On the Formalization of Martingales

Ata Keskin

March 17, 2025

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 definitions[1]. We show the relation

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

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

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

The formalization further contributes by generalizing concepts in Bochner integration by extending their application from the real numbers to arbitrary Banach spaces equipped with a second-countable topology. Induction schemes for integrable simple functions on Banach spaces are introduced, accommodating various scenarios with or without a real vector ordering[3]. 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

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* ⇒ (*'a* ⇒ *'b*) *set* ⇒ *'b measure* ⇒ *'a measure* **where**
family-vimage-algebra Ω *S M* ≡ *sigma* Ω (⋃*f* ∈ *S*. {*f* −‘ *A* ∩ Ω | *A*. *A* ∈ *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 (⋃*f* ∈ *S*. {*f* −‘ *A* ∩ Ω | *A*. *A* ∈ *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* ∈ *S f* ∈ Ω → *space M*
shows *f* ∈ *family-vimage-algebra* Ω *S M* →$_M$ *M*
using *assms* **by** (*intro measurableI*, *auto simp add*: *sets-family-vimage-algebra*)

lemma *measurable-family-vimage-algebra-singleton*:
assumes *f* ∈ Ω → *space M*
shows *f* ∈ *family-vimage-algebra* Ω {*f*} *M* →$_M$ *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* ⊆ *N* →$_M$ *M*) ⟷ *S* ⊆ *space N* → *space M* ∧ *family-vimage-algebra* (*space N*) *S M* ⊆ *N*
proof (*standard*, *goal-cases*)
case *1*
hence *subset*: *S* ⊆ *space N* → *space M* **using** *measurable-space* **by** *fast*
have {*f* −‘ *A* ∩ *space N* |*A*. *A* ∈ *M*} ⊆ *N* **if** *f* ∈ *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*: $S \subseteq$ *space* $N \to$ *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* Ω S M = *sigma* Ω (*sets* (*family-vimage-algebra* Ω
($S - I$) M) $\cup$ *family-vimage-algebra* Ω ($S \cap I$) 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 $s : \Omega \to E$ 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* ⇒ *'a measure* ⇒ (*'a* ⇒ *'b*) ⇒ (*'a* ⇒ *'b*::{*real-normed-vector*, *second-countable-topology*}) ⇒ *bool* **where**
has-cond-exp M F f g = ((∀ *A* ∈ *sets F*. (∫ *x* ∈ *A*. *f x* ∂*M*) = (∫ *x* ∈ *A*. *g x* ∂*M*))
∧ *integrable M f*
∧ *integrable M g*
∧ *g* ∈ *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 ⋀*A*. *A* ∈ *sets F* ⟹ (∫ *x* ∈ *A*. *f x* ∂*M*) = (∫ *x* ∈ *A*. *g x* ∂*M*)
integrable M f
integrable M g
g ∈ *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 ⋀*A*. *A* ∈ *sets F* ⟹ (∫ *x* ∈ *A*. *f x* ∂*M*) = (∫ *x* ∈ *A*. *g x* ∂*M*)
integrable M f
integrable M g
g ∈ *borel-measurable F*
using *assms* **unfolding** *has-cond-exp-def* **by** *simp+*

Now we can use Hilbert's ϵ-operator to define the conditional expectation, if it exists.

definition *cond-exp* :: *'a measure* ⇒ *'a measure* ⇒ (*'a* ⇒ *'b*) ⇒ (*'a* ⇒ *'b*::{*banach*, *second-countable-topology*}) **where**
cond-exp M F f = (*if* ∃ *g*. *has-cond-exp M F f g then* (*SOME g*. *has-cond-exp M F f g*) *else* (λ-. *0*))

lemma *borel-measurable-cond-exp*[*measurable*]: *cond-exp M F f* ∈ *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* ∈ *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 ∄*g*. *has-cond-exp M F f g*
shows *cond-exp M F f* = (λ-. *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 in M. *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* = *restr-to-subalg M F*
interpret *sigma-finite-measure ?MF* **by** (*rule sigma-fin-subalg*)
{
fix *A* **assume** *A* ∈ *sets ?MF*
then have [*measurable*]: *A* ∈ *sets F* **using** *sets-restr-to-subalg*[*OF subalg*] **by** *simp*
have (∫ *x* ∈ *A*. *g x* ∂*?MF*) = (∫ *x* ∈ *A*. *g x* ∂*M*) **using** *assms subalg* **by** (*auto simp add*: *integral-subalgebra2 set-lebesgue-integral-def dest*!: *has-cond-expD*)
also have ... = (∫ *x* ∈ *A*. *cond-exp M F f x* ∂*M*) **using** *assms cond-exp* **by** (*simp add*: *has-cond-exp-def*)
also have ... = (∫ *x* ∈ *A*. *cond-exp M F f x* ∂*?MF*) **using** *subalg* **by** (*auto simp add*: *integral-subalgebra2 set-lebesgue-integral-def*)
finally have (∫ *x* ∈ *A*. *g x* ∂*?MF*) = (∫ *x* ∈ *A*. *cond-exp M F f x* ∂*?MF*) **by** *simp*
}
hence *AE x in ?MF*. *cond-exp M F f x* = *g x* **using** *cond-exp assms subalg* **by** (*intro density-unique-banach*, *auto dest*: *has-cond-expD intro*!: *integrable-in-subalg*)
then show *AE x in M*. *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* ⇒ *'b*::{*second-countable-topology*, *banach*}
assumes ⋀*A*. *A* ∈ *sets F* ⟹ (∫ *x* ∈ *A*. *f x* ∂*M*) = (∫ *x* ∈ *A*. *g x* ∂*M*)
integrable M f
integrable M g
g ∈ *borel-measurable F*
shows *AE x in M*. *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 *cond-exp M F f* = f μ-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* ⇒ *'b*::{*second-countable-topology*, *banach*}
assumes *integrable M f*
f ∈ *borel-measurable F*
shows *AE x in M*. *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* ⋀*x*. *x* ∈ *space M* ⟹ *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 = 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 = 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* ⇒ *'b*::{*second-countable-topology*,*banach*}
assumes *integrable M f integrable M g* ⋀*x*. *x* ∈ *space M* ⟹ *f x* = *g x*
shows *AE x in M*. *cond-exp M F f x* = *cond-exp M F g x*
proof (*cases* ∃ *h*. *has-cond-exp M F f h*)
case *True*
then obtain *h* **where** *h*: *has-cond-exp M F f h has-cond-exp M F g h* **using** *has-cond-exp-cong assms* **by** *metis*
show *?thesis* **using** *h*[*THEN has-cond-exp-charact*(*2*)] **by** *fastforce*
next
case *False*
moreover have ∄ *h*. *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* ∈ *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* ⇒ *'b*::{*second-countable-topology*,*banach*}
assumes *integrable M f integrable M g AE x in M*. *f x* = *g x*
shows *AE x in M*. *cond-exp M F f x* = *cond-exp M F g x*
proof (*cases* ∃ *h*. *has-cond-exp M F f h*)
case *True*
then obtain *h* **where** *h*: *has-cond-exp M F f h has-cond-exp M F g h* **using** *has-cond-exp-cong-AE assms* **by** (*metis* (*mono-tags*, *lifting*) *eventually-mono*)

show *?thesis* **using** *h*[*THEN has-cond-exp-charact*(*2*)] **by** *fastforce*
next
case *False*
moreover have ∄*h*. *has-cond-exp M F g h* **using** *False has-cond-exp-cong-AE assms* **by** *auto*
ultimately show *?thesis* **unfolding** *cond-exp-def* **by** *auto*
qed

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

lemma *has-cond-exp-real*:
fixes *f* :: *'a* ⇒ *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* ⇒ *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* ⇒ *real*
assumes *integrable M f*
shows *AE x in M*. *cond-exp M F* (λ*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* ∈ *sets M emeasure M A* < ∞
shows *has-cond-exp M F* (λ*x*. *indicat-real A x* ∗$_R$ *y*) (λ*x*. *real-cond-exp M F* (*indicator A*) *x* ∗$_R$ *y*)
proof (*intro has-cond-expI'*, *goal-cases*)
case (*1 B*)
have (∫ *x*∈*B*. (*indicat-real A x* ∗$_R$ *y*) ∂*M*) = (∫ *x*∈*B*. *indicat-real A x* ∂*M*) ∗$_R$ *y* **using** *assms* **by** (*intro set-integral-scaleR-left*, *meson 1 in-mono subalg subalgebra-def*, *blast*)

also have ... = $(\int x{\in}B.$ *real-cond-exp M F* (*indicator A*) *x* $\partial M) *_R y$ **using** *1 assms* **by** (*subst real-cond-exp-intA*, *auto*)
also have ... = $(\int x{\in}B.$ (*real-cond-exp M F* (*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*::{*second-countable-topology*,*banach*}
assumes [*measurable*]: $A \in$ *sets M emeasure M A* $< \infty$
shows *AE x in M. cond-exp M F* ($\lambda x.$ *indicat-real A x* $*_R y) x =$ *cond-exp M F* (*indicator A*) *x* $*_R y$
proof −
have *AE x in M. cond-exp M F* ($\lambda x.$ *indicat-real A x* $*_R y) x =$ *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** *fastforce*
qed

Addition

lemma *has-cond-exp-add*:
fixes *f g* :: *'a* ⇒ *'b*::{*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 ⇒ 'b::{second-countable-topology,banach}
  assumes has-cond-exp M F f f'
  shows has-cond-exp M F (λx. c *R f x) (λx. c *R f' x)
  using has-cond-expD[OF assms] by (intro has-cond-expI', auto)

lemma cond-exp-scaleR-right:
  fixes f :: 'a ⇒ 'b::{second-countable-topology,banach}
  assumes integrable M f
  shows AE x in M. cond-exp M F (λx. c *R f x) x = c *R cond-exp M F f x
proof (cases ∃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 ∄f'. has-cond-exp M F (λx. c *R f x) f'
    proof (standard, goal-cases)
      case 1
      then obtain f' where f': has-cond-exp M F (λx. c *R f x) f' by blast
       have has-cond-exp M F f (λx. inverse c *R f' x) using has-cond-expD[OF
f'] divideR-right[OF c-nonzero] assms by (intro has-cond-expI', auto)
      then show ?case using False by blast
    qed
    then show ?thesis using cond-exp-null[OF False] cond-exp-null by force
  qed
qed

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

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

corollary *has-cond-exp-simple*:
fixes $f :: 'a \Rightarrow 'b$::{*second-countable-topology*,*banach*}
assumes *simple-function M f emeasure M* $\{y \in$ *space M*. $f\ y \neq 0\} \neq \infty$
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 $\lambda n.$ *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 convergences in the L1-norm. Luckily, this is enough to show that the operator *cond-exp M F* preserves limits as a function from L1 to L1.

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

lemma *cond-exp-contraction-real*:
fixes $f :: 'a \Rightarrow$ *real*
assumes *integrable*[*measurable*]: *integrable M f*
shows *AE x in M*. *norm* (*cond-exp M F f x*) $\leq$ *cond-exp M F* ($\lambda x.$ *norm* (*f x*)) *x*
proof −
have *int*: *integrable M* ($\lambda x.$ *norm* (*f x*)) **using** *assms* **by** *blast*
have $*$: *AE x in M*. $0 \leq$ *cond-exp M F* ($\lambda x.$ *norm* (*f x*)) *x* **using** *cond-exp-real*[*THEN AE-symmetric*, *OF integrable-norm*[*OF integrable*]] *real-cond-exp-ge-c*[*OF integrable-norm*[*OF integrable*], *of 0*] *norm-ge-zero* **by** *fastforce*
have $**$: *A* $\in$ *sets F* $\Longrightarrow$ ($\int x{\in}A.$ $|f\ x|$ ∂M) = ($\int x{\in}A.$ *real-cond-exp M F* ($\lambda x.$ *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* $\in$ *sets F* $\Longrightarrow$ ($\int x{\in}A.$ $|f\ x|$ ∂M) = ($\int^+ x{\in}A.$ $|f\ x|$ ∂M) **for** *A*

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

have *AE x in M. real-cond-exp M F* (λ*x. norm* (*f x*)) *x* ≥ *0* **using** *int real-cond-exp-ge-c* **by** *force*
hence *cond-exp-norm-int*: *A* ∈ *sets F* ⟹ (∫ *x*∈*A. real-cond-exp M F* (λ*x. norm* (*f x*)) *x* ∂*M*) = (∫ ⁺*x*∈*A. real-cond-exp M F* (λ*x. norm* (*f x*)) *x* ∂*M*) **for** *A* **using** *assms* **by** (*intro nn-set-integral-eq-set-integral*[*symmetric*], *blast*, *fastforce*) (*meson subalg subalgebra-def subsetD*)

have *A* ∈ *sets F* ⟹ (∫ ⁺*x*∈*A.* |*f x*|∂*M*) = (∫ ⁺*x*∈*A. real-cond-exp M F* (λ*x. norm* (*f x*)) *x* ∂*M*) **for** *A* **using** ∗∗ *norm-int cond-exp-norm-int* **by** (*auto simp add*: *nn-integral-set-ennreal*)
moreover have (λ*x. ennreal* |*f x*|) ∈ *borel-measurable M* **by** *measurable*
moreover have (λ*x. ennreal* (*real-cond-exp M F* (λ*x. norm* (*f x*)) *x*)) ∈ *borel-measurable F* **by** *measurable*
ultimately have *AE x in M. nn-cond-exp M F* (λ*x. ennreal* |*f x*|) *x* = *real-cond-exp M F* (λ*x. norm* (*f x*)) *x* **by** (*intro nn-cond-exp-charact*[*THEN AE-symmetric*], *auto*)
hence *AE x in M. nn-cond-exp M F* (λ*x. ennreal* |*f x*|) *x* ≤ *cond-exp M F* (λ*x. norm* (*f x*)) *x* **using** *cond-exp-real*[*OF int*] **by** *force*
moreover have *AE x in M.* |*real-cond-exp M F f x*| = *norm* (*cond-exp M F f x*) **unfolding** *real-norm-def* **using** *cond-exp-real*[*OF assms*] ∗ **by** *force*
ultimately have *AE x in M. ennreal* (*norm* (*cond-exp M F f x*)) ≤ *cond-exp M F* (λ*x. norm* (*f x*)) *x* **using** *real-cond-exp-abs*[*OF assms*[*THEN borel-measurable-integrable*]] **by** *fastforce*
hence *AE x in M. enn2real* (*ennreal* (*norm* (*cond-exp M F f x*))) ≤ *enn2real* (*cond-exp M F* (λ*x. norm* (*f x*)) *x*) **using** *ennreal-le-iff2* **by** *force*
thus *?thesis* **using** ∗ **by** *fastforce*
qed

lemma *cond-exp-contraction-simple*:
fixes *f* :: *'a* ⇒ *'b*::{*second-countable-topology*, *banach*}
assumes *simple-function M f emeasure M* {*y* ∈ *space M. f y* ≠ *0*} ≠ ∞
shows *AE x in M. norm* (*cond-exp M F f x*) ≤ *cond-exp M F* (λ*x. norm* (*f x*)) *x*
using *assms*
proof (*induction rule*: *integrable-simple-function-induct*)
case (*cong f g*)
hence *ae*: *AE x in M. f x* = *g x* **by** *blast*
hence *AE x in M. cond-exp M F f x* = *cond-exp M F g x* **using** *cong has-cond-exp-simple* **by** (*subst cond-exp-cong-AE*) (*auto intro*!: *has-cond-expD*(*2*))
hence *AE x in M. norm* (*cond-exp M F f x*) = *norm* (*cond-exp M F g x*) **by** *force*
moreover have *AE x in M. cond-exp M F* (λ*x. norm* (*f x*)) *x* = *cond-exp M F* (λ*x. norm* (*g x*)) *x* **using** *ae cong has-cond-exp-simple* **by** (*subst cond-exp-cong-AE*) (*auto dest*: *has-cond-expD*)
ultimately show *?case* **using** *cong*(*6*) **by** *fastforce*
next
case (*indicator A y*)

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

have *AE x in M. norm y* $*$ *cond-exp M F* (λx. *norm* (*indicat-real A x*)) *x* = *norm y* $*$ *real-cond-exp M F* (λx. *norm* (*indicat-real A x*)) *x* **using** *cond-exp-real*[*OF integrable-real-indicator*, *OF indicator*] **by** *fastforce*
moreover have *AE x in M. cond-exp M F* (λx. *norm y* $*$ *norm* (*indicat-real A x*)) *x* = *real-cond-exp M F* (λx. *norm y* $*$ *norm* (*indicat-real A x*)) *x* **using** *indicator* **by** (*intro cond-exp-real*, *auto*)
ultimately have *AE x in M. norm y* $*$ *cond-exp M F* (λx. *norm* (*indicat-real A x*)) *x* = *cond-exp M F* (λx. *norm y* $*$ *norm* (*indicat-real A x*)) *x* **using** *real-cond-exp-cmult*[*of* λx. *norm* (*indicat-real A x*) *norm y*] *indicator* **by** *fastforce*
moreover have (λx. *norm y* $*$ *norm* (*indicat-real A x*)) = (λx. *norm* (*indicat-real A x* $*_R$ *y*)) **by** *force*
ultimately show *?case* **using** $*$ **by** *force*
next
case (*add u v*)
have *AE x in M. norm* (*cond-exp M F* (λa. *u a* + *v a*) *x*) = *norm* (*cond-exp M F u x* + *cond-exp M F v x*) **using** *has-cond-exp-charact*(*2*)[*OF has-cond-exp-add*, *OF has-cond-exp-simple*(*1*,*1*), *OF add*(*1*,*2*,*3*,*4*)] **by** *fastforce*
moreover have *AE x in M. norm* (*cond-exp M F u x* + *cond-exp M F v x*) $\leq$ *norm* (*cond-exp M F u x*) + *norm* (*cond-exp M F v x*) **using** *norm-triangle-ineq* **by** *blast*
moreover have *AE x in M. norm* (*cond-exp M F u x*) + *norm* (*cond-exp M F v x*) $\leq$ *cond-exp M F* (λx. *norm* (*u x*)) *x* + *cond-exp M F* (λx. *norm* (*v x*)) *x* **using** *add*(*6*,*7*) **by** *fastforce*
moreover have *AE x in M. cond-exp M F* (λx. *norm* (*u x*)) *x* + *cond-exp M F* (λx. *norm* (*v x*)) *x* = *cond-exp M F* (λx. *norm* (*u x*) + *norm* (*v x*)) *x* **using** *integrable-simple-function*[*OF add*(*1*,*2*)] *integrable-simple-function*[*OF add*(*3*,*4*)] **by** (*intro has-cond-exp-charact*(*2*)[*OF has-cond-exp-add*[*OF has-cond-exp-charact*(*1*,*1*)], *THEN AE-symmetric*], *auto intro*: *has-cond-exp-real*)
moreover have *AE x in M. cond-exp M F* (λx. *norm* (*u x*) + *norm* (*v x*)) *x* = *cond-exp M F* (λx. *norm* (*u x* + *v x*)) *x* **using** *add*(*5*) *integrable-simple-function*[*OF add*(*1*,*2*)] *integrable-simple-function*[*OF add*(*3*,*4*)] **by** (*intro cond-exp-cong*, *auto*)
ultimately show *?case* **by** *force*
qed

lemma *has-cond-exp-simple-lim*:
fixes *f* :: *'a* $\Rightarrow$ *'b*::{*second-countable-topology*, *banach*}
assumes *integrable*[*measurable*]: *integrable M f*
and $\bigwedge i$. *simple-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* $\Longrightarrow$ (λi. *s i x*) $\longrightarrow$ *f x*
and $\bigwedge x\ i$. *x* $\in$ *space M* $\Longrightarrow$ *norm* (*s i x*) $\leq$ *2* $*$ *norm* (*f x*)
obtains *r*
where *strict-mono r has-cond-exp M F f* (λx. *lim* (λi. *cond-exp M F* (*s* (*r i*))

x))
AE x in M. convergent (λ*i. cond-exp M F* (*s* (*r i*)) *x*)
proof −
have [*measurable*]: (*s i*) ∈ *borel-measurable M* **for** *i* **using** *assms*(*2*) **by** (*simp add*: *borel-measurable-simple-function*)
have *integrable-s*: *integrable M* (λ*x. s i x*) **for** *i* **using** *assms integrable-simple-function* **by** *blast*
have *integrable-4f*: *integrable M* (λ*x. 4* ∗ *norm* (*f x*)) **using** *assms*(*1*) **by** *simp*
have *integrable-2f*: *integrable M* (λ*x. 2* ∗ *norm* (*f x*)) **using** *assms*(*1*) **by** *simp*
have *integrable-2-cond-exp-norm-f*: *integrable M* (λ*x. 2* ∗ *cond-exp M F* (λ*x. norm* (*f x*)) *x*) **by** *fast*

have *emeasure M* {*y* ∈ *space M. s i y* − *s j y* ≠ *0*} ≤ *emeasure M* {*y* ∈ *space M. s i y* ≠ *0*} + *emeasure M* {*y* ∈ *space M. s j y* ≠ *0*} **for** *i j* **using** *simple-functionD*(*2*)[*OF assms*(*2*)] **by** (*intro order-trans*[*OF emeasure-mono emeasure-subadditive*], *auto*)
hence *fin-sup*: *emeasure M* {*y* ∈ *space M. s i y* − *s j y* ≠ *0*} ≠ ∞ **for** *i j* **using** *assms*(*3*) **by** (*metis* (*mono-tags*) *ennreal-add-eq-top linorder-not-less top.not-eq-extremum infinity-ennreal-def*)

have *emeasure M* {*y* ∈ *space M. norm* (*s i y* − *s j y*) ≠ *0*} ≤ *emeasure M* {*y* ∈ *space M. s i y* ≠ *0*} + *emeasure M* {*y* ∈ *space M. s j y* ≠ *0*} **for** *i j* **using** *simple-functionD*(*2*)[*OF assms*(*2*)] **by** (*intro order-trans*[*OF emeasure-mono emeasure-subadditive*], *auto*)
hence *fin-sup-norm*: *emeasure M* {*y* ∈ *space M. norm* (*s i y* − *s j y*) ≠ *0*} ≠ ∞ **for** *i j* **using** *assms*(*3*) **by** (*metis* (*mono-tags*) *ennreal-add-eq-top linorder-not-less top.not-eq-extremum infinity-ennreal-def*)

have *Cauchy*: *Cauchy* (λ*n. s n x*) **if** *x* ∈ *space M* **for** *x* **using** *assms*(*4*) *LIMSEQ-imp-Cauchy that* **by** *blast*
hence *bounded-range-s*: *bounded* (*range* (λ*n. s n x*)) **if** *x* ∈ *space M* **for** *x* **using** *that cauchy-imp-bounded* **by** *fast*

Since the sequence λ*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.* (λ*n. diameter* {*s i x* | *i. n* ≤ *i*}) ⟶ *0* **using** *Cauchy cauchy-iff-diameter-tends-to-zero-and-bounded* **by** *fast*
moreover have (λ*x. diameter* {*s i x* |*i. n* ≤ *i*}) ∈ *borel-measurable M* **for** *n* **using** *bounded-range-s borel-measurable-diameter* **by** *measurable*
moreover have *AE x in M. norm* (*diameter* {*s i x* |*i. n* ≤ *i*}) ≤ *4* ∗ *norm* (*f x*) **for** *n*
proof −
{
fix *x* **assume** *x*: *x* ∈ *space M*
have *diameter* {*s i x* |*i. n* ≤ *i*} ≤ *2* ∗ *norm* (*f x*) + *2* ∗ *norm* (*f x*) **by** (*intro diameter-le*, *blast*, *subst dist-norm*[*symmetric*], *intro dist-triangle3*[*THEN*

order-trans, *of 0*], *intro add-mono*) (*auto intro*: *assms*(*5*)[*OF x*])
hence *norm* (*diameter* {*s i x* |*i*. *n* ≤ *i*}) ≤ *4* ∗ *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*: (λ*n*. *LINT x*|*M*. *diameter* {*s i x* | *i*. *n* ≤ *i*}) ⟶ *0* **by** (*intro integral-dominated-convergence*[*OF borel-measurable-const*[*of 0*] - *integrable-4f*, *simplified*]) (*fast*+)

have *diameter-integrable*: *integrable M* (λ*x*. *diameter* {*s i x* | *i*. *n* ≤ *i*}) **for** *n* **using** *assms*(*1*,*5*)
by (*intro integrable-bound-diameter*[*OF bounded-range-s integrable-2f*], *auto*)

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

Since *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 ∃ *N*. ∀ *i*≥*N*. ∀ *j*≥*N*. *LINT x*|*M*. *norm* (*cond-exp M F* (*s i*) *x* − *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* ≥ *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* ≥ *N j* ≥ *N x* ∈ *space M*
have *case-prod dist* ‘ ({*s i x* |*i*. *N* ≤ *i*} × {*s i x* |*i*. *N* ≤ *i*}) = *case-prod* (λ*i j*. *dist* (*s i x*) (*s j x*)) ‘ ({*N*..} × {*N*..}) **by** *fast*
hence *diameter* {*s i x* | *i*. *N* ≤ *i*} = (*SUP* (*i*, *j*) ∈ {*N*..} × {*N*..}. *dist* (*s i x*) (*s j x*)) **unfolding** *diameter-def* **by** *auto*
moreover have (*SUP* (*i*, *j*) ∈ {*N*..} × {*N*..}. *dist* (*s i x*) (*s j x*)) ≥ *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*} ≥ *dist* (*s i x*) (*s j x*) **by** *presburger*
}
hence *LINT x*|*M*. *dist* (*s i x*) (*s j x*) < *e* **if** *i* ≥ *N j* ≥ *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*) ≤ *LINT x*|*M*. *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 (λx. 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 *... ≤ LINT x|M. cond-exp M F (λx. 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[unfolded dist-norm], symmetric]* **by** *(intro has-cond-expD(1)[OF has-cond-exp-simple[OF - fin-sup-norm], symmetric]) (metis assms(2) simple-function-compose1 simple-function-diff, metis sets.top subalg subalgebra-def)*
finally show *?thesis* **unfolding** *dist-norm* **.**
qed
ultimately show *?thesis* **using** *order.strict-trans1* **by** *meson*
qed
then obtain *r* **where** *strict-mono-r*: *strict-mono r* **and** *AE-Cauchy*: *AE x in M. Cauchy (λi. 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. (λn. cond-exp M F (s (r n)) x) ⟶ lim (λn. 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) ≤ cond-exp M F (λx. 2 ∗ norm (f x)) x* **for** *n*
proof −
have *AE x in M. norm (cond-exp M F (s (r n)) x) ≤ cond-exp M F (λx. 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 (λx. norm (s (r n) x)) x ≤ real-cond-exp M F (λx. 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 (λx. lim (λi. 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 $A \in sets\ F$.

```
{
  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-function M* (*s i*) $\bigwedge i$. *emeasure M* {*y* ∈ *space M*. *s i y* ≠ *0*} ≠ ∞ $\bigwedge x$. *x* ∈ *space M* ⟹ (λ*i*. *s i x*) ⟶ *f x* $\bigwedge x$ *i*. *x* ∈ *space M* ⟹ *norm* (*s i x*) ≤ *2* ∗ *norm* (*f x*) **using** *integrable-implies-simple-function-sequence*[*OF assms*] **by** *blast*
show *?thesis* **using** *has-cond-exp-simple-lim*[*OF assms s-is*] *has-cond-exp-charact*(*1*)
by *metis*
qed

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 (∫ *x* ∈ *A*. *f x* ∂*M*) = (∫ *x* ∈ *A*. *cond-exp M F f x* ∂*M*)
using *has-cond-expD*(*1*)[*OF has-cond-expI*, *OF assms*] **by** *argo*

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

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

The conditional expectation is linear.

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

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

using *has-cond-exp-add*[*OF - has-cond-exp-scaleR-right*, *OF has-cond-expI*(*1*,*1*), *OF assms*, *THEN has-cond-exp-charact*(*2*), *of* −*1*] **by** *simp*

lemma *cond-exp-diff'*:
fixes $f :: 'a \Rightarrow 'b :: \{$*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 $f :: 'a \Rightarrow$ *real*
assumes *integrable M f*
shows *AE x in M*. *cond-exp M F* (λ*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 λ*x i*. *s* (*r i*) *x*, such that λ*i*. *cond-exp M F* (*s* (*r i*)) *x* converges to *cond-exp M F f x* a.e. Afterwards, we obtain a sub-subsequence λ*x i*. *s* (*r* (*r' i*)) *x*, such that λ*i*. *cond-exp M F* (λ*x*. *norm* (*s* (*r i*))) *x* converges to *cond-exp M F* (λ*x*. *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 $f :: 'a \Rightarrow 'b ::\{$*second-countable-topology*, *banach*$\}$
assumes *integrable M f*
shows *AE x in M*. *norm* (*cond-exp M F f x*) ≤ *cond-exp M F* (λ*x*. *norm* (*f x*)) *x*
proof −
obtain *s* **where** *s*: $\bigwedge i$. *simple-function M* (*s i*) $\bigwedge i$. *emeasure M* {*y* ∈ *space M*. *s i y* ≠ *0*} ≠ ∞ $\bigwedge x$. *x* ∈ *space M* ⟹ (λ*i*. *s i x*) ⟶ *f x* $\bigwedge i$ *x*. *x* ∈ *space M* ⟹ *norm* (*s i x*) ≤ *2* ∗ *norm* (*f x*)
by (*blast intro*: *integrable-implies-simple-function-sequence*[*OF assms*])

obtain *r* **where** *r*: *strict-mono r* **and** *has-cond-exp M F f* (λ*x*. *lim* (λ*i*. *cond-exp M F* (*s* (*r i*)) *x*)) *AE x in M*. (λ*i*. *cond-exp M F* (*s* (*r i*)) *x*) ⟶ *lim* (λ*i*. *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*. (λ*i*. *cond-exp M F* (*s* (*r i*)) *x*) ⟶ *cond-exp M F f x* **using** *has-cond-exp-charact*(*2*) **by** *force*

have *norm-s-r*: $\bigwedge i.$ *simple-function M* $(\lambda x.$ *norm* $(s\ (r\ i)\ x))$ $\bigwedge i.$ *emeasure M* $\{y \in$ *space M*. *norm* $(s\ (r\ i)\ y) \neq 0\} \neq \infty$ $\bigwedge x.$ $x \in$ *space M* $\Longrightarrow$ $(\lambda i.$ *norm* $(s\ (r\ i)\ x)) \longrightarrow$ *norm* $(f\ x)$ $\bigwedge i\ x.$ $x \in$ *space M* $\Longrightarrow$ *norm* $(norm\ (s\ (r\ i)\ x)) \leq 2 *$ *norm* $(norm\ (f\ x))$
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* $(\lambda x.$ *norm* $(f\ x))$ $(\lambda x.$ *lim* $(\lambda i.$ *cond-exp M F* $(\lambda x.$ *norm* $(s\ (r\ (r'\ i))\ x))\ x))$ *AE x in M*. $(\lambda i.$ *cond-exp M F* $(\lambda x.$ *norm* $(s\ (r\ (r'\ i))\ x))\ x) \longrightarrow$ *lim* $(\lambda i.$ *cond-exp M F* $(\lambda x.$ *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 in M*. $(\lambda i.$ *cond-exp M F* $(\lambda x.$ *norm* $(s\ (r\ (r'\ i))\ x))\ x) \longrightarrow$ *cond-exp M F* $(\lambda x.$ *norm* $(f\ x))\ x$ **using** *has-cond-exp-charact*(*2*) **by** *force*

have *AE x in M*. $\forall i.$ *norm* $(cond\text{-}exp\ M\ F\ (s\ (r\ (r'\ i)))\ x) \leq$ *cond-exp M F* $(\lambda x.$ *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 in M*. $(\lambda i.$ *norm* $(cond\text{-}exp\ M\ F\ (s\ (r\ (r'\ i)))\ x)) \longrightarrow$ *norm* $(cond\text{-}exp\ M\ F\ f\ x)$ **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 $f\ g :: 'a \Rightarrow real$
assumes [*measurable*]: *integrable M* $(\lambda x.\ f\ x * g\ x)$ *integrable M g f* $\in$ *borel-measurable F*
shows *integrable M* $(\lambda x.\ f\ x *$ *cond-exp M F g x*$)$
AE x in M. *cond-exp M F* $(\lambda x.\ f\ x * g\ x)\ x = f\ x *$ *cond-exp M F g x*
proof−
show *integrable*: *integrable M* $(\lambda x.\ f\ x *$ *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*: $A \in$ *sets F*
hence *asm'*: $A \in$ *sets M* **using** *subalg* **by** (*fastforce simp add*: *subalgebra-def*)
have *set-lebesgue-integral M A* $(cond\text{-}exp\ M\ F\ (\lambda x.\ f\ x * g\ x)) =$ *set-lebesgue-integral M A* $(\lambda x.\ f\ x * g\ x)$ **by** (*simp add*: *cond-exp-set-integral*[*OF assms*(*1*) *asm*])
also have ... $=$ *set-lebesgue-integral M A* $(\lambda x.\ f\ x *$ *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* (λx. *f x* ∗ *cond-exp M F g x*) **using** *cond-exp-real*[*OF assms*(*2*)] *asm′ borel-measurable-cond-exp′ borel-measurable-cond-exp2 measurable-from-subalg*[*OF subalg assms*(*3*)] **by** (*auto simp add*: *set-lebesgue-integral-def intro*: *integral-cong-AE*)

finally have *set-lebesgue-integral M A* (*cond-exp M F* (λx. *f x* ∗ *g x*)) = (∫ *x*∈*A*. (*f x* ∗ *cond-exp M F g x*)∂*M*) **.**

}

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

thus *AE x in M*. *cond-exp M F* (λx. *f x* ∗ *g x*) *x* = *f x* ∗ *cond-exp M F g x* **by** (*rule AE-restr-to-subalg*[*OF subalg*])

qed

lemma *cond-exp-measurable-scaleR*:

fixes *f* :: *′a* ⇒ *real* **and** *g* :: *′a* ⇒ *′b* :: {*second-countable-topology*, *banach*}

assumes [*measurable*]: *integrable M* (λx. *f x* ∗*R g x*) *integrable M g f* ∈ *borel-measurable F*

shows *integrable M* (λx. *f x* ∗*R cond-exp M F g x*)

AE x in M. *cond-exp M F* (λx. *f x* ∗*R g x*) *x* = *f x* ∗*R cond-exp M F g x*

proof −

let *?F* = *restr-to-subalg M F*

have *subalg′*: *subalgebra M* (*restr-to-subalg M F*) **by** (*metis sets-eq-imp-space-eq sets-restr-to-subalg subalg subalgebra-def*)

{

fix *z* **assume** *asm*[*measurable*]: *integrable M* (λx. *z x* ∗*R g x*) *z* ∈ *borel-measurable ?F*

hence *asm′*[*measurable*]: *z* ∈ *borel-measurable F* **using** *measurable-in-subalg′ subalg* **by** *blast*

have *integrable M* (λx. *z x* ∗*R cond-exp M F g x*) *LINT x*|*M*. *z x* ∗*R g x* = *LINT x*|*M*. *z x* ∗*R cond-exp M F g x*

proof −

obtain *s* **where** *s-is*: ⋀*i*. *simple-function ?F* (*s i*) ⋀*x*. *x* ∈ *space ?F* ⟹ (λ*i*. *s i x*) ⟶ *z x* ⋀*i x*. *x* ∈ *space ?F* ⟹ *norm* (*s i x*) ≤ *2* ∗ *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*. (λ*i*. *s i x* ∗*R g x*) ⟶ *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*. (λ*i*. *s i x* ∗*R cond-exp M F g x*) ⟶ *z x* ∗*R cond-exp M F g x* **using** *s-is*(*2*) **by** (*simp add*: *tendsto-scaleR*)

have *s-scaleR-g-meas*: (λx. *s i x* ∗*R g x*) ∈ *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*: (λx. *s i x* ∗*R cond-exp M F g x*) ∈ *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*) $\leq$ *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* (λ*x. 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*) $\leq$ *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* (λ*x. norm* (*g x*)) *x* = *cond-exp M F* (λ*x. 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*) $\leq$ *2* $*$ *cond-exp M F* (λ*x. 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* $\in$ *borel-measurable M* **by** (*meson borel-measurable-simple-function measurable-from-subalg s-is*(*1*) *subalg'*)

have *s-meas-F*[*measurable*]: *s i* $\in$ *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* = ($\sum$ *y*$\in$*s i* ‘ *space M*. (*indicator* (*s i* $-$‘ {*y*} $\cap$ *space M*) *x* $*_R$ *y*) $*_R$ *h x*) **if** *x* $\in$ *space M* **for** *x* **and** *h* :: *'a* $\Rightarrow$ *'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.* ($\sum$ *y*$\in$*s i* ‘ *space M. indicator* (*s i* $-$‘ {*y*} $\cap$ *space M*) *x* $*_R$ *y* $*_R$ *g x*) **using** *s-scaleR-eq* **by** (*intro Bochner-Integration.integral-cong*) *auto*

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

also have ... = ($\sum$ *y*$\in$*s i* ‘ *space M. y* $*_R$ *set-lebesgue-integral M* (*s i* $-$‘ {*y*} $\cap$ *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 ... = ($\sum$ *y*$\in$*s i* ‘ *space M. LINT x*|*M. indicator* (*s i* $-$‘ {*y*} $\cap$ *space*

M) *x* $*_R$ *y* $*_R$ *cond-exp M F g x*) **by** (*simp only*: *set-lebesgue-integral-def*[*symmetric*]) *simp*

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

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

finally have *LINT x*|*M*. *s i x* $*_R$ *g x* = *LINT x*|*?F*. *s i x* $*_R$ *cond-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* (λ*x*. *z x* $*_R$ *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 (λ*i*. *LINT x*|*M*. *s i x* $*_R$ *g x*) ⟶ *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 (λ*i*. *LINT x*|*?F*. *s i x* $*_R$ *cond-exp M F g x*) ⟶ *LINT x*|*?F*. *z x* $*_R$ *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 *LINT x*|*M*. *z x* $*_R$ *g x* = *LINT x*|*M*. *z x* $*_R$ *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* = (λ*x*. *indicat-real A x* ∗ *f x*).

show *integrable M* (λ*x*. *f x* $*_R$ *cond-exp M F g x*) **using** ∗ *assms measurable-in-subalg*[*OF subalg*] **by** *blast*

{

fix *A* **assume** *asm*: *A* ∈ *F*

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

}

thus *AE x in M*. *cond-exp M F* (λ*x*. *f x* $*_R$ *g x*) *x* = *f x* $*_R$ *cond-exp M F g x*

using *borel-measurable-cond-exp* **by** (*intro cond-exp-charact*, *auto intro*!: ∗ *assms measurable-in-subalg*[*OF subalg*])
qed

lemma *cond-exp-sum* [*intro*, *simp*]:
fixes $f :: 't \Rightarrow 'a \Rightarrow 'b :: \{$*second-countable-topology*,*banach*$\}$
assumes [*measurable*]: $\bigwedge i.$ *integrable M* (*f i*)
shows *AE x in M*. *cond-exp M F* ($\lambda x. \sum i{\in}I.$ *f i x*) *x* = ($\sum i{\in}I.$ *cond-exp M F* (*f i*) *x*)
proof (*rule has-cond-exp-charact*, *intro has-cond-expI'*)
fix *A* **assume** [*measurable*]: *A* ∈ *sets F*
then have *A-meas* [*measurable*]: *A* ∈ *sets M* **by** (*meson subsetD subalg subalgebra-def*)

have ($\int x{\in}A.$ ($\sum i{\in}I.$ *f i x*)∂*M*) = ($\int x.$ ($\sum i{\in}I.$ *indicator A x* ∗$_R$ *f i x*)∂*M*)
unfolding *set-lebesgue-integral-def* **by** (*simp add*: *scaleR-sum-right*)
also have ... = ($\sum i{\in}I.$ ($\int x.$ *indicator A x* ∗$_R$ *f i x* ∂*M*)) **using** *assms* **by** (*auto intro*!: *Bochner-Integration.integral-sum integrable-mult-indicator*)
also have ... = ($\sum i{\in}I.$ ($\int x.$ *indicator A x* ∗$_R$ *cond-exp M F* (*f i*) *x* ∂*M*)) **using** *cond-exp-set-integral*[*OF assms*] **by** (*simp add*: *set-lebesgue-integral-def*)
also have ... = ($\int x.$ ($\sum i{\in}I.$ *indicator A x* ∗$_R$ *cond-exp M F* (*f i*) *x*)∂*M*) **using** *assms* **by** (*auto intro*!: *Bochner-Integration.integral-sum*[*symmetric*] *integrable-mult-indicator*)
also have ... = ($\int x{\in}A.$ ($\sum i{\in}I.$ *cond-exp M F* (*f i*) *x*)∂*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*)∂*M*) = ($\int x{\in}A.$ ($\sum i{\in}I.$ *cond-exp M F* (*f i*) *x*)∂*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 $f :: 'a \Rightarrow 'b :: \{$*second-countable-topology*, *banach*, *linorder-topology*, *ordered-real-vector*$\}$
assumes *integrable M f* *AE x in M*. *f x* > *c*
shows *AE x in M*. *cond-exp M F f x* > *c*
proof −
define *X* **where** *X* = $\{x \in$ *space M*. *cond-exp M F f x* ≤ *c*$\}$
have [*measurable*]: *X* ∈ *sets F* **unfolding** *X-def* **by** *measurable* (*metis sets.top subalg subalgebra-def*)
hence *X-in-M*: *X* ∈ *sets M* **using** *sets-restr-to-subalg subalg subalgebra-def* **by** *blast*
have *emeasure M X* = *0*
proof (*rule ccontr*)
assume *emeasure M X* ≠ *0*
have *emeasure* (*restr-to-subalg M F*) *X* = *emeasure M X* **by** (*simp add*: *emeasure-restr-to-subalg subalg*)

hence *emeasure* (*restr-to-subalg M F*) *X* > *0* **using** ‹¬(*emeasure M X*) = *0*› *gr-zeroI* **by** *auto*
then obtain *A* **where** *A*: *A* ∈ *sets* (*restr-to-subalg M F*) *A* ⊆ *X emeasure* (*restr-to-subalg M F*) *A* > *0 emeasure* (*restr-to-subalg M F*) *A* < ∞
using *sigma-fin-subalg* **by** (*metis emeasure-notin-sets ennreal-0 infinity-ennreal-def le-less-linear neq-top-trans not-gr-zero order-refl sigma-finite-measure.approx-PInf-emeasure-with-finite*)
hence [*simp*]: *A* ∈ *sets F* **using** *subalg sets-restr-to-subalg* **by** *blast*
hence *A-in-sets-M*[*simp*]: *A* ∈ *sets M* **using** *sets-restr-to-subalg subalg subalgebra-def* **by** *blast*
have [*simp*]: *set-integrable M A* (λ*x*. *c*) **using** *A subalg* **by** (*auto simp add*: *set-integrable-def emeasure-restr-to-subalg*)
have [*simp*]: *set-integrable M A f* **unfolding** *set-integrable-def* **by** (*rule integrable-mult-indicator*, *auto simp add*: *assms*(*1*))
have *AE x in M*. *indicator A x* ∗*R c* = *indicator A x* ∗*R f x*
proof (*rule integral-eq-mono-AE-eq-AE*)
have (∫ *x*∈*A*. *c* ∂*M*) ≤ (∫ *x*∈*A*. *f x* ∂*M*) **using** *assms*(*2*) **by** (*intro set-integral-mono-AE-banach*) *auto*
moreover
{
have (∫ *x*∈*A*. *f x* ∂*M*) = (∫ *x*∈*A*. *cond-exp M F f x* ∂*M*) **by** (*rule cond-exp-set-integral*, *auto simp add*: *assms*)
also have ... ≤ (∫ *x*∈*A*. *c* ∂*M*) **using** *A* **by** (*auto intro*!: *set-integral-mono-banach simp add*: *X-def*)
finally have (∫ *x*∈*A*. *f x* ∂*M*) ≤ (∫ *x*∈*A*. *c* ∂*M*) **by** *simp*
}
ultimately show *LINT x*|*M*. *indicator A x* ∗*R c* = *LINT x*|*M*. *indicator A x* ∗*R f x* **unfolding** *set-lebesgue-integral-def* **by** *simp*
show *AE x in M*. *indicator A x* ∗*R c* ≤ *indicator A x* ∗*R f x* **using** *assms* **by** (*auto simp add*: *X-def indicator-def*)
qed (*auto simp add*: *set-integrable-def*[*symmetric*])
hence *AE x*∈*A in M*. *c* = *f x* **by** *auto*
hence *AE x*∈*A in M*. *False* **using** *assms*(*2*) **by** *auto*
hence *A* ∈ *null-sets M* **using** *AE-iff-null-sets A-in-sets-M* **by** *metis*
thus *False* **using** *A*(*3*) **by** (*simp add*: *emeasure-restr-to-subalg null-setsD1 subalg*)
qed
thus *?thesis* **using** *AE-iff-null-sets*[*OF X-in-M*] **unfolding** *X-def* **by** *auto*
qed

corollary *cond-exp-less-c*:
fixes *f* :: *'a* ⇒ *'b* :: {*second-countable-topology*, *banach*, *linorder-topology*, *ordered-real-vector*}
assumes *integrable M f AE x in M*. *f x* < *c*
shows *AE x in M*. *cond-exp M F f x* < *c*
proof −
have *AE x in M*. *cond-exp M F f x* = − *cond-exp M F* (λ*x*. − *f x*) *x* **using** *cond-exp-uminus*[*OF assms*(*1*)] **by** *auto*
moreover have *AE x in M*. *cond-exp M F* (λ*x*. − *f x*) *x* > − *c* **using** *assms* **by** (*intro cond-exp-gr-c*) *auto*

ultimately show *?thesis* **by** (*force simp add*: *minus-less-iff*)
qed

lemma *cond-exp-mono-strict*:
fixes $f :: 'a \Rightarrow 'b :: \{$*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 $f :: 'a \Rightarrow 'b :: \{$*second-countable-topology*, *banach*, *linorder-topology*, *ordered-real-vector*$\}$
assumes [*measurable*]: *integrable M f*
and *AE x in M*. $f\ x \geq c$
shows *AE x in M*. *cond-exp M F f x* $\geq c$
proof −
let *?F* = *restr-to-subalg M F*
interpret *sigma-finite-measure restr-to-subalg M F* **using** *sigma-fin-subalg* **by** *auto*
{
fix *A* **assume** *asm*: *A* $\in$ *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* $< \infty$ **using** *measure-zero-top asm* **by** (*force simp add*: *top.not-eq-extremum*)
hence *F-A*: *emeasure ?F A* $< \infty$ **using** *asm*(*1*) *emeasure-restr-to-subalg subalg* **by** *fastforce*
have *set-lebesgue-integral M A* (λ-. *c*) $\leq$ *set-lebesgue-integral M A f* **using** *assms asm M-A subalg* **by** (*intro set-integral-mono-AE-banach*, *auto simp add*: *set-integrable-def integrable-mult-indicator subalgebra-def sets-restr-to-subalg*)
also have ... = *set-lebesgue-integral M A* (*cond-exp M F f*) **using** *cond-exp-set-integral*[*OF assms*(*1*)] *asm* **by** *auto*
also have ... = *set-lebesgue-integral ?F A* (*cond-exp M F f*) **unfolding** *set-lebesgue-integral-def* **using** *asm borel-measurable-cond-exp* **by** (*intro integral-subalgebra2*[*OF subalg*, *symmetric*], *simp*)
finally have (*1 / measure ?F A*) $*_R$ *set-lebesgue-integral ?F A* (*cond-exp M F f*) $\in \{c..\}$ **using** *asm subalg M-A* **by** (*auto simp add*: *set-integral-const subalgebra-def intro*!: *pos-divideR-le-eq*[*THEN iffD1*])
}
thus *?thesis* **using** *AE-restr-to-subalg*[*OF subalg*] *averaging-theorem*[*OF integrable-in-subalg closed-atLeast*, *OF subalg borel-measurable-cond-exp integrable-cond-exp*] **by** *auto*
qed

corollary *cond-exp-le-c*:

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

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

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

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

(*cond-exp M F f ξ*) (*cond-exp M F g ξ*) **by** *fastforce*
qed

corollary *cond-exp-inf*:
fixes *f* :: *'a* ⇒ *'b* :: {*second-countable-topology*, *banach*, *linorder-topology*, *ordered-real-vector*, *lattice*}
assumes *integrable M f integrable M g*
shows *AE ξ in M*. *cond-exp M F* (λ*x*. *inf* (*f x*) (*g x*)) *ξ* ≤ *inf* (*cond-exp M F f ξ*) (*cond-exp M F g ξ*)
unfolding *inf-min* **using** *assms* **by** (*rule cond-exp-min*)

corollary *cond-exp-sup*:
fixes *f* :: *'a* ⇒ *'b* :: {*second-countable-topology*, *banach*, *linorder-topology*, *ordered-real-vector*, *lattice*}
assumes *integrable M f integrable M g*
shows *AE ξ in M*. *cond-exp M F* (λ*x*. *sup* (*f x*) (*g x*)) *ξ* ≥ *sup* (*cond-exp M F f ξ*) (*cond-exp M F g ξ*)
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* ⇒ *'b* :: {*second-countable-topology*, *banach*}
assumes *integrable M f*
shows *AE x in M*. *cond-exp M* (*sigma* (*space M*) {}) *f x* = *expectation f*
proof −
interpret *sigma-finite-subalgebra M sigma* (*space M*) {} **by** (*auto intro*: *sigma-finite-subalgebra-restr-to-subalg simp add*: *subalgebra-def sigma-sets-empty-eq*)
show *?thesis* **using** *assms* **by** (*intro cond-exp-charact*) (*auto simp add*: *sigma-sets-empty-eq set-lebesgue-integral-def prob-space cong*: *Bochner-Integration.integral-cong*)
qed

The following lemma shows that independent 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* ⇒ *'b* :: {*second-countable-topology*, *banach*, *real-normed-field*}
assumes *subalgebra*: *subalgebra M F subalgebra M G*
and *independent*: *indep-set G* (*sigma* (*space M*) (*F* ∪ *vimage-algebra* (*space*

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

hence *events*[*measurable*]: *a* ∈ *events b* ∈ *events* **using** *subalgebra* **by** (*auto simp add*: *subalgebra-def*)

have [*simp*]: *sigma-sets* (*space M*) {*indicator b* −‘ *A* ∩ *space M* |*A*. *A* ∈ *borel*} ⊆ *G*
using *borel-measurable-indicator*[*OF A*(*3*), *THEN measurable-sets*] *sets.top subalgebra*
by (*intro sets.sigma-sets-subset*′) (*fastforce simp add*: *subalgebra-def*)+

have *Un-in-sigma*: *F* ∪ *vimage-algebra* (*space M*) *f borel* ⊆ *sigma* (*space M*) (*F* ∪ *vimage-algebra* (*space M*) *f borel*) **by** (*metis equalityE le-supI sets.space-closed sigma-le-sets space-vimage-algebra subalg subalgebra-def*)

have [*intro*]: *indep-var borel* (*indicator b*) *borel* (*λω. indicator a ω ∗R f ω*)
proof −
have [*simp*]: *sigma-sets* (*space M*) {(*λω. indicator a ω ∗R f ω*) −‘ *A* ∩ *space M* |*A*. *A* ∈ *borel*} ⊆ *sigma* (*space M*) (*F* ∪ *vimage-algebra* (*space M*) *f borel*)
proof −
have ∗: (*λω. indicator a ω ∗R f ω*) ∈ *borel-measurable* (*sigma* (*space M*) (*F* ∪ *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* ∩ *space M* |*A*. *A* ∈ *borel*}) (*sigma-sets* (*space M*) {(*λω. indicator a ω ∗R f ω*) −‘ *A* ∩ *space M* |*A*. *A* ∈ *borel*})
using *independent* **unfolding** *indep-set-def* **by** (*rule indep-sets-mono-sets*, *auto split*: *bool.split*)
thus *?thesis* **by** (*subst indep-var-eq*, *auto intro*!: *borel-measurable-scaleR*)
qed

have [*intro*]: *indep-var borel* (*indicator b*) *borel* (*λω. indicat-real a ω ∗R cond-exp M F f ω*)
proof −
have [*simp*]:*sigma-sets* (*space M*) {(*λω. indicator a ω ∗R cond-exp M F f ω*) −‘ *A* ∩ *space M* |*A*. *A* ∈ *borel*} ⊆ *sigma* (*space M*) (*F* ∪ *vimage-algebra* (*space M*) *f borel*)
proof −
have ∗: (*λω. indicator a ω ∗R cond-exp M F f ω*) ∈ *borel-measurable* (*sigma* (*space M*) (*F* ∪ *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* ∩ *space M* |*A*. *A* ∈ *borel*}) (*sigma-sets* (*space M*) {(*λω. indicator a ω ∗R cond-exp M F f ω*) −‘ *A* ∩ *space M* |*A*. *A* ∈ *borel*})
using *independent* **unfolding** *indep-set-def* **by** (*rule indep-sets-mono-sets*, *auto split*: *bool.split*)
thus *?thesis* **by** (*subst indep-var-eq*, *auto intro*!: *borel-measurable-scaleR*)

qed

have *set-lebesgue-integral M A f* = (*LINT x*|*M*. *indicator b x* ∗ (*indicator a x* ∗$_R$ *f x*))
unfolding *set-lebesgue-integral-def A indicator-inter-arith*
by (*intro Bochner-Integration.integral-cong*, *auto simp add*: *scaleR-scaleR*[*symmetric*] *indicator-times-eq-if*(*1*))
also have ... = (*LINT x*|*M*. *indicator b x*) ∗ (*LINT x*|*M*. *indicator a x* ∗$_R$ *f x*)
by (*intro indep-var-lebesgue-integral Bochner-Integration.integrable-bound*[*OF integrable-const*[*of 1* :: *'b*] *borel-measurable-indicator*] *integrable-mult-indicator*[*OF - assms*(*4*)], *blast*) (*auto simp add*: *indicator-def*)
also have ... = (*LINT x*|*M*. *indicator b x*) ∗ (*LINT x*|*M*. *indicator a x* ∗$_R$ *cond-exp M F f x*)
using *cond-exp-set-integral*[*OF assms*(*4*) *A*(*2*)] **unfolding** *set-lebesgue-integral-def* **by** *argo*
also have ... = (*LINT x*|*M*. *indicator b x* ∗ (*indicator a x* ∗$_R$ *cond-exp M F f x*))
by (*intro indep-var-lebesgue-integral*[*symmetric*] *Bochner-Integration.integrable-bound*[*OF integrable-const*[*of 1* :: *'b*] *borel-measurable-indicator*] *integrable-mult-indicator*[*OF - integrable-cond-exp*], *blast*) (*auto simp add*: *indicator-def*)
also have ... = *set-lebesgue-integral M A* (*cond-exp M F f*)
unfolding *set-lebesgue-integral-def A indicator-inter-arith*
by (*intro Bochner-Integration.integral-cong*, *auto simp add*: *scaleR-scaleR*[*symmetric*] *indicator-times-eq-if*(*1*))
finally show *?case* **.**
next
case *empty*
then show *?case* **unfolding** *set-lebesgue-integral-def* **by** *simp*
next
case (*compl A*)
have *A-in-space*: *A* ⊆ *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* ∪ *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* ∪ *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* (⋃ (*range A*)) *f* = (∑ *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 ... = (∑ *i*. *set-lebesgue-integral M* (*A i*) (*cond-exp M F f*)) **using** *union* **by** *presburger*
also have ... = *set-lebesgue-integral M* (⋃ (*range A*)) (*cond-exp M F f*)
using *union in-events*
by (*intro lebesgue-integral-countable-add*[*symmetric*]) (*auto simp add*: *disjoint-family-onD intro*!: *integrable-mult-indicator*[*folded set-integrable-def*, *OF - integrable-cond-exp*])
finally show *?case* **.**
qed
}
moreover have *sigma* (*space M*) {*a* ∩ *b* | *a b*. *a* ∈ *F* ∧ *b* ∈ *G*} = *sigma* (*space M*) (*F* ∪ *G*)
proof −
have *sigma-sets* (*space M*) {*a* ∩ *b* |*a b*. *a* ∈ *sets F* ∧ *b* ∈ *sets G*} = *sigma-sets* (*space M*) (*sets F* ∪ *sets G*)
proof −
{
fix *a b* **assume** *asm*: *a* ∈ *F b* ∈ *G*
hence *a* ∩ *b* ∈ *sigma-sets* (*space M*) (*F* ∪ *G*) **using** *subalgebra* **unfolding** *Int-range-binary* **by** (*intro sigma-sets-Inter*[*OF - binary-in-sigma-sets*]) (*force simp add*: *subalgebra-def dest*: *sets.sets-into-space*)+
}
moreover
{
fix *a*
assume *a* ∈ *sets F*
hence *a* ∈ *sigma-sets* (*space M*) {*a* ∩ *b* |*a b*. *a* ∈ *sets F* ∧ *b* ∈ *sets G*}
using *subalgebra sets.top*[*of G*] *sets.sets-into-space*[*of - F*]
by (*intro sigma-sets.Basic*, *auto simp add*: *subalgebra-def*)
}
moreover
{
fix *a* **assume** *a* ∈ *sets F* ∨ *a* ∈ *sets G a* ∉ *sets F*
hence *a* ∈ *sets G* **by** *blast*
hence *a* ∈ *sigma-sets* (*space M*) {*a* ∩ *b* |*a b*. *a* ∈ *sets F* ∧ *b* ∈ *sets G*}
using *subalgebra sets.top*[*of F*] *sets.sets-into-space*[*of - G*]
by (*intro sigma-sets.Basic*, *auto simp add*: *subalgebra-def*)
}

ultimately show *?thesis* **by** (*intro sigma-sets-eqI*) *auto*
qed
thus *?thesis* **using** *subalgebra* **by** (*intro sigma-eqI*) (*force simp add*: *subalgebra-def dest*: *sets.sets-into-space*)+
qed
moreover have (*cond-exp M F f*) ∈ *borel-measurable* (*sigma* (*space M*) (*sets F* ∪ *sets G*))
proof −
have *F* ⊆ *sigma* (*space M*) (*F* ∪ *G*) **by** (*metis Un-least Un-upper1 measure-of-of-measure sets.space-closed sets-measure-of sigma-sets-subseteq subalg subalgebra*(*2*) *subalgebra-def*)
thus *?thesis* **using** *borel-measurable-cond-exp*[*THEN measurable-sets*, *OF borel-open*, *of - M F f*] *subalgebra* **by** (*intro borel-measurableI*, *force simp only*: *space-measure-of-conv subalgebra-def*)
qed
ultimately show *?thesis* **using** *assms*(*4*) *integrable-cond-exp* **by** (*intro Un-sigma.cond-exp-charact*) *presburger*+
qed

If a random variable is independent of a σ-algebra F, its conditional expectation *cond-exp M F f* is just its expectation.

lemma (**in** *prob-space*) *cond-exp-indep*:
fixes *f* :: *'a* ⇒ *'b* :: {*second-countable-topology*, *banach*, *real-normed-field*}
assumes *subalgebra*: *subalgebra M F*
and *independent*: *indep-set F* (*vimage-algebra* (*space M*) *f borel*)
and *integrable*: *integrable M f*
shows *AE x in M*. *cond-exp M F f x* = *expectation f*
proof −
have *indep-set F* (*sigma* (*space M*) (*sigma* (*space M*) {} ∪ (*vimage-algebra* (*space M*) *f borel*)))
using *independent* **unfolding** *indep-set-def*
by (*rule indep-sets-mono-sets*, *simp add*: *bool.split*)
(*metis bot.extremum dual-order.refl sets.sets-measure-of-eq sets.sigma-sets-subset' sets-vimage-algebra-space space-vimage-algebra sup.absorb-iff2*)
hence *cond-exp-indep*: *AE x in M*. *cond-exp M* (*sigma* (*space M*) (*sigma* (*space M*) {} ∪ *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*) {} ∪ *F*)) = *F*
using *subalgebra sets.top*[*of F*] **unfolding** *subalgebra-def*
by (*simp add*: *sigma-sets-empty-eq*, *subst insert-absorb*[*of space M F*], *blast*)
(*metis insert-absorb*[*OF sets.empty-sets*] *sets.sets-measure-of-eq*)
hence *AE x in M*. *cond-exp M* (*sigma* (*space M*) (*sigma* (*space M*) {} ∪ *F*)) *f x* = *cond-exp M F f x* **by** (*rule cond-exp-sets-cong*)
thus *?thesis* **using** *cond-exp-indep* **by** *force*
qed

end

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

3 Filtered Measure Spaces

3.1 Filtered Measure

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

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

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

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

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

end

locale *linearly-filtered-measure* = *filtered-measure* M F t_0 **for** M **and** F :: - :: {*linorder-topology*, *conditionally-complete-lattice*} ⇒ - **and** t_0

context *linearly-filtered-measure*
begin

$\sigma-$*algebra* at infinity

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

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

lemma *space-F-infinity*[*simp*]: *space* F_∞ = *space M* **unfolding** *F-infinity-def space-measure-of-conv* **by** *simp*

lemma *sets-F-infinity*: *sets* F_∞ = *sigma-sets* (*space M*) ($\bigcup t{\in}\{t_0..\}$. *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_∞ **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* $\in$ *borel-measurable* (*F t*)
shows *f* $\in$ *borel-measurable* (F_∞)
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* :: *nat* ⇒ -
locale *enat-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *enat* ⇒ -
locale *real-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *real* ⇒ -
locale *ennreal-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *ennreal* ⇒ -

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* ⊆ *nat-filtered-measure* **..**
sublocale *enat-sigma-finite-filtered-measure* ⊆ *enat-filtered-measure* **..**
sublocale *real-sigma-finite-filtered-measure* ⊆ *real-filtered-measure* **..**
sublocale *ennreal-sigma-finite-filtered-measure* ⊆ *ennreal-filtered-measure* **..**

sublocale *nat-sigma-finite-filtered-measure* ⊆ *sigma-finite-subalgebra M F i* **by** *blast*
sublocale *enat-sigma-finite-filtered-measure* ⊆ *sigma-finite-subalgebra M F i* **by** *fastforce*
sublocale *real-sigma-finite-filtered-measure* ⊆ *sigma-finite-subalgebra M F* |*i*| **by** *fastforce*
sublocale *ennreal-sigma-finite-filtered-measure* ⊆ *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* ⊆ *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* ⊆ *nat-sigma-finite-filtered-measure* **..**
sublocale *enat-finite-filtered-measure* ⊆ *enat-sigma-finite-filtered-measure* **..**
sublocale *real-finite-filtered-measure* ⊆ *real-sigma-finite-filtered-measure* **..**
sublocale *ennreal-finite-filtered-measure* ⊆ *ennreal-sigma-finite-filtered-measure* **..**

3.4 Constant Filtration

lemma *filtered-measure-constant-filtration*:
assumes *subalgebra M F*
shows *filtered-measure M* (λ-. *F*) t_0
using *assms* **by** (*unfold-locales*) *blast*+

sublocale *sigma-finite-subalgebra* ⊆ *constant-filtration*: *sigma-finite-filtered-measure M* λ- :: *'t* :: {*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* (λ-. *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*} ⇒ *'a* ⇒ *'c* :: {*second-countable-topology*, *banach*}
assumes *random-variable*[*measurable*]: ⋀*i*. $t_0 \leq i \Longrightarrow X\ i \in$ *borel-measurable M*
begin

definition *left-continuous* **where** *left-continuous* = (*AE ξ in M*. ∀ *t*. *continuous* (*at-left t*) (λ*i*. *X i ξ*))
definition *right-continuous* **where** *right-continuous* = (*AE ξ in M*. ∀ *t*. *continuous* (*at-right t*) (λ*i*. *X i ξ*))

end

lemma *stochastic-process-const-fun*:
assumes *f* ∈ *borel-measurable M*
shows *stochastic-process M* t_0 (λ-. *f*) **using** *assms* **by** (*unfold-locales*)

lemma *stochastic-process-const*:
shows *stochastic-process M* t_0 (λ*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 ⋀*i*. $t_0 \leq i \Longrightarrow f\ i \in$ *borel-measurable borel*
shows *stochastic-process M* t_0 (λ*i ξ*. (*f i*) (*X i ξ*))
by (*unfold-locales*) (*intro measurable-compose*[*OF random-variable assms*])

lemma *norm-stochastic*: *stochastic-process M* t_0 (λ*i ξ*. *norm* (*X i ξ*)) **by** (*fastforce intro*: *compose-stochastic*)

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

lemma *scaleR-right-const-fun-stochastic*:
assumes $f \in$ *borel-measurable M*
shows *stochastic-process M* t_0 $(\lambda i\ \xi.\ f\ \xi\ *_R\ (X\ i\ \xi))$
by (*unfold-locales*) (*intro borel-measurable-scaleR assms random-variable*)

lemma *scaleR-right-const-stochastic*: *stochastic-process M* t_0 $(\lambda i\ \xi.\ c\ i\ *_R\ (X\ i\ \xi))$
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* λ*-. −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 $\bigwedge i.\ i \in I \Longrightarrow$ *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 measure* ⇒ *′b* ⇒ (*′b* ⇒ *′a* ⇒ *′c* :: *topologi-*

cal-space) ⇒ *'b* :: {*second-countable-topology*, *order-topology*} ⇒ *'a measure* **where**
natural-filtration M t_0 *Y* = (λ*t*. *family-vimage-algebra* (*space M*) {*Y i* | *i*. *i* ∈ {t_0..*t*}} *borel*)

abbreviation *nat-natural-filtration* ≡ λ*M*. *natural-filtration M* (*0* :: *nat*)
abbreviation *real-natural-filtration* ≡ λ*M*. *natural-filtration M* (*0* :: *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*) (⋃*i*∈{t_0..*t*}. {*X i* −‘ *A* ∩ *space M* | *A*. *A* ∈ *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*) (⋃*i*∈{t_0..*t*}. {*X i* −‘ *A* ∩ *space M* | *A*. *A* ∈ *S*})
proof (*subst sets-natural-filtration*, *intro sigma-sets-eqI*, *clarify*)
fix *i* **and** *A* :: *'a set* **assume** *asm*: *i* ∈ {t_0..*t*} *A* ∈ *sets borel*
hence *A* ∈ *sigma-sets UNIV S* **unfolding** *assms* **by** *simp*
thus *X i* −‘ *A* ∩ *space M* ∈ *sigma-sets* (*space M*) (⋃*i*∈{t_0..*t*}. {*X i* −‘ *A* ∩ *space M* |*A*. *A* ∈ *S*})
proof (*induction*)
case (*Compl a*)
have *X i* −‘ (*UNIV* − *a*) ∩ *space M* = *space M* − (*X i* −‘ *a* ∩ *space M*) **by** *blast*
then show *?case* **using** *Compl*(*2*)[*THEN sigma-sets.Compl*] **by** *presburger*
next
case (*Union a*)
have *X i* −‘ ⋃ (*range a*) ∩ *space M* = ⋃ (*range* (λ*j*. *X i* −‘ *a j* ∩ *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*) (⋃*i*∈{t_0..*t*}. {*X i* −‘ *A* ∩ *space M* | *A*. *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*) (⋃*i*∈{t_0..*t*}. {*X i* −‘ *A* ∩ *space M* | *A* :: - :: {*linorder-topology*, *second-countable-topology*} *set*. *A* ∈ *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* ∩ *space M* | *A* :: - :: {*linorder-topology*, *second-countable-topology*} *set*. *A* ∈ *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* ∩ *space M* | *A* :: *real set*. *A* ∈ *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* $\bigwedge a$. *a* ∈ *A* ⟹ *emeasure M a* ≠ ∞ $\bigwedge a$. *a* ∈ *A* ⟹ ∃ *b* ∈ *borel*. *a* = *X* t_0 −‘ *b* ∩ *space M*
shows *sigma-finite-filtered-measure M* (*natural-filtration M* t_0 *X*) t_0
proof (*unfold-locales*)
have *A* ⊆ *sets* (*restr-to-subalg M* (*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* = *space* (*restr-to-subalg M* (*natural-filtration M* t_0 *X* t_0)) **unfolding** *space-restr-to-subalg* **using** *exhausting-set* **by** *simp*
moreover have ∀ *a*∈*A*. *emeasure* (*restr-to-subalg M* (*natural-filtration M* t_0 *X* t_0)) *a* ≠ ∞ **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 ∃ *A*. *countable A* ∧ *A* ⊆ *sets* (*restr-to-subalg M* (*natural-filtration M* t_0 *X* t_0)) ∧ $\bigcup$ *A* = *space* (*restr-to-subalg M* (*natural-filtration M* t_0 *X* t_0)) ∧ (∀ *a*∈*A*. *emeasure* (*restr-to-subalg M* (*natural-filtration M* t_0 *X* t_0)) *a* ≠ ∞) **using** *exhausting-set* **by** *blast*
show $\bigwedge i\ j$. ⟦$t_0 \le i$; $i \le j$⟧ ⟹ *sets* (*natural-filtration M* t_0 *X i*) ⊆ *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 $t_0 \leq s$ $s < t$ *indep-vars* (λ-. *borel*) *X* {t_0..}
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* ∩ *space M* |*A*. *A* ∈ *sets borel*}) (⋃(*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* ∩ *space M* | *A*. *A* ∈ *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* :: - ⇒ - ⇒ - :: {*second-countable-topology*, *banach*} +
assumes *adapted*[*measurable*]: ⋀*i*. $t_0 \leq i \Longrightarrow X\ i \in$ *borel-measurable* (*F i*)
begin

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

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

end

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

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

Again, we cover basic operations.

context *adapted-process*
begin

lemma *compose-adapted*:
assumes $\bigwedge i.\ t_0 \leq i \Longrightarrow f\ i \in$ *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* $M\ F\ t_0\ (\lambda i\ \xi.\ norm\ (X\ i\ \xi))$ **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 $f \in$ *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* $\lambda\text{-}.\ -1$] **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 $X\ j \in$ *borel-measurable* $(F\ i)$ **if** $t_0 \leq j\ j \leq i$ **for** j **using** *that adaptedE* **by** *meson*
thus $(\lambda\xi.\ \sum i{\in}\{t_0..i\}.\ X\ i\ \xi) \in$ *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 $X\ j \in$ *borel-measurable* $(F\ i)$ **if** $t_0 \leq j\ j < i$ **for** j **using** *that adaptedE* **by** *fastforce*
thus $(\lambda\xi.\ \sum i{\in}\{t_0..{<}i\}.\ X\ i\ \xi) \in$ *borel-measurable* $(F\ i)$ **by** *simp*
qed

end

In the dicrete 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 $X\ j \in$ *borel-measurable* $(F\ i)$ **if** $j \leq i$ **for** j **using** *that adaptedD* **by** *blast*
thus $(\lambda\xi.\ \sum i{<}i.\ X\ (Suc\ i)\ \xi) \in$ *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 $X\ (eSuc\ j) \in$ *borel-measurable* $(F\ i)$ **if** $j < i$ **for** j **using** *that adaptedD* **by** (*simp add*: *ileI1*)
thus $(\lambda\xi.\ \sum i{<}i.\ X\ (eSuc\ i)\ \xi) \in$ *borel-measurable* $(F\ i)$ **by** *auto*
qed

lemma (**in** *filtered-measure*) *adapted-process-sum*:
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *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 $X\ i\ k \in$ *borel-measurable* $M\ X\ i\ k \in$ *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* t_0 *X*) t_0 *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* t_0 **for** *M F* t_0 **and** *X* :: - $\Rightarrow$ - $\Rightarrow$ - :: {*second-countable-topology*, *banach*} +
assumes *progressive*[*measurable*]: $\bigwedge t.\ t_0 \leq t \Longrightarrow (\lambda(i, x).\ X\ i\ x) \in$ *borel-measurable* (*restrict-space borel* {t_0..*t*} $\bigotimes_M$ *F t*)
begin

lemma *progressiveD*:
assumes $S \in$ *borel*
shows $(\lambda(j, \xi).\ X\ j\ \xi) -` S \cap (\{t_0..i\} \times$ *space M*$) \in$ (*restrict-space borel* {t_0..*i*} $\bigotimes_M$ *F i*)
using *measurable-sets*[*OF progressive*, *OF - assms*, *of i*]
by (*cases* $t_0 \leq i$) (*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* t_0)
shows *progressive-process M F* t_0 (λ-. *f*)
proof (*unfold-locales*)
fix *i* **assume** *asm*: $t_0 \leq i$
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* {t_0..*i*} $\bigotimes_M$ *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* t_0 (λi -. *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* ∈ *borel-measurable borel*
shows *progressive-process M F* t_0 (λ*i* ξ. (*f i*) (*X i* ξ))
proof
fix *i* **assume** *asm*: $t_0 \leq i$
have (λ(*j*, ξ). (*j*, *X j* ξ)) ∈ (*restrict-space borel* {t_0..*i*} ⨂$_M$ *F i*) →$_M$ *borel* ⨂$_M$ *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 (λ(*j*, ξ). *f j* (*X j* ξ)) = *case-prod f o* ((λ(*j*, *y*). (*j*, *y*)) *o* (λ(*j*, ξ). (*j*, *X j* ξ))) **by** *fastforce*
ultimately show (λ(*j*, ξ). (*f j*) (*X j* ξ)) ∈ *borel-measurable* (*restrict-space borel* {t_0..*i*} ⨂$_M$ *F i*) **using** *assms* **by** (*simp add*: *borel-prod*)
qed

lemma *norm-progressive*: *progressive-process M F* t_0 (λ*i* ξ. *norm* (*X i* ξ)) **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 (λ*i* ξ. (*R i* ξ) ∗$_R$ (*X i* ξ))
using *progressive-process.progressive*[*OF assms*] **by** (*unfold-locales*) (*simp add*: *progressive assms*)

lemma *scaleR-right-const-fun-progressive*:
assumes *f* ∈ *borel-measurable* (*F* t_0)
shows *progressive-process M F* t_0 (λ*i* ξ. *f* ξ ∗$_R$ (*X i* ξ))
using *assms* **by** (*fast intro*: *scaleR-right-progressive progressive-process-const-fun*)

lemma *scaleR-right-const-progressive*:
assumes *c* ∈ *borel-measurable borel*
shows *progressive-process M F* t_0 (λ*i* ξ. *c i* ∗$_R$ (*X i* ξ))
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 (λ*i* ξ. *X i* ξ + *Y i* ξ)
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 (λ*i* ξ. *X i* ξ − *Y i* ξ)
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* λ*-. −1*] **by** (*simp add*: *fun-Compl-def*)

end

A progressively measurable process is also adapted.

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

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 set* **and** *i* :: *nat* **assume** *open-S*: *open S*
{
fix *j* **assume** *asm*: $j \leq i$
hence *X j −‘ S ∩ space M ∈ F i* **using** *adaptedD*[*of j*, *THEN measurable-sets*] *space-F open-S* **by** *fastforce*
moreover have *case-prod X −‘ S ∩ {j} × space M = {j} × (X j −‘ S ∩ space M)* **for** *j* **by** *fast*
moreover have $\{j :: nat\} \in$ *restrict-space borel* $\{0..i\}$ **using** *asm* **by** (*simp add*: *sets-restrict-space-iff*)
ultimately have *case-prod X −‘ S ∩ {j} × space M ∈ restrict-space borel* $\{0..i\} \bigotimes_M$ *F i* **by** *simp*
}
hence $(\lambda j.\ (\lambda(x,\ y).\ X\ x\ y) -‘ S \cap \{j\} \times space\ M)\ ‘ \{..i\} \subseteq$ *restrict-space borel* $\{0..i\} \bigotimes_M$ *F i* **by** *blast*
moreover have *case-prod X −‘ S ∩ space* (*restrict-space borel* $\{0..i\} \bigotimes_M$ *F i*) $= (\bigcup j \leq i.$ *case-prod X −‘ S ∩ {j} × space M*) **unfolding** *space-pair-measure space-restrict-space space-F* **by** *force*
ultimately show *case-prod X −‘ S ∩ space* (*restrict-space borel* $\{0..i\} \bigotimes_M$ *F i*) $\in$ *restrict-space borel* $\{0..i\} \bigotimes_M$ *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*: $j \leq i$
hence *X j −‘ S ∩ space M ∈ F i* **using** *adaptedD*[*of j*, *THEN measurable-sets*] *space-F open-S* **by** *fastforce*
moreover have *case-prod X −‘ S ∩ {j} × space M = {j} × (X j −‘ S ∩ space M)* **for** *j* **by** *fast*
moreover have *{j :: enat} ∈ restrict-space borel {0..i}* **using** *asm* **by** (*simp add*: *sets-restrict-space-iff*)
ultimately have *case-prod X −‘ S ∩ {j} × space M ∈ restrict-space borel {0..i} ⨂M F i* **by** *simp*
}
hence *(λj. (λ(x, y). X x y) −‘ S ∩ {j} × space M) ‘ {..i} ⊆ restrict-space borel {0..i} ⨂M F i* **by** *blast*
moreover have *case-prod X −‘ S ∩ space (restrict-space borel {0..i} ⨂M F i) = (⋃j≤i. case-prod X −‘ S ∩ {j} × space M)* **unfolding** *space-pair-measure space-restrict-space space-F* **by** *force*
ultimately show *case-prod X −‘ S ∩ space (restrict-space borel {0..i} ⨂M F i) ∈ restrict-space borel {0..i} ⨂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 Σ_P :: *('b × 'a) measure* **where** *predictable-sigma*: $\Sigma_P \equiv$ *sigma* ({t_0..} × *space M*) ({{*s*<..*t*} × *A* | *A s t*. *A ∈ F s ∧* $t_0 \leq s \wedge s < t$} ∪ {{t_0} × *A* | *A*. *A ∈ F* t_0})

lemma *space-predictable-sigma*[*simp*]: *space* Σ_P = ({t_0..} × *space M*) **unfolding** *predictable-sigma space-measure-of-conv* **by** *blast*

lemma *sets-predictable-sigma*: *sets* Σ_P = *sigma-sets* ({t_0..} × *space M*) ({{*s*<..*t*} × *A* | *A s t*. *A ∈ F s ∧* $t_0 \leq s \wedge s < t$} ∪ {{t_0} × *A* | *A*. *A ∈ 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* Σ_P **if** $A \in$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\} \mid s\ t.\ t_0 \leq s \wedge s < t\}$ **for** A **unfolding** *sets-predictable-sigma* **using** *that*
proof (*induction rule*: *sigma-sets.induct*)
case (*Basic a*)
thus *?case* **using** *space-F sets.top* **by** *blast*
next
case (*Compl a*)
have $(\{t_0..\} - a) \times$ *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-sets UNIV* (*range greaterThan*)) = *sigma-sets* $\{t_0..\}$ $\{\{s<..t\} |s\ t.\ t_0 \leq s \wedge s < t\}$
proof (*intro sigma-sets-eqI* ; *clarify*)
fix $A :: 'b\ set$ **assume** *asm*: $A \in$ *sigma-sets UNIV* (*range greaterThan*)
thus $\{t_0..\} \cap A \in$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\} |s\ t.\ t_0 \leq s \wedge s < t\}$
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 $((\cap)\ \{s<..\}\ `\ \mathcal{I}) \subseteq$ *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: $A = \{s'<..t'\}$ $t_0 \leq s'\ s' < t'$ **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 $\{s<..\} \cap A \in$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\}\ |s\ t.\ t_0 \leq s \wedge s < t\}$ **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 $\{t_0..\} \cap (UNIV - a) = \{t_0..\} - (\{t_0..\} \cap a)$ **by** *blast*
then show *?case* **using** *Compl(2)*[*THEN sigma-sets.Compl*] **by** *presburger*
next
case (*Union a*)
have $\{t_0..\} \cap \bigcup$ (*range a*) $= \bigcup$ (*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 $\{s<..\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)\ \{t_0..\}\ `$ *sigma-sets UNIV* (*range greaterThan*)) **using** *asm* **by** (*intro sigma-sets.Basic*) *auto*
moreover have $\{t_0..\} - \{t<..\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)\ \{t_0..\}\ `$ *sigma-sets UNIV* (*range greaterThan*)) **using** *asm* **by** (*intro sigma-sets.Compl sigma-sets.Basic*) *auto*
ultimately show $\{s<..t\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)\ \{t_0..\}\ `$ *sigma-sets UNIV* (*range greaterThan*)) **unfolding** $*$ *Int-range-binary*[*of* $\{s<..\}$] **by** (*intro sigma-sets-Inter*[*OF - binary-in-sigma-sets*]) *auto*
qed
thus *?thesis* **unfolding** *borel-Ioi restrict-space-def emeasure-sigma* **by** (*force intro*: *sigma-eqI*)
qed
ultimately have *restrict-space borel* $\{t_0..\} \bigotimes_M$ *sigma* (*space M*) $\{\} \subseteq$ *sets* Σ_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* $\{t_0..\} \bigotimes_M$ *sigma* (*space M*) $\{\}$) = *space* Σ_P **by** (*simp add*: *space-pair-measure*)
moreover have *fst* $\in$ *restrict-space borel* $\{t_0..\} \bigotimes_M$ *sigma* (*space M*) $\{\} \to_M$ *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* :: - ⇒ - ⇒ - :: {*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* $\in \Sigma_P \to_M$ *F 0*
by (*intro measurable-predictable-sigma-snd*[*of range* $(\lambda x.\ \{Suc\ x\})$]) (*force* | *simp add*: *greaterThan-0*)+

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

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

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

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

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

lemma (**in** *ennreal-filtered-measure*) *measurable-predictable-sigma-snd'*:
shows *snd* ∈ Σ_P →$_M$ *F 0*
by (*intro measurable-predictable-sigma-snd*[*of* {{*0*<..∞}}]) *force*+

lemma (**in** *ennreal-filtered-measure*) *measurable-predictable-sigma-fst'*:
shows *fst* ∈ Σ_P →$_M$ *borel*
by (*intro measurable-predictable-sigma-fst*[*of* {{*0*<..∞}}]) *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* ∈ Σ_P →$_M$ *F* t_0 *f* ∈ *borel-measurable* (*F* t_0)
shows *predictable-process M F* t_0 (λ-. *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* ∈ *borel-measurable* (*F 0*)
shows *predictable-process M F 0* (λ-. *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* ∈ *borel-measurable* (*F 0*)
shows *predictable-process M F 0* (λ-. *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* ∈ *borel-measurable* (*F 0*)
shows *predictable-process M F 0* (λ-. *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* ∈ *borel-measurable* (*F 0*)
shows *predictable-process M F 0* (λ-. *f*)
using *assms* **by** (*intro predictable-process-const-fun*[*OF measurable-predictable-sigma-snd'*])

lemma (**in** *linearly-filtered-measure*) *predictable-process-const*:
assumes *fst* ∈ *borel-measurable* Σ_P *c* ∈ *borel-measurable borel*
shows *predictable-process M F* t_0 (λ*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 (λ- -. *c*)
by (*unfold-locales*) *simp*

lemma (**in** *nat-filtered-measure*) *predictable-process-const'*[*intro*]:
assumes *c* ∈ *borel-measurable borel*

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

context *predictable-process*
begin

lemma *compose-predictable*:
assumes *fst* ∈ *borel-measurable* Σ_P *case-prod f* ∈ *borel-measurable borel*
shows *predictable-process M F* t_0 ($\lambda i\ \xi$. (*f i*) (*X i* ξ))
proof
have ($\lambda(i, \xi)$. (*i*, *X i* ξ)) ∈ Σ_P →$_M$ *borel* ⨂$_M$ *borel* **using** *predictable assms*(*1*)
by (*auto simp add*: *measurable-pair-iff measurable-split-conv*)
moreover have ($\lambda(i, \xi)$. *f i* (*X i* ξ)) = *case-prod f o* ($\lambda(i, \xi)$. (*i*, *X i* ξ)) **by** *fastforce*
ultimately show ($\lambda(i, \xi)$. *f i* (*X i* ξ)) ∈ *borel-measurable* Σ_P **unfolding** *borel-prod*
using *assms* **by** *simp*
qed

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

lemma *scaleR-right-const-fun-predictable*:
assumes *snd* ∈ Σ_P →$_M$ *F* t_0 *f* ∈ *borel-measurable* (*F* t_0)
shows *predictable-process M F* t_0 ($\lambda i\ \xi$. *f* ξ ∗$_R$ (*X i* ξ))
using *assms* **by** (*fast intro*: *scaleR-right-predictable predictable-process-const-fun*)

lemma *scaleR-right-const-predictable*:

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

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

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

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

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

end

Every predictable process is also progressively measurable.

sublocale *predictable-process* ⊆ *progressive-process*
proof (*unfold-locales*)
fix *i* :: *'b* **assume** *asm*: $t_0 \le i$
{
fix *S* :: (*'b* × *'a*) *set* **assume** $S \in \{\{s<..t\} \times A \mid A\ s\ t.\ A \in F\ s \wedge t_0 \le s \wedge s < t\} \cup \{\{t_0\} \times A \mid A.\ A \in F\ t_0\}$
hence ($\lambda x.\ x$) −‘ *S* ∩ ({t_0..*i*} × *space M*) ∈ *restrict-space borel* {t_0..*i*} ⨂$_M$ *F i*
proof
assume $S \in \{\{s<..t\} \times A \mid A\ s\ t.\ A \in F\ s \wedge t_0 \le s \wedge s < t\}$
then obtain *s t A* **where** *S-is*: $S = \{s<..t\} \times A\ t_0 \le s\ s < t\ A \in F\ s$ **by** *blast*
hence ($\lambda x.\ x$) −‘ *S* ∩ ({t_0..*i*} × *space M*) = {*s*<..*min i t*} × *A* **using** *sets.sets-into-space*[*OF S-is*(*4*)] **by** *auto*
then show *?thesis* **using** *S-is sets-F-mono*[*of s i*] **by** (*cases* $s \le i$) (*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-is*: $S = \{t_0\} \times A\ A \in F\ t_0$ **by** *blast*
hence ($\lambda x.\ x$) −‘ *S* ∩ ({t_0..*i*} × *space M*) = {t_0} × *A* **using** *asm sets.sets-into-space*[*OF S-is*(*2*)] **by** *auto*
thus *?thesis* **using** *S-is*(*2*) *sets-F-mono*[*OF order-refl asm*] *asm* **by** (*fastforce simp add*: *sets-restrict-space-iff*)

qed
hence $(\lambda x.\ x) -` S \cap space\ (restrict\text{-}space\ borel\ \{t_0..i\} \bigotimes_M F\ i) \in restrict\text{-}space\ borel\ \{t_0..i\} \bigotimes_M F\ i$ **by** (*simp add*: *space-pair-measure space-F*[*OF asm*])
}
moreover have $\{\{s<..t\} \times A\ |A\ s\ t.\ A \in sets\ (F\ s) \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A\ |A.\ A \in sets\ (F\ t_0)\} \subseteq Pow\ (\{t_0..\} \times space\ M)$ **using** *sets.sets-into-space* **by** *force*
ultimately have $(\lambda x.\ x) \in restrict\text{-}space\ borel\ \{t_0..i\} \bigotimes_M F\ i \rightarrow_M \Sigma_P$ **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* $\in$ *borel-measurable* (*restrict-space borel* $\{t_0..i\} \bigotimes_M F\ i$) **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*: $(\bigcup i.\ \{i\} \times A\ i) = \{s<..t\} \times B\ B \in sets\ (F\ s)\ s < t$ **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 $A\ (Suc\ i) \in sets\ (F\ i)\ A\ 0 \in F\ 0$ **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 ⊆ {0..} × space M using Compl(1) sets.sets-into-space sets-predictable-sigma
space-predictable-sigma by metis
  hence A-in: A i ⊆ space M for i using Compl(4) by blast
  have a: a = {0..} × space M − (⋃ i. {i} × A i) using a-in Compl(4) by blast
  also have ... = − (⋂ j. − ({j} × (space M − A j))) by blast
  also have ... = (⋃ j. {j} × (space M − A j)) by blast
  finally have ∗: (space M − A (Suc i)) ∈ F i (space M − A 0) ∈ 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 ⊆ {0..} × space M for i using Union(1) sets.sets-into-space
sets-predictable-sigma space-predictable-sigma by metis
  hence A-in: A i ⊆ space M for i using Union(4) by blast
  have snd x ∈ snd ‘ (a i ∩ ({fst x} × space M)) if x ∈ a i for i x using that
a-in by fastforce
  hence a-i: a i = (⋃ j. {j} × (snd ‘ (a i ∩ ({j} × space M)))) for i by force
  have A-i: A i = snd ‘ (⋃ (range a) ∩ ({i} × space M)) for i unfolding
Union(4) using A-in by force
  have ∗: snd ‘ (a j ∩ ({Suc i} × space M)) ∈ F i snd ‘ (a j ∩ ({0} × space M))
∈ F 0 for j using Union(2,3)[OF a-i] by auto
  {
    case 1
    have (⋃ j. snd ‘ (a j ∩ ({Suc i} × space M))) ∈ F i using ∗ by fast
    moreover have (⋃ j. snd ‘ (a j ∩ ({Suc i} × space M))) = snd ‘ (⋃ (range
a) ∩ ({Suc i} × space M)) by fast
    ultimately show ?case using A-i by metis
  next
    case 2
    have (⋃ j. snd ‘ (a j ∩ ({0} × space M))) ∈ F 0 using ∗ by fast
    moreover have (⋃ j. snd ‘ (a j ∩ ({0} × space M))) = snd ‘ (⋃ (range a) ∩
({0} × 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*} × *space M* ∈ Σ$_P$ **using** *space-F*[*symmetric*, *of i*] **unfolding** *sets-predictable-sigma* **by** (*intro sigma-sets.Basic*) *blast*
moreover have *case-prod X* −‘ *S* ∩ (*UNIV* × *space M*) ∈ Σ$_P$ **unfolding** *atLeast-0*[*symmetric*] **using** *open-S* **by** (*intro predictableD*, *simp add*: *borel-open*)
ultimately have *case-prod X* −‘ *S* ∩ ({*Suc i*} × *space M*) ∈ Σ$_P$ **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* ∩ ({*Suc i*} × *space M*) = {*Suc i*} × (*X* (*Suc i*) −‘ *S* ∩ *space M*) **by** (*auto simp add*: *le-Suc-eq*)
moreover have ... = (⋃*j*. {*j*} × (*if j* = *Suc i then* (*X* (*Suc i*) −‘ *S* ∩ *space M*) *else* {})) **by** (*force split*: *if-splits*)
ultimately have (⋃*j*. {*j*} × (*if j* = *Suc i then* (*X* (*Suc i*) −‘ *S* ∩ *space M*) *else* {})) ∈ Σ$_P$ **by** *argo*
thus *X* (*Suc i*) −‘ *S* ∩ *space* (*F i*) ∈ *sets* (*F i*) **using** *sets-in-filtration*[*of* $\lambda j.$ *if j* = *Suc i then* (*X* (*Suc i*) −‘ *S* ∩ *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* ⟷ *adapted-process M F 0* ($\lambda i.$ *X* (*Suc i*)) ∧ *X 0* ∈ *borel-measurable* (*F 0*)
proof (*intro iffI*)
assume *asm*: *adapted-process M F 0* ($\lambda i.$ *X* (*Suc i*)) ∧ *X 0* ∈ *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*) ∈ *borel-measurable* Σ$_P$
proof (*intro borel-measurableI*)
fix *S* :: *'b set* **assume** *open-S*: *open S*
have {*i*} × (*X i* −‘ *S* ∩ *space M*) ∈ *sets* Σ$_P$ **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 (λ(*x*, *y*). *X x y*) −‘ *S* ∩ *space* Σ$_P$ = (⋃ *i*. {*i*} × (*X i* −‘ *S* ∩ *space M*)) **by** *fastforce*
ultimately show (λ(*x*, *y*). *X x y*) −‘ *S* ∩ *space* Σ$_P$ ∈ *sets* Σ$_P$ **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* (λ*i*. *X* (*Suc i*)) ∧ *X 0* ∈ *borel-measurable* (*F 0*)
using *predictable-implies-adapted-Suc asm* **by** *auto*
qed

corollary (**in** *nat-filtered-measure*) *predictable-processI*[*intro*!]:
assumes *X 0* ∈ *borel-measurable* (*F 0*) ⋀*i*. *X* (*Suc i*) ∈ *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*: ⋀*i*. $t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *martingale-property*: ⋀*i j*. $t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE* ξ *in M*. *X i* ξ = *cond-exp M* (*F i*) (*X j*) ξ

locale *martingale-order* = *martingale M F* t_0 *X* **for** *M F* t_0 **and** *X* :: - ⇒ - ⇒ - :: {*order-topology*, *ordered-real-vector*}
locale *martingale-linorder* = *martingale M F* t_0 *X* **for** *M F* t_0 **and** *X* :: - ⇒ - ⇒ - :: {*linorder-topology*, *ordered-real-vector*}
sublocale *martingale-linorder* ⊆ *martingale-order* **..**

lemma (**in** *sigma-finite-filtered-measure*) *martingale-const-fun*[*intro*]:
assumes *integrable M f f* ∈ *borel-measurable* (*F* t_0)
shows *martingale M F* t_0 (λ-. *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 (λ*i*. *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 (λ- -. *0*) **by** *fastforce*

corollary (**in** *finite-filtered-measure*) *martingale-const*[*intro*]: *martingale M F* t_0 (λ- -. *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* :: - ⇒ - ⇒ - :: {*order-topology*, *ordered-real-vector*} +
assumes *integrable*: ⋀*i*. $t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *submartingale-property*: ⋀*i j*. $t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE* ξ *in M*. *X i* ξ ≤ *cond-exp M* (*F i*) (*X j*) ξ

locale *submartingale-linorder* = *submartingale M F* t_0 *X* **for** *M F* t_0 **and** *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}

lemma (**in** *sigma-finite-filtered-measure*) *submartingale-const-fun*[*intro*]:
assumes *integrable M f f* ∈ *borel-measurable* (*F* t_0)
shows *submartingale M F* t_0 (λ-. *f*)
proof −
interpret *martingale M F* t_0 λ-. *f* **using** *assms* **by** (*rule martingale-const-fun*)
show *submartingale M F* t_0 (λ-. *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 (λ*i*. *cond-exp M* (*F i*) *f*)
proof −

interpret *martingale M F* t_0 λi. *cond-exp M* (*F i*) *f* **using** *assms* **by** (*rule martingale-cond-exp*)
show *submartingale M F* t_0 (λi. *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 (λ- -. *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* :: - $\Rightarrow$ - $\Rightarrow$ - :: {*order-topology*, *ordered-real-vector*}
+
assumes *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *supermartingale-property*: $\bigwedge i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE* ξ *in M*. *X i* ξ $\geq$ *cond-exp M* (*F i*) (*X j*) ξ

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

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-const-fun*[*intro*]:
assumes *integrable M f f* $\in$ *borel-measurable* (*F* t_0)
shows *supermartingale M F* t_0 (λ-. *f*)
proof −
interpret *martingale M F* t_0 λ-. *f* **using** *assms* **by** (*rule martingale-const-fun*)
show *supermartingale M F* t_0 (λ-. *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 (λi. *cond-exp M* (*F i*) *f*)
proof −
interpret *martingale M F* t_0 λi. *cond-exp M* (*F i*) *f* **using** *assms* **by** (*rule martingale-cond-exp*)
show *supermartingale M F* t_0 (λi. *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 (λ- -. *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 $t_0 \leq i$ $i \leq j$
shows *AE* ξ *in M*. *cond-exp M* (*F i*) ($\lambda\xi$. *X j* ξ − *X i* ξ) ξ = *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* $t_0 \leq i$ $i \leq j$
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* ∂*M*) = ($\int$ *x* $\in$ *A*. *cond-exp M* (*F i*) (*X j*) *x* ∂*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 ... = ($\int$ *x* $\in$ *A*. *X j x* ∂*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 = 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*: *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* ξ + *Y i* ξ = *cond-exp M* (*F i*) ($\lambda x.$ *X j x* + *Y j x*) ξ
using *sigma-finite-subalgebra.cond-exp-add*[*OF - integrable martingale.integrable*[*OF assms*], *of F i j j*, *THEN AE-symmetric*]
martingale-property[*OF asm*] *martingale.martingale-property*[*OF assms asm*] **by** *force*
}
thus *?thesis* **using** *assms*
by (*unfold-locales*) (*auto simp add*: *integrable martingale.integrable*)
qed

lemma *diff*[*intro*]:
assumes *martingale M F* t_0 *Y*
shows *martingale M F* t_0 ($\lambda i\ x.\ X\ i\ x - Y\ i\ x$)
proof −
interpret *Y*: *martingale M F* t_0 *Y* **by** (*rule assms*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i\ i \leq j$
hence *AE* ξ *in M. X i* ξ − *Y i* ξ = *cond-exp M* (*F i*) ($\lambda x.$ *X j x* − *Y j x*) ξ
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* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *diff-zero*: $\bigwedge i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE x in M. cond-exp M* (*F i*) ($\lambda\xi$. *X j* ξ − *X i* ξ) *x* = *0*
shows *martingale M F* t_0 *X*
proof
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix *i j* :: $'b$ **assume** *asm*: $t_0 \leq i$ $i \leq j$
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* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and $\bigwedge A\ i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow A \in F\ i \Longrightarrow$ *set-lebesgue-integral M A* (*X i*) = *set-lebesgue-integral M A* (*X j*)
shows *martingale M F* t_0 *X*
proof (*unfold-locales*)
fix *i j* :: $'b$ **assume** *asm*: $t_0 \leq i$ $i \leq j$
interpret *adapted-process M F* t_0 *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 ξ in restr-to-subalg M (F i). X i ξ = cond-exp M (F i) (X j) ξ* **using** *asm* **by** (*intro r.density-unique-banach*, *auto intro*: *integrable-in-subalg subalg borel-measurable-cond-exp integrable*)
thus *AE ξ in M. X i ξ = cond-exp M (F i) (X j) ξ* **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 $t_0 \leq i$ $i \leq j$
shows *AE x in M. cond-exp M (F i) (λξ. X j ξ − X i ξ) x ≥ 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 *(λi ξ. X i ξ + Y i ξ)*
proof −
interpret *Y*: *submartingale 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 ξ + Y i ξ ≤ cond-exp M (F i) (λx. X j x + Y j x) ξ*
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 *(λi ξ. X i ξ − Y i ξ)*
proof −
interpret *Y*: *supermartingale 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 ξ − Y i ξ ≤ cond-exp M (F i) (λx. X j x − Y j x) ξ*
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 *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* ξ *in M*. $c *_R X\ i\ \xi \leq$ *cond-exp M* $(F\ i)$ $(\lambda\xi.\ c *_R X\ j\ \xi)\ \xi$
using *sigma-finite-subalgebra.cond-exp-scaleR-right*[*OF - integrable*, *of F i j c*] *submartingale-property*[*OF asm*] **by** (*fastforce intro*!: *scaleR-left-mono*[*OF - assms*])
}
qed (*auto simp add*: *borel-measurable-integrable borel-measurable-scaleR integrable random-variable adapted borel-measurable-const-scaleR*)

lemma *scaleR-le-zero*:
assumes $c \leq 0$
shows *supermartingale M F* t_0 $(\lambda i\ \xi.\ c *_R X\ i\ \xi)$
proof
{
fix *i j* :: $'b$ **assume** *asm*: $t_0 \leq i\ i \leq j$
thus *AE* ξ *in M*. $c *_R X\ i\ \xi \geq$ *cond-exp M* $(F\ i)$ $(\lambda\xi.\ c *_R X\ j\ \xi)\ \xi$
using *sigma-finite-subalgebra.cond-exp-scaleR-right*[*OF - integrable*, *of F i j c*] *submartingale-property*[*OF asm*]
by (*fastforce intro*!: *scaleR-left-mono-neg*[*OF - assms*])
}
qed (*auto simp add*: *borel-measurable-integrable borel-measurable-scaleR integrable random-variable adapted borel-measurable-const-scaleR*)

lemma *uminus*[*intro*]:
shows *supermartingale M F* t_0 $(-X)$
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* $(X\ i) \leq$ *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* t_0 *Y*
shows *submartingale M F* t_0 (λ*i* ξ. *max* (*X i* ξ) (*Y i* ξ))
proof (*unfold-locales*)
interpret *Y*: *submartingale-linorder M F* t_0 *Y* **by** (*intro submartingale-linorder.intro assms*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i$ $i \leq j$
have *AE* ξ *in M*. *max* (*X i* ξ) (*Y i* ξ) ≤ *max* (*cond-exp M* (*F i*) (*X j*) ξ) (*cond-exp M* (*F i*) (*Y j*) ξ) **using** *submartingale-property Y.submartingale-property asm* **unfolding** *max-def* **by** *fastforce*
thus *AE* ξ *in M*. *max* (*X i* ξ) (*Y i* ξ) ≤ *cond-exp M* (*F i*) (λξ. *max* (*X j* ξ) (*Y j* ξ)) ξ **using** *sigma-finite-subalgebra.cond-exp-max*[*OF - integrable Y.integrable*, *of F i j j*] *asm* **by** (*fast intro*: *order.trans*)
}
show ⋀*i*. $t_0 \leq i \Longrightarrow$ (λξ. *max* (*X i* ξ) (*Y i* ξ)) ∈ *borel-measurable* (*F i*) ⋀*i*. $t_0 \leq i \Longrightarrow$ *integrable M* (λξ. *max* (*X i* ξ) (*Y i* ξ)) **by** (*force intro*: *Y.integrable integrable assms*)+
qed

lemma *max-0*:
shows *submartingale M F* t_0 (λ*i* ξ. *max 0* (*X i* ξ))
proof −
interpret *zero*: *martingale-linorder M F* t_0 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* t_0 *X*
and *integrable*: ⋀*i*. $t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *diff-nonneg*: ⋀*i j*. $t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE x in M*. *cond-exp M* (*F i*) (λξ. *X j* ξ − *X i* ξ) *x* ≥ *0*
shows *submartingale M F* t_0 *X*
proof (*unfold-locales*)
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i$ $i \leq j$
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 :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and $\bigwedge A\ i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow A \in F\ i \Longrightarrow$ *set-lebesgue-integral M A* (*X i*) ≤ *set-lebesgue-integral M A* (*X j*)
shows *submartingale M F* t_0 *X*
proof (*unfold-locales*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i\ i \leq j$
interpret *adapted-process M F* t_0 *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* ∈ *restr-to-subalg M* (*F i*)
hence ∗: *A* ∈ *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 ... ≤ *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* ≤ *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* (λξ. *cond-exp M* (*F i*) (*X j*) ξ − *X i* ξ) **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* ξ *in restr-to-subalg M* (*F i*). *0* ≤ *cond-exp M* (*F i*) (*X j*) ξ − *X i* ξ
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* ξ *in M*. *X i* ξ ≤ *cond-exp M* (*F i*) (*X j*) ξ **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 $t_0 \leq i\ i \leq j$
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*: $t_0 \leq i\ i \leq j$
hence *AE* ξ *in M*. $X\ i\ \xi + Y\ i\ \xi \geq$ *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*: $t_0 \leq i\ i \leq j$
hence *AE* ξ *in M*. $X\ i\ \xi - Y\ i\ \xi \geq$ *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 $(\lambda i\ \xi.\ c *_R X\ i\ \xi)$

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

lemma *scaleR-le-zero*:
  assumes $c \leq 0$
  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-exp* $M\ (F\ i)\ (\lambda\xi.\ c *_R X\ j\ \xi)\ \xi$
      using *sigma-finite-subalgebra.cond-exp-scaleR-right*[*OF - integrable*, *of F i j c*] *supermartingale-property*[*OF asm*] **by** (*fastforce intro*!: *scaleR-left-mono-neg*[*OF - assms*])
  }
qed (*auto simp add*: *borel-measurable-integrable borel-measurable-scaleR integrable random-variable adapted borel-measurable-const-scaleR*)

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

end

context *supermartingale-linorder*
begin

lemma *set-integral-ge*:
  assumes $A \in F\ i$ $t_0 \leq i$ $i \leq j$
  shows *set-lebesgue-integral* $M\ A\ (X\ i) \geq$ *set-lebesgue-integral* $M\ A\ (X\ j)$
  using *supermartingale-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 *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-locales*)
  interpret Y: *supermartingale-linorder* $M\ F\ t_0\ Y$ **by** (*intro supermartingale-linorder.intro assms*)

{
fix $i\ j :: 'b$ **assume** *asm*: $t_0 \leq i\ i \leq j$
have $AE\ \xi$ *in* M. $min\ (X\ i\ \xi)\ (Y\ i\ \xi) \geq min\ (cond\text{-}exp\ M\ (F\ i)\ (X\ j)\ \xi)\ (cond\text{-}exp\ M\ (F\ i)\ (Y\ j)\ \xi)$ **using** *supermartingale-property Y.supermartingale-property asm* **unfolding** *min-def* **by** *fastforce*
thus $AE\ \xi$ *in* M. $min\ (X\ i\ \xi)\ (Y\ i\ \xi) \geq cond\text{-}exp\ M\ (F\ i)\ (\lambda\xi.\ min\ (X\ j\ \xi)\ (Y\ j\ \xi))\ \xi$ **using** *sigma-finite-subalgebra.cond-exp-min*[*OF - integrable Y.integrable*, *of F i j j*] *asm* **by** (*fast intro*: *order.trans*)
}
show $\bigwedge i.\ t_0 \leq i \Longrightarrow (\lambda\xi.\ min\ (X\ i\ \xi)\ (Y\ i\ \xi)) \in borel\text{-}measurable\ (F\ i)$ $\bigwedge i.\ t_0 \leq i \Longrightarrow integrable\ M\ (\lambda\xi.\ min\ (X\ i\ \xi)\ (Y\ i\ \xi))$ **by** (*force intro*: *Y.integrable integrable assms*)+
qed

lemma *min-0*:
shows *supermartingale M F* t_0 $(\lambda i\ \xi.\ min\ 0\ (X\ i\ \xi))$
proof −
interpret *zero*: *martingale-linorder M F* t_0 λ- -. *0* **by** (*force intro*: *martingale-linorder.intro*)
show *?thesis* **by** (*intro zero.min supermartingale-linorder.intro supermartingale-axioms*)
qed

end

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-of-cond-exp-diff-le-zero*:
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow integrable\ M\ (X\ i)$
and *diff-le-zero*: $\bigwedge i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow AE\ x\ in\ M.\ cond\text{-}exp\ M\ (F\ i)\ (\lambda\xi.\ X\ j\ \xi - X\ i\ \xi)\ x \leq 0$
shows *supermartingale M F* t_0 *X*
proof
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix $i\ j :: 'b$ **assume** *asm*: $t_0 \leq i\ i \leq j$
thus $AE\ \xi$ *in M*. $X\ i\ \xi \geq cond\text{-}exp\ M\ (F\ i)\ (X\ j)\ \xi$
using *diff-le-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*) *supermartingale-of-set-integral-ge*:
fixes $X :: - \Rightarrow - \Rightarrow - :: \{linorder\text{-}topology\}$
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow integrable\ M\ (X\ i)$
and $\bigwedge A\ i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow A \in F\ i \Longrightarrow set\text{-}lebesgue\text{-}integral\ M\ A\ (X\ j) \leq set\text{-}lebesgue\text{-}integral\ M\ A\ (X\ i)$

shows *supermartingale M F* t_0 *X*
proof −
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
note ∗ = *set-integral-uminus*[*unfolded set-integrable-def*, *OF integrable-mult-indicator*[*OF - integrable*]]
have *supermartingale M F* t_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*[*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 $i + 1$, 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*: $\bigwedge i$. *integrable M* (*X i*)

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

lemma *martingale-nat*:
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: $\bigwedge i.$ *integrable M* $(X\ i)$
and $\bigwedge i.$ *AE* ξ *in M*. $X\ i\ \xi =$ *cond-exp M* $(F\ i)\ (X\ (Suc\ i))\ \xi$
shows *martingale M F 0 X*
proof (*unfold-locales*)
interpret *adapted-process M F 0 X* **by** (*rule adapted*)
fix *i j* :: *nat* **assume** *asm*: $i \leq j$
show *AE* ξ *in M*. $X\ i\ \xi =$ *cond-exp M* $(F\ i)\ (X\ j)\ \xi$ **using** *asm*
proof (*induction* $j - i$ *arbitrary*: *i j*)
case *0*
hence $j = i$ **by** *simp*
thus *?case* **using** *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable adapted*, *THEN AE-symmetric*] **by** *blast*
next
case (*Suc n*)
have *j*: $j = Suc\ (n + i)$ **using** *Suc* **by** *linarith*
have *n*: $n = n + i - i$ **using** *Suc* **by** *linarith*
have $*$: *AE* ξ *in M*. *cond-exp M* $(F\ (n + i))\ (X\ j)\ \xi = X\ (n + i)\ \xi$ **unfolding** *j* **using** *assms*(*3*)[*THEN AE-symmetric*] **by** *blast*
have *AE* ξ *in M*. *cond-exp M* $(F\ i)\ (X\ j)\ \xi =$ *cond-exp M* $(F\ i)$ (*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* ξ *in M*. *cond-exp M* $(F\ i)\ (X\ j)\ \xi =$ *cond-exp M* $(F\ i)\ (X\ (n + i))\ \xi$ **using** *cond-exp-cong-AE*[*OF integrable-cond-exp integrable* $*$] **by** *force*
thus *?case* **using** *Suc*(*1*)[*OF n*] **by** *fastforce*
qed
qed (*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*: $\bigwedge i$. *integrable M* (*X i*)
and $\bigwedge i$. *AE* ξ *in M*. *cond-exp M* (*F i*) ($\lambda\xi$. *X* (*Suc i*) ξ − *X i* ξ) ξ = *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* ξ *in M*. *X i* ξ = *cond-exp M* (*F i*) (*X* (*Suc i*)) ξ **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* ≤ *j*
shows *AE* ξ *in M*. *X i* ξ ≤ *X j* ξ
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* λi. *X* (*Suc i*) **by** (*intro predictable-implies-adapted-Suc assms*)
have *Suc i* ≤ *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* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: $\bigwedge i$. *integrable M* (*X i*)
and $\bigwedge A\ i$. *A* ∈ *F i* ⟹ *set-lebesgue-integral M A* (*X i*) ≤ *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* ≤ *j A* ∈ *sets* (*F i*)
show *set-lebesgue-integral M A* (*X i*) ≤ *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*: ⋀*i*. *integrable M* (*X i*)
and ⋀*i*. *AE ξ in M*. *X i ξ* ≤ *cond-exp M* (*F i*) (*X* (*Suc i*)) *ξ*
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 subalgebra-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*: ⋀*i*. *integrable M* (*X i*)
and ⋀*i*. *AE ξ in M*. *cond-exp M* (*F i*) (λξ. *X* (*Suc i*) *ξ* − *X i ξ*) *ξ* ≥ *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 ξ in M*. *X i ξ* ≤ *cond-exp M* (*F i*) (*X* (*Suc i*)) *ξ* **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* ⋀*i*. *AE ξ in M*. *0* ≤ *C i ξ* ⋀*i*. *AE ξ in M*. *C i ξ* ≤ *R*
shows *submartingale M F 0* (λ*n ξ*. ∑ *i*<*n*. *C i ξ* ∗$_R$ (*X* (*Suc i*) *ξ* − *X i ξ*))
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* ξ *in M.* $0 \le$ *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* $\bigwedge i.$ *AE* ξ *in M.* $0 \le C\ i\ \xi \bigwedge i.$ *AE* ξ *in M.* $C\ i\ \xi \le R$
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* $i \le j$
shows *AE* ξ *in M.* $X\ i\ \xi \ge X\ j\ \xi$
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 *supermartingale M F 0 X* **by** (*rule assms*)
interpret *S*: *adapted-process M F 0* λ*i*. *X* (*Suc i*) **by** (*intro predictable-implies-adapted-Suc assms*)
have *Suc i* ≤ *j* **using** *Suc*(*2*,*3*) **by** *linarith*
thus *?case* **using** *Suc*(*1*)[*OF* ∗] *S.adapted*[*of i*] *supermartingale-property*[*OF - le-SucI*, *of i*] *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable*, *of F i Suc i*] **by** *fastforce*
qed

lemma *supermartingale-of-set-integral-ge-Suc*:
fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: ⋀*i*. *integrable M* (*X i*)
and ⋀*A i*. *A* ∈ *F i* ⟹ *set-lebesgue-integral M A* (*X i*) ≥ *set-lebesgue-integral M A* (*X* (*Suc i*))
shows *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* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: ⋀*i*. *integrable M* (*X i*)
and ⋀*i*. *AE* ξ *in M*. *X i* ξ ≥ *cond-exp M* (*F i*) (*X* (*Suc i*)) ξ
shows *supermartingale M F 0 X*
proof −
interpret *adapted-process M F 0 X* **by** (*rule assms*)
have *AE* ξ *in M*. − *X i* ξ ≤ *cond-exp M* (*F i*) (λ*x*. − *X* (*Suc i*) *x*) ξ **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 $X :: - \Rightarrow - \Rightarrow - :: \{linorder\text{-}topology\}$
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: $\bigwedge i$. *integrable M* $(X\ i)$
and $\bigwedge i$. *AE* ξ *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* ξ *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 gambler's 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 gambler's 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 gambler's 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 gambler's 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* ⇒ (*bool stream*) *measure* **where**
bernoulli-stream p = *stream-space* (*measure-pmf* (*bernoulli-pmf p*))

lemma *space-bernoulli-stream*[*simp*]: *space* (*bernoulli-stream p*) = *UNIV* **by** (*simp add*: *bernoulli-stream-def space-stream-space*)

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

definition *fortune* :: *nat* ⇒ *bool stream* ⇒ *real* **where**
fortune n = (λ*s*. ∑ *b* ← *stake* (*Suc n*) *s*. *if b then 1 else* −*1*)

definition *toss* :: *nat* ⇒ *bool stream* ⇒ *real* **where**
toss n = (λ*s*. *if snth s n then 1 else* −*1*)

lemma *toss-indicator-def*: *toss n* = *indicator* {*s*. *s* !! *n*} − *indicator* {*s*. ¬ *s* !! *n*}
unfolding *toss-def indicator-def* **by** *force*

lemma *range-toss*: *range* (*toss n*) = {−*1*, *1*}
proof −
have *sconst True* !! *n* **by** *simp*
moreover have ¬*sconst False* !! *n* **by** *simp*
ultimately have ∃ *x*. *x* !! *n* ∃ *x*. ¬*x* !! *n* **by** *blast*+
thus *?thesis* **unfolding** *toss-def image-def* **by** *auto*
qed

lemma *vimage-toss*: *toss n* −‘ *A* = (*if 1* ∈ *A then* {*s*. *s* !! *n*} *else* {}) ∪ (*if* −*1* ∈ *A then* {*s*. ¬*s* !! *n*} *else* {})
unfolding *vimage-def toss-def* **by** *auto*

lemma *fortune-Suc*: *fortune* (*Suc n*) *s* = *fortune n s* + *toss* (*Suc n*) *s*
by (*induction n arbitrary*: *s*) (*simp add*: *fortune-def toss-def*)+

lemma *fortune-toss-sum*: *fortune n s* = (∑ *i* ∈ {..*n*}. *toss i s*)
by (*induction n arbitrary*: *s*) (*simp add*: *fortune-def toss-def*, *simp add*: *fortune-Suc*)

lemma *fortune-bound*: *norm* (*fortune n s*) ≤ *Suc n* **by** (*induction n*) (*force simp add*: *fortune-toss-sum toss-def*)+

Our definition of *bernoulli-stream* constitutes a probability space.

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

abbreviation *toss-filtration p* ≡ *nat-natural-filtration* (*bernoulli-stream p*) *toss*

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

interpretation *toss*: *adapted-process bernoulli-stream p toss-filtration p 0 toss*
by (*intro adapted-process.intro stochastic-process.adapted-process-natural-filtration*)
(*unfold-locales*, *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-bound*
by (*intro Bochner-Integration.integrable-bound*[*OF integrable-const*[*of - Suc n*] *fortune.random-variable*]) *auto*

We provide the following lemma to explicitly calculate the probability of events in this probability space.

lemma *measure-bernoulli-stream-snth-pred*:
assumes *0* ≤ *p* **and** *p* ≤ *1* **and** *finite J*
shows *prob p* {*w* ∈ *space* (*bernoulli-stream p*). ∀ *j*∈*J*. *P j* = *w* !! *j*} = *p* ^ *card* (*J* ∩ *Collect P*) ∗ (*1* − *p*) ^ (*card* (*J* − *Collect P*))
proof −
let *?PiE* = (Π$_E$ *i*∈*J*. *if P i then* {*True*} *else* {*False*})
have *product-prob-space* (λ-. *measure-pmf* (*bernoulli-pmf p*)) **by** *unfold-locales*

hence ∗: *to-stream* −‘ {*s*. ∀ *i*∈*J*. *P i* = *s* !! *i*} = {*s*. ∀ *i*∈*J*. *P i* = *s i*} **using** *assms* **by** (*simp add*: *to-stream-def*)
also have ... = *prod-emb UNIV* (λ-. *measure-pmf* (*bernoulli-pmf p*)) *J ?PiE*
proof −
{
fix *s* **assume** (∀ *i*∈*J*. *P i* = *s i*)
hence (∀ *i*∈*J*. *P i* = *s i*) = (*s* ∈ *prod-emb UNIV* (λ-. *measure-pmf* (*bernoulli-pmf p*)) *J ?PiE*)
by (*subst prod-emb-iff*[*of s*]) (*smt* (*verit*, *best*) *not-def assms*(*3*) *id-def PiE-eq-singleton UNIV-I extensional-UNIV insert-iff singletonD space-measure-pmf*)
}
moreover
{
fix *s* **assume** ¬(∀ *i*∈*J*. *P i* = *s i*)

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

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

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

lemma *integral-toss*:
assumes $0 \leq p$ $p \leq 1$

shows *expectation p (toss n) = 2 * p - 1*
proof −
have [*simp*]:{*s. s !! n*} ∈ *events p* **using** *measurable-snth*[*THEN measurable-sets*, *of* {*True*} *measure-pmf* (*bernoulli-pmf p*) *n*, *folded bernoulli-stream-def*]
by (*simp add*: *vimage-def*)
have *expectation p* (*toss n*) = *Bochner-Integration.simple-bochner-integral* (*bernoulli-stream p*) (*toss n*)
using *toss.random-variable*[*of n*, *THEN measurable-sets*]
by (*intro simple-bochner-integrable-eq-integral*[*symmetric*] *simple-bochner-integrable.intros*) (*auto simp add*: *toss-def simple-function-def image-def*)
also have ... = *p* − *prob p* {*s.* ¬ *s !! n*} **unfolding** *simple-bochner-integral-def* **using** *measure-bernoulli-stream-snth*[*OF assms*]
by (*simp add*: *range-toss*, *simp add*: *toss-def*)
also have ... = *p* − (*1* − *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* ≤ *p p* ≤ *1*
shows *indep-vars p* (λ*-. borel*) *toss* {*0..*}
proof (*subst indep-vars-def*, *intro conjI indep-sets-sigma*)
{
fix *A J* **assume** *asm*: *J* ≠ {} *finite J* ∀*j*∈*J. A j* ∈ {*toss j* −‘ *A* ∩ *space* (*bernoulli-stream p*) |*A. A* ∈ *borel*}
hence ∀*j*∈*J*. ∃*B* ∈ *borel. A j* = *toss j* −‘ *B* ∩ *space* (*bernoulli-stream p*) **by** *auto*
then obtain *B* **where** *B-is*: *A j* = *toss j* −‘ *B j* ∩ *space* (*bernoulli-stream p*) *B j* ∈ *borel* **if** *j* ∈ *J* **for** *j* **by** *metis*

have *prob p* (⋂ (*A* ‘ *J*)) = (∏*j*∈*J. prob p* (*A j*))
proof *cases*

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

assume ∃*j* ∈ *J. 1* ∉ *B j* ∧ *−1* ∉ *B j*
then obtain *j* **where** *j-is*: *j* ∈ *J 1* ∉ *B j −1* ∉ *B j* **by** *blast*
hence *A-j-empty*: *A j* = {} **using** *B-is* **by** (*force simp add*: *toss-def vimage-def*)
hence ⋂ (*A* ‘ *J*) = {} **using** *j-is* **by** *blast*
moreover have *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 ¬(∃*j* ∈ *J. 1* ∉ *B j* ∧ *−1* ∉ *B j*)
hence ∗: *1* ∈ *B j* ∨ *−1* ∈ *B j* **if** *j* ∈ *J* **for** *j* **using** *that* **by** *blast*

define *J′* **where** [*simp*]: *J′* = {*j* ∈ *J*. (*1* ∈ *B j*) ⟷ (*−1* ∉ *B j*)}

hence *toss j w* $\in$ *B j* $\longleftrightarrow$ (*1* $\in$ *B j*) = *w* !! *j* **if** *j* $\in$ *J'* **for** *w j* **using** *that* **unfolding** *toss-def* **by** *simp*
hence ($\bigcap$ (*A* ' *J'*)) = {*w* $\in$ *space* (*bernoulli-stream p*). $\forall$ *j*$\in$*J'*. (*1* $\in$ *B j*) = *w* !! *j*} **using** *B-is* **by** *force*
hence *prob-J'*: *prob p* ($\bigcap$ (*A* ' *J'*)) = *p* ^ *card* (*J'* $\cap$ {*j*. *1* $\in$ *B j*}) * (*1* − *p*) ^ *card* (*J'* − {*j*. *1* $\in$ *B j*})
using *measure-bernoulli-stream-snth-pred*[*OF assms finite-subset*[*OF* - *asm*(*2*)], *of J'* λ*j*. *1* $\in$ *B j*] **by** *auto*

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

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

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

have *A-j-top*: *A j* = *space* (*bernoulli-stream p*) **if** *j* $\in$ *J* − *J'* **for** *j* **using** *that* * **by** (*auto simp add*: *B-is toss-def*)
hence $\bigcap$ (*A* ' *J*) = $\bigcap$ (*A* ' *J'*) **by** *auto*
hence *prob p* ($\bigcap$ (*A* ' *J*)) = *prob p* ($\bigcap$ (*A* ' *J'*)) **by** *presburger*
also have ... = ($\prod$ *j*$\in$*J'* $\cap$ {*j*. *1* $\in$ *B j*}. *prob p* (*A j*)) * ($\prod$ *j*$\in$*J'* − {*j*. *1* $\in$ *B j*}. *prob p* (*A j*))
by (*simp only*: *prob-J' A-j-True A-j-False measure-bernoulli-stream-snth*[*OF assms*] *measure-bernoulli-stream-neg-snth*[*OF assms*] *cong*: *prod.cong*) *simp*
also have ... = ($\prod$ *j*$\in$*J'*. *prob p* (*A j*)) **using** *asm*(*2*) **by** (*intro prod.Int-Diff*[*symmetric*]) *auto*
also have ... = ($\prod$ *j*$\in$*J'*. *prob p* (*A j*)) * ($\prod$ *j*$\in$*J* − *J'*. *prob p* (*A j*)) **using** *A-j-top prob-space* **by** *simp*
also have ... = ($\prod$ *j*$\in$*J*. *prob p* (*A j*)) **using** *asm*(*2*) **by** (*metis* (*no-types*, *lifting*) *J'-def mem-Collect-eq mult.commute prod.subset-diff subsetI*)
finally show *?thesis* **.**
qed
}
thus *indep-sets p* (λ*i*. {*toss i* −' *A* $\cap$ *space* (*bernoulli-stream p*) |*A*. *A* $\in$ *sets borel*}) {*0*..} **using** *measurable-sets*[*OF toss.random-variable*]
by (*intro indep-setsI subsetI*) *fastforce*
qed (*simp*, *intro Int-stableI*, *simp*, *metis sets.Int vimage-Int*)

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

theorem *fortune-martingale*:
assumes *p* = *1*/*2*
shows *martingale* (*bernoulli-stream p*) (*toss-filtration p*) *0 fortune*
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*]
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* ≤ *p p* ≤ *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* ≤ *cond-exp* (*bernoulli-stream p*) (*toss-filtration p n*) (λ*ξ*. *fortune* (*Suc n*) *ξ* − *fortune n ξ*) *ξ*
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* ≤ *p p* ≤ *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* ≥ *cond-exp* (*bernoulli-stream p*) (*toss-filtration p n*) (λ*ξ*. *fortune* (*Suc n*) *ξ* − *fortune n ξ*) *ξ*
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.