

On the Formalization of Martingales

Ata Keskin

March 29, 2024

Abstract

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

The current formalization of conditional expectation in the Isabelle library is limited to real-valued functions. To overcome this limitation, we extend the construction of conditional expectation to general Banach spaces, employing an approach similar to the one described in [2]. We use measure theoretic arguments to construct the conditional expectation using suitable limits of simple functions.

Subsequently, we define stochastic processes and introduce the concepts of adapted, progressively measurable and predictable processes using suitable locale definitions¹. We show the relation

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

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

Moving forward, we rigorously define martingales, submartingales, and supermartingales, presenting their first consequences and corollaries². Discrete-time martingales are given special attention in the formalization. In every step of our formalization, we make extensive use of the powerful locale system of Isabelle.

The formalization further contributes by generalizing concepts in Bochner integration by extending their application from the real numbers to arbitrary Banach spaces equipped with a second-countable topology. Induction schemes for integrable simple functions on Banach spaces are introduced, accommodating various scenarios with or without a real vector ordering³. Specifically, we formalize a powerful result called the ‘‘Averaging Theorem’’[4] which allows us to show that densities are unique in Banach spaces.

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

¹Martingale.Stochastic_Process

²Martingale.Martingale

³Martingale.Bochner_Integration_Addendum

Contents

1	Supplementary Lemmas for Measure Spaces	4
1.1	σ -Algebra Generated by a Family of Functions	4
2	Supplementary Lemmas for Elementary Metric Spaces	5
2.1	Diameter Lemma	5
3	Supplementary Lemmas for Bochner Integration	7
3.1	Integrable Simple Functions	7
3.2	Totally Ordered Banach Spaces	16
3.3	Integrability and Measurability of the Diameter	19
3.4	Auxiliary Lemmas for Set Integrals	20
3.5	Averaging Theorem	21
4	Conditional Expectation in Banach Spaces	27
4.1	Existence	32
4.2	Properties	42
4.3	Linearly Ordered Banach Spaces	48
4.4	Probability Spaces	52
5	Filtered Measure Spaces	57
5.1	Filtered Measure	57
5.2	σ -Finite Filtered Measure	58
5.3	Finite Filtered Measure	59
5.4	Constant Filtration	59
6	Stochastic Processes	59
6.1	Stochastic Process	59
6.1.1	Natural Filtration	61
6.2	Adapted Process	64
6.3	Progressively Measurable Process	66
6.4	Predictable Process	69
7	Martingales	79
7.1	Additional Locale Definitions	79
7.2	Martingale	80
7.3	Submartingale	81
7.4	Supermartingale	81
7.5	Martingale Lemmas	82
7.6	Submartingale Lemmas	85
7.7	Supermartingale Lemmas	88
7.8	Discrete Time Martingales	92
7.9	Discrete Time Submartingales	94
7.10	Discrete Time Supermartingales	96


```

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

```

1 Supplementary Lemmas for Measure Spaces

1.1 σ -Algebra Generated by a Family of Functions

definition *family-vimage-algebra* :: 'a set \Rightarrow ('a \Rightarrow 'b) set \Rightarrow 'b measure \Rightarrow 'a measure **where**

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

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

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

lemma

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

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

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

lemma *measurable-family-vimage-algebra*:

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

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

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

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

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

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

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

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

lemma *measurable-family-iff-sets*:

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

proof (*standard, goal-cases*)

case 1

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

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

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

```

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

```

```

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

```

```

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

```

2 Supplementary Lemmas for Elementary Metric Spaces

2.1 Diameter Lemma

```

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

```

```

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

```

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

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

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

lemma *cauchy-iff-diameter-tends-to-zero-and-bounded*:
fixes $s :: \text{nat} \Rightarrow 'a :: \text{metric-space}$
shows $\text{Cauchy } s \longleftrightarrow ((\lambda n. \text{diameter } \{s \ i \mid i. i \geq n\}) \longrightarrow 0 \wedge \text{bounded } (\text{range } s))$

proof –

have $\{s \ i \mid i. N \leq i\} \neq \{\}$ **for** N **by** *blast*
hence *diameter-SUP: diameter* $\{s \ i \mid i. N \leq i\} = (\text{SUP } (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i) \ (s \ j))$ **for** N **unfolding** *diameter-def* **by** (*auto intro!: arg-cong[of - - Sup]*)
show *?thesis*
proof (*intro iffI*)
assume *asm: Cauchy s*
have $\exists N. \forall n \geq N. \text{norm } (\text{diameter } \{s \ i \mid i. n \leq i\}) < e$ **if** *e-pos: $e > 0$* **for** e
proof –
obtain N **where** *dist-less: dist* $(s \ n) \ (s \ m) < (1/2) * e$ **if** $n \geq N \ m \geq N$
for $n \ m$ **using** *asm e-pos* **by** (*metis Cauchy-def mult-pos-pos zero-less-divide-iff zero-less-numeral zero-less-one*)
{
fix r **assume** $r \geq N$
hence *dist* $(s \ n) \ (s \ m) < (1/2) * e$ **if** $n \geq r \ m \geq r$ **for** $n \ m$ **using** *dist-less*
that **by** *simp*
hence $(\text{SUP } (i, j) \in \{r..\} \times \{r..\}. \text{dist } (s \ i) \ (s \ j)) \leq (1/2) * e$ **by** (*intro cSup-least*) *fastforce+*
also **have** $\dots < e$ **using** *e-pos* **by** *simp*
finally **have** *diameter* $\{s \ i \mid i. r \leq i\} < e$ **using** *diameter-SUP* **by** *presburger*
}
moreover **have** *diameter* $\{s \ i \mid i. r \leq i\} \geq 0$ **for** r **unfolding** *diameter-SUP*
using *bounded-imp-dist-bounded[OF cauchy-imp-bounded, THEN bounded-imp-bdd-above, OF asm]* **by** (*intro cSup-upper2, auto*)
ultimately **show** *?thesis* **by** *auto*

qed

thus $(\lambda n. \text{diameter } \{s \ i \mid i. n \leq i\}) \longrightarrow 0 \wedge \text{bounded } (\text{range } s)$ **using** *cauchy-imp-bounded[OF asm]* **by** (*simp add: LIMSEQ-iff*)

next

assume *asm: $(\lambda n. \text{diameter } \{s \ i \mid i. n \leq i\}) \longrightarrow 0 \wedge \text{bounded } (\text{range } s)$*

have $\exists N. \forall n \geq N. \forall m \geq N. \text{dist } (s \ n) \ (s \ m) < e$ **if** *e-pos: $e > 0$* **for** e

proof –

obtain N **where** *diam-less: diameter* $\{s \ i \mid i. r \leq i\} < e$ **if** $r \geq N$ **for** r

```

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

end

```

```

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

```

3 Supplementary Lemmas for Bochner Integration

3.1 Integrable Simple Functions

We restate some basic results concerning Bochner-integrable functions.

lemma *integrable-implies-simple-function-sequence*:

```

fixes f :: 'a  $\Rightarrow$  'b::{banach, second-countable-topology}
assumes integrable M f
obtains s where  $\bigwedge i.$  simple-function M (s i)
  and  $\bigwedge i.$  emeasure M {y  $\in$  space M. s i y  $\neq$  0}  $\neq$   $\infty$ 
  and  $\bigwedge x.$   $x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x$ 
  and  $\bigwedge x\ i.$   $x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$ 

```

proof –

```

have f:  $f \in \text{borel-measurable } M (\int^+ x. \text{norm } (f\ x) \partial M) < \infty$  using assms
unfolding integrable-iff-bounded by auto

```

```

obtain s where  $s: \bigwedge i.$  simple-function M (s i)  $\bigwedge x.$   $x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x$ 
 $\bigwedge i\ x.$   $x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$  using
borel-measurable-implies-sequence-metric[OF f(1)] unfolding norm-conv-dist by
metis

```

```
{
```

```
  fix i
```

```
  have  $(\int^+ x. \text{norm } (s\ i\ x) \partial M) \leq (\int^+ x. \text{ennreal } (2 * \text{norm } (f\ x)) \partial M)$  using
  s by (intro nn-integral-mono, auto)
```

```
  also have  $\dots < \infty$  using f by (simp add: nn-integral-cmult ennreal-mult-less-top
  ennreal-mult)
```

```
  finally have sbi: Bochner-Integration.simple-bochner-integrable M (s i) using
  s by (intro simple-bochner-integrableI-bounded) auto
```

```
  hence emeasure M {y  $\in$  space M. s i y  $\neq$  0}  $\neq$   $\infty$  by (auto intro: inte-
```

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

Simple functions can be represented by sums of indicator functions.

lemma *simple-function-indicator-representation:*

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

assumes *simple-function* $M f x \in \text{space } M$

shows $f x = (\sum y \in f \text{ ' space } M. \text{indicator } (f \text{ -' } \{y\} \cap \text{space } M) x *_R y)$

(**is** $?l = ?r$)

proof –

have $?r = (\sum y \in f \text{ ' space } M.$

(*if* $y = f x$ *then* $\text{indicator } (f \text{ -' } \{y\} \cap \text{space } M) x *_R y$ *else* 0) **by** (*auto intro!*:
sum.cong)

also have $\dots = \text{indicator } (f \text{ -' } \{f x\} \cap \text{space } M) x *_R f x$ **using** *assms* **by** (*auto*
dest: simple-functionD)

also have $\dots = f x$ **using** *assms* **by** (*auto simp: indicator-def*)

finally show *?thesis* **by** *auto*

qed

lemma *simple-function-indicator-representation-AE:*

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

assumes *simple-function* $M f$

shows *AE* x *in* $M. f x = (\sum y \in f \text{ ' space } M. \text{indicator } (f \text{ -' } \{y\} \cap \text{space } M) x$
 $*_R y)$

by (*metis (mono-tags, lifting) AE-I2 simple-function-indicator-representation*
assms)

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

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

Induction rule for simple integrable functions.

lemma *integrable-simple-function-induct*[*consumes 2, case-names cong indicator*
add, induct set: simple-function]:

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

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

assumes $\text{cong}: \bigwedge f g. \text{simple-function } M f \implies \text{emeasure } M \{y \in \text{space } M. f y \neq$
 $0\} \neq \infty$

$\implies \text{simple-function } M g \implies \text{emeasure } M \{y \in \text{space } M. g y \neq$
 $0\} \neq \infty$

$\implies (\bigwedge x. x \in \text{space } M \implies f x = g x) \implies P f \implies P g$

assumes *indicator*: $\bigwedge A y. A \in \text{sets } M \implies \text{emeasure } M A < \infty \implies P (\lambda x.$
indicator $A x *_R y)$

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

$\text{simple-function } M g \implies \text{emeasure } M \{y \in \text{space } M. g y \neq 0\} \neq$
 $\infty \implies$

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

have $*$: $(\sum y \in \text{insert } x F. \text{indicat-real } (f \text{ - ' } \{y\} \cap \text{space } M) x *_R y) = (\sum y \in F. \text{indicat-real } (f \text{ - ' } \{y\} \cap \text{space } M) x *_R y) + \text{indicat-real } (f \text{ - ' } \{x\} \cap \text{space } M) x *_R x$ **for** x **by** $(\text{metis } (\text{no-types, lifting}) \text{Diff-empty Diff-insert0 add commute insert.hyps}(1) \text{insert.hyps}(2) \text{sum.insert-remove})$

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

```

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

```

Induction rule for non-negative simple integrable functions

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

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

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

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

assumes *indicator: ∧ A y. y ≥ 0 ⇒ A ∈ sets M ⇒ emeasure M A < ∞ ⇒ P (λx. indicator A x *_R y)*

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

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

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

P f ⇒ P g ⇒ P (λx. f x + g x)

shows *P f*

proof–

let $?f = \lambda x. (\sum y \in f^{-1} \text{ space } M. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y)$

have $f\text{-ae-eq}: f x = ?f x$ **if** $x \in \text{space } M$ **for** x **using** *simple-function-indicator-representation*[*OF* $f(1)$ *that*].

moreover have $\text{emeasure } M \{y \in \text{space } M. ?f y \neq 0\} \neq \infty$ **by** (*metis* (*no-types*, *lifting*) *Collect-cong* *calculation* $f(2)$)

moreover have $P (\lambda x. \sum y \in S. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y)$
simple-function $M (\lambda x. \sum y \in S. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y)$

$\text{emeasure } M \{y \in \text{space } M. (\sum x \in S. \text{indicat-real } (f^{-1} \{x\} \cap \text{space } M) y *_R x) \neq 0\} \neq \infty$

$\bigwedge x. x \in \text{space } M \implies 0 \leq (\sum y \in S. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y)$

if $S \subseteq f^{-1} \text{ space } M$ **for** S **using** *simple-functionD(1)*[*OF* *assms(1)*], *THEN* *rev-finite-subset*, *OF* *that* *that*

proof (*induction rule: finite-subset-induct'*)

case *empty*

{

case 1

then show $?case$ **using** *indicator*[*of* 0 {}] **by** *force*

next

case 2

then show $?case$ **by** *force*

next

case 3

then show $?case$ **by** *force*

next

case 4

then show $?case$ **by** *force*

}

next

case (*insert* $x F$)

have $(f^{-1} \{x\} \cap \text{space } M) \subseteq \{y \in \text{space } M. f y \neq 0\}$ **if** $x \neq 0$ **using** *that* **by** *blast*

moreover have $\{y \in \text{space } M. f y \neq 0\} = \text{space } M - (f^{-1} \{0\} \cap \text{space } M)$ **by** *blast*

moreover have $\text{space } M - (f^{-1} \{0\} \cap \text{space } M) \in \text{sets } M$ **using** *simple-functionD(2)*[*OF* $f(1)$] **by** *blast*

ultimately have $\text{fin-0}: \text{emeasure } M (f^{-1} \{x\} \cap \text{space } M) < \infty$ **if** $x \neq 0$ **using** *that* **by** (*metis* *emeasure-mono* $f(2)$ *infinity-ennreal-def* *top.not-eq-extremum* *top-unique*)

hence $\text{fin-1}: \text{emeasure } M \{y \in \text{space } M. \text{indicat-real } (f^{-1} \{x\} \cap \text{space } M) y *_R x \neq 0\} \neq \infty$ **if** $x \neq 0$ **by** (*metis* (*mono-tags*, *lifting*) *emeasure-mono* $f(1)$ *indicator-simps(2)* *linorder-not-less* *mem-Collect-eq* *scaleR-eq-0-iff* *simple-functionD(2)* *subsetI* *that*)

have *nonneg-x*: $x \geq 0$ **using** *insert* f **by** *blast*

have $*$: $(\sum y \in \text{insert } x F. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y) = (\sum y \in F. \text{indicat-real } (f^{-1} \{y\} \cap \text{space } M) x *_R y) + \text{indicat-real } (f^{-1} \{x\} \cap$

$space\ M)$ $xa *_{R} x$ **for** xa **by** (*metis (no-types, lifting) add commute insert.hyps(1) insert.hyps(4) sum.insert*)

have **: $\{y \in space\ M. (\sum x \in insert\ x\ F. indicat-real\ (f - \{x\} \cap space\ M)\ y *_{R} x) \neq 0\} \subseteq \{y \in space\ M. (\sum x \in F. indicat-real\ (f - \{x\} \cap space\ M)\ y *_{R} x) \neq 0\} \cup \{y \in space\ M. indicat-real\ (f - \{x\} \cap space\ M)\ y *_{R} x \neq 0\}$ **unfolding**

*** by** *fastforce*

{

case 1

show *?case*

proof (*cases x = 0*)

case True

then show *?thesis unfolding * using insert by simp*

next

case False

have *norm-argument: norm ((\sum y \in F. indicat-real (f - \{y\} \cap space M) z *_{R} y) + indicat-real (f - \{x\} \cap space M) z *_{R} x) = norm (\sum y \in F. indicat-real (f - \{y\} \cap space M) z *_{R} y) + norm (indicat-real (f - \{x\} \cap space M) z *_{R} x)*

if $z \in space\ M$ **for** z

proof (*cases f z = x*)

case True

have *indicat-real (f - \{y\} \cap space M) z *_{R} y = 0* **if** $y \in F$ **for** y **using** *True insert z that 1 unfolding indicator-def by force*

hence $(\sum y \in F. indicat-real (f - \{y\} \cap space M) z *_{R} y) = 0$ **by** (*meson sum.neutral*)

thus *?thesis by force*

qed (*force*)

show *?thesis using False fin-0 fin-1 f norm-argument by (subst *, subst add, presburger add: insert, intro insert, intro insert, fastforce simp add: indicator-def intro!: insert(2) f(3), auto intro!: indicator insert nonneg-x)*

qed

next

case 2

show *?case*

proof (*cases x = 0*)

case True

then show *?thesis unfolding * using insert by simp*

next

case False

then show *?thesis unfolding * using insert simple-functionD(2)[OF f(1)]*

by *fast*

qed

next

case 3

show *?case*

proof (*cases x = 0*)

case True

then show *?thesis unfolding * using insert by simp*

next

case False

have $\text{emeasure } M \{y \in \text{space } M. (\sum x \in \text{insert } x F. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x) \neq 0\} \leq \text{emeasure } M \{y \in \text{space } M. (\sum x \in F. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x) \neq 0\} \cup \{y \in \text{space } M. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x \neq 0\}$
using $** \text{simple-functionD}(2)[OF \text{insert}(6)] \text{simple-functionD}(2)[OF f(1)] \text{insert.IH}(2)$ **by** $(\text{intro } \text{emeasure-mono}, \text{blast}, \text{simp})$
also have $\dots \leq \text{emeasure } M \{y \in \text{space } M. (\sum x \in F. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x) \neq 0\} + \text{emeasure } M \{y \in \text{space } M. \text{indicat-real } (f - \{x\} \cap \text{space } M) y *_R x \neq 0\}$
using $\text{simple-functionD}(2)[OF \text{insert}(6)] \text{simple-functionD}(2)[OF f(1)]$
by $(\text{intro } \text{emeasure-subadditive}, \text{force+})$
also have $\dots < \infty$ **using** $\text{insert}(7) \text{fin-1}[OF \text{False}]$ **by** $(\text{simp } \text{add: less-top})$
finally show $?thesis$ **by** simp
qed
next
case (ξ)
thus $?case$ **using** $\text{insert nonneg-x } f(3)$ **by** $(\text{auto } \text{simp } \text{add: scaleR-nonneg-nonneg } \text{intro: sum-nonneg})$
}
qed
moreover have $\text{simple-function } M (\lambda x. \sum y \in f^{-1} \text{space } M. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y)$ **using** calculation **by** blast
moreover have $P (\lambda x. \sum y \in f^{-1} \text{space } M. \text{indicat-real } (f - \{y\} \cap \text{space } M) x *_R y)$ **using** calculation **by** blast
moreover have $\bigwedge x. x \in \text{space } M \implies 0 \leq f x$ **using** $f(3)$ **by** simp
ultimately show $?thesis$ **by** $(\text{intro } \text{cong}[OF \text{---} f(1,2)], \text{blast}, \text{blast}, \text{fast})$
 presburger+
qed

lemma $\text{finite-nn-integral-imp-ae-finite}$:

fixes $f :: 'a \Rightarrow \text{ennreal}$
assumes $f \in \text{borel-measurable } M (\int^+ x. f x \partial M) < \infty$
shows $AE x \text{ in } M. f x < \infty$
proof $(\text{rule } \text{ccontr}, \text{goal-cases})$
case 1
let $?A = \text{space } M \cap \{x. f x = \infty\}$
have $*$: $\text{emeasure } M ?A > 0$ **using** $1 \text{assms}(1)$ **by** $(\text{metis } (\text{mono-tags}, \text{lifting}) \text{assms}(2) \text{eventually-mono } \text{infinity-ennreal-def } \text{nn-integral-noteq-infinite } \text{top.not-eq-extremum})$
have $(\int^+ x. f x * \text{indicator } ?A x \partial M) = (\int^+ x. \infty * \text{indicator } ?A x \partial M)$ **by** $(\text{metis } (\text{mono-tags}, \text{lifting}) \text{indicator-inter-arith } \text{indicator-simps}(2) \text{mem-Collect-eq } \text{mult-eq-0-iff})$
also have $\dots = \infty * \text{emeasure } M ?A$ **using** $\text{assms}(1)$ **by** $(\text{intro } \text{nn-integral-cmult-indicator}, \text{simp})$
also have $\dots = \infty$ **using** $*$ **by** fastforce
finally have $(\int^+ x. f x * \text{indicator } ?A x \partial M) = \infty$.
moreover have $(\int^+ x. f x * \text{indicator } ?A x \partial M) \leq (\int^+ x. f x \partial M)$ **by** $(\text{intro } \text{nn-integral-mono}, \text{simp } \text{add: indicator-def})$
ultimately show $?case$ **using** $\text{assms}(2)$ **by** simp
qed

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

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

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

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

and $\bigwedge e. e > 0 \implies \exists N. \forall i \geq N. \forall j \geq N. \text{LINT } x | M. \text{norm } (s\ i\ x - s\ j\ x) < e$

obtains r **where** *strict-mono* r *AE* x *in* M . *Cauchy* $(\lambda i. s\ (r\ i)\ x)$

proof –

have $\exists r. \forall n. (\forall i \geq r\ n. \forall j \geq r\ n. \text{LINT } x | M. \text{norm } (s\ i\ x - s\ j\ x) < (1 / 2) \wedge n) \wedge (r\ (\text{Suc } n) > r\ n)$

proof (*intro dependent-nat-choice, goal-cases*)

case 1

then show *?case using assms(2) by presburger*

next

case (2 $x\ n$)

obtain N **where** $*$: $\text{LINT } x | M. \text{norm } (s\ i\ x - s\ j\ x) < (1 / 2) \wedge \text{Suc } n$ **if** $i \geq N$ **for** $i\ j$ **using** *assms(2)[of (1 / 2) ^ Suc n]* **by** *auto*

{

fix $i\ j$ **assume** $i \geq \max N (\text{Suc } x)$ $j \geq \max N (\text{Suc } x)$

hence $\text{integral}^L M (\lambda x. \text{norm } (s\ i\ x - s\ j\ x)) < (1 / 2) \wedge \text{Suc } n$ **using** $*$ **by** *fastforce*

}

then show *?case by fastforce*

qed

then obtain r **where** *strict-mono*: *strict-mono* r **and** $\forall i \geq r\ n. \forall j \geq r\ n. \text{LINT } x | M. \text{norm } (s\ i\ x - s\ j\ x) < (1 / 2) \wedge n$ **for** n **using** *strict-mono-Suc-iff* **by** *blast*

hence *r-is*: $\text{LINT } x | M. \text{norm } (s\ (r\ (\text{Suc } n))\ x - s\ (r\ n)\ x) < (1 / 2) \wedge n$ **for** n **by** (*simp add: strict-mono-leD*)

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

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

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

proof –

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

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

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

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

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

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

qed

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

ultimately have *ae-summable*: *AE* x *in* M . *summable* $(\lambda i. s (r (Suc i)) x - s (r i) x)$ **using** *summable-norm-cancel* **unfolding** *dist-norm* **by** *force*

{
fix x **assume** *summable* $(\lambda i. s (r (Suc i)) x - s (r i) x)$
hence $(\lambda n. \sum_{i < n}. s (r (Suc i)) x - s (r i) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x)$ **using** *summable-LIMSEQ* **by** *blast*
moreover have $(\lambda n. (\sum_{i < n}. s (r (Suc i)) x - s (r i) x)) = (\lambda n. s (r n) x - s (r 0) x)$ **using** *sum-lessThan-telescope* **by** *fastforce*
ultimately have $(\lambda n. s (r n) x - s (r 0) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x)$ **by** *argo*
hence $(\lambda n. s (r n) x - s (r 0) x + s (r 0) x) \longrightarrow (\sum i. s (r (Suc i)) x - s (r i) x) + s (r 0) x$ **by** (*intro isCont-tendsto-compose* [*of - λz. z + s (r 0) x*], *auto*)
hence *Cauchy* $(\lambda n. s (r n) x)$ **by** (*simp add: LIMSEQ-imp-Cauchy*)
}
hence *AE* x *in* M . *Cauchy* $(\lambda i. s (r i) x)$ **using** *ae-summable* **by** *fast*
thus *?thesis* **by** (*rule that* [*OF strict-mono(1)*])
qed

3.2 Totally Ordered Banach Spaces

lemma *integrable-max* [*simp*, *intro*]:

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

assumes *fg* [*measurable*]: *integrable* M *f* *integrable* M g

shows *integrable* M $(\lambda x. \max (f x) (g x))$

proof (*rule Bochner-Integration.integrable-bound*)


```

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

```

```

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

```

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

```

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

```

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

lemma *integral-mono-AE-banach*:

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

lemma *integral-mono-banach*:

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

3.3 Integrability and Measurability of the Diameter

context

fixes $s :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, banach}\}$ **and** M
assumes $\text{bounded}: \bigwedge x. x \in \text{space } M \implies \text{bounded } (\text{range } (\lambda i. s \ i \ x))$

begin

lemma *borel-measurable-diameter*:

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

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

proof –

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

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

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

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

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

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

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

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

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

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

ultimately show *?thesis* **unfolding** * **by** (*auto intro!*: *borel-measurable-cSUP bdd*)

qed

lemma *integrable-bound-diameter*:

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

assumes *integrable* $M \ f$

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

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

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

proof –

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

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

{

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

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

have *bounded* $?S$ **by** (*rule bounded-imp-dist-bounded[OF bounded[OF x]]*)
hence *Sup-S-nonneg:0* $\leq \text{Sup } ?S$ **by** (*auto intro!: cSup-upper2 x bounded-imp-bdd-above*)

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

3.4 Auxiliary Lemmas for Set Integrals

lemma *set-integral-scaleR-left*:

assumes $A \in \text{sets } M \ c \neq 0 \implies \text{integrable } M \ f$
shows $\text{LINT } t:A|M. f \ t *_{\mathbb{R}} c = (\text{LINT } t:A|M. f \ t) *_{\mathbb{R}} c$
unfolding *set-lebesgue-integral-def*
using *integrable-mult-indicator[OF assms]*
by (*subst integral-scaleR-left[symmetric], auto*)

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

assumes [*measurable*]:*integrable M f*
and *AE x ∈ A in M. 0 ≤ f x A ∈ sets M*
shows $(\int^{+x \in A. f \ x} \partial M) = (\int \ x \in A. f \ x \ \partial M)$

proof –

have $(\int^{+x. \text{indicator } A \ x *_{\mathbb{R}} f \ x} \ \partial M) = (\int \ x \in A. f \ x \ \partial M)$
unfolding *set-lebesgue-integral-def* **using** *assms(2)* **by** (*intro nn-integral-eq-integral[of - λx. indicat-real A x *_R f x], blast intro: assms integrable-mult-indicator, fastforce*)
moreover have $(\int^{+x. \text{indicator } A \ x *_{\mathbb{R}} f \ x} \ \partial M) = (\int^{+x \in A. f \ x} \ \partial M)$ **by** (*metis ennreal-0 indicator-simps(1) indicator-simps(2) mult commute mult-1 mult-zero-left real-scaleR-def*)

ultimately show *?thesis* **by** *argo*

qed

lemma *set-integral-restrict-space*:

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

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

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

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

3.5 Averaging Theorem

We aim to lift results from the real case to arbitrary Banach spaces. Our fundamental tool in this regard will be the averaging theorem. The proof of this theorem is due to Serge Lang (Real and Functional Analysis) [4]. The theorem allows us to make statements about a functions value almost everywhere, depending on the value its integral takes on various sets of the measure space.

Before we introduce and prove the averaging theorem, we will first show the following lemma which is crucial for our proof. While not stated exactly in this manner, our proof makes use of the characterization of second-countable topological spaces given in the book General Topology by Ryszard Engelking (Theorem 4.1.15) [1].

lemma *balls-countable-basis*:

obtains $D :: 'a :: \{\text{metric-space, second-countable-topology}\}$ set
where *topological-basis* (*case-prod ball ' (D × (Q ∩ {0<..}))*)
and *countable D*
and $D \neq \{\}$

proof –
obtain $D :: 'a$ set **where** *dense-subset: countable D D ≠ {}* $[[\text{open } U; U \neq \{\}]]$
 $\implies \exists y \in D. y \in U$ **for** U **using** *countable-dense-exists by blast*
have *topological-basis* (*case-prod ball ' (D × (Q ∩ {0<..}))*)
proof (*intro topological-basis-iff[THEN iffD2], fast, clarify*)
fix U **and** $x :: 'a$ **assume** *asm: open U x ∈ U*
obtain e **where** $e > 0$ *ball x e ⊆ U* **using** *asm openE by blast*
obtain y **where** $y \in D$ $y \in \text{ball } x (e / 3)$ **using** *dense-subset(3)[OF open-ball, of x e / 3] centre-in-ball[THEN iffD2, OF divide-pos-pos[OF e(1), of 3]] by force*
obtain r **where** $r \in \mathbf{Q} \cap \{e/3 <.. < e/2\}$ **unfolding** *Rats-def* **using** *of-rat-dense[OF divide-strict-left-mono[OF - e(1)], of 2 3] by auto*

have $*$: $x \in \text{ball } y r$ **using** r y **by** (*simp add: dist-commute*)
hence $\text{ball } y r \subseteq U$ **using** r **by** (*intro order-trans[OF - e(2)], simp, metric*)
moreover **have** $\text{ball } y r \in (\text{case-prod ball ' (D × (Q ∩ {0<..}))})$ **using** $y(1)$
 r **by force**
ultimately show $\exists B' \in (\text{case-prod ball ' (D × (Q ∩ {0<..}))}). x \in B' \wedge B' \subseteq U$ **using** $*$ **by meson**
qed
thus *?thesis* **using** *that dense-subset by blast*
qed

context *sigma-finite-measure*

begin

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

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

assumes $\bigwedge(N :: 'a \text{ measure}) \Omega. \text{finite-measure } N$
 $\implies N = \text{restrict-space } M \Omega$
 $\implies \Omega \in \text{sets } M$
 $\implies \text{emeasure } N \Omega \neq \infty$
 $\implies \text{emeasure } N \Omega \neq 0$
 $\implies \text{almost-everywhere } N Q$

and [*measurable*]: *Measurable.pred M Q*
shows *almost-everywhere M Q*

proof –
have $*$: *almost-everywhere N Q* **if** *finite-measure N N = restrict-space M Ω Ω ∈ sets M emeasure N Ω ≠ ∞* **for** $N \Omega$ **using** *that by (cases emeasure N Ω = 0, auto intro: emeasure-0-AE assms(1))*

obtain $A :: \text{nat} \Rightarrow 'a$ set **where** A : *range A ⊆ sets M (∪ i. A i) = space M* **and** *emeasure-finite: emeasure M (A i) ≠ ∞* **for** i **using** *sigma-finite by metis*

```

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

```

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

lemma *averaging-theorem*:

```

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

```

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

```

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

```

space-restrict-space that(1))

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

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

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

qed

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

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

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

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

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

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

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

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

have *Union-range-b: ∪(range b) = -S* **using** *B-balls range-b* **by** *simp*

{

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

define A **where** $A = f^{-1} \text{ ball } v r \cap \text{space } N$

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

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

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

have *emeasure N A = 0*

proof -

{

assume *asm: emeasure N A > 0*

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

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

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

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

lemma *density-zero:*

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

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

lemma *density-unique-banach:*

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

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

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

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

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

proof

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

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

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

```

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

end

end

```

```

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

```

4 Conditional Expectation in Banach Spaces

While constructing the conditional expectation operator, we have come up with the following approach, which is based on the construction in [2]. Both our approach, and the one in [2] are based on showing that the conditional expectation is a contraction on some dense subspace of the space of functions $L^1(E)$. In our approach, we start by constructing the conditional expectation explicitly for simple functions. Then we show that the conditional expectation is a contraction on simple functions, i.e. $\|E(s|F)(x)\| \leq E(\|s(x)\||F)$ for μ -almost all $x \in \Omega$ with $s : \Omega \rightarrow E$ simple and integrable. Using this, we can show that the conditional expectation of a convergent sequence of simple functions is again convergent. Finally, we show that this limit exhibits the properties of a conditional expectation. This approach has the benefit of being straightforward and easy to implement, since we could make use of the existing formalization for real-valued functions. To use the construction in [2] we need more tools from functional analysis, which Isabelle/HOL currently does not have.

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

```

definition has-cond-exp :: 'a measure  $\Rightarrow$  'a measure  $\Rightarrow$  ('a  $\Rightarrow$  'b)  $\Rightarrow$  ('a  $\Rightarrow$  'b::{real-normed-vector,
second-countable-topology})  $\Rightarrow$  bool where
  has-cond-exp M F f g = (( $\forall A \in \text{sets } F. (\int x \in A. f\ x\ \partial M) = (\int x \in A. g\ x\ \partial M)$ )

```

```

     $\wedge$  integrable M f
     $\wedge$  integrable M g
     $\wedge$  g  $\in$  borel-measurable F)

```

This predicate precisely characterizes what it means for a function f to have a conditional expectation g , with respect to the measure M and the sub- σ -

algebra F .

lemma *has-cond-expI'*:

assumes $\bigwedge A. A \in \text{sets } F \implies (\int x \in A. f x \partial M) = (\int x \in A. g x \partial M)$
integrable $M f$
integrable $M g$
 $g \in \text{borel-measurable } F$
shows *has-cond-exp* $M F f g$
using *assms unfolding has-cond-exp-def* **by** *simp*

lemma *has-cond-expD*:

assumes *has-cond-exp* $M F f g$
shows $\bigwedge A. A \in \text{sets } F \implies (\int x \in A. f x \partial M) = (\int x \in A. g x \partial M)$
integrable $M f$
integrable $M g$
 $g \in \text{borel-measurable } F$
using *assms unfolding has-cond-exp-def* **by** *simp+*

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

definition *cond-exp* :: $'a \text{ measure} \Rightarrow 'a \text{ measure} \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b :: \{\text{banach, second-countable-topology}\})$ **where**

cond-exp $M F f = (\text{if } \exists g. \text{has-cond-exp } M F f g \text{ then } (\text{SOME } g. \text{has-cond-exp } M F f g) \text{ else } (\lambda-. 0))$

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

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

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

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

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

assumes $A \in \text{sets } M$

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 $\text{sets } F = \text{sets } G$

shows *has-cond-exp* $M F = \text{has-cond-exp } M G$

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

lemma *cond-exp-sets-cong*:

assumes $\text{sets } F = \text{sets } G$

shows $AE\ x\ in\ M.\ cond\text{-}exp\ M\ F\ f\ x = cond\text{-}exp\ M\ G\ f\ x$
by (*intro AE-I2, simp add: cond-exp-def has-cond-exp-sets-cong[OF assms, of M]*)

context *sigma-finite-subalgebra*
begin

lemma *borel-measurable-cond-exp'[measurable]: cond-exp M F f ∈ borel-measurable M*

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

lemma *cond-exp-null:*

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

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

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

We state the tower property of the conditional expectation in terms of the predicate *has-cond-exp*.

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

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

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

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

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

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

proof –

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

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

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

{

fix A **assume** $A \in sets\ ?MF$

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

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

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

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

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

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

corollary $cond\text{-}exp\text{-}charact$:

fixes $f :: 'a \Rightarrow 'b::\{second\text{-}countable\text{-}topology,\ banach\}$
assumes $\bigwedge A.\ A \in sets\ F \implies (\int x \in A.\ f\ x\ \partial M) = (\int x \in A.\ g\ x\ \partial M)$
 $integrable\ M\ f$
 $integrable\ M\ g$
 $g \in borel\text{-}measurable\ F$
shows $AE\ x\ in\ M.\ cond\text{-}exp\ M\ F\ f\ x = g\ x$
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 $cond\text{-}exp\ M\ F\ f = f$ μ -a.e. This is a corollary of the lemma on the characterization of $cond\text{-}exp$.

corollary $cond\text{-}exp\text{-}F\text{-meas}$ *[intro, simp]*:

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

Congruence

lemma $has\text{-}cond\text{-}exp\text{-}cong$:

assumes $integrable\ M\ f \bigwedge x.\ x \in space\ M \implies f\ x = g\ x$ $has\text{-}cond\text{-}exp\ M\ F\ g\ h$
shows $has\text{-}cond\text{-}exp\ M\ F\ f\ h$

proof *(intro has-cond-expI'[OF - assms(1)])*

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

hence $set\text{-}lebesgue\text{-}integral\ M\ A\ f = set\text{-}lebesgue\text{-}integral\ M\ A\ g$ **by** *(intro set-lebesgue-integral-cong)*
(meson assms(2) subalg in-mono subalgebra-def sets.sets-into-space subalgebra-def subsetD)+

thus $set\text{-}lebesgue\text{-}integral\ M\ A\ f = set\text{-}lebesgue\text{-}integral\ M\ A\ h$ **using** $asm\ assms(3)$

by *(simp add: has-cond-exp-def)*

qed *(auto simp add: has-cond-expD[OF assms(3)])*

lemma $cond\text{-}exp\text{-}cong$:

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

assumes $integrable\ M\ f\ integrable\ M\ g \bigwedge x.\ x \in space\ M \implies f\ x = g\ x$

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

proof *(cases $\exists h.\ has\text{-}cond\text{-}exp\ M\ F\ f\ h$)*

case $True$

then obtain h **where** $h: has\text{-}cond\text{-}exp\ M\ F\ f\ h\ has\text{-}cond\text{-}exp\ M\ F\ g\ h$ **using**
 $has\text{-}cond\text{-}exp\text{-}cong\ assms$ **by** $metis$

show $?thesis$ **using** $h[THEN\ has\text{-}cond\text{-}exp\text{-}charact(2)]$ **by** $fastforce$

next
 case *False*
 moreover have $\nexists h. \text{has-cond-exp } M F g h$ using *False has-cond-exp-cong assms*
 by *auto*
 ultimately show *?thesis unfolding cond-exp-def by auto*
qed

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

lemma *has-cond-exp-cong-AE'*:
 assumes *h ∈ 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 $\nexists h. \text{has-cond-exp } M F g h$ using *False has-cond-exp-cong-AE assms* by *auto*
 ultimately show *?thesis unfolding cond-exp-def by auto*
qed

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

lemma *has-cond-exp-real*:
 fixes *f :: 'a ⇒ real*
 assumes *integrable M f*
 shows *has-cond-exp M F f (real-cond-exp M F f)*
 by (*intro has-cond-expI', auto intro!: real-cond-exp-intA assms*)

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

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

4.1 Existence

Showing the existence is a bit involved. Specifically, what we aim to show is that *has-cond-exp* $M\ F\ f$ (*cond-exp* $M\ F\ f$) holds for any Bochner-integrable f . We will employ the standard machinery of measure theory. First, we will prove existence for indicator functions. Then we will extend our proof by linearity to simple functions. Finally we use a limiting argument to show that the conditional expectation exists for all Bochner-integrable functions.

Indicator functions

lemma *has-cond-exp-indicator*:
assumes $A \in \text{sets}\ M\ \text{emeasure}\ M\ A < \infty$
shows $\text{has-cond-exp}\ M\ F\ (\lambda x. \text{indicat-real}\ A\ x *_{\mathbb{R}}\ y)\ (\lambda x. \text{real-cond-exp}\ M\ F\ (\text{indicator}\ A)\ x *_{\mathbb{R}}\ y)$
proof (*intro* *has-cond-expI'*, *goal-cases*)
case ($1\ B$)
have $\int_{x \in B}. (\text{indicat-real}\ A\ x *_{\mathbb{R}}\ y)\ \partial M = (\int_{x \in B}. \text{indicat-real}\ A\ x\ \partial M) *_{\mathbb{R}}\ y$ **using** *assms* **by** (*intro* *set-integral-scaleR-left*, *meson* 1 *in-mono* *subalg* *subalgebra-def*, *blast*)
also **have** $\dots = (\int_{x \in B}. \text{real-cond-exp}\ M\ F\ (\text{indicator}\ A)\ x\ \partial M) *_{\mathbb{R}}\ y$ **using** 1 *assms* **by** (*subst* *real-cond-exp-intA*, *auto*)
also **have** $\dots = \int_{x \in B}. (\text{real-cond-exp}\ M\ F\ (\text{indicator}\ A)\ x *_{\mathbb{R}}\ y)\ \partial M$ **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** *fast-force*

qed

Addition

lemma *has-cond-exp-add*:

fixes $f g :: 'a \Rightarrow 'b :: \{second-countable-topology, banach\}$
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** $\dots = (\int_{x \in A. f'$ $x \partial M) + (\int_{x \in A. g'$ $x \partial M)$ **using** *assms*[*THEN*
has-cond-expD(*1*)[*OF* - *1*]] **by** *argo*

also **have** $\dots = \int_{x \in A. (f'$ $x + g'$ $x) \partial M$ **using** *assms*[*THEN* *has-cond-expD*(*3*)]
subalg *1* **by** (*intro* *set-integral-add*(*2*)[*symmetric*], *auto* *simp* *add: subalgebra-def*
set-integrable-def *intro: integrable-mult-indicator*)

finally **show** *?case* .

next

case *2*

show *?case* **by** (*metis* *Bochner-Integration.integrable-add* *assms* *has-cond-expD*(*2*))

next

case *3*

show *?case* **by** (*metis* *Bochner-Integration.integrable-add* *assms* *has-cond-expD*(*3*))

next

case *4*

show *?case* **using** *assms* *borel-measurable-add* *has-cond-expD*(*4*) **by** *blast*

qed

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

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

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

shows *has-cond-exp* M F $(\lambda x. c *_R$ f $x)$ $(\lambda x. c *_R$ f' $x)$

using *has-cond-expD*[*OF assms*] **by** (*intro* *has-cond-expI'*, *auto*)

```

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

```

```

lemma cond-exp-uminus:
  fixes  $f :: 'a \Rightarrow 'b::\{\text{second-countable-topology}, \text{banach}\}$ 
  assumes integrable  $M f$ 
  shows  $\text{AE } x \text{ in } M. \text{cond-exp } M F (\lambda x. - f x) x = - \text{cond-exp } M F f x$ 
  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  $f :: 'a \Rightarrow 'b::\{\text{second-countable-topology}, \text{banach}\}$ 
  assumes simple-function  $M f$  emeasure  $M \{y \in \text{space } M. f y \neq 0\} \neq \infty$ 
  shows  $\text{has-cond-exp } M F f (\text{cond-exp } M F f)$ 
  using assms
proof (induction rule: integrable-simple-function-induct)
  case (cong  $f g$ )
    then show ?case using has-cond-exp-cong by (metis (no-types, opaque-lifting) Bochner-Integration.integrable-cong has-cond-expD(2) has-cond-exp-charact(1))
  next
    case (indicator  $A y$ )
    then show ?case using has-cond-exp-charact[OF has-cond-exp-indicator] by fast

```

```

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

```

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

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

lemma *cond-exp-contraction-real*:

```

fixes f :: 'a  $\Rightarrow$  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  $\implies$   $\int x \in A. |f x| \partial M = \int x \in A. \text{real-cond-exp } M F (\lambda x.
\text{norm } (f x)) x \partial M$  for A unfolding real-norm-def using assms integrable-abs
real-cond-exp-intA by blast

```

```

have norm-int: A  $\in$  sets F  $\implies$  ( $\int x \in A. |f x| \partial M$ ) = ( $\int ^+ x \in A. |f x| \partial M$ ) for A
using assms by (intro nn-set-integral-eq-set-integral[symmetric], blast, fastforce)
(meson subalg subalgebra-def subsetD)

```

```

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

```

```

hence cond-exp-norm-int: A  $\in$  sets F  $\implies$  ( $\int x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm}
(f x)) x \partial M$ ) = ( $\int ^+ x \in A. \text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x \partial M$ ) for A using
assms by (intro nn-set-integral-eq-set-integral[symmetric], blast, fastforce) (meson
subalg subalgebra-def subsetD)

```

```

have A  $\in$  sets F  $\implies$   $\int ^+ x \in A. |f x| \partial M = \int ^+ x \in A. \text{real-cond-exp } M F (\lambda x.
\text{norm } (f x)) x \partial M$  for A using ** norm-int cond-exp-norm-int by (auto simp
add: nn-integral-set-ennreal)

```

```

moreover have ( $\lambda x. \text{ennreal } |f x|$ )  $\in$  borel-measurable M by measurable
moreover have ( $\lambda x. \text{ennreal } (\text{real-cond-exp } M F (\lambda x. \text{norm } (f x)) x)$ )  $\in$  borel-measurable
F by measurable

```

ultimately have $AE\ x\ in\ M. nn\text{-}cond\text{-}exp\ M\ F\ (\lambda x. ennreal\ |f\ x|)\ x = real\text{-}cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x$ **by** (*intro nn-cond-exp-charact*[*THEN AE-symmetric*], *auto*)
hence $AE\ x\ in\ M. nn\text{-}cond\text{-}exp\ M\ F\ (\lambda x. ennreal\ |f\ x|)\ x \leq cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x$ **using** *cond-exp-real*[*OF int*] **by** *force*
moreover have $AE\ x\ in\ M. |real\text{-}cond\text{-}exp\ M\ F\ f\ x| = norm\ (cond\text{-}exp\ M\ F\ f\ x)$
unfolding *real-norm-def* **using** *cond-exp-real*[*OF assms*] *** by** *force*
ultimately have $AE\ x\ in\ M. ennreal\ (norm\ (cond\text{-}exp\ M\ F\ f\ x)) \leq cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x$ **using** *real-cond-exp-abs*[*OF assms*[*THEN borel-measurable-integrable*]] **by** *fastforce*
hence $AE\ x\ in\ M. enn2real\ (ennreal\ (norm\ (cond\text{-}exp\ M\ F\ f\ x))) \leq enn2real\ (cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x)$ **using** *ennreal-le-iff2* **by** *force*
thus *?thesis* **using** ***** **by** *fastforce*
qed

lemma *cond-exp-contraction-simple*:

fixes $f :: 'a \Rightarrow 'b :: \{second\text{-}countable\text{-}topology, banach\}$
assumes *simple-function* $M\ f\ emeasure\ M\ \{y \in space\ M. f\ y \neq 0\} \neq \infty$
shows $AE\ x\ in\ M. norm\ (cond\text{-}exp\ M\ F\ f\ x) \leq cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x$
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 $AE\ x\ in\ M. cond\text{-}exp\ M\ F\ f\ x = cond\text{-}exp\ M\ F\ g\ x$ **using** *cong has-cond-exp-simple*
by (*subst cond-exp-cong-AE*) (*auto intro!*: *has-cond-expD*(2))
hence $AE\ x\ in\ M. norm\ (cond\text{-}exp\ M\ F\ f\ x) = norm\ (cond\text{-}exp\ M\ F\ g\ x)$ **by** *force*
moreover have $AE\ x\ in\ M. cond\text{-}exp\ M\ F\ (\lambda x. norm\ (f\ x))\ x = cond\text{-}exp\ M\ F\ (\lambda x. norm\ (g\ x))\ x$ **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 $AE\ x\ in\ M. cond\text{-}exp\ M\ F\ (\lambda a. indicator\ A\ a\ *_R\ y)\ x = cond\text{-}exp\ M\ F\ (indicator\ A)\ x\ *_R\ y$ **by** *blast*
hence $*$: $AE\ x\ in\ M. norm\ (cond\text{-}exp\ M\ F\ (\lambda a. indicat\text{-}real\ A\ a\ *_R\ y)\ x) \leq norm\ y\ * cond\text{-}exp\ M\ F\ (\lambda x. norm\ (indicat\text{-}real\ A\ x))\ x$ **using** *cond-exp-contraction-real*[*OF integrable-real-indicator, OF indicator*] **by** *fastforce*
have $AE\ x\ in\ M. norm\ y\ * cond\text{-}exp\ M\ F\ (\lambda x. norm\ (indicat\text{-}real\ A\ x))\ x = norm\ y\ * real\text{-}cond\text{-}exp\ M\ F\ (\lambda x. norm\ (indicat\text{-}real\ A\ x))\ x$ **using** *cond-exp-real*[*OF integrable-real-indicator, OF indicator*] **by** *fastforce*
moreover have $AE\ x\ in\ M. cond\text{-}exp\ M\ F\ (\lambda x. norm\ y\ * norm\ (indicat\text{-}real\ A\ x))\ x = real\text{-}cond\text{-}exp\ M\ F\ (\lambda x. norm\ y\ * norm\ (indicat\text{-}real\ A\ x))\ x$ **using** *indicator* **by** (*intro cond-exp-real, auto*)
ultimately have $AE\ x\ in\ M. norm\ y\ * cond\text{-}exp\ M\ F\ (\lambda x. norm\ (indicat\text{-}real\ A\ x))\ x = cond\text{-}exp\ M\ F\ (\lambda x. norm\ y\ * norm\ (indicat\text{-}real\ A\ x))\ x$ **using** *real-cond-exp-cmult*[*of* $\lambda x. norm\ (indicat\text{-}real\ A\ x)\ norm\ y$] *indicator* **by** *fastforce*
moreover have $(\lambda x. norm\ y\ * norm\ (indicat\text{-}real\ A\ x)) = (\lambda x. norm\ (indicat\text{-}real\ A\ x))$

$A x *_R y$) **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) ≤ norm (cond-exp M F u x) + norm (cond-exp M F v x) using norm-triangle-ineq by blast*
moreover have *AE x in M. norm (cond-exp M F u x) + norm (cond-exp M F v x) ≤ cond-exp M F (λx. norm (u x)) x + cond-exp M F (λx. norm (v x)) x using add(6,7) by fastforce*
moreover have *AE x in M. cond-exp M F (λx. norm (u x)) x + cond-exp M F (λx. norm (v x)) x = cond-exp M F (λx. norm (u x) + norm (v x)) x using 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 $\bigwedge i. \text{simple-function } M (s\ i)$
and $\bigwedge i. \text{emeasure } M \{y \in \text{space } M. s\ i\ y \neq 0\} \neq \infty$
and $\bigwedge x. x \in \text{space } M \implies (\lambda i. s\ i\ x) \longrightarrow f\ x$
and $\bigwedge x\ i. x \in \text{space } M \implies \text{norm } (s\ i\ x) \leq 2 * \text{norm } (f\ x)$
obtains r
where *strict-mono r has-cond-exp M F f (λx. lim (λi. cond-exp M F (s (r i)) x))*
 $AE\ x\ \text{in } M. \text{convergent } (\lambda i. \text{cond-exp } M\ F\ (s\ (r\ i))\ x)$

proof –

have *[measurable]: (s i) ∈ borel-measurable M for i using assms(2) by (simp add: borel-measurable-simple-function)*
have *integrable-s: integrable M (λx. s i x) for i using assms integrable-simple-function by blast*
have *integrable-4f: integrable M (λx. 4 * norm (f x)) using assms(1) by simp*
have *integrable-2f: integrable M (λx. 2 * norm (f x)) using assms(1) by simp*
have *integrable-2-cond-exp-norm-f: integrable M (λx. 2 * cond-exp M F (λx. 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*: $\text{emeasure } M \{y \in \text{space } M. s \ i \ y - s \ j \ y \neq 0\} \neq \infty$ **for** $i \ j$ **using** *assms(3)* **by** (*metis (mono-tags) ennreal-add-eq-top linorder-not-less top.not-eq-extremum infinity-ennreal-def*)

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

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

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

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

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

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

have $AE \ x \ \text{in } M. (\lambda n. \text{diameter } \{s \ i \ x \mid i. n \leq i\}) \longrightarrow 0$ **using** *Cauchy cauchy-iff-diameter-tends-to-zero-and-bounded* **by** *fast*

moreover **have** $(\lambda x. \text{diameter } \{s \ i \ x \mid i. n \leq i\}) \in \text{borel-measurable } M$ **for** n **using** *bounded-range-s borel-measurable-diameter* **by** *measurable*

moreover **have** $AE \ x \ \text{in } M. \text{norm } (\text{diameter } \{s \ i \ x \mid i. n \leq i\}) \leq 4 * \text{norm } (f \ x)$ **for** n

proof –

{

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

have $\text{diameter } \{s \ i \ x \mid i. n \leq i\} \leq 2 * \text{norm } (f \ x) + 2 * \text{norm } (f \ x)$ **by** (*intro diameter-le, blast, subst dist-norm[symmetric], intro dist-triangle3[THEN order-trans, of 0], intro add-mono*) (*auto intro: assms(5)[OF x]*)

hence $\text{norm } (\text{diameter } \{s \ i \ x \mid i. n \leq i\}) \leq 4 * \text{norm } (f \ x)$ **using** *diameter-ge-0[OF bounded-subset[OF bounded-range-s], OF x, of {s i x | i. n ≤ i}]* **by** *force*

}

thus *?thesis* **by** *fast*

qed

ultimately **have** *diameter-tendsto-zero*: $(\lambda n. \text{LINT } x \mid M. \text{diameter } \{s \ i \ x \mid i. n \leq i\}) \longrightarrow 0$ **by** (*intro integral-dominated-convergence[OF borel-measurable-const[of 0] - integrable-4f, simplified]*) (*fast+*)

have *diameter-integrable*: *integrable* $M (\lambda x. \text{diameter } \{s \ i \ x \mid i. n \leq i\})$ **for** n **using** *assms(1,5)*

by (*intro integrable-bound-diameter[OF bounded-range-s integrable-2f], auto*)

have *dist-integrable*: *integrable* $M (\lambda x. \text{dist } (s \ i \ x) (s \ j \ x))$ **for** $i \ j$ **using** *assms(5)*

dist-triangle3[of $s \ i \ - \ 0$, *THEN order-trans*, *OF add-mono*, of $- \ 2 \ * \ norm \ (f \ -)$]
by (*intro Bochner-Integration.integrable-bound*[*OF integrable-4f*]) *fastforce+*

Since *cond-exp M F* is a contraction for simple functions, the following sequence of integral values is also Cauchy.

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

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

have $\exists N. \forall i \geq N. \forall j \geq N. \ LINT \ x|M. \ norm \ (cond\text{-}exp \ M \ F \ (s \ i) \ x \ - \ cond\text{-}exp \ M \ F \ (s \ j) \ x) < e$ **if** *e-pos*: $e > 0$ **for** *e*

proof $-$

obtain *N* **where** $*$: *LINT x|M. diameter* $\{s \ i \ x \mid i. n \leq i\} < e$ **if** $n \geq N$ **for** *n* **using** *that order-tendsto-iff*[*THEN iffD1*, *OF diameter-tendsto-zero*, *unfolded eventually-sequentially*] *e-pos* **by** *presburger*

{

fix *i j x* **assume** *asm*: $i \geq N \ j \geq N \ x \in \text{space } M$

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

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

moreover **have** $(SUP \ (i, j) \in \{N..\} \times \{N..\}. \text{dist } (s \ i \ x) \ (s \ j \ x)) \geq \text{dist } (s \ i \ x) \ (s \ j \ x)$ **using** *asm bounded-imp-bdd-above*[*OF bounded-imp-dist-bounded*, *OF bounded-range-s*] **by** (*intro cSup-upper*, *auto*)

ultimately **have** *diameter* $\{s \ i \ x \mid i. N \leq i\} \geq \text{dist } (s \ i \ x) \ (s \ j \ x)$ **by** *presburger*

}

hence *LINT x|M. dist* $(s \ i \ x) \ (s \ j \ x) < e$ **if** $i \geq N \ j \geq N$ **for** *i j* **using** *that ** **by** (*intro integral-mono*[*OF dist-integrable diameter-integrable*, *THEN order.strict-trans1*], *blast+*)

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

proof $-$

have *LINT x|M. norm* $(cond\text{-}exp \ M \ F \ (s \ i) \ x \ - \ cond\text{-}exp \ M \ F \ (s \ j) \ x) = \ LINT \ x|M. \ \text{norm } (cond\text{-}exp \ M \ F \ (s \ i) \ x \ + \ - \ 1 \ *R \ cond\text{-}exp \ M \ F \ (s \ j) \ x)$ **unfolding** *dist-norm* **by** *simp*

also **have** $\dots = \ LINT \ x|M. \ \text{norm } (cond\text{-}exp \ M \ F \ (\lambda x. \ s \ i \ x \ - \ s \ j \ x) \ x)$ **using** *has-cond-exp-charact(2)*[*OF has-cond-exp-add*[*OF - has-cond-exp-scaleR-right*, *OF has-cond-exp-charact(1,1)*, *OF has-cond-exp-simple(1,1)*][*OF assms(2,3)*]], *THEN AE-symmetric*, of $i - 1 \ j$] **by** (*intro integral-cong-AE*) *force+*

also **have** $\dots \leq \ LINT \ x|M. \ \text{cond-exp } M \ F \ (\lambda x. \ \text{norm } (s \ i \ x \ - \ s \ j \ x)) \ x$ **using** *cond-exp-contraction-simple*[*OF - fin-sup*, of $i \ j$] *integrable-cond-exp assms(2)* **by** (*intro integral-mono-AE*, *fast+*)

also **have** $\dots = \ LINT \ x|M. \ \text{norm } (s \ i \ x \ - \ s \ j \ x)$ **unfolding** *set-integral-space(1)*[*OF integrable-cond-exp*, *symmetric*] *set-integral-space*[*OF dist-integrable*[*unfolded dist-norm*], *symmetric*] **by** (*intro has-cond-expD(1)*[*OF has-cond-exp-simple*[*OF - fin-sup-norm*], *symmetric*]) (*metis assms(2) simple-function-compose1 simple-function-diff*, *metis sets.top subalg subalgebra-def*)

finally show *?thesis unfolding dist-norm* .
qed
ultimately show *?thesis using order.strict-trans1 by meson*
qed
then obtain *r where strict-mono-r: strict-mono r and AE-Cauchy: AE x in M. Cauchy (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 expectation property holds, i.e. the value of the integral of the candidate, agrees with f on sets $A \in \text{sets } F$.

{
fix *A assume A-in-sets-F: A ∈ 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 $AE\ x\ in\ M.\ norm\ (indicator\ A\ x\ *_R\ s\ (r\ n)\ x)\ \leq\ 2\ *\ norm\ (f\ x)$ **for** n
proof –
have $AE\ x\ in\ M.\ norm\ (indicator\ A\ x\ *_R\ s\ (r\ n)\ x)\ \leq\ norm\ (s\ (r\ n)\ x)$
unfolding *indicator-def* **by** *simp*
thus *?thesis* **using** *assms(5)[of - r n]* **by** *fastforce*
qed
hence $lim\text{-}s\text{-}int:\ (\lambda n.\ LINT\ x:A|M.\ s\ (r\ n)\ x)\ \longrightarrow\ LINT\ x:A|M.\ f\ x$
using *measurable-from-subalg*[*OF subalg borel-measurable-indicator, OF A-in-sets-F*]
LIMSEQ-subseq-LIMSEQ[*OF assms(4) strict-mono-r*] *assms(5)*
unfolding *set-lebesgue-integral-def comp-def*
by (*intro integral-dominated-convergence*[*OF borel-measurable-scaleR borel-measurable-scaleR integrable-2f*]) (*fastforce simp add: tendsto-scaleR*)
have $LINT\ x:A|M.\ lim\ (\lambda n.\ cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)\ =\ lim\ (\lambda n.\ LINT\ x:A|M.\ cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)$ **using** *limI*[*OF lim-cond-exp-int*] **by** *argo*
also **have** $\dots = lim\ (\lambda n.\ LINT\ x:A|M.\ s\ (r\ n)\ x)$ **using** *has-cond-expD(1)*[*OF has-cond-exp-simple*][*OF assms(2,3)*] *A-in-sets-F, symmetric*] **by** *presburger*
also **have** $\dots = LINT\ x:A|M.\ f\ x$ **using** *limI*[*OF lim-s-int*] **by** *argo*
finally **have** $LINT\ x:A|M.\ lim\ (\lambda n.\ cond\text{-}exp\ M\ F\ (s\ (r\ n))\ x)\ =\ LINT\ x:A|M.\ f\ x$.
}

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

hence $has\text{-}cond\text{-}exp\ M\ F\ f\ (\lambda x.\ lim\ (\lambda i.\ cond\text{-}exp\ M\ F\ (s\ (r\ i))\ x))$ **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 $f :: 'a \Rightarrow 'b :: \{second\text{-}countable\text{-}topology, banach\}$

assumes *integrable M f*

shows $has\text{-}cond\text{-}exp\ M\ F\ f\ (cond\text{-}exp\ M\ F\ f)$

proof –

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

show *?thesis* **using** *has-cond-exp-simple-lim*[*OF assms s-is*] *has-cond-exp-charact(1)*

by *metis*

qed

4.2 Properties

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

lemma *cond-exp-set-integral*:

fixes $f :: 'a \Rightarrow 'b :: \{second-countable-topology, banach\}$
assumes *integrable M f A ∈ sets F*
shows $(\int x \in A. f x \partial M) = (\int x \in A. cond-exp M F f x \partial M)$
using *has-cond-expD(1)[OF has-cond-expI, OF assms]* **by** *argo*

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

lemma *cond-exp-nested-subalg*:

fixes $f :: 'a \Rightarrow 'b :: \{second-countable-topology, banach\}$
assumes *integrable M f subalgebra M G subalgebra G F*
shows *AE 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 (λx. 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 (λx. 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 (λx. f x *_R c) x = cond-exp M F f x *_R c*

using *cond-exp-set-integral*[*OF assms*] *subalg assms* **unfolding** *subalgebra-def*
by (*intro cond-exp-charact*,
subst set-integral-scaleR-left, blast, intro assms,
subst set-integral-scaleR-left, blast, intro integrable-cond-exp)
auto

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

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

lemma *cond-exp-contraction*:

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

assumes *integrable M f*

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

proof –

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

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

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

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

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

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

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

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

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

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

ultimately show *?thesis* **using** $LIMSEQ\text{-}le\ r'\text{-tendsto}$ **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" [5].

lemma *cond-exp-measurable-mult*:

fixes $f\ g :: 'a \Rightarrow real$

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

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

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

proof –

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

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

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

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

qed

lemma *cond-exp-measurable-scaleR*:

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

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

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

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

proof –

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

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

{

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

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

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

proof –

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

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

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

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

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

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

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

{

fix i

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

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

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

subalg]], *auto*)

ultimately have *AE* *x* in *?F*. *norm* (*s i x *_R cond-exp M F g x*) $\leq 2 * \text{cond-exp } M F (\lambda x. \text{norm } (z x *_{R} g x)) x$ **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* = $(\sum_{y \in s \text{ 'space } M. (\text{indicator } (s \text{ ' } \{y\} \cap \text{space } M) x *_{R} y) *_{R} h x)}$ **if** $x \in \text{space } M$ **for** *x* **and** *h* :: '*a* \Rightarrow '*b*
using *simple-function-indicator-representation*[*OF s-is(1)*, *of x i*] **that unfolding space-restr-to-subalg scaleR-left.sum[*of - - h x, symmetric*] **by** *presburger***

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

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

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

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

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

also have ... = *LINT x|M. (* $\sum_{y \in s \text{ 'space } M. (\text{indicator } (s \text{ ' } \{y\} \cap \text{space } M) x *_{R} y *_{R} \text{cond-exp } M F g)}$ *)* **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* ($\lambda x. z x *_{R} \text{cond-exp } M F g x$) **using** *s-scaleR-cond-exp-g-meas*

asm(2) *borel-measurable-cond-exp'* **by** (*intro integrable-from-subalg*[*OF subalg*] *integrable-cond-exp integrable-dominated-convergence*[*OF - - - s-scaleR-cond-exp-g-tendsto s-scaleR-cond-exp-g-AE-bdd*]) (*auto intro: measurable-from-subalg*[*OF subalg*] *integrable-in-subalg measurable-in-subalg subalg*)

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

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

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

qed
}
note $*$ = *this*

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

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

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

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

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

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

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

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

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

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

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

unfolding *set-lebesgue-integral-def* **by** (*simp add: scaleR-sum-right*)
also have $\dots = (\sum_{i \in I}. (\int x. \text{indicator } A \ x \ *_{\mathbb{R}} \ f \ i \ x \ \partial M))$ **using** *assms* **by** (*auto intro!: Bochner-Integration.integral-sum integrable-mult-indicator*)
also have $\dots = (\sum_{i \in I}. (\int x. \text{indicator } A \ x \ *_{\mathbb{R}} \ \text{cond-exp } M \ F \ (f \ i) \ x \ \partial M))$ **using** *cond-exp-set-integral[OF assms]* **by** (*simp add: set-lebesgue-integral-def*)
also have $\dots = (\int x. (\sum_{i \in I}. \text{indicator } A \ x \ *_{\mathbb{R}} \ \text{cond-exp } M \ F \ (f \ i) \ x) \ \partial M)$ **using** *assms* **by** (*auto intro!: Bochner-Integration.integral-sum[symmetric] integrable-mult-indicator*)
also have $\dots = (\int x \in A. (\sum_{i \in I}. \text{cond-exp } M \ F \ (f \ i) \ x) \ \partial M)$ **unfolding** *set-lebesgue-integral-def* **by** (*simp add: scaleR-sum-right*)
finally show $(\int x \in A. (\sum_{i \in I}. f \ i \ x) \ \partial M) = (\int x \in A. (\sum_{i \in I}. \text{cond-exp } M \ F \ (f \ i) \ x) \ \partial M)$ **by** *auto*
qed (*auto simp add: assms integrable-cond-exp*)

4.3 Linearly Ordered Banach Spaces

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

lemma *cond-exp-gr-c*:

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

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

shows *AE x in M. cond-exp M F f x > c*

proof –

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

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

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

have *emeasure M X = 0*

proof (*rule ccontr*)

assume *emeasure M X \neq 0*

have *emeasure (restr-to-subalg M F) X = emeasure M X* **by** (*simp add: emeasure-restr-to-subalg subalg*)

hence *emeasure (restr-to-subalg M F) X > 0* **using** $\neg(\text{emeasure } M \ X) = 0$ *gr-zeroI* **by** *auto*

then obtain A **where** $A: A \in \text{sets } (\text{restr-to-subalg } M \ F) \ A \subseteq X$ *emeasure (restr-to-subalg M F) A > 0* *emeasure (restr-to-subalg M F) A < \infty*

using *sigma-fin-subalg* **by** (*metis emeasure-notin-sets ennreal-0 infinity-ennreal-def le-less-linear neq-top-trans not-gr-zero order-refl sigma-finite-measure.approx-PInf-emeasure-with-finite*)

hence [*simp*]: $A \in \text{sets } F$ **using** *subalg sets-restr-to-subalg* **by** *blast*

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

have [*simp*]: *set-integrable M A* $(\lambda x. c)$ **using** A *subalg* **by** (*auto simp add: set-integrable-def emeasure-restr-to-subalg*)

have [*simp*]: *set-integrable M A f* **unfolding** *set-integrable-def* **by** (*rule integrable-mult-indicator, auto simp add: assms(1)*)

have *AE x in M. indicator A x *_R c = indicator A x *_R f x*

proof (*rule integral-eq-mono-AE-eq-AE*)

have $(\int x \in A. c \partial M) \leq (\int x \in A. f x \partial M)$ **using** *assms(2)* **by** (*intro set-integral-mono-AE-banach*)
auto
moreover
{
have $(\int x \in A. f x \partial M) = (\int x \in A. \text{cond-exp } M F f x \partial M)$ **by** (*rule*
cond-exp-set-integral, auto simp add: assms)
also have $\dots \leq (\int x \in A. c \partial M)$ **using** *A* **by** (*auto intro!: set-integral-mono-banach*
simp add: X-def)
finally have $(\int x \in A. f x \partial M) \leq (\int x \in A. c \partial M)$ **by** *simp*
}
ultimately show $LINT x|M. \text{indicator } A x *_R c = LINT x|M. \text{indicator } A$
 $x *_R f x$ **unfolding** *set-lebesgue-integral-def* **by** *simp*
show $AE x \text{ in } M. \text{indicator } A x *_R c \leq \text{indicator } A x *_R f x$ **using** *assms* **by**
(*auto simp add: X-def indicator-def*)
qed (*auto simp add: set-integrable-def[symmetric]*)
hence $AE x \in A \text{ in } M. c = f x$ **by** *auto*
hence $AE x \in A \text{ in } M. \text{False}$ **using** *assms(2)* **by** *auto*
hence $A \in \text{null-sets } M$ **using** *AE-iff-null-sets A-in-sets-M* **by** *metis*
thus False **using** *A(3)* **by** (*simp add: emeasure-restr-to-subalg null-setsD1*
subalg)
qed
thus *?thesis* **using** *AE-iff-null-sets[OF X-in-M]* **unfolding** *X-def* **by** *auto*
qed

corollary *cond-exp-less-c*:

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

lemma *cond-exp-mono-strict*:

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

lemma *cond-exp-ge-c*:

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

assumes [*measurable*]: *integrable* $M f$
and $AE x \text{ in } M. f x \geq c$
shows $AE x \text{ in } M. \text{cond-exp } M F f x \geq c$
proof –
let $?F = \text{restr-to-subalg } M F$
interpret *sigma-finite-measure* *restr-to-subalg* $M F$ **using** *sigma-fin-subalg* **by**
auto
{
fix A **assume** *asm*: $A \in \text{sets } ?F \ 0 < \text{measure } ?F A$
have [*simp*]: $\text{sets } ?F = \text{sets } F \ \text{measure } ?F A = \text{measure } M A$ **using** *asm* **by** (*auto*
simp add: measure-def sets-restr-to-subalg[OF subalg] emeasure-restr-to-subalg[OF
subalg])
have $M\text{-}A: \text{emeasure } M A < \infty$ **using** *measure-zero-top asm* **by** (*force simp*
add: top.not-eq-extremum)
hence $F\text{-}A: \text{emeasure } ?F A < \infty$ **using** *asm(1) emeasure-restr-to-subalg subalg*
by *fastforce*
have $\text{set-lebesgue-integral } M A (\lambda\cdot. c) \leq \text{set-lebesgue-integral } M A f$ **using**
assms asm M-A subalg **by** (*intro set-integral-mono-AE-banach, auto simp add:*
set-integrable-def integrable-mult-indicator subalgebra-def sets-restr-to-subalg)
also have $\dots = \text{set-lebesgue-integral } M A (\text{cond-exp } M F f)$ **using** *cond-exp-set-integral[OF*
assms(1)] asm **by** *auto*
also have $\dots = \text{set-lebesgue-integral } ?F A (\text{cond-exp } M F f)$ **unfolding** *set-lebesgue-integral-def*
using *asm borel-measurable-cond-exp* **by** (*intro integral-subalgebra2[OF subalg, sym-*
metric], simp)
finally have $(1 / \text{measure } ?F A) *_R \text{set-lebesgue-integral } ?F A (\text{cond-exp } M F f)$
 $\in \{c..\}$ **using** *asm subalg M-A* **by** (*auto simp add: set-integral-const subalgebra-def*
intro!: pos-divideR-le-eq[THEN iffD1])
}
thus *?thesis* **using** *AE-restr-to-subalg[OF subalg] averaging-theorem[OF inte-*
grable-in-subalg closed-atLeast, OF subalg borel-measurable-cond-exp integrable-cond-exp]
by *auto*
qed

corollary *cond-exp-le-c*:

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

corollary *cond-exp-mono*:

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

dered-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-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 ⇒ 'b :: {second-countable-topology, banach, linorder-topology, ordered-real-vector}*
assumes *integrable M f integrable M g*
shows *AE ξ in M. cond-exp M F (λx. min (f x) (g x)) ξ ≤ min (cond-exp M F f ξ) (cond-exp M F g ξ)*
proof –
have *AE ξ in M. cond-exp M F (λx. min (f x) (g x)) ξ ≤ 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 (λx. min (f x) (g x)) ξ ≤ 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 (λx. min (f x) (g x)) ξ ≤ min (cond-exp M F f ξ) (cond-exp M F g ξ) by fastforce*
qed

corollary *cond-exp-max:*

fixes *f :: 'a ⇒ 'b :: {second-countable-topology, banach, linorder-topology, ordered-real-vector}*
assumes *integrable M f integrable M g*
shows *AE ξ in M. cond-exp M F (λx. max (f x) (g x)) ξ ≥ max (cond-exp M F f ξ) (cond-exp M F g ξ)*
proof –
have *AE ξ in M. cond-exp M F (λx. max (f x) (g x)) ξ ≥ 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 (λx. max (f x) (g x)) ξ ≥ 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 (λx. max (f x) (g x)) ξ ≥ 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 (λx. 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 $AE \xi$ in M . $cond-exp M F (\lambda x. sup (f x) (g x)) \xi \geq sup (cond-exp M F f \xi) (cond-exp M F g \xi)$

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

end

4.4 Probability Spaces

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

assumes $subalgebra M F$

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

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

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

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

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

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

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 [5].

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

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

assumes $subalgebra: subalgebra M F subalgebra M G$

and $independent: indep-set G (sigma (space M) (F \cup vimage-algebra (space M) f borel))$

assumes $[measurable]: integrable M f$

shows $AE x$ in M . $cond-exp M (sigma (space M) (F \cup G)) f x = cond-exp M F f x$

proof –

interpret $Un-sigma: sigma$ -finite-subalgebra $M sigma (space M) (F \cup G)$ **using** $assms(1,2)$ **by** $(auto intro!: sigma$ -finite-subalgebra-restr-to-subalg $sets.sigma$ -sets-subset
 $simp add: subalgebra-def space-measure-of-conv sets-measure-of-conv)$

interpret $sigma$ -finite-subalgebra $M F$ **using** $assms$ **by** $(auto intro: sigma$ -finite-subalgebra-restr-to-subalg
 $\{$

$fix A$

$assume asm: A \in sigma (space M) \{a \cap b \mid a \in F \wedge b \in G\}$

$have in-events: sigma$ -sets $(space M) \{a \cap b \mid a \in sets F \wedge b \in sets G\} \subseteq events$ **using** $subalgebra$ **by** $(intro sets.sigma$ -sets-subset, $auto simp add: subalgebra-def)$

have *Int-stable* $\{a \cap b \mid a \in F \wedge b \in G\}$
proof –
{
 fix *af bf ag bg*
 assume *F: af ∈ 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 $\exists a \ b. \ af \cap \ ag \cap \ (bf \cap \ bg) = a \cap \ b \wedge a \in \ sets \ F \wedge b \in \ sets \ G$ **by** (*metis inf-assoc inf-left-commute*)
}
thus *?thesis by (force intro!: Int-stableI)*
qed
 moreover have $\{a \cap b \mid a \in F \wedge b \in G\} \subseteq Pow \ (space \ M)$ **using** *subalgebra by (force simp add: subalgebra-def dest: sets.sets-into-space)*
 moreover have $A \in \ sigma\text{-sets} \ (space \ M) \ \{a \cap b \mid a \in F \wedge b \in G\}$ **using** *calculation asm by force*
 ultimately have *set-lebesgue-integral M A f = set-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 *: $(\lambda\omega. \ indicator \ a \ \omega \ *_{R} \ f \ \omega) \in \ borel\text{-measurable} \ (sigma \ (space \ M) \ (F \cup \ vimage\text{-algebra} \ (space \ M) \ f \ borel))$
 using *borel-measurable-indicator[OF A(2), THEN measurable-sets, OF borel-open] subalgebra*
 by (*intro borel-measurable-scaleR borel-measurableI Un-in-sigma[THEN subsetD]*)
 (*auto simp add: space-measure-of-conv subalgebra-def sets-vimage-algebra2*)
 thus *?thesis using measurable-sets[OF *] by (intro sets.sigma-sets-subset', auto simp add: space-measure-of-conv)*

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

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

have *indep-set* (*sigma-sets* (*space M*) {*indicator b* - ‘*A* \cap *space M* | *A*. *A* \in *borel*}) (*sigma-sets* (*space M*) {($\lambda\omega$. *indicator a* $\omega *_{\mathbb{R}}$ *cond-exp M F f* ω) - ‘*A* \cap *space M* | *A*. *A* \in *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* $*_{\mathbb{R}}$ *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* $*_{\mathbb{R}}$ *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* $*_{\mathbb{R}}$

```

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.integrable-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 $\sigma(\text{space } M) \{a \cap b \mid a \in F \wedge b \in G\} = \sigma(\text{space } M) (F \cup G)$

proof -

have $\sigma\text{-sets}(\text{space } M) \{a \cap b \mid a \in \text{sets } F \wedge b \in \text{sets } G\} = \sigma\text{-sets}(\text{space } M) (\text{sets } F \cup \text{sets } G)$

proof -

{

fix $a \ b$ assume $asm: a \in F \ b \in G$

hence $a \cap b \in \sigma\text{-sets}(\text{space } M) (F \cup G)$ using subalgebra unfolding Int-range-binary by (intro sigma-sets-Inter[OF - binary-in-sigma-sets]) (force simp add: subalgebra-def dest: sets.sets-into-space)+

}

moreover

{

fix a

assume $a \in \text{sets } F$

hence $a \in \sigma\text{-sets}(\text{space } M) \{a \cap b \mid a \in \text{sets } F \wedge b \in \text{sets } G\}$ using subalgebra sets.top[of G] sets.sets-into-space[of - F] by (intro sigma-sets.Basic, auto simp add: subalgebra-def)

}

moreover

{

fix a assume $a \in \text{sets } F \vee a \in \text{sets } G \ a \notin \text{sets } F$

hence $a \in \text{sets } G$ by blast

hence $a \in \sigma\text{-sets}(\text{space } M) \{a \cap b \mid a \in \text{sets } F \wedge b \in \text{sets } G\}$ 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 $(\text{cond-exp } M \ F \ f) \in \text{borel-measurable}(\sigma(\text{space } M) (\text{sets } F \cup \text{sets } G))$

proof -

have $F \subseteq \sigma(\text{space } M) (F \cup G)$ by (metis Un-least Un-upper1 measure-of-of-measure sets.space-closed sets-measure-of-sigma-sets-subseteq subalg subalgebra(2) subalgebra-def)

thus ?thesis using borel-measurable-cond-exp[THEN measurable-sets, OF borel-open, of - M F f] subalgebra by (intro borel-measurableI, force simp only: space-measure-of-conv subalgebra-def)

qed

ultimately show *?thesis using assms(4) integrable-cond-exp by (intro Un-sigma.cond-exp-charact)*
presburger+
qed

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

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

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

5 Filtered Measure Spaces

5.1 Filtered Measure

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

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

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

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

end

locale *linearly-filtered-measure* = *filtered-measure* $M F t_0$ **for** M **and** $F :: - :: \{ \text{linorder-topology} \} \Rightarrow -$ **and** t_0

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

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

5.2 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 $\text{sigma-finite-subalgebra } M (F i)$
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* $M F 0 :: \text{nat}$ **for** $M F$

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

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

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

5.3 Finite Filtered Measure

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

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

using *subalgebras* **by** (*unfold-locales*, *blast*, *meson dual-order.refl finite-measure-axioms*
finite-measure-def finite-measure-restr-to-subalg sigma-finite-measure.sigma-finite-countable)

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

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

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

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

5.4 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* \subseteq *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

6 Stochastic Processes

6.1 Stochastic Process

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

locale *stochastic-process* =

fixes *M* *t*₀ **and** *X* :: '*b* :: {*second-countable-topology*, *order-topology*, *t2-space*} \Rightarrow
'*a* \Rightarrow '*c* :: {*second-countable-topology*, *banach*}

assumes *random-variable[measurable]*: $\bigwedge i. t_0 \leq i \implies X\ i \in \text{borel-measurable } M$

begin

definition *left-continuous* **where** *left-continuous* = (AE ξ in M . $\forall t$. *continuous* (at-left t) (λi . $X i \xi$))

definition *right-continuous* **where** *right-continuous* = (AE ξ in M . $\forall t$. *continuous* (at-right t) (λi . $X i \xi$))

end

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

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

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

lemma *stochastic-process-const-fun*:

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

shows *stochastic-process* $M t_0 (\lambda -. f)$ **using** *assms* **by** (*unfold-locales*)

lemma *stochastic-process-const*:

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

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

context *stochastic-process*

begin

lemma *compose-stochastic*:

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

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

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

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

lemma *scaleR-right-stochastic*:

assumes *stochastic-process* $M t_0 Y$

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

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

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

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

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

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*
 $\lambda \cdot. -1$] **by** (*simp add: fun-Compl-def*)

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

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

end

lemma *stochastic-process-sum*:

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

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

6.1.1 Natural Filtration

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

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

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

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

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

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

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

lemma *sets-natural-filtration'*:

assumes *borel* = *sigma UNIV S*

shows *sets* (natural-filtration $M t_0 X t$) = *sigma-sets* (space M) ($\bigcup_{i \in \{t_0..t\}} \{X i - ' A \cap \text{space } M \mid A. A \in S\}$)
proof (*subst sets-natural-filtration, intro sigma-sets-eqI, clarify*)
fix i **and** $A :: 'a \text{ set}$ **assume** $asm: i \in \{t_0..t\} A \in \text{sets borel}$
hence $A \in \text{sigma-sets UNIV } S$ **unfolding** *assms by simp*
thus $X i - ' A \cap \text{space } M \in \text{sigma-sets (space } M) (\bigcup_{i \in \{t_0..t\}} \{X i - ' A \cap \text{space } M \mid A. A \in S\})$
proof (*induction*)
case (*Compl a*)
have $X i - ' (UNIV - a) \cap \text{space } M = \text{space } M - (X i - ' a \cap \text{space } M)$ **by** *blast*
then show *?case using Compl(2)[THEN sigma-sets.Compl]* **by** *presburger*
next
case (*Union a*)
have $X i - ' \bigcup (\text{range } a) \cap \text{space } M = \bigcup (\text{range } (\lambda j. X i - ' a j \cap \text{space } M))$
by *blast*
then show *?case using Union(2)[THEN sigma-sets.Union]* **by** *presburger*
qed (*auto intro: asm sigma-sets.Empty*)
qed (*intro sigma-sets.Basic, force simp add: assms*)

lemma *sets-natural-filtration-open:*

sets (natural-filtration $M t_0 X t$) = *sigma-sets* (space M) ($\bigcup_{i \in \{t_0..t\}} \{X i - ' A \cap \text{space } M \mid A. \text{open } A\}$)
using *sets-natural-filtration'* **by** (*force simp only: borel-def mem-Collect-eq*)

lemma *sets-natural-filtration-oi:*

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

lemma *sets-natural-filtration-io:*

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

lemma *sets-natural-filtration-ci:*

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

context *stochastic-process*

begin

lemma *subalgebra-natural-filtration:*

shows *subalgebra* M (natural-filtration $M t_0 X i$)
unfolding *subalgebra-def using measurable-family-iff-sets* **by** (*force simp add: natural-filtration-def*)

lemma *filtered-measure-natural-filtration:*

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

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

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

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

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

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

proof (*unfold-locales*)

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

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

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

show $\bigwedge i j. \llbracket t_0 \leq i; i \leq j \rrbracket \implies$ *sets* (*natural-filtration* M t_0 X i) \subseteq *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* ($\lambda.$ *borel*) X $\{t_0..s\}$

shows *indep-set* (*natural-filtration* M t_0 X s) (*vimage-algebra* (*space* M) (X t) *borel*)

proof –

have *indep-sets* ($\lambda i. \{X$ $i - ' A \cap$ *space* M $| A. A \in$ *sets borel* $\}$) (\bigcup (*range* (*case-bool* $\{t_0..s\}$ $\{t\}$)))

using *assms*

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

6.2 Adapted Process

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

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

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

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

end

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

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

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

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

Again, we cover basic operations.

context *adapted-process*
begin

lemma *compose-adapted*:

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

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

lemma *scaleR-right-adapted*:

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

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

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

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

lemma *add-adapted*:

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

lemma *diff-adapted*:

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

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

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

proof (*unfold-locales*)

fix $i :: 'b$
have $X j \in \text{borel-measurable} (F i)$ **if** $t_0 \leq j \leq i$ **for** j **using** *that adaptedE by*
meson
thus $(\lambda \xi. \sum_{i \in \{t_0..i\}} X i \xi) \in \text{borel-measurable} (F i)$ **by** *simp*
qed

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

proof (*unfold-locales*)

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

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

end

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

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

proof (*unfold-locales*)

fix i

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

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

qed

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

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

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

proof –

{

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

then interpret *adapted-process* $M \ F \ t_0 \ X \ i$ **using** *assms* **by** *simp*

have $X \ i \ k \in \text{borel-measurable } M \ X \ i \ k \in \text{borel-measurable } (F \ k)$ **using** *measurable-from-subalg subalgebras adapted asm* **by** (*blast, simp*)

}

thus *?thesis* **by** (*unfold-locales simp*)

qed

An adapted process is necessarily a stochastic process.

sublocale *adapted-process* \subseteq *stochastic-process* **using** *measurable-from-subalg subalgebras adapted* **by** (*unfold-locales blast*)

sublocale *nat-adapted-process* \subseteq *nat-stochastic-process* ..

sublocale *real-adapted-process* \subseteq *real-stochastic-process* ..

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

lemma (*in stochastic-process*) *adapted-process-natural-filtration: adapted-process* $M \ (\text{natural-filtration } M \ t_0 \ X) \ t_0 \ X$

using *filtered-measure-natural-filtration*

by (*intro-locales*) (*auto simp add: natural-filtration-def intro!: adapted-process-axioms.intro measurable-family-vimage-algebra*)

6.3 Progressively Measurable Process

locale *progressive-process* = *filtered-measure* $M \ F \ t_0$ **for** $M \ F \ t_0$ **and** $X :: - \implies - \implies - :: \{\text{second-countable-topology, banach}\} +$

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

begin

lemma *progressiveD*:
assumes $S \in \text{borel}$
shows $(\lambda(j, \xi). X j \xi) - ' S \cap (\{t_0..i\} \times \text{space } M) \in (\text{restrict-space borel } \{t_0..i\} \otimes_M F i)$
using *measurable-sets*[*OF progressive, OF - assms, of i*]
by (*cases* $t_0 \leq i$) (*auto simp add: space-restrict-space sets-pair-measure space-pair-measure*)

end

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

lemma (*in filtered-measure*) *progressive-process-const-fun*:
assumes $f \in \text{borel-measurable } (F t_0)$
shows *progressive-process* $M F t_0 (\lambda-. f)$
proof (*unfold-locales*)
fix i **assume** *asm*: $t_0 \leq i$
have $f \in \text{borel-measurable } (F i)$ **using** *borel-measurable-mono*[*OF order.refl asm*]
assms **by** *blast*
thus *case-prod* $(\lambda-. f) \in \text{borel-measurable } (\text{restrict-space borel } \{t_0..i\} \otimes_M F i)$
using *measurable-compose*[*OF measurable-snd*] **by** *simp*
qed

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

context *progressive-process*
begin

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

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

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

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

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

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

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

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

end

A progressively measurable process is also adapted.

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

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

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

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

```

M F X
proof (intro iffI)
  assume asm: nat-progressive-process M F X
  interpret nat-progressive-process M F X by (rule asm)
  show nat-adapted-process M F X ..
next
  assume asm: nat-adapted-process M F X
  interpret nat-adapted-process M F X by (rule asm)
  show nat-progressive-process M F X
  proof (unfold-locales, intro borel-measurableI)
    fix S :: 'b set and i :: nat assume open-S: open S
    {
      fix j assume asm: j ≤ 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} ∈ 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

```

6.4 Predictable Process

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

context linearly-filtered-measure
begin

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

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

lemma sets-predictable-sigma: sets $\Sigma_P =$ sigma-sets ($\{t_0..\} \times$ space M) ($\{\{s<..t\}$
 $\times A \mid A \text{ s t. } A \in F \text{ s } \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in F t_0\}$)
unfolding predictable-sigma **using** space-F sets.sets-into-space **by** (subst sets-measure-of)
 fastforce+

lemma *measurable-predictable-sigma-snd*:
assumes *countable* \mathcal{I} $\mathcal{I} \subseteq \{\{s<..t\} \mid s t. t_0 \leq s \wedge s < t\} \{t_0<..\} \subseteq (\bigcup \mathcal{I})$
shows $snd \in \Sigma_P \rightarrow_M F t_0$
proof (*intro measurableI*)
fix $S :: 'a$ set **assume** *asm*: $S \in F t_0$
have *countable*: *countable* $((\lambda I. I \times S) ' \mathcal{I})$ **using** *assms(1)* **by** *blast*
have $(\lambda I. I \times S) ' \mathcal{I} \subseteq \{\{s<..t\} \times A \mid A s t. A \in F s \wedge t_0 \leq s \wedge s < t\}$ **using**
sets-F-mono[*OF order-refl*, *THEN subsetD*, *OF - asm*] *assms(2)* **by** *blast*
hence $(\bigcup I \in \mathcal{I}. I \times S) \cup \{t_0\} \times S \in \Sigma_P$ **unfolding** *sets-predictable-sigma* **using**
asm **by** (*intro sigma-sets-Un*[*OF sigma-sets-UNION*[*OF countable*] *sigma-sets.Basic*]
sigma-sets.Basic) *blast+*
moreover **have** $snd - ' S \cap space \Sigma_P = \{t_0..\} \times S$ **using** *sets.sets-into-space*[*OF*
asm] **by** *fastforce*
moreover **have** $\{t_0\} \cup \{t_0<..\} = \{t_0..\}$ **by** *auto*
moreover **have** $(\bigcup I \in \mathcal{I}. I \times S) \cup \{t_0\} \times S = \{t_0..\} \times S$ **using** *assms(2,3)*
calculation(3) **by** *fastforce*
ultimately show $snd - ' S \cap space \Sigma_P \in \Sigma_P$ **by** *argo*
qed (*auto*)

lemma *measurable-predictable-sigma-fst*:
assumes *countable* \mathcal{I} $\mathcal{I} \subseteq \{\{s<..t\} \mid s t. t_0 \leq s \wedge s < t\} \{t_0<..\} \subseteq (\bigcup \mathcal{I})$
shows $fst \in \Sigma_P \rightarrow_M borel$
proof –
have $A \times space M \in sets \Sigma_P$ **if** $A \in sigma-sets \{t_0..\} \{\{s<..t\} \mid s t. t_0 \leq s \wedge s < t\}$ **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 $\{t_0..\} \{\{s<..t\} \mid s t. t_0 \leq s \wedge s < t\}$
proof (*intro sigma-sets-eqI* ; *clarify*)
fix $A :: 'b$ set **assume** *asm*: $A \in sigma-sets UNIV (range greaterThan)$
thus $\{t_0..\} \cap A \in sigma-sets \{t_0..\} \{\{s<..t\} \mid s t. t_0 \leq s \wedge s < t\}$
proof (*induction rule: sigma-sets.induct*)
case (*Basic a*)
then obtain s **where** $s = \{s<..\}$ **by** *blast*

```

show ?case
proof (cases  $t_0 \leq s$ )
  case True
  hence *:  $\{t_0..\} \cap a = (\bigcup i \in \mathcal{I}. \{s<..\} \cap i)$  using  $s$  assms(3) by force
  have  $((\cap) \{s<..\} \text{ ' } \mathcal{I}) \subseteq \text{sigma-sets } \{t_0..\} \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$ 
  proof (clarify)
    fix  $A$  assume  $A \in \mathcal{I}$ 
    then obtain  $s' \ t'$  where  $A: A = \{s'<..t'\}$   $t_0 \leq s' \ s' < t'$  using assms(2)
  by blast
    hence  $\{s<..\} \cap A = \{\max s \ s' <..t'\}$  by fastforce
    moreover have  $t_0 \leq \max s \ s'$  using  $A$  True by linarith
    moreover have  $\max s \ s' < t'$  if  $s < t'$  using  $A$  that by linarith
    moreover have  $\{s<..\} \cap A = \{\}$  if  $\neg s < t'$  using  $A$  that by force
    ultimately show  $\{s<..\} \cap A \in \text{sigma-sets } \{t_0..\} \{\{s<..t\} \mid s \ t. \ t_0 \leq s \wedge s < t\}$  by (cases  $s < t'$ ) (blast, simp add: sigma-sets.Empty)
  qed
  thus ?thesis unfolding * using assms(1) by (intro sigma-sets-UNION)
auto
next
  case False
  hence  $\{t_0..\} \cap a = \{t_0..\}$  using  $s$  by force
  thus ?thesis using sigma-sets-top by auto
qed
next
  case (Compl a)
  have  $\{t_0..\} \cap (\text{UNIV} - a) = \{t_0..\} - (\{t_0..\} \cap a)$  by blast
  then show ?case using Compl(2)[THEN sigma-sets.Compl] by presburger
next
  case (Union a)
  have  $\{t_0..\} \cap \bigcup (\text{range } a) = \bigcup (\text{range } (\lambda i. \{t_0..\} \cap a \ i))$  by blast
  then show ?case using Union(2)[THEN sigma-sets.Union] by presburger
qed (simp add: sigma-sets.Empty)
next
  fix  $s \ t$  assume asm:  $t_0 \leq s \ s < t$ 
  hence *:  $\{s<..t\} = \{s<..\} \cap (\{t_0..\} - \{t<..\})$  by force
  have  $\{s<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets UNIV } (\text{range } \text{greaterThan}))$ 
using asm by (intro sigma-sets.Basic) auto
  moreover have  $\{t_0..\} - \{t<..\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets UNIV } (\text{range } \text{greaterThan}))$  using asm by (intro sigma-sets.Compl sigma-sets.Basic)
auto
  ultimately show  $\{s<..t\} \in \text{sigma-sets } \{t_0..\} ((\cap) \{t_0..\} \text{ ' } \text{sigma-sets UNIV } (\text{range } \text{greaterThan}))$  unfolding * Int-range-binary[of  $\{s<..\}$ ] by (intro sigma-sets-Inter[OF - binary-in-sigma-sets]) auto
  qed
  thus ?thesis unfolding borel-Ioi restrict-space-def emeasure-sigma by (force intro: sigma-eqI)
qed
ultimately have restrict-space borel  $\{t_0..\} \otimes_M \text{sigma } (\text{space } M) \{\} \subseteq \text{sets } \Sigma_P$ 

```

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

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

end

locale *predictable-process = linearly-filtered-measure M F t_0 for M F t_0 and X ::*
- \Rightarrow - \Rightarrow - :: {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

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

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

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

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

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

We show sufficient conditions for functions constant in one argument to

constitute a predictable process. In contrast to the cases before, this is not a triviality.

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

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

shows *predictable-process* $M F t_0 (\lambda \cdot. f)$

using *measurable-compose-rev*[*OF* *assms*(2)] *assms*(1) **by** (*unfold-locales*) (*auto simp add: measurable-split-conv*)

lemma (in *nat-filtered-measure*) *predictable-process-const-fun'*[*intro*]:

assumes $f \in \text{borel-measurable } (F 0)$

shows *nat-predictable-process* $M F (\lambda \cdot. f)$

using *assms* **by** (*intro predictable-process-const-fun*[*OF* *measurable-predictable-sigma-snd'*, *THEN nat-predictable-process.intro*])

lemma (in *real-filtered-measure*) *predictable-process-const-fun'*[*intro*]:

assumes $f \in \text{borel-measurable } (F 0)$

shows *real-predictable-process* $M F (\lambda \cdot. f)$

using *assms* **by** (*intro predictable-process-const-fun*[*OF* *measurable-predictable-sigma-snd'*, *THEN real-predictable-process.intro*])

lemma (in *linearly-filtered-measure*) *predictable-process-const*:

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

shows *predictable-process* $M F t_0 (\lambda i \cdot. c i)$

using *assms* **by** (*unfold-locales*) (*simp add: measurable-split-conv*)

lemma (in *linearly-filtered-measure*) *predictable-process-const-const*[*intro*]:

shows *predictable-process* $M F t_0 (\lambda \cdot. c)$

by (*unfold-locales*) *simp*

lemma (in *nat-filtered-measure*) *predictable-process-const'*[*intro*]:

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

shows *nat-predictable-process* $M F (\lambda i \cdot. c i)$

using *assms* **by** (*intro predictable-process-const*[*OF* *measurable-predictable-sigma-fst'*, *THEN nat-predictable-process.intro*])

lemma (in *real-filtered-measure*) *predictable-process-const'*[*intro*]:

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

shows *real-predictable-process* $M F (\lambda i \cdot. c i)$

using *assms* **by** (*intro predictable-process-const*[*OF* *measurable-predictable-sigma-fst'*, *THEN real-predictable-process.intro*])

context *predictable-process*

begin

lemma *compose-predictable*:

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

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

proof

have $(\lambda(i, \xi). (i, X i \xi)) \in \Sigma_P \rightarrow_M \text{borel} \otimes_M \text{borel}$ **using** *predictable* *assms*(1)

by (*auto simp add: measurable-pair-iff measurable-split-conv*)
 moreover have $(\lambda(i, \xi). f i (X i \xi)) = \text{case-prod } f \circ (\lambda(i, \xi). (i, X i \xi))$ by
fastforce
 ultimately show $(\lambda(i, \xi). f i (X i \xi)) \in \text{borel-measurable } \Sigma_P$ **unfolding** *borel-prod*
 using *assms* by *simp*
 qed

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

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

lemma *scaleR-right-const-fun-predictable:*
 assumes $\text{snd} \in \Sigma_P \rightarrow_M F t_0 f \in \text{borel-measurable } (F t_0)$
 shows *predictable-process* $M F t_0 (\lambda i \xi. f \xi *_{\mathbb{R}} (X i \xi))$
 using *assms* **by** (*fast intro: scaleR-right-predictable predictable-process-const-fun*)

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

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

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

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

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

end

Every predictable process is also progressively measurable.

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

proof (*unfold-locales*)

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

{

fix $S :: ('b \times 'a)$ **set** **assume** $S \in \{\{s <..t\} \times A \mid A \text{ s } t. A \in F \text{ s } \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A \mid A. A \in F \text{ } t_0\}$

hence $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) \in \text{restrict-space borel } \{t_0..i\} \otimes_M F$
 i

proof

assume $S \in \{\{s <..t\} \times A \mid A \text{ s } t. A \in F \text{ s } \wedge t_0 \leq s \wedge s < t\}$

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

hence $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) = \{s <.. \min i \text{ } t\} \times A$ **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 ≤ i) (fastforce simp add: sets-restrict-space-iff)+*

next

assume $S \in \{\{t_0\} \times A \mid A. A \in F \text{ } t_0\}$

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

hence $(\lambda x. x) - ' S \cap (\{t_0..i\} \times \text{space } M) = \{t_0\} \times A$ **using** *asm sets.sets-into-space[OF S-is(2)] by auto*

thus *?thesis using S-is(2) sets-F-mono[OF order-refl asm] asm by (fastforce simp add: sets-restrict-space-iff)*

qed

hence $(\lambda x. x) - ' S \cap \text{space } (\text{restrict-space borel } \{t_0..i\} \otimes_M F \text{ } i) \in \text{restrict-space}$
 $\text{borel } \{t_0..i\} \otimes_M F \text{ } i$ **by** (*simp add: space-pair-measure space-F[OF asm]*)

}

moreover have $\{\{s <..t\} \times A \mid A \text{ s } t. A \in \text{sets } (F \text{ } s) \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\}$
 $\times A \mid A. A \in \text{sets } (F \text{ } t_0)\} \subseteq \text{Pow } (\{t_0..\} \times \text{space } M)$ **using** *sets.sets-into-space by*
force

ultimately have $(\lambda x. x) \in \text{restrict-space borel } \{t_0..i\} \otimes_M F \text{ } i \rightarrow_M \Sigma_P$ **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} ⊗_M F i) using*
predictable by simp

qed

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

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

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

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

assumes $(\bigcup i. \{i\} \times A \text{ } i) \in \Sigma_P$

shows $A \text{ } (\text{Suc } i) \in F \text{ } i \text{ } A \text{ } 0 \in F \text{ } 0$

using *assms unfolding sets-predictable-sigma*

proof (*induction* $(\bigcup i. \{i\} \times A \text{ } i)$ *arbitrary: A*)

case *Basic*

{

assume $\exists S. (\bigcup i. \{i\} \times A \text{ } i) = \{0\} \times S$

then obtain S where $S: (\bigcup i. \{i\} \times A i) = \{bot\} \times S$ **unfolding** *bot-nat-def*
by *blast*
 hence $S \in F bot$ **using** *Basic* **by** (*fastforce simp add: times-eq-iff bot-nat-def*)
 moreover have $A i = \{\}$ **if** $i \neq bot$ **for** i **using** *that S* **by** *blast*
 moreover have $A bot = S$ **using** S **by** *blast*
 ultimately have $A (Suc i) \in F i$ $A 0 \in F 0$ **for** i **unfolding** *bot-nat-def* **by**
(auto simp add: bot-nat-def)
 }
 note $*$ = *this*
 {
 assume $\nexists S. (\bigcup i. \{i\} \times A i) = \{0\} \times S$
 then obtain $s t B$ where $B: (\bigcup i. \{i\} \times A i) = \{s<..t\} \times B$ $B \in sets (F s)$
 $s < t$ **using** *Basic* **by** *auto*
 hence $A i = B$ **if** $i \in \{s<..t\}$ **for** i **using** *that* **by** *fast*
 moreover have $A i = \{\}$ **if** $i \notin \{s<..t\}$ **for** i **using** B **that** **by** *fastforce*
 ultimately have $A (Suc i) \in F i$ $A 0 \in F 0$ **for** i **unfolding** *bot-nat-def* **using**
 B *sets-F-mono* **by** (*auto simp add: bot-nat-def*) (*metis less-Suc-eq-le sets.empty-sets*
subset-eq)
 }
 note $**$ = *this*
 show $A (Suc i) \in sets (F i)$ $A 0 \in sets (F 0)$ **using** $*(1)[of i]$ $*(2)$ $** (1)[of i]$
 $** (2)$ **by** *blast+*
next
 case *Empty*
 {
 case 1
 then show *?case* **using** *Empty* **by** *simp*
next
 case 2
 then show *?case* **using** *Empty* **by** *simp*
 }
next
 case (*Compl a*)
 have $a-in: a \subseteq \{0..\} \times space M$ **using** *Compl(1)* *sets.sets-into-space* *sets-predictable-sigma*
space-predictable-sigma **by** *metis*
 hence $A-in: A i \subseteq space M$ **for** i **using** *Compl(4)* **by** *blast*
 have $a: a = \{0..\} \times space M - (\bigcup i. \{i\} \times A i)$ **using** $a-in$ *Compl(4)* **by** *blast*
 also have $... = - (\bigcap j. - (\{j\} \times (space M - A j)))$ **by** *blast*
 also have $... = (\bigcup j. \{j\} \times (space M - A j))$ **by** *blast*
 finally have $*$: $(space M - A (Suc i)) \in F i$ $(space M - A 0) \in F 0$ **using**
Compl(2,3) **by** *auto*
 {
 case 1
 then show *?case* **using** $*$ $A-in$ **by** (*metis bot-nat-0.extremum double-diff*
sets.Diff sets.top sets-F-mono sets-le-imp-space-le space-F)
next
 case 2
 then show *?case* **using** $*$ $A-in$ **by** (*metis bot-nat-0.extremum double-diff*
sets.Diff sets.top sets-F-mono sets-le-imp-space-le space-F)

```

}
next
  case (Union a)
    have a-in:  $a \ i \subseteq \{0..\} \times \text{space } M$  for  $i$  using Union(1) sets.sets-into-space
    sets-predictable-sigma space-predictable-sigma by metis
    hence A-in:  $A \ i \subseteq \text{space } M$  for  $i$  using Union(4) by blast
    have snd  $x \in \text{snd} \ ' (a \ i \cap (\{fst \ x\} \times \text{space } M))$  if  $x \in a \ i$  for  $i \ x$  using that
    a-in by fastforce
    hence a-i:  $a \ i = (\bigcup j. \{j\} \times (\text{snd} \ ' (a \ i \cap (\{j\} \times \text{space } M))))$  for  $i$  by force
    have A-i:  $A \ i = \text{snd} \ ' (\bigcup (\text{range } a) \cap (\{i\} \times \text{space } M))$  for  $i$  unfolding
    Union(4) using A-in by force
    have *:  $\text{snd} \ ' (a \ j \cap (\{Suc \ i\} \times \text{space } M)) \in F \ i \ \text{snd} \ ' (a \ j \cap (\{0\} \times \text{space } M))$ 
     $\in F \ 0$  for  $j$  using Union(2,3)[OF a-i] by auto
    {
      case 1
        have  $(\bigcup j. \text{snd} \ ' (a \ j \cap (\{Suc \ i\} \times \text{space } M))) \in F \ i$  using * by fast
        moreover have  $(\bigcup j. \text{snd} \ ' (a \ j \cap (\{Suc \ i\} \times \text{space } M))) = \text{snd} \ ' (\bigcup (\text{range } a) \cap$ 
         $(\{Suc \ i\} \times \text{space } M))$  by fast
        ultimately show ?case using A-i by metis
      next
        case 2
          have  $(\bigcup j. \text{snd} \ ' (a \ j \cap (\{0\} \times \text{space } M))) \in F \ 0$  using * by fast
          moreover have  $(\bigcup j. \text{snd} \ ' (a \ j \cap (\{0\} \times \text{space } M))) = \text{snd} \ ' (\bigcup (\text{range } a) \cap$ 
           $(\{0\} \times \text{space } M))$  by fast
          ultimately show ?case using A-i by metis
        }
    }
qed

```

This leads to the following useful fact.

lemma (in *nat-predictable-process*) *adapted-Suc*: *nat-adapted-process* $M \ F \ (\lambda i. \ X \ (Suc \ i))$

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

```

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

```

thus $X (Suc\ i) -' S \cap space\ (F\ i) \in sets\ (F\ i)$ **using** *sets-in-filtration*[of λj .
if $j = Suc\ i$ then $(X (Suc\ i) -' S \cap space\ M)$ else $\{\}$] *space-F*[OF zero-le] **by**
presburger
qed

The following lemma characterizes predictability in the discrete setting.

theorem *nat-predictable-process-iff*: *nat-predictable-process* $M\ F\ X \longleftrightarrow$ *nat-adapted-process*
 $M\ F\ (\lambda i. X (Suc\ i)) \wedge X\ 0 \in borel-measurable\ (F\ 0)$

proof (*intro iffI*)

assume *asm*: *nat-adapted-process* $M\ F\ (\lambda i. X (Suc\ i)) \wedge X\ 0 \in borel-measurable$
 $(F\ 0)$

interpret *nat-adapted-process* $M\ F\ \lambda i. X (Suc\ i)$ **using** *asm* **by** *blast*

have $(\lambda(x, y). X\ x\ y) \in borel-measurable\ \Sigma_P$

proof (*intro borel-measurableI*)

fix $S :: 'b\ set$ **assume** *open-S*: *open* S

have $\{i\} \times (X\ i -' S \cap space\ M) \in sets\ \Sigma_P$ **for** i

proof (*cases i*)

case 0

then show *?thesis unfolding sets-predictable-sigma*

using *measurable-sets*[OF - *borel-open*[OF *open-S*], of $X\ 0\ F\ 0$] *asm* **by** *auto*

next

case $(Suc\ i)$

have $\{Suc\ i\} = \{i <.. Suc\ i\}$ **by** *fastforce*

then show *?thesis unfolding sets-predictable-sigma*

using *measurable-sets*[OF *adapted borel-open*[OF *open-S*], of i]

by (*intro sigma-sets.Basic, auto simp add: Suc*)

qed

moreover have $(\lambda(x, y). X\ x\ y) -' S \cap space\ \Sigma_P = (\bigcup i. \{i\} \times (X\ i -' S \cap$
 $space\ M))$ **by** *fastforce*

ultimately show $(\lambda(x, y). X\ x\ y) -' S \cap space\ \Sigma_P \in sets\ \Sigma_P$ **by** *simp*

qed

thus *nat-predictable-process* $M\ F\ X$ **by** (*unfold-locales*)

next

assume *asm*: *nat-predictable-process* $M\ F\ X$

interpret *nat-predictable-process* $M\ F\ X$ **by** (*rule asm*)

show *nat-adapted-process* $M\ F\ (\lambda i. X (Suc\ i)) \wedge X\ 0 \in borel-measurable\ (F\ 0)$

using *adapted-Suc* **by** *simp*

qed

end

theory *Martingale*

imports *Stochastic-Process Conditional-Expectation-Banach*

begin

7 Martingales

The following locales are necessary for defining martingales.

7.1 Additional Locale Definitions

locale *sigma-finite-adapted-process* = *sigma-finite-filtered-measure* $M F t_0$ + *adapted-process* $M F t_0 X$ **for** $M F t_0 X$

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

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

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

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

locale *finite-adapted-process* = *finite-filtered-measure* $M F t_0$ + *adapted-process* $M F t_0 X$ **for** $M F t_0 X$

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

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

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

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

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

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

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

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

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

..

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

..

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

locale *nat-finite-adapted-process-order* = *finite-adapted-process-order* $M F 0$:: *nat*

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

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

locale *sigma-finite-adapted-process-linorder* = *sigma-finite-adapted-process-order M F t₀ X* **for** *M F t₀ X* **and** *X* :: - \Rightarrow - \Rightarrow - :: {*linorder-topology*}

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

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

locale *finite-adapted-process-linorder* = *finite-adapted-process-order M F t₀ X* **for** *M F t₀ X* **and** *X* :: - \Rightarrow - \Rightarrow - :: {*linorder-topology*}

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

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

7.2 Martingale

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

locale *martingale-order* = *martingale M F t₀ X* **for** *M F t₀ X* **and** *X* :: - \Rightarrow - \Rightarrow -
 :: {*order-topology, ordered-real-vector*}

locale *martingale-linorder* = *martingale M F t₀ X* **for** *M F t₀ X* **and** *X* :: - \Rightarrow - \Rightarrow

- :: {linorder-topology, ordered-real-vector}
sublocale martingale-linorder \subseteq martingale-order ..

lemma (in sigma-finite-filtered-measure) martingale-const-fun[*intro*]:
assumes integrable $M f f \in$ borel-measurable $(F t_0)$
shows martingale $M F t_0 (\lambda-. f)$
using *assms* sigma-finite-subalgebra.cond-exp-F-meas[*OF - assms*(1), *THEN AE-symmetric*]
borel-measurable-mono
by (unfold-locales) blast+

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

by (unfold-locales) (auto *intro*: sigma-finite-subalgebra.cond-exp-nested-subalg[*OF - assms*]
simp add: subalgebra-F subalgebras)

corollary (in sigma-finite-filtered-measure) martingale-zero[*intro*]: martingale $M F t_0 (\lambda-. 0)$ **by** fastforce

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

7.3 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-adapted-process-order +
assumes integrable: $\bigwedge i. t_0 \leq i \implies$ integrable $M (X i)$
and submartingale-property: $\bigwedge i j. t_0 \leq i \implies i \leq j \implies$ AE ξ in $M. X i \xi \leq$
cond-exp $M (F i) (X j) \xi$

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

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

7.4 Supermartingale

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-adapted-process-order +
assumes integrable: $\bigwedge i. t_0 \leq i \implies$ integrable $M (X i)$

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

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

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

7.5 Martingale Lemmas

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

context *martingale*

begin

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

assumes $t_0 \leq i \leq j$

shows $AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X j \xi - X i \xi) \xi = 0$

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

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

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

fastforce

lemma *set-integral-eq*:

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

shows *set-lebesgue-integral* $M A (X i) =$ *set-lebesgue-integral* $M A (X j)$

proof –

interpret *sigma-finite-subalgebra* $M F i$ **using** *assms(2)* **by** *blast*

have $\int x \in A. X i x \partial M = \int x \in A. \text{cond-exp } M (F i) (X j) x \partial M$ **using** *martingale-property*[*OF assms(2,3)*] *borel-measurable-cond-exp' assms subalgebras*

subalgebra-def **by** (*intro set-lebesgue-integral-cong-AE[OF - random-variable]*) *fast-force+*

also have $\dots = \int x \in A. X j x \partial M$ **using** *assms* **by** (*auto simp: integrable intro: cond-exp-set-integral[symmetric]*)

finally show *?thesis* .

qed

lemma *scaleR-const[intro]*:

shows *martingale* $M F t_0 (\lambda i x. c *_R X i x)$

proof –

{

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

interpret *sigma-finite-subalgebra* $M F i$ **using** *asm* **by** *blast*

have *AE* x *in* $M. c *_R X i x = \text{cond-exp } M (F i) (\lambda x. c *_R X j x) x$ **using** *asm cond-exp-scaleR-right[OF integrable, of j, THEN AE-symmetric]* *martingale-property[OF asm]* **by** *force*

}

thus *?thesis* **by** (*unfold-locales*) (*auto simp add: integrable martingale.integrable*)

qed

lemma *uminus[intro]*:

shows *martingale* $M F t_0 (- X)$

using *scaleR-const[of -1]* **by** (*force intro: back-subst[of martingale M F t_0]*)

lemma *add[intro]*:

assumes *martingale* $M F t_0 Y$

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

proof –

interpret $Y: \text{martingale } M F t_0 Y$ **by** (*rule assms*)

{

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

hence *AE* ξ *in* $M. X i \xi + Y i \xi = \text{cond-exp } M (F i) (\lambda x. X j x + Y j x) \xi$

using *sigma-finite-subalgebra.cond-exp-add[OF - integrable martingale.integrable[OF assms], of F i j j, THEN AE-symmetric]*

martingale-property[OF asm] martingale.martingale-property[OF assms asm] **by** *force*

}

thus *?thesis* **using** *assms*

by (*unfold-locales*) (*auto simp add: integrable martingale.integrable*)

qed

lemma *diff[intro]*:

assumes *martingale* $M F t_0 Y$

shows *martingale* $M F t_0 (\lambda i x. X i x - Y i x)$

proof –

interpret $Y: \text{martingale } M F t_0 Y$ **by** (*rule assms*)

{

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

hence *AE* ξ *in* $M. X i \xi - Y i \xi = \text{cond-exp } M (F i) (\lambda x. X j x - Y j x) \xi$

```

using sigma-finite-subalgebra.cond-exp-diff[OF - integrable martingale.integrable[OF
assms], of F i j j, THEN AE-symmetric]
      martingale-property[OF asm] martingale.martingale-property[OF assms
asm] by fastforce
}
thus ?thesis using assms by (unfold-locales) (auto simp add: integrable martin-
gale.integrable)
qed

end

```

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

```

lemma (in sigma-finite-adapted-process) martingale-of-cond-exp-diff-eq-zero:
  assumes integrable:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$ 
    and diff-zero:  $\bigwedge i j. t_0 \leq i \implies i \leq j \implies AE \ x \text{ in } M. \text{ cond-exp } M (F i) (\lambda \xi. X j \xi - X i \xi) \ x = 0$ 
  shows martingale M F t_0 X

```

```

proof
{
  fix i j :: 'b assume asm:  $t_0 \leq i \leq j$ 
  thus AE \xi in M. X i \xi = cond-exp M (F i) (X j) \xi
  using diff-zero[OF asm] sigma-finite-subalgebra.cond-exp-diff[OF - integrable(1,1),
of F i j i]
    sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable adapted, of i] by
fastforce
}
qed (intro integrable)

```

```

lemma (in sigma-finite-adapted-process) martingale-of-set-integral-eq:
  assumes integrable:  $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$ 
    and  $\bigwedge A \ i \ j. t_0 \leq i \implies i \leq j \implies A \in F i \implies \text{set-lebesgue-integral } M A (X i) = \text{set-lebesgue-integral } M A (X j)$ 
  shows martingale M F t_0 X

```

```

proof (unfold-locales)
  fix i j :: 'b assume asm:  $t_0 \leq i \leq j$ 
  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  $A \in \text{restr-to-subalg } M (F i)$ 
    hence  $*$ :  $A \in F i$  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  $\dots = \text{set-lebesgue-integral } M A (\text{cond-exp } M (F i) (X j))$  using  $*$ 
assms(2)[OF asm] cond-exp-set-integral[OF integrable] asm by auto
    finally have set-lebesgue-integral (restr-to-subalg M (F i)) A (X i) = set-lebesgue-integral (restr-to-subalg M (F i)) A (cond-exp M (F i) (X j)) using * subalg by (auto simp: set-lebesgue-integral-def intro!: integral-subalgebra2[symmetric] borel-measurable-scaleR

```

borel-measurable-cond-exp borel-measurable-indicator)
 }
hence $AE \xi$ in *restr-to-subalg* $M (F i)$. $X i \xi = \text{cond-exp } M (F i) (X j) \xi$ **using** *asm* **by** (*intro r.density-unique-banach, auto intro: integrable-in-subalg subalg borel-measurable-cond-exp integrable*)
thus $AE \xi$ in M . $X i \xi = \text{cond-exp } M (F i) (X j) \xi$ **using** *AE-restr-to-subalg[OF subalg]* **by** *blast*
qed (*simp add: integrable*)

7.6 Submartingale Lemmas

context *submartingale*
begin

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

lemma *add[intro]*:
assumes *submartingale* $M F t_0 Y$
shows *submartingale* $M F t_0 (\lambda i \xi. X i \xi + Y i \xi)$
proof –
interpret Y : *submartingale* $M F t_0 Y$ **by** (*rule assms*)
 {
fix $i j$:: 'b **assume** *asm*: $t_0 \leq i \leq j$
hence $AE \xi$ in M . $X i \xi + Y i \xi \leq \text{cond-exp } M (F i) (\lambda x. X j x + Y j x) \xi$
using *sigma-finite-subalgebra.cond-exp-add[OF - integrable submartingale.integrable[OF assms], of F i j j]*
submartingale-property[OF asm] *submartingale.submartingale-property[OF assms asm]* *add-mono[of X i - - Y i -]* **by** *force*
 }
thus *?thesis* **using** *assms* **by** (*unfold-locales*) (*auto simp add: borel-measurable-add random-variable adapted integrable Y.random-variable Y.adapted submartingale.integrable*)

qed

lemma *diff[intro]*:
assumes *supermartingale* $M F t_0 Y$
shows *submartingale* $M F t_0 (\lambda i \xi. X i \xi - Y i \xi)$
proof –
interpret Y : *supermartingale* $M F t_0 Y$ **by** (*rule assms*)
 {
fix $i j$:: 'b **assume** *asm*: $t_0 \leq i \leq j$
hence $AE \xi$ in M . $X i \xi - Y i \xi \leq \text{cond-exp } M (F i) (\lambda x. X j x - Y j x) \xi$
using *sigma-finite-subalgebra.cond-exp-diff[OF - integrable supermartingale.integrable[OF assms], of F i j j]*

```

      submartingale-property[OF asm] supermartingale.supermartingale-property[OF
assms asm] diff-mono[of X i - - Y i -] by force
    }
    thus ?thesis using assms by (unfold-locales) (auto simp add: borel-measurable-diff
random-variable adapted integrable Y.random-variable Y.adapted supermartingale.integrable)

```

qed

lemma *scaleR-nonneg*:

```

  assumes  $c \geq 0$ 
  shows supermartingale M F t0 ( $\lambda i \xi. c *_{\mathbb{R}} X i \xi$ )

```

proof

```

{
  fix i j :: 'b assume asm: t0 ≤ i i ≤ j
  thus AE ξ in M. c *ℝ X i ξ ≤ cond-exp M (F i) (λξ. c *ℝ X j ξ) ξ
    using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i
j c] submartingale-property[OF asm] by (fastforce intro!: scaleR-left-mono[OF -
assms])
}

```

qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

lemma *scaleR-le-zero*:

```

  assumes  $c \leq 0$ 
  shows supermartingale M F t0 ( $\lambda i \xi. c *_{\mathbb{R}} X i \xi$ )

```

proof

```

{
  fix i j :: 'b assume asm: t0 ≤ i i ≤ j
  thus AE ξ in M. c *ℝ X i ξ ≥ cond-exp M (F i) (λξ. c *ℝ X j ξ) ξ
    using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i j
c] submartingale-property[OF asm]
    by (fastforce intro!: scaleR-left-mono-neg[OF - assms])
}

```

qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

lemma *uminus[intro]*:

```

  shows supermartingale M F t0 (- X)
  unfolding fun-Compl-def using scaleR-le-zero[of -1] by simp

```

end

context *submartingale-linorder*

begin

lemma *set-integral-le*:

```

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

```

by (*subst sigma-finite-subalgebra.cond-exp-set-integral*[*OF - integrable assms*(1), of *j*])
(auto intro!: scaleR-left-mono integral-mono-AE-banach integrable-mult-indicator integrable simp add: subalgebra-def set-lebesgue-integral-def)

lemma *max*:

assumes *submartingale-linorder* *M F t₀ Y*
shows *submartingale-linorder* *M F t₀ (λ i ξ. max (X i ξ) (Y i ξ))*
proof (*unfold-locales*)
interpret *Y: submartingale-linorder M F t₀ Y* **by** (*rule assms*)
{
fix *i j :: 'b* **assume** *asm: t₀ ≤ i i ≤ j*
have *AE ξ in M. max (X i ξ) (Y i ξ) ≤ max (cond-exp M (F i) (X j) ξ)*
(*cond-exp M (F i) (Y j) ξ*) **using** *submartingale-property Y.submartingale-property*
asm unfolding max-def by fastforce
thus *AE ξ in M. max (X i ξ) (Y i ξ) ≤ cond-exp M (F i) (λξ. max (X j ξ) (Y*
j ξ)) ξ **using** *sigma-finite-subalgebra.cond-exp-max*[*OF - integrable Y.integrable, of*
F i j j] *asm by (fast intro: order.trans)*
}
show $\bigwedge i. t_0 \leq i \implies (\lambda \xi. \max (X i \xi) (Y i \xi)) \in \text{borel-measurable } (F i) \bigwedge i.$
 $t_0 \leq i \implies \text{integrable } M (\lambda \xi. \max (X i \xi) (Y i \xi))$ **by** (*force intro: Y.integrable*
integrable assms)
qed

lemma *max-0*:

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

end

lemma (*in sigma-finite-adapted-process-order*) *submartingale-of-cond-exp-diff-nonneg*:

assumes *integrable: $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$*
and *diff-nonneg: $\bigwedge i j. t_0 \leq i \implies i \leq j \implies \text{AE } x \text{ in } M. \text{cond-exp } M (F i)$*
($\lambda \xi. X j \xi - X i \xi$) $x \geq 0$
shows *submartingale* *M F t₀ X*
proof (*unfold-locales*)
{
fix *i j :: 'b* **assume** *asm: t₀ ≤ i i ≤ j*
thus *AE ξ in M. X i ξ ≤ cond-exp M (F i) (X j) ξ*
using *diff-nonneg*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable(1,1), of F i j i*]
sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted, of i*] **by**
fastforce
}
qed (*intro integrable*)

lemma (in *sigma-finite-adapted-process-linorder*) *submartingale-of-set-integral-le*:
assumes *integrable*: $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$
and $\bigwedge A i j. t_0 \leq i \implies i \leq j \implies A \in F i \implies \text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X j)$
shows *submartingale* $M F t_0 X$
proof (*unfold-locales*)
{
 fix $i j :: 'b$ **assume** *asm*: $t_0 \leq i \ i \leq j$
 interpret r : *sigma-finite-measure restr-to-subalg* $M (F i)$ **using** *asm sigma-finite-subalgebra.sigma-fin-subalg*
by *blast*
 {
 fix A **assume** $A \in \text{restr-to-subalg } M (F i)$
 hence $*$: $A \in F i$ **using** *asm sets-restr-to-subalg subalgebras* **by** *blast*
 have $\text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (X i) = \text{set-lebesgue-integral } M A (X i)$ **using** $*$ *asm subalgebras* **by** (*auto simp: set-lebesgue-integral-def intro: integral-subalgebra2 borel-measurable-scaleR adapted borel-measurable-indicator*)
 also have $\dots \leq \text{set-lebesgue-integral } M A (\text{cond-exp } M (F i) (X j))$ **using** $*$ *assms(2)[OF asm] asm sigma-finite-subalgebra.cond-exp-set-integral[OF - integrable]* **by** *fastforce*
 also have $\dots = \text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (\text{cond-exp } M (F i) (X j))$ **using** $*$ *asm subalgebras* **by** (*auto simp: set-lebesgue-integral-def intro!: integral-subalgebra2[symmetric] borel-measurable-scaleR borel-measurable-cond-exp borel-measurable-indicator*)
 finally have $0 \leq \text{set-lebesgue-integral } (\text{restr-to-subalg } M (F i)) A (\lambda \xi. \text{cond-exp } M (F i) (X j) \xi - X i \xi)$ **using** $*$ *asm subalgebras* **by** (*subst set-integral-diff, auto simp add: set-integrable-def sets-restr-to-subalg intro!: integrable adapted integrable-in-subalg borel-measurable-scaleR borel-measurable-indicator borel-measurable-cond-exp integrable-mult-indicator*)
 }
 hence $A E \xi$ in *restr-to-subalg* $M (F i)$. $0 \leq \text{cond-exp } M (F i) (X j) \xi - X i \xi$
 by (*intro r.density-nonneg integrable-in-subalg asm subalgebras borel-measurable-diff borel-measurable-cond-exp adapted Bochner-Integration.integrable-diff integrable-cond-exp integrable*)
 thus $A E \xi$ in M . $X i \xi \leq \text{cond-exp } M (F i) (X j) \xi$ **using** *AE-restr-to-subalg[OF subalgebras] asm* **by** *simp*
 }
qed (*intro integrable*)

7.7 Supermartingale Lemmas

The following lemmas are exact duals of the ones for submartingales.

context *supermartingale*
begin

lemma *cond-exp-diff-nonneg*:
assumes $t_0 \leq i \ i \leq j$
shows $A E x$ in M . $\text{cond-exp } M (F i) (\lambda \xi. X i \xi - X j \xi) x \geq 0$
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 t₀ Y*
shows *supermartingale M F t₀ (λ i ξ. X i ξ + Y i ξ)*

proof -

interpret *Y: supermartingale M F t₀ Y by (rule assms)*

{
fix *i j :: 'b assume asm: t₀ ≤ i i ≤ j*
hence *AE ξ in M. X i ξ + Y i ξ ≥ cond-exp M (F i) (λ x. X j x + Y j x) ξ*
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 t₀ Y*
shows *supermartingale M F t₀ (λ i ξ. X i ξ - Y i ξ)*

proof -

interpret *Y: submartingale M F t₀ Y by (rule assms)*

{
fix *i j :: 'b assume asm: t₀ ≤ i i ≤ j*
hence *AE ξ in M. X i ξ - Y i ξ ≥ cond-exp M (F i) (λ x. X j x - Y j x) ξ*
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 ≥ 0*
shows *supermartingale M F t₀ (λ i ξ. c *_R X i ξ)*

proof

{
fix *i j :: 'b assume asm: t₀ ≤ i i ≤ j*
thus *AE ξ in M. c *_R X i ξ ≥ cond-exp M (F i) (λ ξ. c *_R X j ξ) ξ*
using *sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i*
j c] supermartingale-property[OF asm] by (fastforce intro!: scaleR-left-mono[OF -

```

assms])
}
qed (auto simp add: borel-measurable-integrable borel-measurable-scaleR integrable
random-variable adapted borel-measurable-const-scaleR)

```

lemma *scaleR-le-zero*:

```

assumes  $c \leq 0$ 
shows submartingale  $M F t_0 (\lambda i \xi. c *_{\mathbb{R}} X i \xi)$ 
proof
{
  fix  $i j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
  thus  $AE \xi \text{ in } M. c *_{\mathbb{R}} X i \xi \leq \text{cond-exp } M (F i) (\lambda \xi. c *_{\mathbb{R}} X j \xi) \xi$ 
  using sigma-finite-subalgebra.cond-exp-scaleR-right[OF - integrable, of F i j c]
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  $A \in F i t_0 \leq i \leq j$ 
shows set-lebesgue-integral  $M A (X i) \geq \text{set-lebesgue-integral } M A (X j)$ 
using supermartingale-property[OF assms(2), of j] assms subalgebras
by (subst sigma-finite-subalgebra.cond-exp-set-integral[OF - integrable assms(1),
of j])
(auto intro!: scaleR-left-mono integral-mono-AE-banach integrable-mult-indicator
integrable simp add: subalgebra-def set-lebesgue-integral-def)

```

lemma *min*:

```

assumes supermartingale-linorder  $M F t_0 Y$ 
shows supermartingale-linorder  $M F t_0 (\lambda i \xi. \min (X i \xi) (Y i \xi))$ 
proof (unfold-locales)
interpret  $Y$ : supermartingale-linorder  $M F t_0 Y$  by (rule assms)
{
  fix  $i j :: 'b$  assume asm:  $t_0 \leq i \leq j$ 
  have  $AE \xi \text{ in } M. \min (X i \xi) (Y i \xi) \geq \min (\text{cond-exp } M (F i) (X j) \xi) (\text{cond-exp }
M (F i) (Y j) \xi)$  using supermartingale-property Y.supermartingale-property asm
unfolding min-def by fastforce
  thus  $AE \xi \text{ in } M. \min (X i \xi) (Y i \xi) \geq \text{cond-exp } M (F i) (\lambda \xi. \min (X j \xi) (Y
j \xi)) \xi$  using sigma-finite-subalgebra.cond-exp-min[OF - integrable Y.integrable, of

```

$F i j j]$ *asm* **by** (*fast intro: order.trans*)
}
show $\bigwedge i. t_0 \leq i \implies (\lambda \xi. \min (X i \xi) (Y i \xi)) \in \text{borel-measurable } (F i) \bigwedge i.$
 $t_0 \leq i \implies \text{integrable } M (\lambda \xi. \min (X i \xi) (Y i \xi))$ **by** (*force intro: Y.integrable*
integrable assms)
qed

lemma *min-0*:

shows *supermartingale-linorder* $M F t_0 (\lambda i \xi. \min 0 (X i \xi))$

proof –

interpret *zero: martingale-linorder* $M F t_0 \lambda \cdot . 0$ **by** (*force intro: martin-*
gale-linorder.intro)

show *?thesis* **by** (*intro zero.min supermartingale-linorder.intro supermartin-*
gale-axioms)

qed

end

lemma (*in sigma-finite-adapted-process-order*) *supermartingale-of-cond-exp-diff-le-zero*:

assumes *integrable*: $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$

and *diff-le-zero*: $\bigwedge i j. t_0 \leq i \implies i \leq j \implies \text{AE } x \text{ in } M. \text{ cond-exp } M (F i)$

$(\lambda \xi. X j \xi - X i \xi) x \leq 0$

shows *supermartingale* $M F t_0 X$

proof

{

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

thus $\text{AE } \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X j) \xi$

using *diff-le-zero*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - inte-*
grable(1,1), of F i j i]

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

fastforce

}

qed (*intro integrable*)

lemma (*in sigma-finite-adapted-process-linorder*) *supermartingale-of-set-integral-ge*:

assumes *integrable*: $\bigwedge i. t_0 \leq i \implies \text{integrable } M (X i)$

and $\bigwedge A \ i j. t_0 \leq i \implies i \leq j \implies A \in F i \implies \text{set-lebesgue-integral } M A (X$
 $j) \leq \text{set-lebesgue-integral } M A (X i)$

shows *supermartingale* $M F t_0 X$

proof –

interpret $-$: *adapted-process* $M F t_0 -X$ **by** (*rule uminus-adapted*)

interpret *uminus-X*: *sigma-finite-adapted-process-linorder* $M F t_0 -X$..

note $*$ = *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(2)*]] **[symmetric]*]]
subalgebras

by (*intro submartingale.uminus uminus-X.submartingale-of-set-integral-le*)

(*clarsimp simp add: fun-Compl-def subalgebra-def integrable | fastforce*)
 $+$

thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

Many of the statements we have made concerning martingales can be simplified when the indexing set is the natural numbers. Given a point in time $i \in \mathbb{N}$, it suffices to consider the successor $i + (1::'a)$, instead of all future times $i \leq j$.

7.8 Discrete Time Martingales

locale *nat-martingale* = *martingale* $M F 0 :: nat X$ **for** $M F X$

locale *nat-submartingale* = *submartingale* $M F 0 :: nat X$ **for** $M F X$

locale *nat-supermartingale* = *supermartingale* $M F 0 :: nat X$ **for** $M F X$

locale *nat-submartingale-linorder* = *submartingale-linorder* $M F 0 :: nat X$ **for** $M F X$

locale *nat-supermartingale-linorder* = *supermartingale-linorder* $M F 0 :: nat X$ **for** $M F X$

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

sublocale *nat-supermartingale-linorder* \subseteq *nat-supermartingale* ..

A predictable martingale is necessarily constant.

lemma (**in** *nat-martingale*) *predictable-const*:

assumes *nat-predictable-process* $M F X$

shows $\forall \xi$ in M . $X i \xi = X j \xi$

proof –

have $*$: $\forall \xi$ in M . $X i \xi = X 0 \xi$ **for** i

proof (*induction* i)

case 0

then show *?case* **by** (*simp add: bot-nat-def*)

next

case (*Suc* i)

interpret S : *nat-adapted-process* $M F \lambda i$. X (*Suc* i) **by** (*intro nat-predictable-process.adapted-Suc assms*)

show *?case* **using** *Suc S.adapted[of i] martingale-property[OF - le-SucI, of i] sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i]* **by** *fastforce*

qed

show *?thesis* **using** $*$ [*of i*] $*$ [*of j*] **by** *force*

qed

lemma (**in** *nat-sigma-finite-adapted-process*) *martingale-of-set-integral-eq-Suc*:

assumes *integrable*: $\bigwedge i$. *integrable* M ($X i$)

and $\bigwedge A$ i . $A \in F i \implies$ *set-lebesgue-integral* $M A$ ($X i$) = *set-lebesgue-integral* $M A$ (X (*Suc* i))

shows *nat-martingale* $M F X$

proof (*intro nat-martingale.intro martingale-of-set-integral-eq*)

fix $i j A$ **assume** *asm*: $i \leq j$ $A \in$ *sets* ($F i$)

show *set-lebesgue-integral* $M A (X i) = \text{set-lebesgue-integral } M A (X j)$ **using** *asm*
proof (*induction j - i arbitrary: i j*)
 case 0
 then show ?*case* **using** *asm* **by** *simp*
next
 case (*Suc n*)
 hence *: $n = j - \text{Suc } i$ **by** *linarith*
 have $\text{Suc } i \leq j$ **using** *Suc(2,3)* **by** *linarith*
 thus ?*case* **using** *sets-F-mono[OF - le-SucI] Suc(4) Suc(1)[OF *]* **by** (*auto*
intro: assms(2)[THEN trans])
 qed
qed (*simp add: integrable*)

lemma (*in nat-sigma-finite-adapted-process*) *martingale-nat*:
 assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$
 and $\bigwedge i. AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X (\text{Suc } i)) \xi$
 shows *nat-martingale* $M F X$
proof (*unfold-locales*)
 fix $i j :: \text{nat}$ **assume** *asm*: $i \leq j$
 show $AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X j) \xi$ **using** *asm*
 proof (*induction j - i arbitrary: i j*)
 case 0
 hence $j = i$ **by** *simp*
 thus ?*case* **using** *sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable adapted, THEN AE-symmetric]* **by** *blast*
next
 case (*Suc n*)
 have $j: j = \text{Suc } (n + i)$ **using** *Suc* **by** *linarith*
 have $n: n = n + i - i$ **using** *Suc* **by** *linarith*
 have *: $AE \xi \text{ in } M. \text{cond-exp } M (F (n + i)) (X j) \xi = X (n + i) \xi$ **unfolding**
j **using** *assms(2)[THEN AE-symmetric]* **by** *blast*
 have $AE \xi \text{ in } M. \text{cond-exp } M (F i) (X j) \xi = \text{cond-exp } M (F i) (\text{cond-exp } M$
 $(F (n + i)) (X j)) \xi$ **by** (*intro cond-exp-nested-subalg integrable subalg, simp add:*
subalgebra-def sets-F-mono)
 hence $AE \xi \text{ in } M. \text{cond-exp } M (F i) (X j) \xi = \text{cond-exp } M (F i) (X (n + i))$
 ξ **using** *cond-exp-cong-AE[OF integrable-cond-exp integrable *]* **by** *force*
 thus ?*case* **using** *Suc(1)[OF n]* **by** *fastforce*
 qed
qed (*simp add: integrable*)

lemma (*in nat-sigma-finite-adapted-process*) *martingale-of-cond-exp-diff-Suc-eq-zero*:
 assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$
 and $\bigwedge i. AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X (\text{Suc } i) \xi - X i \xi) \xi = 0$
 shows *nat-martingale* $M F X$
proof (*intro martingale-nat integrable*)
 fix i
 show $AE \xi \text{ in } M. X i \xi = \text{cond-exp } M (F i) (X (\text{Suc } i)) \xi$ **using** *cond-exp-diff[OF*
integrable(1,1), of i Suc i i] cond-exp-F-meas[OF integrable adapted, of i] assms(2)[of

$i]$ by *fastforce*
 qed

7.9 Discrete Time Submartingales

lemma (in *nat-submartingale*) *predictable-mono*:

assumes *nat-predictable-process* $M F X i \leq j$

shows $AE \xi$ in M . $X i \xi \leq X j \xi$

using *assms(2)*

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

case 0

then show *?case* by *simp*

next

case (*Suc* n)

hence $*$: $n = j - \text{Suc } i$ by *linarith*

interpret S : *nat-adapted-process* $M F \lambda i. X (\text{Suc } i)$ by (*intro nat-predictable-process.adapted-Suc assms*)

have $\text{Suc } i \leq j$ using *Suc(2,3)* by *linarith*

thus *?case* using *Suc(1)[OF *]* *S.adapted[of i]* *submartingale-property[OF - le-SucI, of i]* *sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i]* by *fastforce*

qed

lemma (in *nat-sigma-finite-adapted-process-linorder*) *submartingale-of-set-integral-le-Suc*:

assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$

and $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X (\text{Suc } i))$

shows *nat-submartingale* $M F X$

proof (*intro nat-submartingale.intro submartingale-of-set-integral-le*)

fix $i j A$ **assume** *asm*: $i \leq j$ $A \in \text{sets } (F i)$

show $\text{set-lebesgue-integral } M A (X i) \leq \text{set-lebesgue-integral } M A (X j)$ using *asm*

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

case 0

then show *?case* using *asm* by *simp*

next

case (*Suc* n)

hence $*$: $n = j - \text{Suc } i$ by *linarith*

have $\text{Suc } i \leq j$ using *Suc(2,3)* by *linarith*

thus *?case* using *sets-F-mono[OF - le-SucI]* *Suc(4)* *Suc(1)[OF *]* by (*auto intro: assms(2)[THEN order-trans]*)

qed

qed (*simp add: integrable*)

lemma (in *nat-sigma-finite-adapted-process-linorder*) *submartingale-nat*:

assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$

and $\bigwedge i. AE \xi$ in M . $X i \xi \leq \text{cond-exp } M (F i) (X (\text{Suc } i)) \xi$

shows *nat-submartingale* $M F X$

using *subalg integrable assms(2)*

by (intro submartingale-of-set-integral-le-Suc ord-le-eq-trans[OF set-integral-mono-AE-banach cond-exp-set-integral[symmetric]], simp)

(meson in-mono integrable-mult-indicator set-integrable-def subalgebra-def, meson integrable-cond-exp in-mono integrable-mult-indicator set-integrable-def subalgebra-def, fast+)

lemma (in nat-sigma-finite-adapted-process-linorder) submartingale-of-cond-exp-diff-Suc-nonneg:

assumes integrable: $\bigwedge i. \text{integrable } M (X i)$

and $\bigwedge i. AE \xi \text{ in } M. \text{cond-exp } M (F i) (\lambda \xi. X (Suc i) \xi - X i \xi) \xi \geq 0$

shows nat-submartingale $M F X$

proof (intro submartingale-nat integrable)

fix i

show $AE \xi \text{ in } M. X i \xi \leq \text{cond-exp } M (F i) (X (Suc i)) \xi$ using cond-exp-diff[OF integrable(1,1), of i Suc i] cond-exp-F-meas[OF integrable adapted, of i] assms(2)[of i] by fastforce

qed

lemma (in nat-submartingale-linorder) partial-sum-scaleR:

assumes nat-adapted-process $M F C \bigwedge i. AE \xi \text{ in } M. 0 \leq C i \xi \bigwedge i. AE \xi \text{ in } M. C i \xi \leq R$

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

proof –

interpret $C: \text{nat-adapted-process } M F C$ by (rule assms)

interpret $C': \text{nat-adapted-process } M F \lambda i \xi. C (i - 1) \xi *_{R} (X i \xi - X (i - 1) \xi)$ by (intro nat-adapted-process.intro adapted-process.scaleR-right-adapted adapted-process.diff-adapted, unfold-locales) (auto intro: adaptedD C.adaptedD)+

interpret $C'': \text{nat-adapted-process } M F \lambda n \xi. \sum i < n. C i \xi *_{R} (X (Suc i) \xi - X i \xi)$ by (rule $C'.\text{partial-sum-Suc-adapted[unfolded diff-Suc-1]}$)

interpret $S: \text{nat-sigma-finite-adapted-process-linorder } M F (\lambda n \xi. \sum i < n. C i \xi *_{R} (X (Suc i) \xi - X i \xi))$..

have integrable $M (\lambda x. C i x *_{R} (X (Suc i) x - X i x))$ for i using assms(2,3)[of i] by (intro Bochner-Integration.integrable-bound[OF integrable-scaleR-right, OF Bochner-Integration.integrable-diff, OF integrable(1,1), of R Suc i i]) (auto simp add: mult-mono)

moreover have $AE \xi \text{ in } M. 0 \leq \text{cond-exp } M (F i) (\lambda \xi. (\sum i < Suc i. C i \xi *_{R} (X (Suc i) \xi - X i \xi)) - (\sum i < i. C i \xi *_{R} (X (Suc i) \xi - X i \xi))) \xi$ 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(2,3)[of i]

by (fastforce simp add: scaleR-nonneg-nonneg integrable)

ultimately show ?thesis by (intro S.submartingale-of-cond-exp-diff-Suc-nonneg Bochner-Integration.integrable-sum, blast+)

qed

lemma (in nat-submartingale-linorder) partial-sum-scaleR':

assumes nat-predictable-process $M F C \bigwedge i. AE \xi \text{ in } M. 0 \leq C i \xi \bigwedge i. AE \xi \text{ in } M. C i \xi \leq R$

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

proof –

interpret C : *nat-predictable-process* $M F C$ **by** (*rule assms*)

interpret $Suc\text{-}C$: *nat-adapted-process* $M F \lambda i. C (Suc i)$ **using** $C.adapted\text{-}Suc$.

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

qed

7.10 Discrete Time Supermartingales

lemma (*in nat-supermartingale*) *predictable-mono*:

assumes *nat-predictable-process* $M F X i \leq j$

shows $AE \xi \text{ in } M. X i \xi \geq X j \xi$

using *assms*(2)

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

case 0

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

next

case ($Suc n$)

hence $*$: $n = j - Suc i$ **by** *linarith*

interpret S : *nat-adapted-process* $M F \lambda i. X (Suc i)$ **by** (*intro nat-predictable-process.adapted-Suc assms*)

have $Suc i \leq j$ **using** $Suc(2,3)$ **by** *linarith*

thus *?case* **using** $Suc(1)$ [*OF **] $S.adapted$ [*of i*] *supermartingale-property*[*OF - le-SucI, of i*] *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable, of F i Suc i*] **by** *fastforce*

qed

lemma (*in nat-sigma-finite-adapted-process-linorder*) *supermartingale-of-set-integral-ge-Suc*:

assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$

and $\bigwedge A i. A \in F i \implies \text{set-lebesgue-integral } M A (X i) \geq \text{set-lebesgue-integral } M A (X (Suc i))$

shows *nat-supermartingale* $M F X$

proof –

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

interpret $uminus\text{-}X$: *nat-sigma-finite-adapted-process-linorder* $M F -X$..

note $*$ = *set-integral-uminus*[*unfolded set-integrable-def, OF integrable-mult-indicator*[*OF - integrable*]]

have *nat-supermartingale* $M F (-(- X))$

using *ord-eq-le-trans*[*OF * ord-le-eq-trans*[*OF le-imp-neg-le*[*OF assms*(2)] $*$ [*symmetric*]]] *subalgebras*

by (*intro nat-supermartingale.intro submartingale.uminus nat-submartingale.axioms uminus-X.submartingale-of-set-integral-le-Suc*)

(*clarsimp simp add: fun-Compl-def subalgebra-def integrable | fastforce*)+

thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*

qed

lemma (*in nat-sigma-finite-adapted-process-linorder*) *supermartingale-nat*:

assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$

and $\bigwedge i. AE \xi \text{ in } M. X i \xi \geq \text{cond-exp } M (F i) (X (Suc i)) \xi$

shows *nat-supermartingale* $M F X$

proof –

interpret -: *adapted-process* $M F 0 -X$ **by** (*rule* *uminus-adapted*)
interpret *uminus-X*: *nat-sigma-finite-adapted-process-linorder* $M F -X$..
have $AE \xi$ *in* M . $- X i \xi \leq \text{cond-exp } M (F i) (\lambda x. - X (Suc i) x) \xi$ **for** i **using**
assms(2) *cond-exp-uminus*[*OF integrable, of i Suc i*] **by** *force*
hence *nat-supermartingale* $M F (-(- X))$ **by** (*intro* *nat-supermartingale.intro*
submartingale.uminus *nat-submartingale.axioms* *uminus-X.submartingale-nat*) (*auto*
simp *add: fun-Compl-def integrable*)
thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

lemma (*in* *nat-sigma-finite-adapted-process-linorder*) *supermartingale-of-cond-exp-diff-Suc-le-zero*:

assumes *integrable*: $\bigwedge i. \text{integrable } M (X i)$
and $\bigwedge i. AE \xi$ *in* M . *cond-exp* $M (F i) (\lambda \xi. X (Suc i) \xi - X i \xi) \xi \leq 0$
shows *nat-supermartingale* $M F X$

proof (*intro* *supermartingale-nat integrable*)

fix i
show $AE \xi$ *in* M . $X i \xi \geq \text{cond-exp } M (F i) (X (Suc i)) \xi$ **using** *cond-exp-diff*[*OF*
integrable(1,1), of i Suc i i] *cond-exp-F-meas*[*OF integrable adapted, of i*] *assms(2)*[*of*
i] **by** *fastforce*
qed

end

theory *Example-Coin-Toss*

imports *Martingale HOL-Probability.Stream-Space HOL-Probability.Probability-Mass-Function*
begin

8 Example: Coin Toss

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

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

the gamblers fortune decreases on average. Therefore, $(X_n)_{n \in \mathbb{N}}$ is a super-martingale.

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

definition *bernoulli-stream* :: *real* \Rightarrow (*bool stream*) *measure* **where**
bernoulli-stream $p = \text{stream-space } (\text{measure-pmf } (\text{bernoulli-pmf } p))$

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

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

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

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

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

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

proof –

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

qed

lemma *vimage-toss*: *toss* $n -^{\cdot} A = (\text{if } 1 \in A \text{ then } \{s. s !! n\} \text{ else } \{\}) \cup (\text{if } -1 \in A \text{ then } \{s. \neg s !! n\} \text{ else } \{\})$

unfolding *vimage-def toss-def* **by** *auto*

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

lemma *fortune-toss-sum*: *fortune* $n \ s = (\sum i \in \{..n\}. \text{toss } i \ s)$

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

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

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

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

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

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

interpretation *toss*: *nat-adapted-process bernoulli-stream p nat-natural-filtration (bernoulli-stream p) toss toss*
by (*intro nat-adapted-process.intro stochastic-process.adapted-process-natural-filtration*)
(unfold-locales, auto simp add: toss-def bernoulli-stream-def)

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

interpretation *fortune*: *nat-finite-adapted-process-linorder bernoulli-stream p nat-natural-filtration (bernoulli-stream p) toss fortune*

proof –

show *nat-finite-adapted-process-linorder (bernoulli-stream p) (toss-filtration p) fortune*

unfolding *fortune-toss-sum*

by (*intro nat-finite-adapted-process-linorder.intro*
finite-adapted-process-linorder.intro
finite-adapted-process-order.intro
finite-adapted-process.intro
toss.partial-sum-adapted[folded atMost-atLeast0] intro-locales)

qed

lemma *integrable-toss*: *integrable (bernoulli-stream p) (toss n)*

using *toss.random-variable*

by (*intro Bochner-Integration.integrable-bound[OF integrable-const[of - 1 :: real]]*)
(auto simp add: toss-def)

lemma *integrable-fortune*: *integrable (bernoulli-stream p) (fortune n) using fortune-bound*

by (*intro Bochner-Integration.integrable-bound[OF integrable-const[of - Suc n] fortune.random-variable]*) *auto*

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

lemma *measure-bernoulli-stream-snth-pred*:

assumes $0 \leq p$ **and** $p \leq 1$ **and** *finite J*

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

proof –

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

have *product-prob-space* $(\lambda-. \text{measure-pmf } (\text{bernoulli-pmf } p))$ **by** *unfold-locales*

hence $*$: *to-stream* $- \{s. \forall i \in J. P i = s !! i\} = \{s. \forall i \in J. P i = s i\}$ **using** *assms* **by** (*simp add: to-stream-def*)

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

proof –

{

fix s **assume** $(\forall i \in J. P\ i = s\ i)$
hence $(\forall i \in J. P\ i = s\ i) = (s \in \text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf } p)))\ J\ ?PiE)$
by $(\text{subst prod-emb-iff[of } s])\ (\text{smt (verit, best) not-def assms(3) id-def PiE-eq-singleton UNIV-I extensional-UNIV insert-iff singletonD space-measure-pmf})$
}
moreover
{
fix s **assume** $\neg(\forall i \in J. P\ i = s\ i)$
then obtain i **where** $i \in J\ P\ i \neq s\ i$ **by** *blast*
hence $(\forall i \in J. P\ i = s\ i) = (s \in \text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf } p)))\ J\ ?PiE)$
by $(\text{simp add: restrict-def prod-emb-iff[of } s])\ (\text{smt (verit, ccfv-SIG) PiE-mem assms(3) id-def insert-iff singleton-iff})$
}
ultimately show *?thesis by auto*
qed
finally have $\text{inteq: (to-stream -' \{s. \forall i \in J. P\ i = s\ !!\ i\}) = prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf } p))\ J\ ?PiE .$
let $?M = (Pi_M\ UNIV\ (\lambda-. \text{measure-pmf (bernoulli-pmf } p)))$
have $\text{emeasure (bernoulli-stream } p)\ \{s \in \text{space (bernoulli-stream } p).\ \forall i \in J. P\ i = s\ !!\ i\} = \text{emeasure } ?M\ (\text{to-stream -' \{s. \forall i \in J. P\ i = s\ !!\ i\}})$
using $\text{assms emeasure-distr[of to-stream } ?M\ (\text{vimage-algebra (streams (space (measure-pmf (bernoulli-pmf } p))))\ (!)\ ?M)\ \{s. \forall i \in J. P\ i = s\ !!\ i\}, \text{symmetric] measurable-to-stream[of (measure-pmf (bernoulli-pmf } p))}]$
by $(\text{simp only: bernoulli-stream-def stream-space-def *, simp add: space-PiM})\ (\text{smt (verit, best) emeasure-notin-sets in-vimage-algebra inf-top.right-neutral sets-distr vimage-Collect})$
also have $\dots = \text{emeasure } ?M\ (\text{prod-emb UNIV } (\lambda-. \text{measure-pmf (bernoulli-pmf } p)))\ J\ ?PiE$ **using** *inteq by (simp add: space-PiM)*
also have $\dots = (\prod_{i \in J. \text{emeasure (measure-pmf (bernoulli-pmf } p))\ (if\ P\ i\ then\ \{True\}\ else\ \{False\})})$
by $(\text{subst emeasure-PiM-emb})\ (\text{auto simp add: prob-space-measure-pmf assms(3)})$
also have $\dots = (\prod_{i \in J \cap \text{Collect } P. \text{ennreal } p}) * (\prod_{i \in J - \text{Collect } P. \text{ennreal } (1 - p)})$
unfolding $\text{emeasure-pmf-single[of bernoulli-pmf } p\ True, \text{unfolded pmf-bernoulli-True[OF assms(1,2)]}, \text{symmetric}]$
 $\text{emeasure-pmf-single[of bernoulli-pmf } p\ False, \text{unfolded pmf-bernoulli-False[OF assms(1,2)]}, \text{symmetric}]$
by $(\text{simp add: prod.Int-Diff[OF assms(3), of - Collect } P])$
also have $\dots = p^{\text{card } (J \cap \text{Collect } P)} * (1 - p)^{\text{card } (J - \text{Collect } P)}$ **using** *assms by (simp add: prod-ennreal ennreal-mult' ennreal-power)*
finally show *?thesis using assms by (intro measure-eq-emeasure-eq-ennreal) auto*
qed

lemma

assumes $0 \leq p$ **and** $p \leq 1$

shows $\text{measure-bernoulli-stream-snth: prob } p\ \{w \in \text{space (bernoulli-stream } p).\ w$

```

!! i} = p
  and measure-bernoulli-stream-neg-snth: prob p {w ∈ space (bernoulli-stream p)}.
¬w !! i} = 1 - p
  using measure-bernoulli-stream-snth-pred[OF assms, of {i} λx. True]
      measure-bernoulli-stream-snth-pred[OF assms, of {i} λx. False] by auto

```

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

```

lemma integral-toss:
  assumes 0 ≤ p p ≤ 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 $\neg(\exists j \in J. 1 \notin B j \wedge -1 \notin B j)$
hence $*$: $1 \in B j \vee -1 \in B j$ **if** $j \in J$ **for** j **using** *that* **by** *blast*

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

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

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

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

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

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

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

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

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

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

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

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

finally show *?thesis* .

qed

}

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

by (*intro indep-setsI subsetI*) *fastforce*

qed (*simp, intro Int-stableI, simp, metis sets.Int vimage-Int*)

The fortune of a player is a martingale (resp. sub- or supermartingale) with

respect to the filtration generated by the coin tosses.

theorem *fortune-martingale*:

assumes $p = 1/2$

shows *nat-martingale* (*bernoulli-stream* p) (*toss-filtration* p) *fortune*

using *cond-exp-indep*[*OF* *fortune.subalg indep-set-natural-filtration integrable-toss*,
OF zero-order(1) lessI indep-vars-toss, of p]

integral-toss *assms*

by (*intro fortune.martingale-of-cond-exp-diff-Suc-eq-zero integrable-fortune*)
(*force simp add: fortune-toss-sum*)

theorem *fortune-submartingale*:

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

shows *nat-submartingale* (*bernoulli-stream* p) (*toss-filtration* p) *fortune*

proof (*intro fortune.submartingale-of-cond-exp-diff-Suc-nonneg integrable-fortune*)

fix n

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

using *cond-exp-indep*[*OF* *fortune.subalg indep-set-natural-filtration integrable-toss*,
OF zero-order(1) lessI indep-vars-toss, of p n]

integral-toss[of p *Suc* n] *assms*

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

qed

theorem *fortune-supermartingale*:

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

shows *nat-supermartingale* (*bernoulli-stream* p) (*toss-filtration* p) *fortune*

proof (*intro fortune.supermartingale-of-cond-exp-diff-Suc-le-zero integrable-fortune*)

fix n

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

using *cond-exp-indep*[*OF* *fortune.subalg indep-set-natural-filtration integrable-toss*,
OF zero-order(1) lessI indep-vars-toss, of p n]

integral-toss[of p *Suc* n] *assms*

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

qed

end

References

- [1] R. Engelking. *General Topology*. Sigma series in pure mathematics. Heldermann, 1989.
- [2] 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.
- [3] A. Keskin. A formalization of martingales in Isabelle/HOL. 2023.

- [4] S. Lang. *Real and Functional Analysis*. Springer, 1993.
- [5] 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.