

On the Formalization of Martingales

Ata Keskin

April 18, 2024

Abstract

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

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 definitions¹. 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 corollaries². 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 ordering³. Specifically, we formalize a powerful result called the “Averaging Theorem”[3] 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 [2].

¹`Martingale.Stochastic_Process`

²`Martingale.Martingale`

³`Martingale.Bochner_Integration_Addendum`

Contents

1	Supplementary Lemmas for Measure Spaces	3
1.1	σ -Algebra Generated by a Family of Functions	3
2	Conditional Expectation in Banach Spaces	4
2.1	Existence	9
2.2	Properties	19
2.3	Linearly Ordered Banach Spaces	25
2.4	Probability Spaces	29
3	Filtered Measure Spaces	35
3.1	Filtered Measure	35
3.2	σ -Finite Filtered Measure	36
3.3	Finite Filtered Measure	37
3.4	Constant Filtration	37
4	Stochastic Processes	38
4.1	Stochastic Process	38
4.1.1	Natural Filtration	39
4.2	Adapted Process	42
4.3	Progressively Measurable Process	45
4.4	Predictable Process	48
5	Martingales	58
5.1	Martingale	58
5.2	Submartingale	59
5.3	Supermartingale	60
5.4	Martingale Lemmas	61
5.5	Submartingale Lemmas	64
5.6	Supermartingale Lemmas	67
5.7	Discrete Time Martingales	71
5.8	Discrete Time Submartingales	73
5.9	Discrete Time Supermartingales	75
6	Example: Coin Toss	77

```

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

```

1 Supplementary Lemmas for Measure Spaces

1.1 σ -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 Ω S $M \equiv$ sigma Ω ($\bigcup f \in S. \{f - ' A \cap \Omega \mid A. A \in M\}$)

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* Ω $\{f\}$ $M =$ *vimage-algebra* Ω f M **unfolding** *family-vimage-algebra-def* *vimage-algebra-def* **by** *simp*

lemma

shows *sets-family-vimage-algebra*: *sets* (*family-vimage-algebra* Ω S M) = *sigma-sets* Ω ($\bigcup f \in S. \{f - ' A \cap \Omega \mid A. A \in M\}$)

and *space-family-vimage-algebra[simp]*: *space* (*family-vimage-algebra* Ω S M) = Ω

by (*auto simp add: family-vimage-algebra-def sets-measure-of-conv space-measure-of-conv*)

lemma *measurable-family-vimage-algebra*:

assumes $f \in S$ $f \in \Omega \rightarrow$ *space* M

shows $f \in$ *family-vimage-algebra* Ω S $M \rightarrow_M$ M

using *assms* **by** (*intro measurableI, auto simp add: sets-family-vimage-algebra*)

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

assumes $f \in \Omega \rightarrow$ *space* M

shows $f \in$ *family-vimage-algebra* Ω $\{f\}$ $M \rightarrow_M$ M

using *assms* *measurable-family-vimage-algebra* **by** *blast*

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

lemma *measurable-family-iff-sets*:

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

proof (*standard, goal-cases*)

case 1

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

have $\{f - ' A \cap$ *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 (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 Conditional-Expectation-Banach
  imports HOL-Probability.Conditional-Expectation HOL-Probability.Independent-Family

```

begin

2 Conditional Expectation in Banach Spaces

While constructing the conditional expectation operator, we have come up with the following approach, which is based on the construction in [1]. Both our approach, and the one in [1] are based on showing that the conditional expectation is a contraction on some dense subspace of the space of functions $L^1(E)$. In our approach, we start by constructing the conditional expectation explicitly for simple functions. Then we show that the conditional expectation is a contraction on simple functions, i.e. $\|E(s|F)(x)\| \leq E(\|s(x)\||F)$ for μ -almost all $x \in \Omega$ with $s : \Omega \rightarrow E$ simple and integrable. Using this, we can show that the conditional expectation of a convergent sequence of simple functions is again convergent. Finally, we show that this limit exhibits the properties of a conditional expectation. This approach has the benefit of being straightforward and easy to implement, since we could make use of the existing formalization for real-valued functions. To use the construction in [1] we need more tools from functional analysis, which

Isabelle/HOL currently does not have.

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

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

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

\wedge integrable M f
 \wedge integrable M g
 \wedge g \in borel-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- σ -algebra F .

lemma *has-cond-expI'*:

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 borel-measurable F

shows *has-cond-exp* M F f g

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

lemma *has-cond-expD*:

assumes *has-cond-exp* M F f g

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

integrable M f

integrable M g

g \in borel-measurable F

using *assms unfolding has-cond-exp-def* **by** *simp+*

Now we can use Hilberts ϵ -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 ∈ 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 argo*

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

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 $\text{AE } x \text{ in } ?MF. \text{cond-exp } M F f x = g x$ **using** *cond-exp assms subalg* **by**
(*intro density-unique-banach, auto dest: has-cond-expD intro!: integrable-in-subalg*)
then show $\text{AE } x \text{ in } M. \text{cond-exp } M F f x = g x$ **using** *AE-restr-to-subalg[OF subalg]* **by** *simp*
qed

corollary *cond-exp-charact*:

fixes $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$
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 $\text{AE } x \text{ in } M. \text{cond-exp } M F f x = g x$
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 $\text{cond-exp } M F f = f \mu$ -a.e. This is a corollary of the lemma on the characterization of *cond-exp*.

corollary *cond-exp-F-meas[intro, simp]*:

fixes $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$
assumes *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 cond-exp-charact, auto intro: assms*)

Congruence

lemma *has-cond-exp-cong*:

assumes *integrable M f* $\bigwedge x. x \in \text{space } M \implies f x = g x$ *has-cond-exp M F g h*
shows *has-cond-exp M F f h*
proof (*intro has-cond-expI'[OF - assms(1)]*)
fix A **assume** *asm: A ∈ sets F*

hence *set-lebesgue-integral* $M A f = \text{set-lebesgue-integral } M A g$ **by** (*intro set-lebesgue-integral-cong*)
(meson assms(2) subalg in-mono subalgebra-def sets.sets-into-space subalgebra-def subsetD)+
thus *set-lebesgue-integral* $M A f = \text{set-lebesgue-integral } M A h$ **using** *asm assms(3)*
by (*simp add: has-cond-exp-def*)
qed (*auto simp add: has-cond-expD[OF assms(3)]*)

lemma *cond-exp-cong*:

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

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

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

lemma *has-cond-exp-cong-AE'*:

assumes $h \in \text{borel-measurable } F$ *AE* x *in* $M. h x = h' x$ *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 $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$
assumes *integrable* $M f$ *integrable* $M g$ *AE* x *in* $M. f x = g x$
shows *AE* x *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 *True*
then obtain h **where** $h: \text{has-cond-exp } M F f h \text{ has-cond-exp } M F g h$ **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  $\nexists h. \text{has-cond-exp } M F g h$  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  $\Rightarrow$  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  $\Rightarrow$  real
  assumes integrable M f
  shows AE x in M. cond-exp M F f x = 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  $\Rightarrow$  real
  assumes integrable M f
  shows AE x in M. cond-exp M F ( $\lambda x. c * f x$ ) x = c * 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

```

2.1 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  $\in$  sets M emeasure M A <  $\infty$ 
  shows has-cond-exp M F ( $\lambda x. \text{indicat-real } A x *_R y$ ) ( $\lambda x. \text{real-cond-exp } M F$ 
  (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 subalge-
  bra-def, blast)

```

also have ... = $(\int x \in B. \text{real-cond-exp } M F (\text{indicator } A) x \partial M) *_R y$ **using** 1
assms by (subst real-cond-exp-intA, auto)
also have ... = $(\int x \in B. (\text{real-cond-exp } M F (\text{indicator } A) x *_R y) \partial M)$ **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 $y :: 'b :: \{\text{second-countable-topology, banach}\}$
assumes [measurable]: $A \in \text{sets } M \text{ emeasure } M A < \infty$
shows $AE x \text{ in } M. \text{cond-exp } M F (\lambda x. \text{indicat-real } A x *_R y) x = \text{cond-exp } M F$
*(indicator A) x *_R y*
proof –
have $AE x \text{ in } M. \text{cond-exp } M F (\lambda x. \text{indicat-real } A x *_R y) x = \text{real-cond-exp } M F$
*(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 $f g :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$
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 ... = $(\int x \in A. f' x \partial M) + (\int x \in A. g' x \partial M)$ **using** *assms[THEN*
has-cond-expD(1)[OF - 1]] by argo
also have ... = $(\int x \in A. (f' x + g' x) \partial M)$ **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::{second-countable-topology,banach}
  assumes has-cond-exp M F f f'
  shows has-cond-exp M F ( $\lambda x. c *_R f x$ ) ( $\lambda x. c *_R f' x$ )
  using has-cond-expD[OF assms] by (intro has-cond-expI', auto)

```

```

lemma cond-exp-scaleR-right:
  fixes f :: 'a  $\Rightarrow$  'b::{second-countable-topology,banach}
  assumes integrable M f
  shows AE x in M. cond-exp M F ( $\lambda x. c *_R f x$ ) x = c *_R cond-exp M F f x
proof (cases  $\exists f'. has-cond-exp M F f f'$ )
  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  $\nexists f'. has-cond-exp M F (\lambda x. c *_R f x) f'$ 
    proof (standard, goal-cases)
      case 1
      then obtain f' where f': has-cond-exp M F ( $\lambda x. c *_R f x$ ) f' by blast
      have has-cond-exp M F f ( $\lambda x. inverse c *_R f' x$ ) 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  $\Rightarrow$  'b::{second-countable-topology,banach}
  assumes integrable M f
  shows AE x in M. cond-exp M F ( $\lambda x. - f x$ ) 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 ⇒ real
assumes integrable[measurable]: integrable M f
shows AE x in M. norm (cond-exp M F f x) ≤ cond-exp M F (λx. norm (f x)) x
proof –
  have int: integrable M (λx. norm (f x)) using assms by blast
  have *: AE x in M. 0 ≤ cond-exp M F (λx. 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 ∈ sets F ⇒ (∫ x∈A. |f x| ∂M) = (∫ x∈A. real-cond-exp M F (λx.
norm (f x)) x ∂M) for A unfolding real-norm-def using assms integrable-abs
real-cond-exp-intA by blast

```

```

have norm-int: A ∈ sets F ⇒ (∫ x∈A. |f x| ∂M) = (∫ +x∈A. |f x| ∂M) for A

```

using *assms* **by** (*intro nn-set-integral-eq-set-integral[symmetric]*, *blast*, *fastforce*)
(*meson subalg subalgebra-def subsetD*)

have $\text{AE } x \text{ in } M. \text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x \geq 0$ **using** *int real-cond-exp-ge-c*
by *force*

hence $\text{cond-exp-norm-int: } A \in \text{sets } F \implies (\int x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x \partial M) = (\int ^+ x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x \partial M)$ **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** $\text{AE } x \text{ in } M. \text{nn-cond-exp } M F (\lambda x. \text{ennreal } |f x|) x = \text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x$ **by** (*intro nn-cond-exp-charact[THEN AE-symmetric]*,
auto)

hence $\text{AE } x \text{ in } M. \text{nn-cond-exp } M F (\lambda x. \text{ennreal } |f x|) x \leq \text{cond-exp } M F (\lambda x. \text{norm } (f x)) x$ **using** *cond-exp-real[OF int]* **by** *force*

moreover **have** $\text{AE } x \text{ in } M. |\text{real-cond-exp } M F f x| = \text{norm } (\text{cond-exp } M F f x)$

unfolding *real-norm-def* **using** *cond-exp-real[OF assms]* ***** **by** *force*

ultimately **have** $\text{AE } x \text{ in } M. \text{ennreal } (\text{norm } (\text{cond-exp } M F f x)) \leq \text{cond-exp } M F (\lambda x. \text{norm } (f x)) x$ **using** *real-cond-exp-abs[OF assms[THEN borel-measurable-integrable]]*
by *fastforce*

hence $\text{AE } x \text{ in } M. \text{enn2real } (\text{ennreal } (\text{norm } (\text{cond-exp } M F f x))) \leq \text{enn2real } (\text{cond-exp } M F (\lambda x. \text{norm } (f x)) x)$ **using** *ennreal-le-iff2* **by** *force*

thus *?thesis* **using** *** **by** *fastforce*

qed

lemma *cond-exp-contraction-simple*:

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

assumes *simple-function* $M f \text{ emeasure } M \{y \in \text{space } M. f y \neq 0\} \neq \infty$

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

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 $\text{AE } x \text{ in } M. \text{cond-exp } M F f x = \text{cond-exp } M F g x$ **using** *cong has-cond-exp-simple*
by (*subst cond-exp-cong-AE*) (*auto intro!: has-cond-expD(2)*)

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

moreover **have** $\text{AE } x \text{ in } M. \text{cond-exp } M F (\lambda x. \text{norm } (f x)) x = \text{cond-exp } M F (\lambda x. \text{norm } (g x)) x$ **using** *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 . $cond\text{-}exp$ M F $(\lambda a. indicator$ A $a *_R y)$ $x = cond\text{-}exp$ M F $(indicator$ $A)$ $x *_R y$ **by** *blast*

hence $*$: AE x in M . $norm$ $(cond\text{-}exp$ M F $(\lambda a. indicat\text{-}real$ A $a *_R y)$ $x) \leq norm$ y $* cond\text{-}exp$ M F $(\lambda x. norm$ $(indicat\text{-}real$ A $x))$ x **using** $cond\text{-}exp\text{-}contraction\text{-}real$ $[OF$ $integrable\text{-}real\text{-}indicator, OF$ $indicator]$ **by** *fastforce*

have AE x in M . $norm$ $y * cond\text{-}exp$ M F $(\lambda x. norm$ $(indicat\text{-}real$ A $x))$ $x = norm$ $y * real\text{-}cond\text{-}exp$ M F $(\lambda x. norm$ $(indicat\text{-}real$ A $x))$ x **using** $cond\text{-}exp\text{-}real$ $[OF$ $integrable\text{-}real\text{-}indicator, OF$ $indicator]$ **by** *fastforce*

moreover **have** AE x in M . $cond\text{-}exp$ M F $(\lambda x. norm$ $y * norm$ $(indicat\text{-}real$ A $x))$ $x = real\text{-}cond\text{-}exp$ M F $(\lambda x. norm$ $y * norm$ $(indicat\text{-}real$ A $x))$ x **using** $indicator$ **by** $(intro$ $cond\text{-}exp\text{-}real, auto)$

ultimately **have** AE x in M . $norm$ $y * cond\text{-}exp$ M F $(\lambda x. norm$ $(indicat\text{-}real$ A $x))$ $x = cond\text{-}exp$ M F $(\lambda x. norm$ $y * norm$ $(indicat\text{-}real$ A $x))$ x **using** $real\text{-}cond\text{-}exp\text{-}cmult$ $[of$ $\lambda x. norm$ $(indicat\text{-}real$ A $x)$ $norm$ $y]$ $indicator$ **by** *fastforce*

moreover **have** $(\lambda x. norm$ $y * norm$ $(indicat\text{-}real$ A $x)) = (\lambda x. norm$ $(indicat\text{-}real$ A $x *_R y))$ **by** *force*

ultimately **show** $?case$ **using** $*$ **by** *force*

next

case $(add$ u $v)$

have AE x in M . $norm$ $(cond\text{-}exp$ M F $(\lambda a. u$ $a + v$ $a)$ $x) = norm$ $(cond\text{-}exp$ M F u $x + cond\text{-}exp$ M F v $x)$ **using** $has\text{-}cond\text{-}exp\text{-}character$ $(2)[OF$ $has\text{-}cond\text{-}exp\text{-}add, OF$ $has\text{-}cond\text{-}exp\text{-}simple$ $(1,1), OF$ add $(1,2,3,4)]$ **by** *fastforce*

moreover **have** AE x in M . $norm$ $(cond\text{-}exp$ M F u $x + cond\text{-}exp$ M F v $x) \leq norm$ $(cond\text{-}exp$ M F u $x) + norm$ $(cond\text{-}exp$ M F v $x)$ **using** $norm\text{-}triangle\text{-}ineq$ **by** *blast*

moreover **have** AE x in M . $norm$ $(cond\text{-}exp$ M F u $x) + norm$ $(cond\text{-}exp$ M F v $x) \leq cond\text{-}exp$ M F $(\lambda x. norm$ $(u$ $x))$ $x + cond\text{-}exp$ M F $(\lambda x. norm$ $(v$ $x))$ x **using** add $(6,7)$ **by** *fastforce*

moreover **have** AE x in M . $cond\text{-}exp$ M F $(\lambda x. norm$ $(u$ $x))$ $x + cond\text{-}exp$ M F $(\lambda x. norm$ $(v$ $x))$ $x = cond\text{-}exp$ M F $(\lambda x. norm$ $(u$ $x) + norm$ $(v$ $x))$ x **using** $integrable\text{-}simple\text{-}function$ $[OF$ add $(1,2)]$ $integrable\text{-}simple\text{-}function$ $[OF$ add $(3,4)]$ **by** $(intro$ $has\text{-}cond\text{-}exp\text{-}character$ $(2)[OF$ $has\text{-}cond\text{-}exp\text{-}add[OF$ $has\text{-}cond\text{-}exp\text{-}character$ $(1,1)]$, $THEN$ $AE\text{-}symmetric]$, $auto$ $intro: has\text{-}cond\text{-}exp\text{-}real$

moreover **have** AE x in M . $cond\text{-}exp$ M F $(\lambda x. norm$ $(u$ $x) + norm$ $(v$ $x))$ $x = cond\text{-}exp$ M F $(\lambda x. norm$ $(u$ $x + v$ $x))$ x **using** add (5) $integrable\text{-}simple\text{-}function$ $[OF$ add $(1,2)]$ $integrable\text{-}simple\text{-}function$ $[OF$ add $(3,4)]$ **by** $(intro$ $cond\text{-}exp\text{-}cong, auto)$

ultimately **show** $?case$ **by** *force*

qed

lemma $has\text{-}cond\text{-}exp\text{-}simple\text{-}lim$:

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

assumes $integrable$ $[measurable]: integrable$ M f

and $\bigwedge i. simple\text{-}function$ M $(s$ $i)$

and $\bigwedge i. emeasure$ M $\{y \in space$ $M. s$ i $y \neq 0\} \neq \infty$

and $\bigwedge x. x \in space$ $M \implies (\lambda i. s$ i $x) \longrightarrow f$ x

and $\bigwedge x$ $i. x \in space$ $M \implies norm$ $(s$ i $x) \leq 2 * norm$ $(f$ $x)$

obtains r

where $strict\text{-}mono$ r $has\text{-}cond\text{-}exp$ M F f $(\lambda x. lim$ $(\lambda i. cond\text{-}exp$ M F $(s$ $(r$ $i)))$

x)

$AE x$ in M . *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** $(\text{simp add: borel-measurable-simple-function})$

have $\text{integrable-}s: \text{integrable } M (\lambda x. s i x)$ **for** i **using** $assms$ $\text{integrable-simple-function}$ **by** blast

have $\text{integrable-}4f: \text{integrable } M (\lambda x. 4 * \text{norm } (f x))$ **using** $assms(1)$ **by** simp

have $\text{integrable-}2f: \text{integrable } M (\lambda x. 2 * \text{norm } (f x))$ **using** $assms(1)$ **by** simp

have $\text{integrable-}2\text{-cond-exp-norm-}f: \text{integrable } M (\lambda x. 2 * \text{cond-exp } M F (\lambda x. \text{norm } (f x)) x)$ **by** fast

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

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

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

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

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

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

moreover **have** $(\lambda x. \text{diameter } \{s i x \mid i. n \leq i\}) \in \text{borel-measurable } M$ **for** n **using** $\text{bounded-range-}s \text{borel-measurable-diameter}$ **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 $(\text{intro diameter-le, blast, subst dist-norm}[symmetric], \text{intro dist-triangle3}[THEN$

order-trans, of 0], *intro add-mono*) (*auto intro: assms(5)[OF x]*)

hence *norm* (*diameter* {*s i x* | *i. n ≤ 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 $- 2 * \text{norm } (f -)$]

by (*intro Bochner-Integration.integrable-bound*[*OF integrable-4f*]) *fastforce+*

Since *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** $*$: *LINT* *x|M. diameter* {*s i x* | *i. n ≤ 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*} \times {*s i x* | *i. N ≤ i*}) = *case-prod* ($\lambda i j. \text{dist } (s i x) (s j x)$) '({*N..*} \times {*N..*}) **by** *fast*

hence *diameter* {*s i x* | *i. N ≤ i*} = (*SUP* (*i, j*) \in {*N..*} \times {*N..*}. *dist* (*s i x*) (*s j x*)) **unfolding** *diameter-def* **by** *auto*

moreover have (*SUP* (*i, j*) \in {*N..*} \times {*N..*}. *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 *diameter* {*s i x* | *i. N ≤ i*} $\geq \text{dist } (s i x) (s j x)$ **by** *presburger*

}

hence *LINT* *x|M. 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 *LINT* *x|M. norm* (*cond-exp* *M F* (*s i*) *x* - *cond-exp* *M F* (*s j*) *x*) $\leq \text{LINT } x|M. \text{dist } (s i x) (s j x)$ **for** *i j*

proof –
have $LINT\ x|M. norm\ (cond-exp\ M\ F\ (s\ i)\ x - cond-exp\ M\ F\ (s\ j)\ x) = LINT\ x|M. norm\ (cond-exp\ M\ F\ (s\ i)\ x + -1 *R\ cond-exp\ M\ F\ (s\ j)\ x)$ **unfolding**
dist-norm **by** *simp*
also have $... = LINT\ x|M. norm\ (cond-exp\ M\ F\ (\lambda x. s\ i\ x - s\ j\ x)\ x)$ **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 $... \leq LINT\ x|M. cond-exp\ M\ F\ (\lambda x. norm\ (s\ i\ x - s\ j\ x))\ x$ **using**
cond-exp-contraction-simple[OF - fin-sup, of i j] integrable-cond-exp assms(2) **by**
(*intro integral-mono-AE, fast+*)
also have $... = LINT\ x|M. norm\ (s\ i\ x - s\ j\ x)$ **unfolding** *set-integral-space(1)[OF*
integrable-cond-exp, symmetric] set-integral-space[OF dist-integrable[unfolding 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. 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. cond-exp M F (s (r n)) x) \longrightarrow lim*
(\lambda n. 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 * norm (f x)) x* **for** *n*

proof –
have *AE x in M. norm (cond-exp M F (s (r n)) x) \leq cond-exp M F (\lambda x. norm (s (r n) x)) x* **by** (*rule cond-exp-contraction-simple[OF assms(2,3)]*)
moreover have *AE x in M. real-cond-exp M F (\lambda x. norm (s (r n) x)) x \leq real-cond-exp M F (\lambda x. 2 * 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-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 expecta-

tion 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 ∈ sets F
  have AE x in M. norm (indicator A x *R cond-exp M F (s (r n)) x) ≤ cond-exp
M F (λx. 2 * norm (f x)) x for n
  proof -
    have AE x in M. norm (indicator A x *R cond-exp M F (s (r n)) x) ≤ norm
(cond-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: (λn. LINT x:A|M. cond-exp M F (s (r n)) x) →
(LINT x:A|M. lim (λn. cond-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. norm (indicator A x *R s (r n) x) ≤ 2 * norm (f x) for n
  proof -
    have AE x in M. norm (indicator A x *R s (r n) x) ≤ 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: (λn. LINT x:A|M. s (r n) x) → (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 (λn. cond-exp M F (s (r n)) x)) = lim (λn. LINT
x:A|M. cond-exp M F (s (r n)) x) using limI[OF lim-cond-exp-int] by argo
    also have ... = lim (λn. 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 $f :: 'a \Rightarrow 'b :: \{second-countable-topology, banach\}$

assumes *integrable M f*

shows *has-cond-exp M F f (cond-exp M F f)*

proof –

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

2.2 Properties

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

lemma *cond-exp-set-integral*:

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

assumes *integrable M f A ∈ sets F*

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

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

assumes *integrable M f subalgebra M G subalgebra G F*

shows $AE\ \xi\ in\ M. cond\text{-exp } M\ F\ f\ \xi = cond\text{-exp } M\ F\ (cond\text{-exp } M\ G\ f)\ \xi$

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

assumes *integrable M f integrable M g*

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

using *has-cond-exp-add[OF has-cond-expI(1,1), OF assms, THEN has-cond-exp-charact(2)]*

.

lemma *cond-exp-diff*:

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

assumes *integrable M f integrable M g*

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

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

assumes *integrable M f integrable M g*

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

assumes *integrable M f*

shows *AE x in M. norm (cond-exp M F f x) ≤ cond-exp M F (λx. norm (f x)) x*

proof –

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

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

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

hence *r'-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* $x \text{ in } M. (\lambda i. \text{norm } (\text{cond-exp } M F (s (r (r' i))) x)) \longrightarrow \text{norm } (\text{cond-exp } M F f x)$ **using** *r-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" [4].

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* $M A (\text{cond-exp } M F (\lambda x. f x * g x)) = \text{set-lebesgue-integral } M A (\lambda x. f x * g x)$ **by** (*simp add: cond-exp-set-integral[OF assms(1) asm]*)

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

}

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

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 in M . $(\lambda i. s i x *_R g x) \longrightarrow z x *_R g x$ **using** *s-is*(2) **by** (*simp add: space-restr-to-subalg tendsto-scaleR*)

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

have *s-scaleR-g-meas*: $(\lambda x. s i x *_R g x) \in \text{borel-measurable } M$ **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*: $(\lambda x. s i x *_R \text{cond-exp } M F g x) \in \text{borel-measurable } ?F$ **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 in M. norm (s i x *_R g x) ≤ 2 * norm (z x *_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: integrable M (λx. norm (z x) * norm (g x))* **using** *asm(1)* [*THEN integrable-norm*] **by** *simp*

have *AE x in ?F. norm (s i x *_R cond-exp M F g x) ≤ 2 * norm (z x) * norm (cond-exp M F g x)* **using** *s-is(3)* **by** (*fastforce simp add: mult-mono*)

moreover **have** *AE x in ?F. norm (z x) * cond-exp M F (λx. norm (g x)) x = cond-exp M F (λx. norm (z x) * 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 in ?F. norm (s i x *_R cond-exp M F g x) ≤ 2 * cond-exp M F (λx. norm (z x *_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 ∈ borel-measurable M* **by** (*meson borel-measurable-simple-function measurable-from-subalg s-is(1) subalg'*)

have *s-meas-F* [*measurable*]: *s i ∈ borel-measurable F* **by** (*meson borel-measurable-simple-function measurable-in-subalg' s-is(1) subalg*)

have *s-scaleR-eq*: *s i x *_R h x = (∑_{y∈s i ' space M. (indicator (s i - ' {y} ∩ space M) x *_R y) *_R h x)}* **if** *x ∈ space M* **for** *x* **and** *h :: 'a ⇒ 'b*

using *simple-function-indicator-representation-banach* [*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. (∑_{y∈s i ' space M. indicator (s i - ' {y} ∩ space M) x *_R y *_R g x)}* **using** *s-scaleR-eq* **by** (*intro Bochner-Integration.integral-cong*) *auto*

also **have** *... = (∑_{y∈s i ' space M. LINT x|M. indicator (s i - ' {y} ∩ 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** *... = (∑_{y∈s i ' space M. y *_R set-lebesgue-integral M (s i - ' {y} ∩ space M) g)}* **by** (*simp only: set-lebesgue-integral-def* [*symmetric*]) *simp*

also **have** *... = (∑_{y∈s i ' space M. y *_R set-lebesgue-integral M (s i - ' {y} ∩ space M) (cond-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** *... = (∑_{y∈s i ' space M. LINT x|M. indicator (s i - ' {y} ∩ space}*

$M) x *_R y *_R \text{ cond-exp } M F g x$ **by** (*simp only: set-lebesgue-integral-def[symmetric]*)
simp
also have $\dots = \text{LINT } x|M. (\sum y \in s \text{ i ' space } M. \text{ indicator } (s \text{ i} - \{y\} \cap \text{ space } M) x *_R y *_R \text{ cond-exp } M F g x)$ **by** (*intro Bochner-Integration.integral-sum[symmetric]*)
integrable-mult-indicator[OF - integrable-scaleR-right]) *auto*
also have $\dots = \text{LINT } x|M. s \text{ i } x *_R \text{ cond-exp } M F g x$ **using** *s-scaleR-eq*
by (*intro Bochner-Integration.integral-cong*) *auto*
finally have $\text{LINT } x|M. s \text{ i } x *_R g x = \text{LINT } x|?F. s \text{ i } x *_R \text{ cond-exp } M F g x$ **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* $M (\lambda x. z x *_R \text{ cond-exp } M F g x)$ **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 $(\lambda i. \text{LINT } x|M. s \text{ i } x *_R g x) \longrightarrow \text{LINT } x|M. z x *_R g x$ **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 $(\lambda i. \text{LINT } x|?F. s \text{ i } x *_R \text{ cond-exp } M F g x) \longrightarrow \text{LINT } x|?F. z x *_R \text{ cond-exp } M F g x$ **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 $\text{LINT } x|M. z x *_R g x = \text{LINT } x|M. z x *_R \text{ cond-exp } M F g x$ **using** *integral-s-eq* **using** *subalg* **by** (*simp add: LIMSEQ-unique integral-subalgebra2*)

qed

}

note $*$ = *this*

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

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

{

fix A **assume** $asm: A \in F$

hence *integrable* $M (\lambda x. \text{ indicat-real } A x *_R f x *_R g x)$ **using** *subalg* **by** (*fastforce simp add: subalgebra-def intro!: integrable-mult-indicator assms(1)*)

hence *set-lebesgue-integral* $M A (\lambda x. f x *_R g x) = \text{set-lebesgue-integral } M A (\lambda x. f x *_R \text{ cond-exp } M F g x)$ **unfolding** *set-lebesgue-integral-def* **using** asm **by** (*auto intro!: * measurable-in-subalg[OF subalg]*)

}

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

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 $f :: 't \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$

assumes [*measurable*]: $\bigwedge i. \text{integrable } M (f i)$

shows $\text{AE } x \text{ in } M. \text{cond-exp } M F (\lambda x. \sum_{i \in I}. f i x) x = (\sum_{i \in I}. \text{cond-exp } M F (f i) x)$

proof (*rule has-cond-exp-charact, intro has-cond-expI'*)

fix A **assume** [*measurable*]: $A \in \text{sets } F$

then have $A\text{-meas}$ [*measurable*]: $A \in \text{sets } M$ **by** (*meson subsetD subalg subalgebra-def*)

have $(\int x \in A. (\sum_{i \in I}. f i x) \partial M) = (\int x. (\sum_{i \in I}. \text{indicator } A x *_R f i x) \partial M)$

unfolding *set-lebesgue-integral-def* **by** (*simp add: scaleR-sum-right*)

also have $\dots = (\sum_{i \in I}. (\int x. \text{indicator } A x *_R f i x \partial M))$ **using** *assms* **by** (*auto intro! Bochner-Integration.integral-sum integrable-mult-indicator*)

also have $\dots = (\sum_{i \in I}. (\int x. \text{indicator } A x *_R \text{cond-exp } M F (f i) x \partial M))$ **using** *cond-exp-set-integral[OF assms]* **by** (*simp add: set-lebesgue-integral-def*)

also have $\dots = (\int x. (\sum_{i \in I}. \text{indicator } A x *_R \text{cond-exp } M F (f i) x) \partial M)$ **using** *assms* **by** (*auto intro! Bochner-Integration.integral-sum[symmetric] integrable-mult-indicator*)

also have $\dots = (\int x \in A. (\sum_{i \in I}. \text{cond-exp } M F (f i) x) \partial M)$ **unfolding** *set-lebesgue-integral-def* **by** (*simp add: scaleR-sum-right*)

finally show $(\int x \in A. (\sum_{i \in I}. f i x) \partial M) = (\int x \in A. (\sum_{i \in I}. \text{cond-exp } M F (f i) x) \partial M)$ **by** *auto*

qed (*auto simp add: assms integrable-cond-exp*)

2.3 Linearly Ordered Banach Spaces

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

lemma *cond-exp-gr-c*:

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

assumes *integrable M f AE x in M. f x > c*

shows $\text{AE } x \text{ in } M. \text{cond-exp } M F f x > c$

proof –

define X **where** $X = \{x \in \text{space } M. \text{cond-exp } M F f x \leq c\}$

have [*measurable*]: $X \in \text{sets } F$ **unfolding** $X\text{-def}$ **by** *measurable (metis sets.top subalg subalgebra-def)*

hence $X\text{-in-}M$: $X \in \text{sets } M$ **using** *sets-restr-to-subalg subalg subalgebra-def* **by** *blast*

have $\text{emeasure } M X = 0$

proof (*rule ccontr*)

assume $\text{emeasure } M X \neq 0$

have $\text{emeasure } (\text{restr-to-subalg } M F) X = \text{emeasure } M X$ **by** (*simp add: emeasure-restr-to-subalg subalg*)

hence $\text{emeasure } (\text{restr-to-subalg } M F) X > 0$ **using** $\langle \neg(\text{emeasure } M X) = 0 \rangle$
gr-zeroI **by** *auto*
then obtain A **where** $A: A \in \text{sets } (\text{restr-to-subalg } M F) A \subseteq X$ $\text{emeasure } (\text{restr-to-subalg } M F) A > 0$ $\text{emeasure } (\text{restr-to-subalg } M F) A < \infty$
using sigma-fin-subalg **by** $(\text{metis } \text{emeasure-notin-sets } \text{ennreal-0 } \text{infinity-ennreal-def } \text{le-less-linear } \text{neg-top-trans } \text{not-gr-zero } \text{order-refl } \text{sigma-finite-measure.approx-PInf-emeasure-with-finite})$
hence $[\text{simp}]: A \in \text{sets } F$ **using** $\text{subalg } \text{sets-restr-to-subalg}$ **by** *blast*
hence $A\text{-in-sets-}M[\text{simp}]: A \in \text{sets } M$ **using** $\text{sets-restr-to-subalg } \text{subalg } \text{subalgebra-def}$ **by** *blast*
have $[\text{simp}]: \text{set-integrable } M A (\lambda x. c)$ **using** A subalg **by** $(\text{auto } \text{simp } \text{add}: \text{set-integrable-def } \text{emeasure-restr-to-subalg})$
have $[\text{simp}]: \text{set-integrable } M A f$ **unfolding** $\text{set-integrable-def}$ **by** $(\text{rule } \text{integrable-mult-indicator}, \text{auto } \text{simp } \text{add}: \text{assms}(1))$
have $AE x \text{ in } M. \text{indicator } A x *_R c = \text{indicator } A x *_R f x$
proof $(\text{rule } \text{integral-eq-mono-AE-eq-AE})$
have $(\int x \in A. c \partial M) \leq (\int x \in A. f x \partial M)$ **using** $\text{assms}(2)$ **by** $(\text{intro } \text{set-integral-mono-AE-banach})$
auto
moreover
{
have $(\int x \in A. f x \partial M) = (\int x \in A. \text{cond-exp } M F f x \partial M)$ **by** $(\text{rule } \text{cond-exp-set-integral}, \text{auto } \text{simp } \text{add}: \text{assms})$
also have $\dots \leq (\int x \in A. c \partial M)$ **using** A **by** $(\text{auto } \text{intro}!: \text{set-integral-mono-banach } \text{simp } \text{add}: X\text{-def})$
finally have $(\int x \in A. f x \partial M) \leq (\int x \in A. c \partial M)$ **by** *simp*
}
ultimately show $LINT x|M. \text{indicator } A x *_R c = LINT x|M. \text{indicator } A x *_R f x$ **unfolding** $\text{set-lebesgue-integral-def}$ **by** *simp*
show $AE x \text{ in } M. \text{indicator } A x *_R c \leq \text{indicator } A x *_R f x$ **using** assms **by**
 $(\text{auto } \text{simp } \text{add}: X\text{-def } \text{indicator-def})$
qed $(\text{auto } \text{simp } \text{add}: \text{set-integrable-def}[\text{symmetric}])$
hence $AE x \in A \text{ in } M. c = f x$ **by** *auto*
hence $AE x \in A \text{ in } M. \text{False}$ **using** $\text{assms}(2)$ **by** *auto*
hence $A \in \text{null-sets } M$ **using** $AE\text{-iff-null-sets } A\text{-in-sets-}M$ **by** *metis*
thus False **using** $A(3)$ **by** $(\text{simp } \text{add}: \text{emeasure-restr-to-subalg } \text{null-sets}D1 \text{subalg})$
qed
thus *?thesis* **using** $AE\text{-iff-null-sets}[OF X\text{-in-}M]$ **unfolding** $X\text{-def}$ **by** *auto*
qed

corollary cond-exp-less-c :

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

assumes $\text{integrable } M f AE x \text{ in } M. f x < c$

shows $AE x \text{ in } M. \text{cond-exp } M F f x < c$

proof –

have $AE x \text{ in } M. \text{cond-exp } M F f x = - \text{cond-exp } M F (\lambda x. - f x) x$ **using** $\text{cond-exp-uminus}[OF \text{assms}(1)]$ **by** *auto*

moreover have $AE x \text{ in } M. \text{cond-exp } M F (\lambda x. - f x) x > - c$ **using** assms **by** $(\text{intro } \text{cond-exp-gr-c})$ *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 < 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 ≥ c*

shows *AE x in M. cond-exp M F f x ≥ 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 ∈ sets ?F 0 < measure ?F A*

have [*simp*]: *sets ?F = sets F measure ?F A = measure M A* **using** *asm* **by** (*auto simp add: measure-def sets-restr-to-subalg[OF subalg] emeasure-restr-to-subalg[OF subalg]*)

have $M-A$: *emeasure M A < ∞* **using** *measure-zero-top asm* **by** (*force simp add: top.not-eq-extremum*)

hence $F-A$: *emeasure ?F A < ∞* **using** *asm(1) emeasure-restr-to-subalg subalg* **by** *fastforce*

have *set-lebesgue-integral M A (λ-. c) ≤ 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 $\dots = \text{set-lebesgue-integral } M A (\text{cond-exp } M F f)$ **using** *cond-exp-set-integral[OF assms(1)] asm* **by** *auto*

also have $\dots = \text{set-lebesgue-integral } ?F A (\text{cond-exp } M F f)$ **unfolding** *set-lebesgue-integral-def* **using** *asm borel-measurable-cond-exp* **by** (*intro integral-subalgebra2[OF subalg, symmetric], simp*)

finally have $(1 / \text{measure } ?F A) *_{\mathbb{R}} \text{set-lebesgue-integral } ?F A (\text{cond-exp } M F f) \in \{c..\}$ **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 $\text{integrable } M f$
and $AE\ x\ \text{in } M. f\ x \leq c$
shows $AE\ x\ \text{in } M. \text{cond-exp } M\ F\ f\ x \leq c$
proof –
have $AE\ x\ \text{in } M. \text{cond-exp } M\ F\ f\ x = - \text{cond-exp } M\ F\ (\lambda x. - f\ x)\ x$ **using** $\text{cond-exp-uminus}[OF\ \text{assms}(1)]$ **by** force
moreover **have** $AE\ x\ \text{in } M. \text{cond-exp } M\ F\ (\lambda x. - f\ x)\ x \geq - c$ **using** assms
by $(\text{intro } \text{cond-exp-ge-c})\ \text{auto}$
ultimately show $?thesis$ **by** $(\text{force } \text{simp } \text{add: } \text{minus-le-iff})$
qed

corollary cond-exp-mono :

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

corollary cond-exp-min :

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

corollary cond-exp-max :

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

(*cond-exp* $M F f \xi$) (*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 $AE \xi$ in M . *cond-exp* $M F (\lambda x. \inf (f x) (g x)) \xi \leq \inf$ (*cond-exp* $M F f \xi$) (*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 $AE \xi$ in M . *cond-exp* $M F (\lambda x. \sup (f x) (g x)) \xi \geq \sup$ (*cond-exp* $M F f \xi$) (*cond-exp* $M F g \xi$)

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

end

2.4 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-exp* M (*sigma* (*space* M) $\{\}$) $f x = \text{expectation } f$

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 σ -algebras don't matter for the conditional expectation. The proof is adapted from [4].

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

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

assumes *subalgebra: subalgebra* $M F$ *subalgebra* $M G$

and *independent: indep-set* G (*sigma* (*space* M)) ($F \cup \text{vimage-algebra } (\text{space}$

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

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

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 $\text{borel-measurable-indicator}[OF A(3), \text{THEN measurable-sets}] \text{sets.top subalgebra}$
by $(\text{intro sets.sigma-sets-subset}') (\text{fastforce simp add}: \text{subalgebra-def})+$

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

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

by *blast*

have *set-lebesgue-integral* $M (space\ M - A) f = set-lebesgue-integral\ M (space\ M - A \cup A) f - set-lebesgue-integral\ M A f$

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* $M (space\ M - A \cup A) (cond-exp\ M\ F\ f) - set-lebesgue-integral\ M A (cond-exp\ M\ F\ f)$

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* $M (space\ M - A) (cond-exp\ M\ F\ f)$

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 ( $\bigcup$  (range A)) f = ( $\sum$  i. 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. set-lebesgue-integral M (A i) (cond-exp M F f)) using
union by presburger
    also have ... = set-lebesgue-integral M ( $\bigcup$  (range A)) (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 | a b. a  $\in$  F  $\wedge$  b  $\in$  G} = sigma (space
M) (F  $\cup$  G)
proof -
  have sigma-sets (space M) {a  $\cap$  b | a b. a  $\in$  sets F  $\wedge$  b  $\in$  sets G} = sigma-sets
(space M) (sets F  $\cup$  sets G)
  proof -
    {
      fix a b assume asm: a  $\in$  F b  $\in$  G
      hence a  $\cap$  b  $\in$  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  $\in$  sets F
    hence a  $\in$  sigma-sets (space M) {a  $\cap$  b | a b. a  $\in$  sets F  $\wedge$  b  $\in$  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  $\in$  sets F  $\vee$  a  $\in$  sets G a  $\notin$  sets F
    hence a  $\in$  sets G by blast
    hence a  $\in$  sigma-sets (space M) {a  $\cap$  b | a b. a  $\in$  sets F  $\wedge$  b  $\in$  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*) \in *borel-measurable (sigma (space M) (sets F \cup sets G))*
proof –
have $F \subseteq$ *sigma (space M) (F \cup 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 σ -algebra F , its conditional expectation $\text{cond-exp } M F f$ is just its expectation.

lemma (*in prob-space*) *cond-exp-indep*:
fixes $f :: 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach, real-normed-field}\}$
assumes *subalgebra: subalgebra M F*
and *independent: indep-set F (vimage-algebra (space M) f borel)*
and *integrable: integrable M f*
shows $\text{AE } x \text{ in } M. \text{cond-exp } M F f x = \text{expectation } f$
proof –
have *indep-set F (sigma (space M) (sigma (space M) $\{\}$) \cup (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 $\text{AE } x \text{ in } M. \text{cond-exp } M (\text{sigma (space M) (sigma (space M) $\{\}$ \cup F)) f x = \text{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

```

3 Filtered Measure Spaces

3.1 Filtered Measure

```

locale filtered-measure =
  fixes M F and t0 :: 'b :: {second-countable-topology, order-topology, t2-space}
  assumes subalgebras:  $\bigwedge i. t_0 \leq i \implies \text{subalgebra } M (F i)$ 
  and sets-F-mono:  $\bigwedge i j. t_0 \leq i \implies i \leq j \implies \text{sets } (F i) \leq \text{sets } (F j)$ 
begin

```

```

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

```

```

lemma sets-F-subset[simp]:
  assumes  $t_0 \leq i$ 
  shows  $\text{sets } (F i) \subseteq \text{sets } M$ 
  using subalgebras assms by (simp add: subalgebra-def)

```

```

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

```

```

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

```

end

```

locale linearly-filtered-measure = filtered-measure M F t0 for M and F :: - ::
  {linorder-topology, conditionally-complete-lattice}  $\implies$  - and t0

```

```

context linearly-filtered-measure
begin

```

σ -algebra at infinity

```

definition F-infinity :: 'a measure where
  F-infinity = sigma (space M) ( $\bigcup t \in \{t_0..\}. \text{sets } (F t)$ )

```

```

notation F-infinity ( $\langle F_\infty \rangle$ )

```

lemma *space-F-infinity[simp]*: *space* $F_\infty = \text{space } M$ **unfolding** *F-infinity-def space-measure-of-conv*
by *simp*

lemma *sets-F-infinity*: *sets* $F_\infty = \text{sigma-sets } (\text{space } M) (\bigcup t \in \{t_0..\}. \text{sets } (F t))$
unfolding *F-infinity-def* **using** *sets.space-closed[of F -] space-F* **by** (*blast intro!*:
sets-measure-of)

lemma *subset-F-infinity*:
assumes $t \geq t_0$
shows $F t \subseteq F_\infty$ **unfolding** *sets-F-infinity* **using** *assms* **by** *blast*

lemma *F-infinity-subset*: $F_\infty \subseteq M$
unfolding *sets-F-infinity* **using** *sets-F-subset*
by (*simp add: SUP-le-iff sets.sigma-sets-subset*)

lemma *F-infinity-measurableI*:
assumes $t \geq t_0$ $f \in \text{borel-measurable } (F t)$
shows $f \in \text{borel-measurable } (F_\infty)$
by (*metis assms borel-measurable-subalgebra space-F space-F-infinity subset-F-infinity*)

end

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

3.2 σ -Finite Filtered Measure

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

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

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

locale *nat-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure* $M F 0 ::$
 nat **for** $M F$

locale *enat-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure* $M F 0 ::$
 enat **for** $M F$

locale *real-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure* $M F 0$::
real for $M F$

locale *ennreal-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure* $M F 0$
 :: *ennreal for* $M F$

sublocale *nat-sigma-finite-filtered-measure* \subseteq *nat-filtered-measure* ..

sublocale *enat-sigma-finite-filtered-measure* \subseteq *enat-filtered-measure* ..

sublocale *real-sigma-finite-filtered-measure* \subseteq *real-filtered-measure* ..

sublocale *ennreal-sigma-finite-filtered-measure* \subseteq *ennreal-filtered-measure* ..

sublocale *nat-sigma-finite-filtered-measure* \subseteq *sigma-finite-subalgebra* $M F i$ **by**
blast

sublocale *enat-sigma-finite-filtered-measure* \subseteq *sigma-finite-subalgebra* $M F i$ **by**
fastforce

sublocale *real-sigma-finite-filtered-measure* \subseteq *sigma-finite-subalgebra* $M F |i|$ **by**
fastforce

sublocale *ennreal-sigma-finite-filtered-measure* \subseteq *sigma-finite-subalgebra* $M F i$ **by**
fastforce

3.3 Finite Filtered Measure

locale *finite-filtered-measure* = *filtered-measure* + *finite-measure*

sublocale *finite-filtered-measure* \subseteq *sigma-finite-filtered-measure*

using *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 *nat-finite-filtered-measure* = *finite-filtered-measure* $M F 0$:: **nat for** $M F$

locale *enat-finite-filtered-measure* = *finite-filtered-measure* $M F 0$:: **enat for** $M F$

locale *real-finite-filtered-measure* = *finite-filtered-measure* $M F 0$:: **real for** $M F$

locale *ennreal-finite-filtered-measure* = *finite-filtered-measure* $M F 0$:: **ennreal for**
 $M F$

sublocale *nat-finite-filtered-measure* \subseteq *nat-sigma-finite-filtered-measure* ..

sublocale *enat-finite-filtered-measure* \subseteq *enat-sigma-finite-filtered-measure* ..

sublocale *real-finite-filtered-measure* \subseteq *real-sigma-finite-filtered-measure* ..

sublocale *ennreal-finite-filtered-measure* \subseteq *ennreal-sigma-finite-filtered-measure* ..

3.4 Constant Filtration

lemma *filtered-measure-constant-filtration*:

assumes *subalgebra* $M F$

shows *filtered-measure* $M (\lambda-. F) t_0$

using *assms by* (*unfold-locales*) *blast+*

sublocale *sigma-finite-subalgebra* \subseteq *constant-filtration: sigma-finite-filtered-measure*
 $M \lambda- :: 't :: \{\text{second-countable-topology, linorder-topology}\}. F t_0$

using *subalg by* (*unfold-locales*) *blast+*

lemma (**in** *finite-measure*) *filtered-measure-constant-filtration*:

```

assumes subalgebra  $M F$ 
shows finite-filtered-measure  $M (\lambda-. F) t_0$ 
using assms by (unfold-locales) blast+

```

end

```

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

```

4 Stochastic Processes

4.1 Stochastic Process

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

```

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

```

```

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

```

```

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

```

end

```

lemma stochastic-process-const-fun:
  assumes  $f \in borel-measurable M$ 
  shows stochastic-process  $M t_0 (\lambda-. f)$  using assms by (unfold-locales)

```

```

lemma stochastic-process-const:
  shows stochastic-process  $M t_0 (\lambda i -. c i)$  by (unfold-locales) simp

```

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

```

context stochastic-process
begin

```

```

lemma compose-stochastic:
  assumes  $\bigwedge i. t_0 \leq i \implies f i \in borel-measurable borel$ 
  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  $M t_0 (\lambda i \xi. norm (X i \xi))$  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 *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*)

4.1.1 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* :: $'a \text{ measure} \Rightarrow 'b \Rightarrow ('b \Rightarrow 'a \Rightarrow 'c :: \text{topologi-}$

cal-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]*: *space* (*natural-filtration* M t_0 X t) = *space* M **unfolding** *natural-filtration-def space-family-vimage-algebra ..*

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

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

lemma *sets-natural-filtration'*:

assumes *borel* = *sigma UNIV S*

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

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

fix i **and** $A ::$ 'a 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 σ -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 $\wedge a. a \in A \implies \text{emeasure } M a \neq \infty$ $\wedge a. a \in A \implies \exists b \in \text{borel. } a = X t_0 - ' b \cap \text{space } M$

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

proof (unfold-locales)

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

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

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

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

by (auto simp add: sets-restr-to-subalg[OF subalgebra-natural-filtration] emeasure-restr-to-subalg[OF subalgebra-natural-filtration])

ultimately show $\exists A. \text{countable } A \wedge A \subseteq \text{sets (restr-to-subalg } M \text{ (natural-filtration } M t_0 X t_0)) \wedge \bigcup A = \text{space (restr-to-subalg } M \text{ (natural-filtration } M t_0 X t_0)) \wedge (\forall a \in A. \text{emeasure (restr-to-subalg } M \text{ (natural-filtration } M t_0 X t_0)) a \neq \infty)$ **using** exhausting-set **by** blast

show $\bigwedge i j. \llbracket t_0 \leq i; i \leq j \rrbracket \implies \text{sets (natural-filtration } M t_0 X i) \subseteq \text{sets (natural-filtration } M t_0 X j)$ **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_0 X) t_0
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 $t_0 \leq s < t$ *indep-vars* (λ -. *borel*) X $\{t_0..s\}$
shows *indep-set* (*natural-filtration* M t_0 X s) (*vimage-algebra* (*space* M) (X t) *borel*)
proof –
have *indep-sets* (λ i . $\{X$ i – ‘ $A \cap$ *space* M | A . $A \in$ *sets borel*}) (\bigcup (*range* (*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* λ i . $\{X$ i – ‘ $A \cap$ *space* M | A . $A \in$ *borel* } *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

4.2 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_0 **for** M F t_0 **and** X :: $- \Rightarrow - \Rightarrow -$
:: {*second-countable-topology, banach*} +
assumes *adapted*[*measurable*]: $\bigwedge i$. $t_0 \leq i \Rightarrow X$ $i \in$ *borel-measurable* (F i)
begin

lemma *adaptedE*[*elim*]:
assumes $\llbracket \bigwedge j$ i . $t_0 \leq j \Rightarrow j \leq i \Rightarrow X$ $j \in$ *borel-measurable* (F i) $\rrbracket \Rightarrow P$
shows P
using *assms* **using** *adapted* **by** (*metis* *dual-order.trans* *borel-measurable-subalgebra* *sets-F-mono* *space-F*)

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

end

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* $M \ F \ t_0 \ (\lambda i \ -. \ c \ i)$ **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 $\lambda \cdot. -1$] **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** $t_0 \leq j \leq i$ **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** $t_0 \leq j < i$ **for** j **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 lemmas which will be useful later on.

lemma (*in nat-filtered-measure*) *partial-sum-Suc-adapted:*

assumes *adapted-process* $M F 0 X$
shows *adapted-process* $M F 0 (\lambda n \xi. \sum_{i < n} X (Suc i) \xi)$
proof (*unfold-locales*)

interpret *adapted-process* $M F 0 X$ **using** *assms* **by** *blast*
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 enat-filtered-measure*) *partial-sum-eSuc-adapted:*

assumes *adapted-process* $M F 0 X$
shows *adapted-process* $M F 0 (\lambda n \xi. \sum_{i < n} X (eSuc i) \xi)$
proof (*unfold-locales*)

interpret *adapted-process* $M F 0 X$ **using** *assms* **by** *blast*
fix i
have $X (eSuc j) \in \text{borel-measurable } (F i)$ **if** $j < i$ **for** j **using** *that adaptedD* **by**
(simp add: ileI1)
thus $(\lambda \xi. \sum_{i < i} X (eSuc i) \xi) \in \text{borel-measurable } (F i)$ **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* $M F t_0 X i$ **using** *assms* **by** *simp*
have $X i k \in \text{borel-measurable } M X i k \in \text{borel-measurable } (F k)$ **using** *measurable-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* \subseteq *stochastic-process* **using** *measurable-from-subalg sub-algebras adapted* **by** (*unfold-locales*) *blast*

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*)

4.3 Progressively Measurable Process

locale *progressive-process* = *filtered-measure M F t₀* **for** *M F t₀* **and** *X* :: - \Rightarrow -
 \Rightarrow - :: {*second-countable-topology, banach*} +

assumes *progressive[measurable]:* $\bigwedge t. t_0 \leq t \implies (\lambda(i, x). X i x) \in \text{borel-measurable}$
(*restrict-space borel {t₀..t}*) $\otimes_M F t$)

begin

lemma *progressiveD:*

assumes *S* \in *borel*

shows $(\lambda(j, \xi). X j \xi) - ' S \cap (\{t_0..i\} \times \text{space } M) \in (\text{restrict-space borel } \{t_0..i\})$
 $\otimes_M F i$)

using *measurable-sets[OF progressive, OF - assms, of i]*

by (*cases t₀ \leq i*) (*auto simp add: space-restrict-space sets-pair-measure space-pair-measure*)

end

lemma (**in** *filtered-measure*) *progressive-process-const-fun:*

assumes *f* \in *borel-measurable (F t₀)*

shows *progressive-process M F t₀ (λ -. f)*

proof (*unfold-locales*)

fix *i* **assume** *asm: t₀ \leq i*

have *f* \in *borel-measurable (F i)* **using** *borel-measurable-mono[OF order.refl asm]*
assms **by** *blast*

thus *case-prod (λ -. f) \in borel-measurable (restrict-space borel {t₀..i}) $\otimes_M F i$*

using *measurable-compose[OF measurable-snd]* **by** *simp*

qed

lemma (**in** *filtered-measure*) *progressive-process-const:*

assumes *c* \in *borel-measurable borel*

shows *progressive-process M F t₀ (λ i -. c i)*

using *assms* **by** (*unfold-locales*) (*auto simp add: measurable-split-conv intro!: measurable-compose[OF measurable-fst] measurable-restrict-space1*)

context *progressive-process*

begin

lemma *compose-progressive*:

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

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

proof

fix i **assume** *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 *progressive[OF asm] measurable-fst''[OF measurable-restrict-space1, OF measurable-id]*

by (*auto simp add: measurable-pair-iff measurable-split-conv*)

moreover have $(\lambda(j, \xi). f j (X j \xi)) = \text{case-prod } f \circ ((\lambda(j, y). (j, y)) \circ (\lambda(j, \xi). (j, X j \xi)))$ **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** *assms* **by** (*simp add: borel-prod*)

qed

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

lemma *scaleR-right-progressive*:

assumes *progressive-process* $M F t_0 R$

shows *progressive-process* $M F t_0 (\lambda i \xi. (R i \xi) *_R (X i \xi))$

using *progressive-process.progressive[OF assms]* **by** (*unfold-locales (simp add: progressive assms)*)

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

assumes $f \in \text{borel-measurable } (F t_0)$

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

using *assms* **by** (*fast intro: scaleR-right-progressive progressive-process-const-fun*)

lemma *scaleR-right-const-progressive*:

assumes $c \in \text{borel-measurable borel}$

shows *progressive-process* $M F t_0 (\lambda i \xi. c i *_R (X i \xi))$

using *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* $\lambda \cdot -1$] **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*] \uparrow

unfolding *prod.case space-restrict-space*
by *unfold-locales simp*

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

theorem (**in** *nat-filtered-measure*) *progressive-iff-adapted: progressive-process* $M F 0 X \longleftrightarrow$ *adapted-process* $M F 0 X$

proof (*intro iffI*)

assume *asm: progressive-process* $M F 0 X$

interpret *progressive-process* $M F 0 X$ **by** (*rule asm*)

show *adapted-process* $M F 0 X$..

next

assume *asm: adapted-process* $M F 0 X$

interpret *adapted-process* $M F 0 X$ **by** (*rule asm*)

show *progressive-process* $M F 0 X$

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

}

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

moreover have *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 *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

theorem (**in** *enat-filtered-measure*) *progressive-iff-adapted: progressive-process* $M F 0 X \longleftrightarrow$ *adapted-process* $M F 0 X$

proof (*intro iffI*)
assume *asm: progressive-process M F 0 X*
interpret *progressive-process M F 0 X* **by** (*rule asm*)
show *adapted-process M F 0 X ..*
next
assume *asm: adapted-process M F 0 X*
interpret *adapted-process M F 0 X* **by** (*rule asm*)
show *progressive-process M F 0 X*
proof (*unfold-locales, intro borel-measurableI*)
fix *S :: 'b set and i :: enat assume open-S: open S*
{
fix *j assume asm: j ≤ 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** $\text{case-prod } X -' S \cap \{j\} \times \text{space } M = \{j\} \times (X j -' S \cap \text{space } M)$ **for** *j* **by** *fast*
moreover **have** $\{j :: enat\} \in \text{restrict-space borel } \{0..i\}$ **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

4.4 Predictable Process

We introduce the constant Σ_P to denote the predictable σ -algebra.

context *linearly-filtered-measure*
begin

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

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

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 $snd \in \Sigma_P \rightarrow_M F \ t_0$
proof (*intro measurableI*)
fix $S :: 'a \ set$ **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** $snd - ' S \cap \ space \ \Sigma_P = \{t_0..\} \times S$ **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 $snd - ' S \cap \ space \ \Sigma_P \in \Sigma_P$ **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 $fst \in \Sigma_P \rightarrow_M \ borel$
proof –
have $A \times \ space \ M \in \ sets \ \Sigma_P$ **if** $A \in \ sigma\text{-sets} \ \{t_0..\} \ \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$ **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 $(\{t_0..\} - a) \times \ space \ M = \{t_0..\} \times \ space \ M - a \times \ space \ M$ **by** *blast*
then show *?case* **using** *Compl(2)*[*THEN sigma-sets.Compl*] **by** *presburger*
next
case (*Union a*)
have $\bigcup (\ range \ a) \times \ space \ M = \bigcup (\ range \ (\lambda i. a \ i \times \ space \ M))$ **by** *blast*
then show *?case* **using** *Union(2)*[*THEN sigma-sets.Union*] **by** *presburger*
qed (*auto*)
moreover **have** *restrict-space borel* $\{t_0..\} = \ sigma \ \{t_0..\} \ \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$
proof –
have *sigma-sets* $\{t_0..\} \ ((\cap) \ \{t_0..\} \ ' \ sigma\text{-sets} \ UNIV \ (\ range \ greaterThan)) =$
sigma-sets $\{t_0..\} \ \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$
proof (*intro sigma-sets-eqI* ; *clarify*)
fix $A :: 'b \ set$ **assume** *asm*: $A \in \ sigma\text{-sets} \ UNIV \ (\ range \ greaterThan)$
thus $\{t_0..\} \cap A \in \ sigma\text{-sets} \ \{t_0..\} \ \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$
proof (*induction rule: sigma-sets.induct*)
case (*Basic a*)
then obtain s **where** $s: a = \{s<..\}$ **by** *blast*
show *?case*

```

proof (cases  $t_0 \leq s$ )
  case True
    hence *:  $\{t_0..\} \cap a = (\bigcup i \in \mathcal{I}. \{s<..\} \cap i)$  using  $s$  assms(3) by force
    have  $((\cap) \{s<..\} \text{ ' } \mathcal{I}) \subseteq \text{sigma-sets } \{t_0..\} \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$ 
    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  $A$  True by linarith
  moreover have  $\max s \ s' < t'$  if  $s < t'$  using  $A$  that by linarith
  moreover have  $\{s<..\} \cap A = \{\}$  if  $\neg s < t'$  using  $A$  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 * 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  $\{t_0..\} \cap (\text{UNIV} - a) = \{t_0..\} - (\{t_0..\} \cap a)$  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:  $t_0 \leq s \ s < t$ 
  hence *:  $\{s<..t\} = \{s<..\} \cap (\{t_0..\} - \{t<..\})$  by force
  have  $\{s<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets UNIV (range greaterThan)})$ 
using asm by (intro sigma-sets.Basic) auto
  moreover have  $\{t_0..\} - \{t<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets$ 
UNIV (range greaterThan)) using asm by (intro sigma-sets.Compl sigma-sets.Basic)
  auto
  ultimately show  $\{s<..t\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets UNIV$ 
(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  $\{t_0..\} \otimes_M \text{sigma (space } M) \{\} \subseteq \text{sets } \Sigma_P$ 

unfolding sets-pair-measure space-restrict-space space-measure-of-conv

```

using *space-predictable-sigma sets.sigma-algebra-axioms*[of Σ_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 *sigma* (*space* M) $\{\}$) = *space* Σ_P **by** (*simp add: space-pair-measure*)
moreover have *fst* \in *restrict-space borel* $\{t_0..\}$ \otimes_M *sigma* (*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** $X ::$
 $- \Rightarrow - \Rightarrow - :: \{\text{second-countable-topology, banach}\} +$
assumes *predictable*: ($\lambda(t, x). X t x \in$ *borel-measurable* Σ_P)
begin

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

end

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* ($\lambda x. \{\text{Suc } x\}$)]) (*force | simp add: greaterThan-0*) $+$

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

lemma (**in** *enat-filtered-measure*) *measurable-predictable-sigma-snd'*:
shows *snd* \in $\Sigma_P \rightarrow_M F 0$
by (*intro measurable-predictable-sigma-snd*[*of* $\{\{0 < .. \infty\}\}$]) *force* $+$

lemma (**in** *enat-filtered-measure*) *measurable-predictable-sigma-fst'*:
shows *fst* \in $\Sigma_P \rightarrow_M$ *borel*
by (*intro measurable-predictable-sigma-fst*[*of* $\{\{0 < .. \infty\}\}$]) *force* $+$

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* ($\lambda x :: \text{nat. } \{0 < .. \text{real } (\text{Suc } x)\}$)]) (*fastforce intro: add-increasing*) $+$

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

lemma (in *ennreal-filtered-measure*) *measurable-predictable-sigma-snd'*:
shows $snd \in \Sigma_P \rightarrow_M F 0$
by (*intro measurable-predictable-sigma-snd*[of $\{\{0 < .. \infty\}\}$]) *force+*

lemma (in *ennreal-filtered-measure*) *measurable-predictable-sigma-fst'*:
shows $fst \in \Sigma_P \rightarrow_M \text{borel}$
by (*intro measurable-predictable-sigma-fst*[of $\{\{0 < .. \infty\}\}$]) *force+*

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* $M F t_0 (\lambda-. f)$
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 *predictable-process* $M F 0 (\lambda-. f)$
using *assms* **by** (*intro predictable-process-const-fun*[*OF measurable-predictable-sigma-snd'*])

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

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

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

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-locales*) (*simp add: measurable-split-conv*)

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

lemma (in *nat-filtered-measure*) *predictable-process-const'*[*intro*]:
assumes $c \in \text{borel-measurable borel}$

shows *predictable-process* $M F 0 (\lambda i -. c i)$
using *assms* **by** (*intro predictable-process-const*[*OF measurable-predictable-sigma-fst*'])

lemma (*in enat-filtered-measure*) *predictable-process-const'*[*intro*]:
assumes $c \in \text{borel-measurable borel}$
shows *predictable-process* $M F 0 (\lambda i -. c i)$
using *assms* **by** (*intro predictable-process-const*[*OF measurable-predictable-sigma-fst*'])

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

lemma (*in ennreal-filtered-measure*) *predictable-process-const'*[*intro*]:
assumes $c \in \text{borel-measurable borel}$
shows *predictable-process* $M F 0 (\lambda i -. c i)$
using *assms* **by** (*intro predictable-process-const*[*OF measurable-predictable-sigma-fst*'])

context *predictable-process*
begin

lemma *compose-predictable*:
assumes $\text{fst} \in \text{borel-measurable } \Sigma_P \text{ case-prod } f \in \text{borel-measurable borel}$
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** $(\lambda(i, \xi). f i (X i \xi)) = \text{case-prod } f \circ (\lambda(i, \xi). (i, X i \xi))$ **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-locales*) (*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-locales*)
(*auto simp add: measurable-split-conv*)

lemma *scaleR-right-const-fun-predictable*:
assumes $\text{snd} \in \Sigma_P \rightarrow_M F t_0 f \in \text{borel-measurable } (F t_0)$
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 $\text{predictable-process } M \ F \ t_0 \ (\lambda i \ \xi. \ c \ i \ *_R \ (X \ i \ \xi))$
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 $\text{predictable-process } M \ F \ t_0 \ Y$
shows $\text{predictable-process } M \ F \ t_0 \ (\lambda i \ \xi. \ X \ i \ \xi + Y \ i \ \xi)$
using *predictable predictable-process.predictable[OF assms]* **by** (*unfold-locales*)
(auto simp add: measurable-split-conv)

lemma *diff-predictable:*
assumes $\text{predictable-process } M \ F \ t_0 \ Y$
shows $\text{predictable-process } M \ F \ t_0 \ (\lambda i \ \xi. \ X \ i \ \xi - Y \ i \ \xi)$
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 ≤ 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** $S\text{-is}(2)$ *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 $\{\{s<..t\} \times A \mid A \text{ s t. } A \in \text{sets } (F s) \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in \text{sets } (F t_0)\} \subseteq \text{Pow } (\{t_0..\} \times \text{space } M)$ **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

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 $A (Suc i) \in F i \ A \ 0 \in F \ 0$
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** $S: (\bigcup i. \{i\} \times A i) = \{0\} \times S$ **by** *blast*
hence $S \in F \ 0$ **using** *Basic* **by** (*fastforce simp add: times-eq-iff*)
moreover have $A \ i = \{\}$ **if** $i \neq 0$ **for** i **using** *that S unfolding bot-nat-def[symmetric]*
by *blast*
moreover have $A \ 0 = S$ **using** S **by** *blast*
ultimately have $A \ 0 \in F \ 0 \ A \ (Suc \ i) \in F \ i$ **for** i **by** *auto*
}
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 $A \ 0 \in F \ 0 \ A \ (Suc \ i) \in F \ i$ **for** i **using** B *sets-F-mono*
by (*simp, metis less-Suc-eq-le sets.empty-sets subset-eq bot-nat-0.extremum greaterThanAtMost-iff*)
}
note $**$ = *this*
show $A \ (Suc \ i) \in \text{sets } (F \ i) \ A \ 0 \in F \ 0$ **using** $*(2)[of \ i] \ *(1) \ ***(2)[of \ i] \ ***(1)$
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:  $a \subseteq \{0..\} \times \text{space } M$  using Compl(1) sets.sets-into-space sets-predictable-sigma
  space-predictable-sigma by metis
  hence A-in:  $A \ i \subseteq \text{space } M$  for  $i$  using Compl(4) by blast
  have a:  $a = \{0..\} \times \text{space } M - (\bigcup i. \{i\} \times A \ i)$  using a-in Compl(4) by blast
  also have ... =  $-(\bigcap j. -(\{j\} \times (\text{space } M - A \ j)))$  by blast
  also have ... =  $(\bigcup j. \{j\} \times (\text{space } M - A \ j))$  by blast
  finally have *:  $(\text{space } M - A \ (\text{Suc } i)) \in F \ i \ (\text{space } M - A \ 0) \in F \ 0$  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:  $a \ i \subseteq \{0..\} \times \text{space } M$  for  $i$  using Union(1) sets.sets-into-space
  sets-predictable-sigma space-predictable-sigma by metis
    hence A-in:  $A \ i \subseteq \text{space } M$  for  $i$  using Union(4) by blast
    have snd  $x \in \text{snd } ' (a \ i \cap (\{fst \ x\} \times \text{space } M))$  if  $x \in a \ i$  for  $i \ x$  using that
  a-in by fastforce
    hence a-i:  $a \ i = (\bigcup j. \{j\} \times (\text{snd } ' (a \ i \cap (\{j\} \times \text{space } M))))$  for  $i$  by force
    have A-i:  $A \ i = \text{snd } ' (\bigcup (\text{range } a) \cap (\{i\} \times \text{space } M))$  for  $i$  unfolding
  Union(4) using A-in by force
    have *:  $\text{snd } ' (a \ j \cap (\{\text{Suc } i\} \times \text{space } M)) \in F \ i \ \text{snd } ' (a \ j \cap (\{0\} \times \text{space } M))$ 
   $\in F \ 0$  for  $j$  using Union(2,3)[OF a-i] by auto
    {
      case 1
      have  $(\bigcup j. \text{snd } ' (a \ j \cap (\{\text{Suc } i\} \times \text{space } M))) \in F \ i$  using * by fast
      moreover have  $(\bigcup j. \text{snd } ' (a \ j \cap (\{\text{Suc } i\} \times \text{space } M))) = \text{snd } ' (\bigcup (\text{range }
  a) \cap (\{\text{Suc } i\} \times \text{space } M))$  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-filtered-measure*) *predictable-implies-adapted-Suc*:
assumes *predictable-process* $M F 0 X$
shows *adapted-process* $M F 0 (\lambda i. X (Suc i))$
proof (*unfold-locales, intro borel-measurableI*)
interpret *predictable-process* $M F 0 X$ **by** (*rule assms*)
fix $S :: 'b$ set **and** i **assume** *open-S*: $open S$
have $\{Suc i\} = \{i <.. Suc i\}$ **by** *fastforce*
hence $\{Suc i\} \times space M \in \Sigma_P$ **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** $case-prod X -' S \cap (\{Suc i\} \times space M) \in \Sigma_P$ **unfolding**
sets-predictable-sigma **using** *space-F sets.sets-into-space*
by (*subst Times-Int-distrib1[of {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** $case-prod X -' S \cap (\{Suc i\} \times space M) = \{Suc i\} \times (X (Suc$
 $i) -' S \cap space M)$ **by** (*auto simp add: le-Suc-eq*)
moreover **have** $\dots = (\bigcup j. \{j\} \times (if j = Suc i then (X (Suc i) -' S \cap space M)$
 $else \{\}))$ **by** (*force split: if-splits*)
ultimately **have** $(\bigcup j. \{j\} \times (if j = Suc i then (X (Suc i) -' S \cap space M) else$
 $\{\})) \in \Sigma_P$ **by** *argo*
thus $X (Suc i) -' S \cap space (F i) \in sets (F i)$ **using** *sets-in-filtration[of $\lambda j.$*
if j = Suc i then (X (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 (in *nat-filtered-measure*) *predictable-process-iff*: *predictable-process* $M F 0 X \longleftrightarrow$ *adapted-process* $M F 0 (\lambda i. X (Suc i)) \wedge X 0 \in borel-measurable (F 0)$
proof (*intro iffI*)
assume *asm*: *adapted-process* $M F 0 (\lambda i. X (Suc i)) \wedge X 0 \in borel-measurable (F 0)$
interpret *adapted-process* $M F 0 \lambda i. X (Suc i)$ **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  $(\lambda(x, y). X x y) - ' S \cap \text{space } \Sigma_P = (\bigcup i. \{i\} \times (X i - ' S \cap \text{space } M))$  by fastforce
  ultimately show  $(\lambda(x, y). X x y) - ' S \cap \text{space } \Sigma_P \in \text{sets } \Sigma_P$  by simp
  qed
  thus predictable-process M F 0 X by (unfold-locales)
next
  assume asm: predictable-process M F 0 X
  interpret predictable-process M F 0 X using asm by blast
  show adapted-process M F 0  $(\lambda i. X (Suc i)) \wedge X 0 \in \text{borel-measurable } (F 0)$ 
using predictable-implies-adapted-Suc asm by auto
qed

corollary (in nat-filtered-measure) predictable-processI[intro!]:
  assumes  $X 0 \in \text{borel-measurable } (F 0) \wedge i. X (Suc i) \in \text{borel-measurable } (F i)$ 
  shows predictable-process M F 0 X
  unfolding predictable-process-iff
  using assms
  by (meson adapted-process.intro adapted-process-axioms-def filtered-measure-axioms)

end

```

```

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

```

5 Martingales

The following locales are necessary for defining martingales.

5.1 Martingale

A martingale is an adapted process where the expected value of the next observation, given all past observations, is equal to the current value.

```

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

```

```

locale martingale-order = martingale M F t_0 X for M F t_0 and X :: -  $\Rightarrow$  -  $\Rightarrow$  -
  :: {order-topology, ordered-real-vector}
locale martingale-linorder = martingale M F t_0 X for M F t_0 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* $M f f \in \text{borel-measurable } (F t_0)$
shows *martingale* $M F t_0 (\lambda-. f)$
using *assms sigma-finite-subalgebra.cond-exp-F-meas*[*OF - assms(1)*], *THEN AE-symmetric*
borel-measurable-mono
by (*unfold-locales*) *blast+*

lemma (in *sigma-finite-filtered-measure*) *martingale-cond-exp*[intro]:
assumes *integrable* $M f$
shows *martingale* $M F t_0 (\lambda i. \text{cond-exp } M (F i) f)$
using *sigma-finite-subalgebra.borel-measurable-cond-exp'* *borel-measurable-cond-exp*

by (*unfold-locales*) (*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* $M F t_0 (\lambda-. 0)$ **by** *fastforce*

corollary (in *finite-filtered-measure*) *martingale-const*[intro]: *martingale* $M F t_0 (\lambda-. c)$ **by** *fastforce*

5.2 Submartingale

A submartingale is an adapted process where the expected value of the next observation, given all past observations, is greater than or equal to the current value.

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

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

lemma (in *sigma-finite-filtered-measure*) *submartingale-const-fun*[intro]:
assumes *integrable* $M f f \in \text{borel-measurable } (F t_0)$
shows *submartingale* $M F t_0 (\lambda-. f)$
proof –
interpret *martingale* $M F t_0 \lambda-. f$ **using** *assms* **by** (*rule martingale-const-fun*)
show *submartingale* $M F t_0 (\lambda-. f)$ **using** *martingale-property* **by** (*unfold-locales*)
(*force simp add: integrable*)
qed

lemma (in *sigma-finite-filtered-measure*) *submartingale-cond-exp*[intro]:
assumes *integrable* $M f$
shows *submartingale* $M F t_0 (\lambda i. \text{cond-exp } M (F i) f)$
proof –

interpret *martingale* $M F t_0 \lambda i. \text{cond-exp } M (F i) f$ **using** *assms* **by** (*rule martingale-cond-exp*)

show *submartingale* $M F t_0 (\lambda i. \text{cond-exp } M (F i) f)$ **using** *martingale-property* **by** (*unfold-locales*) (*force simp add: integrable*)+
qed

corollary (**in** *finite-filtered-measure*) *submartingale-const*[*intro*]: *submartingale* $M F t_0 (\lambda-. c)$ **by** *fastforce*

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

sublocale *martingale-linorder* \subseteq *submartingale-linorder* ..

5.3 Supermartingale

A supermartingale is an adapted process where the expected value of the next observation, given all past observations, is less than or equal to the current value.

locale *supermartingale* = *sigma-finite-filtered-measure* $M F t_0$ + *adapted-process* $M F t_0 X$ **for** $M F t_0$ **and** $X :: - \Rightarrow - \Rightarrow - :: \{\text{order-topology, ordered-real-vector}\}$ +

assumes *integrable*: $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$
and *supermartingale-property*: $\bigwedge i j. t_0 \leq i \implies i \leq j \implies AE \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X j) \xi$

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

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-const-fun*[*intro*]:

assumes *integrable* $M f f \in \text{borel-measurable } (F t_0)$

shows *supermartingale* $M F t_0 (\lambda-. f)$

proof –

interpret *martingale* $M F t_0 \lambda-. f$ **using** *assms* **by** (*rule martingale-const-fun*)

show *supermartingale* $M F t_0 (\lambda-. f)$ **using** *martingale-property* **by** (*unfold-locales*) (*force simp add: integrable*)+
qed

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-cond-exp*[*intro*]:

assumes *integrable* $M f$

shows *supermartingale* $M F t_0 (\lambda i. \text{cond-exp } M (F i) f)$

proof –

interpret *martingale* $M F t_0 \lambda i. \text{cond-exp } M (F i) f$ **using** *assms* **by** (*rule martingale-cond-exp*)

show *supermartingale* $M F t_0 (\lambda i. \text{cond-exp } M (F i) f)$ **using** *martingale-property* **by** (*unfold-locales*) (*force simp add: integrable*)+

qed

corollary (**in** *finite-filtered-measure*) *supermartingale-const*[*intro*]: *supermartingale* $M F t_0 (\lambda-. c)$ **by** *fastforce*

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* $M F t_0 X \longleftrightarrow$ *submartingale* $M F t_0 X \wedge$ *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* $M F t_0 X \wedge$ *supermartingale* $M F t_0 X$ **using** *submartingale-axioms supermartingale-axioms* **by** *blast*

next

assume *asm*: *submartingale* $M F t_0 X \wedge$ *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

5.4 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 $AE \xi$ in M . *cond-exp* $M (F i) (\lambda \xi. X j \xi - X i \xi) \xi = 0$

using *martingale-property*[*OF assms*] *assms*

sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted, of i*]

sigma-finite-subalgebra.cond-exp-diff[*OF - integrable(1,1), of F i j i*] **by**

fastforce

lemma *set-integral-eq*:

assumes $A \in F i t_0 \leq i \leq j$

shows *set-lebesgue-integral* $M A (X i) =$ *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*]) *fastforce*+

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* $M F t_0 (\lambda i x. c *_R X i x)$
proof –
{
 fix $i j :: 'b$ **assume** *asm*: $t_0 \leq i i \leq j$
 interpret *sigma-finite-subalgebra* $M F i$ **using** *asm* **by** *blast*
 have *AE* x *in* $M. c *_R X i x = \text{cond-exp } M (F i) (\lambda x. c *_R X j x) x$ **using** *asm cond-exp-scaleR-right*[*OF integrable, of j, THEN AE-symmetric*] *martingale-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*: $t_0 \leq i i \leq j$
 hence *AE* ξ *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 $Y: \text{martingale } M F t_0 Y$ **by** (*rule assms*)
{
 fix $i j :: 'b$ **assume** *asm*: $t_0 \leq i i \leq j$
 hence *AE* ξ *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 martingale.integrable)
qed

end

```

Using properties of the conditional expectation, we present the following alternative characterizations of martingales.

```

lemma (in sigma-finite-filtered-measure) martingale-of-cond-exp-diff-eq-zero:
  assumes adapted: adapted-process M F t0 X
    and 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 t0 X
proof
  interpret adapted-process M F t0 X by (rule adapted)
  {
    fix i j :: 'b assume asm: t0 ≤ i i ≤ j
    thus AE ξ in M. X i ξ = cond-exp M (F i) (X j) ξ
    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 (auto intro: integrable adapted[THEN adapted-process.adapted])

```

```

lemma (in sigma-finite-filtered-measure) martingale-of-set-integral-eq:
  assumes adapted: adapted-process M F t0 X
    and 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 t0 X
proof (unfold-locales)
  fix i j :: 'b assume asm: t0 ≤ i i ≤ j
  interpret adapted-process M F t0 X by (rule adapted)
  interpret sigma-finite-subalgebra M F i using asm by blast
  interpret r: sigma-finite-measure restr-to-subalg M (F i) by (simp add: sigma-fin-subalg)
  {
    fix A assume A ∈ restr-to-subalg M (F i)
    hence *: A ∈ F i using sets-restr-to-subalg subalgebras asm by blast
    have set-lebesgue-integral (restr-to-subalg M (F i)) A (X i) = 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 ... = set-lebesgue-integral M A (cond-exp M (F i) (X j)) using * assms(3)[OF asm] cond-exp-set-integral[OF integrable] asm by auto
    finally have set-lebesgue-integral (restr-to-subalg M (F i)) A (X i) = set-lebesgue-integral (restr-to-subalg M (F i)) A (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 $AE \xi$ in *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 $AE \xi$ in M . $X i \xi = \text{cond-exp } M (F i) (X j) \xi$ **using** *AE-restr-to-subalg[OF subalg]* **by** *blast*
qed (*auto intro: integrable adapted[THEN adapted-process.adapted]*)

5.5 Submartingale Lemmas

context *submartingale*
begin

lemma *cond-exp-diff-nonneg*:
assumes $t_0 \leq i \leq j$
shows $AE x$ in M . $\text{cond-exp } M (F i) (\lambda x. X j x - X i x) \geq 0$
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 : *submartingale* $M F t_0 Y$ **by** (*rule assms*)
 {
fix $i j$:: 'b **assume** *asm*: $t_0 \leq i \leq j$
hence $AE \xi$ in M . $X i \xi + Y i \xi \leq \text{cond-exp } M (F i) (\lambda x. X j x + Y j x) \xi$
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-locales*) (*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 . $X i \xi - Y i \xi \leq \text{cond-exp } M (F i) (\lambda x. X j x - Y j x) \xi$
using *sigma-finite-subalgebra.cond-exp-diff[OF - integrable supermartingale.integrable[OF assms], of F i j 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-locales) (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 supermartingale M F t0 ( $\lambda i \xi. c *_{\mathbb{R}} X i \xi$ )

```

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] 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 t0 ( $\lambda i \xi. c *_{\mathbb{R}} X i \xi$ )

```

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] 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  $A \in F i t_0 \leq i i \leq j$ 
  shows set-lebesgue-integral M A (X i) ≤ set-lebesgue-integral M A (X j)
  using submartingale-property[OF assms(2), of j] assms subsetD[OF sets-F-subset]

```

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: set-lebesgue-integral-def)

lemma *max*:

assumes *submartingale M F t₀ Y*
shows *submartingale M F t₀ (λ i ξ. max (X i ξ) (Y i ξ))*
proof (*unfold-locales*)
interpret *Y: submartingale-linorder M F t₀ Y* **by** (*intro submartingale-linorder.intro 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) (λ ξ. 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 $\bigwedge i. t_0 \leq i \implies (\lambda \xi. \max (X i \xi) (Y i \xi)) \in \text{borel-measurable } (F i) \bigwedge i. t_0 \leq i \implies \text{integrable } M (\lambda \xi. \max (X i \xi) (Y i \xi))$ **by** (*force intro: Y.integrable integrable assms*)
qed

lemma *max-0*:

shows *submartingale M F t₀ (λ i ξ. max 0 (X i ξ))*
proof –
interpret *zero: martingale-linorder M F t₀ λ- -. 0* **by** (*force intro: martingale-linorder.intro martingale-order.intro*)
show *?thesis* **by** (*intro zero.max submartingale-linorder.intro submartingale-axioms*)
qed

end

lemma (*in sigma-finite-filtered-measure*) *submartingale-of-cond-exp-diff-nonneg*:

assumes *adapted: adapted-process M F t₀ X*
and *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 t₀ X*
proof (*unfold-locales*)
interpret *adapted-process M F t₀ X* **by** (*rule adapted*)
{
fix *i j :: 'b* **assume** *asm: t₀ ≤ i i ≤ j*
thus *AE ξ in M. X i ξ ≤ cond-exp M (F i) (X j) ξ*
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 (auto intro: integrable adapted[THEN adapted-process.adapted])

lemma (in sigma-finite-filtered-measure) submartingale-of-set-integral-le:
  fixes X :: -  $\Rightarrow$  -  $\Rightarrow$  - :: {linorder-topology}
  assumes adapted: adapted-process M F t0 X
    and 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 t0 X
proof (unfold-locales)
  {
    fix i j :: 'b assume asm: t0  $\leq$  i i  $\leq$  j
    interpret adapted-process M F t0 X by (rule adapted)
    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  $\in$  restr-to-subalg M (F i)
      hence *: A  $\in$  F i using asm sets-restr-to-subalg subalgebras by blast
      have set-lebesgue-integral (restr-to-subalg M (F i)) A (X i) = 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 ...  $\leq$  set-lebesgue-integral M A (cond-exp M (F i) (X j)) using * assms(3)[OF asm] asm sigma-finite-subalgebra.cond-exp-set-integral[OF - integrable] by fastforce
      also have ... = set-lebesgue-integral (restr-to-subalg M (F i)) A (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  $\leq$  set-lebesgue-integral (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 AE  $\xi$  in restr-to-subalg M (F i). 0  $\leq$  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 AE  $\xi$  in M. X i  $\xi \leq$  cond-exp M (F i) (X j)  $\xi$  using AE-restr-to-subalg[OF subalgebras] asm by simp
  }
qed (auto intro: integrable adapted[THEN adapted-process.adapted])

```

5.6 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 *AE* x *in* M . *cond-exp* M $(F\ i)$ $(\lambda \xi. X\ i\ \xi - X\ j\ \xi)\ x \geq 0$

using *assms* *supermartingale-property*[*OF assms*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable(1,1), of F i i j*]

sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted, of i*] **by**
fastforce

lemma *add[intro]*:

assumes *supermartingale* $M\ F\ t_0\ Y$

shows *supermartingale* $M\ F\ t_0\ (\lambda i\ \xi. X\ i\ \xi + Y\ i\ \xi)$

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* ξ *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 *sigma-finite-subalgebra.cond-exp-add*[*OF - integrable supermartingale.integrable*[*OF assms*], *of F i j j*]

supermartingale-property[*OF asm*] *supermartingale.supermartingale-property*[*OF assms asm*] *add-mono*[*of - X i - - Y i -*] **by** *force*

}

thus *?thesis using assms by (unfold-locales) (auto simp add: borel-measurable-add random-variable adapted integrable Y.random-variable Y.adapted supermartingale.integrable)*

qed

lemma *diff[intro]*:

assumes *submartingale* $M\ F\ t_0\ Y$

shows *supermartingale* $M\ F\ t_0\ (\lambda i\ \xi. X\ i\ \xi - Y\ i\ \xi)$

proof –

interpret Y : *submartingale* $M\ F\ t_0\ Y$ **by** (*rule assms*)

{

fix $i\ j$:: 'b **assume** *asm*: $t_0 \leq i \leq j$

hence *AE* ξ *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 *sigma-finite-subalgebra.cond-exp-diff*[*OF - 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-locales) (auto simp add: borel-measurable-diff random-variable adapted integrable Y.random-variable Y.adapted submartingale.integrable)*

qed

lemma *scaleR-nonneg*:

assumes $c \geq 0$

shows *supermartingale* $M\ F\ t_0\ (\lambda i\ \xi. c *_{\mathbb{R}} X\ i\ \xi)$

```

proof
  {
    fix  $i j :: 'b$  assume  $asm: t_0 \leq i \ i \leq j$ 
    thus  $AE \ \xi$  in  $M$ .  $c *_R X \ i \ \xi \geq cond\text{-}exp \ M \ (F \ i) \ (\lambda \xi. \ c *_R X \ j \ \xi) \ \xi$ 
    using  $\sigma\text{-finite}\text{-subalgebra.cond}\text{-}exp\text{-}scaleR\text{-}right[OF \ \text{integrable}, \ of \ F \ i \ j \ c]$   $supermartingale\text{-}property[OF \ asm]$  by ( $fastforce \ intro!:$   $scaleR\text{-}left\text{-}mono[OF \ \text{assms}]$ )
  }
qed ( $auto \ simp \ add:$   $borel\text{-}measurable\text{-}integrable \ borel\text{-}measurable\text{-}scaleR \ integrable \ random\text{-}variable \ adapted \ borel\text{-}measurable\text{-}const\text{-}scaleR$ )

```

```

lemma  $scaleR\text{-}le\text{-}zero:$ 
  assumes  $c \leq 0$ 
  shows  $submartingale \ M \ F \ t_0 \ (\lambda i \ \xi. \ c *_R X \ i \ \xi)$ 

```

```

proof
  {
    fix  $i j :: 'b$  assume  $asm: t_0 \leq i \ i \leq j$ 
    thus  $AE \ \xi$  in  $M$ .  $c *_R X \ i \ \xi \leq cond\text{-}exp \ M \ (F \ i) \ (\lambda \xi. \ c *_R X \ j \ \xi) \ \xi$ 
    using  $\sigma\text{-finite}\text{-subalgebra.cond}\text{-}exp\text{-}scaleR\text{-}right[OF \ \text{integrable}, \ of \ F \ i \ j \ c]$   $supermartingale\text{-}property[OF \ asm]$  by ( $fastforce \ intro!:$   $scaleR\text{-}left\text{-}mono\text{-}neg[OF \ \text{assms}]$ )
  }
qed ( $auto \ simp \ add:$   $borel\text{-}measurable\text{-}integrable \ borel\text{-}measurable\text{-}scaleR \ integrable \ random\text{-}variable \ adapted \ borel\text{-}measurable\text{-}const\text{-}scaleR$ )

```

```

lemma  $uminus[intro]:$ 
  shows  $submartingale \ M \ F \ t_0 \ (- \ X)$ 
  unfolding  $fun\text{-}Compl\text{-}def$  using  $scaleR\text{-}le\text{-}zero[of \ -1]$  by  $simp$ 

```

end

```

context  $supermartingale\text{-}linorder$ 
begin

```

```

lemma  $set\text{-}integral\text{-}ge:$ 
  assumes  $A \in F \ i \ t_0 \leq i \ i \leq j$ 
  shows  $set\text{-}lebesgue\text{-}integral \ M \ A \ (X \ i) \geq set\text{-}lebesgue\text{-}integral \ M \ A \ (X \ j)$ 
  using  $supermartingale\text{-}property[OF \ assms(2), \ of \ j]$   $assms \ subsetD[OF \ sets\text{-}F\text{-}subset]$ 
  by ( $subst \ \sigma\text{-finite}\text{-subalgebra.cond}\text{-}exp\text{-}set\text{-}integral[OF \ \text{integrable} \ assms(1), \ of \ j]$ )
  ( $auto \ intro!:$   $scaleR\text{-}left\text{-}mono \ integral\text{-}mono\text{-}AE\text{-}banach \ integrable\text{-}mult\text{-}indicator \ integrable \ simp \ add:$   $set\text{-}lebesgue\text{-}integral\text{-}def$ )

```

```

lemma  $min:$ 
  assumes  $supermartingale \ M \ F \ t_0 \ Y$ 
  shows  $supermartingale \ M \ F \ t_0 \ (\lambda i \ \xi. \ min \ (X \ i \ \xi) \ (Y \ i \ \xi))$ 
proof ( $unfold\text{-}locales$ )
  interpret  $Y:$   $supermartingale\text{-}linorder \ M \ F \ t_0 \ Y$  by ( $intro \ supermartingale\text{-}linorder.intro \ assms$ )

```

```

{
  fix i j :: 'b assume asm: t0 ≤ 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) (λξ. min (X j ξ) (Y
j ξ)) ξ using sigma-finite-subalgebra.cond-exp-min[OF - integrable Y.integrable, of
F i j] asm by (fast intro: order.trans)
}
show ∧i. t0 ≤ i ⇒ (λξ. min (X i ξ) (Y i ξ)) ∈ borel-measurable (F i) ∧i.
t0 ≤ i ⇒ integrable M (λξ. min (X i ξ) (Y i ξ)) by (force intro: Y.integrable
integrable assms)+
qed

```

lemma min-0:

```

shows supermartingale M F t0 (λi ξ. min 0 (X i ξ))
proof -
  interpret zero: martingale-linorder M F t0 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-filtered-measure) supermartingale-of-cond-exp-diff-le-zero:

```

assumes adapted: adapted-process M F t0 X
  and integrable: ∧i. t0 ≤ i ⇒ integrable M (X i)
  and diff-le-zero: ∧i j. t0 ≤ i ⇒ i ≤ j ⇒ AE x in M. cond-exp M (F i)
(λξ. X j ξ - X i ξ) x ≤ 0
shows supermartingale M F t0 X
proof
  interpret adapted-process M F t0 X by (rule adapted)
  {
    fix i j :: 'b assume asm: t0 ≤ 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 (auto intro: integrable adapted[THEN adapted-process.adapted])

```

lemma (in sigma-finite-filtered-measure) supermartingale-of-set-integral-ge:

```

fixes X :: - ⇒ - ⇒ - :: {linorder-topology}
assumes adapted: adapted-process M F t0 X
  and integrable: ∧i. t0 ≤ i ⇒ integrable M (X i)
  and ∧A i j. t0 ≤ i ⇒ i ≤ j ⇒ A ∈ F i ⇒ set-lebesgue-integral M A (X
j) ≤ set-lebesgue-integral M A (X i)

```

```

    shows supermartingale M F t0 X
  proof -
    interpret adapted-process M F t0 X by (rule adapted)
    note * = set-integral-uminus[unfolded set-integrable-def, OF integrable-mult-indicator[OF
- integrable]]
    have supermartingale M F t0 (-( - X))
    using ord-eq-le-trans[OF * ord-le-eq-trans[OF le-imp-neg-le[OF assms(β)] *[symmetric]]]
sets-F-subset[THEN subsetD]
    by (intro submartingale.uminus submartingale-of-set-integral-le[OF uminus-adapted])

    (clarsimp simp add: fun-Compl-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 $i \leq j$.

5.7 Discrete Time Martingales

```

context nat-sigma-finite-filtered-measure
begin

```

A predictable martingale is necessarily constant.

```

lemma predictable-const:
  assumes martingale M F 0 X
    and predictable-process M F 0 X
  shows AE ξ in M. X i ξ = X j ξ
proof -
  interpret martingale M F 0 X by (rule assms)
  have *: AE ξ in M. X i ξ = X 0 ξ for i
  proof (induction i)
    case 0
    then show ?case by (simp add: bot-nat-def)
  next
    case (Suc i)
    interpret S: adapted-process M F 0 λi. X (Suc i) by (intro predictable-implies-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 martingale-of-set-integral-eq-Suc:
  assumes adapted: adapted-process M F 0 X
    and integrable:  $\bigwedge i. \text{integrable } M (X i)$ 

```

and $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) = \text{set-lebesgue-integral } M A (X (\text{Suc } i))$
shows *martingale* $M F 0 X$
proof (*intro martingale-of-set-integral-eq adapted integrable*)
fix $i j A$ **assume** $\text{asm}: i \leq j A \in \text{sets } (F i)$
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 ($\text{Suc } n$)
hence $*$: $n = j - \text{Suc } i$ **by** *linarith*
have $\text{Suc } i \leq j$ **using** $\text{Suc}(2,3)$ **by** *linarith*
thus *?case using sets-F-mono[OF - le-SucI] Suc(4) Suc(1)[OF *]* **by** (*auto intro: assms(3)[THEN trans]*)
qed
qed

lemma *martingale-nat*:

assumes *adapted: adapted-process M F 0 X*
and *integrable: $\bigwedge i. \text{integrable } M (X i)$*
and $\bigwedge i. AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X (\text{Suc } i)) \xi$
shows *martingale M F 0 X*
proof (*unfold-locales*)
interpret *adapted-process M F 0 X by (rule adapted)*
fix $i j :: \text{nat}$ **assume** $\text{asm}: i \leq j$
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 ($\text{Suc } n$)
have $j: j = \text{Suc } (n + i)$ **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** $\text{assms}(3)[\text{THEN } AE\text{-symmetric}]$ **by** *blast*
have $AE \xi \text{ in } M. \text{cond-exp } M (F i) (X j) \xi = \text{cond-exp } M (F i) (\text{cond-exp } M (F (n + i)) (X j)) \xi$ **by** (*intro cond-exp-nested-subalg integrable subalg, simp add: subalgebra-def sets-F-mono*)
hence $AE \xi \text{ in } M. \text{cond-exp } M (F i) (X j) \xi = \text{cond-exp } M (F i) (X (n + i)) \xi$ **using** $\text{cond-exp-cong-AE}[OF \text{integrable-cond-exp integrable } *]$ **by** *force*
thus *?case using Suc(1)[OF n]* **by** *fastforce*
qed
qed (*auto simp add: integrable adapted[THEN adapted-process.adapted]*)

lemma *martingale-of-cond-exp-diff-Suc-eq-zero*:

assumes *adapted*: *adapted-process* $M F 0 X$
and *integrable*: $\bigwedge i. \text{integrable } M (X i)$
and $\bigwedge i. AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X (Suc i) \xi - X i \xi) \xi = 0$
shows *martingale* $M F 0 X$
proof (*intro martingale-nat integrable adapted*)
interpret *adapted-process* $M F 0 X$ **by** (*rule adapted*)
fix i
show $AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X (Suc i)) \xi$ **using** *cond-exp-diff*[*OF integrable(1,1), of i Suc i i*] *cond-exp-F-meas*[*OF integrable adapted, of i*] *assms(3)*[*of i*] **by** *fastforce*
qed
end

5.8 Discrete Time Submartingales

context *nat-sigma-finite-filtered-measure*
begin

lemma *predictable-mono*:

assumes *submartingale* $M F 0 X$
and *predictable-process* $M F 0 X i \leq j$
shows $AE \xi \text{ in } M. X i \xi \leq X j \xi$
using *assms(3)*
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 *submartingale* $M F 0 X$ **by** (*rule assms*)
interpret S : *adapted-process* $M F 0 \lambda i. X (Suc i)$ **by** (*intro predictable-implies-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 *submartingale-of-set-integral-le-Suc*:

fixes $X :: - \Rightarrow - \Rightarrow - :: \{\text{linorder-topology}\}$
assumes *adapted*: *adapted-process* $M F 0 X$
and *integrable*: $\bigwedge i. \text{integrable } M (X i)$
and $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X (Suc i))$
shows *submartingale* $M F 0 X$
proof (*intro submartingale-of-set-integral-le adapted integrable*)
fix $i j A$ **assume** *asm*: $i \leq j \wedge A \in \text{sets } (F i)$
show $\text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X j)$ **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 ≤ 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(3)[THEN order-trans])
qed
qed

```

```

lemma submartingale-nat:
  fixes X :: - ⇒ - ⇒ - :: {linorder-topology}
  assumes adapted: adapted-process M F 0 X
  and integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
  and  $\bigwedge i. AE \xi \text{ in } M. X i \xi \leq \text{cond-exp } M (F i) (X (Suc i)) \xi$ 
  shows submartingale M F 0 X
proof -
  show ?thesis using subalg assms(3) integrable
  by (intro submartingale-of-set-integral-le-Suc adapted integrable ord-le-eq-trans[OF
set-integral-mono-AE-banach cond-exp-set-integral[symmetric]])
  (meson in-mono integrable-mult-indicator set-integrable-def subalgebra-def,
meson integrable-cond-exp in-mono integrable-mult-indicator set-integrable-def sub-
algebra-def, fast+)
qed

```

```

lemma submartingale-of-cond-exp-diff-Suc-nonneg:
  fixes X :: - ⇒ - ⇒ - :: {linorder-topology}
  assumes adapted: adapted-process M F 0 X
  and integrable:  $\bigwedge i. \text{integrable } M (X i)$ 
  and  $\bigwedge i. AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X (Suc i) \xi - X i \xi) \xi \geq 0$ 
  shows submartingale M F 0 X
proof (intro submartingale-nat integrable adapted)
  interpret adapted-process M F 0 X by (rule assms)
  fix i
  show AE  $\xi \text{ in } M. X i \xi \leq \text{cond-exp } M (F i) (X (Suc i)) \xi$  using cond-exp-diff[OF
integrable(1,1), of i Suc i i] cond-exp-F-meas[OF integrable adapted, of i] assms(3)[of
i] by fastforce
qed

```

```

lemma submartingale-partial-sum-scaleR:
  assumes submartingale-linorder M F 0 X
  and adapted-process M F 0 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 submartingale M F 0  $(\lambda n \xi. \sum_{i < n}. C i \xi *_R (X (Suc i) \xi - X i \xi))$ 
proof -
  interpret submartingale-linorder M F 0 X by (rule assms)

```

interpret C : *adapted-process* $M F 0 C$ **by** (*rule* *assms*)
interpret C' : *adapted-process* $M F 0 \lambda i \xi. C (i - 1) \xi *R (X i \xi - X (i - 1) \xi)$ **by** (*intro* *adapted-process.scaleR-right-adapted* *adapted-process.diff-adapted*, *unfold-locales*) (*auto* *intro: adaptedD C.adaptedD*)
interpret S : *adapted-process* $M F 0 \lambda n \xi. \sum i < n. C i \xi *R (X (Suc i) \xi - X i \xi)$ **using** $C'.adapted-process-axioms[THEN partial-sum-Suc-adapted]$ *diff-Suc-1* **by** *simp*
have *integrable* $M (\lambda x. C i x *R (X (Suc i) x - X i x))$ **for** i **using** *assms(3,4)[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** $AE \xi$ *in* $M. 0 \leq cond-exp M (F i) (\lambda \xi. (\sum i < Suc i. C i \xi *R (X (Suc i) \xi - X i \xi)) - (\sum i < i. C i \xi *R (X (Suc i) \xi - X i \xi))) \xi$ **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(3,4)[of i]*
by (*fastforce* *simp add: scaleR-nonneg-nonneg integrable*)
ultimately **show** *?thesis* **by** (*intro* *submartingale-of-cond-exp-diff-Suc-nonneg S.adapted-process-axioms Bochner-Integration.integrable-sum, blast+*)
qed

lemma *submartingale-partial-sum-scaleR'*:

assumes *submartingale-linorder* $M F 0 X$
and *predictable-process* $M F 0 C \wedge i. AE \xi$ *in* $M. 0 \leq C i \xi \wedge i. AE \xi$ *in* $M. C i \xi \leq R$

shows *submartingale* $M F 0 (\lambda n \xi. \sum i < n. C (Suc i) \xi *R (X (Suc i) \xi - X i \xi))$

proof –

interpret $Suc-C$: *adapted-process* $M F 0 \lambda i. C (Suc i)$ **using** *predictable-implies-adapted-Suc* *assms* **by** *blast*

show *?thesis* **by** (*intro* *submartingale-partial-sum-scaleR[OF assms(1), of - R]* *assms*) (*intro-locales*)

qed

end

5.9 Discrete Time Supermartingales

context *nat-sigma-finite-filtered-measure*

begin

lemma *predictable-mono'*:

assumes *supermartingale* $M F 0 X$

and *predictable-process* $M F 0 X i \leq j$

shows $AE \xi$ *in* $M. X i \xi \geq X j \xi$

using *assms(3)*

proof (*induction* $j - i$ *arbitrary: i j*)

case 0

then **show** *?case* **by** *simp*

next
case (*Suc n*)
hence *: $n = j - \text{Suc } i$ **by** *linarith*
interpret *supermartingale M F 0 X* **by** (*rule assms*)
interpret *S: adapted-process M F 0 λi. X (Suc i)* **by** (*intro predictable-implies-adapted-Suc assms*)
have $\text{Suc } i \leq j$ **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 *supermartingale-of-set-integral-ge-Suc*:
fixes $X :: - \Rightarrow - \Rightarrow - :: \{\text{linorder-topology}\}$
assumes *adapted: adapted-process M F 0 X*
and *integrable: $\bigwedge i. \text{integrable } M (X i)$*
and $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) \geq \text{set-lebesgue-integral } M A (X (\text{Suc } i))$
shows *supermartingale M F 0 X*
proof –
interpret *adapted-process M F 0 X* **by** (*rule assms*)
interpret *uminus-X: adapted-process M F 0 -X* **by** (*rule uminus-adapted*)
note * = *set-integral-uminus[unfolded set-integrable-def, OF integrable-mult-indicator[OF -integrable]]*
have *supermartingale M F 0 (-(- X))*
using *ord-eq-le-trans[OF * ord-le-eq-trans[OF le-imp-neg-le[OF assms(3)] *[symmetric]]]* *sets-F-subset[THEN subsetD]*
by (*intro submartingale.uminus submartingale-of-set-integral-le-Suc[OF uminus-adapted]*)
(clarsimp simp add: fun-Compl-def integrable | fastforce)+
thus ?*thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

lemma *supermartingale-nat*:
fixes $X :: - \Rightarrow - \Rightarrow - :: \{\text{linorder-topology}\}$
assumes *adapted: adapted-process M F 0 X*
and *integrable: $\bigwedge i. \text{integrable } M (X i)$*
and $\bigwedge i. AE \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X (\text{Suc } i)) \xi$
shows *supermartingale M F 0 X*
proof –
interpret *adapted-process M F 0 X* **by** (*rule assms*)
have $AE \xi \text{ in } M. - X i \xi \leq \text{cond-exp } M (F i) (\lambda x. - X (\text{Suc } i) x) \xi$ **for** i **using** *assms(3) cond-exp-uminus[OF integrable, of i Suc i]* **by** *force*
hence *supermartingale M F 0 (-(- X))* **by** (*intro submartingale.uminus submartingale-nat[OF uminus-adapted]*) *(auto simp add: fun-Compl-def integrable)*
thus ?*thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

lemma *supermartingale-of-cond-exp-diff-Suc-le-zero*:

```

fixes  $X :: - \Rightarrow - \Rightarrow - :: \{linorder-topology\}$ 
assumes adapted: adapted-process M F 0 X
  and integrable:  $\bigwedge i. integrable\ M\ (X\ i)$ 
  and  $\bigwedge i. AE\ \xi\ in\ M. cond-exp\ M\ (F\ i)\ (\lambda\xi. X\ (Suc\ i)\ \xi - X\ i\ \xi)\ \xi \leq 0$ 
shows supermartingale M F 0 X
proof (intro supermartingale-nat integrable adapted)
  interpret adapted-process M F 0 X by (rule assms)
  fix  $i$ 
  show  $AE\ \xi\ in\ M. X\ i\ \xi \geq cond-exp\ M\ (F\ i)\ (X\ (Suc\ i))\ \xi$  using  $cond-exp-diff[OF\ integrable(1,1),\ of\ i\ Suc\ i\ i]\ cond-exp-F-meas[OF\ integrable\ adapted,\ of\ i]\ assms(3)[of\ i]$  by fastforce
qed

end

end

```

```

theory Example-Coin-Toss
imports Martingale HOL-Probability.Stream-Space HOL-Probability.Probability-Mass-Function
begin

```

6 Example: Coin Toss

Consider a coin-tossing game, where the coin lands on heads with probability $p \in [0, 1]$. Assume that the gambler wins a fixed amount $c > 0$ on a heads outcome and loses the same amount c on a tails outcome. Let $(X_n)_{n \in \mathbb{N}}$ be a stochastic process, where X_n denotes the gamblers fortune after the n -th coin toss. Then, we have the following three cases.

1. If $p = 1/2$, it means the coin is fair and has an equal chance of landing heads or tails. In this case, the gambler, on average, neither wins nor loses money over time. The expected value of the gamblers fortune stays the same over time. Therefore, $(X_n)_{n \in \mathbb{N}}$ is a martingale.
2. If $p \geq 1/2$, it means the coin is biased in favor of heads. In this case, the gambler is more likely to win money on each bet. Over time, the gamblers fortune tends to increase on average. Therefore, $(X_n)_{n \in \mathbb{N}}$ is a submartingale.
3. If $p \leq 1/2$, it means the coin is biased in favor of tails. In this scenario, the gambler is more likely to lose money on each bet. Over time, the gamblers fortune decreases on average. Therefore, $(X_n)_{n \in \mathbb{N}}$ is a supermartingale.

To formalize this example, we first consider a probability space consisting of infinite sequences of coin tosses.

definition *bernoulli-stream :: real \Rightarrow (bool stream) measure where*

$\text{bernoulli-stream } p = \text{stream-space } (\text{measure-pmf } (\text{bernoulli-pmf } p))$

lemma $\text{space-bernoulli-stream}[simp]$: $\text{space } (\text{bernoulli-stream } p) = \text{UNIV}$ **by** $(\text{simp add: bernoulli-stream-def space-stream-space})$

We define the fortune of the player at time n to be the number of heads minus number of tails.

definition $\text{fortune} :: \text{nat} \Rightarrow \text{bool stream} \Rightarrow \text{real}$ **where**
 $\text{fortune } n = (\lambda s. \sum b \leftarrow \text{stake } (\text{Suc } n) s. \text{if } b \text{ then } 1 \text{ else } -1)$

definition $\text{toss} :: \text{nat} \Rightarrow \text{bool stream} \Rightarrow \text{real}$ **where**
 $\text{toss } n = (\lambda s. \text{if } \text{snth } s \ n \text{ then } 1 \text{ else } -1)$

lemma $\text{toss-indicator-def}$: $\text{toss } n = \text{indicator } \{s. s !! n\} - \text{indicator } \{s. \neg s !! n\}$
unfolding $\text{toss-def indicator-def}$ **by** force

lemma range-toss : $\text{range } (\text{toss } n) = \{-1, 1\}$

proof –

have $\text{sconst } \text{True} !! n$ **by** simp
moreover have $\neg \text{sconst } \text{False} !! n$ **by** simp
ultimately have $\exists x. x !! n \ \exists x. \neg x !! n$ **by** blast+
thus $?thesis$ **unfolding** $\text{toss-def image-def}$ **by** auto

qed

lemma vimage-toss : $\text{toss } n - 'A = (\text{if } 1 \in A \text{ then } \{s. s !! n\} \text{ else } \{\}) \cup (\text{if } -1 \in A \text{ then } \{s. \neg s !! n\} \text{ else } \{\})$
unfolding $\text{vimage-def toss-def}$ **by** auto

lemma fortune-Suc : $\text{fortune } (\text{Suc } n) s = \text{fortune } n s + \text{toss } (\text{Suc } n) s$
by $(\text{induction } n \text{ arbitrary: } s) (\text{simp add: fortune-def toss-def})+$

lemma fortune-toss-sum : $\text{fortune } n s = (\sum i \in \{..n\}. \text{toss } i s)$
by $(\text{induction } n \text{ arbitrary: } s) (\text{simp add: fortune-def toss-def, simp add: fortune-Suc})$

lemma fortune-bound : $\text{norm } (\text{fortune } n s) \leq \text{Suc } n$ **by** $(\text{induction } n) (\text{force simp add: fortune-toss-sum toss-def})+$

Our definition of bernoulli-stream constitutes a probability space.

interpretation $\text{prob-space bernoulli-stream } p$ **unfolding** $\text{bernoulli-stream-def}$ **by**
 $(\text{simp add: measure-pmf.prob-space-axioms prob-space.prob-space-stream-space})$

abbreviation $\text{toss-filtration } p \equiv \text{nat-natural-filtration } (\text{bernoulli-stream } p) \ \text{toss}$

The stochastic process toss is adapted to the filtration it generates.

interpretation toss : $\text{adapted-process bernoulli-stream } p \ \text{toss-filtration } p \ 0 \ \text{toss}$
by $(\text{intro adapted-process.intro stochastic-process.adapted-process-natural-filtration})$
 $(\text{unfold-locale, auto simp add: toss-def bernoulli-stream-def})$

interpretation *bernoulli-stream-natural-filtration*: *nat-finite-filtered-measure bernoulli-stream p toss-filtration p*

by (*simp add: nat-finite-filtered-measure-def toss.finite-filtered-measure-natural-filtration*)

Similarly, the stochastic process *fortune* is adapted to the filtration generated by the tosses.

interpretation *fortune*: *adapted-process bernoulli-stream p toss-filtration p 0 fortune*

proof –

show *adapted-process (bernoulli-stream p) (toss-filtration p) 0 fortune*

unfolding *fortune-toss-sum*

by (*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-random-variable*

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 $\text{prob } p \{w \in \text{space } (\text{bernoulli-stream } p). \forall j \in J. P j = w !! j\} = p^{\wedge \text{card } (J \cap \text{Collect } P)} * (1 - p)^{\wedge (\text{card } (J - \text{Collect } P))}$

proof –

let $?PiE = (\Pi_E i \in J. \text{if } P i \text{ then } \{\text{True}\} \text{ else } \{\text{False}\})$

have *product-prob-space* $(\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p))$ **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 $\dots = \text{prod-emb UNIV } (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p)) J ?PiE$

proof –

{

fix s **assume** $(\forall i \in J. P i = s i)$

hence $(\forall i \in J. P i = s i) = (s \in \text{prod-emb UNIV } (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p)) J ?PiE)$

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 $(\forall i \in J. P i = s i) = (s \in \text{prod-emb UNIV } (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p))) J ?PiE$
by $(\text{simp add: restrict-def prod-emb-iff}[of s]) (\text{smt } (\text{verit, ccfv-SIG}) PiE\text{-mem } \text{assms}(3) \text{id-def insert-iff singleton-iff})$
}
ultimately show *?thesis* **by** *auto*
qed
finally have $\text{inteq: } (\text{to-stream } -' \{s. \forall i \in J. P i = s !! i\}) = \text{prod-emb UNIV } (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p)) J ?PiE .$
let $?M = (Pi_M UNIV (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p)))$
have $\text{emeasure } (\text{bernoulli-stream } p) \{s \in \text{space } (\text{bernoulli-stream } p). \forall i \in J. P i = s !! i\} = \text{emeasure } ?M (\text{to-stream } -' \{s. \forall i \in J. P i = s !! i\})$
using $\text{assms } \text{emeasure-distr}[of \text{to-stream } ?M (\text{vimage-algebra } (\text{streams } (\text{space } (\text{measure-pmf } (\text{bernoulli-pmf } p)))) (!) ?M) \{s. \forall i \in J. P i = s !! i\}, \text{symmetric}] \text{measurable-to-stream}[of (\text{measure-pmf } (\text{bernoulli-pmf } p))]$
by $(\text{simp only: bernoulli-stream-def stream-space-def } *, \text{simp add: space-PiM}) (\text{smt } (\text{verit, best}) \text{emeasure-notin-sets in-vimage-algebra inf-top.right-neutral sets-distr vimage-Collect})$
also have $\dots = \text{emeasure } ?M (\text{prod-emb UNIV } (\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p))) J ?PiE$ **using** inteq **by** $(\text{simp add: space-PiM})$
also have $\dots = (\prod_{i \in J. \text{emeasure } (\text{measure-pmf } (\text{bernoulli-pmf } p))} (\text{if } P i \text{ then } \{True\} \text{ else } \{False\}))$
by $(\text{subst } \text{emeasure-PiM-emb}) (\text{auto simp add: prob-space-measure-pmf } \text{assms}(3))$
also have $\dots = (\prod_{i \in J \cap \text{Collect } P. \text{ennreal } p}) * (\prod_{i \in J - \text{Collect } P. \text{ennreal } (1 - p)})$
unfolding $\text{emeasure-pmf-single}[of \text{bernoulli-pmf } p \text{ True, unfolded pmf-bernoulli-True}[OF \text{assms}(1,2)], \text{symmetric}]$
 $\text{emeasure-pmf-single}[of \text{bernoulli-pmf } p \text{ False, unfolded pmf-bernoulli-False}[OF \text{assms}(1,2)], \text{symmetric}]$
by $(\text{simp add: prod.Int-Diff}[OF \text{assms}(3), of - \text{Collect } P])$
also have $\dots = p \wedge \text{card } (J \cap \text{Collect } P) * (1 - p) \wedge \text{card } (J - \text{Collect } P)$ **using** assms **by** $(\text{simp add: prod-ennreal ennreal-mult' ennreal-power})$
finally show *?thesis* **using** assms **by** $(\text{intro } \text{measure-eq-emeasure-eq-ennreal})$
auto
qed

lemma

assumes $0 \leq p$ **and** $p \leq 1$
shows $\text{measure-bernoulli-stream-snth: } \text{prob } p \{w \in \text{space } (\text{bernoulli-stream } p). w !! i\} = p$
and $\text{measure-bernoulli-stream-neg-snth: } \text{prob } p \{w \in \text{space } (\text{bernoulli-stream } p). \neg w !! i\} = 1 - p$
using $\text{measure-bernoulli-stream-snth-pred}[OF \text{assms}, of \{i\} \lambda x. True]$
 $\text{measure-bernoulli-stream-snth-pred}[OF \text{assms}, of \{i\} \lambda x. False]$ **by** *auto*

Now we can express the expected value of a single coin toss.

lemma *integral-toss:*

assumes $0 \leq p$ **and** $p \leq 1$
shows $\text{expectation } p (\text{toss } n) = 2 * p - 1$

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 $\dots = p - \text{prob } p \{s. \neg s !! n\}$ **unfolding** *simple-bochner-integral-def*
using *measure-bernoulli-stream-snth* [*OF assms*]
by (*simp add: range-toss, simp add: toss-def*)
also have $\dots = p - (1 - \text{prob } p \{s. s !! n\})$ **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 ($\lambda \cdot$. *borel*) *toss* $\{0..n\}$
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-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 $\text{prob } p (\bigcap (A - ' J)) = (\prod_{j \in J}. \text{prob } p (A j))$
proof *cases*

We consider the case where there is a zero probability event.

assume $\exists j \in J. 1 \notin B j \wedge -1 \notin B j$
then obtain j **where** *j-is*: $j \in J \ 1 \notin B j \ -1 \notin B j$ **by** *blast*
hence *A-j-empty*: $A j = \{\}$ **using** *B-is* **by** (*force simp add: toss-def vimage-def*)
hence $\bigcap (A - ' J) = \{\}$ **using** *j-is* **by** *blast*
moreover have $\text{prob } p (A j) = 0$ **using** *A-j-empty* **by** *simp*
ultimately show *?thesis* **using** *j-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 $\text{toss } j \ w \in B \ j \longleftrightarrow (1 \in B \ j) = w \ !! \ j \ \text{if } j \in J' \ \text{for } w \ j \ \text{using that}$
unfolding toss-def **by** simp
hence $(\bigcap (A \ ' \ J')) = \{w \in \text{space } (\text{bernoulli-stream } p). \forall j \in J'. (1 \in B \ j) = w \ !! \ j\}$ **using** $B\text{-is}$ **by** force
hence $\text{prob-J': } \text{prob } p \ (\bigcap (A \ ' \ J')) = p \wedge \text{card } (J' \cap \{j. 1 \in B \ j\}) * (1 - p) \wedge \text{card } (J' - \{j. 1 \in B \ j\})$
using $\text{measure-bernoulli-stream-snth-pred[OF assms finite-subset[OF -asm(2)]]}$, of $J' \ \lambda j. 1 \in B \ j$ **by** auto

The index set J' consists of the indices of all non-trivial events.

have $A\text{-j-True: } A \ j = \{w \in \text{space } (\text{bernoulli-stream } p). w \ !! \ j\}$ **if** $j \in J' \cap \{j. 1 \in B \ j\}$ **for** j
using that **by** $(\text{auto simp add: toss-def } B\text{-is}(1) \ \text{split: if-splits})$

have $A\text{-j-False: } A \ j = \{w \in \text{space } (\text{bernoulli-stream } p). \neg w \ !! \ j\}$ **if** $j \in J' - \{j. 1 \in B \ j\}$ **for** j
using $\text{that } B\text{-is}$ **by** $(\text{auto simp add: toss-def})$

have $A\text{-j-top: } A \ j = \text{space } (\text{bernoulli-stream } p)$ **if** $j \in J - J'$ **for** j **using** that
***** **by** $(\text{auto simp add: } B\text{-is toss-def})$

hence $\bigcap (A \ ' \ J) = \bigcap (A \ ' \ J')$ **by** auto

hence $\text{prob } p \ (\bigcap (A \ ' \ J)) = \text{prob } p \ (\bigcap (A \ ' \ J'))$ **by** presburger

also have $\dots = (\prod_{j \in J' \cap \{j. 1 \in B \ j\}} \text{prob } p \ (A \ j)) * (\prod_{j \in J' - \{j. 1 \in B \ j\}} \text{prob } p \ (A \ j))$

by $(\text{simp only: prob-J' } A\text{-j-True } A\text{-j-False measure-bernoulli-stream-snth[OF assms] measure-bernoulli-stream-neg-snth[OF assms] cong: prod.cong})$ simp

also have $\dots = (\prod_{j \in J'} \text{prob } p \ (A \ j))$ **using** $\text{asm}(2)$ **by** $(\text{intro prod.Int-Diff[symmetric]})$
 auto

also have $\dots = (\prod_{j \in J'} \text{prob } p \ (A \ j)) * (\prod_{j \in J - J'} \text{prob } p \ (A \ j))$ **using** $A\text{-j-top prob-space}$ **by** simp

also have $\dots = (\prod_{j \in J} \text{prob } p \ (A \ j))$ **using** $\text{asm}(2)$ **by** $(\text{metis (no-types, lifting) } J'\text{-def mem-Collect-eq mult.commute prod.subset-diff subsetI})$

finally show $?thesis$.

qed

}

thus $\text{indep-sets } p \ (\lambda i. \{\text{toss } i - ' A \cap \text{space } (\text{bernoulli-stream } p) \mid A. A \in \text{sets borel}\}) \ \{0..\}$ **using** $\text{measurable-sets[OF toss.random-variable]}$

by $(\text{intro indep-setsI subsetI})$ fastforce

qed $(\text{simp, intro Int-stableI, simp, metis sets.Int vimage-Int})$

The fortune of a player is a martingale (resp. sub- or supermartingale) with respect to the filtration generated by the coin tosses.

theorem $\text{fortune-martingale}$:

assumes $p = 1/2$

shows $\text{martingale } (\text{bernoulli-stream } p) \ (\text{toss-filtration } p) \ 0 \ \text{fortune}$

using $\text{cond-exp-indep[OF bernoulli-stream-natural-filtration.subalg indep-set-natural-filtration integrable-toss, OF zero-order(1) lessI indep-vars-toss, of p]}$

$\text{integral-toss assms}$

by (intro bernoulli-stream-natural-filtration.martingale-of-cond-exp-diff-Suc-eq-zero
integrable-fortune fortune.adapted-process-axioms) (force simp add: fortune-toss-sum)

theorem fortune-submartingale:

assumes $1/2 \leq p \leq 1$

shows submartingale (bernoulli-stream p) (toss-filtration p) 0 fortune

proof (intro bernoulli-stream-natural-filtration.submartingale-of-cond-exp-diff-Suc-nonneg
integrable-fortune fortune.adapted-process-axioms)

fix n

show AE ξ in bernoulli-stream p. $0 \leq \text{cond-exp (bernoulli-stream p) (toss-filtration p n) } (\lambda \xi. \text{fortune (Suc n) } \xi - \text{fortune n } \xi) \xi$

using cond-exp-indep[OF bernoulli-stream-natural-filtration.subalg indep-set-natural-filtration
integrable-toss, OF zero-order(1) lessI indep-vars-toss, of p n]
integral-toss[of p Suc n] assms

by (force simp add: fortune-toss-sum)

qed

theorem fortune-supermartingale:

assumes $0 \leq p \leq 1/2$

shows supermartingale (bernoulli-stream p) (toss-filtration p) 0 fortune

proof (intro bernoulli-stream-natural-filtration.supermartingale-of-cond-exp-diff-Suc-le-zero
integrable-fortune fortune.adapted-process-axioms)

fix n

show AE ξ in bernoulli-stream p. $0 \geq \text{cond-exp (bernoulli-stream p) (toss-filtration p n) } (\lambda \xi. \text{fortune (Suc n) } \xi - \text{fortune n } \xi) \xi$

using cond-exp-indep[OF bernoulli-stream-natural-filtration.subalg indep-set-natural-filtration
integrable-toss, OF zero-order(1) lessI indep-vars-toss, of p n]
integral-toss[of p 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. 2023.
- [3] S. Lang. *Real and Functional Analysis*. Springer, 1993.
- [4] G. Zitkovic. Lecture notes on conditional expectation, theory of probability I, UT Austin, Jan 2015. https://web.ma.utexas.edu/users/gordanz/notes/conditional_expectation.pdf.