  
*(auto dest: has-cond-expD)*  
**ultimately show** *?case using cong(6) by fastforce*  
**next**  
**case** (*indicator A y*)  
**hence** *AE x in M. cond-exp M F (λa. indicator A a \*<sub>R</sub> y) x = cond-exp M F*  
*(indicator A) x \*<sub>R</sub> y by blast*  
**hence** *\*: AE x in M. norm (cond-exp M F (λa. indicat-real A a \*<sub>R</sub> y) x) ≤ norm y*  
*\* cond-exp M F (λx. norm (indicat-real A x)) x using cond-exp-contraction-real[OF*  
*integrable-real-indicator, OF indicator] by fastforce*  
  
**have** *AE x in M. norm y \* cond-exp M F (λx. norm (indicat-real A x)) x = norm*  
*y \* real-cond-exp M F (λx. norm (indicat-real A x)) x using cond-exp-real[OF*  
*integrable-real-indicator, OF indicator] by fastforce*  
**moreover have** *AE x in M. cond-exp M F (λx. norm y \* norm (indicat-real*  
*A x)) x = real-cond-exp M F (λx. norm y \* norm (indicat-real A x)) x using*  
*indicator by (intro cond-exp-real, auto)*  
**ultimately have** *AE x in M. norm y \* cond-exp M F (λx. norm (indicat-real A*  
*x)) x = cond-exp M F (λx. norm y \* norm (indicat-real A x)) x using real-cond-exp-cmult[of*  
*λx. norm (indicat-real A x) norm y] indicator by fastforce*  
**moreover have** *(λx. norm y \* norm (indicat-real A x)) = (λx. norm (indicat-real*  
*A x \*<sub>R</sub> y)) by force*  
**ultimately show** *?case using \* by force*  
**next**  
**case** (*add u v*)  
**have** *AE x in M. norm (cond-exp M F (λa. u a + v a) x) = norm (cond-exp M*  
*F u x + cond-exp M F v x) using has-cond-exp-charact(2)[OF has-cond-exp-add,*  
*OF has-cond-exp-simple(1,1), OF add(1,2,3,4)] by fastforce*  
**moreover have** *AE x in M. norm (cond-exp M F u x + cond-exp M F v x) ≤*  
*norm (cond-exp M F u x) + norm (cond-exp M F v x) using norm-triangle-ineq*  
**by blast**  
**moreover have** *AE x in M. norm (cond-exp M F u x) + norm (cond-exp M F v*  
*x) ≤ cond-exp M F (λx. norm (u x)) x + cond-exp M F (λx. norm (v x)) x using*  
*add(6,7) by fastforce*  
**moreover have** *AE x in M. cond-exp M F (λx. norm (u x)) x + cond-exp M F*  
*(λx. norm (v x)) x = cond-exp M F (λx. norm (u x) + norm (v x)) x using in-*  
*tegrable-simple-function[OF add(1,2)] integrable-simple-function[OF add(3,4)] by*  
*(intro has-cond-exp-charact(2)[OF has-cond-exp-add[OF has-cond-exp-charact(1,1)],*  
*THEN AE-symmetric], auto intro: has-cond-exp-real)*  
**moreover have** *AE x in M. cond-exp M F (λx. norm (u x) + norm (v x)) x =*

*cond-exp*  $M F (\lambda x. \text{norm } (u x + v x)) x$  **using** *add*(5) *integrable-simple-function*[*OF add*(1,2)] *integrable-simple-function*[*OF add*(3,4)] **by** (*intro cond-exp-cong*, *auto*)  
**ultimately show** ?*case* **by** *force*  
**qed**

**lemma** *has-cond-exp-simple-lim*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*[*measurable*]: *integrable*  $M f$   
**and**  $\bigwedge i. \text{simple-function } M (s i)$   
**and**  $\bigwedge i. \text{emeasure } M \{y \in \text{space } M. s i y \neq 0\} \neq \infty$   
**and**  $\bigwedge x. x \in \text{space } M \implies (\lambda i. s i x) \longrightarrow f x$   
**and**  $\bigwedge x i. x \in \text{space } M \implies \text{norm } (s i x) \leq 2 * \text{norm } (f x)$   
**obtains**  $r$   
**where** *strict-mono*  $r$  *has-cond-exp*  $M F f (\lambda x. \text{lim } (\lambda i. \text{cond-exp } M F (s (r i)) x))$   
 $AE x \text{ in } M. \text{convergent } (\lambda i. \text{cond-exp } M F (s (r i)) x)$

**proof** –

**have** [*measurable*]:  $(s i) \in \text{borel-measurable } M$  **for**  $i$  **using** *assms*(2) **by** (*simp add: borel-measurable-simple-function*)  
**have** *integrable-s*: *integrable*  $M (\lambda x. s i x)$  **for**  $i$  **using** *assms integrable-simple-function*  
**by** *blast*  
**have** *integrable-4f*: *integrable*  $M (\lambda x. 4 * \text{norm } (f x))$  **using** *assms*(1) **by** *simp*  
**have** *integrable-2f*: *integrable*  $M (\lambda x. 2 * \text{norm } (f x))$  **using** *assms*(1) **by** *simp*  
**have** *integrable-2-cond-exp-norm-f*: *integrable*  $M (\lambda x. 2 * \text{cond-exp } M F (\lambda x. \text{norm } (f x)) x)$  **by** *fast*

**have** *emeasure*  $M \{y \in \text{space } M. s i y - s j y \neq 0\} \leq \text{emeasure } M \{y \in \text{space } M. s i y \neq 0\} + \text{emeasure } M \{y \in \text{space } M. s j y \neq 0\}$  **for**  $i j$  **using** *simple-functionD*(2)[*OF assms*(2)] **by** (*intro order-trans*[*OF emeasure-mono emeasure-subadditive*], *auto*)

**hence** *fin-sup*: *emeasure*  $M \{y \in \text{space } M. s i y - s j y \neq 0\} \neq \infty$  **for**  $i j$  **using** *assms*(3) **by** (*metis* (*mono-tags*) *ennreal-add-eq-top linorder-not-less top.not-eq-extremum infinity-ennreal-def*)

**have** *emeasure*  $M \{y \in \text{space } M. \text{norm } (s i y - s j y) \neq 0\} \leq \text{emeasure } M \{y \in \text{space } M. s i y \neq 0\} + \text{emeasure } M \{y \in \text{space } M. s j y \neq 0\}$  **for**  $i j$  **using** *simple-functionD*(2)[*OF assms*(2)] **by** (*intro order-trans*[*OF emeasure-mono emeasure-subadditive*], *auto*)

**hence** *fin-sup-norm*: *emeasure*  $M \{y \in \text{space } M. \text{norm } (s i y - s j y) \neq 0\} \neq \infty$  **for**  $i j$  **using** *assms*(3) **by** (*metis* (*mono-tags*) *ennreal-add-eq-top linorder-not-less top.not-eq-extremum infinity-ennreal-def*)

**have** *Cauchy*: *Cauchy*  $(\lambda n. s n x)$  **if**  $x \in \text{space } M$  **for**  $x$  **using** *assms*(4) *LIM-SEQ-imp-Cauchy* **that** **by** *blast*

**hence** *bounded-range-s*: *bounded*  $(\text{range } (\lambda n. s n x))$  **if**  $x \in \text{space } M$  **for**  $x$  **using** *that cauchy-imp-bounded* **by** *fast*

Since the sequence  $\lambda n. s n x$  is Cauchy for almost all  $x$ , we know that the diameter tends to zero almost everywhere.

Dominated convergence tells us that the integral of the diameter also converges to zero.

```

have AE  $x$  in  $M$ .  $(\lambda n. \text{diameter } \{s\ i\ x \mid i. n \leq i\}) \longrightarrow 0$  using Cauchy cauchy-iff-diameter-tends-to-zero-and-bounded by fast
moreover have  $(\lambda x. \text{diameter } \{s\ i\ x \mid i. n \leq i\}) \in \text{borel-measurable } M$  for  $n$ 
using bounded-range-s borel-measurable-diameter by measurable
moreover have AE  $x$  in  $M$ .  $\text{norm } (\text{diameter } \{s\ i\ x \mid i. n \leq i\}) \leq 4 * \text{norm } (f\ x)$  for  $n$ 
proof -
{
  fix  $x$  assume  $x: x \in \text{space } M$ 
    have  $\text{diameter } \{s\ i\ x \mid i. n \leq i\} \leq 2 * \text{norm } (f\ x) + 2 * \text{norm } (f\ x)$ 
by (intro diameter-le, blast, subst dist-norm[symmetric], intro dist-triangle3[THEN order-trans, of 0], intro add-mono) (auto intro: assms(5)[OF x])
    hence  $\text{norm } (\text{diameter } \{s\ i\ x \mid i. n \leq i\}) \leq 4 * \text{norm } (f\ x)$  using diameter-ge-0[OF bounded-subset[OF bounded-range-s], OF x, of {s i x | i. n ≤ i}] by force
  }
  thus ?thesis by fast
qed
ultimately have diameter-tendsto-zero:  $(\lambda n. \text{LINT } x | M. \text{diameter } \{s\ i\ x \mid i. n \leq i\}) \longrightarrow 0$  by (intro integral-dominated-convergence[OF borel-measurable-const[of 0] - integrable-4f, simplified]) (fast+)

  have diameter-integrable: integrable  $M$   $(\lambda x. \text{diameter } \{s\ i\ x \mid i. n \leq i\})$  for  $n$ 
using assms(1,5)
  by (intro integrable-bound-diameter[OF bounded-range-s integrable-2f], auto)

  have dist-integrable: integrable  $M$   $(\lambda x. \text{dist } (s\ i\ x) (s\ j\ x))$  for  $i\ j$  using assms(5)
dist-triangle3[of s i - 0, THEN order-trans, OF add-mono, of - 2 * norm (f -)]
  by (intro Bochner-Integration.integrable-bound[OF integrable-4f]) fastforce+

```

Since  $\text{cond-exp } M\ F$  is a contraction for simple functions, the following sequence of integral values is also Cauchy.

This follows, since the distance between the terms of this sequence are always less than or equal to the diameter, which itself converges to zero.

Hence, we obtain a subsequence which is Cauchy almost everywhere.

```

have  $\exists N. \forall i \geq N. \forall j \geq N. \text{LINT } x | M. \text{norm } (\text{cond-exp } M\ F\ (s\ i)\ x - \text{cond-exp } M\ F\ (s\ j)\ x) < e$  if e-pos:  $e > 0$  for  $e$ 
proof -
  obtain  $N$  where *:  $\text{LINT } x | M. \text{diameter } \{s\ i\ x \mid i. n \leq i\} < e$  if  $n \geq N$  for  $n$ 
using that order-tendsto-iff[THEN iffD1, OF diameter-tendsto-zero, unfolded eventually-sequentially] e-pos by presburger
  {
    fix  $i\ j\ x$  assume asm:  $i \geq N\ j \geq N\ x \in \text{space } M$ 
    have case-prod dist ' ({s i x | i. N ≤ i} × {s i x | i. N ≤ i}) = case-prod (λi j. dist (s i x) (s j x)) ' ({N..} × {N..}) by fast
  }

```

**hence**  $\text{diameter } \{s \ i \ x \mid i. N \leq i\} = (\text{SUP } (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i \ x) (s \ j \ x))$  **unfolding** *diameter-def* **by** *auto*  
**moreover** **have**  $(\text{SUP } (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i \ x) (s \ j \ x)) \geq \text{dist } (s \ i \ x) (s \ j \ x)$  **using** *asm bounded-imp-bdd-above*[*OF bounded-imp-dist-bounded, OF bounded-range-s*] **by** (*intro cSup-upper, auto*)  
**ultimately** **have**  $\text{diameter } \{s \ i \ x \mid i. N \leq i\} \geq \text{dist } (s \ i \ x) (s \ j \ x)$  **by** *presburger*  
**}**  
**hence**  $\text{LINT } x \mid M. \text{dist } (s \ i \ x) (s \ j \ x) < e$  **if**  $i \geq N \ j \geq N$  **for**  $i \ j$  **using** *that \** **by** (*intro integral-mono*[*OF dist-integrable diameter-integrable, THEN order.strict-trans1*], *blast+*)  
**moreover** **have**  $\text{LINT } x \mid M. \text{norm } (\text{cond-exp } M \ F \ (s \ i) \ x - \text{cond-exp } M \ F \ (s \ j) \ x) \leq \text{LINT } x \mid M. \text{dist } (s \ i \ x) (s \ j \ x)$  **for**  $i \ j$   
**proof**  $-$   
**have**  $\text{LINT } x \mid M. \text{norm } (\text{cond-exp } M \ F \ (s \ i) \ x - \text{cond-exp } M \ F \ (s \ j) \ x) = \text{LINT } x \mid M. \text{norm } (\text{cond-exp } M \ F \ (s \ i) \ x + -1 * \text{cond-exp } M \ F \ (s \ j) \ x)$  **unfolding** *dist-norm* **by** *simp*  
**also** **have**  $\dots = \text{LINT } x \mid M. \text{norm } (\text{cond-exp } M \ F \ (\lambda x. s \ i \ x - s \ j \ x) \ x)$  **using** *has-cond-exp-charact(2)*[*OF has-cond-exp-add*[*OF - has-cond-exp-scaleR-right, OF has-cond-exp-charact(1,1), OF has-cond-exp-simple(1,1)*][*OF assms(2,3)*]], *THEN AE-symmetric, of i -1 j*] **by** (*intro integral-cong-AE*) *force+*  
**also** **have**  $\dots \leq \text{LINT } x \mid M. \text{cond-exp } M \ F \ (\lambda x. \text{norm } (s \ i \ x - s \ j \ x)) \ x$  **using** *cond-exp-contraction-simple*[*OF - fin-sup, of i j*] *integrable-cond-exp assms(2)* **by** (*intro integral-mono-AE, fast+*)  
**also** **have**  $\dots = \text{LINT } x \mid M. \text{norm } (s \ i \ x - s \ j \ x)$  **unfolding** *set-integral-space(1)*[*OF integrable-cond-exp, symmetric*] *set-integral-space*[*OF dist-integrable*][*unfolded dist-norm*], *symmetric*] **by** (*intro has-cond-expD(1)*[*OF has-cond-exp-simple*[*OF - fin-sup-norm*], *symmetric*]) (*metis assms(2) simple-function-compose1 simple-function-diff, metis sets.top subalg subalgebra-def*)  
**finally** **show** *?thesis* **unfolding** *dist-norm* .  
**qed**  
**ultimately** **show** *?thesis* **using** *order.strict-trans1* **by** *meson*  
**qed**  
**then** **obtain**  $r$  **where** *strict-mono-r: strict-mono r* **and** *AE-Cauchy: AE x in M. Cauchy ( $\lambda i. \text{cond-exp } M \ F \ (s \ (r \ i)) \ x$ )*  
**by** (*rule cauchy-L1-AE-cauchy-subseq*[*OF integrable-cond-exp*], *auto*)  
**hence** *ae-lim-cond-exp: AE x in M. ( $\lambda n. \text{cond-exp } M \ F \ (s \ (r \ n)) \ x$ )  $\longrightarrow \lim (\lambda n. \text{cond-exp } M \ F \ (s \ (r \ n)) \ x)$*  **using** *Cauchy-convergent-iff convergent-LIMSEQ-iff* **by** *fastforce*

Now that we have a candidate for the conditional expectation, we must show that it actually has the required properties.

Dominated convergence shows that this limit is indeed integrable.

Here, we again use the fact that conditional expectation is a contraction on simple functions.

**have** *cond-exp-bounded: AE x in M. norm (cond-exp M F (s (r n)) x)  $\leq$  cond-exp M F ( $\lambda x. 2 * \text{norm } (f \ x)$ ) x* **for**  $n$

**proof** –  
**have**  $AE\ x\ in\ M. \text{norm}\ (cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x) \leq cond\text{-}exp\ M\ F\ (\lambda x. \text{norm}\ (s\ (r\ n)\ x))\ x$  **by** (rule *cond-exp-contraction-simple*[*OF* *assms*(2,3)])  
**moreover have**  $AE\ x\ in\ M. \text{real}\text{-}cond\text{-}exp\ M\ F\ (\lambda x. \text{norm}\ (s\ (r\ n)\ x))\ x \leq \text{real}\text{-}cond\text{-}exp\ M\ F\ (\lambda x. 2 * \text{norm}\ (f\ x))\ x$  **using** *integrable-s integrable-2f assms*(5)  
**by** (*intro real-cond-exp-mono, auto*)  
**ultimately show** *?thesis* **using** *cond-exp-real*[*OF* *integrable-norm, OF integrable-s, of r n*] *cond-exp-real*[*OF* *integrable-2f*] **by** *force*  
**qed**  
**have** *lim-integrable: integrable*  $M\ (\lambda x. \lim\ (\lambda i. cond\text{-}exp\ M\ F\ (s\ (r\ i))\ x))$   
**by** (*intro integrable-dominated-convergence*[*OF* - *borel-measurable-cond-exp'* *integrable-cond-exp ae-lim-cond-exp cond-exp-bounded*], *simp*)

Moreover, we can use the DCT twice to show that the conditional expectation property holds, i.e. the value of the integral of the candidate, agrees with  $f$  on sets  $A \in \text{sets } F$ .

{  
**fix**  $A$  **assume** *A-in-sets-F*:  $A \in \text{sets } F$   
**have**  $AE\ x\ in\ M. \text{norm}\ (indicator\ A\ x *_{\mathbb{R}} cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x) \leq cond\text{-}exp\ M\ F\ (\lambda x. 2 * \text{norm}\ (f\ x))\ x$  **for**  $n$   
**proof** –  
**have**  $AE\ x\ in\ M. \text{norm}\ (indicator\ A\ x *_{\mathbb{R}} cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x) \leq \text{norm}\ (cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)$  **unfolding** *indicator-def* **by** *simp*  
**thus** *?thesis* **using** *cond-exp-bounded*[*of n*] **by** *force*  
**qed**  
**hence** *lim-cond-exp-int*:  $(\lambda n. LINT\ x:A|M. cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x) \longrightarrow LINT\ x:A|M. \lim\ (\lambda n. cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)$   
**using** *ae-lim-cond-exp measurable-from-subalg*[*OF* *subalg borel-measurable-indicator, OF A-in-sets-F*] *cond-exp-bounded*  
**unfolding** *set-lebesgue-integral-def*  
**by** (*intro integral-dominated-convergence*[*OF* *borel-measurable-scaleR borel-measurable-scaleR integrable-cond-exp*]) (*fastforce simp add: tendsto-scaleR*) +  
  
**have**  $AE\ x\ in\ M. \text{norm}\ (indicator\ A\ x *_{\mathbb{R}} s\ (r\ n)\ x) \leq 2 * \text{norm}\ (f\ x)$  **for**  $n$   
**proof** –  
**have**  $AE\ x\ in\ M. \text{norm}\ (indicator\ A\ x *_{\mathbb{R}} s\ (r\ n)\ x) \leq \text{norm}\ (s\ (r\ n)\ x)$   
**unfolding** *indicator-def* **by** *simp*  
**thus** *?thesis* **using** *assms*(5)[*of - r n*] **by** *fastforce*  
**qed**  
**hence** *lim-s-int*:  $(\lambda n. LINT\ x:A|M. s\ (r\ n)\ x) \longrightarrow LINT\ x:A|M. f\ x$   
**using** *measurable-from-subalg*[*OF* *subalg borel-measurable-indicator, OF A-in-sets-F*] *LIMSEQ-subseq-LIMSEQ*[*OF* *assms*(4) *strict-mono-r*] *assms*(5)  
**unfolding** *set-lebesgue-integral-def comp-def*  
**by** (*intro integral-dominated-convergence*[*OF* *borel-measurable-scaleR borel-measurable-scaleR integrable-2f*]) (*fastforce simp add: tendsto-scaleR*) +  
  
**have**  $LINT\ x:A|M. \lim\ (\lambda n. cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x) = \lim\ (\lambda n. LINT\ x:A|M. cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)$  **using** *limI*[*OF* *lim-cond-exp-int*] **by** *argo*  
**also have**  $\dots = \lim\ (\lambda n. LINT\ x:A|M. s\ (r\ n)\ x)$  **using** *has-cond-expD*(1)[*OF*



```

has-cond-exp-simple[OF assms(2,3)] A-in-sets-F, symmetric] by presburger
  also have ... = LINT x:A|M. f x using limI[OF lim-s-int] by argo
  finally have LINT x:A|M. lim (λn. cond-exp M F (s (r n)) x) = LINT x:A|M.
f x .
}

```

Putting it all together, we have the statement we are looking for.

```

  hence has-cond-exp M F f (λx. lim (λi. cond-exp M F (s (r i)) x)) using
assms(1) lim-integrable by (intro has-cond-expI', auto)
  thus thesis using AE-Cauchy Cauchy-convergent strict-mono-r by (auto intro!:
that)
qed

```

Now, we can show that the conditional expectation is well-defined for all integrable functions.

```

corollary has-cond-expI:
  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes integrable M f
  shows has-cond-exp M F f (cond-exp M F f)
proof -
  obtain s where s-is: ∧i. simple-function M (s i) ∧i. emeasure M {y ∈ space M.
s i y ≠ 0} ≠ ∞ ∧x. x ∈ space M ⇒ (λi. s i x) ⟶ f x ∧x i. x ∈ space M ⇒
norm (s i x) ≤ 2 * norm (f x) using integrable-implies-simple-function-sequence[OF
assms] by blast
  show ?thesis using has-cond-exp-simple-lim[OF assms s-is] has-cond-exp-charact(1)
by metis
qed

```

## 4.2 Properties

The defining property of the conditional expectation now always holds, given that the function  $f$  is integrable.

```

lemma cond-exp-set-integral:
  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes integrable M f A ∈ sets F
  shows (∫ x ∈ A. f x ∂M) = (∫ x ∈ A. cond-exp M F f x ∂M)
  using has-cond-expD(1)[OF has-cond-expI, OF assms] by argo

```

The following property of the conditional expectation is called the "Tower Property".

```

lemma cond-exp-nested-subalg:
  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes integrable M f subalgebra M G subalgebra G F
  shows AE ξ in M. cond-exp M F f ξ = cond-exp M F (cond-exp M G f) ξ
  using has-cond-expI assms sigma-finite-subalgebra-def by (auto intro!: has-cond-exp-nested-subalg[THEN
has-cond-exp-charact(2), THEN AE-symmetric] sigma-finite-subalgebra.has-cond-expI[OF
sigma-finite-subalgebra.intro[OF assms(2)]] nested-subalg-is-sigma-finite)

```

The conditional expectation is linear.

**lemma** *cond-exp-add*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   
**shows**  $AE\ x\ in\ M. \text{cond-exp}\ M\ F\ (\lambda x. f\ x + g\ x)\ x = \text{cond-exp}\ M\ F\ f\ x + \text{cond-exp}\ M\ F\ g\ x$   
**using** *has-cond-exp-add*[*OF* *has-cond-expI*(1,1), *OF* *assms*, *THEN* *has-cond-exp-charact*(2)]  
**.**

**lemma** *cond-exp-diff*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   
**shows**  $AE\ x\ in\ M. \text{cond-exp}\ M\ F\ (\lambda x. f\ x - g\ x)\ x = \text{cond-exp}\ M\ F\ f\ x - \text{cond-exp}\ M\ F\ g\ x$   
**using** *has-cond-exp-add*[*OF* - *has-cond-exp-scaleR-right*, *OF* *has-cond-expI*(1,1), *OF* *assms*, *THEN* *has-cond-exp-charact*(2), *of* -1] **by** *simp*

**lemma** *cond-exp-diff'*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   
**shows**  $AE\ x\ in\ M. \text{cond-exp}\ M\ F\ (f - g)\ x = \text{cond-exp}\ M\ F\ f\ x - \text{cond-exp}\ M\ F\ g\ x$   
**unfolding** *fun-diff-def* **using** *assms* **by** (*rule* *cond-exp-diff*)

**lemma** *cond-exp-scaleR-left*:

**fixes**  $f :: 'a \Rightarrow \text{real}$   
**assumes** *integrable*  $M f$   
**shows**  $AE\ x\ in\ M. \text{cond-exp}\ M\ F\ (\lambda x. f\ x *_{\mathbb{R}} c)\ x = \text{cond-exp}\ M\ F\ f\ x *_{\mathbb{R}} c$   
**using** *cond-exp-set-integral*[*OF* *assms*] *subalg* *assms* **unfolding** *subalgebra-def*  
**by** (*intro* *cond-exp-charact*,  
*subst* *set-integral-scaleR-left*, *blast*, *intro* *assms*,  
*subst* *set-integral-scaleR-left*, *blast*, *intro* *integrable-cond-exp*)  
*auto*

The conditional expectation operator is a contraction, i.e. a bounded linear operator with operator norm less than or equal to 1.

To show this we first obtain a subsequence  $\lambda x\ i. s\ (r\ i)\ x$ , such that  $\lambda i. \text{cond-exp}\ M\ F\ (s\ (r\ i))\ x$  converges to  $\text{cond-exp}\ M\ F\ f\ x$  a.e. Afterwards, we obtain a sub-subsequence  $\lambda x\ i. s\ (r\ (r'\ i))\ x$ , such that  $\lambda i. \text{cond-exp}\ M\ F\ (\lambda x. \text{norm}\ (s\ (r\ i)))\ x$  converges to  $\text{cond-exp}\ M\ F\ (\lambda x. \text{norm}\ (f\ x))\ x$  a.e. Finally, we show that the inequality holds by showing that the terms of the subsequences obey the inequality and the fact that a subsequence of a convergent sequence converges to the same limit.

**lemma** *cond-exp-contraction*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology}, \text{banach}\}$   
**assumes** *integrable*  $M f$   
**shows**  $AE\ x\ in\ M. \text{norm}\ (\text{cond-exp}\ M\ F\ f\ x) \leq \text{cond-exp}\ M\ F\ (\lambda x. \text{norm}\ (f\ x))$

$x$

**proof** –

**obtain**  $s$  **where**  $s$ :  $\bigwedge i. \text{simple-function } M (s\ i) \bigwedge i. \text{emeasure } M \{y \in \text{space } M. s\ i\ y \neq 0\} \neq \infty \bigwedge x. x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x \bigwedge i. x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$

**by** (*blast intro: integrable-implies-simple-function-sequence[OF assms]*)

**obtain**  $r$  **where**  $r$ : *strict-mono*  $r$  **and** *has-cond-exp*  $M\ F\ f\ (\lambda x. \lim (\lambda i. \text{cond-exp } M\ F\ (s\ (r\ i))\ x))\ AE\ x\ \text{in } M. (\lambda i. \text{cond-exp } M\ F\ (s\ (r\ i))\ x) \longrightarrow \lim (\lambda i. \text{cond-exp } M\ F\ (s\ (r\ i))\ x)$

**using** *has-cond-exp-simple-lim[OF assms s]* **unfolding** *convergent-LIMSEQ-iff* **by** *blast*

**hence**  $r\text{-tendsto}$ :  $AE\ x\ \text{in } M. (\lambda i. \text{cond-exp } M\ F\ (s\ (r\ i))\ x) \longrightarrow \text{cond-exp } M\ F\ f\ x$  **using** *has-cond-exp-charact(2)* **by** *force*

**have**  $\text{norm-s-r}$ :  $\bigwedge i. \text{simple-function } M (\lambda x. \text{norm } (s\ (r\ i)\ x)) \bigwedge i. \text{emeasure } M \{y \in \text{space } M. \text{norm } (s\ (r\ i)\ y) \neq 0\} \neq \infty \bigwedge x. x \in \text{space } M \implies (\lambda i. \text{norm } (s\ (r\ i)\ x)) \longrightarrow \text{norm } (f\ x) \bigwedge i. x \in \text{space } M \implies \text{norm } (\text{norm } (s\ (r\ i)\ x)) \leq 2 * \text{norm } (\text{norm } (f\ x))$

**using**  $s$  **by** (*auto intro: LIMSEQ-subseq-LIMSEQ[OF tendsto-norm r, unfolded comp-def] simple-function-compose1*)

**obtain**  $r'$  **where**  $r'$ : *strict-mono*  $r'$  **and** *has-cond-exp*  $M\ F\ (\lambda x. \text{norm } (f\ x)) (\lambda x. \lim (\lambda i. \text{cond-exp } M\ F\ (\lambda x. \text{norm } (s\ (r\ (r'\ i))\ x))\ x))\ AE\ x\ \text{in } M. (\lambda i. \text{cond-exp } M\ F\ (\lambda x. \text{norm } (s\ (r\ (r'\ i))\ x))\ x) \longrightarrow \lim (\lambda i. \text{cond-exp } M\ F\ (\lambda x. \text{norm } (s\ (r\ (r'\ i))\ x))\ x)$  **using** *has-cond-exp-simple-lim[OF integrable-norm norm-s-r, OF assms]* **unfolding** *convergent-LIMSEQ-iff* **by** *blast*

**hence**  $r'\text{-tendsto}$ :  $AE\ x\ \text{in } M. (\lambda i. \text{cond-exp } M\ F\ (\lambda x. \text{norm } (s\ (r\ (r'\ i))\ x))\ x) \longrightarrow \text{cond-exp } M\ F\ (\lambda x. \text{norm } (f\ x))\ x$  **using** *has-cond-exp-charact(2)* **by** *force*

**have**  $AE\ x\ \text{in } M. \forall i. \text{norm } (\text{cond-exp } M\ F\ (s\ (r\ (r'\ i))\ x)) \leq \text{cond-exp } M\ F\ (\lambda x. \text{norm } (s\ (r\ (r'\ i))\ x))\ x$  **using**  $s$  **by** (*auto intro: cond-exp-contraction-simple simp add: AE-all-countable*)

**moreover** **have**  $AE\ x\ \text{in } M. (\lambda i. \text{norm } (\text{cond-exp } M\ F\ (s\ (r\ (r'\ i))\ x))\ x) \longrightarrow \text{norm } (\text{cond-exp } M\ F\ f\ x)$  **using**  $r\text{-tendsto}$  *LIMSEQ-subseq-LIMSEQ[OF tendsto-norm r', unfolded comp-def]* **by** *fast*

**ultimately show** *?thesis* **using** *LIMSEQ-le r'-tendsto* **by** *fast*  
**qed**

The following lemmas are called "pulling out whats known". We first show the statement for real-valued functions using the lemma *real-cond-exp-intg*, which is already present. We then show it for arbitrary  $g$  using the lecture notes of Gordan Zitkovic for the course "Theory of Probability I".

**lemma** *cond-exp-measurable-mult*:

**fixes**  $f\ g :: 'a \Rightarrow \text{real}$

**assumes** [*measurable*]: *integrable*  $M (\lambda x. f\ x * g\ x)$  *integrable*  $M\ g\ f \in \text{borel-measurable } F$

**shows** *integrable*  $M (\lambda x. f\ x * \text{cond-exp } M\ F\ g\ x)$

$AE\ x\ \text{in } M. \text{cond-exp } M\ F\ (\lambda x. f\ x * g\ x)\ x = f\ x * \text{cond-exp } M\ F\ g\ x$

**proof** –

**show** *integrable*: *integrable*  $M$   $(\lambda x. f x * \text{cond-exp } M F g x)$  **using** *cond-exp-real*[*OF* *assms*(2)] **by** (*intro integrable-cong-AE-imp*[*OF* *real-cond-exp-intg*(1), *OF* *assms*(1,3) *assms*(2)[*THEN borel-measurable-integrable*]] *measurable-from-subalg*[*OF* *subalg*]) *auto*

**interpret** *sigma-finite-measure restr-to-subalg*  $M F$  **by** (*rule sigma-fin-subalg*)

{  
**fix**  $A$  **assume** *asm*:  $A \in \text{sets } F$   
**hence** *asm'*:  $A \in \text{sets } M$  **using** *subalg* **by** (*fastforce simp add: subalgebra-def*)  
**have** *set-lebesgue-integral*  $M A$  (*cond-exp*  $M F$   $(\lambda x. f x * g x)$ ) = *set-lebesgue-integral*  $M A$   $(\lambda x. f x * g x)$  **by** (*simp add: cond-exp-set-integral*[*OF* *assms*(1) *asm*])  
**also have** ... = *set-lebesgue-integral*  $M A$   $(\lambda x. f x * \text{real-cond-exp } M F g x)$  **using** *borel-measurable-times*[*OF* *borel-measurable-indicator*[*OF* *asm*] *assms*(3)] *borel-measurable-integrable*[*OF* *assms*(2)] *integrable-mult-indicator*[*OF* *asm'* *assms*(1)]  
**by** (*fastforce simp add: set-lebesgue-integral-def mult.assoc[symmetric] intro: real-cond-exp-intg(2)[symmetric]*)  
**also have** ... = *set-lebesgue-integral*  $M A$   $(\lambda x. f x * \text{cond-exp } M F g x)$  **using** *cond-exp-real*[*OF* *assms*(2)] *asm'* *borel-measurable-cond-exp'* *borel-measurable-cond-exp2* *measurable-from-subalg*[*OF* *subalg* *assms*(3)] **by** (*auto simp add: set-lebesgue-integral-def intro: integral-cong-AE*)

**finally have** *set-lebesgue-integral*  $M A$  (*cond-exp*  $M F$   $(\lambda x. f x * g x)$ ) =  $\int_{x \in A. (f x * \text{cond-exp } M F g x) \partial M}$ .

}  
**hence** *AE*  $x$  *in* *restr-to-subalg*  $M F$ . *cond-exp*  $M F$   $(\lambda x. f x * g x)$   $x$  =  $f x * \text{cond-exp } M F g x$  **by** (*intro density-unique-banach integrable-cond-exp integrable-integrable-in-subalg subalg, measurable, simp add: set-lebesgue-integral-def integral-subalgebra2*[*OF* *subalg*] *sets-restr-to-subalg*[*OF* *subalg*])

**thus** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F$   $(\lambda x. f x * g x)$   $x$  =  $f x * \text{cond-exp } M F g x$  **by** (*rule AE-restr-to-subalg*[*OF* *subalg*])

**qed**

**lemma** *cond-exp-measurable-scaleR*:

**fixes**  $f :: 'a \Rightarrow \text{real}$  **and**  $g :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$   
**assumes** [*measurable*]: *integrable*  $M$   $(\lambda x. f x *_R g x)$  *integrable*  $M g f \in \text{borel-measurable } F$

**shows** *integrable*  $M$   $(\lambda x. f x *_R \text{cond-exp } M F g x)$

*AE*  $x$  *in*  $M$ . *cond-exp*  $M F$   $(\lambda x. f x *_R g x)$   $x$  =  $f x *_R \text{cond-exp } M F g x$

**proof** –

**let**  $?F = \text{restr-to-subalg } M F$   
**have** *subalg'*: *subalgebra*  $M$  (*restr-to-subalg*  $M F$ ) **by** (*metis sets-eq-imp-space-eq sets-restr-to-subalg subalg subalgebra-def*)

{  
**fix**  $z$  **assume** *asm*[*measurable*]: *integrable*  $M$   $(\lambda x. z x *_R g x)$   $z \in \text{borel-measurable } ?F$

**hence** *asm'*[*measurable*]:  $z \in \text{borel-measurable } F$  **using** *measurable-in-subalg'* *subalg* **by** *blast*

**have** *integrable*  $M$   $(\lambda x. z x *_R \text{cond-exp } M F g x)$  *LINT*  $x|M$ .  $z x *_R g x$  = *LINT*  $x|M$ .  $z x *_R \text{cond-exp } M F g x$

**proof** –

**obtain**  $s$  **where**  $s$ -*is*:  $\bigwedge i. \text{simple-function } ?F (s i) \bigwedge x. x \in \text{space } ?F \implies (\lambda i.$

$s \ i \ x) \longrightarrow z \ x \ \bigwedge i \ x. \ x \in \text{space } ?F \implies \text{norm } (s \ i \ x) \leq 2 * \text{norm } (z \ x)$  **using** *borel-measurable-implies-sequence-metric*[*OF asm(2), of 0*] **by force**

We need to apply the dominated convergence theorem twice, therefore we need to show the following prerequisites.

**have** *s-scaleR-g-tendsto*:  $AE \ x \ \text{in } M. (\lambda i. s \ i \ x *_{\mathbb{R}} g \ x) \longrightarrow z \ x *_{\mathbb{R}} g \ x$   
**using** *s-is(2)* **by** (*simp add: space-restr-to-subalg tendsto-scaleR*)

**have** *s-scaleR-cond-exp-g-tendsto*:  $AE \ x \ \text{in } ?F. (\lambda i. s \ i \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x) \longrightarrow z \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x$  **using** *s-is(2)* **by** (*simp add: tendsto-scaleR*)

**have** *s-scaleR-g-meas*:  $(\lambda x. s \ i \ x *_{\mathbb{R}} g \ x) \in \text{borel-measurable } M$  **for** *i* **using** *s-is(1)*[*THEN borel-measurable-simple-function, THEN subalg'[THEN measurable-from-subalg]*] **by** *simp*

**have** *s-scaleR-cond-exp-g-meas*:  $(\lambda x. s \ i \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x) \in \text{borel-measurable } ?F$  **for** *i* **using** *s-is(1)*[*THEN borel-measurable-simple-function*] *measurable-in-subalg*[*OF subalg borel-measurable-cond-exp*] **by** (*fastforce intro: borel-measurable-scaleR*)

**have** *s-scaleR-g-AE-bdd*:  $AE \ x \ \text{in } M. \text{norm } (s \ i \ x *_{\mathbb{R}} g \ x) \leq 2 * \text{norm } (z \ x *_{\mathbb{R}} g \ x)$  **for** *i* **using** *s-is(3)* **by** (*fastforce simp add: space-restr-to-subalg mult.assoc[symmetric] mult-right-mono*)

{

**fix** *i*

**have** *asm*:  $\text{integrable } M (\lambda x. \text{norm } (z \ x) * \text{norm } (g \ x))$  **using** *asm(1)*[*THEN integrable-norm*] **by** *simp*

**have**  $AE \ x \ \text{in } ?F. \text{norm } (s \ i \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x) \leq 2 * \text{norm } (z \ x) * \text{norm } (\text{cond-exp } M \ F \ g \ x)$  **using** *s-is(3)* **by** (*fastforce simp add: mult-mono*)

**moreover have**  $AE \ x \ \text{in } ?F. \text{norm } (z \ x) * \text{cond-exp } M \ F (\lambda x. \text{norm } (g \ x)) \ x = \text{cond-exp } M \ F (\lambda x. \text{norm } (z \ x) * \text{norm } (g \ x)) \ x$  **by** (*rule cond-exp-measurable-mult(2)*)[*THEN AE-symmetric, OF asm integrable-norm, OF assms(2), THEN AE-restr-to-subalg2*][*OF subalg*], *auto*)

**ultimately have**  $AE \ x \ \text{in } ?F. \text{norm } (s \ i \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x) \leq 2 * \text{cond-exp } M \ F (\lambda x. \text{norm } (z \ x *_{\mathbb{R}} g \ x)) \ x$  **using** *cond-exp-contraction*[*OF assms(2), THEN AE-restr-to-subalg2*][*OF subalg*] *order-trans*[*OF - mult-mono*] **by** *fastforce*

}

**note** *s-scaleR-cond-exp-g-AE-bdd* = *this*

In the following section we need to pay attention to which measures we are using for integration. The rhs is F-measurable while the lhs is only M-measurable.

{

**fix** *i*

**have** *s-meas-M*[*measurable*]:  $s \ i \in \text{borel-measurable } M$  **by** (*meson borel-measurable-simple-function measurable-from-subalg s-is(1) subalg'*)

**have** *s-meas-F*[*measurable*]:  $s \ i \in \text{borel-measurable } F$  **by** (*meson borel-measurable-simple-function measurable-in-subalg' s-is(1) subalg*)

**have** *s-scaleR-eq*:  $s \ i \ x *_{\mathbb{R}} h \ x = (\sum_{y \in s \ i} \text{space } M. (\text{indicator } (s \ i - \{y\} \cap \text{space } M) \ x *_{\mathbb{R}} y) *_{\mathbb{R}} h \ x)$  **if**  $x \in \text{space } M$  **for**  $x$  **and**  $h :: 'a \Rightarrow 'b$

**using** *simple-function-indicator-representation*[*OF s-is(1)*, *of x i*] **that unfolding**  
*space-restr-to-subalg scaleR-left.sum*[*of - - h x, symmetric*] **by** *presburger*

**have**  $LINT\ x|M. s\ i\ x *_R g\ x = LINT\ x|M. (\sum_{y \in s\ i} \text{space } M. \text{indicator } (s\ i - \{y\} \cap \text{space } M) \ x *_R y *_R g\ x)$  **using** *s-scaleR-eq* **by** (*intro Bochner-Integration.integral-cong*) *auto*

**also have**  $\dots = (\sum_{y \in s\ i} \text{space } M. LINT\ x|M. \text{indicator } (s\ i - \{y\} \cap \text{space } M) \ x *_R y *_R g\ x)$  **by** (*intro Bochner-Integration.integral-sum integrable-mult-indicator*[*OF - integrable-scaleR-right*] *assms(2)*) *simp*

**also have**  $\dots = (\sum_{y \in s\ i} \text{space } M. y *_R \text{set-lebesgue-integral } M\ (s\ i - \{y\} \cap \text{space } M) \ g)$  **by** (*simp only: set-lebesgue-integral-def*[*symmetric*]) *simp*

**also have**  $\dots = (\sum_{y \in s\ i} \text{space } M. y *_R \text{set-lebesgue-integral } M\ (s\ i - \{y\} \cap \text{space } M) \ (cond\text{-exp } M\ F\ g))$  **using** *assms(2)* *subalg borel-measurable-vimage*[*OF s-meas-F*] **by** (*subst cond-exp-set-integral, auto simp add: subalgebra-def*)

**also have**  $\dots = (\sum_{y \in s\ i} \text{space } M. LINT\ x|M. \text{indicator } (s\ i - \{y\} \cap \text{space } M) \ x *_R y *_R cond\text{-exp } M\ F\ g\ x)$  **by** (*simp only: set-lebesgue-integral-def*[*symmetric*]) *simp*

**also have**  $\dots = LINT\ x|M. (\sum_{y \in s\ i} \text{space } M. \text{indicator } (s\ i - \{y\} \cap \text{space } M) \ x *_R y *_R cond\text{-exp } M\ F\ g\ x)$  **by** (*intro Bochner-Integration.integral-sum*[*symmetric*] *integrable-mult-indicator*[*OF - integrable-scaleR-right*]) *auto*

**also have**  $\dots = LINT\ x|M. s\ i\ x *_R cond\text{-exp } M\ F\ g\ x$  **using** *s-scaleR-eq* **by** (*intro Bochner-Integration.integral-cong*) *auto*

**finally have**  $LINT\ x|M. s\ i\ x *_R g\ x = LINT\ x|?F. s\ i\ x *_R cond\text{-exp } M\ F\ g\ x$  **by** (*simp add: integral-subalgebra2*[*OF subalg*])

}

**note** *integral-s-eq = this*

Now we just plug in the results we obtained into DCT, and use the fact that limits are unique.

**show** *integrable*  $M\ (\lambda x. z\ x *_R cond\text{-exp } M\ F\ g\ x)$  **using** *s-scaleR-cond-exp-g-meas asm(2)* *borel-measurable-cond-exp'* **by** (*intro integrable-from-subalg*[*OF subalg*] *integrable-cond-exp integrable-dominated-convergence*[*OF - - s-scaleR-cond-exp-g-tendsto s-scaleR-cond-exp-g-AE-bdd*]) (*auto intro: measurable-from-subalg*[*OF subalg*] *integrable-in-subalg measurable-in-subalg subalg*)

**have**  $(\lambda i. LINT\ x|M. s\ i\ x *_R g\ x) \longrightarrow LINT\ x|M. z\ x *_R g\ x$  **using** *s-scaleR-g-meas asm(1)*[*THEN integrable-norm*] *asm' borel-measurable-cond-exp'* **by** (*intro integral-dominated-convergence*[*OF - - s-scaleR-g-tendsto s-scaleR-g-AE-bdd*]) (*auto intro: measurable-from-subalg*[*OF subalg*])

**moreover have**  $(\lambda i. LINT\ x|?F. s\ i\ x *_R cond\text{-exp } M\ F\ g\ x) \longrightarrow LINT\ x|?F. z\ x *_R cond\text{-exp } M\ F\ g\ x$  **using** *s-scaleR-cond-exp-g-meas asm(2)* *borel-measurable-cond-exp'* **by** (*intro integral-dominated-convergence*[*OF - - s-scaleR-cond-exp-g-tendsto s-scaleR-cond-exp-g-AE-bdd*]) (*auto intro: measurable-from-subalg*[*OF subalg*] *integrable-in-subalg measurable-in-subalg subalg*)

**ultimately show**  $LINT\ x|M. z\ x *_R g\ x = LINT\ x|M. z\ x *_R cond\text{-exp } M\ F\ g\ x$  **using** *integral-s-eq* **using** *subalg* **by** (*simp add: LIMSEQ-unique integral-subalgebra2*)

**qed**

}

**note**  $*$  = *this*

The main statement now follows with  $z = (\lambda x. \text{indicat-real } A \ x * f \ x)$ .

**show** *integrable*  $M$   $(\lambda x. f \ x *_{\mathbb{R}} \text{cond-exp } M \ F \ g \ x)$  **using**  $*$  *assms measurable-in-subalg*[*OF subalg*] **by** *blast*

```
{
  fix A assume asm: A ∈ F
  hence integrable M (λx. indicat-real A x *ℝ f x *ℝ g x) using subalg by
  (fastforce simp add: subalgebra-def intro!: integrable-mult-indicator assms(1))
  hence set-lebesgue-integral M A (λx. f x *ℝ g x) = set-lebesgue-integral M A
  (λx. f x *ℝ cond-exp M F g x) unfolding set-lebesgue-integral-def using asm by
  (auto intro!: * measurable-in-subalg[OF subalg])
}
thus AE x in M. cond-exp M F (λx. f x *ℝ g x) x = f x *ℝ cond-exp M F g x
using borel-measurable-cond-exp by (intro cond-exp-charact, auto intro!: * assms
measurable-in-subalg[OF subalg])
qed
```

**lemma** *cond-exp-sum* [*intro, simp*]:

```
fixes f :: 't ⇒ 'a ⇒ 'b :: {second-countable-topology,banach}
assumes [measurable]: ∧i. integrable M (f i)
shows AE x in M. cond-exp M F (λx. ∑ i∈I. f i x) x = (∑ i∈I. cond-exp M F
(f i) x)
proof (rule has-cond-exp-charact, intro has-cond-expI')
  fix A assume [measurable]: A ∈ sets F
  then have A-meas [measurable]: A ∈ sets M by (meson subsetD subalg subalge-
bra-def)
```

```
  have (∫ x∈A. (∑ i∈I. f i x) ∂M) = (∫ x. (∑ i∈I. indicator A x *ℝ f i x) ∂M)
  unfolding set-lebesgue-integral-def by (simp add: scaleR-sum-right)
  also have ... = (∑ i∈I. (∫ x. indicator A x *ℝ f i x ∂M)) using assms by (auto
intro!: Bochner-Integration.integral-sum integrable-mult-indicator)
  also have ... = (∑ i∈I. (∫ x. indicator A x *ℝ cond-exp M F (f i) x ∂M)) using
cond-exp-set-integral[OF assms] by (simp add: set-lebesgue-integral-def)
  also have ... = (∫ x. (∑ i∈I. indicator A x *ℝ cond-exp M F (f i) x) ∂M)
  using assms by (auto intro!: Bochner-Integration.integral-sum[symmetric] inte-
grable-mult-indicator)
  also have ... = (∫ x∈A. (∑ i∈I. cond-exp M F (f i) x) ∂M) unfolding set-lebesgue-integral-def
  by (simp add: scaleR-sum-right)
  finally show (∫ x∈A. (∑ i∈I. f i x) ∂M) = (∫ x∈A. (∑ i∈I. cond-exp M F (f i)
x) ∂M) by auto
qed (auto simp add: assms integrable-cond-exp)
```

### 4.3 Linearly Ordered Banach Spaces

In this subsection we show monotonicity results concerning the conditional expectation operator.

**lemma** *cond-exp-gr-c*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{second-countable-topology, banach, linorder-topology, ordered-real-vector\}$   
**assumes**  $integrable\ M\ f\ AE\ x\ in\ M. f\ x > c$   
**shows**  $AE\ x\ in\ M. cond-exp\ M\ F\ f\ x > c$   
**proof** –  
**define**  $X$  **where**  $X = \{x \in space\ M. cond-exp\ M\ F\ f\ x \leq c\}$   
**have**  $[measurable]: X \in sets\ F$  **unfolding**  $X-def$  **by**  $measurable\ (metis\ sets.top\ subalg\ subalgebra-def)$   
**hence**  $X-in-M: X \in sets\ M$  **using**  $sets-restr-to-subalg\ subalg\ subalgebra-def$  **by**  $blast$   
**have**  $emeasure\ M\ X = 0$   
**proof**  $(rule\ ccontr)$   
**assume**  $emeasure\ M\ X \neq 0$   
**have**  $emeasure\ (restr-to-subalg\ M\ F)\ X = emeasure\ M\ X$  **by**  $(simp\ add: emeasure-restr-to-subalg\ subalg)$   
**hence**  $emeasure\ (restr-to-subalg\ M\ F)\ X > 0$  **using**  $\neg(emeasure\ M\ X) = 0$  **by**  $gr-zeroI$  **by**  $auto$   
**then obtain**  $A$  **where**  $A: A \in sets\ (restr-to-subalg\ M\ F)\ A \subseteq X$   $emeasure\ (restr-to-subalg\ M\ F)\ A > 0$   $emeasure\ (restr-to-subalg\ M\ F)\ A < \infty$   
**using**  $sigma-fin-subalg$  **by**  $(metis\ emeasure-notin-sets\ ennreal-0\ infinity-ennreal-def\ le-less-linear\ neq-top-trans\ not-gr-zero\ order-refl\ sigma-finite-measure.approx-PInf-emeasure-with-finite)$   
**hence**  $[simp]: A \in sets\ F$  **using**  $subalg\ sets-restr-to-subalg$  **by**  $blast$   
**hence**  $A-in-sets-M[simp]: A \in sets\ M$  **using**  $sets-restr-to-subalg\ subalg\ subalgebra-def$  **by**  $blast$   
**have**  $[simp]: set-integrable\ M\ A\ (\lambda x. c)$  **using**  $A\ subalg$  **by**  $(auto\ simp\ add: set-integrable-def\ emeasure-restr-to-subalg)$   
**have**  $[simp]: set-integrable\ M\ A\ f$  **unfolding**  $set-integrable-def$  **by**  $(rule\ integrable-mult-indicator, auto\ simp\ add: assms(1))$   
**have**  $AE\ x\ in\ M. indicator\ A\ x\ *_R\ c = indicator\ A\ x\ *_R\ f\ x$   
**proof**  $(rule\ integral-eq-mono-AE-eq-AE)$   
**have**  $(\int x \in A. c\ \partial M) \leq (\int x \in A. f\ x\ \partial M)$  **using**  $assms(2)$  **by**  $(intro\ set-integral-mono-AE-banach)$   
**auto**  
**moreover**  
**{**  
**have**  $(\int x \in A. f\ x\ \partial M) = (\int x \in A. cond-exp\ M\ F\ f\ x\ \partial M)$  **by**  $(rule\ cond-exp-set-integral, auto\ simp\ add: assms)$   
**also have**  $\dots \leq (\int x \in A. c\ \partial M)$  **using**  $A$  **by**  $(auto\ intro!: set-integral-mono-banach\ simp\ add: X-def)$   
**finally have**  $(\int x \in A. f\ x\ \partial M) \leq (\int x \in A. c\ \partial M)$  **by**  $simp$   
**}**  
**ultimately show**  $LINT\ x|M. indicator\ A\ x\ *_R\ c = LINT\ x|M. indicator\ A\ x\ *_R\ f\ x$  **unfolding**  $set-lebesgue-integral-def$  **by**  $simp$   
**show**  $AE\ x\ in\ M. indicator\ A\ x\ *_R\ c \leq indicator\ A\ x\ *_R\ f\ x$  **using**  $assms$  **by**  $(auto\ simp\ add: X-def\ indicator-def)$   
**qed**  $(auto\ simp\ add: set-integrable-def[symmetric])$   
**hence**  $AE\ x \in A\ in\ M. c = f\ x$  **by**  $auto$   
**hence**  $AE\ x \in A\ in\ M. False$  **using**  $assms(2)$  **by**  $auto$   
**hence**  $A \in null-sets\ M$  **using**  $AE-iff-null-sets\ A-in-sets-M$  **by**  $metis$   
**thus**  $False$  **using**  $A(3)$  **by**  $(simp\ add: emeasure-restr-to-subalg\ null-setsD1)$



subalg)  
 qed  
 thus ?thesis using AE-iff-null-sets[OF X-in-M] unfolding X-def by auto  
 qed

**corollary** cond-exp-less-c:

fixes  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
 assumes integrable  $M f$  AE  $x$  in  $M$ .  $f x < c$   
 shows AE  $x$  in  $M$ . cond-exp  $M F f x < c$   
**proof** –  
 have AE  $x$  in  $M$ . cond-exp  $M F f x = - \text{cond-exp } M F (\lambda x. - f x) x$  using  
 cond-exp-uminus[OF assms(1)] by auto  
 moreover have AE  $x$  in  $M$ . cond-exp  $M F (\lambda x. - f x) x > - c$  using assms  
 by (intro cond-exp-gr-c) auto  
 ultimately show ?thesis by (force simp add: minus-less-iff)  
 qed

**lemma** cond-exp-mono-strict:

fixes  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
 assumes integrable  $M f$  integrable  $M g$  AE  $x$  in  $M$ .  $f x < g x$   
 shows AE  $x$  in  $M$ . cond-exp  $M F f x < \text{cond-exp } M F g x$   
 using cond-exp-less-c[OF Bochner-Integration.integrable-diff, OF assms(1,2), of  
 0]  
 cond-exp-diff[OF assms(1,2)] assms(3) by auto

**lemma** cond-exp-ge-c:

fixes  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
 assumes [measurable]: integrable  $M f$   
 and AE  $x$  in  $M$ .  $f x \geq c$   
 shows AE  $x$  in  $M$ . cond-exp  $M F f x \geq c$   
**proof** –  
 let  $?F = \text{restr-to-subalg } M F$   
 interpret sigma-finite-measure restr-to-subalg  $M F$  using sigma-fin-subalg by  
 auto  
 {  
 fix  $A$  assume asm:  $A \in \text{sets } ?F$   $0 < \text{measure } ?F A$   
 have [simp]:  $\text{sets } ?F = \text{sets } F$   $\text{measure } ?F A = \text{measure } M A$  using asm by (auto  
 simp add: measure-def sets-restr-to-subalg[OF subalg] emeasure-restr-to-subalg[OF  
 subalg])  
 have  $M-A$ :  $\text{emeasure } M A < \infty$  using measure-zero-top asm by (force simp  
 add: top.not-eq-extremum)  
 hence  $F-A$ :  $\text{emeasure } ?F A < \infty$  using asm(1) emeasure-restr-to-subalg subalg  
 by fastforce  
 have  $\text{set-lebesgue-integral } M A (\lambda \cdot. c) \leq \text{set-lebesgue-integral } M A f$  using  
 assms asm  $M-A$  subalg by (intro set-integral-mono-AE-banach, auto simp add:  
 set-integrable-def integrable-mult-indicator subalgebra-def sets-restr-to-subalg)

**also have** ... = *set-lebesgue-integral*  $M A$  (*cond-exp*  $M F f$ ) **using** *cond-exp-set-integral*[*OF* *assms*(1)] *asm* **by** *auto*  
**also have** ... = *set-lebesgue-integral*  $?F A$  (*cond-exp*  $M F f$ ) **unfolding** *set-lebesgue-integral-def*  
**using** *asm* *borel-measurable-cond-exp* **by** (*intro* *integral-subalgebra2*[*OF* *subalg*, *sym-metric*], *simp*)  
**finally have** ( $1 / \text{measure } ?F A$ )  $*_R \text{set-lebesgue-integral } ?F A$  (*cond-exp*  $M F f$ )  
 $\in \{c..\}$  **using** *asm* *subalg*  $M-A$  **by** (*auto* *simp* *add*: *set-integral-const subalgebra-def*  
*intro*!: *pos-divideR-le-eq*[*THEN* *iffD1*])  
**}**  
**thus** *?thesis* **using** *AE-restr-to-subalg*[*OF* *subalg*] *averaging-theorem*[*OF* *integrable-in-subalg* *closed-atLeast*, *OF* *subalg* *borel-measurable-cond-exp* *integrable-cond-exp*]  
**by** *auto*  
**qed**

**corollary** *cond-exp-le-c*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$   
**and** *AE*  $x$  *in*  $M$ .  $f x \leq c$   
**shows** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F f x \leq c$   
**proof** –  
**have** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F f x = - \text{cond-exp } M F (\lambda x. - f x) x$  **using**  
*cond-exp-uminus*[*OF* *assms*(1)] **by** *force*  
**moreover have** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F (\lambda x. - f x) x \geq - c$  **using** *assms*  
**by** (*intro* *cond-exp-ge-c*) *auto*  
**ultimately show** *?thesis* **by** (*force* *simp* *add*: *minus-le-iff*)  
**qed**

**corollary** *cond-exp-mono*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$  *AE*  $x$  *in*  $M$ .  $f x \leq g x$   
**shows** *AE*  $x$  *in*  $M$ . *cond-exp*  $M F f x \leq \text{cond-exp } M F g x$   
**using** *cond-exp-le-c*[*OF* *Bochner-Integration.integrable-diff*, *OF* *assms*(1,2), *of* 0]  
*cond-exp-diff*[*OF* *assms*(1,2)] *assms*(3) **by** *auto*

**corollary** *cond-exp-min*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$   
**assumes** *integrable*  $M f$  *integrable*  $M g$   
**shows** *AE*  $\xi$  *in*  $M$ . *cond-exp*  $M F (\lambda x. \min (f x) (g x)) \xi \leq \min (\text{cond-exp } M F f \xi) (\text{cond-exp } M F g \xi)$   
**proof** –  
**have** *AE*  $\xi$  *in*  $M$ . *cond-exp*  $M F (\lambda x. \min (f x) (g x)) \xi \leq \text{cond-exp } M F f \xi$  **by**  
(*intro* *cond-exp-mono* *integrable-min* *assms*, *simp*)  
**moreover have** *AE*  $\xi$  *in*  $M$ . *cond-exp*  $M F (\lambda x. \min (f x) (g x)) \xi \leq \text{cond-exp } M F g \xi$  **by** (*intro* *cond-exp-mono* *integrable-min* *assms*, *simp*)  
**ultimately show** *AE*  $\xi$  *in*  $M$ . *cond-exp*  $M F (\lambda x. \min (f x) (g x)) \xi \leq \min$

(*cond-exp*  $M F f \xi$ ) (*cond-exp*  $M F g \xi$ ) **by** *fastforce*  
**qed**

**corollary** *cond-exp-max*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector}\}$

**assumes** *integrable*  $M f$  *integrable*  $M g$

**shows**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \max (f x) (g x)) \xi \geq \max (\text{cond-exp } M F f \xi) (\text{cond-exp } M F g \xi)$

**proof** –

**have**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \max (f x) (g x)) \xi \geq \text{cond-exp } M F f \xi$  **by** (*intro cond-exp-mono integrable-max assms, simp*)

**moreover have**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \max (f x) (g x)) \xi \geq \text{cond-exp } M F g \xi$  **by** (*intro cond-exp-mono integrable-max assms, simp*)

**ultimately show**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \max (f x) (g x)) \xi \geq \max (\text{cond-exp } M F f \xi) (\text{cond-exp } M F g \xi)$  **by** *fastforce*

**qed**

**corollary** *cond-exp-inf*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector, lattice}\}$

**assumes** *integrable*  $M f$  *integrable*  $M g$

**shows**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \inf (f x) (g x)) \xi \leq \inf (\text{cond-exp } M F f \xi) (\text{cond-exp } M F g \xi)$

**unfolding** *inf-min* **using** *assms* **by** (*rule cond-exp-min*)

**corollary** *cond-exp-sup*:

**fixes**  $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, linorder-topology, ordered-real-vector, lattice}\}$

**assumes** *integrable*  $M f$  *integrable*  $M g$

**shows**  $AE \xi \text{ in } M. \text{cond-exp } M F (\lambda x. \sup (f x) (g x)) \xi \geq \sup (\text{cond-exp } M F f \xi) (\text{cond-exp } M F g \xi)$

**unfolding** *sup-max* **using** *assms* **by** (*rule cond-exp-max*)

**end**

## 4.4 Probability Spaces

**lemma** (*in prob-space*) *sigma-finite-subalgebra-restr-to-subalg*:

**assumes** *subalgebra*  $M F$

**shows** *sigma-finite-subalgebra*  $M F$

**proof** (*intro sigma-finite-subalgebra.intro*)

**interpret**  $F$ : *prob-space restr-to-subalg*  $M F$  **using** *assms* *prob-space-restr-to-subalg* *prob-space-axioms* **by** *blast*

**show** *sigma-finite-measure* (*restr-to-subalg*  $M F$ ) **by** (*rule F.sigma-finite-measure-axioms*)

**qed** (*rule assms*)

**lemma** (*in prob-space*) *cond-exp-trivial*:

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

```

assumes integrable  $M f$ 
shows  $AE\ x\ in\ M.\ cond\text{-}exp\ M\ (\sigma\ (space\ M)\ \{\})\ f\ x = expectation\ f$ 
proof –
  interpret sigma-finite-subalgebra  $M\ \sigma\ (space\ M)\ \{\}$  by (auto intro: sigma-finite-subalgebra-restr-to-subalg)
  simp add: subalgebra-def sigma-sets-empty-eq
  show ?thesis using assms by (intro cond-exp-charact) (auto simp add: sigma-sets-empty-eq)
set-lebesgue-integral-def prob-space cong: Bochner-Integration.integral-cong)
qed

```

The following lemma shows that independent  $\sigma$ -algebras don't matter for the conditional expectation.

```

lemma (in prob-space) cond-exp-indep-subalgebra:
  fixes  $f :: 'a \Rightarrow 'b :: \{second-countable-topology, banach, real-normed-field\}$ 
  assumes subalgebra: subalgebra  $M\ F\ subalgebra\ M\ G$ 
  and independent: indep-set  $G\ (\sigma\ (space\ M)\ (F \cup \text{vimage-algebra}\ (space\ M)\ f\ borel))$ 
  assumes [measurable]: integrable  $M f$ 
  shows  $AE\ x\ in\ M.\ cond\text{-}exp\ M\ (\sigma\ (space\ M)\ (F \cup G))\ f\ x = cond\text{-}exp\ M\ F\ f\ x$ 
proof –
  interpret Un-sigma: sigma-finite-subalgebra  $M\ \sigma\ (space\ M)\ (F \cup G)$  using
assms(1,2) by (auto intro!: sigma-finite-subalgebra-restr-to-subalg sets.sigma-sets-subset)
  simp add: subalgebra-def space-measure-of-conv sets-measure-of-conv)
  interpret sigma-finite-subalgebra  $M\ F$  using assms by (auto intro: sigma-finite-subalgebra-restr-to-subalg)
  {
    fix  $A$ 
    assume asm:  $A \in \sigma\ (space\ M)\ \{a \cap b \mid a \in F \wedge b \in G\}$ 
    have in-events: sigma-sets  $(space\ M)\ \{a \cap b \mid a \in F \wedge b \in G\} \subseteq events$  using subalgebra by (intro sets.sigma-sets-subset, auto simp add: subalgebra-def)
    have Int-stable  $\{a \cap b \mid a \in F \wedge b \in G\}$ 
    proof –
      {
        fix  $af\ bf\ ag\ bg$ 
        assume  $F$ :  $af \in F\ bf \in F$  and  $G$ :  $ag \in G\ bg \in G$ 
        have  $af \cap bf \in F$  by (intro sets.Int F)
        moreover have  $ag \cap bg \in G$  by (intro sets.Int G)
        ultimately have  $\exists a\ b.\ af \cap ag \cap (bf \cap bg) = a \cap b \wedge a \in F \wedge b \in G$  by (metis inf-assoc inf-left-commute)
      }
    thus ?thesis by (force intro!: Int-stableI)
  }
qed
  moreover have  $\{a \cap b \mid a \in F \wedge b \in G\} \subseteq Pow\ (space\ M)$  using
subalgebra by (force simp add: subalgebra-def dest: sets.sets-into-space)
  moreover have  $A \in \sigma\ (space\ M)\ \{a \cap b \mid a \in F \wedge b \in G\}$  using
calculation asm by force
  ultimately have set-lebesgue-integral  $M\ A\ f = set\text{-}lebesgue\text{-}integral\ M\ A\ (cond\text{-}exp\ M\ F\ f)$ 
  proof (induction rule: sigma-sets-induct-disjoint)

```

**case** (*basic A*)  
**then obtain**  $a \ b$  **where**  $A: A = a \cap b \ a \in F \ b \in G$  **by** *blast*

**hence**  $\text{events}[\text{measurable}]: a \in \text{events} \ b \in \text{events}$  **using** *subalgebra* **by** (*auto simp add: subalgebra-def*)

**have**  $[\text{simp}]: \text{sigma-sets}(\text{space } M) \{ \text{indicator } b - 'A \cap \text{space } M \mid A. A \in \text{borel} \}$   
 $\subseteq G$   
**using** *borel-measurable-indicator[OF A(3), THEN measurable-sets] sets.top subalgebra*  
**by** (*intro sets.sigma-sets-subset'*) (*fastforce simp add: subalgebra-def*)+

**have**  $\text{Un-in-sigma}: F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel} \subseteq \text{sigma}(\text{space } M) (F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel})$  **by** (*metis equalityE le-supI sets.space-closed sigma-le-sets space-vimage-algebra subalg subalgebra-def*)

**have**  $[\text{intro}]: \text{indep-var borel}(\text{indicator } b) \text{ borel}(\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} f \ \omega)$   
**proof** –  
**have**  $[\text{simp}]: \text{sigma-sets}(\text{space } M) \{ (\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} f \ \omega) - 'A \cap \text{space } M \mid A. A \in \text{borel} \} \subseteq \text{sigma}(\text{space } M) (F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel})$   
**proof** –  
**have**  $*$ :  $(\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} f \ \omega) \in \text{borel-measurable}(\text{sigma}(\text{space } M) (F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel}))$   
**using** *borel-measurable-indicator[OF A(2), THEN measurable-sets, OF borel-open] subalgebra*  
**by** (*intro borel-measurable-scaleR borel-measurableI Un-in-sigma[THEN subsetD]*)  
*(auto simp add: space-measure-of-conv subalgebra-def sets-vimage-algebra2)*  
**thus**  $?thesis$  **using** *measurable-sets[OF \*]* **by** (*intro sets.sigma-sets-subset'*, *auto simp add: space-measure-of-conv*)  
**qed**

**have**  $\text{indep-set}(\text{sigma-sets}(\text{space } M) \{ \text{indicator } b - 'A \cap \text{space } M \mid A. A \in \text{borel} \}) (\text{sigma-sets}(\text{space } M) \{ (\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} f \ \omega) - 'A \cap \text{space } M \mid A. A \in \text{borel} \})$   
**using** *independent unfolding indep-set-def* **by** (*rule indep-sets-mono-sets, auto split: bool.split*)  
**thus**  $?thesis$  **by** (*subst indep-var-eq, auto intro!: borel-measurable-scaleR*)  
**qed**

**have**  $[\text{intro}]: \text{indep-var borel}(\text{indicator } b) \text{ borel}(\lambda \omega. \text{indicat-real } a \ \omega *_{\mathbb{R}} \text{cond-exp } M \ F \ f \ \omega)$   
**proof** –  
**have**  $[\text{simp}]: \text{sigma-sets}(\text{space } M) \{ (\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} \text{cond-exp } M \ F \ f \ \omega) - 'A \cap \text{space } M \mid A. A \in \text{borel} \} \subseteq \text{sigma}(\text{space } M) (F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel})$   
**proof** –  
**have**  $*$ :  $(\lambda \omega. \text{indicator } a \ \omega *_{\mathbb{R}} \text{cond-exp } M \ F \ f \ \omega) \in \text{borel-measurable}(\text{sigma}(\text{space } M) (F \cup \text{vimage-algebra}(\text{space } M) f \text{ borel}))$   
**using** *borel-measurable-indicator[OF A(2), THEN measurable-sets, OF*

```

borel-open] subalgebra
      borel-measurable-cond-exp[THEN measurable-sets, OF borel-open, of
- M F f]
      by (intro borel-measurable-scaleR borel-measurableI Un-in-sigma[THEN
subsetD])
          (auto simp add: space-measure-of-conv subalgebra-def)
      thus ?thesis using measurable-sets[OF *] by (intro sets.sigma-sets-subset',
auto simp add: space-measure-of-conv)
    qed
    have indep-set (sigma-sets (space M) {indicator b - 'A ∩ space M | A. A ∈
borel}) (sigma-sets (space M) {(λω. indicator a ω *R cond-exp M F f ω) - 'A ∩
space M | A. A ∈ borel})
      using independent unfolding indep-set-def by (rule indep-sets-mono-sets,
auto split: bool.split)
    thus ?thesis by (subst indep-var-eq, auto intro!: borel-measurable-scaleR)
  qed

  have set-lebesgue-integral M A f = (LINT x|M. indicator b x * (indicator a
x *R f x))
    unfolding set-lebesgue-integral-def A indicator-inter-arith
    by (intro Bochner-Integration.integral-cong, auto simp add: scaleR-scaleR[symmetric]
indicator-times-eq-if(1))
    also have ... = (LINT x|M. indicator b x) * (LINT x|M. indicator a x *R f
x)
      by (intro indep-var-lebesgue-integral
Bochner-Integration.integrable-bound[OF integrable-const[of 1 :: 'b]
borel-measurable-indicator]
integrable-mult-indicator[OF - assms(4)], blast) (auto simp add:
indicator-def)
    also have ... = (LINT x|M. indicator b x) * (LINT x|M. indicator a x *R
cond-exp M F f x)
      using cond-exp-set-integral[OF assms(4) A(2)] unfolding set-lebesgue-integral-def
by argo
    also have ... = (LINT x|M. indicator b x * (indicator a x *R cond-exp M
F f x))
      by (intro indep-var-lebesgue-integral[symmetric]
Bochner-Integration.integrable-bound[OF integrable-const[of 1 :: 'b]
borel-measurable-indicator]
integrable-mult-indicator[OF - integrable-cond-exp], blast) (auto simp
add: indicator-def)
    also have ... = set-lebesgue-integral M A (cond-exp M F f)
      unfolding set-lebesgue-integral-def A indicator-inter-arith
    by (intro Bochner-Integration.integral-cong, auto simp add: scaleR-scaleR[symmetric]
indicator-times-eq-if(1))
    finally show ?case .
  next
  case empty
  then show ?case unfolding set-lebesgue-integral-def by simp
next

```

```

    case (compl A)
    have A-in-space:  $A \subseteq \text{space } M$  using compl using in-events sets.sets-into-space
  by blast
    have set-lebesgue-integral  $M$  ( $\text{space } M - A$ )  $f = \text{set-lebesgue-integral } M$  ( $\text{space } M - A \cup A$ )  $f - \text{set-lebesgue-integral } M$   $A$   $f$ 
    using compl(1) in-events
    by (subst set-integral-Un[of space  $M - A$   $A$ ], blast)
      (simp | intro integrable-mult-indicator[folded set-integrable-def, OF -
assms(4)], fast)+
    also have ... = set-lebesgue-integral  $M$  ( $\text{space } M - A \cup A$ ) ( $\text{cond-exp } M$   $F$   $f$ )
    - set-lebesgue-integral  $M$   $A$  ( $\text{cond-exp } M$   $F$   $f$ )
    using cond-exp-set-integral[OF assms(4) sets.top] compl subalgebra by (simp
add: subalgebra-def Un-absorb2[OF A-in-space])
    also have ... = set-lebesgue-integral  $M$  ( $\text{space } M - A$ ) ( $\text{cond-exp } M$   $F$   $f$ )
    using compl(1) in-events
    by (subst set-integral-Un[of space  $M - A$   $A$ ], blast)
      (simp | intro integrable-mult-indicator[folded set-integrable-def, OF -
integrable-cond-exp], fast)+
    finally show ?case .
  next
    case (union A)
    have set-lebesgue-integral  $M$  ( $\bigcup (\text{range } A)$ )  $f = (\sum i. \text{set-lebesgue-integral } M$ 
( $A$   $i$ )  $f$ )
    using union in-events
    by (intro lebesgue-integral-countable-add) (auto simp add: disjoint-family-onD
intro!: integrable-mult-indicator[folded set-integrable-def, OF - assms(4)])
    also have ... =  $(\sum i. \text{set-lebesgue-integral } M$  ( $A$   $i$ ) ( $\text{cond-exp } M$   $F$   $f$ )) using
union by presburger
    also have ... = set-lebesgue-integral  $M$  ( $\bigcup (\text{range } A)$ ) ( $\text{cond-exp } M$   $F$   $f$ )
    using union in-events
    by (intro lebesgue-integral-countable-add[symmetric]) (auto simp add: dis-
joint-family-onD intro!: integrable-mult-indicator[folded set-integrable-def, OF - in-
tegrable-cond-exp])
    finally show ?case .
  qed
}
moreover have sigma (space  $M$ )  $\{a \cap b \mid a \text{ b. } a \in F \wedge b \in G\} = \text{sigma (space } M) (F \cup G)$ 
proof -
  have sigma-sets (space  $M$ )  $\{a \cap b \mid a \text{ b. } a \in \text{sets } F \wedge b \in \text{sets } G\} = \text{sigma-sets (space } M) (\text{sets } F \cup \text{sets } G)$ 
  proof -
    {
      fix a b assume asm:  $a \in F$   $b \in G$ 
      hence  $a \cap b \in \text{sigma-sets (space } M) (F \cup G)$  using subalgebra unfolding
Int-range-binary by (intro sigma-sets-Inter[OF - binary-in-sigma-sets]) (force simp
add: subalgebra-def dest: sets.sets-into-space)+
    }
  moreover

```

```

{
  fix a
  assume a ∈ sets F
  hence a ∈ sigma-sets (space M) {a ∩ b | a b. a ∈ sets F ∧ b ∈ sets G}
    using subalgebra sets.top[of G] sets.sets-into-space[of - F]
    by (intro sigma-sets.Basic, auto simp add: subalgebra-def)
}
moreover
{
  fix a assume a ∈ sets F ∨ a ∈ sets G a ∉ sets F
  hence a ∈ sets G by blast
  hence a ∈ sigma-sets (space M) {a ∩ b | a b. a ∈ sets F ∧ b ∈ sets G}
    using subalgebra sets.top[of F] sets.sets-into-space[of - G]
    by (intro sigma-sets.Basic, auto simp add: subalgebra-def)
}
ultimately show ?thesis by (intro sigma-sets.eqI) auto
qed
thus ?thesis using subalgebra by (intro sigma-eqI) (force simp add: subalgebra-def dest: sets.sets-into-space)+
qed
moreover have (cond-exp M F f) ∈ borel-measurable (sigma (space M) (sets F ∪ sets G))
proof -
  have F ⊆ sigma (space M) (F ∪ G) by (metis Un-least Un-upper1 measure-of-of-measure sets.space-closed sets-measure-of sigma-sets-subseteq subalg subalgebra(2) subalgebra-def)
  thus ?thesis using borel-measurable-cond-exp[THEN measurable-sets, OF borel-open, of - M F f] subalgebra by (intro borel-measurableI, force simp only: space-measure-of-conv subalgebra-def)
qed
ultimately show ?thesis using assms(4) integrable-cond-exp by (intro Un-sigma.cond-exp-charact) presburger+
qed

```

If a random variable is independent of a  $\sigma$ -algebra  $F$ , its conditional expectation  $\text{cond-exp } M F f$  is just its expectation.

**lemma** (in prob-space) *cond-exp-indep*:

```

fixes f :: 'a ⇒ 'b :: {second-countable-topology, banach, real-normed-field}
assumes subalgebra: subalgebra M F
      and independent: indep-set F (vimage-algebra (space M) f borel)
      and integrable: integrable M f
shows AE x in M. cond-exp M F f x = expectation f
proof -
  have indep-set F (sigma (space M) (sigma (space M) {} ∪ (vimage-algebra (space M) f borel)))
    using independent unfolding indep-set-def
  by (rule indep-sets-mono-sets, simp add: bool.split)
  (metis bot.extremum dual-order.refl sets.sets-measure-of-eq sets.sigma-sets-subset'
  sets-vimage-algebra-space space-vimage-algebra sup.absorb-iff2)

```



```

hence cond-exp-indep:  $AE\ x\ in\ M.\ cond-exp\ M\ (\sigma\ (space\ M)\ (\sigma\ (space\ M)\ \{\} \cup F))\ f\ x = expectation\ f$ 
using cond-exp-indep-subalgebra[OF - subalgebra - integrable, of sigma (space M)  $\{\}$ ] cond-exp-trivial[OF integrable]
by (auto simp add: subalgebra-def sigma-sets-empty-eq)
have sets ( $\sigma\ (space\ M)\ (\sigma\ (space\ M)\ \{\} \cup F)$ ) = F
using subalgebra sets.top[of F] unfolding subalgebra-def
by (simp add: sigma-sets-empty-eq, subst insert-absorb[of space M F], blast)
(metis insert-absorb[OF sets.empty-sets] sets.sets-measure-of-eq)
hence  $AE\ x\ in\ M.\ cond-exp\ M\ (\sigma\ (space\ M)\ (\sigma\ (space\ M)\ \{\} \cup F))\ f\ x = cond-exp\ M\ F\ f\ x$  by (rule cond-exp-sets-cong)
thus ?thesis using cond-exp-indep by force
qed

end

```

```

theory Filtered-Measure
imports HOL-Probability.Conditional-Expectation
begin

```

## 5 Filtered Measure Spaces

### 5.1 Filtered Measure

```

locale filtered-measure =
  fixes M F and  $t_0 :: 'b :: \{second-countable-topology, order-topology, t2-space\}$ 
  assumes subalgebras:  $\bigwedge i.\ t_0 \leq i \implies subalgebra\ M\ (F\ i)$ 
  and sets-F-mono:  $\bigwedge i\ j.\ t_0 \leq i \implies i \leq j \implies sets\ (F\ i) \subseteq sets\ (F\ j)$ 
begin

```

```

lemma space-F[simp]:
  assumes  $t_0 \leq i$ 
  shows  $space\ (F\ i) = space\ M$ 
  using subalgebras assms by (simp add: subalgebra-def)

```

```

lemma subalgebra-F[intro]:
  assumes  $t_0 \leq i\ i \leq j$ 
  shows  $subalgebra\ (F\ j)\ (F\ i)$ 
  unfolding subalgebra-def using assms by (simp add: sets-F-mono)

```

```

lemma borel-measurable-mono:
  assumes  $t_0 \leq i\ i \leq j$ 
  shows  $borel-measurable\ (F\ i) \subseteq borel-measurable\ (F\ j)$ 
  unfolding subset-iff by (metis assms subalgebra-F measurable-from-subalg)

```

**end**

```

locale linearly-filtered-measure = filtered-measure M F t_0 for M and F :: - ::

```

$\{\text{linorder-topology}\} \Rightarrow -$  **and**  $t_0$

**locale**  $\text{nat-filtered-measure} = \text{linearly-filtered-measure } M \ F \ 0$  **for**  $M$  **and**  $F :: \text{nat}$   
 $\Rightarrow -$   
**locale**  $\text{real-filtered-measure} = \text{linearly-filtered-measure } M \ F \ 0$  **for**  $M$  **and**  $F :: \text{real}$   
 $\Rightarrow -$

## 5.2 $\sigma$ -Finite Filtered Measure

The locale presented here is a generalization of the *sigma-finite-subalgebra* for a particular filtration.

**locale**  $\text{sigma-finite-filtered-measure} = \text{filtered-measure} +$   
**assumes**  $\text{sigma-finite-initial: sigma-finite-subalgebra } M \ (F \ t_0)$

**lemma** (in  $\text{sigma-finite-filtered-measure}$ )  $\text{sigma-finite-subalgebra-}F[\text{intro}]$ :  
**assumes**  $t_0 \leq i$   
**shows**  $\text{sigma-finite-subalgebra } M \ (F \ i)$   
**using**  $\text{assms by (metis dual-order.refl sets-F-mono sigma-finite-initial sigma-finite-subalgebra.nested-subalg-is-subalgebras subalgebra-def)}$

**locale**  $\text{nat-sigma-finite-filtered-measure} = \text{sigma-finite-filtered-measure } M \ F \ 0 ::$   
 $\text{nat for } M \ F$   
**locale**  $\text{real-sigma-finite-filtered-measure} = \text{sigma-finite-filtered-measure } M \ F \ 0 ::$   
 $\text{real for } M \ F$

**sublocale**  $\text{nat-sigma-finite-filtered-measure} \subseteq \text{sigma-finite-subalgebra } M \ F \ i$  **by**  
 $\text{blast}$   
**sublocale**  $\text{real-sigma-finite-filtered-measure} \subseteq \text{sigma-finite-subalgebra } M \ F \ |i|$  **by**  
 $\text{fastforce}$

## 5.3 Finite Filtered Measure

**locale**  $\text{finite-filtered-measure} = \text{filtered-measure} + \text{finite-measure}$

**sublocale**  $\text{finite-filtered-measure} \subseteq \text{sigma-finite-filtered-measure}$   
**using**  $\text{subalgebras by (unfold-locales, blast, meson dual-order.refl finite-measure-axioms finite-measure-def finite-measure-restr-to-subalg sigma-finite-measure.sigma-finite-countable)}$

**locale**  $\text{nat-finite-filtered-measure} = \text{finite-filtered-measure } M \ F \ 0 :: \text{nat for } M \ F$   
**locale**  $\text{real-finite-filtered-measure} = \text{finite-filtered-measure } M \ F \ 0 :: \text{real for } M \ F$

**sublocale**  $\text{nat-finite-filtered-measure} \subseteq \text{nat-sigma-finite-filtered-measure} ..$   
**sublocale**  $\text{real-finite-filtered-measure} \subseteq \text{real-sigma-finite-filtered-measure} ..$

## 5.4 Constant Filtration

**lemma**  $\text{filtered-measure-constant-filtration}$ :  
**assumes**  $\text{subalgebra } M \ F$   
**shows**  $\text{filtered-measure } M \ (\lambda \cdot. F) \ t_0$

```

using assms by (unfold-locales) blast+

sublocale sigma-finite-subalgebra  $\subseteq$  constant-filtration: sigma-finite-filtered-measure
M  $\lambda$ - :: 't :: {second-countable-topology, linorder-topology}. F t0
  using subalg by (unfold-locales) blast+

lemma (in finite-measure) filtered-measure-constant-filtration:
  assumes subalgebra M F
  shows finite-filtered-measure M ( $\lambda$ -. F) t0
  using assms by (unfold-locales) blast+

end

theory Stochastic-Process
imports Filtered-Measure Measure-Space-Supplement HOL-Probability.Independent-Family
begin

```

## 6 Stochastic Processes

### 6.1 Stochastic Process

A stochastic process is a collection of random variables, indexed by a type '*b*.

```

locale stochastic-process =
  fixes M t0 and X :: 'b :: {second-countable-topology, order-topology, t2-space}  $\Rightarrow$ 
  'a  $\Rightarrow$  'c :: {second-countable-topology, banach}
  assumes random-variable[measurable]:  $\bigwedge i. t_0 \leq i \implies X\ i \in \text{borel-measurable } M$ 
begin

```

**definition** *left-continuous* **where** *left-continuous* = (*AE*  $\xi$  *in* *M*.  $\forall t. \text{continuous}$  (*at-left* *t*) ( $\lambda i. X\ i\ \xi$ ))

**definition** *right-continuous* **where** *right-continuous* = (*AE*  $\xi$  *in* *M*.  $\forall t. \text{continuous}$  (*at-right* *t*) ( $\lambda i. X\ i\ \xi$ ))

**end**

We specify the following locales to formalize discrete time and continuous time processes.

```

locale nat-stochastic-process = stochastic-process M 0 :: nat X for M X
locale real-stochastic-process = stochastic-process M 0 :: real X for M X

```

```

lemma stochastic-process-const-fun:
  assumes f  $\in \text{borel-measurable } M$ 
  shows stochastic-process M t0 ( $\lambda$ -. f) using assms by (unfold-locales)

```

```

lemma stochastic-process-const:
  shows stochastic-process M t0 ( $\lambda i$  -. c i) by (unfold-locales) simp

```

In the following segment, we cover basic operations on stochastic processes.

**context** *stochastic-process*

**begin**

**lemma** *compose-stochastic*:

**assumes**  $\bigwedge i. t_0 \leq i \implies f\ i \in \text{borel-measurable borel}$

**shows** *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. (f\ i)\ (X\ i\ \xi))$

**by** (*unfold-locales*) (*intro measurable-compose* [*OF random-variable assms*])

**lemma** *norm-stochastic*: *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. \text{norm}\ (X\ i\ \xi))$  **by** (*fastforce intro: compose-stochastic*)

**lemma** *scaleR-right-stochastic*:

**assumes** *stochastic-process*  $M\ t_0\ Y$

**shows** *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. (Y\ i\ \xi) *_{\mathbb{R}} (X\ i\ \xi))$

**using** *stochastic-process.random-variable* [*OF assms*] *random-variable* **by** (*unfold-locales simp*)

**lemma** *scaleR-right-const-fun-stochastic*:

**assumes**  $f \in \text{borel-measurable } M$

**shows** *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. f\ \xi *_{\mathbb{R}} (X\ i\ \xi))$

**by** (*unfold-locales*) (*intro borel-measurable-scaleR assms random-variable*)

**lemma** *scaleR-right-const-stochastic*: *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. c\ i *_{\mathbb{R}} (X\ i\ \xi))$

**by** (*unfold-locales simp*)

**lemma** *add-stochastic*:

**assumes** *stochastic-process*  $M\ t_0\ Y$

**shows** *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. X\ i\ \xi + Y\ i\ \xi)$

**using** *stochastic-process.random-variable* [*OF assms*] *random-variable* **by** (*unfold-locales simp*)

**lemma** *diff-stochastic*:

**assumes** *stochastic-process*  $M\ t_0\ Y$

**shows** *stochastic-process*  $M\ t_0\ (\lambda i\ \xi. X\ i\ \xi - Y\ i\ \xi)$

**using** *stochastic-process.random-variable* [*OF assms*] *random-variable* **by** (*unfold-locales simp*)

**lemma** *uminus-stochastic*: *stochastic-process*  $M\ t_0\ (-X)$  **using** *scaleR-right-const-stochastic* [*of*  $\lambda -. -1$ ] **by** (*simp add: fun-Compl-def*)

**lemma** *partial-sum-stochastic*: *stochastic-process*  $M\ t_0\ (\lambda n\ \xi. \sum_{i \in \{t_0..n\}} X\ i\ \xi)$

**by** (*unfold-locales simp*)

**lemma** *partial-sum'-stochastic*: *stochastic-process*  $M\ t_0\ (\lambda n\ \xi. \sum_{i \in \{t_0..<n\}} X\ i\ \xi)$  **by** (*unfold-locales simp*)

**end**

**lemma** *stochastic-process-sum*:

**assumes**  $\bigwedge i. i \in I \implies \text{stochastic-process } M \ t_0 \ (X \ i)$   
**shows**  $\text{stochastic-process } M \ t_0 \ (\lambda k \ \xi. \sum i \in I. X \ i \ k \ \xi)$  **using** *assms*[*THEN* *stochastic-process.random-variable*] **by** (*unfold-locales*, *auto*)

### 6.1.1 Natural Filtration

The natural filtration induced by a stochastic process  $X$  is the filtration generated by all events involving the process up to the time index  $t$ , i.e.  $F_t = \sigma(\{X \ s \mid s. s \leq t\})$ .

**definition** *natural-filtration* :: 'a measure  $\Rightarrow$  'b  $\Rightarrow$  ('b  $\Rightarrow$  'a  $\Rightarrow$  'c :: topological-space)  $\Rightarrow$  'b :: {second-countable-topology, order-topology}  $\Rightarrow$  'a measure **where**  
*natural-filtration*  $M \ t_0 \ Y = (\lambda t. \text{family-vimage-algebra } (\text{space } M) \ \{Y \ i \mid i. i \in \{t_0..t\}\} \text{ borel})$

**abbreviation** *nat-natural-filtration*  $\equiv \lambda M. \text{natural-filtration } M \ (0 :: \text{nat})$

**abbreviation** *real-natural-filtration*  $\equiv \lambda M. \text{natural-filtration } M \ (0 :: \text{real})$

**lemma** *space-natural-filtration[simp]*:  $\text{space } (\text{natural-filtration } M \ t_0 \ X \ t) = \text{space } M$  **unfolding** *natural-filtration-def* *space-family-vimage-algebra* ..

**lemma** *sets-natural-filtration*:  $\text{sets } (\text{natural-filtration } M \ t_0 \ X \ t) = \text{sigma-sets } (\text{space } M) \ (\bigcup i \in \{t_0..t\}. \{X \ i - 'A \cap \text{space } M \mid A. A \in \text{borel}\})$

**unfolding** *natural-filtration-def* *sets-family-vimage-algebra* **by** (*intro sigma-sets-eqI*) *blast+*

**lemma** *sets-natural-filtration'*:

**assumes** *borel* = *sigma UNIV S*

**shows**  $\text{sets } (\text{natural-filtration } M \ t_0 \ X \ t) = \text{sigma-sets } (\text{space } M) \ (\bigcup i \in \{t_0..t\}. \{X \ i - 'A \cap \text{space } M \mid A. A \in S\})$

**proof** (*subst sets-natural-filtration*, *intro sigma-sets-eqI*, *clarify*)

**fix**  $i$  **and**  $A :: 'a \text{ set}$  **assume** *asm*:  $i \in \{t_0..t\} \ A \in \text{sets borel}$

**hence**  $A \in \text{sigma-sets UNIV } S$  **unfolding** *assms* **by** *simp*

**thus**  $X \ i - 'A \cap \text{space } M \in \text{sigma-sets } (\text{space } M) \ (\bigcup i \in \{t_0..t\}. \{X \ i - 'A \cap \text{space } M \mid A. A \in S\})$

**proof** (*induction*)

**case** (*Compl a*)

**have**  $X \ i - '(UNIV - a) \cap \text{space } M = \text{space } M - (X \ i - 'a \cap \text{space } M)$  **by** *blast*

**then show** *?case* **using** *Compl(2)*[*THEN sigma-sets.Compl*] **by** *presburger*

**next**

**case** (*Union a*)

**have**  $X \ i - '\bigcup (\text{range } a) \cap \text{space } M = \bigcup (\text{range } (\lambda j. X \ i - 'a \ j \cap \text{space } M))$

**by** *blast*

**then show** *?case* **using** *Union(2)*[*THEN sigma-sets.Union*] **by** *presburger*

**qed** (*auto intro: asm sigma-sets.Empty*)

**qed** (*intro sigma-sets.Basic*, *force simp add: assms*)

**lemma** *sets-natural-filtration-open*:

*sets* (natural-filtration  $M \ t_0 \ X \ t$ ) = *sigma-sets* (space  $M$ ) ( $\bigcup_{i \in \{t_0..t\}} \{X \ i - 'A \cap \text{space } M \mid A. \text{open } A\}$ )

**using** *sets-natural-filtration'* **by** (force simp only: borel-def mem-Collect-eq)

**lemma** *sets-natural-filtration-oi*:

*sets* (natural-filtration  $M \ t_0 \ X \ t$ ) = *sigma-sets* (space  $M$ ) ( $\bigcup_{i \in \{t_0..t\}} \{X \ i - 'A \cap \text{space } M \mid A :: - :: \{\text{linorder-topology, second-countable-topology}\} \text{ set. } A \in \text{range greaterThan}\}$ )

**by** (rule *sets-natural-filtration'*[OF borel-Ioi])

**lemma** *sets-natural-filtration-io*:

*sets* (natural-filtration  $M \ t_0 \ X \ t$ ) = *sigma-sets* (space  $M$ ) ( $\bigcup_{i \in \{t_0..t\}} \{X \ i - 'A \cap \text{space } M \mid A :: - :: \{\text{linorder-topology, second-countable-topology}\} \text{ set. } A \in \text{range lessThan}\}$ )

**by** (rule *sets-natural-filtration'*[OF borel-Iio])

**lemma** *sets-natural-filtration-ci*:

*sets* (natural-filtration  $M \ t_0 \ X \ t$ ) = *sigma-sets* (space  $M$ ) ( $\bigcup_{i \in \{t_0..t\}} \{X \ i - 'A \cap \text{space } M \mid A :: \text{real set. } A \in \text{range atLeast}\}$ )

**by** (rule *sets-natural-filtration'*[OF borel-Ici])

**context** *stochastic-process*

**begin**

**lemma** *subalgebra-natural-filtration*:

**shows** *subalgebra*  $M$  (natural-filtration  $M \ t_0 \ X \ i$ )

**unfolding** *subalgebra-def* **using** *measurable-family-iff-sets* **by** (force simp add: *natural-filtration-def*)

**lemma** *filtered-measure-natural-filtration*:

**shows** *filtered-measure*  $M$  (natural-filtration  $M \ t_0 \ X$ )  $t_0$

**by** (*unfold-locales*) (*intro subalgebra-natural-filtration, simp only: sets-natural-filtration, intro sigma-sets-subseteq, force*)

In order to show that the natural filtration constitutes a filtered  $\sigma$ -finite measure, we need to provide a countable exhausting set in the preimage of  $X \ t_0$ .

**lemma** *sigma-finite-filtered-measure-natural-filtration*:

**assumes** *exhausting-set*: countable  $A$  ( $\bigcup A$ ) = space  $M \ \bigwedge a. a \in A \implies \text{emeasure } M \ a \neq \infty \ \bigwedge a. a \in A \implies \exists b \in \text{borel. } a = X \ t_0 - 'b \cap \text{space } M$

**shows** *sigma-finite-filtered-measure*  $M$  (natural-filtration  $M \ t_0 \ X$ )  $t_0$

**proof** (*unfold-locales*)

**have**  $A \subseteq \text{sets} (\text{restr-to-subalg } M \ (\text{natural-filtration } M \ t_0 \ X \ t_0))$  **using** *exhausting-set* **by** (*simp add: sets-restr-to-subalg*[OF *subalgebra-natural-filtration*] *sets-natural-filtration*) **fast**

**moreover have**  $\bigcup A = \text{space} (\text{restr-to-subalg } M \ (\text{natural-filtration } M \ t_0 \ X \ t_0))$

**unfolding** *space-restr-to-subalg* **using** *exhausting-set* **by** *simp*

**moreover have**  $\forall a \in A. \text{emeasure} (\text{restr-to-subalg } M \ (\text{natural-filtration } M \ t_0 \ X \ t_0)) \ a \neq \infty$  **using** *calculation*(1) *exhausting-set*(3)

**by** (*auto simp add: sets-restr-to-subalg[OF subalgebra-natural-filtration] emea-*  
*sure-restr-to-subalg[OF subalgebra-natural-filtration]*)  
**ultimately show**  $\exists A. \text{countable } A \wedge A \subseteq \text{sets } (\text{restr-to-subalg } M \text{ (natural-filtration } M \ t_0 \ X \ t_0)) \wedge \bigcup A = \text{space } (\text{restr-to-subalg } M \text{ (natural-filtration } M \ t_0 \ X \ t_0)) \wedge$   
 $(\forall a \in A. \text{emeasure } (\text{restr-to-subalg } M \text{ (natural-filtration } M \ t_0 \ X \ t_0)) \ a \neq \infty)$  **using**  
*exhausting-set* **by** *blast*  
**show**  $\bigwedge i \ j. \llbracket t_0 \leq i; i \leq j \rrbracket \implies \text{sets } (\text{natural-filtration } M \ t_0 \ X \ i) \subseteq \text{sets } (\text{natural-filtration } M \ t_0 \ X \ j)$  **using** *filtered-measure.subalgebra-F[OF filtered-measure-natural-filtration]*  
**by** (*simp add: subalgebra-def*)  
**qed** (*auto intro: subalgebra-natural-filtration*)

**lemma** *finite-filtered-measure-natural-filtration:*

**assumes** *finite-measure M*  
**shows** *finite-filtered-measure M (natural-filtration M t<sub>0</sub> X) t<sub>0</sub>*  
**using** *finite-measure.axioms[OF assms] filtered-measure-natural-filtration* **by** *intro-locales*

**end**

Filtration generated by independent variables.

**lemma** (*in prob-space*) *indep-set-natural-filtration:*

**assumes**  $t_0 \leq s \leq t$  *indep-vars*  $(\lambda-. \text{borel}) \ X \ \{t_0..s\}$   
**shows** *indep-set (natural-filtration M t<sub>0</sub> X s) (vimage-algebra (space M) (X t) borel)*  
**proof** –  
**have** *indep-sets*  $(\lambda i. \{X \ i - 'A \cap \text{space } M \mid A. A \in \text{sets borel}\}) (\bigcup (\text{range } (\text{case-bool } \{t_0..s\} \ \{t\})))$   
**using** *assms*  
**by** (*intro assms(3)[unfolded indep-vars-def, THEN conjunct2, THEN indep-sets-mono]*)  
*(auto simp add: case-bool-if)*  
**thus** *?thesis* **unfolding** *indep-set-def* **using** *assms*  
**by** (*intro indep-sets-cong[THEN iffD1, OF refl - indep-sets-collect-sigma[of  $\lambda i. \{X \ i - 'A \cap \text{space } M \mid A. A \in \text{borel}\} \text{ case-bool } \{t_0..s\} \ \{t\}]]$* )  
*(simp add: sets-natural-filtration sets-vimage-algebra split: bool.split, simp,*  
*intro Int-stableI, clarsimp,metis sets.Int vimage-Int Int-commute Int-left-absorb*  
*Int-left-commute, force simp add: disjoint-family-on-def split: bool.split)*  
**qed**

## 6.2 Adapted Process

We call a collection a stochastic process  $X$  adapted if  $X \ i$  is  $F \ i$ -borel-measurable for all indices  $i$ .

**locale** *adapted-process* = *filtered-measure M F t<sub>0</sub>* **for**  $M \ F \ t_0$  **and**  $X :: - \Rightarrow - \Rightarrow -$   
 $:: \{\text{second-countable-topology, banach}\} +$   
**assumes** *adapted[measurable]:  $\bigwedge i. t_0 \leq i \implies X \ i \in \text{borel-measurable } (F \ i)$*   
**begin**

**lemma** *adaptedE[elim]:*

**assumes**  $\llbracket \bigwedge j \ i. \ t_0 \leq j \implies j \leq i \implies X \ j \in \text{borel-measurable } (F \ i) \rrbracket \implies P$   
**shows**  $P$   
**using** *assms* **using** *adapted* **by** (*metis dual-order.trans borel-measurable-subalgebra sets-F-mono space-F*)

**lemma** *adaptedD*:  
**assumes**  $t_0 \leq j \ j \leq i$   
**shows**  $X \ j \in \text{borel-measurable } (F \ i)$  **using** *assms adaptedE* **by** *meson*

**end**

**locale** *nat-adapted-process* = *adapted-process*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
**locale** *real-adapted-process* = *adapted-process*  $M \ F \ 0 :: \text{real } X$  **for**  $M \ F \ X$

**sublocale** *nat-adapted-process*  $\subseteq$  *nat-filtered-measure* ..  
**sublocale** *real-adapted-process*  $\subseteq$  *real-filtered-measure* ..

**lemma** (*in filtered-measure*) *adapted-process-const-fun*:  
**assumes**  $f \in \text{borel-measurable } (F \ t_0)$   
**shows** *adapted-process*  $M \ F \ t_0 \ (\lambda \cdot. f)$   
**using** *measurable-from-subalg subalgebra-F assms* **by** (*unfold-locales*) *blast*

**lemma** (*in filtered-measure*) *adapted-process-const*:  
**shows** *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \cdot. \ c \ i)$  **by** (*unfold-locales*) *simp*

Again, we cover basic operations.

**context** *adapted-process*  
**begin**

**lemma** *compose-adapted*:  
**assumes**  $\bigwedge i. \ t_0 \leq i \implies f \ i \in \text{borel-measurable borel}$   
**shows** *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ (f \ i) \ (X \ i \ \xi))$   
**by** (*unfold-locales*) (*intro measurable-compose[OF adapted assms]*)

**lemma** *norm-adapted*: *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ \text{norm } (X \ i \ \xi))$  **by** (*fastforce intro: compose-adapted*)

**lemma** *scaleR-right-adapted*:  
**assumes** *adapted-process*  $M \ F \ t_0 \ R$   
**shows** *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ (R \ i \ \xi) \ *_R (X \ i \ \xi))$   
**using** *adapted-process.adapted[OF assms]* *adapted* **by** (*unfold-locales*) *simp*

**lemma** *scaleR-right-const-fun-adapted*:  
**assumes**  $f \in \text{borel-measurable } (F \ t_0)$   
**shows** *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ f \ \xi \ *_R (X \ i \ \xi))$   
**using** *assms* **by** (*fast intro: scaleR-right-adapted adapted-process-const-fun*)

**lemma** *scaleR-right-const-adapted*: *adapted-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ c \ i \ *_R (X \ i \ \xi))$   
**by** (*unfold-locales*) *simp*



**lemma** *add-adapted*:

**assumes** *adapted-process*  $M F t_0 Y$   
**shows** *adapted-process*  $M F t_0 (\lambda i \xi. X i \xi + Y i \xi)$   
**using** *adapted-process.adapted*[*OF assms*] **adapted by** (*unfold-locales*) *simp*

**lemma** *diff-adapted*:

**assumes** *adapted-process*  $M F t_0 Y$   
**shows** *adapted-process*  $M F t_0 (\lambda i \xi. X i \xi - Y i \xi)$   
**using** *adapted-process.adapted*[*OF assms*] **adapted by** (*unfold-locales*) *simp*

**lemma** *uminus-adapted*: *adapted-process*  $M F t_0 (-X)$  **using** *scaleR-right-const-adapted*[*of*  $\lambda -. -1$ ] **by** (*simp add: fun-Compl-def*)

**lemma** *partial-sum-adapted*: *adapted-process*  $M F t_0 (\lambda n \xi. \sum_{i \in \{t_0..n\}} X i \xi)$

**proof** (*unfold-locales*)

**fix**  $i :: 'b$

**have**  $X j \in \text{borel-measurable } (F i)$  **if**  $t_0 \leq j$  **for**  $j$  **using** *that adaptedE* **by** *meson*

**thus**  $(\lambda \xi. \sum_{i \in \{t_0..i\}} X i \xi) \in \text{borel-measurable } (F i)$  **by** *simp*

**qed**

**lemma** *partial-sum'-adapted*: *adapted-process*  $M F t_0 (\lambda n \xi. \sum_{i \in \{t_0..<n\}} X i \xi)$

**proof** (*unfold-locales*)

**fix**  $i :: 'b$

**have**  $X j \in \text{borel-measurable } (F i)$  **if**  $t_0 \leq j$  **for**  $j < i$  **using** *that adaptedE* **by** *fastforce*

**thus**  $(\lambda \xi. \sum_{i \in \{t_0..<i\}} X i \xi) \in \text{borel-measurable } (F i)$  **by** *simp*

**qed**

**end**

In the discrete time case, we have the following lemma which will be useful later on.

**lemma** (*in nat-adapted-process*) *partial-sum-Suc-adapted*: *nat-adapted-process*  $M F (\lambda n \xi. \sum_{i < n} X (Suc i) \xi)$

**proof** (*unfold-locales*)

**fix**  $i$

**have**  $X j \in \text{borel-measurable } (F i)$  **if**  $j \leq i$  **for**  $j$  **using** *that adaptedD* **by** *blast*

**thus**  $(\lambda \xi. \sum_{i < i} X (Suc i) \xi) \in \text{borel-measurable } (F i)$  **by** *auto*

**qed**

**lemma** (*in filtered-measure*) *adapted-process-sum*:

**assumes**  $\bigwedge i. i \in I \implies \text{adapted-process } M F t_0 (X i)$

**shows** *adapted-process*  $M F t_0 (\lambda k \xi. \sum_{i \in I} X i k \xi)$

**proof** –

{

**fix**  $i k$  **assume**  $i \in I$  **and** *asm*:  $t_0 \leq k$

```

    then interpret adapted-process M F t0 X i using assms by simp
    have X i k ∈ borel-measurable M X i k ∈ borel-measurable (F k) using mea-
surable-from-subalg subalgebras adapted asm by (blast, simp)
  }
  thus ?thesis by (unfold-locales) simp
qed

```

An adapted process is necessarily a stochastic process.

```

sublocale adapted-process ⊆ stochastic-process using measurable-from-subalg sub-
algebras adapted by (unfold-locales) blast

```

```

sublocale nat-adapted-process ⊆ nat-stochastic-process ..
sublocale real-adapted-process ⊆ real-stochastic-process ..

```

A stochastic process is always adapted to the natural filtration it generates.

```

lemma (in stochastic-process) adapted-process-natural-filtration: adapted-process
M (natural-filtration M t0 X) t0 X
  using filtered-measure-natural-filtration
  by (intro-locales) (auto simp add: natural-filtration-def intro!: adapted-process-axioms.intro
measurable-family-vimage-algebra)

```

### 6.3 Progressively Measurable Process

```

locale progressive-process = filtered-measure M F t0 for M F t0 and X :: - ⇒ -
⇒ - :: {second-countable-topology, banach} +
  assumes progressive[measurable]: ⋀t. t0 ≤ t ⇒ (λ(i, x). X i x) ∈ borel-measurable
(restrict-space borel {t0..t} ⊗M F t)
begin

```

```

lemma progressiveD:
  assumes S ∈ borel
  shows (λ(j, ξ). X j ξ) -' S ∩ ({t0..i} × space M) ∈ (restrict-space borel {t0..i}
⊗M F i)
  using measurable-sets[OF progressive, OF - assms, of i]
  by (cases t0 ≤ i) (auto simp add: space-restrict-space sets-pair-measure space-pair-measure)
end

```

```

locale nat-progressive-process = progressive-process M F 0 :: nat X for M F X
locale real-progressive-process = progressive-process M F 0 :: real X for M F X

```

```

lemma (in filtered-measure) progressive-process-const-fun:
  assumes f ∈ borel-measurable (F t0)
  shows progressive-process M F t0 (λ-. f)
proof (unfold-locales)
  fix i assume asm: t0 ≤ i
  have f ∈ borel-measurable (F i) using borel-measurable-mono[OF order.refl asm]
  assms by blast

```

**thus**  $\text{case-prod } (\lambda \cdot. f) \in \text{borel-measurable } (\text{restrict-space borel } \{t_0..i\} \otimes_M F i)$   
**using**  $\text{measurable-compose}[OF \text{ measurable-snd}]$  **by**  $\text{simp}$   
**qed**

**lemma** (*in filtered-measure*) *progressive-process-const*:  
**assumes**  $c \in \text{borel-measurable borel}$   
**shows**  $\text{progressive-process } M F t_0 (\lambda i \cdot. c i)$   
**using**  $\text{assms}$  **by** (*unfold-locales*) (*auto simp add: measurable-split-conv intro!*:  
 $\text{measurable-compose}[OF \text{ measurable-fst}] \text{ measurable-restrict-space1}$ )

**context** *progressive-process*  
**begin**

**lemma** *compose-progressive*:  
**assumes**  $\text{case-prod } f \in \text{borel-measurable borel}$   
**shows**  $\text{progressive-process } M F t_0 (\lambda i \xi. (f i) (X i \xi))$   
**proof**  
**fix**  $i$  **assume**  $\text{asm}: t_0 \leq i$   
**have**  $(\lambda(j, \xi). (j, X j \xi)) \in (\text{restrict-space borel } \{t_0..i\} \otimes_M F i) \rightarrow_M \text{borel } \otimes_M \text{borel}$   
**using**  $\text{progressive}[OF \text{ asm}] \text{ measurable-fst''}[OF \text{ measurable-restrict-space1}, OF \text{ measurable-id}]$   
**by** (*auto simp add: measurable-pair-iff measurable-split-conv*)  
**moreover** **have**  $(\lambda(j, \xi). f j (X j \xi)) = \text{case-prod } f o ((\lambda(j, y). (j, y)) o (\lambda(j, \xi). (j, X j \xi)))$  **by** *fastforce*  
**ultimately show**  $(\lambda(j, \xi). (f j) (X j \xi)) \in \text{borel-measurable } (\text{restrict-space borel } \{t_0..i\} \otimes_M F i)$  **using**  $\text{assms}$  **by** (*simp add: borel-prod*)  
**qed**

**lemma** *norm-progressive*:  $\text{progressive-process } M F t_0 (\lambda i \xi. \text{norm } (X i \xi))$  **using**  $\text{measurable-compose}[OF \text{ progressive borel-measurable-norm}]$  **by** (*unfold-locales*) *simp*

**lemma** *scaleR-right-progressive*:  
**assumes**  $\text{progressive-process } M F t_0 R$   
**shows**  $\text{progressive-process } M F t_0 (\lambda i \xi. (R i \xi) *_R (X i \xi))$   
**using**  $\text{progressive-process.progressive}[OF \text{ assms}]$  **by** (*unfold-locales*) (*simp add: progressive assms*)

**lemma** *scaleR-right-const-fun-progressive*:  
**assumes**  $f \in \text{borel-measurable } (F t_0)$   
**shows**  $\text{progressive-process } M F t_0 (\lambda i \xi. f \xi *_R (X i \xi))$   
**using**  $\text{assms}$  **by** (*fast intro: scaleR-right-progressive progressive-process-const-fun*)

**lemma** *scaleR-right-const-progressive*:  
**assumes**  $c \in \text{borel-measurable borel}$   
**shows**  $\text{progressive-process } M F t_0 (\lambda i \xi. c i *_R (X i \xi))$   
**using**  $\text{assms}$  **by** (*fastforce intro: scaleR-right-progressive progressive-process-const*)

**lemma** *add-progressive*:

**assumes** *progressive-process*  $M F t_0 Y$   
**shows** *progressive-process*  $M F t_0 (\lambda i \xi. X i \xi + Y i \xi)$   
**using** *progressive-process.progressive*[*OF assms*] **by** (*unfold-locales*) (*simp add: progressive assms*)

**lemma** *diff-progressive*:

**assumes** *progressive-process*  $M F t_0 Y$   
**shows** *progressive-process*  $M F t_0 (\lambda i \xi. X i \xi - Y i \xi)$   
**using** *progressive-process.progressive*[*OF assms*] **by** (*unfold-locales*) (*simp add: progressive assms*)

**lemma** *uminus-progressive*: *progressive-process*  $M F t_0 (-X)$  **using** *scaleR-right-const-progressive*[*of*  $\lambda-. -1$ ] **by** (*simp add: fun-Compl-def*)

**end**

A progressively measurable process is also adapted.

**sublocale** *progressive-process*  $\subseteq$  *adapted-process* **using** *measurable-compose-rev*[*OF progressive measurable-Pair1*] **unfolding** *prod.case space-restrict-space* **by** *unfold-locales simp*

**sublocale** *nat-progressive-process*  $\subseteq$  *nat-adapted-process* ..

**sublocale** *real-progressive-process*  $\subseteq$  *real-adapted-process* ..

In the discrete setting, adaptedness is equivalent to progressive measurability.

**theorem** *nat-progressive-iff-adapted*: *nat-progressive-process*  $M F X \longleftrightarrow$  *nat-adapted-process*  $M F X$

**proof** (*intro iffI*)

**assume** *asm*: *nat-progressive-process*  $M F X$   
**interpret** *nat-progressive-process*  $M F X$  **by** (*rule asm*)  
**show** *nat-adapted-process*  $M F X$  ..

**next**

**assume** *asm*: *nat-adapted-process*  $M F X$   
**interpret** *nat-adapted-process*  $M F X$  **by** (*rule asm*)  
**show** *nat-progressive-process*  $M F X$

**proof** (*unfold-locales, intro borel-measurableI*)

**fix**  $S :: 'b \text{ set}$  **and**  $i :: \text{nat}$  **assume** *open-S*: *open S*

{

**fix**  $j$  **assume** *asm*:  $j \leq i$

**hence**  $X j - ' S \cap \text{space } M \in F i$  **using** *adaptedD*[*of j, THEN measurable-sets*] *space-F open-S* **by** *fastforce*

**moreover have** *case-prod*  $X - ' S \cap \{j\} \times \text{space } M = \{j\} \times (X j - ' S \cap \text{space } M)$  **for**  $j$  **by** *fast*

**moreover have**  $\{j :: \text{nat}\} \in \text{restrict-space borel } \{0..i\}$  **using** *asm* **by** (*simp add: sets-restrict-space-iff*)

ultimately have  $\text{case-prod } X - 'S \cap \{j\} \times \text{space } M \in \text{restrict-space borel } \{0..i\} \otimes_M F i$  **by** *simp*

**}**

hence  $(\lambda j. (\lambda(x, y). X x y) - 'S \cap \{j\} \times \text{space } M) ' \{..i\} \subseteq \text{restrict-space borel } \{0..i\} \otimes_M F i$  **by** *blast*

moreover have  $\text{case-prod } X - 'S \cap \text{space } (\text{restrict-space borel } \{0..i\} \otimes_M F i) = (\bigcup j \leq i. \text{case-prod } X - 'S \cap \{j\} \times \text{space } M)$  **unfolding** *space-pair-measure space-restrict-space space-F* **by** *force*

ultimately show  $\text{case-prod } X - 'S \cap \text{space } (\text{restrict-space borel } \{0..i\} \otimes_M F i) \in \text{restrict-space borel } \{0..i\} \otimes_M F i$  **by** *(metis sets.countable-UN)*

**qed**

**qed**

## 6.4 Predictable Process

We introduce the constant  $\Sigma_P$  to denote the predictable  $\sigma$ -algebra.

**context** *linearly-filtered-measure*  
**begin**

**definition**  $\Sigma_P :: ('b \times 'a) \text{ measure where predictable-sigma: } \Sigma_P \equiv \text{sigma } (\{t_0..\} \times \text{space } M) (\{\{s <..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in F t_0\})$

**lemma** *space-predictable-sigma[simp]: space  $\Sigma_P = (\{t_0..\} \times \text{space } M)$  unfolding predictable-sigma space-measure-of-conv* **by** *blast*

**lemma** *sets-predictable-sigma: sets  $\Sigma_P = \text{sigma-sets } (\{t_0..\} \times \text{space } M) (\{\{s <..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in F t_0\})$*

**unfolding** *predictable-sigma using space-F sets.sets-into-space* **by** *(subst sets-measure-of) fastforce+*

**lemma** *measurable-predictable-sigma-snd:*

**assumes** *countable  $\mathcal{I}$   $\mathcal{I} \subseteq \{\{s <..t\} \mid s t. t_0 \leq s \wedge s < t\} \{t_0 <..\} \subseteq (\bigcup \mathcal{I})$*

**shows**  *$\text{snd} \in \Sigma_P \rightarrow_M F t_0$*

**proof** *(intro measurableI)*

**fix**  $S :: 'a \text{ set}$  **assume** *asm:  $S \in F t_0$*

**have** *countable: countable  $((\lambda I. I \times S) ' \mathcal{I})$  using *assms(1)* by *blast**

**have**  $(\lambda I. I \times S) ' \mathcal{I} \subseteq \{\{s <..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\}$  **using** *sets-F-mono[OF order-refl, THEN subsetD, OF - asm] assms(2)* **by** *blast*

**hence**  $(\bigcup I \in \mathcal{I}. I \times S) \cup \{t_0\} \times S \in \Sigma_P$  **unfolding** *sets-predictable-sigma using asm* **by** *(intro sigma-sets-Un[OF sigma-sets-UNION[OF countable] sigma-sets.Basic] sigma-sets.Basic) blast+*

**moreover** have  $\text{snd} - 'S \cap \text{space } \Sigma_P = \{t_0..\} \times S$  **using** *sets.sets-into-space[OF asm]* **by** *fastforce*

**moreover** have  $\{t_0\} \cup \{t_0 <..\} = \{t_0..\}$  **by** *auto*

**moreover** have  $(\bigcup I \in \mathcal{I}. I \times S) \cup \{t_0\} \times S = \{t_0..\} \times S$  **using** *assms(2,3) calculation(3)* **by** *fastforce*

**ultimately** show  $\text{snd} - 'S \cap \text{space } \Sigma_P \in \Sigma_P$  **by** *argo*

**qed** *(auto)*

**lemma** *measurable-predictable-sigma-fst*:

**assumes** *countable*  $\mathcal{I}$   $\mathcal{I} \subseteq \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\} \ \{t_0 <..\} \subseteq (\bigcup \mathcal{I})$

**shows**  $\text{fst} \in \Sigma_P \rightarrow_M \text{borel}$

**proof** –

**have**  $A \times \text{space } M \in \text{sets } \Sigma_P$  **if**  $A \in \text{sigma-sets } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$  **for**  $A$  **unfolding** *sets-predictable-sigma* **using** *that*

**proof** (*induction rule: sigma-sets.induct*)

**case** (*Basic a*)

**thus** *?case* **using** *space-F sets.top* **by** *blast*

**next**

**case** (*Compl a*)

**have**  $(\{t_0..\} - a) \times \text{space } M = \{t_0..\} \times \text{space } M - a \times \text{space } M$  **by** *blast*

**then show** *?case* **using** *Compl(2)* [*THEN sigma-sets.Compl*] **by** *presburger*

**next**

**case** (*Union a*)

**have**  $\bigcup (\text{range } a) \times \text{space } M = \bigcup (\text{range } (\lambda i. a \ i \times \text{space } M))$  **by** *blast*

**then show** *?case* **using** *Union(2)* [*THEN sigma-sets.Union*] **by** *presburger*

**qed** (*auto*)

**moreover have** *restrict-space borel*  $\{t_0..\} = \text{sigma } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$

**proof** –

**have** *sigma-sets*  $\{t_0..\} \ ((\cap) \ \{t_0..\} \ ' \text{sigma-sets UNIV } (\text{range greaterThan})) = \text{sigma-sets } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$

**proof** (*intro sigma-sets-eqI ; clarify*)

**fix**  $A :: 'b \text{ set}$  **assume** *asm*:  $A \in \text{sigma-sets UNIV } (\text{range greaterThan})$

**thus**  $\{t_0..\} \cap A \in \text{sigma-sets } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$

**proof** (*induction rule: sigma-sets.induct*)

**case** (*Basic a*)

**then obtain**  $s$  **where**  $a = \{s <..\}$  **by** *blast*

**show** *?case*

**proof** (*cases*  $t_0 \leq s$ )

**case** *True*

**hence**  $\ast: \{t_0..\} \cap a = (\bigcup i \in \mathcal{I}. \{s <..\} \cap i)$  **using** *s assms(3)* **by** *force*

**have**  $((\cap) \ \{s <..\} \ ' \mathcal{I}) \subseteq \text{sigma-sets } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$

**proof** (*clarify*)

**fix**  $A$  **assume**  $A \in \mathcal{I}$

**then obtain**  $s' \ t'$  **where**  $A = \{s' <..t'\} \ t_0 \leq s' \ s' < t'$  **using** *assms(2)*

**by** *blast*

**hence**  $\{s <..\} \cap A = \{\max s \ s' <..t'\}$  **by** *fastforce*

**moreover have**  $t_0 \leq \max s \ s'$  **using**  $A \ \text{True}$  **by** *linarith*

**moreover have**  $\max s \ s' < t'$  **if**  $s < t'$  **using**  $A \ \text{that}$  **by** *linarith*

**moreover have**  $\{s <..\} \cap A = \{\}$  **if**  $\neg s < t'$  **using**  $A \ \text{that}$  **by** *force*

**ultimately show**  $\{s <..\} \cap A \in \text{sigma-sets } \{t_0..\} \ \{\{s <..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$  **by** (*cases*  $s < t'$ ) (*blast, simp add: sigma-sets.Empty*)

**qed**

**thus** *?thesis* **unfolding**  $\ast$  **using** *assms(1)* **by** (*intro sigma-sets-UNION*)

*auto*

**next**

```

    case False
    hence  $\{t_0..\} \cap a = \{t_0..\}$  using s by force
    thus ?thesis using sigma-sets-top by auto
  qed
next
case (Compl a)
have  $\{t_0..\} \cap (UNIV - a) = \{t_0..\} - (\{t_0..\} \cap a)$  by blast
then show ?case using Compl(2)[THEN sigma-sets.Compl] by presburger
next
case (Union a)
have  $\{t_0..\} \cap \bigcup (\text{range } a) = \bigcup (\text{range } (\lambda i. \{t_0..\} \cap a \ i))$  by blast
then show ?case using Union(2)[THEN sigma-sets.Union] by presburger
qed (simp add: sigma-sets.Empty)
next
fix s t assume asm:  $t_0 \leq s < t$ 
hence *:  $\{s<..t\} = \{s<..\} \cap (\{t_0..\} - \{t<..\})$  by force
have  $\{s<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\})$  ‘sigma-sets UNIV (range greaterThan))
using asm by (intro sigma-sets.Basic) auto
moreover have  $\{t_0..\} - \{t<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\})$  ‘sigma-sets UNIV (range greaterThan)) using asm by (intro sigma-sets.Compl sigma-sets.Basic) auto
ultimately show  $\{s<..t\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\})$  ‘sigma-sets UNIV (range greaterThan)) unfolding * Int-range-binary[of  $\{s<..\}$ ] by (intro sigma-sets.Inter[OF - binary-in-sigma-sets]) auto
qed
thus ?thesis unfolding borel-Ioi restrict-space-def emeasure-sigma by (force intro: sigma-eqI)
qed
ultimately have restrict-space borel  $\{t_0..\} \otimes_M \text{sigma } (\text{space } M) \{\} \subseteq \text{sets } \Sigma_P$ 

    unfolding sets-pair-measure space-restrict-space space-measure-of-conv
    using space-predictable-sigma sets.sigma-algebra-axioms[of  $\Sigma_P$ ]
    by (intro sigma-algebra.sigma-sets-subset) (auto simp add: sigma-sets.empty-eq sets-measure-of-conv)
    moreover have space (restrict-space borel  $\{t_0..\} \otimes_M \text{sigma } (\text{space } M) \{\}$ ) = space  $\Sigma_P$  by (simp add: space-pair-measure)
    moreover have fst  $\in$  restrict-space borel  $\{t_0..\} \otimes_M \text{sigma } (\text{space } M) \{\} \rightarrow_M$  borel by (fastforce intro: measurable-fst'[OF measurable-restrict-space1, of  $\lambda x. x$ ])

    ultimately show ?thesis by (meson borel-measurable-subalgebra)
  qed
end

locale predictable-process = linearly-filtered-measure M F t_0 for M F t_0 and X ::
-  $\Rightarrow$  -  $\Rightarrow$  - :: {second-countable-topology, banach} +
    assumes predictable:  $(\lambda(t, x). X \ t \ x) \in \text{borel-measurable } \Sigma_P$ 
begin

```

**lemmas** *predictableD = measurable-sets[OF predictable, unfolded space-predictable-sigma]*

**end**

**locale** *nat-predictable-process = predictable-process M F 0 :: nat X for M F X*  
**locale** *real-predictable-process = predictable-process M F 0 :: real X for M F X*

**lemma** (in *nat-filtered-measure*) *measurable-predictable-sigma-snd'*:  
**shows** *snd*  $\in \Sigma_P \rightarrow_M F 0$   
**by** (intro *measurable-predictable-sigma-snd*[of range ( $\lambda x. \{Suc\ x\}$ )] (force | simp  
*add: greaterThan-0*)+

**lemma** (in *nat-filtered-measure*) *measurable-predictable-sigma-fst'*:  
**shows** *fst*  $\in \Sigma_P \rightarrow_M \text{borel}$   
**by** (intro *measurable-predictable-sigma-fst*[of range ( $\lambda x. \{Suc\ x\}$ )] (force | simp  
*add: greaterThan-0*)+

**lemma** (in *real-filtered-measure*) *measurable-predictable-sigma-snd'*:  
**shows** *snd*  $\in \Sigma_P \rightarrow_M F 0$   
**using** *real-arch-simple* **by** (intro *measurable-predictable-sigma-snd*[of range ( $\lambda x::nat.$   
 $\{0 <..real\ (Suc\ x)\}$ )] (fastforce intro: *add-increasing*)+

**lemma** (in *real-filtered-measure*) *measurable-predictable-sigma-fst'*:  
**shows** *fst*  $\in \Sigma_P \rightarrow_M \text{borel}$   
**using** *real-arch-simple* **by** (intro *measurable-predictable-sigma-fst*[of range ( $\lambda x::nat.$   
 $\{0 <..real\ (Suc\ x)\}$ )] (fastforce intro: *add-increasing*)+

We show sufficient conditions for functions constant in one argument to constitute a predictable process. In contrast to the cases before, this is not a triviality.

**lemma** (in *linearly-filtered-measure*) *predictable-process-const-fun*:  
**assumes** *snd*  $\in \Sigma_P \rightarrow_M F t_0$  *f*  $\in \text{borel-measurable}\ (F\ t_0)$   
**shows** *predictable-process M F t\_0* ( $\lambda\cdot. f$ )  
**using** *measurable-compose-rev*[OF *assms*(2)] *assms*(1) **by** (*unfold-locales*) (*auto*  
*simp add: measurable-split-conv*)

**lemma** (in *nat-filtered-measure*) *predictable-process-const-fun'*[intro]:  
**assumes** *f*  $\in \text{borel-measurable}\ (F\ 0)$   
**shows** *nat-predictable-process M F* ( $\lambda\cdot. f$ )  
**using** *assms* **by** (intro *predictable-process-const-fun*[OF *measurable-predictable-sigma-snd'*,  
*THEN nat-predictable-process.intro*])

**lemma** (in *real-filtered-measure*) *predictable-process-const-fun'*[intro]:  
**assumes** *f*  $\in \text{borel-measurable}\ (F\ 0)$   
**shows** *real-predictable-process M F* ( $\lambda\cdot. f$ )  
**using** *assms* **by** (intro *predictable-process-const-fun*[OF *measurable-predictable-sigma-snd'*,  
*THEN real-predictable-process.intro*])



**lemma** (in *linearly-filtered-measure*) *predictable-process-const*:  
 assumes  $fst \in \text{borel-measurable } \Sigma_P \ c \in \text{borel-measurable borel}$   
 shows *predictable-process*  $M \ F \ t_0 \ (\lambda i \ -. \ c \ i)$   
 using *assms* by (unfold-locale) (simp add: measurable-split-conv)

**lemma** (in *linearly-filtered-measure*) *predictable-process-const-const[intro]*:  
 shows *predictable-process*  $M \ F \ t_0 \ (\lambda \cdot \ -. \ c)$   
 by (unfold-locale) simp

**lemma** (in *nat-filtered-measure*) *predictable-process-const'[intro]*:  
 assumes  $c \in \text{borel-measurable borel}$   
 shows *nat-predictable-process*  $M \ F \ (\lambda i \ -. \ c \ i)$   
 using *assms* by (intro *predictable-process-const*[OF *measurable-predictable-sigma-fst'*,  
 THEN *nat-predictable-process.intro*])

**lemma** (in *real-filtered-measure*) *predictable-process-const'[intro]*:  
 assumes  $c \in \text{borel-measurable borel}$   
 shows *real-predictable-process*  $M \ F \ (\lambda i \ -. \ c \ i)$   
 using *assms* by (intro *predictable-process-const*[OF *measurable-predictable-sigma-fst'*,  
 THEN *real-predictable-process.intro*])

**context** *predictable-process*  
**begin**

**lemma** *compose-predictable*:  
 assumes  $fst \in \text{borel-measurable } \Sigma_P \ \text{case-prod } f \in \text{borel-measurable borel}$   
 shows *predictable-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ (f \ i) \ (X \ i \ \xi))$   
**proof**  
 have  $(\lambda(i, \xi). \ (i, X \ i \ \xi)) \in \Sigma_P \rightarrow_M \text{borel} \otimes_M \text{borel}$  using *predictable assms*(1)  
 by (auto simp add: measurable-pair-iff measurable-split-conv)  
 moreover have  $(\lambda(i, \xi). \ f \ i \ (X \ i \ \xi)) = \text{case-prod } f \ o \ (\lambda(i, \xi). \ (i, X \ i \ \xi))$  by  
*fastforce*  
 ultimately show  $(\lambda(i, \xi). \ f \ i \ (X \ i \ \xi)) \in \text{borel-measurable } \Sigma_P$  unfolding *borel-prod*  
 using *assms* by simp  
**qed**

**lemma** *norm-predictable*: *predictable-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ \text{norm } (X \ i \ \xi))$  using  
*measurable-compose*[OF *predictable borel-measurable-norm*]  
 by (unfold-locale) (simp add: prod.case-distrib)

**lemma** *scaleR-right-predictable*:  
 assumes *predictable-process*  $M \ F \ t_0 \ R$   
 shows *predictable-process*  $M \ F \ t_0 \ (\lambda i \ \xi. \ (R \ i \ \xi) *_R (X \ i \ \xi))$   
 using *predictable predictable-process.predictable*[OF *assms*] by (unfold-locale)  
 (auto simp add: measurable-split-conv)

**lemma** *scaleR-right-const-fun-predictable*:  
 assumes  $snd \in \Sigma_P \rightarrow_M \ F \ t_0 \ f \in \text{borel-measurable } (F \ t_0)$

**shows** *predictable-process*  $M F t_0 (\lambda i \xi. f \xi *_R (X i \xi))$   
**using** *assms* **by** (*fast intro: scaleR-right-predictable predictable-process-const-fun*)

**lemma** *scaleR-right-const-predictable*:  
**assumes**  $fst \in \text{borel-measurable } \Sigma_P \ c \in \text{borel-measurable borel}$   
**shows** *predictable-process*  $M F t_0 (\lambda i \xi. c i *_R (X i \xi))$   
**using** *assms* **by** (*fastforce intro: scaleR-right-predictable predictable-process-const*)

**lemma** *scaleR-right-const'-predictable*: *predictable-process*  $M F t_0 (\lambda i \xi. c *_R (X i \xi))$   
**by** (*fastforce intro: scaleR-right-predictable*)

**lemma** *add-predictable*:  
**assumes** *predictable-process*  $M F t_0 Y$   
**shows** *predictable-process*  $M F t_0 (\lambda i \xi. X i \xi + Y i \xi)$   
**using** *predictable predictable-process.predictable[OF assms]* **by** (*unfold-locales*)  
*(auto simp add: measurable-split-conv)*

**lemma** *diff-predictable*:  
**assumes** *predictable-process*  $M F t_0 Y$   
**shows** *predictable-process*  $M F t_0 (\lambda i \xi. X i \xi - Y i \xi)$   
**using** *predictable predictable-process.predictable[OF assms]* **by** (*unfold-locales*)  
*(auto simp add: measurable-split-conv)*

**lemma** *uminus-predictable*: *predictable-process*  $M F t_0 (-X)$  **using** *scaleR-right-const'-predictable[of -1]* **by** (*simp add: fun-Compl-def*)

**end**

Every predictable process is also progressively measurable.

**sublocale** *predictable-process*  $\subseteq$  *progressive-process*

**proof** (*unfold-locales*)

**fix**  $i :: 'b$  **assume**  $asm: t_0 \leq i$   
 $\{$   
**fix**  $S :: ('b \times 'a)$  *set* **assume**  $S \in \{\{s <..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in F t_0\}$   
**hence**  $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) \in \text{restrict-space borel } \{t_0..i\} \otimes_M F$   
 $i$

**proof**

**assume**  $S \in \{\{s <..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\}$   
**then obtain**  $s t A$  **where**  $S\text{-is: } S = \{s <..t\} \times A \ t_0 \leq s \ s < t \ A \in F s$  **by**  
*blast*

**hence**  $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) = \{s <.. \min i t\} \times A$  **using**  
*sets.sets-into-space[OF S-is(4)]* **by** *auto*

**then show** *?thesis* **using**  $S\text{-is sets-F-mono[of } s \ i]$  **by** (*cases*  $s \leq i$ ) (*fastforce simp add: sets-restrict-space-iff*)**+**

**next**

**assume**  $S \in \{\{t_0\} \times A \mid A. A \in F t_0\}$   
**then obtain**  $A$  **where**  $S\text{-is: } S = \{t_0\} \times A \ A \in F t_0$  **by** *blast*

**hence**  $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) = \{t_0\} \times A$  **using** *asm sets.sets-into-space*[*OF S-is(2)*] **by** *auto*  
**thus** *?thesis using S-is(2) sets-F-mono*[*OF order-refl asm*] *asm by (fastforce simp add: sets-restrict-space-iff)*  
**qed**  
**hence**  $(\lambda x. x) - ' S \cap \text{space } (\text{restrict-space borel } \{t_0..i\} \otimes_M F i) \in \text{restrict-space borel } \{t_0..i\} \otimes_M F i$  **by** (*simp add: space-pair-measure space-F*[*OF asm*])  
**}**  
**moreover have**  $\{\{s<..t\} \times A \mid A s t. A \in \text{sets } (F s) \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in \text{sets } (F t_0)\} \subseteq \text{Pow } (\{t_0..i\} \times \text{space } M)$  **using** *sets.sets-into-space* **by** *force*  
**ultimately have**  $(\lambda x. x) \in \text{restrict-space borel } \{t_0..i\} \otimes_M F i \rightarrow_M \Sigma_P$  **using** *space-F*[*OF asm*] **by** (*intro measurable-sigma-sets*[*OF sets-predictable-sigma*]) (*fast, force simp add: space-pair-measure*)  
**thus** *case-prod*  $X \in \text{borel-measurable } (\text{restrict-space borel } \{t_0..i\} \otimes_M F i)$  **using** *predictable* **by** *simp*  
**qed**

**sublocale** *nat-predictable-process*  $\subseteq$  *nat-progressive-process* ..  
**sublocale** *real-predictable-process*  $\subseteq$  *real-progressive-process* ..

The following lemma characterizes predictability in a discrete-time setting.

**lemma** (*in nat-filtered-measure*) *sets-in-filtration*:

**assumes**  $(\bigcup i. \{i\} \times A i) \in \Sigma_P$   
**shows**  $A (Suc i) \in F i \wedge 0 \in F 0$   
**using** *assms unfolding sets-predictable-sigma*  
**proof** (*induction*  $(\bigcup i. \{i\} \times A i)$  *arbitrary: A*)  
**case Basic**  
**{**  
**assume**  $\exists S. (\bigcup i. \{i\} \times A i) = \{0\} \times S$   
**then obtain**  $S$  **where**  $S: (\bigcup i. \{i\} \times A i) = \{bot\} \times S$  **unfolding** *bot-nat-def*  
**by** *blast*  
**hence**  $S \in F bot$  **using** *Basic* **by** (*fastforce simp add: times-eq-iff bot-nat-def*)  
**moreover have**  $A i = \{\}$  **if**  $i \neq bot$  **for**  $i$  **using** *that S* **by** *blast*  
**moreover have**  $A bot = S$  **using**  $S$  **by** *blast*  
**ultimately have**  $A (Suc i) \in F i \wedge 0 \in F 0$  **for**  $i$  **unfolding** *bot-nat-def* **by** (*auto simp add: bot-nat-def*)  
**}**  
**note**  $*$  **=** *this*  
**{**  
**assume**  $\nexists S. (\bigcup i. \{i\} \times A i) = \{0\} \times S$   
**then obtain**  $s t B$  **where**  $B: (\bigcup i. \{i\} \times A i) = \{s<..t\} \times B$   $B \in \text{sets } (F s)$   
 $s < t$  **using** *Basic* **by** *auto*  
**hence**  $A i = B$  **if**  $i \in \{s<..t\}$  **for**  $i$  **using** *that* **by** *fast*  
**moreover have**  $A i = \{\}$  **if**  $i \notin \{s<..t\}$  **for**  $i$  **using**  $B$  **that** **by** *fastforce*  
**ultimately have**  $A (Suc i) \in F i \wedge 0 \in F 0$  **for**  $i$  **unfolding** *bot-nat-def* **using**  $B$  *sets-F-mono* **by** (*auto simp add: bot-nat-def*) (*metis less-Suc-eq-le sets.empty-sets subset-eq*)  
**}**

```

note  $** = this$ 
show  $A (Suc\ i) \in sets\ (F\ i)\ A\ 0 \in sets\ (F\ 0)$  using  $*(1)[of\ i]\ *(2)\ ***(1)[of\ i]$ 
 $**(2)$  by blast+
next
  case Empty
  {
    case 1
    then show  $?case$  using Empty by simp
  next
    case 2
    then show  $?case$  using Empty by simp
  }
next
  case (Compl a)
  have  $a-in: a \subseteq \{0..\} \times space\ M$  using Compl(1) sets.sets-into-space sets-predictable-sigma
space-predictable-sigma by metis
  hence  $A-in: A\ i \subseteq space\ M$  for  $i$  using Compl(4) by blast
  have  $a: a = \{0..\} \times space\ M - (\bigcup i. \{i\} \times A\ i)$  using  $a-in$  Compl(4) by blast
  also have  $\dots = -(\bigcap j. -(\{j\} \times (space\ M - A\ j)))$  by blast
  also have  $\dots = (\bigcup j. \{j\} \times (space\ M - A\ j))$  by blast
  finally have  $*$ :  $(space\ M - A\ (Suc\ i)) \in F\ i\ (space\ M - A\ 0) \in F\ 0$  using
Compl(2,3) by auto
  {
    case 1
    then show  $?case$  using  $*$   $A-in$  by (metis bot-nat-0.extremum double-diff
sets.Diff sets.top sets-F-mono sets-le-imp-space-le space-F)
  next
    case 2
    then show  $?case$  using  $*$   $A-in$  by (metis bot-nat-0.extremum double-diff
sets.Diff sets.top sets-F-mono sets-le-imp-space-le space-F)
  }
next
  case (Union a)
  have  $a-in: a \subseteq \{0..\} \times space\ M$  for  $i$  using Union(1) sets.sets-into-space
sets-predictable-sigma space-predictable-sigma by metis
  hence  $A-in: A\ i \subseteq space\ M$  for  $i$  using Union(4) by blast
  have  $snd\ x \in snd\ ' (a\ i \cap (\{fst\ x\} \times space\ M))$  if  $x \in a\ i$  for  $i\ x$  using that
a-in by fastforce
  hence  $a-i: a\ i = (\bigcup j. \{j\} \times (snd\ ' (a\ i \cap (\{j\} \times space\ M))))$  for  $i$  by force
  have  $A-i: A\ i = snd\ ' (\bigcup (range\ a) \cap (\{i\} \times space\ M))$  for  $i$  unfolding
Union(4) using  $A-in$  by force
  have  $*$ :  $snd\ ' (a\ j \cap (\{Suc\ i\} \times space\ M)) \in F\ i\ snd\ ' (a\ j \cap (\{0\} \times space\ M))$ 
 $\in F\ 0$  for  $j$  using Union(2,3)[OF a-i] by auto
  {
    case 1
    have  $(\bigcup j. snd\ ' (a\ j \cap (\{Suc\ i\} \times space\ M))) \in F\ i$  using  $*$  by fast
    moreover have  $(\bigcup j. snd\ ' (a\ j \cap (\{Suc\ i\} \times space\ M))) = snd\ ' (\bigcup (range$ 
a) \cap (\{Suc\ i\} \times space\ M)) by fast
    ultimately show  $?case$  using  $A-i$  by metis
  }

```

```

next
  case 2
  have ( $\bigcup j. \text{snd } (a\ j \cap (\{0\} \times \text{space } M)) \in F\ 0$ ) using * by fast
  moreover have ( $\bigcup j. \text{snd } (a\ j \cap (\{0\} \times \text{space } M)) = \text{snd } (\bigcup (\text{range } a) \cap (\{0\} \times \text{space } M))$ ) by fast
  ultimately show ?case using A-i by metis
}
qed

```

This leads to the following useful fact.

```

lemma (in nat-predictable-process) adapted-Suc: nat-adapted-process M F ( $\lambda i. X$ 
( $\text{Suc } i$ ))
proof (unfold-locales, intro borel-measurableI)
  fix S :: 'b set and i assume open-S: open S
  have { $\text{Suc } i$ } = { $i <.. \text{Suc } i$ } by fastforce
  hence { $\text{Suc } i$ }  $\times$  space M  $\in \Sigma_P$  using space-F[symmetric, of i] unfolding
sets-predictable-sigma by (intro sigma-sets.Basic) blast
  moreover have case-prod X -' S  $\cap$  (UNIV  $\times$  space M)  $\in \Sigma_P$  unfolding
atLeast-0[symmetric] using open-S by (intro predictableD, simp add: borel-open)
  ultimately have case-prod X -' S  $\cap$  ({ $\text{Suc } i$ }  $\times$  space M)  $\in \Sigma_P$  unfolding
sets-predictable-sigma using space-F sets.sets-into-space
  by (subst Times-Int-distrib1[of { $\text{Suc } i$ } UNIV space M, simplified], subst
inf commute, subst Int-assoc[symmetric], subst Int-range-binary)
  (intro sigma-sets-Inter binary-in-sigma-sets, fast)+
  moreover have case-prod X -' S  $\cap$  ({ $\text{Suc } i$ }  $\times$  space M) = { $\text{Suc } i$ }  $\times$  (X ( $\text{Suc } i$ )
- ' S  $\cap$  space M) by (auto simp add: le-Suc-eq)
  moreover have ... = ( $\bigcup j. \{j\} \times$  (if  $j = \text{Suc } i$  then (X ( $\text{Suc } i$ ) - ' S  $\cap$  space M)
else {})) by (force split: if-splits)
  ultimately have ( $\bigcup j. \{j\} \times$  (if  $j = \text{Suc } i$  then (X ( $\text{Suc } i$ ) - ' S  $\cap$  space M) else
{}))  $\in \Sigma_P$  by argo
  thus X ( $\text{Suc } i$ ) - ' S  $\cap$  space (F i)  $\in$  sets (F i) using sets-in-filtration[of  $\lambda j.$ 
if  $j = \text{Suc } i$  then (X ( $\text{Suc } i$ ) - ' S  $\cap$  space M) else {}] space-F[OF zero-le] by
presburger
qed

```

The following lemma characterizes predictability in the discrete setting.

```

theorem nat-predictable-process-iff: nat-predictable-process M F X  $\longleftrightarrow$  nat-adapted-process
M F ( $\lambda i. X$  ( $\text{Suc } i$ ))  $\wedge$  X 0  $\in$  borel-measurable (F 0)
proof (intro iffI)
  assume asm: nat-adapted-process M F ( $\lambda i. X$  ( $\text{Suc } i$ ))  $\wedge$  X 0  $\in$  borel-measurable
(F 0)
  interpret nat-adapted-process M F  $\lambda i. X$  ( $\text{Suc } i$ ) using asm by blast
  have ( $\lambda(x, y). X\ x\ y$ )  $\in$  borel-measurable  $\Sigma_P$ 
  proof (intro borel-measurableI)
    fix S :: 'b set assume open-S: open S
    have { $i$ }  $\times$  (X i - ' S  $\cap$  space M)  $\in$  sets  $\Sigma_P$  for i
    proof (cases i)
      case 0
      then show ?thesis unfolding sets-predictable-sigma

```

```

    using measurable-sets[OF - borel-open[OF open-S], of X 0 F 0] asm by auto
  next
    case (Suc i)
    have {Suc i} = {i <.. Suc i} by fastforce
    then show ?thesis unfolding sets-predictable-sigma
      using measurable-sets[OF adapted borel-open[OF open-S], of i]
      by (intro sigma-sets.Basic, auto simp add: Suc)
    qed
    moreover have ( $\lambda(x, y). X\ x\ y$ ) - ' S  $\cap$  space  $\Sigma_P = (\bigcup i. \{i\} \times (X\ i - ' S \cap$ 
space M)) by fastforce
    ultimately show ( $\lambda(x, y). X\ x\ y$ ) - ' S  $\cap$  space  $\Sigma_P \in$  sets  $\Sigma_P$  by simp
    qed
    thus nat-predictable-process M F X by (unfold-locales)
  next
    assume asm: nat-predictable-process M F X
    interpret nat-predictable-process M F X by (rule asm)
    show nat-adapted-process M F ( $\lambda i. X\ (Suc\ i)$ )  $\wedge$  X 0  $\in$  borel-measurable (F 0)
  using adapted-Suc by simp
  qed
end

```

```

theory Martingale
  imports Stochastic-Process Conditional-Expectation-Banach
begin

```

## 7 Martingales

The following locales are necessary for defining martingales.

### 7.1 Additional Locale Definitions

```

locale sigma-finite-adapted-process = sigma-finite-filtered-measure M F t0 + adapted-process
M F t0 X for M F t0 X

```

```

locale nat-sigma-finite-adapted-process = sigma-finite-adapted-process M F 0 :: nat
X for M F X

```

```

locale real-sigma-finite-adapted-process = sigma-finite-adapted-process M F 0 ::
real X for M F X

```

```

sublocale nat-sigma-finite-adapted-process  $\subseteq$  nat-sigma-finite-filtered-measure ..

```

```

sublocale real-sigma-finite-adapted-process  $\subseteq$  real-sigma-finite-filtered-measure ..

```

```

locale finite-adapted-process = finite-filtered-measure M F t0 + adapted-process M
F t0 X for M F t0 X

```

```

sublocale finite-adapted-process  $\subseteq$  sigma-finite-adapted-process ..

```

**locale** *nat-finite-adapted-process* = *finite-adapted-process* *M F 0* :: *nat X* **for** *M F X*  
**locale** *real-finite-adapted-process* = *finite-adapted-process* *M F 0* :: *real X* **for** *M F X*

**sublocale** *nat-finite-adapted-process*  $\subseteq$  *nat-sigma-finite-adapted-process* ..  
**sublocale** *real-finite-adapted-process*  $\subseteq$  *real-sigma-finite-adapted-process* ..

**locale** *sigma-finite-adapted-process-order* = *sigma-finite-adapted-process* *M F t<sub>0</sub> X* **for** *M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  - :: {*order-topology*, *ordered-real-vector*}

**locale** *nat-sigma-finite-adapted-process-order* = *sigma-finite-adapted-process-order* *M F 0* :: *nat X* **for** *M F X*  
**locale** *real-sigma-finite-adapted-process-order* = *sigma-finite-adapted-process-order* *M F 0* :: *real X* **for** *M F X*

**sublocale** *nat-sigma-finite-adapted-process-order*  $\subseteq$  *nat-sigma-finite-adapted-process* ..  
**sublocale** *real-sigma-finite-adapted-process-order*  $\subseteq$  *real-sigma-finite-adapted-process* ..

**locale** *finite-adapted-process-order* = *finite-adapted-process* *M F t<sub>0</sub> X* **for** *M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  - :: {*order-topology*, *ordered-real-vector*}

**locale** *nat-finite-adapted-process-order* = *finite-adapted-process-order* *M F 0* :: *nat X* **for** *M F X*  
**locale** *real-finite-adapted-process-order* = *finite-adapted-process-order* *M F 0* :: *real X* **for** *M F X*

**sublocale** *nat-finite-adapted-process-order*  $\subseteq$  *nat-sigma-finite-adapted-process-order* ..  
**sublocale** *real-finite-adapted-process-order*  $\subseteq$  *real-sigma-finite-adapted-process-order* ..

**locale** *sigma-finite-adapted-process-linorder* = *sigma-finite-adapted-process-order* *M F t<sub>0</sub> X* **for** *M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  - :: {*linorder-topology*}

**locale** *nat-sigma-finite-adapted-process-linorder* = *sigma-finite-adapted-process-linorder* *M F 0* :: *nat X* **for** *M F X*  
**locale** *real-sigma-finite-adapted-process-linorder* = *sigma-finite-adapted-process-linorder* *M F 0* :: *real X* **for** *M F X*

**sublocale** *nat-sigma-finite-adapted-process-linorder*  $\subseteq$  *nat-sigma-finite-adapted-process-order* ..

**sublocale** *real-sigma-finite-adapted-process-linorder*  $\subseteq$  *real-sigma-finite-adapted-process-order*  
 ..

**locale** *finite-adapted-process-linorder* = *finite-adapted-process-order* *M F t<sub>0</sub> X* **for**  
*M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  - :: {*linorder-topology*}

**locale** *nat-finite-adapted-process-linorder* = *finite-adapted-process-linorder* *M F 0*  
 :: *nat X* **for** *M F X*  
**locale** *real-finite-adapted-process-linorder* = *finite-adapted-process-linorder* *M F 0*  
 :: *real X* **for** *M F X*

**sublocale** *nat-finite-adapted-process-linorder*  $\subseteq$  *nat-sigma-finite-adapted-process-linorder*  
 ..  
**sublocale** *real-finite-adapted-process-linorder*  $\subseteq$  *real-sigma-finite-adapted-process-linorder*  
 ..

## 7.2 Martingale

**locale** *martingale* = *sigma-finite-adapted-process* +  
**assumes** *integrable*:  $\bigwedge i. t_0 \leq i \Rightarrow \text{integrable } M (X i)$   
**and** *martingale-property*:  $\bigwedge i j. t_0 \leq i \Rightarrow i \leq j \Rightarrow AE \xi \text{ in } M. X i \xi =$   
*cond-exp M (F i) (X j) \xi*

**locale** *martingale-order* = *martingale* *M F t<sub>0</sub> X* **for** *M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  -  
 :: {*order-topology*, *ordered-real-vector*}  
**locale** *martingale-linorder* = *martingale* *M F t<sub>0</sub> X* **for** *M F t<sub>0</sub>* **and** *X* :: -  $\Rightarrow$  -  $\Rightarrow$  -  
 - :: {*linorder-topology*, *ordered-real-vector*}  
**sublocale** *martingale-linorder*  $\subseteq$  *martingale-order* ..

**lemma** (**in** *sigma-finite-filtered-measure*) *martingale-const-fun*[*intro*]:  
**assumes** *integrable M f f*  $\in$  *borel-measurable (F t<sub>0</sub>)*  
**shows** *martingale M F t<sub>0</sub> (\lambda-. f)*  
**using** *assms sigma-finite-subalgebra.cond-exp-F-meas*[*OF - assms(1)*], *THEN AE-symmetric*  
*borel-measurable-mono*  
**by** (*unfold-locale*) *blast+*

**lemma** (**in** *sigma-finite-filtered-measure*) *martingale-cond-exp*[*intro*]:  
**assumes** *integrable M f*  
**shows** *martingale M F t<sub>0</sub> (\lambda i. cond-exp M (F i) f)*  
**using** *sigma-finite-subalgebra.borel-measurable-cond-exp'* *borel-measurable-cond-exp*  
  
**by** (*unfold-locale*) (*auto intro: sigma-finite-subalgebra.cond-exp-nested-subalg*[*OF*  
 - *assms*]) *simp add: subalgebra-F subalgebras*)

**corollary** (**in** *sigma-finite-filtered-measure*) *martingale-zero*[*intro*]: *martingale M*  
*F t<sub>0</sub> (\lambda-. 0)* **by** *fastforce*

**corollary** (**in** *finite-filtered-measure*) *martingale-const*[*intro*]: *martingale M F t<sub>0</sub>*  
*(\lambda-. c)* **by** *fastforce*



### 7.3 Submartingale

**locale** *submartingale* = *sigma-finite-adapted-process-order* +  
**assumes** *integrable*:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M \ (X \ i)$   
**and** *submartingale-property*:  $\bigwedge i \ j. t_0 \leq i \implies i \leq j \implies AE \ \xi \text{ in } M. X \ i \ \xi \leq$   
*cond-exp*  $M \ (F \ i) \ (X \ j) \ \xi$

**locale** *submartingale-linorder* = *submartingale*  $M \ F \ t_0 \ X$  **for**  $M \ F \ t_0$  **and**  $X :: -$   
 $\Rightarrow - \Rightarrow - :: \{\text{linorder-topology}\}$

**sublocale** *martingale-order*  $\subseteq$  *submartingale* **using** *martingale-property* **by** (*unfold-locales*)  
(*force simp add: integrable*)+  
**sublocale** *martingale-linorder*  $\subseteq$  *submartingale-linorder* ..

### 7.4 Supermartingale

**locale** *supermartingale* = *sigma-finite-adapted-process-order* +  
**assumes** *integrable*:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M \ (X \ i)$   
**and** *supermartingale-property*:  $\bigwedge i \ j. t_0 \leq i \implies i \leq j \implies AE \ \xi \text{ in } M. X \ i \ \xi$   
 $\geq$  *cond-exp*  $M \ (F \ i) \ (X \ j) \ \xi$

**locale** *supermartingale-linorder* = *supermartingale*  $M \ F \ t_0 \ X$  **for**  $M \ F \ t_0$  **and**  $X$   
 $:: - \Rightarrow - \Rightarrow - :: \{\text{linorder-topology}\}$

**sublocale** *martingale-order*  $\subseteq$  *supermartingale* **using** *martingale-property* **by** (*unfold-locales*)  
(*force simp add: integrable*)+  
**sublocale** *martingale-linorder*  $\subseteq$  *supermartingale-linorder* ..

A stochastic process is a martingale, if and only if it is both a submartingale and a supermartingale.

**lemma** *martingale-iff*:

**shows** *martingale*  $M \ F \ t_0 \ X \longleftrightarrow \text{submartingale } M \ F \ t_0 \ X \wedge \text{supermartingale } M \ F \ t_0 \ X$

**proof** (*rule iffI*)

**assume** *asm*: *martingale*  $M \ F \ t_0 \ X$

**interpret** *martingale-order*  $M \ F \ t_0 \ X$  **by** (*intro martingale-order.intro asm*)

**show** *submartingale*  $M \ F \ t_0 \ X \wedge \text{supermartingale } M \ F \ t_0 \ X$  **using** *submartingale-axioms supermartingale-axioms* **by** *blast*

**next**

**assume** *asm*: *submartingale*  $M \ F \ t_0 \ X \wedge \text{supermartingale } M \ F \ t_0 \ X$

**interpret** *submartingale*  $M \ F \ t_0 \ X$  **by** (*simp add: asm*)

**interpret** *supermartingale*  $M \ F \ t_0 \ X$  **by** (*simp add: asm*)

**show** *martingale*  $M \ F \ t_0 \ X$  **using** *submartingale-property supermartingale-property*  
**by** (*unfold-locales*) (*intro integrable, blast, force*)

**qed**

### 7.5 Martingale Lemmas

In the following segment, we cover basic properties of martingales.

**context** *martingale*  
**begin**

**lemma** *cond-exp-diff-eq-zero*:

**assumes**  $t_0 \leq i \leq j$   
**shows**  $AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X j \xi - X i \xi) \xi = 0$   
**using** *martingale-property*[*OF assms*] *assms*  
 $\text{sigma-finite-subalgebra.cond-exp-F-meas}[OF - \text{integrable adapted, of } i]$   
 $\text{sigma-finite-subalgebra.cond-exp-diff}[OF - \text{integrable}(1,1), \text{ of } F i j i]$  **by**  
*fastforce*

**lemma** *set-integral-eq*:

**assumes**  $A \in F i \ t_0 \leq i \leq j$   
**shows**  $\text{set-lebesgue-integral } M A (X i) = \text{set-lebesgue-integral } M A (X j)$   
**proof** –  
**interpret** *sigma-finite-subalgebra*  $M F i$  **using** *assms*(2) **by** *blast*  
**have**  $\int x \in A. X i x \ \partial M = \int x \in A. \text{cond-exp } M (F i) (X j) x \ \partial M$  **using**  
*martingale-property*[*OF assms*(2,3)] *borel-measurable-cond-exp'* *assms* *subalgebras*  
*subalgebra-def* **by** (*intro set-lebesgue-integral-cong-AE*[*OF - random-variable*]) *fast-*  
*force*+  
**also have**  $\dots = \int x \in A. X j x \ \partial M$  **using** *assms* **by** (*auto simp: integrable intro:*  
*cond-exp-set-integral[symmetric]*)  
**finally show** *?thesis* .  
**qed**

**lemma** *scaleR-const*[*intro*]:

**shows** *martingale*  $M F t_0 (\lambda i x. c *_{\mathbb{R}} X i x)$   
**proof** –  
 $\{$   
**fix**  $i j :: 'b$  **assume** *asm*:  $t_0 \leq i \leq j$   
**interpret** *sigma-finite-subalgebra*  $M F i$  **using** *asm* **by** *blast*  
**have**  $AE x \text{ in } M. c *_{\mathbb{R}} X i x = \text{cond-exp } M (F i) (\lambda x. c *_{\mathbb{R}} X j x) x$  **using**  
*asm* *cond-exp-scaleR-right*[*OF integrable, of j, THEN AE-symmetric*] *martin-*  
*gale-property*[*OF asm*] **by** *force*  
 $\}$   
**thus** *?thesis* **by** (*unfold-locales*) (*auto simp add: integrable martingale.integrable*)  
**qed**

**lemma** *uminus*[*intro*]:

**shows** *martingale*  $M F t_0 (- X)$   
**using** *scaleR-const*[*of -1*] **by** (*force intro: back-subst*[*of martingale M F t\_0*])

**lemma** *add*[*intro*]:

**assumes** *martingale*  $M F t_0 Y$   
**shows** *martingale*  $M F t_0 (\lambda i \xi. X i \xi + Y i \xi)$   
**proof** –  
**interpret**  $Y: \text{martingale } M F t_0 Y$  **by** (*rule assms*)  
 $\{$   
**fix**  $i j :: 'b$  **assume** *asm*:  $t_0 \leq i \leq j$

```

    hence AE  $\xi$  in  $M$ .  $X\ i\ \xi + Y\ i\ \xi = \text{cond-exp } M\ (F\ i)\ (\lambda x. X\ j\ x + Y\ j\ x)\ \xi$ 
    using sigma-finite-subalgebra.cond-exp-add[OF - integrable martingale.integrable[OF
assms], of  $F\ i\ j\ j$ , THEN AE-symmetric]
    martingale-property[OF asm] martingale.martingale-property[OF assms
asm] by force
  }
  thus ?thesis using assms
  by (unfold-locales) (auto simp add: integrable martingale.integrable)
qed

```

```

lemma diff[intro]:
  assumes martingale  $M\ F\ t_0\ Y$ 
  shows martingale  $M\ F\ t_0\ (\lambda i\ x. X\ i\ x - Y\ i\ x)$ 
proof -
  interpret  $Y$ : martingale  $M\ F\ t_0\ Y$  by (rule assms)
  {
    fix  $i\ j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
    hence AE  $\xi$  in  $M$ .  $X\ i\ \xi - Y\ i\ \xi = \text{cond-exp } M\ (F\ i)\ (\lambda x. X\ j\ x - Y\ j\ x)\ \xi$ 
    using sigma-finite-subalgebra.cond-exp-diff[OF - integrable martingale.integrable[OF
assms], of  $F\ i\ j\ j$ , THEN AE-symmetric]
    martingale-property[OF asm] martingale.martingale-property[OF assms
asm] by fastforce
  }
  thus ?thesis using assms by (unfold-locales) (auto simp add: integrable martin-
gale.integrable)
qed

```

end

Using properties of the conditional expectation, we present the following alternative characterizations of martingales.

```

lemma (in sigma-finite-adapted-process) martingale-of-cond-exp-diff-eq-zero:
  assumes integrable:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M\ (X\ i)$ 
  and diff-zero:  $\bigwedge i\ j. t_0 \leq i \implies i \leq j \implies \text{AE } x \text{ in } M. \text{cond-exp } M\ (F\ i)\ (\lambda \xi. X\ j\ \xi - X\ i\ \xi)\ x = 0$ 
  shows martingale  $M\ F\ t_0\ X$ 
proof
  {
    fix  $i\ j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
    thus AE  $\xi$  in  $M$ .  $X\ i\ \xi = \text{cond-exp } M\ (F\ i)\ (X\ j)\ \xi$ 
    using diff-zero[OF asm] sigma-finite-subalgebra.cond-exp-diff[OF - integrable(1,1),
of  $F\ i\ j\ i$ ]
    sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable adapted, of  $i$ ] by
fastforce
  }
qed (intro integrable)

```

```

lemma (in sigma-finite-adapted-process) martingale-of-set-integral-eq:
  assumes integrable:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M\ (X\ i)$ 

```

**and**  $\bigwedge A \ i \ j. \ t_0 \leq i \implies i \leq j \implies A \in F \ i \implies \text{set-lebesgue-integral } M \ A \ (X \ i) = \text{set-lebesgue-integral } M \ A \ (X \ j)$   
**shows** *martingale*  $M \ F \ t_0 \ X$   
**proof** (*unfold-locales*)  
**fix**  $i \ j :: 'b$  **assume**  $asm: t_0 \leq i \ i \leq j$   
**interpret** *sigma-finite-subalgebra*  $M \ F \ i$  **using**  $asm$  **by** *blast*  
**interpret**  $r: \text{sigma-finite-measure restr-to-subalg } M \ (F \ i)$  **by** (*simp add: sigma-fin-subalg*)  
**{**  
**fix**  $A$  **assume**  $A \in \text{restr-to-subalg } M \ (F \ i)$   
**hence**  $*$ :  $A \in F \ i$  **using** *sets-restr-to-subalg subalgebras*  $asm$  **by** *blast*  
**have**  $\text{set-lebesgue-integral } (\text{restr-to-subalg } M \ (F \ i)) \ A \ (X \ i) = \text{set-lebesgue-integral } M \ A \ (X \ i)$  **using**  $*$  *subalg*  $asm$  **by** (*auto simp: set-lebesgue-integral-def intro: integral-subalgebra2 borel-measurable-scaleR adapted borel-measurable-indicator*)  
**also** **have**  $\dots = \text{set-lebesgue-integral } M \ A \ (\text{cond-exp } M \ (F \ i) \ (X \ j))$  **using**  $*$  *assms(2)[OF asm] cond-exp-set-integral[OF integrable] asm* **by** *auto*  
**finally** **have**  $\text{set-lebesgue-integral } (\text{restr-to-subalg } M \ (F \ i)) \ A \ (X \ i) = \text{set-lebesgue-integral } (\text{restr-to-subalg } M \ (F \ i)) \ A \ (\text{cond-exp } M \ (F \ i) \ (X \ j))$  **using**  $*$  *subalg* **by** (*auto simp: set-lebesgue-integral-def intro!: integral-subalgebra2[symmetric] borel-measurable-scaleR borel-measurable-cond-exp borel-measurable-indicator*)  
**}**  
**hence**  $AE \ \xi \text{ in } \text{restr-to-subalg } M \ (F \ i). \ X \ i \ \xi = \text{cond-exp } M \ (F \ i) \ (X \ j) \ \xi$  **using**  $asm$  **by** (*intro r.density-unique-banach, auto intro: integrable-in-subalg subalg borel-measurable-cond-exp integrable*)  
**thus**  $AE \ \xi \text{ in } M. \ X \ i \ \xi = \text{cond-exp } M \ (F \ i) \ (X \ j) \ \xi$  **using** *AE-restr-to-subalg[OF subalg]* **by** *blast*  
**qed** (*simp add: integrable*)

## 7.6 Submartingale Lemmas

**context** *submartingale*  
**begin**

**lemma** *cond-exp-diff-nonneg*:  
**assumes**  $t_0 \leq i \ i \leq j$   
**shows**  $AE \ x \text{ in } M. \ \text{cond-exp } M \ (F \ i) \ (\lambda \xi. \ X \ j \ \xi - X \ i \ \xi) \ x \geq 0$   
**using** *submartingale-property[OF assms] assms sigma-finite-subalgebra.cond-exp-diff[OF - integrable(1,1), of - j i] sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable adapted, of i]* **by** *fastforce*

**lemma** *add[intro]*:  
**assumes** *submartingale*  $M \ F \ t_0 \ Y$   
**shows** *submartingale*  $M \ F \ t_0 \ (\lambda i \ \xi. \ X \ i \ \xi + Y \ i \ \xi)$   
**proof** –  
**interpret**  $Y: \text{submartingale } M \ F \ t_0 \ Y$  **by** (*rule assms*)  
**{**  
**fix**  $i \ j :: 'b$  **assume**  $asm: t_0 \leq i \ i \leq j$   
**hence**  $AE \ \xi \text{ in } M. \ X \ i \ \xi + Y \ i \ \xi \leq \text{cond-exp } M \ (F \ i) \ (\lambda x. \ X \ j \ x + Y \ j \ x) \ \xi$   
**using** *sigma-finite-subalgebra.cond-exp-add[OF - integrable submartingale.integrable[OF assms], of F i j j]*

```

      submartingale-property[OF asm] submartingale.submartingale-property[OF
assms asm] add-mono[of  $X\ i - -\ Y\ i$ ] by force
    }
    thus ?thesis using assms by (unfold-locale) (auto simp add: borel-measurable-add
random-variable adapted integrable  $Y$ .random-variable  $Y$ .adapted submartingale.integrable)

```

qed

**lemma** *diff[intro]*:

```

  assumes supermartingale  $M\ F\ t_0\ Y$ 
  shows submartingale  $M\ F\ t_0\ (\lambda i\ \xi. X\ i\ \xi - Y\ i\ \xi)$ 
proof -
  interpret  $Y$ : supermartingale  $M\ F\ t_0\ Y$  by (rule assms)
  {
    fix  $i\ j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
    hence  $AE\ \xi\ in\ M. X\ i\ \xi - Y\ i\ \xi \leq cond-exp\ M\ (F\ i)\ (\lambda x. X\ j\ x - Y\ j\ x)\ \xi$ 
    using sigma-finite-subalgebra.cond-exp-diff[OF - integrable supermartingale.integrable[OF
assms], of  $F\ i\ j$ ]
      submartingale-property[OF asm] supermartingale.supermartingale-property[OF
assms asm] diff-mono[of  $X\ i - -\ Y\ i$ ] by force
  }
  thus ?thesis using assms by (unfold-locale) (auto simp add: borel-measurable-diff
random-variable adapted integrable  $Y$ .random-variable  $Y$ .adapted supermartingale.integrable)

```

qed

**lemma** *scaleR-nonneg*:

```

  assumes  $c \geq 0$ 
  shows submartingale  $M\ F\ t_0\ (\lambda i\ \xi. c *_{\mathbb{R}} X\ i\ \xi)$ 
proof
  {
    fix  $i\ j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
    thus  $AE\ \xi\ in\ M. c *_{\mathbb{R}} X\ i\ \xi \leq cond-exp\ M\ (F\ i)\ (\lambda \xi. c *_{\mathbb{R}} X\ j\ \xi)\ \xi$ 
    using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of  $F\ i\ j\ c$ ] submartingale-property[OF asm] by (fastforce intro!: scaleR-left-mono[OF -
assms])
  }
qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

```

**lemma** *scaleR-le-zero*:

```

  assumes  $c \leq 0$ 
  shows supermartingale  $M\ F\ t_0\ (\lambda i\ \xi. c *_{\mathbb{R}} X\ i\ \xi)$ 
proof
  {
    fix  $i\ j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
    thus  $AE\ \xi\ in\ M. c *_{\mathbb{R}} X\ i\ \xi \geq cond-exp\ M\ (F\ i)\ (\lambda \xi. c *_{\mathbb{R}} X\ j\ \xi)\ \xi$ 
    using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of  $F\ i\ j\ c$ ] submartingale-property[OF asm]

```

```

      by (fastforce intro!: scaleR-left-mono-neg[OF - assms])
    }
  qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
    random-variable adapted borel-measurable-const-scaleR)

lemma uminus[intro]:
  shows supermartingale M F t0 (− X)
  unfolding fun-Compl-def using scaleR-le-zero[of − 1] by simp

end

context submartingale-linorder
begin

lemma set-integral-le:
  assumes A ∈ F i t0 ≤ i i ≤ j
  shows set-lebesgue-integral M A (X i) ≤ set-lebesgue-integral M A (X j)
  using submartingale-property[OF assms(2), of j] assms subalgebras
  by (subst sigma-finite-subalgebra.cond-exp-set-integral[OF - integrable assms(1),
    of j])
  (auto intro!: scaleR-left-mono integral-mono-AE-banach integrable-mult-indicator
    integrable simp add: subalgebra-def set-lebesgue-integral-def)

lemma max:
  assumes submartingale-linorder M F t0 Y
  shows submartingale-linorder M F t0 (λi ξ. max (X i ξ) (Y i ξ))
  proof (unfold-locales)
    interpret Y: submartingale-linorder M F t0 Y by (rule assms)
    {
      fix i j :: 'b assume asm: t0 ≤ i i ≤ j
      have AE ξ in M. max (X i ξ) (Y i ξ) ≤ max (cond-exp M (F i) (X j) ξ)
        (cond-exp M (F i) (Y j) ξ) using submartingale-property Y.submartingale-property
      asm unfolding max-def by fastforce
      thus AE ξ in M. max (X i ξ) (Y i ξ) ≤ cond-exp M (F i) (λξ. max (X j ξ) (Y
        j ξ)) ξ using sigma-finite-subalgebra.cond-exp-max[OF - integrable Y.integrable, of
        F i j j] asm by (fast intro: order.trans)
    }
    show ∧i. t0 ≤ i ⇒ (λξ. max (X i ξ) (Y i ξ)) ∈ borel-measurable (F i) ∧i.
      t0 ≤ i ⇒ integrable M (λξ. max (X i ξ) (Y i ξ)) by (force intro: Y.integrable
      integrable assms)+
  qed

lemma max-0:
  shows submartingale-linorder M F t0 (λi ξ. max 0 (X i ξ))
  proof −
    interpret zero: martingale-linorder M F t0 λ- -. 0 by (force intro: martin-
      gale-linorder.intro martingale-order.intro)
    show ?thesis by (intro zero.max submartingale-linorder.intro submartingale-axioms)
  qed

```

end

**lemma** (in *sigma-finite-adapted-process-order*) *submartingale-of-cond-exp-diff-nonneg*:  
**assumes** *integrable*:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$   
**and** *diff-nonneg*:  $\bigwedge i j. t_0 \leq i \implies i \leq j \implies \text{AE } x \text{ in } M. \text{ cond-exp } M (F i)$   
 $(\lambda \xi. X j \xi - X i \xi) x \geq 0$   
**shows** *submartingale*  $M F t_0 X$   
**proof** (*unfold-locales*)  
{  
  **fix**  $i j :: 'b$  **assume** *asm*:  $t_0 \leq i \leq j$   
  **thus**  $\text{AE } \xi \text{ in } M. X i \xi \leq \text{cond-exp } M (F i) (X j) \xi$   
  **using** *diff-nonneg*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable(1,1), of F i j i*]  
  *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable adapted, of i*] **by**  
*fastforce*  
}  
**qed** (*intro integrable*)

**lemma** (in *sigma-finite-adapted-process-linorder*) *submartingale-of-set-integral-le*:  
**assumes** *integrable*:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$   
**and**  $\bigwedge A i j. t_0 \leq i \implies i \leq j \implies A \in F i \implies \text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X j)$   
**shows** *submartingale*  $M F t_0 X$   
**proof** (*unfold-locales*)  
{  
  **fix**  $i j :: 'b$  **assume** *asm*:  $t_0 \leq i \leq j$   
  **interpret** *r*: *sigma-finite-measure restr-to-subalg*  $M (F i)$  **using** *asm sigma-finite-subalgebra.sigma-fin-subalg*  
**by** *blast*  
  {  
    **fix**  $A$  **assume**  $A \in \text{restr-to-subalg } M (F i)$   
    **hence**  $*$ :  $A \in F i$  **using** *asm sets-restr-to-subalg subalgebras* **by** *blast*  
    **have**  $\text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (X i) = \text{set-lebesgue-integral } M A (X i)$  **using**  $*$  *asm subalgebras* **by** (*auto simp: set-lebesgue-integral-def intro: integral-subalgebra2 borel-measurable-scaleR adapted borel-measurable-indicator*)  
    **also have**  $\dots \leq \text{set-lebesgue-integral } M A (\text{cond-exp } M (F i) (X j))$  **using**  
 $*$  *assms(2)*[*OF asm*] *asm sigma-finite-subalgebra.cond-exp-set-integral*[*OF - integrable*] **by** *fastforce*  
    **also have**  $\dots = \text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (\text{cond-exp } M (F i) (X j))$  **using**  $*$  *asm subalgebras* **by** (*auto simp: set-lebesgue-integral-def intro!: integral-subalgebra2[symmetric] borel-measurable-scaleR borel-measurable-cond-exp borel-measurable-indicator*)  
    **finally have**  $0 \leq \text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (\lambda \xi. \text{cond-exp } M (F i) (X j) \xi - X i \xi)$  **using**  $*$  *asm subalgebras* **by** (*subst set-integral-diff, auto simp add: set-integrable-def sets-restr-to-subalg intro!: integrable adapted integrable-in-subalg borel-measurable-scaleR borel-measurable-indicator borel-measurable-cond-exp integrable-mult-indicator*)  
  }  
  **hence**  $\text{AE } \xi \text{ in } \text{restr-to-subalg } M (F i). 0 \leq \text{cond-exp } M (F i) (X j) \xi - X i \xi$

```

    by (intro r.density-nonneg integrable-in-subalg asm subalgebras borel-measurable-diff
borel-measurable-cond-exp adapted Bochner-Integration.integrable-diff integrable-cond-exp
integrable)
    thus  $\text{AE } \xi \text{ in } M. X \ i \ \xi \leq \text{cond-exp } M \ (F \ i) \ (X \ j) \ \xi$  using  $\text{AE-restr-to-subalg}[OF$ 
 $\text{subalgebras}] \text{ asm by simp}$ 
  }
qed (intro integrable)

```

## 7.7 Supermartingale Lemmas

The following lemmas are exact duals of the ones for submartingales.

```

context supermartingale
begin

```

```

lemma cond-exp-diff-nonneg:
  assumes  $t_0 \leq i \leq j$ 
  shows  $\text{AE } x \text{ in } M. \text{cond-exp } M \ (F \ i) \ (\lambda \xi. X \ i \ \xi - X \ j \ \xi) \ x \geq 0$ 
  using  $\text{assms supermartingale-property}[OF \ \text{assms}] \ \text{sigma-finite-subalgebra.cond-exp-diff}[OF$ 
 $- \text{integrable}(1,1), \text{ of } F \ i \ i \ j]$ 
     $\text{sigma-finite-subalgebra.cond-exp-F-meas}[OF - \text{integrable adapted, of } i]$  by
fastforce

```

```

lemma add[intro]:
  assumes  $\text{supermartingale } M \ F \ t_0 \ Y$ 
  shows  $\text{supermartingale } M \ F \ t_0 \ (\lambda i \ \xi. X \ i \ \xi + Y \ i \ \xi)$ 
proof -
  interpret  $Y: \text{supermartingale } M \ F \ t_0 \ Y$  by (rule assms)
  {
    fix  $i \ j :: 'b$  assume  $\text{asm}: t_0 \leq i \leq j$ 
    hence  $\text{AE } \xi \text{ in } M. X \ i \ \xi + Y \ i \ \xi \geq \text{cond-exp } M \ (F \ i) \ (\lambda x. X \ j \ x + Y \ j \ x) \ \xi$ 
    using  $\text{sigma-finite-subalgebra.cond-exp-add}[OF - \text{integrable supermartingale.integrable}[OF$ 
 $\text{assms}], \text{ of } F \ i \ j]$ 
       $\text{supermartingale-property}[OF \ \text{asm}] \ \text{supermartingale.supermartingale-property}[OF$ 
 $\text{assms } \text{asm}] \ \text{add-mono}[\text{of } - \ X \ i \ - \ Y \ i \ -]$  by force
  }
  thus ?thesis using  $\text{assms by (unfold-locales) (auto simp add: borel-measurable-add$ 
 $\text{random-variable adapted integrable } Y.\text{random-variable } Y.\text{adapted supermartingale.integrable})$ 
qed

```

```

lemma diff[intro]:
  assumes  $\text{submartingale } M \ F \ t_0 \ Y$ 
  shows  $\text{supermartingale } M \ F \ t_0 \ (\lambda i \ \xi. X \ i \ \xi - Y \ i \ \xi)$ 
proof -
  interpret  $Y: \text{submartingale } M \ F \ t_0 \ Y$  by (rule assms)
  {
    fix  $i \ j :: 'b$  assume  $\text{asm}: t_0 \leq i \leq j$ 
    hence  $\text{AE } \xi \text{ in } M. X \ i \ \xi - Y \ i \ \xi \geq \text{cond-exp } M \ (F \ i) \ (\lambda x. X \ j \ x - Y \ j \ x) \ \xi$ 
    using  $\text{sigma-finite-subalgebra.cond-exp-diff}[OF - \text{integrable submartingale.integrable}[OF$ 

```



```

assms], of F i j j, unfolded fun-diff-def]
      supermartingale-property[OF asm] submartingale.submartingale-property[OF
assms asm] diff-mono[of - X i - Y i -] by force
    }
    thus ?thesis using assms by (unfold-locates) (auto simp add: borel-measurable-diff
random-variable adapted integrable Y.random-variable Y.adapted submartingale.integrable)

```

qed

**lemma** *scaleR-nonneg*:

```

assumes c ≥ 0
shows supermartingale M F t0 (λi ξ. c *R X i ξ)
proof
  {
    fix i j :: 'b assume asm: t0 ≤ i i ≤ j
    thus AE ξ in M. c *R X i ξ ≥ cond-exp M (F i) (λξ. c *R X j ξ) ξ
      using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i
j c] supermartingale-property[OF asm] by (fastforce intro!: scaleR-left-mono[OF -
assms])
  }
qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

```

**lemma** *scaleR-le-zero*:

```

assumes c ≤ 0
shows submartingale M F t0 (λi ξ. c *R X i ξ)
proof
  {
    fix i j :: 'b assume asm: t0 ≤ i i ≤ j
    thus AE ξ in M. c *R X i ξ ≤ cond-exp M (F i) (λξ. c *R X j ξ) ξ
      using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i j c]
supermartingale-property[OF asm] by (fastforce intro!: scaleR-left-mono-neg[OF -
assms])
  }
qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

```

**lemma** *uminus[intro]*:

```

shows submartingale M F t0 (- X)
unfolding fun-Compl-def using scaleR-le-zero[of -1] by simp

```

end

**context** *supermartingale-linorder*

**begin**

**lemma** *set-integral-ge*:

```

assumes A ∈ F i t0 ≤ i i ≤ j
shows set-lebesgue-integral M A (X i) ≥ set-lebesgue-integral M A (X j)

```

**using** *supermartingale-property*[*OF* *assms*(2), *of j*] *assms subalgebras*  
**by** (*subst sigma-finite-subalgebra.cond-exp-set-integral*[*OF* - *integrable assms*(1),  
*of j*])  
(*auto intro!*: *scaleR-left-mono integral-mono-AE-banach integrable-mult-indicator*  
*integrable simp add: subalgebra-def set-lebesgue-integral-def*)

**lemma** *min*:

**assumes** *supermartingale-linorder* *M F t<sub>0</sub> Y*  
**shows** *supermartingale-linorder* *M F t<sub>0</sub> (λ i ξ. min (X i ξ) (Y i ξ))*  
**proof** (*unfold-locales*)  
**interpret** *Y: supermartingale-linorder* *M F t<sub>0</sub> Y* **by** (*rule assms*)  
{  
  **fix** *i j :: 'b* **assume** *asm: t<sub>0</sub> ≤ i i ≤ j*  
  **have** *AE ξ in M. min (X i ξ) (Y i ξ) ≥ min (cond-exp M (F i) (X j) ξ) (cond-exp*  
*M (F i) (Y j) ξ)* **using** *supermartingale-property Y.supermartingale-property asm*  
**unfolding** *min-def* **by** *fastforce*  
  **thus** *AE ξ in M. min (X i ξ) (Y i ξ) ≥ cond-exp M (F i) (λ ξ. min (X j ξ) (Y*  
*j ξ)) ξ* **using** *sigma-finite-subalgebra.cond-exp-min*[*OF* - *integrable Y.integrable, of*  
*F i j j*] *asm* **by** (*fast intro: order.trans*)  
}  
**show**  $\bigwedge i. t_0 \leq i \implies (\lambda \xi. \min (X i \xi) (Y i \xi)) \in \text{borel-measurable } (F i) \bigwedge i.$   
 $t_0 \leq i \implies \text{integrable } M (\lambda \xi. \min (X i \xi) (Y i \xi))$  **by** (*force intro: Y.integrable*  
*integrable assms*)+  
**qed**

**lemma** *min-0*:

**shows** *supermartingale-linorder* *M F t<sub>0</sub> (λ i ξ. min 0 (X i ξ))*  
**proof** –  
**interpret** *zero: martingale-linorder* *M F t<sub>0</sub> λ- -. 0* **by** (*force intro: martin-*  
*gale-linorder.intro*)  
**show** *?thesis* **by** (*intro zero.min supermartingale-linorder.intro supermartin-*  
*gale-axioms*)  
**qed**

**end**

**lemma** (*in sigma-finite-adapted-process-order*) *supermartingale-of-cond-exp-diff-le-zero*:

**assumes** *integrable:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$*   
**and** *diff-le-zero:  $\bigwedge i j. t_0 \leq i \implies i \leq j \implies \text{AE } x \text{ in } M. \text{ cond-exp } M (F i)$*   
 $(\lambda \xi. X j \xi - X i \xi) x \leq 0$   
**shows** *supermartingale* *M F t<sub>0</sub> X*  
**proof**  
{  
  **fix** *i j :: 'b* **assume** *asm: t<sub>0</sub> ≤ i i ≤ j*  
  **thus** *AE ξ in M. X i ξ ≥ cond-exp M (F i) (X j) ξ*  
  **using** *diff-le-zero*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF* - *inte-*  
*grable(1,1), of F i j i*]  
  *sigma-finite-subalgebra.cond-exp-F-meas*[*OF* - *integrable adapted, of i*] **by**  
*fastforce*

}  
qed (intro integrable)

**lemma** (in *sigma-finite-adapted-process-linorder*) *supermartingale-of-set-integral-ge*:  
 assumes *integrable*:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M \ (X \ i)$   
 and  $\bigwedge A \ i \ j. t_0 \leq i \implies i \leq j \implies A \in F \ i \implies \text{set-lebesgue-integral } M \ A \ (X \ j) \leq \text{set-lebesgue-integral } M \ A \ (X \ i)$   
 shows *supermartingale*  $M \ F \ t_0 \ X$   
**proof** –  
 interpret -: *adapted-process*  $M \ F \ t_0 - X$  by (rule *uminus-adapted*)  
 interpret *uminus-X*: *sigma-finite-adapted-process-linorder*  $M \ F \ t_0 - X$  ..  
 note  $\ast = \text{set-integral-uminus}[\text{unfolded set-integrable-def}, \text{OF integrable-mult-indicator}[\text{OF} - \text{integrable}]]$   
 have *supermartingale*  $M \ F \ t_0 \ (-(- X))$   
 using *ord-eq-le-trans*  $[\text{OF} \ast \text{ord-le-eq-trans}[\text{OF le-imp-neg-le}[\text{OF assms}(2)] \ast [\text{symmetric}]]]$   
*subalgebras*  
 by (intro *submartingale.uminus uminus-X.submartingale-of-set-integral-le*)  
 (clarsimp simp add: *fun-Compl-def subalgebra-def integrable | fastforce*) +  
 thus ?thesis unfolding *fun-Compl-def* by simp  
 qed

Many of the statements we have made concerning martingales can be simplified when the indexing set is the natural numbers. Given a point in time  $i \in \mathbb{N}$ , it suffices to consider the successor  $i + (1::'a)$ , instead of all future times  $i \leq j$ .

## 7.8 Discrete Time Martingales

**locale** *nat-martingale* = *martingale*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
**locale** *nat-submartingale* = *submartingale*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
**locale** *nat-supermartingale* = *supermartingale*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
  
**locale** *nat-submartingale-linorder* = *submartingale-linorder*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
**locale** *nat-supermartingale-linorder* = *supermartingale-linorder*  $M \ F \ 0 :: \text{nat } X$  **for**  $M \ F \ X$   
  
**sublocale** *nat-submartingale-linorder*  $\subseteq$  *nat-submartingale* ..  
**sublocale** *nat-supermartingale-linorder*  $\subseteq$  *nat-supermartingale* ..

A predictable martingale is necessarily constant.

**lemma** (in *nat-martingale*) *predictable-const*:  
 assumes *nat-predictable-process*  $M \ F \ X$   
 shows  $\text{AE } \xi \text{ in } M. X \ i \ \xi = X \ j \ \xi$   
**proof** –  
 have  $\ast: \text{AE } \xi \text{ in } M. X \ i \ \xi = X \ 0 \ \xi$  **for**  $i$   
**proof** (*induction i*)  
 case 0  
 then show ?case by (*simp add: bot-nat-def*)

```

next
  case (Suc i)
  interpret S: nat-adapted-process M F  $\lambda i. X (Suc i)$  by (intro nat-predictable-process.adapted-Suc
    assms)
  show ?case using Suc S.adapted[of i] martingale-property[OF - le-SucI, of i]
    sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i] by fastforce
  qed
  show ?thesis using *[of i] *[of j] by force
qed

lemma (in nat-sigma-finite-adapted-process) martingale-of-set-integral-eq-Suc:
  assumes integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
  and  $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) = \text{set-lebesgue-integral}$ 
     $M A (X (Suc i))$ 
  shows nat-martingale M F X
proof (intro nat-martingale.intro martingale-of-set-integral-eq)
  fix i j A assume asm:  $i \leq j \wedge A \in \text{sets } (F i)$ 
  show  $\text{set-lebesgue-integral } M A (X i) = \text{set-lebesgue-integral } M A (X j)$  using
    asm
  proof (induction j - i arbitrary: i j)
    case 0
    then show ?case using asm by simp
  next
    case (Suc n)
    hence *:  $n = j - Suc i$  by linarith
    have  $Suc i \leq j$  using Suc(2,3) by linarith
    thus ?case using sets-F-mono[OF - le-SucI] Suc(4) Suc(1)[OF *] by (auto
      intro: assms(2)[THEN trans])
  qed
qed (simp add: integrable)

lemma (in nat-sigma-finite-adapted-process) martingale-nat:
  assumes integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
  and  $\bigwedge i. AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X (Suc i)) \xi$ 
  shows nat-martingale M F X
proof (unfold-locale)
  fix i j :: nat assume asm:  $i \leq j$ 
  show  $AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X j) \xi$  using asm
  proof (induction j - i arbitrary: i j)
    case 0
    hence  $j = i$  by simp
    thus ?case using sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable adapted,
      THEN AE-symmetric] by blast
  next
    case (Suc n)
    have j:  $j = Suc (n + i)$  using Suc by linarith
    have n:  $n = n + i - i$  using Suc by linarith
    have *:  $AE \xi \text{ in } M. \text{cond-exp } M (F (n + i)) (X j) \xi = X (n + i) \xi$  unfolding
      j using assms(2)[THEN AE-symmetric] by blast
  qed

```

**have**  $AE \xi$  in  $M$ .  $cond\text{-}exp\ M\ (F\ i)\ (X\ j)\ \xi = cond\text{-}exp\ M\ (F\ i)\ (cond\text{-}exp\ M\ (F\ (n + i))\ (X\ j))\ \xi$  **by** (*intro cond-exp-nested-subalg integrable subalg, simp add: subalgebra-def sets-F-mono*)  
**hence**  $AE \xi$  in  $M$ .  $cond\text{-}exp\ M\ (F\ i)\ (X\ j)\ \xi = cond\text{-}exp\ M\ (F\ i)\ (X\ (n + i))\ \xi$  **using** *cond-exp-cong-AE[OF integrable-cond-exp integrable \*]* **by** *force*  
**thus**  $?case$  **using** *Suc(1)[OF n]* **by** *fastforce*  
**qed**  
**qed** (*simp add: integrable*)

**lemma** (*in nat-sigma-finite-adapted-process*) *martingale-of-cond-exp-diff-Suc-eq-zero*:  
**assumes** *integrable:  $\bigwedge i$ . integrable  $M\ (X\ i)$*   
**and**  $\bigwedge i$ .  $AE \xi$  in  $M$ .  $cond\text{-}exp\ M\ (F\ i)\ (\lambda \xi. X\ (Suc\ i)\ \xi - X\ i\ \xi)\ \xi = 0$   
**shows** *nat-martingale  $M\ F\ X$*   
**proof** (*intro martingale-nat integrable*)  
**fix**  $i$   
**show**  $AE \xi$  in  $M$ .  $X\ i\ \xi = cond\text{-}exp\ M\ (F\ i)\ (X\ (Suc\ i))\ \xi$  **using** *cond-exp-diff[OF integrable(1,1), of i Suc i i] cond-exp-F-meas[OF integrable adapted, of i] assms(2)[of i]* **by** *fastforce*  
**qed**

## 7.9 Discrete Time Submartingales

**lemma** (*in nat-submartingale*) *predictable-mono*:  
**assumes** *nat-predictable-process  $M\ F\ X\ i \leq j$*   
**shows**  $AE \xi$  in  $M$ .  $X\ i\ \xi \leq X\ j\ \xi$   
**using** *assms(2)*  
**proof** (*induction  $j - i$  arbitrary:  $i\ j$* )  
**case** 0  
**then show**  $?case$  **by** *simp*  
**next**  
**case** (*Suc n*)  
**hence**  $*$ :  $n = j - Suc\ i$  **by** *linarith*  
**interpret**  $S$ : *nat-adapted-process  $M\ F\ \lambda i. X\ (Suc\ i)$*  **by** (*intro nat-predictable-process.adapted-Suc assms*)  
**have**  $Suc\ i \leq j$  **using** *Suc(2,3)* **by** *linarith*  
**thus**  $?case$  **using** *Suc(1)[OF \*] S.adapted[of i] submartingale-property[OF - le-SucI, of i] sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i]* **by** *fastforce*  
**qed**

**lemma** (*in nat-sigma-finite-adapted-process-linorder*) *submartingale-of-set-integral-le-Suc*:  
**assumes** *integrable:  $\bigwedge i$ . integrable  $M\ (X\ i)$*   
**and**  $\bigwedge A\ i$ .  $A \in F\ i \implies set\text{-lebesgue-integral}\ M\ A\ (X\ i) \leq set\text{-lebesgue-integral}\ M\ A\ (X\ (Suc\ i))$   
**shows** *nat-submartingale  $M\ F\ X$*   
**proof** (*intro nat-submartingale.intro submartingale-of-set-integral-le*)  
**fix**  $i\ j\ A$  **assume** *asm:  $i \leq j\ A \in sets\ (F\ i)$*   
**show**  $set\text{-lebesgue-integral}\ M\ A\ (X\ i) \leq set\text{-lebesgue-integral}\ M\ A\ (X\ j)$  **using** *asm*

**proof** (*induction*  $j - i$  *arbitrary*:  $i$   $j$ )  
**case** 0  
**then show** ?*case* **using** *asm* **by** *simp*  
**next**  
**case** (*Suc*  $n$ )  
**hence** \*:  $n = j - \text{Suc } i$  **by** *linarith*  
**have**  $\text{Suc } i \leq j$  **using** *Suc*(2,3) **by** *linarith*  
**thus** ?*case* **using** *sets-F-mono*[*OF* - *le-SucI*] *Suc*(4) *Suc*(1)[*OF* \*] **by** (*auto*  
*intro: assms*(2)[*THEN order-trans*])  
**qed**  
**qed** (*simp add: integrable*)

**lemma** (*in nat-sigma-finite-adapted-process-linorder*) *submartingale-nat*:  
**assumes** *integrable*:  $\bigwedge i. \text{integrable } M (X \ i)$   
**and**  $\bigwedge i. AE \ \xi \text{ in } M. X \ i \ \xi \leq \text{cond-exp } M (F \ i) (X (\text{Suc } i)) \ \xi$   
**shows** *nat-submartingale*  $M \ F \ X$   
**using** *subalg integrable assms*(2)  
**by** (*intro submartingale-of-set-integral-le-Suc ord-le-eq-trans*[*OF set-integral-mono-AE-banach*  
*cond-exp-set-integral*[*symmetric*]], *simp*)  
*(meson in-mono integrable-mult-indicator set-integrable-def subalgebra-def, me-*  
*son integrable-cond-exp in-mono integrable-mult-indicator set-integrable-def subal-*  
*gebra-def, fast+)*

**lemma** (*in nat-sigma-finite-adapted-process-linorder*) *submartingale-of-cond-exp-diff-Suc-nonneg*:  
**assumes** *integrable*:  $\bigwedge i. \text{integrable } M (X \ i)$   
**and**  $\bigwedge i. AE \ \xi \text{ in } M. \text{cond-exp } M (F \ i) (\lambda \xi. X (\text{Suc } i) \ \xi - X \ i \ \xi) \ \xi \geq 0$   
**shows** *nat-submartingale*  $M \ F \ X$   
**proof** (*intro submartingale-nat integrable*)  
**fix**  $i$   
**show**  $AE \ \xi \text{ in } M. X \ i \ \xi \leq \text{cond-exp } M (F \ i) (X (\text{Suc } i)) \ \xi$  **using** *cond-exp-diff*[*OF*  
*integrable*(1,1), *of i Suc i i*] *cond-exp-F-meas*[*OF integrable adapted, of i*] *assms*(2)[*of*  
*i*] **by** *fastforce*  
**qed**

**lemma** (*in nat-submartingale-linorder*) *partial-sum-scaleR*:  
**assumes** *nat-adapted-process*  $M \ F \ C \ \bigwedge i. AE \ \xi \text{ in } M. 0 \leq C \ i \ \xi \ \bigwedge i. AE \ \xi \text{ in } M. C \ i \ \xi \leq R$   
**shows** *nat-submartingale*  $M \ F (\lambda n \ \xi. \sum i < n. C \ i \ \xi *_{\mathbb{R}} (X (\text{Suc } i) \ \xi - X \ i \ \xi))$   
**proof** –  
**interpret**  $C$ : *nat-adapted-process*  $M \ F \ C$  **by** (*rule assms*)  
**interpret**  $C'$ : *nat-adapted-process*  $M \ F \ \lambda i \ \xi. C \ (i - 1) \ \xi *_{\mathbb{R}} (X \ i \ \xi - X \ (i - 1) \ \xi)$  **by** (*intro nat-adapted-process.intro adapted-process.scaleR-right-adapted*  
*adapted-process.diff-adapted, unfold-locales*) (*auto intro: adaptedD C.adaptedD*) +  
**interpret**  $C''$ : *nat-adapted-process*  $M \ F \ \lambda n \ \xi. \sum i < n. C \ i \ \xi *_{\mathbb{R}} (X (\text{Suc } i) \ \xi - X \ i \ \xi)$  **by** (*rule C'.partial-sum-Suc-adapted*[*unfolded diff-Suc-1*])  
**interpret**  $S$ : *nat-sigma-finite-adapted-process-linorder*  $M \ F (\lambda n \ \xi. \sum i < n. C \ i \ \xi *_{\mathbb{R}} (X (\text{Suc } i) \ \xi - X \ i \ \xi)) \dots$   
**have** *integrable*  $M (\lambda x. C \ i \ x *_{\mathbb{R}} (X (\text{Suc } i) \ x - X \ i \ x))$  **for**  $i$  **using** *assms*(2,3)[*of*  
*i*] **by** (*intro Bochner-Integration.integrable-bound*[*OF integrable-scaleR-right, OF*

*Bochner-Integration.integrable-diff*, *OF integrable(1,1)*, *of R Suc i i]*) (*auto simp add: mult-mono*)

**moreover have** *AE ξ in M. 0 ≤ cond-exp M (F i) (λξ. (∑ i<Suc i. C i ξ \*<sub>R</sub> (X (Suc i) ξ - X i ξ)) - (∑ i<i. C i ξ \*<sub>R</sub> (X (Suc i) ξ - X i ξ))) ξ for i*

**using** *sigma-finite-subalgebra.cond-exp-measurable-scaleR[OF - calculation - C.adapted, of i]*

*cond-exp-diff-nonneg[OF - le-SucI, OF - order.refl, of i] assms(2,3)[of i]*  
**by** (*fastforce simp add: scaleR-nonneg-nonneg integrable*)

**ultimately show** *?thesis by (intro S.submartingale-of-cond-exp-diff-Suc-nonneg Bochner-Integration.integrable-sum, blast+)*

**qed**

**lemma** (*in nat-submartingale-linorder*) *partial-sum-scaleR'*:

**assumes** *nat-predictable-process M F C ∧ i. AE ξ in M. 0 ≤ C i ξ ∧ i. AE ξ in M. C i ξ ≤ R*

**shows** *nat-submartingale M F (λn ξ. ∑ i<n. C (Suc i) ξ \*<sub>R</sub> (X (Suc i) ξ - X i ξ))*

**proof** –

**interpret** *C: nat-predictable-process M F C by (rule assms)*

**interpret** *Suc-C: nat-adapted-process M F λi. C (Suc i) using C.adapted-Suc .*

**show** *?thesis by (intro partial-sum-scaleR[of - R] assms) (intro-locales)*

**qed**

## 7.10 Discrete Time Supermartingales

**lemma** (*in nat-supermartingale*) *predictable-mono*:

**assumes** *nat-predictable-process M F X i ≤ j*

**shows** *AE ξ in M. X i ξ ≥ X j ξ*

**using** *assms(2)*

**proof** (*induction j – i arbitrary: i j*)

**case** *0*

**then show** *?case by simp*

**next**

**case** (*Suc n*)

**hence** *\*: n = j - Suc i by linarith*

**interpret** *S: nat-adapted-process M F λi. X (Suc i) by (intro nat-predictable-process.adapted-Suc assms)*

**have** *Suc i ≤ j using Suc(2,3) by linarith*

**thus** *?case using Suc(1)[OF \*] S.adapted[of i] supermartingale-property[OF - le-SucI, of i] sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i] by fastforce*

**qed**

**lemma** (*in nat-sigma-finite-adapted-process-linorder*) *supermartingale-of-set-integral-ge-Suc*:

**assumes** *integrable: ∧ i. integrable M (X i)*

**and** *∧ A i. A ∈ F i ⇒ set-lebesgue-integral M A (X i) ≥ set-lebesgue-integral M A (X (Suc i))*

**shows** *nat-supermartingale M F X*

**proof** –

```

interpret -: adapted-process  $M F 0 -X$  by (rule uminus-adapted)
interpret uminus- $X$ : nat-sigma-finite-adapted-process-linorder  $M F -X$  ..
note * = set-integral-uminus[unfolded set-integrable-def, OF integrable-mult-indicator[OF
- integrable]]
have nat-supermartingale  $M F (-(- X))$ 
using ord-eq-le-trans[OF * ord-le-eq-trans[OF le-imp-neg-le[OF assms(2)] *[symmetric]]]
subalgebras
by (intro nat-supermartingale.intro submartingale.uminus nat-submartingale.axioms
uminus- $X$ .submartingale-of-set-integral-le-Suc)
(clarsimp simp add: fun-Compl-def subalgebra-def integrable | fastforce)+
thus ?thesis unfolding fun-Compl-def by simp
qed

```

**lemma** (in nat-sigma-finite-adapted-process-linorder) supermartingale-nat:

```

assumes integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
and  $\bigwedge i. AE \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X (Suc i)) \xi$ 
shows nat-supermartingale  $M F X$ 

```

**proof** –

```

interpret -: adapted-process  $M F 0 -X$  by (rule uminus-adapted)
interpret uminus- $X$ : nat-sigma-finite-adapted-process-linorder  $M F -X$  ..
have  $AE \xi \text{ in } M. - X i \xi \leq \text{cond-exp } M (F i) (\lambda x. - X (Suc i) x) \xi$  for  $i$  using
assms(2) cond-exp-uminus[OF integrable, of  $i$  Suc  $i$ ] by force
hence nat-supermartingale  $M F (-(- X))$  by (intro nat-supermartingale.intro
submartingale.uminus nat-submartingale.axioms uminus- $X$ .submartingale-nat) (auto
simp add: fun-Compl-def integrable)
thus ?thesis unfolding fun-Compl-def by simp
qed

```

**lemma** (in nat-sigma-finite-adapted-process-linorder) supermartingale-of-cond-exp-diff-Suc-le-zero:

```

assumes integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
and  $\bigwedge i. AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X (Suc i) \xi - X i \xi) \xi \leq 0$ 
shows nat-supermartingale  $M F X$ 

```

**proof** (intro supermartingale-nat integrable)

```

fix  $i$ 
show  $AE \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X (Suc i)) \xi$  using cond-exp-diff[OF
integrable(1,1), of  $i$  Suc  $i$ ] cond-exp-F-meas[OF integrable adapted, of  $i$ ] assms(2)[of
 $i$ ] by fastforce
qed

```

**end**

**theory** Example-Coin-Toss

```

imports Martingale HOL-Probability.Stream-Space HOL-Probability.Probability-Mass-Function
begin

```



## 8 Example: Coin Toss

We consider a probability space consisting of infinite sequences of coin tosses.

**definition** *bernoulli-stream* :: *real*  $\Rightarrow$  (*bool stream*) *measure* **where**  
*bernoulli-stream* *p* = *stream-space* (*measure-pmf* (*bernoulli-pmf* *p*))

**lemma** *space-bernoulli-stream[simp]*: *space* (*bernoulli-stream* *p*) = *UNIV* **by** (*simp add: bernoulli-stream-def space-stream-space*)

We define the fortune of the player at time *n* to be the number of heads minus number of tails.

**definition** *fortune* :: *nat*  $\Rightarrow$  *bool stream*  $\Rightarrow$  *real* **where**  
*fortune* *n* = ( $\lambda s. \sum b \leftarrow \text{stake } (\text{Suc } n) \ s. \text{ if } b \text{ then } 1 \text{ else } -1$ )

**definition** *toss* :: *nat*  $\Rightarrow$  *bool stream*  $\Rightarrow$  *real* **where**  
*toss* *n* = ( $\lambda s. \text{ if } \text{snth } s \ n \text{ then } 1 \text{ else } -1$ )

**lemma** *toss-indicator-def*: *toss* *n* = *indicator* {*s*. *s* !! *n*} − *indicator* {*s*.  $\neg s$  !! *n*}  
**unfolding** *toss-def indicator-def* **by** *force*

**lemma** *range-toss*: *range* (*toss* *n*) = {−1, 1}

**proof** −

**have** *sconst True* !! *n* **by** *simp*  
**moreover** **have**  $\neg \text{sconst False}$  !! *n* **by** *simp*  
**ultimately** **have**  $\exists x. x$  !! *n*  $\exists x. \neg x$  !! *n* **by** *blast+*  
**thus** *?thesis* **unfolding** *toss-def image-def* **by** *auto*

**qed**

**lemma** *vimage-toss*: *toss* *n* − ‘*A* = (*if* 1 ∈ *A* *then* {*s*. *s* !! *n*} *else* { }) ∪ (*if* −1 ∈ *A* *then* {*s*.  $\neg s$  !! *n*} *else* { })

**unfolding** *vimage-def toss-def* **by** *auto*

**lemma** *fortune-Suc*: *fortune* (*Suc* *n*) *s* = *fortune* *n* *s* + *toss* (*Suc* *n*) *s*  
**by** (*induction* *n* *arbitrary: s*) (*simp add: fortune-def toss-def*) +

**lemma** *fortune-toss-sum*: *fortune* *n* *s* = ( $\sum i \in \{..n\}. \text{toss } i \ s$ )  
**by** (*induction* *n* *arbitrary: s*) (*simp add: fortune-def toss-def, simp add: fortune-Suc*)

**lemma** *fortune-bound*: *norm* (*fortune* *n* *s*) ≤ *Suc* *n* **by** (*induction* *n*) (*force simp add: fortune-toss-sum toss-def*) +

Our definition of *bernoulli-stream* constitutes a probability space.

**interpretation** *prob-space* *bernoulli-stream* *p* **unfolding** *bernoulli-stream-def* **by**  
(*simp add: measure-pmf.prob-space-axioms prob-space.prob-space-stream-space*)

**abbreviation** *toss-filtration* *p*  $\equiv$  *nat-natural-filtration* (*bernoulli-stream* *p*) *toss*

The stochastic process *toss* is adapted to the filtration it generates.

**interpretation** *toss*: *nat-adapted-process bernoulli-stream p nat-natural-filtration*  
*(bernoulli-stream p) toss toss*  
**by** (*intro nat-adapted-process.intro stochastic-process.adapted-process-natural-filtration*)  
*(unfold-locales, auto simp add: toss-def bernoulli-stream-def)*

Similarly, the stochastic process *fortune* is adapted to the filtration generated by the tosses.

**interpretation** *fortune*: *nat-finite-adapted-process-linorder bernoulli-stream p nat-natural-filtration*  
*(bernoulli-stream p) toss fortune*

**proof** –

**show** *nat-finite-adapted-process-linorder (bernoulli-stream p) (toss-filtration p)*  
*fortune*

**unfolding** *fortune-toss-sum*

**by** (*intro nat-finite-adapted-process-linorder.intro*  
*finite-adapted-process-linorder.intro*  
*finite-adapted-process-order.intro*  
*finite-adapted-process.intro*  
*toss.partial-sum-adapted[folded atMost-atLeast0]) intro-locales*

**qed**

**lemma** *integrable-toss*: *integrable (bernoulli-stream p) (toss n)*

**using** *toss.random-variable*

**by** (*intro Bochner-Integration.integrable-bound[OF integrable-const[of - 1 :: real]]*)  
*(auto simp add: toss-def)*

**lemma** *integrable-fortune*: *integrable (bernoulli-stream p) (fortune n)* **using** *fortune-bound*

**by** (*intro Bochner-Integration.integrable-bound[OF integrable-const[of - Suc n]*  
*fortune.random-variable]*) *auto*

We provide the following lemma to explicitly calculate the probability of events in this probability space.

**lemma** *measure-bernoulli-stream-snth-pred*:

**assumes**  $0 \leq p$  **and**  $p \leq 1$  **and** *finite J*

**shows** *prob p {w ∈ space (bernoulli-stream p).  $\forall j \in J. P j = w !! j$ } =  $p^{\wedge \text{card } (J \cap \text{Collect } P)} * (1 - p)^{\wedge (\text{card } (J - \text{Collect } P)}$*

**proof** –

**let** *?PiE* = ( $\Pi_E i \in J. \text{if } P i \text{ then } \{\text{True}\} \text{ else } \{\text{False}\}$ )

**have** *product-prob-space ( $\lambda \cdot. \text{measure-pmf (bernoulli-pmf p)}$ )* **by** *unfold-locales*

**hence** *\**: *to-stream* – ‘ $\{s. \forall i \in J. P i = s !! i\} = \{s. \forall i \in J. P i = s i\}$ ’ **using**  
*assms by (simp add: to-stream-def)*

**also have** *... = prod-emb UNIV ( $\lambda \cdot. \text{measure-pmf (bernoulli-pmf p)}$ ) J ?PiE*

**proof** –

{

**fix** *s* **assume** ( $\forall i \in J. P i = s i$ )

**hence** ( $\forall i \in J. P i = s i$ ) = *(s ∈ prod-emb UNIV ( $\lambda \cdot. \text{measure-pmf (bernoulli-pmf p)}$ ) J ?PiE)*

by (subst prod-emb-iff[of s]) (smt (verit, best) not-def assms(3) id-def  
 PiE-eq-singleton UNIV-I extensional-UNIV insert-iff singletonD space-measure-pmf)  
 }  
 moreover  
 {  
 fix s assume  $\neg(\forall i \in J. P\ i = s\ i)$   
 then obtain i where  $i \in J$   $P\ i \neq s\ i$  by blast  
 hence  $(\forall i \in J. P\ i = s\ i) = (s \in \text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf p)}))\ J\ ?PiE$   
 by (simp add: restrict-def prod-emb-iff[of s]) (smt (verit, ccfv-SIG) PiE-mem  
 assms(3) id-def insert-iff singleton-iff)  
 }  
 ultimately show ?thesis by auto  
 qed  
 finally have integ:  $(\text{to-stream } -' \{s. \forall i \in J. P\ i = s\ i\}) = \text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf p)})\ J\ ?PiE$ .  
 let ?M =  $(Pi_M\ UNIV\ (\lambda-. \text{measure-pmf (bernoulli-pmf p)}))$   
 have emeasure  $(\text{bernoulli-stream } p)\ \{s \in \text{space (bernoulli-stream } p). \forall i \in J. P\ i = s\ i\} = \text{emeasure } ?M\ (\text{to-stream } -' \{s. \forall i \in J. P\ i = s\ i\})$   
 using assms emeasure-distr[of to-stream ?M (vimage-algebra (streams (space (measure-pmf (bernoulli-pmf p)))) (!) ?M) {s.  $\forall i \in J. P\ i = s\ i$ }, symmetric]  
 measurable-to-stream[of (measure-pmf (bernoulli-pmf p))]  
 by (simp only: bernoulli-stream-def stream-space-def \*, simp add: space-PiM )  
 (smt (verit, best) emeasure-notin-sets in-vimage-algebra inf-top.right-neutral sets-distr  
 vimage-Collect)  
 also have ... =  $\text{emeasure } ?M\ (\text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf p)}))\ J\ ?PiE$  using integ by (simp add: space-PiM)  
 also have ... =  $(\prod_{i \in J. \text{emeasure (measure-pmf (bernoulli-pmf p)) (if } P\ i \text{ then } \{True\} \text{ else } \{False\})})$   
 by (subst emeasure-PiM-emb) (auto simp add: prob-space-measure-pmf assms(3))  
 also have ... =  $(\prod_{i \in J \cap \text{Collect } P. \text{ennreal } p}) * (\prod_{i \in J - \text{Collect } P. \text{ennreal } (1 - p)})$   
 unfolding emeasure-pmf-single[of bernoulli-pmf p True, unfolded pmf-bernoulli-True[OF  
 assms(1,2)], symmetric]  
 emeasure-pmf-single[of bernoulli-pmf p False, unfolded pmf-bernoulli-False[OF  
 assms(1,2)], symmetric]  
 by (simp add: prod.Int-Diff[OF assms(3), of - Collect P])  
 also have ... =  $p^{\text{card } (J \cap \text{Collect } P)} * (1 - p)^{\text{card } (J - \text{Collect } P)}$  using  
 assms by (simp add: prod-ennreal ennreal-mult' ennreal-power)  
 finally show ?thesis using assms by (intro measure-eq-emeasure-eq-ennreal)  
 auto  
 qed

**lemma**  
 assumes  $0 \leq p$  and  $p \leq 1$   
 shows  $\text{measure-bernoulli-stream-snth: prob } p\ \{w \in \text{space (bernoulli-stream } p). w\ i\} = p$   
 and  $\text{measure-bernoulli-stream-neg-snth: prob } p\ \{w \in \text{space (bernoulli-stream } p). \neg w\ i\} = 1 - p$

**using** *measure-bernoulli-stream-snth-pred*[*OF* *assms*, of  $\{i\} \lambda x. \text{True}$ ]  
*measure-bernoulli-stream-snth-pred*[*OF* *assms*, of  $\{i\} \lambda x. \text{False}$ ] **by** *auto*

Now we can express the expected value of a single coin toss.

**lemma** *integral-toss*:  
**assumes**  $0 \leq p \leq 1$   
**shows** *expectation*  $p$  (*toss*  $n$ ) =  $2 * p - 1$   
**proof** –  
**have** [*simp*]: $\{s. s !! n\} \in \text{events } p$  **using** *measurable-snth*[*THEN* *measurable-sets*,  
of  $\{\text{True}\}$  *measure-pmf* (*bernoulli-pmf*  $p$ )  $n$ , *folded* *bernoulli-stream-def*]  
**by** (*simp* *add*: *vimage-def*)  
**have** *expectation*  $p$  (*toss*  $n$ ) = *Bochner-Integration.simple-bochner-integral* (*bernoulli-stream*  
 $p$ ) (*toss*  $n$ )  
**using** *toss.random-variable*[of  $n$ , *THEN* *measurable-sets*]  
**by** (*intro* *simple-bochner-integrable-eq-integral*[*symmetric*] *simple-bochner-integrable.intros*)  
(*auto* *simp* *add*: *toss-def* *simple-function-def* *image-def*)  
**also have** ... =  $p - \text{prob } p \{s. \neg s !! n\}$  **unfolding** *simple-bochner-integral-def*  
**using** *measure-bernoulli-stream-snth*[*OF* *assms*]  
**by** (*simp* *add*: *range-toss*, *simp* *add*: *toss-def*)  
**also have** ... =  $p - (1 - \text{prob } p \{s. s !! n\})$  **by** (*subst* *prob-compl*[*symmetric*],  
*auto* *simp* *add*: *Collect-neg-eq* *Compl-eq-Diff-UNIV*)  
**finally show** ?*thesis* **using** *measure-bernoulli-stream-snth*[*OF* *assms*] **by** *simp*  
**qed**

Now, we show that the tosses are independent from one another.

**lemma** *indep-vars-toss*:  
**assumes**  $0 \leq p \leq 1$   
**shows** *indep-vars*  $p$  ( $\lambda-. \text{borel}$ ) *toss*  $\{0..\}$   
**proof** (*subst* *indep-vars-def*, *intro* *conjI* *indep-sets-sigma*)  
{  
**fix**  $A \ J$  **assume** *asm*:  $J \neq \{\}$  *finite*  $J \ \forall j \in J. A \ j \in \{\text{toss } j - 'A \cap \text{space}$   
(*bernoulli-stream*  $p$ )  $| A. A \in \text{borel}\}$   
**hence**  $\forall j \in J. \exists B \in \text{borel}. A \ j = \text{toss } j - 'B \cap \text{space}$  (*bernoulli-stream*  $p$ ) **by**  
*auto*  
**then obtain**  $B$  **where**  $B\text{-is: } A \ j = \text{toss } j - 'B \ j \cap \text{space}$  (*bernoulli-stream*  $p$ )  
 $B \ j \in \text{borel}$  **if**  $j \in J$  **for**  $j$  **by** *metis*  
  
**have** *prob*  $p$  ( $\bigcap (A - 'J)$ ) =  $(\prod_{j \in J}. \text{prob } p (A \ j))$   
**proof** *cases*

We consider the case where there is a zero probability event.

**assume**  $\exists j \in J. 1 \notin B \ j \wedge \neg 1 \notin B \ j$   
**then obtain**  $j$  **where**  $j\text{-is: } j \in J \ 1 \notin B \ j \ \neg 1 \notin B \ j$  **by** *blast*  
**hence**  $A\text{-}j\text{-empty: } A \ j = \{\}$  **using**  $B\text{-is}$  **by** (*force* *simp* *add*: *toss-def* *vimage-def*)  
**hence**  $\bigcap (A - 'J) = \{\}$  **using**  $j\text{-is}$  **by** *blast*  
**moreover have** *prob*  $p$  ( $A \ j$ ) = 0 **using**  $A\text{-}j\text{-empty}$  **by** *simp*  
**ultimately show** ?*thesis* **using**  $j\text{-is}$  *asm*(2) **by** *auto*  
**next**

We now assume all events have positive probability.

**assume**  $\neg(\exists j \in J. 1 \notin B j \wedge -1 \notin B j)$   
**hence** \*:  $1 \in B j \vee -1 \in B j$  **if**  $j \in J$  **for**  $j$  **using** *that* **by** *blast*

**define**  $J'$  **where** [*simp*]:  $J' = \{j \in J. (1 \in B j) \longleftrightarrow (-1 \notin B j)\}$   
**hence**  $\text{toss } j \ w \in B j \longleftrightarrow (1 \in B j) = w !! j$  **if**  $j \in J'$  **for**  $w j$  **using** *that*  
**unfolding** *toss-def* **by** *simp*  
**hence**  $(\bigcap (A \text{ ' } J')) = \{w \in \text{space } (\text{bernoulli-stream } p). \forall j \in J'. (1 \in B j) = w !! j\}$  **using** *B-is* **by** *force*  
**hence**  $\text{prob-}J'$ :  $\text{prob } p (\bigcap (A \text{ ' } J')) = p \wedge \text{card } (J' \cap \{j. 1 \in B j\}) * (1 - p) \wedge \text{card } (J' - \{j. 1 \in B j\})$   
**using** *measure-bernoulli-stream-snth-pred*[*OF assms finite-subset* [*OF - asm*(2)], *of*  $J' \lambda j. 1 \in B j$ ] **by** *auto*

The index set  $J'$  consists of the indices of all non-trivial events.

**have**  $A\text{-}j\text{-True}$ :  $A j = \{w \in \text{space } (\text{bernoulli-stream } p). w !! j\}$  **if**  $j \in J' \cap \{j. 1 \in B j\}$  **for**  $j$   
**using** *that* **by** (*auto simp add: toss-def B-is*(1) *split: if-splits*)

**have**  $A\text{-}j\text{-False}$ :  $A j = \{w \in \text{space } (\text{bernoulli-stream } p). \neg w !! j\}$  **if**  $j \in J' - \{j. 1 \in B j\}$  **for**  $j$   
**using** *that* *B-is* **by** (*auto simp add: toss-def*)

**have**  $A\text{-}j\text{-top}$ :  $A j = \text{space } (\text{bernoulli-stream } p)$  **if**  $j \in J - J'$  **for**  $j$  **using** *that*  
\* **by** (*auto simp add: B-is toss-def*)  
**hence**  $\bigcap (A \text{ ' } J) = \bigcap (A \text{ ' } J')$  **by** *auto*  
**hence**  $\text{prob } p (\bigcap (A \text{ ' } J)) = \text{prob } p (\bigcap (A \text{ ' } J'))$  **by** *presburger*  
**also have** ... =  $(\prod_{j \in J' \cap \{j. 1 \in B j\}} \text{prob } p (A j)) * (\prod_{j \in J' - \{j. 1 \in B j\}} \text{prob } p (A j))$   
**by** (*simp only: prob-}J' A-j-True A-j-False measure-bernoulli-stream-snth*[*OF assms*] *measure-bernoulli-stream-neg-snth*[*OF assms*] *cong: prod.cong*) *simp*  
**also have** ... =  $(\prod_{j \in J'} \text{prob } p (A j))$  **using** *asm*(2) **by** (*intro prod.Int-Diff*[*symmetric*])  
*auto*  
**also have** ... =  $(\prod_{j \in J'} \text{prob } p (A j)) * (\prod_{j \in J - J'} \text{prob } p (A j))$  **using**  
*A-j-top prob-space* **by** *simp*  
**also have** ... =  $(\prod_{j \in J} \text{prob } p (A j))$  **using** *asm*(2) **by** (*metis* (*no-types*, *lifting*) *J'-def mem-Collect-eq mult.commute prod.subset-diff subsetI*)  
**finally show** *?thesis* .

**qed**

}

**thus** *indep-sets*  $p (\lambda i. \{ \text{toss } i - ' A \cap \text{space } (\text{bernoulli-stream } p) \mid A. A \in \text{sets borel} \}) \{0..\}$  **using** *measurable-sets*[*OF toss.random-variable*]

**by** (*intro indep-setsI subsetI*) *fastforce*

**qed** (*simp, intro Int-stableI, simp, metis sets.Int vimage-Int*)

The fortune of a player is a martingale (resp. sub- or supermartingale) with respect to the filtration generated by the coin tosses.

**theorem** *fortune-martingale*:

**assumes**  $p = 1/2$   
**shows** *nat-martingale* (*bernoulli-stream*  $p$ ) (*toss-filtration*  $p$ ) *fortune*  
**using** *cond-exp-indep*[*OF fortune.subalg indep-set-natural-filtration integrable-toss*,  
*OF zero-order(1) lessI indep-vars-toss*, of  $p$ ]  
*integral-toss* *assms*  
**by** (*intro fortune.martingale-of-cond-exp-diff-Suc-eq-zero integrable-fortune*)  
*(force simp add: fortune-toss-sum)*

**theorem** *fortune-submartingale*:

**assumes**  $1/2 \leq p \leq 1$   
**shows** *nat-submartingale* (*bernoulli-stream*  $p$ ) (*toss-filtration*  $p$ ) *fortune*  
**proof** (*intro fortune.submartingale-of-cond-exp-diff-Suc-nonneg integrable-fortune*)  
**fix**  $n$   
**show** *AE  $\xi$  in bernoulli-stream  $p$ .  $0 \leq \text{cond-exp}(\text{bernoulli-stream } p) (\text{toss-filtration } p \ n) (\lambda \xi. \text{fortune}(\text{Suc } n) \ \xi - \text{fortune } n \ \xi) \ \xi$*   
**using** *cond-exp-indep*[*OF fortune.subalg indep-set-natural-filtration integrable-toss*,  
*OF zero-order(1) lessI indep-vars-toss*, of  $p \ n$ ]  
*integral-toss*[of  $p \ \text{Suc } n$ ] *assms*  
**by** (*force simp add: fortune-toss-sum*)  
**qed**

**theorem** *fortune-supermartingale*:

**assumes**  $0 \leq p \leq 1/2$   
**shows** *nat-supermartingale* (*bernoulli-stream*  $p$ ) (*toss-filtration*  $p$ ) *fortune*  
**proof** (*intro fortune.supermartingale-of-cond-exp-diff-Suc-le-zero integrable-fortune*)  
**fix**  $n$   
**show** *AE  $\xi$  in bernoulli-stream  $p$ .  $0 \geq \text{cond-exp}(\text{bernoulli-stream } p) (\text{toss-filtration } p \ n) (\lambda \xi. \text{fortune}(\text{Suc } n) \ \xi - \text{fortune } n \ \xi) \ \xi$*   
**using** *cond-exp-indep*[*OF fortune.subalg indep-set-natural-filtration integrable-toss*,  
*OF zero-order(1) lessI indep-vars-toss*, of  $p \ n$ ]  
*integral-toss*[of  $p \ \text{Suc } n$ ] *assms*  
**by** (*force simp add: fortune-toss-sum*)  
**qed**

**end**

## References

- [1] T. Hytönen, J. v. Neerven, M. Veraar, and L. Weis. *Analysis in Banach Spaces Volume I: Martingales and Littlewood-Paley theory*. Springer International Publishing, 2016.
- [2] A. Keskin. A Formalization of Martingales in Isabelle/HOL, Sept. 2023. <https://github.com/atakeskinn/martingales>.
- [3] A. Keskin. A formalization of martingales in isabelle/hol, 2023.
- [4] S. Lang. *Real and Functional Analysis*. Springer, 1993.