Doob's Upcrossing Inequality and Martingale Convergence Theorem

Ata Keskin

March 29, 2024

Abstract

In this entry, we formalize Doob's upcrossing inequality and subsequently prove Doob's first martingale convergence theorem. The upcrossing inequality is a fundamental result in the study of martingales. It provides a bound on the expected number of times a submartingale crosses a certain threshold within a given interval. Doob's martingale convergence theorem states that, if we have a submartingale where the supremum over the mean of the positive parts is finite, then the limit process exists almost surely and is integrable. Equivalent statements for martingales and supermartingales are also provided as corollaries.

The proofs provided are based mostly on the formalization done in the Lean mathematical library [1,2].

Contents

1 Introduction

Martingales, in the context of stochastic processes, are encountered in various real-world scenarios where outcomes are influenced by past events but are not entirely predictable due to randomness or uncertainty. A martingale is a stochastic process in which the expected value of the next observation, given all past observations, is equal to the current observation.

One real-world example can be encountered in environmental monitoring, particularly in the study of river flow rates. Consider a hydrologist tasked with monitoring the flow rate of a river to understand its behavior over time. The flow rate of a river is influenced by various factors such as rainfall, snowmelt, groundwater levels, and human activities like dam releases or water diversions. These factors contribute to the variability and unpredictability of the flow rate. In this scenario, the flow rate of the river can be modeled as a martingale. The flow rate at any given time is influenced by past events but is not entirely predictable due to the random nature of rainfall and other factors.

One concept that comes up frequently in the study of martingales are upcrossings and downcrossings. Upcrossings and downcrossings are random variables representing when the value of a stochastic process leaves a fixed interval. Specifically, an upcrossing occurs when the process moves from below the lower bound of the interval to above the upper bound [4], indicating a potential upward trend or positive movement. Conversely, a downcrossing happens when the process crosses below the lower bound of the interval, suggesting a potential downward trend or negative movement. By analyzing the frequency and timing of these crossings, researchers can infer information about the underlying dynamics of the process and detect shifts in its behavior.

For instance, consider tracking the movement of a stock price over time. The process representing the stock's price might cross above a certain threshold (upcrossing) or below it (downcrossing) multiple times during a trading session. The number of such crossings provides insights into the volatility and the trend of the stock.

Doob's upcrossing inequality is a fundamental result in the study of martingales. It provides a bound on the expected number of upcrossings a submartingale undertakes before some point in time.

Let's consider our example concerning river flow rates again. In this context, upcrossings represent instances where the flow rate of the river rises above a certain threshold. For example, the flow rate might cross a threshold indicating flood risk. Downcrossings, on the other hand, represent instances where the flow rate decreases below a certain threshold. This could indicate drought conditions or low-flow periods.

Doob's first martingale convergence theorem gives sufficient conditions for a submartingale to converge to a random variable almost surely. The proof is based on controlling the rate of growth or fluctuations of the submartingale, which is where the *upcrossing inequality* comes into play. By bounding these fluctuations, we can ensure that the submartingale does not exhibit wild behavior or grow too quickly, which is essential for proving convergence.

Formally, the convergence theorem states that, if $(M_n)_{n\geq 0}$ is a submartingale with $\sup_n \mathbb{E}[M_n^+] < \infty$, where M_n^+ denotes the positive part of M_n, then the limit process $M_\infty := \lim_n M_n$ exists almost surely and is integrable. Furthermore, the limit process is measurable with respect to the smallest σ-algebra containing all of the σ-algebras in the filtration. In our formalization, we also show equivalent convergence statements for martingales and supermartingales. The theorem can be used to easily show convergence results for simple scenarios.

Consider the following example: Imagine a casino game where a player bets on the outcome of a random coin toss, where the coin comes up heads with odds $p \in [0, \frac{1}{2})$. Assume that the player goes bust when they have no money remaining. The player's wealth over time can be modeled as a supermartingale, where the value of their wealth at each time step depends only on the outcome of the previous coin toss. Doob's martingale convergence theorem assures us that the player will go bankrupt as the number of coin tosses increases.

The theorem that we have described here and formalized in the scope of our project is called *Doob's first martingale convergence theorem*. It is important to note that the convergence in this theorem is pointwise, not uniform, and is unrelated to convergence in mean square, or indeed in any L^p space. In order to obtain convergence in L^1 (i.e., convergence in mean), one requires uniform integrability of the random variables. In this form, the theorem is called *Doob's second martingale convergence theorem*. Since uniform integrability is not yet formalized in Isabelle/HOL, we have decided to confine our formalization to the first convergence theorem only.

2 Updates for the entry Martingales

This section contains the changes done for the entry Martingales [7]. We simplified the locale hierarchy by removing unnecessary locales and moving lemmas under more general locales where possible. We have to redefine almost all of the constants, in order to make sure we use the new locale hierarchy. The changes will be incorporated into the entry Martingales [7] and this file will be removed when the next Isabelle version rolls out.

theory *Martingales-Updates*
imports *Martingales.Martingale*
begin

2.1 Updates for *Martingales.Filtered-Measure*

lemma (**in** *filtered-measure*) *sets-F-subset*[*simp*]:
  assumes $t_0 \leq t$
  shows *sets* (*F t*) ⊆ *sets M*
  using *subalgebras assms* **by** (*simp add*: *subalgebra-def*)

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

context *linearly-filtered-measure*
begin

— We define F_∞ to be the smallest σ-algebra containing all the σ-algebras in the filtration.

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

notation *F-infinity* (‹F_∞›)

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

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

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

lemma *F-infinity-subset*: F_∞ ⊆ *M*
  unfolding *sets-F-infinity* **using** *sets-F-subset*
  by (*simp add*: *SUP-le-iff sets.sigma-sets-subset*)

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

end

locale *nat-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *nat* ⇒ -
locale *enat-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *enat* ⇒ -
locale *real-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *real* ⇒ -
locale *ennreal-filtered-measure* = *linearly-filtered-measure M F 0* **for** *M* **and** *F* :: *ennreal* ⇒ -

locale *nat-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure M F 0* :: *nat* **for** *M F*
locale *enat-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure M F 0* :: *enat* **for** *M F*
locale *real-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure M F 0* :: *real* **for** *M F*
locale *ennreal-sigma-finite-filtered-measure* = *sigma-finite-filtered-measure M F 0* :: *ennreal* **for** *M F*

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

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

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

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

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

M F

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

2.2 Updates for *Martingales.Stochastic-Process*

lemma (**in** *nat-filtered-measure*) *partial-sum-Suc-adapted*:
  assumes *adapted-process M F 0 X*
  shows *adapted-process M F 0* ($\lambda n\ \xi.\ \sum i{<}n.\ X\ (Suc\ i)\ \xi$)
proof (*unfold-locales*)
  interpret *adapted-process M F 0 X* **using** *assms* **by** *blast*
  fix *i*
  have $X\ j \in$ *borel-measurable* ($F\ i$) **if** $j \le i$ **for** *j* **using** *that adaptedD* **by** *blast*
  thus ($\lambda\xi.\ \sum i{<}i.\ X\ (Suc\ i)\ \xi$) ∈ *borel-measurable* ($F\ i$) **by** *auto*
qed

lemma (**in** *enat-filtered-measure*) *partial-sum-eSuc-adapted*:
  assumes *adapted-process M F 0 X*
  shows *adapted-process M F 0* ($\lambda n\ \xi.\ \sum i{<}n.\ X\ (eSuc\ i)\ \xi$)
proof (*unfold-locales*)
  interpret *adapted-process M F 0 X* **using** *assms* **by** *blast*
  fix *i*
  have $X\ (eSuc\ j) \in$ *borel-measurable* ($F\ i$) **if** $j < i$ **for** *j* **using** *that adaptedD* **by** (*simp add*: *ileI1*)
  thus ($\lambda\xi.\ \sum i{<}i.\ X\ (eSuc\ i)\ \xi$) ∈ *borel-measurable* ($F\ i$) **by** *auto*
qed

lemma (**in** *filtered-measure*) *adapted-process-sum*:
  assumes $\bigwedge i.\ i \in I \Longrightarrow$ *adapted-process M F* t_0 ($X\ i$)
  shows *adapted-process M F* t_0 ($\lambda k\ \xi.\ \sum i \in I.\ X\ i\ k\ \xi$)
proof −
  {
    fix *i k* **assume** $i \in I$ **and** *asm*: $t_0 \le k$
    then interpret *adapted-process M F* t_0 *X i* **using** *assms* **by** *simp*
      have $X\ i\ k \in$ *borel-measurable M* $X\ i\ k \in$ *borel-measurable* ($F\ k$) **using** *measurable-from-subalg subalgebras adapted asm* **by** (*blast*, *simp*)
  }
  thus *?thesis* **by** (*unfold-locales*) *simp*
qed

context *linearly-filtered-measure*
begin

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

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

lemma *sets-predictable-sigma*: *sets* Σ_P = *sigma-sets* ({t_0..} × *space M*) ({{*s*<..*t*} × *A* | *A s t*. *A* ∈ *F s* ∧ t_0 ≤ *s* ∧ *s* < *t*} ∪ {{t_0} × *A* | *A*. *A* ∈ *F* t_0})
unfolding *predictable-sigma* **using** *space-F sets.sets-into-space* **by** (*subst sets-measure-of*) *fastforce*+

lemma *measurable-predictable-sigma-snd*:
assumes *countable* $\mathcal{I}$ $\mathcal{I}$ ⊆ {{*s*<..*t*} | *s t*. t_0 ≤ *s* ∧ *s* < *t*} {t_0<..} ⊆ (⋃$\mathcal{I}$)
shows *snd* ∈ Σ_P $\to_M$ *F* t_0
proof (*intro measurableI*)
fix *S* :: *'a set* **assume** *asm*: *S* ∈ *F* t_0
have *countable*: *countable* ((λ*I*. *I* × *S*) ' $\mathcal{I}$) **using** *assms*(*1*) **by** *blast*
have (λ*I*. *I* × *S*) ' $\mathcal{I}$ ⊆ {{*s*<..*t*} × *A* | *A s t*. *A* ∈ *F s* ∧ t_0 ≤ *s* ∧ *s* < *t*} **using** *sets-F-mono*[*OF order-refl*, *THEN subsetD*, *OF - asm*] *assms*(*2*) **by** *blast*
hence (⋃*I*∈$\mathcal{I}$. *I* × *S*) ∪ {t_0} × *S* ∈ Σ_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* ∩ *space* Σ_P = {t_0..} × *S* **using** *sets.sets-into-space*[*OF asm*] **by** *fastforce*
moreover have {t_0} ∪ {t_0<..} = {t_0..} **by** *auto*
moreover have (⋃*I*∈$\mathcal{I}$. *I* × *S*) ∪ {t_0} × *S* = {t_0..} × *S* **using** *assms*(*2*,*3*) *calculation*(*3*) **by** *fastforce*
ultimately show *snd* −' *S* ∩ *space* Σ_P ∈ Σ_P **by** *argo*
qed (*auto*)

lemma *measurable-predictable-sigma-fst*:
assumes *countable* $\mathcal{I}$ $\mathcal{I}$ ⊆ {{*s*<..*t*} | *s t*. t_0 ≤ *s* ∧ *s* < *t*} {t_0<..} ⊆ (⋃$\mathcal{I}$)
shows *fst* ∈ Σ_P $\to_M$ *borel*
proof −
have *A* × *space M* ∈ *sets* Σ_P **if** *A* ∈ *sigma-sets* {t_0..} {{*s*<..*t*} | *s t*. t_0 ≤ *s* ∧ *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*) × *space M* = {t_0..} × *space M* − *a* × *space M* **by** *blast*
then show *?case* **using** *Compl*(*2*)[*THEN sigma-sets.Compl*] **by** *presburger*
next
case (*Union a*)
have ⋃ (*range a*) × *space M* = ⋃ (*range* (λ*i*. *a i* × *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*} | *s t*. t_0 ≤ *s* ∧ *s* < *t*}
proof −

have *sigma-sets* $\{t_0..\}$ $((\cap)$ $\{t_0..\}$ ‘ *sigma-sets UNIV* (*range greaterThan*)) = *sigma-sets* $\{t_0..\}$ $\{\{s<..t\}$ $|s\ t.\ t_0 \le s \wedge s < t\}$
proof (*intro sigma-sets-eqI* ; *clarify*)
fix A :: *'b set* **assume** *asm*: $A \in$ *sigma-sets UNIV* (*range greaterThan*)
thus $\{t_0..\} \cap A \in$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\}$ $|s\ t.\ t_0 \le s \wedge s < t\}$
proof (*induction rule*: *sigma-sets.induct*)
case (*Basic a*)
then obtain s **where** s: $a = \{s<..\}$ **by** *blast*
show *?case*
proof (*cases* $t_0 \le s$)
case *True*
hence $*$: $\{t_0..\} \cap a = (\bigcup i \in \mathcal{I}.\ \{s<..\} \cap i)$ **using** *s assms*(*3*) **by** *force*
have $((\cap)$ $\{s<..\}$ ‘ $\mathcal{I}) \subseteq$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\}$ $|\ s\ t.\ t_0 \le s \wedge s < t\}$
proof (*clarify*)
fix A **assume** $A \in \mathcal{I}$
then obtain s' t' **where** A: $A = \{s'<..t'\}$ $t_0 \le s'$ $s' < t'$ **using** *assms*(*2*) **by** *blast*
hence $\{s<..\} \cap A = \{max\ s\ s'<..t'\}$ **by** *fastforce*
moreover have $t_0 \le max\ s\ s'$ **using** *A True* **by** *linarith*
moreover have $max\ s\ s' < t'$ **if** $s < t'$ **using** *A that* **by** *linarith*
moreover have $\{s<..\} \cap A = \{\}$ **if** $\neg\ s < t'$ **using** *A that* **by** *force*
ultimately show $\{s<..\} \cap A \in$ *sigma-sets* $\{t_0..\}$ $\{\{s<..t\}$ $|s\ t.\ t_0 \le s \wedge s < t\}$ **by** (*cases* $s < t'$) (*blast*, *simp add*: *sigma-sets.Empty*)
qed
thus *?thesis* **unfolding** $*$ **using** *assms*(*1*) **by** (*intro sigma-sets-UNION*) *auto*
next
case *False*
hence $\{t_0..\} \cap a = \{t_0..\}$ **using** *s* **by** *force*
thus *?thesis* **using** *sigma-sets-top* **by** *auto*
qed
next
case (*Compl a*)
have $\{t_0..\} \cap (UNIV - a) = \{t_0..\} - (\{t_0..\} \cap a)$ **by** *blast*
then show *?case* **using** *Compl*(*2*)[*THEN sigma-sets.Compl*] **by** *presburger*
next
case (*Union a*)
have $\{t_0..\} \cap \bigcup$ (*range a*) $= \bigcup$ (*range* $(\lambda i.\ \{t_0..\} \cap a\ i)$) **by** *blast*
then show *?case* **using** *Union*(*2*)[*THEN sigma-sets.Union*] **by** *presburger*
qed (*simp add*: *sigma-sets.Empty*)
next
fix s t **assume** *asm*: $t_0 \le s$ $s < t$
hence $*$: $\{s<..t\} = \{s<..\} \cap (\{t_0..\} - \{t<..\})$ **by** *force*
have $\{s<..\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)$ $\{t_0..\}$ ‘ *sigma-sets UNIV* (*range greaterThan*)) **using** *asm* **by** (*intro sigma-sets.Basic*) *auto*
moreover have $\{t_0..\} - \{t<..\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)$ $\{t_0..\}$ ‘ *sigma-sets UNIV* (*range greaterThan*)) **using** *asm* **by** (*intro sigma-sets.Compl sigma-sets.Basic*) *auto*
ultimately show $\{s<..t\} \in$ *sigma-sets* $\{t_0..\}$ $((\cap)$ $\{t_0..\}$ ‘ *sigma-sets*

UNIV (*range greaterThan*)) **unfolding** ∗ *Int-range-binary*[*of* {*s*<..}] **by** (*intro sigma-sets-Inter*[*OF* - *binary-in-sigma-sets*]) *auto*
qed
thus *?thesis* **unfolding** *borel-Ioi restrict-space-def emeasure-sigma* **by** (*force intro*: *sigma-eqI*)
qed
ultimately have *restrict-space borel* {t_0..} $\bigotimes_M$ *sigma* (*space M*) {} ⊆ *sets* Σ_P
unfolding *sets-pair-measure space-restrict-space space-measure-of-conv*
using *space-predictable-sigma sets.sigma-algebra-axioms*[*of* Σ_P]
by (*intro sigma-algebra.sigma-sets-subset*) (*auto simp add*: *sigma-sets-empty-eq sets-measure-of-conv*)
moreover have *space* (*restrict-space borel* {t_0..} $\bigotimes_M$ *sigma* (*space M*) {}) = *space* Σ_P **by** (*simp add*: *space-pair-measure*)
moreover have *fst* ∈ *restrict-space borel* {t_0..} $\bigotimes_M$ *sigma* (*space M*) {} $\to_M$ *borel* **by** (*fastforce intro*: *measurable-fst''*[*OF measurable-restrict-space1*, *of* λ*x*. *x*])
ultimately show *?thesis* **by** (*meson borel-measurable-subalgebra*)
qed

end

locale *predictable-process* = *linearly-filtered-measure M F* t_0 **for** *M F* t_0 **and** *X* :: - ⇒ - ⇒ - :: {*second-countable-topology*, *banach*} +
assumes *predictable*: (λ(*t*, *x*). *X t x*) ∈ *borel-measurable* Σ_P
begin

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

end

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

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

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

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

lemma (**in** *real-filtered-measure*) *measurable-predictable-sigma-snd'*:
shows *snd* ∈ Σ_P $\to_M$ *F 0*

using *real-arch-simple* **by** (*intro measurable-predictable-sigma-snd*[*of range* (λx::*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* (λx::*nat*. {*0*<..*real* (*Suc x*)})]) (*fastforce intro*: *add-increasing*)+

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

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

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

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

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

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

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

lemma (**in** *linearly-filtered-measure*) *predictable-process-const*:
assumes *fst* $\in$ *borel-measurable* Σ_P *c* $\in$ *borel-measurable borel*
shows *predictable-process M F* t_0 (λi -. *c i*)
using *assms* **by** (*unfold-locales*) (*simp add*: *measurable-split-conv*)

lemma (**in** *linearly-filtered-measure*) *predictable-process-const-const*[*intro*]:
shows *predictable-process M F* t_0 (λ- -. *c*)

by (*unfold-locales*) *simp*

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

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

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

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

context *predictable-process*
begin

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

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

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

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

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

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

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

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

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

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

end

sublocale *predictable-process* $\subseteq$ *progressive-process*
proof (*unfold-locales*)
fix i :: *'b* **assume** *asm*: $t_0 \le i$
{
fix S :: (*'b* $\times$ *'a*) *set* **assume** $S \in \{\{s<..t\} \times A \mid A\ s\ t.\ A \in F\ s \wedge t_0 \le s \wedge s < t\} \cup \{\{t_0\} \times A \mid A.\ A \in F\ t_0\}$
hence ($\lambda x.\ x$) $-$‘ $S \cap (\{t_0..i\} \times$ *space M*) $\in$ *restrict-space borel* $\{t_0..i\} \bigotimes_M F\ i$
proof
assume $S \in \{\{s<..t\} \times A \mid A\ s\ t.\ A \in F\ s \wedge t_0 \le s \wedge s < t\}$
then obtain $s\ t\ A$ **where** *S-is*: $S = \{s<..t\} \times A$ $t_0 \le s$ $s < t$ $A \in F\ s$ **by** *blast*
hence ($\lambda x.\ x$) $-$‘ $S \cap (\{t_0..i\} \times$ *space M*) $= \{s<..min\ i\ t\} \times A$ **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* $s \le i$) (*fastforce simp add*: *sets-restrict-space-iff*)+
next
assume $S \in \{\{t_0\} \times A \mid A.\ A \in F\ t_0\}$
then obtain A **where** *S-is*: $S = \{t_0\} \times A$ $A \in F\ t_0$ **by** *blast*
hence ($\lambda x.\ x$) $-$‘ $S \cap (\{t_0..i\} \times$ *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 ($\lambda x.\ x$) $-$‘ $S \cap$ *space* (*restrict-space borel* $\{t_0..i\}$ $\bigotimes_M$ *F i*) $\in$ *restrict-space borel* $\{t_0..i\}$ $\bigotimes_M$ *F i* **by** (*simp add*: *space-pair-measure space-F*[*OF asm*])
}
moreover have $\{\{s<..t\} \times A\ |A\ s\ t.\ A \in sets\ (F\ s) \wedge t_0 \leq s \wedge s < t\} \cup \{\{t_0\} \times A\ |A.\ A \in sets\ (F\ t_0)\} \subseteq Pow\ (\{t_0..\} \times space\ M)$ **using** *sets.sets-into-space* **by** *force*
ultimately have ($\lambda x.\ x$) $\in$ *restrict-space borel* $\{t_0..i\}$ $\bigotimes_M$ *F i* $\rightarrow_M \Sigma_P$ **using** *space-F*[*OF asm*] **by** (*intro measurable-sigma-sets*[*OF sets-predictable-sigma*]) (*fast*, *force simp add*: *space-pair-measure*)
thus *case-prod X* $\in$ *borel-measurable* (*restrict-space borel* $\{t_0..i\}$ $\bigotimes_M$ *F i*) **using** *predictable* **by** *simp*
qed

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

}
note $** = this$
show $A\ (Suc\ i) \in sets\ (F\ i)\ A\ 0 \in F\ 0$ **using** $*(2)[of\ i]\ *(1)\ **(2)[of\ i]\ **(1)$ **by** *blast+*
next
case *Empty*

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

ultimately show *?case* **using** *A-i* **by** *metis*
}
qed

lemma (**in** *nat-filtered-measure*) *predictable-implies-adapted-Suc*:
assumes *predictable-process M F 0 X*
shows *adapted-process M F 0* (λi. *X* (*Suc i*))
proof (*unfold-locales*, *intro borel-measurableI*)
interpret *predictable-process M F 0 X* **by** (*rule assms*)
fix *S* :: *'b set* **and** *i* **assume** *open-S*: *open S*
have {*Suc i*} = {*i*<..*Suc i*} **by** *fastforce*
hence {*Suc i*} × *space M* ∈ Σ_P **using** *space-F*[*symmetric*, *of i*] **unfolding** *sets-predictable-sigma* **by** (*intro sigma-sets.Basic*) *blast*
moreover have *case-prod X* −' *S* ∩ (*UNIV* × *space M*) ∈ Σ_P **unfolding** *atLeast-0*[*symmetric*] **using** *open-S* **by** (*intro predictableD*, *simp add*: *borel-open*)
ultimately have *case-prod X* −' *S* ∩ ({*Suc i*} × *space M*) ∈ Σ_P **unfolding** *sets-predictable-sigma* **using** *space-F sets.sets-into-space*
by (*subst Times-Int-distrib1*[*of* {*Suc i*} *UNIV space M*, *simplified*], *subst inf.commute*, *subst Int-assoc*[*symmetric*], *subst Int-range-binary*)
(*intro sigma-sets-Inter binary-in-sigma-sets*, *fast*)+
moreover have *case-prod X* −' *S* ∩ ({*Suc i*} × *space M*) = {*Suc i*} × (*X* (*Suc i*) −' *S* ∩ *space M*) **by** (*auto simp add*: *le-Suc-eq*)
moreover have ... = (⋃*j*. {*j*} × (*if j* = *Suc i then* (*X* (*Suc i*) −' *S* ∩ *space M*) *else* {})) **by** (*force split*: *if-splits*)
ultimately have (⋃*j*. {*j*} × (*if j* = *Suc i then* (*X* (*Suc i*) −' *S* ∩ *space M*) *else* {})) ∈ Σ_P **by** *argo*
thus *X* (*Suc i*) −' *S* ∩ *space* (*F i*) ∈ *sets* (*F i*) **using** *sets-in-filtration*[*of* λj. *if j* = *Suc i then* (*X* (*Suc i*) −' *S* ∩ *space M*) *else* {}] *space-F*[*OF zero-le*] **by** *presburger*
qed

theorem (**in** *nat-filtered-measure*) *predictable-process-iff*: *predictable-process M F 0 X* ⟷ *adapted-process M F 0* (λi. *X* (*Suc i*)) ∧ *X 0* ∈ *borel-measurable* (*F 0*)
proof (*intro iffI*)
assume *asm*: *adapted-process M F 0* (λi. *X* (*Suc i*)) ∧ *X 0* ∈ *borel-measurable* (*F 0*)
interpret *adapted-process M F 0* λi. *X* (*Suc i*) **using** *asm* **by** *blast*
have (λ(*x*, *y*). *X x y*) ∈ *borel-measurable* Σ_P
proof (*intro borel-measurableI*)
fix *S* :: *'b set* **assume** *open-S*: *open S*
have {*i*} × (*X i* −' *S* ∩ *space M*) ∈ *sets* Σ_P **for** *i*
proof (*cases i*)
case *0*
then show *?thesis* **unfolding** *sets-predictable-sigma*
using *measurable-sets*[*OF* - *borel-open*[*OF open-S*], *of X 0 F 0*] *asm* **by** *auto*
next
case (*Suc i*)
have {*Suc i*} = {*i*<..*Suc i*} **by** *fastforce*
then show *?thesis* **unfolding** *sets-predictable-sigma*

using *measurable-sets*[*OF adapted borel-open*[*OF open-S*], *of i*]
by (*intro sigma-sets.Basic*, *auto simp add*: *Suc*)
qed
moreover have $(\lambda(x, y).\ X\ x\ y)$ −‘ $S \cap$ *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* Σ_P **by** *simp*
qed
thus *predictable-process M F 0 X* **by** (*unfold-locales*)
next
assume *asm*: *predictable-process M F 0 X*
interpret *predictable-process M F 0 X* **using** *asm* **by** *blast*
show *adapted-process M F 0* $(\lambda i.\ X\ (Suc\ i)) \wedge X\ 0 \in$ *borel-measurable* (*F 0*)
using *predictable-implies-adapted-Suc asm* **by** *auto*
qed

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

2.3 Updates for *Martingales.Martingale*

locale *martingale* = *sigma-finite-filtered-measure* + *adapted-process* +
assumes *integrable*: $\bigwedge i.\ t_0 \le i \Longrightarrow$ *integrable M* (*X i*)
and *martingale-property*: $\bigwedge i\ j.\ t_0 \le i \Longrightarrow i \le j \Longrightarrow$ *AE* ξ *in M*. $X\ i\ \xi$ = *cond-exp M* (*F i*) (*X j*) ξ

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

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

lemma (**in** *sigma-finite-filtered-measure*) *martingale-cond-exp*[*intro*]:
assumes *integrable M f*
shows *martingale M F* t_0 (λ*i*. *cond-exp M* (*F i*) *f*)
using *sigma-finite-subalgebra.borel-measurable-cond-exp*′ *borel-measurable-cond-exp*

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

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

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

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

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

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

lemma (**in** *sigma-finite-filtered-measure*) *submartingale-cond-exp*[*intro*]:
assumes *integrable M f*
shows *submartingale M F* t_0 (λi. *cond-exp M* (*F i*) *f*)
proof −
interpret *martingale M F* t_0 λi. *cond-exp M* (*F i*) *f* **using** *assms* **by** (*rule martingale-cond-exp*)
show *submartingale M F* t_0 (λi. *cond-exp M* (*F i*) *f*) **using** *martingale-property* **by** (*unfold-locales*) (*force simp add*: *integrable*)+
qed

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

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

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

locale *supermartingale-linorder* = *supermartingale M F* t_0 *X* **for** *M F* t_0 **and** *X* ::

- ⇒ - ⇒ - :: {*linorder-topology*}

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-const-fun*[*intro*]:
assumes *integrable M f f* ∈ *borel-measurable* (*F* t_0)
shows *supermartingale M F* t_0 (λ-. *f*)
proof −
interpret *martingale M F* t_0 λ-. *f* **using** *assms* **by** (*rule martingale-const-fun*)
show *supermartingale M F* t_0 (λ-. *f*) **using** *martingale-property* **by** (*unfold-locales*) (*force simp add*: *integrable*)+
qed

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-cond-exp*[*intro*]:
assumes *integrable M f*
shows *supermartingale M F* t_0 (λ*i*. *cond-exp M* (*F i*) *f*)
proof −
interpret *martingale M F* t_0 λ*i*. *cond-exp M* (*F i*) *f* **using** *assms* **by** (*rule martingale-cond-exp*)
show *supermartingale M F* t_0 (λ*i*. *cond-exp M* (*F i*) *f*) **using** *martingale-property* **by** (*unfold-locales*) (*force simp add*: *integrable*)+
qed

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

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

lemma *martingale-iff*:
shows *martingale M F* t_0 *X* ⟷ *submartingale M F* t_0 *X* ∧ *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* ∧ *supermartingale M F* t_0 *X* **using** *submartingale-axioms supermartingale-axioms* **by** *blast*
next
assume *asm*: *submartingale M F* t_0 *X* ∧ *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

context *martingale*
begin

lemma *cond-exp-diff-eq-zero*:
assumes $t_0 \leq i$ $i \leq j$

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

lemma *set-integral-eq*:
assumes $A \in F\ i\ t_0 \le i\ i \le j$
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 $\int x \in A.\ X\ i\ x\ \partial M = \int x \in A.$ *cond-exp M* (*F i*) (*X j*) *x* ∂M **using** *martingale-property*[*OF assms*(*2*,*3*)] *borel-measurable-cond-exp′ assms subalgebras subalgebra-def* **by** (*intro set-lebesgue-integral-cong-AE*[*OF - random-variable*]) *fastforce+*
also have ... = $\int x \in A.\ X\ j\ x\ \partial M$ **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*: $t_0 \le i\ i \le j$
interpret *sigma-finite-subalgebra M F i* **using** *asm* **by** *blast*
have *AE x in M.* $c *_R X\ i\ x$ = *cond-exp M* (*F i*) ($\lambda x.\ c *_R X\ j\ x$) *x* **using** *asm cond-exp-scaleR-right*[*OF integrable*, *of j*, *THEN AE-symmetric*] *martingale-property*[*OF asm*] **by** *force*
}
thus *?thesis* **by** (*unfold-locales*) (*auto simp add*: *integrable martingale.integrable*)
qed

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

lemma *add*[*intro*]:
assumes *martingale M F* t_0 *Y*
shows *martingale M F* t_0 ($\lambda i\ \xi$. *X i* ξ + *Y i* ξ)
proof −
interpret *Y*: *martingale M F* t_0 *Y* **by** (*rule assms*)
{
fix *i j* :: *′b* **assume** *asm*: $t_0 \le i\ i \le j$
hence *AE* ξ *in M. X i* ξ + *Y i* ξ = *cond-exp M* (*F i*) (λx. *X j x* + *Y j x*) ξ
using *sigma-finite-subalgebra.cond-exp-add*[*OF - integrable martingale.integrable*[*OF assms*], *of F i j j*, *THEN AE-symmetric*]
martingale-property[*OF asm*] *martingale.martingale-property*[*OF assms asm*] **by** *force*

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

lemma *diff*[*intro*]:
assumes *martingale M F* t_0 *Y*
shows *martingale M F* t_0 ($\lambda i\ x.\ X\ i\ x - Y\ i\ x$)
proof −
interpret *Y*: *martingale M F* t_0 *Y* **by** (*rule assms*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
hence *AE* ξ *in M*. $X\ i\ \xi - Y\ i\ \xi$ = *cond-exp M* (*F i*) ($\lambda x.\ X\ j\ x - Y\ j\ x$) ξ
using *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable martingale.integrable*[*OF assms*], *of F i j j*, *THEN AE-symmetric*]
martingale-property[*OF asm*] *martingale.martingale-property*[*OF assms asm*] **by** *fastforce*
}
thus *?thesis* **using** *assms* **by** (*unfold-locales*) (*auto simp add*: *integrable martingale.integrable*)
qed

end

lemma (**in** *sigma-finite-filtered-measure*) *martingale-of-cond-exp-diff-eq-zero*:
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \le i \Longrightarrow$ *integrable M* (*X i*)
and *diff-zero*: $\bigwedge i\ j.\ t_0 \le i \Longrightarrow i \le j \Longrightarrow$ *AE x in M*. *cond-exp M* (*F i*) ($\lambda\xi.\ X\ j\ \xi - X\ i\ \xi$) $x = 0$
shows *martingale M F* t_0 *X*
proof
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
thus *AE* ξ *in M*. $X\ i\ \xi$ = *cond-exp M* (*F i*) (*X j*) ξ
using *diff-zero*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable(1,1)*, *of F i j i*]
sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted*, *of i*] **by** *fastforce*
}
qed (*auto intro*: *integrable adapted*[*THEN adapted-process.adapted*])

lemma (**in** *sigma-finite-filtered-measure*) *martingale-of-set-integral-eq*:
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \le i \Longrightarrow$ *integrable M* (*X i*)
and $\bigwedge A\ i\ j.\ t_0 \le i \Longrightarrow i \le j \Longrightarrow A \in F\ i \Longrightarrow$ *set-lebesgue-integral M A* (*X i*) = *set-lebesgue-integral M A* (*X j*)
shows *martingale M F* t_0 *X*
proof (*unfold-locales*)

fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i$ $i \leq j$
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
interpret *sigma-finite-subalgebra M F i* **using** *asm* **by** *blast*
interpret *r*: *sigma-finite-measure restr-to-subalg M* (*F i*) **by** (*simp add*: *sigma-fin-subalg*)
{
fix *A* **assume** $A \in$ *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 ... = *set-lebesgue-integral M A* (*cond-exp M* (*F i*) (*X j*)) **using** $*$ *assms*(*3*)[*OF asm*] *cond-exp-set-integral*[*OF integrable*] *asm* **by** *auto*
finally have *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* (*X i*) = *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* (*cond-exp M* (*F i*) (*X j*)) **using** $*$ *subalg* **by** (*auto simp*: *set-lebesgue-integral-def intro*!: *integral-subalgebra2*[*symmetric*] *borel-measurable-scaleR borel-measurable-cond-exp borel-measurable-indicator*)
}
hence *AE* ξ *in restr-to-subalg M* (*F i*). *X i* ξ = *cond-exp M* (*F i*) (*X j*) ξ **using** *asm* **by** (*intro r.density-unique-banach*, *auto intro*: *integrable-in-subalg subalg borel-measurable-cond-exp integrable*)
thus *AE* ξ *in M*. *X i* ξ = *cond-exp M* (*F i*) (*X j*) ξ **using** *AE-restr-to-subalg*[*OF subalg*] **by** *blast*
qed (*auto intro*: *integrable adapted*[*THEN adapted-process.adapted*])

context *submartingale*
begin

lemma *cond-exp-diff-nonneg*:
assumes $t_0 \leq i$ $i \leq j$
shows *AE x in M*. *cond-exp M* (*F i*) ($\lambda\xi$. *X j* ξ − *X i* ξ) *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* ξ + *Y i* ξ)
proof −
interpret *Y*: *submartingale M F* t_0 *Y* **by** (*rule assms*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i$ $i \leq j$
hence *AE* ξ *in M*. *X i* ξ + *Y i* ξ $\leq$ *cond-exp M* (*F i*) (λx. *X j x* + *Y j x*) ξ
using *sigma-finite-subalgebra.cond-exp-add*[*OF - integrable submartingale.integrable*[*OF assms*], *of F i j j*]
submartingale-property[*OF asm*] *submartingale.submartingale-property*[*OF assms asm*] *add-mono*[*of X i - - Y i -*] **by** *force*
}
thus *?thesis* **using** *assms* **by** (*unfold-locales*) (*auto simp add*: *borel-measurable-add random-variable adapted integrable Y.random-variable Y.adapted submartingale.integrable*)

qed

lemma *diff*[*intro*]:
  assumes *supermartingale M F* t_0 *Y*
  shows *submartingale M F* t_0 ($\lambda i\ \xi.\ X\ i\ \xi\ -\ Y\ i\ \xi$)
proof −
  interpret *Y*: *supermartingale M F* t_0 *Y* **by** (*rule assms*)
  {
    fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
    hence *AE* ξ *in M*. $X\ i\ \xi\ -\ Y\ i\ \xi \le$ *cond-exp M* (*F i*) ($\lambda x.\ X\ j\ x\ -\ Y\ j\ x$) ξ
        using *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable supermartingale.integrable*[*OF assms*], *of F i j j*]
        submartingale-property[*OF asm*] *supermartingale.supermartingale-property*[*OF assms asm*] *diff-mono*[*of X i - - - Y i -*] **by** *force*
  }
  thus *?thesis* **using** *assms* **by** (*unfold-locales*) (*auto simp add*: *borel-measurable-diff random-variable adapted integrable Y.random-variable Y.adapted supermartingale.integrable*)

qed

lemma *scaleR-nonneg*:
  assumes $c \ge 0$
  shows *submartingale M F* t_0 ($\lambda i\ \xi.\ c *_R X\ i\ \xi$)
proof
  {
    fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
    thus *AE* ξ *in M*. $c *_R X\ i\ \xi \le$ *cond-exp M* (*F i*) ($\lambda\xi.\ c *_R X\ j\ \xi$) 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 $c \le 0$
  shows *supermartingale M F* t_0 ($\lambda i\ \xi.\ c *_R X\ i\ \xi$)
proof
  {
    fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
    thus *AE* ξ *in M*. $c *_R X\ i\ \xi \ge$ *cond-exp M* (*F i*) ($\lambda\xi.\ c *_R X\ j\ \xi$) 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* t_0 (− *X*)
unfolding *fun-Compl-def* **using** *scaleR-le-zero*[*of* −*1*] **by** *simp*

end

context *submartingale-linorder*
begin

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

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

lemma *max-0*:
shows *submartingale M F* t_0 (λ*i* ξ. *max 0* (*X i* ξ))
proof −
interpret *zero*: *martingale-linorder M F* t_0 λ- -. *0* **by** (*force intro*: *martingale-linorder.intro martingale-order.intro*)
show *?thesis* **by** (*intro zero.max submartingale-linorder.intro submartingale-axioms*)
qed

end

lemma (**in** *sigma-finite-filtered-measure*) *submartingale-of-cond-exp-diff-nonneg*:
assumes *adapted*: *adapted-process M F* t_0 *X*

and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and *diff-nonneg*: $\bigwedge i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow$ *AE x in M. cond-exp M* (*F i*) ($\lambda\xi$. *X j* ξ − *X i* ξ) *x* ≥ *0*
shows *submartingale M F* t_0 *X*
proof (*unfold-locales*)
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i\ i \leq j$
thus *AE* ξ *in M. X i* ξ ≤ *cond-exp M* (*F i*) (*X j*) ξ
using *diff-nonneg*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable*(*1*,*1*), *of F i j i*]
sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted*, *of i*] **by** *fastforce*
}
qed (*auto intro*: *integrable adapted*[*THEN adapted-process.adapted*])

lemma (**in** *sigma-finite-filtered-measure*) *submartingale-of-set-integral-le*:
fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable M* (*X i*)
and $\bigwedge A\ i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow A \in F\ i \Longrightarrow$ *set-lebesgue-integral M A* (*X i*) ≤ *set-lebesgue-integral M A* (*X j*)
shows *submartingale M F* t_0 *X*
proof (*unfold-locales*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \leq i\ i \leq j$
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
interpret *r*: *sigma-finite-measure restr-to-subalg M* (*F i*) **using** *asm sigma-finite-subalgebra.sigma-fin-subalg* **by** *blast*
{
fix *A* **assume** *A* ∈ *restr-to-subalg M* (*F i*)
hence ∗: *A* ∈ *F i* **using** *asm sets-restr-to-subalg subalgebras* **by** *blast*
have *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* (*X i*) = *set-lebesgue-integral M A* (*X i*) **using** ∗ *asm subalgebras* **by** (*auto simp*: *set-lebesgue-integral-def intro*: *integral-subalgebra2 borel-measurable-scaleR adapted borel-measurable-indicator*)
also have ... ≤ *set-lebesgue-integral M A* (*cond-exp M* (*F i*) (*X j*)) **using** ∗ *assms*(*3*)[*OF asm*] *asm sigma-finite-subalgebra.cond-exp-set-integral*[*OF - integrable*] **by** *fastforce*
also have ... = *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* (*cond-exp M* (*F i*) (*X j*)) **using** ∗ *asm subalgebras* **by** (*auto simp*: *set-lebesgue-integral-def intro*!: *integral-subalgebra2*[*symmetric*] *borel-measurable-scaleR borel-measurable-cond-exp borel-measurable-indicator*)
finally have *0* ≤ *set-lebesgue-integral* (*restr-to-subalg M* (*F i*)) *A* ($\lambda\xi$. *cond-exp M* (*F i*) (*X j*) ξ − *X i* ξ) **using** ∗ *asm subalgebras* **by** (*subst set-integral-diff*, *auto simp add*: *set-integrable-def sets-restr-to-subalg intro*!: *integrable adapted integrable-in-subalg borel-measurable-scaleR borel-measurable-indicator borel-measurable-cond-exp integrable-mult-indicator*)
}
hence *AE* ξ *in restr-to-subalg M* (*F i*). *0* ≤ *cond-exp M* (*F i*) (*X j*) ξ − *X i* ξ

by (*intro r.density-nonneg integrable-in-subalg asm subalgebras borel-measurable-diff borel-measurable-cond-exp adapted Bochner-Integration.integrable-diff integrable-cond-exp integrable*)
thus *AE* ξ *in M. X i* $\xi \leq$ *cond-exp M* (*F i*) (*X j*) ξ **using** *AE-restr-to-subalg*[*OF subalgebras*] *asm* **by** *simp*
}
qed (*auto intro*: *integrable adapted*[*THEN adapted-process.adapted*])

context *supermartingale*
begin

lemma *cond-exp-diff-nonneg*:
assumes $t_0 \leq i$ $i \leq j$
shows *AE x in M. cond-exp M* (*F i*) ($\lambda\xi$. *X i* $\xi -$ *X j* ξ) $x \geq 0$
using *assms supermartingale-property*[*OF assms*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable*(*1*,*1*), *of F i i j*]
sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted*, *of i*] **by** *fastforce*

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

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

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

end

context *supermartingale-linorder*
begin

lemma *set-integral-ge*:
assumes $A \in F\ i$ $t_0 \leq i$ $i \leq j$
shows *set-lebesgue-integral M A* (*X i*) $\geq$ *set-lebesgue-integral M A* (*X j*)
using *supermartingale-property*[*OF assms*(*2*), *of j*] *assms subsetD*[*OF sets-F-subset*]
by (*subst sigma-finite-subalgebra.cond-exp-set-integral*[*OF - integrable assms*(*1*), *of j*])

(*auto intro*!: *scaleR-left-mono integral-mono-AE-banach integrable-mult-indicator integrable simp add*: *set-lebesgue-integral-def*)

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

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

end

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-of-cond-exp-diff-le-zero*:
assumes *adapted*: *adapted-process M F* t_0 *X*
and *integrable*: $\bigwedge i.\ t_0 \le i \Longrightarrow$ *integrable M* (*X i*)
and *diff-le-zero*: $\bigwedge i\ j.\ t_0 \le i \Longrightarrow i \le j \Longrightarrow$ *AE x in M*. *cond-exp M* (*F i*) ($\lambda\xi$. *X j* ξ − *X i* ξ) $x \le 0$
shows *supermartingale M F* t_0 *X*
proof
interpret *adapted-process M F* t_0 *X* **by** (*rule adapted*)
{
fix *i j* :: *'b* **assume** *asm*: $t_0 \le i$ $i \le j$
thus *AE* ξ *in M*. *X i* $\xi \ge$ *cond-exp M* (*F i*) (*X j*) ξ
using *diff-le-zero*[*OF asm*] *sigma-finite-subalgebra.cond-exp-diff*[*OF - integrable(1,1)*, *of F i j i*]
sigma-finite-subalgebra.cond-exp-F-meas[*OF - integrable adapted*, *of i*] **by** *fastforce*

}
qed (*auto intro*: *integrable adapted*[*THEN adapted-process.adapted*])

lemma (**in** *sigma-finite-filtered-measure*) *supermartingale-of-set-integral-ge*:
fixes X :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process* M F t_0 X
and *integrable*: $\bigwedge i.\ t_0 \leq i \Longrightarrow$ *integrable* M $(X\ i)$
and $\bigwedge A\ i\ j.\ t_0 \leq i \Longrightarrow i \leq j \Longrightarrow A \in F\ i \Longrightarrow$ *set-lebesgue-integral* M A $(X\ j) \leq$ *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 adapted*)
note ∗ = *set-integral-uminus*[*unfolded set-integrable-def*, *OF integrable-mult-indicator*[*OF* - *integrable*]]
have *supermartingale* M F t_0 $(-(-\ X))$
using *ord-eq-le-trans*[*OF* ∗ *ord-le-eq-trans*[*OF le-imp-neg-le*[*OF assms*(*3*)] ∗[*symmetric*]]] *sets-F-subset*[*THEN subsetD*]
by (*intro submartingale.uminus submartingale-of-set-integral-le*[*OF uminus-adapted*])

(*clarsimp simp add*: *fun-Compl-def integrable* | *fastforce*)+
thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

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

lemma *predictable-const*:
assumes *martingale* M F *0* X
and *predictable-process* M F *0* X
shows *AE* ξ *in* M. $X\ i\ \xi = X\ j\ \xi$
proof −
interpret *martingale* M F *0* X **by** (*rule assms*)
have ∗: *AE* ξ *in* M. $X\ i\ \xi = X$ *0* ξ **for** i
proof (*induction i*)
case *0*
then show *?case* **by** (*simp add*: *bot-nat-def*)
next
case (*Suc i*)
interpret *S*: *adapted-process* M F *0* $\lambda i.\ X$ (*Suc i*) **by** (*intro predictable-implies-adapted-Suc assms*)
show *?case* **using** *Suc S.adapted*[*of i*] *martingale-property*[*OF* - *le-SucI*, *of i*] *sigma-finite-subalgebra.cond-exp-F-meas*[*OF* - *integrable*, *of F i Suc i*] **by** *fastforce*
qed
show *?thesis* **using** ∗[*of i*] ∗[*of j*] **by** *force*
qed

lemma *martingale-of-set-integral-eq-Suc*:
assumes *adapted*: *adapted-process* M F *0* X
and *integrable*: $\bigwedge i.$ *integrable* M $(X\ i)$

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

lemma *martingale-nat*:
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: $\bigwedge i.$ *integrable M* $(X\ i)$
and $\bigwedge i.$ *AE* ξ *in M*. $X\ i\ \xi$ = *cond-exp M* $(F\ i)$ $(X\ (Suc\ i))$ ξ
shows *martingale M F 0 X*
proof (*unfold-locales*)
interpret *adapted-process M F 0 X* **by** (*rule adapted*)
fix *i j* :: *nat* **assume** *asm*: $i \le j$
show *AE* ξ *in M*. $X\ i\ \xi$ = *cond-exp M* $(F\ i)$ $(X\ j)$ ξ **using** *asm*
proof (*induction* $j - i$ *arbitrary*: *i j*)
case *0*
hence $j = i$ **by** *simp*
thus *?case* **using** *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable adapted*, *THEN AE-symmetric*] **by** *blast*
next
case (*Suc n*)
have *j*: $j = Suc\ (n + i)$ **using** *Suc* **by** *linarith*
have *n*: $n = n + i - i$ **using** *Suc* **by** *linarith*
have $*$: *AE* ξ *in M*. *cond-exp M* $(F\ (n + i))$ $(X\ j)$ ξ = $X\ (n + i)\ \xi$ **unfolding** *j* **using** *assms*(*3*)[*THEN AE-symmetric*] **by** *blast*
have *AE* ξ *in M*. *cond-exp M* $(F\ i)$ $(X\ j)$ ξ = *cond-exp M* $(F\ i)$ (*cond-exp M* $(F\ (n + i))$ $(X\ j)$) ξ **by** (*intro cond-exp-nested-subalg integrable subalg*, *simp add*: *subalgebra-def sets-F-mono*)
hence *AE* ξ *in M*. *cond-exp M* $(F\ i)$ $(X\ j)$ ξ = *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 (*auto simp add*: *integrable adapted*[*THEN adapted-process.adapted*])

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

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

end

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

lemma *predictable-mono*:
  assumes *submartingale M F 0 X*
    and *predictable-process M F 0 X i* ≤ *j*
  shows *AE* ξ *in M*. *X i* ξ ≤ *X j* ξ
  using *assms*(*3*)
proof (*induction j* − *i arbitrary*: *i j*)
  case *0*
  then show *?case* **by** *simp*
next
  case (*Suc n*)
  hence ∗: *n* = *j* − *Suc i* **by** *linarith*
  interpret *submartingale M F 0 X* **by** (*rule assms*)
  interpret *S*: *adapted-process M F 0* λi. *X* (*Suc i*) **by** (*intro predictable-implies-adapted-Suc assms*)
  have *Suc i* ≤ *j* **using** *Suc*(*2*,*3*) **by** *linarith*
  thus *?case* **using** *Suc*(*1*)[*OF* ∗] *S.adapted*[*of i*] *submartingale-property*[*OF - le-SucI*, *of i*] *sigma-finite-subalgebra.cond-exp-F-meas*[*OF - integrable*, *of F i Suc i*] **by** *fastforce*
qed

lemma *submartingale-of-set-integral-le-Suc*:
  fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
  assumes *adapted*: *adapted-process M F 0 X*
    and *integrable*: $\bigwedge i$. *integrable M* (*X i*)
    and $\bigwedge$*A i*. *A* ∈ *F i* ⟹ *set-lebesgue-integral M A* (*X i*) ≤ *set-lebesgue-integral M A* (*X* (*Suc i*))
  shows *submartingale M F 0 X*
proof (*intro submartingale-of-set-integral-le adapted integrable*)
  fix *i j A* **assume** *asm*: *i* ≤ *j A* ∈ *sets* (*F i*)
  show *set-lebesgue-integral M A* (*X i*) ≤ *set-lebesgue-integral M A* (*X j*) **using** *asm*
  proof (*induction j* − *i arbitrary*: *i j*)

case *0*
then show *?case* **using** *asm* **by** *simp*
next
case (*Suc n*)
hence ∗: $n = j - Suc\ i$ **by** *linarith*
have $Suc\ i \le j$ **using** *Suc*(*2*,*3*) **by** *linarith*
thus *?case* **using** *sets-F-mono*[*OF - le-SucI*] *Suc*(*4*) *Suc*(*1*)[*OF* ∗] **by** (*auto intro*: *assms*(*3*)[*THEN order-trans*])
qed
qed

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

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

lemma *submartingale-partial-sum-scaleR*:
assumes *submartingale-linorder M F 0 X*
and *adapted-process M F 0 C* $\bigwedge i.$ *AE* ξ *in M*. $0 \le C\ i\ \xi$ $\bigwedge i.$ *AE* ξ *in M*. $C\ i\ \xi \le R$
shows *submartingale M F 0* ($\lambda n\ \xi.\ \sum i<n.\ C\ i\ \xi *_R (X\ (Suc\ i)\ \xi - X\ i\ \xi)$)
proof −
interpret *submartingale-linorder M F 0 X* **by** (*rule assms*)
interpret *C*: *adapted-process M F 0 C* **by** (*rule assms*)
interpret *C'*: *adapted-process M F 0* $\lambda i\ \xi.\ C\ (i - 1)\ \xi *_R (X\ i\ \xi - X\ (i -$

1) *ξ*) **by** (*intro adapted-process.scaleR-right-adapted adapted-process.diff-adapted*, *unfold-locales*) (*auto intro*: *adaptedD C.adaptedD*)+
interpret *S*: *adapted-process M F 0* $\lambda n\ \xi.\ \sum i{<}n.\ C\ i\ \xi *_R (X\ (Suc\ i)\ \xi - X\ i\ \xi)$ **using** *C'.adapted-process-axioms*[*THEN partial-sum-Suc-adapted*] *diff-Suc-1* **by** *simp*
have *integrable M* $(\lambda x.\ C\ i\ x *_R (X\ (Suc\ i)\ x - X\ i\ x))$ **for** *i* **using** *assms*(*3*,*4*)[*of i*] **by** (*intro Bochner-Integration.integrable-bound*[*OF integrable-scaleR-right*, *OF Bochner-Integration.integrable-diff*, *OF integrable*(*1*,*1*), *of R Suc i i*]) (*auto simp add*: *mult-mono*)
moreover have *AE* ξ *in M*. $0 \le$ *cond-exp M* (*F i*) $(\lambda\xi.\ (\sum i{<}Suc\ i.\ C\ i\ \xi *_R (X\ (Suc\ i)\ \xi - X\ i\ \xi)) - (\sum i{<}i.\ C\ i\ \xi *_R (X\ (Suc\ i)\ \xi - X\ i\ \xi)))\ \xi$ **for** *i*
using *sigma-finite-subalgebra.cond-exp-measurable-scaleR*[*OF - calculation - C.adapted*, *of i*]
cond-exp-diff-nonneg[*OF - le-SucI*, *OF - order.refl*, *of i*] *assms*(*3*,*4*)[*of i*]
by (*fastforce simp add*: *scaleR-nonneg-nonneg integrable*)
ultimately show *?thesis* **by** (*intro submartingale-of-cond-exp-diff-Suc-nonneg S.adapted-process-axioms Bochner-Integration.integrable-sum*, *blast*+)
qed

lemma *submartingale-partial-sum-scaleR'*:
assumes *submartingale-linorder M F 0 X*
and *predictable-process M F 0 C* $\bigwedge i.$ *AE* ξ *in M*. $0 \le C\ i\ \xi$ $\bigwedge i.$ *AE* ξ *in M*. $C\ i\ \xi \le R$
shows *submartingale M F 0* $(\lambda n\ \xi.\ \sum i{<}n.\ C\ (Suc\ i)\ \xi *_R (X\ (Suc\ i)\ \xi - X\ i\ \xi))$
proof −
interpret *Suc-C*: *adapted-process M F 0* $\lambda i.\ C\ (Suc\ i)$ **using** *predictable-implies-adapted-Suc assms* **by** *blast*
show *?thesis* **by** (*intro submartingale-partial-sum-scaleR*[*OF assms*(*1*), *of - R*] *assms*) (*intro-locales*)
qed

end

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

lemma *predictable-mono'*:
assumes *supermartingale M F 0 X*
and *predictable-process M F 0 X* $i \le j$
shows *AE* ξ *in M*. $X\ i\ \xi \ge X\ j\ \xi$
using *assms*(*3*)
proof (*induction* $j - i$ *arbitrary*: *i j*)
case *0*
then show *?case* **by** *simp*
next
case (*Suc n*)
hence $*$: $n = j - Suc\ i$ **by** *linarith*
interpret *supermartingale M F 0 X* **by** (*rule assms*)

interpret *S*: *adapted-process M F 0 λi. X (Suc i)* **by** (*intro predictable-implies-adapted-Suc assms*)
have *Suc i ≤ j* **using** *Suc(2,3)* **by** *linarith*
thus *?case* **using** *Suc(1)[OF ∗] S.adapted[of i] supermartingale-property[OF - le-SucI, of i] sigma-finite-subalgebra.cond-exp-F-meas[OF - integrable, of F i Suc i]* **by** *fastforce*
qed

lemma *supermartingale-of-set-integral-ge-Suc*:
fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: ⋀*i. integrable M (X i)*
and ⋀*A i. A ∈ F i ⟹ set-lebesgue-integral M A (X i) ≥ set-lebesgue-integral M A (X (Suc i))*
shows *supermartingale M F 0 X*
proof −
interpret *adapted-process M F 0 X* **by** (*rule assms*)
interpret *uminus-X*: *adapted-process M F 0 −X* **by** (*rule uminus-adapted*)
note ∗ = *set-integral-uminus[unfolded set-integrable-def, OF integrable-mult-indicator[OF - integrable]]*
have *supermartingale M F 0 (−(− X))*
using *ord-eq-le-trans[OF ∗ ord-le-eq-trans[OF le-imp-neg-le[OF assms(3)] ∗[symmetric]]] sets-F-subset[THEN subsetD]*
by (*intro submartingale.uminus submartingale-of-set-integral-le-Suc[OF uminus-adapted]*)
(*clarsimp simp add*: *fun-Compl-def integrable* | *fastforce*)+
thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

lemma *supermartingale-nat*:
fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: ⋀*i. integrable M (X i)*
and ⋀*i. AE ξ in M. X i ξ ≥ cond-exp M (F i) (X (Suc i)) ξ*
shows *supermartingale M F 0 X*
proof −
interpret *adapted-process M F 0 X* **by** (*rule assms*)
have *AE ξ in M. − X i ξ ≤ cond-exp M (F i) (λx. − X (Suc i) x) ξ* **for** *i* **using** *assms(3) cond-exp-uminus[OF integrable, of i Suc i]* **by** *force*
hence *supermartingale M F 0 (−(− X))* **by** (*intro submartingale.uminus submartingale-nat[OF uminus-adapted]*) (*auto simp add*: *fun-Compl-def integrable*)
thus *?thesis* **unfolding** *fun-Compl-def* **by** *simp*
qed

lemma *supermartingale-of-cond-exp-diff-Suc-le-zero*:
fixes *X* :: - ⇒ - ⇒ - :: {*linorder-topology*}
assumes *adapted*: *adapted-process M F 0 X*
and *integrable*: ⋀*i. integrable M (X i)*
and ⋀*i. AE ξ in M. cond-exp M (F i) (λξ. X (Suc i) ξ − X i ξ) ξ ≤ 0*

shows *supermartingale M F 0 X*
proof (*intro supermartingale-nat integrable adapted*)
interpret *adapted-process M F 0 X* **by** (*rule assms*)
fix *i*
show *AE* ξ *in M*. *X i* $\xi \geq$ *cond-exp M* (*F i*) (*X* (*Suc i*)) ξ **using** *cond-exp-diff*[*OF integrable*(*1*,*1*), *of i Suc i i*] *cond-exp-F-meas*[*OF integrable adapted*, *of i*] *assms*(*3*)[*of i*] **by** *fastforce*
qed

end

end

3 Stopping Times and Hitting Times

In this section we formalize stopping times and hitting times. A stopping time is a random variable that represents the time at which a certain event occurs within a stochastic process. A hitting time, also known as first passage time or first hitting time, is a specific type of stopping time that represents the first time a stochastic process reaches a particular state or crosses a certain threshold.

theory *Stopping-Time*
imports *Martingales-Updates*
begin

3.1 Stopping Time

The formalization of stopping times here is simply a rewrite of the document *HOL−Probability.Stopping-Time* [5]. We have adapted the document to use the locales defined in our formalization of filtered measure spaces [6] [7]. This way we can omit the partial formalization of filtrations in the original document. Furthermore, we can include the initial time index t_0 that we introduced as well.

context *linearly-filtered-measure*
begin

— A stopping time is a measurable function from the measure space (possible events) into the time axis.

definition *stopping-time* :: ('*a* ⇒ '*b*) ⇒ *bool* **where**
stopping-time T = (($T \in$ *space M* $\to \{t_0..\}$) $\wedge$ ($\forall t \geq t_0$. *Measurable.pred* (*F t*) (λx. $T\ x \leq t$)))

lemma *stopping-time-cong*:
assumes $\bigwedge t\ x.\ t \geq t_0 \Longrightarrow x \in$ *space* (*F t*) $\Longrightarrow T\ x = S\ x$
shows *stopping-time T* = *stopping-time S*

proof (*cases* $T \in$ *space* $M \to \{t_0..\}$)
  case *True*
  hence $S \in$ *space* $M \to \{t_0..\}$ **using** *assms space-F* **by** *force*
  then show *?thesis* **unfolding** *stopping-time-def*
    using *assms arg-cong*[**where** $f=(\lambda P.\ \forall t \geq t_0.\ P\ t)$] *measurable-cong*[**where** $M=F$ - **and** $f=\lambda x.\ T\ x \leq$ - **and** $g=\lambda x.\ S\ x \leq$ -] *True* **by** *metis*
next
  case *False*
  hence $S \notin$ *space* $M \to \{t_0..\}$ **using** *assms space-F* **by** *force*
  then show *?thesis* **unfolding** *stopping-time-def* **using** *False* **by** *blast*
qed

lemma *stopping-time-ge-zero*:
  assumes *stopping-time* $T\ \omega \in$ *space* M
  shows $T\ \omega \geq t_0$
  using *assms* **unfolding** *stopping-time-def* **by** *auto*

lemma *stopping-timeD*:
  assumes *stopping-time* $T\ t \geq t_0$
  shows *Measurable.pred* $(F\ t)\ (\lambda x.\ T\ x \leq t)$
  using *assms* **unfolding** *stopping-time-def* **by** *simp*

lemma *stopping-timeI*[*intro?*]:
  assumes $\bigwedge x.\ x \in$ *space* $M \Longrightarrow T\ x \geq t_0$
    $(\bigwedge t.\ t \geq t_0 \Longrightarrow$ *Measurable.pred* $(F\ t)\ (\lambda x.\ T\ x \leq t))$
  shows *stopping-time* T
  using *assms* **by** (*auto simp*: *stopping-time-def*)

lemma *stopping-time-measurable*:
  assumes *stopping-time* T
  shows $T \in$ *borel-measurable* M
proof (*rule borel-measurableI-le*)
  {
    fix t **assume** $\neg\ t \geq t_0$
    hence $\{x \in$ *space* $M.\ T\ x \leq t\} = \{\}$ **using** *assms dual-order.trans*[*of* - $t\ t_0$] **unfolding** *stopping-time-def* **by** *fastforce*
    hence $\{x \in$ *space* $M.\ T\ x \leq t\} \in$ *sets* M **by** (*metis sets.empty-sets*)
  }
  moreover
  {
    fix t **assume** *asm*: $t \geq t_0$
    hence $\{x \in$ *space* $M.\ T\ x \leq t\} \in$ *sets* M **using** *stopping-timeD*[*OF assms asm*] *sets-F-subset* **unfolding** *Measurable.pred-def space-F*[*OF asm*] **by** *blast*
  }
  ultimately show $\{x \in$ *space* $M.\ T\ x \leq t\} \in$ *sets* M **for** t **by** *blast*
qed

lemma *stopping-time-const*:
  assumes $t \geq t_0$

shows *stopping-time* ($\lambda x.\ t$) **using** *assms* **by** (*auto simp*: *stopping-time-def*)

lemma *stopping-time-min*:
assumes *stopping-time T stopping-time S*
shows *stopping-time* ($\lambda x.$ *min* ($T\ x$) ($S\ x$))
using *assms* **by** (*auto simp*: *stopping-time-def min-le-iff-disj intro*!: *pred-intros-logic*)

lemma *stopping-time-max*:
assumes *stopping-time T stopping-time S*
shows *stopping-time* ($\lambda x.$ *max* ($T\ x$) ($S\ x$))
using *assms* **by** (*auto simp*: *stopping-time-def intro*!: *pred-intros-logic max.coboundedI1*)

3.2 σ-algebra of a Stopping Time

Moving on, we define the σ-algebra associated with a stopping time T. It contains all the information up to time T, the same way $F\ t$ contains all the information up to time t.

definition *pre-sigma* :: $('a \Rightarrow 'b) \Rightarrow 'a$ *measure* **where**
pre-sigma T = *sigma* (*space* M) $\{A \in$ *sets* $M.\ \forall t \geq t_0.\ \{\omega \in A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)\}$

lemma *measure-pre-sigma*[*simp*]: *emeasure* (*pre-sigma* T) = (λ-. *0*) **by** (*simp add*: *pre-sigma-def emeasure-sigma*)

lemma *sigma-algebra-pre-sigma*:
assumes *stopping-time T*
shows *sigma-algebra* (*space* M) $\{A \in$ *sets* $M.\ \forall t \geq t_0.\ \{\omega{\in}A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)\}$
proof −
let $?\Sigma = \{A \in$ *sets* $M.\ \forall t \geq t_0.\ \{\omega{\in}A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)\}$
{
fix A **assume** *asm*: $A \in ?\Sigma$
{
fix t **assume** *asm'*: $t \geq t_0$
hence $\{\omega{\in}A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)$ **using** *asm* **by** *blast*
then have $\{\omega \in$ *space* $(F\ t).\ T\ \omega \leq t\} - \{\omega \in A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)$ **using** *assms*[*THEN stopping-timeD*, *OF asm'*] **by** *auto*
also have $\{\omega \in$ *space* $(F\ t).\ T\ \omega \leq t\} - \{\omega \in A.\ T\ \omega \leq t\} = \{\omega \in$ *space* $M - A.\ T\ \omega \leq t\}$ **using** *space-F*[*OF asm'*] **by** *blast*
finally have $\{\omega \in$ (*space* M) $- A.\ T\ \omega \leq t\} \in$ *sets* $(F\ t)$ **.**
}
hence *space* $M - A \in ?\Sigma$ **using** *asm* **by** *blast*
}
moreover
{
fix A :: *nat* $\Rightarrow$ *'a set* **assume** *asm*: *range* $A \subseteq ?\Sigma$
{
fix t **assume** $t \geq t_0$
then have $(\bigcup i.\ \{\omega \in A\ i.\ T\ \omega \leq t\}) \in$ *sets* $(F\ t)$ **using** *asm* **by** *auto*

also have $(\bigcup i.\ \{\omega \in A\ i.\ T\ \omega \le t\}) = \{\omega \in \bigcup(A\ `\ UNIV).\ T\ \omega \le t\}$ **by** *auto*
finally have $\{\omega \in \bigcup(range\ A).\ T\ \omega \le t\} \in sets\ (F\ t)$ **.**
}
hence $\bigcup(range\ A) \in ?\Sigma$ **using** *asm* **by** *blast*
}
ultimately show *?thesis* **unfolding** *sigma-algebra-iff2* **by** (*auto intro*!: *sets.sets-into-space*[*THEN PowI*, *THEN subsetI*])
qed

lemma *space-pre-sigma*[*simp*]: *space* (*pre-sigma* T) = *space* M **unfolding** *pre-sigma-def*
by (*intro space-measure-of-conv*)

lemma *sets-pre-sigma*:
assumes *stopping-time* T
shows *sets* (*pre-sigma* T) $= \{A \in sets\ M.\ \forall t \ge t_0.\ \{\omega{\in}A.\ T\ \omega \le t\} \in F\ t\}$
unfolding *pre-sigma-def* **using** *sigma-algebra.sets-measure-of-eq*[*OF sigma-algebra-pre-sigma*, *OF assms*] **by** *blast*

lemma *sets-pre-sigmaI*:
assumes *stopping-time* T
and $\bigwedge t.\ t \ge t_0 \Longrightarrow \{\omega \in A.\ T\ \omega \le t\} \in F\ t$
shows $A \in$ *pre-sigma* T
proof (*cases* $\exists t \ge t_0.\ \forall t'.\ t' \le t$)
case *True*
then obtain t **where** $t \ge t_0$ $\{\omega \in A.\ T\ \omega \le t\} = A$ **by** *blast*
hence $A \in M$ **using** *assms*(*2*)[*of t*] *sets-F-subset*[*of t*] **by** *fastforce*
thus *?thesis* **using** *assms*(*2*) **unfolding** *sets-pre-sigma*[*OF assms*(*1*)] **by** *blast*
next
case *False*
hence $*$: $\{t{<}..\} \ne \{\}$ **if** $t \ge t_0$ **for** t **by** (*metis not-le empty-iff greaterThan-iff*)
obtain D :: *'b set* **where** D: *countable* D $\bigwedge X.$ *open* $X \Longrightarrow X \ne \{\} \Longrightarrow D \cap X \ne \{\}$ **by** (*metis countable-dense-setE disjoint-iff*)
have $**$: $D \cap \{t{<}..\} \ne \{\}$ **if** $t \ge t_0$ **for** t **using** $*$ *that* **by** (*intro D*(*2*)) *auto*
{
fix ω
obtain t **where** t: $t \ge t_0$ $T\ \omega \le t$ **using** *linorder-linear* **by** *auto*
moreover obtain t' **where** $t' \in D \cap \{t{<}..\} \cap \{t_0..\}$ **using** $**$ t **by** *fastforce*
moreover have $T\ \omega \le t'$ **using** *calculation* **by** *fastforce*
ultimately have $\exists t.\ \exists t' \in D \cap \{t{<}..\} \cap \{t_0..\}.\ T\ \omega \le t'$ **by** *blast*
}
hence $(\bigcup t'{\in}(\bigcup t.\ D \cap \{t{<}..\} \cap \{t_0..\}).\ \{\omega \in A.\ T\ \omega \le t'\}) = A$ **by** *fast*
moreover have $(\bigcup t'{\in}(\bigcup t.\ D \cap \{t{<}..\} \cap \{t_0..\}).\ \{\omega \in A.\ T\ \omega \le t'\}) \in M$ **using** D *assms*(*2*) *sets-F-subset* **by** (*intro sets.countable-UN''*, *fastforce*, *fast*)
ultimately have $A \in M$ **by** *argo*
thus *?thesis* **using** *assms*(*2*) **unfolding** *sets-pre-sigma*[*OF assms*(*1*)] **by** *blast*
qed

lemma *pred-pre-sigmaI*:

assumes *stopping-time T*
shows ($\bigwedge t.\ t \geq t_0 \Longrightarrow$ *Measurable.pred* ($F\ t$) ($\lambda\omega.\ P\ \omega \wedge T\ \omega \leq t$)) $\Longrightarrow$ *Measurable.pred* (*pre-sigma T*) P
unfolding *pred-def space-pre-sigma* **using** *assms* **by** (*auto intro*: *sets-pre-sigmaI*[*OF assms*(*1*)])

lemma *sets-pre-sigmaD*:
assumes *stopping-time* T $A \in$ *pre-sigma* T $t \geq t_0$
shows $\{\omega \in A.\ T\ \omega \leq t\} \in$ *sets* ($F\ t$)
using *assms sets-pre-sigma* **by** *auto*

lemma *borel-measurable-stopping-time-pre-sigma*:
assumes *stopping-time T*
shows $T \in$ *borel-measurable* (*pre-sigma T*)
proof (*intro borel-measurableI-le sets-pre-sigmaI*[*OF assms*])
fix $t\ t'$ **assume** *asm*: $t \geq t_0$
{
assume $\neg\ t' \geq t_0$
hence $\{\omega \in \{x \in$ *space* (*pre-sigma T*). $T\ x \leq t'\}.\ T\ \omega \leq t\} = \{\}$ **using** *assms dual-order.trans*[*of - t' t_0*] **unfolding** *stopping-time-def* **by** *fastforce*
hence $\{\omega \in \{x \in$ *space* (*pre-sigma T*). $T\ x \leq t'\}.\ T\ \omega \leq t\} \in$ *sets* ($F\ t$) **by** (*metis sets.empty-sets*)
}
moreover
{
assume *asm'*: $t' \geq t_0$
have $\{\omega \in$ *space* (F (*min* t' t)). $T\ \omega \leq$ *min* t' $t\} \in$ *sets* (F (*min* t' t))
using *assms asm asm'* **unfolding** *pred-def*[*symmetric*] **by** (*intro stopping-timeD*) *auto*
also have ... $\subseteq$ *sets* ($F\ t$)
using *assms asm asm'* **by** (*intro sets-F-mono*) *auto*
finally have $\{\omega \in \{x \in$ *space* (*pre-sigma T*). $T\ x \leq t'\}.\ T\ \omega \leq t\} \in$ *sets* ($F\ t$)
using *asm asm'* **by** *simp*
}
ultimately show $\{\omega \in \{x \in$ *space* (*pre-sigma T*). $T\ x \leq t'\}.\ T\ \omega \leq t\} \in$ *sets* ($F\ t$) **by** *blast*
qed

lemma *mono-pre-sigma*:
assumes *stopping-time T stopping-time S*
and $\bigwedge x.\ x \in$ *space* $M \Longrightarrow T\ x \leq S\ x$
shows *pre-sigma* $T \subseteq$ *pre-sigma* S
proof
fix A **assume** $A \in$ *pre-sigma T*
hence *asm*: $A \in$ *sets* M $t \geq t_0 \Longrightarrow \{\omega \in A.\ T\ \omega \leq t\} \in$ *sets* ($F\ t$) **for** t **using** *assms sets-pre-sigma* **by** *blast+*
{
fix t **assume** *asm'*: $t \geq t_0$
then have $A \subseteq$ *space* M **using** *sets.sets-into-space asm* **by** *blast*

have $\{\omega\in A.\ T\ \omega \leq t\} \cap \{\omega\in space\ (F\ t).\ S\ \omega \leq t\} \in sets\ (F\ t)$
using *asm asm′ stopping-timeD*[*OF assms*(*2*)] **by** (*auto simp*: *pred-def*)
also have $\{\omega\in A.\ T\ \omega \leq t\} \cap \{\omega\in space\ (F\ t).\ S\ \omega \leq t\} = \{\omega\in A.\ S\ \omega \leq t\}$
using *sets.sets-into-space*[*OF asm*(*1*)] *assms*(*3*) *order-trans asm′* **by** *fastforce*
finally have $\{\omega\in A.\ S\ \omega \leq t\} \in sets\ (F\ t)$ **by** *simp*
}
thus $A \in$ *pre-sigma S* **by** (*intro sets-pre-sigmaI assms asm*) *blast*
qed

lemma *stopping-time-measurable-le*:
assumes *stopping-time T* $s \geq t_0$ $t \geq s$
shows *Measurable.pred* $(F\ t)$ $(\lambda\omega.\ T\ \omega \leq s)$
using *assms stopping-timeD*[*of T*] *sets-F-mono*[*of - t*] **by** (*auto simp*: *pred-def*)

lemma *stopping-time-measurable-less*:
assumes *stopping-time T* $s \geq t_0$ $t \geq s$
shows *Measurable.pred* $(F\ t)$ $(\lambda\omega.\ T\ \omega < s)$
proof −
have *Measurable.pred* $(F\ t)$ $(\lambda\omega.\ T\ \omega < t)$ **if** *asm*: *stopping-time T* $t \geq t_0$ **for** *T t*
proof −
obtain *D* :: *′b set* **where** *D*: *countable D* $\bigwedge X.$ *open X* $\Longrightarrow X \neq \{\} \Longrightarrow D \cap X \neq \{\}$ **by** (*metis countable-dense-setE disjoint-iff*)
show *?thesis*
proof *cases*
assume $*$: $\forall\, t'\in\{t_0..<t\}.\ \{t'<..<t\} \neq \{\}$
hence $**$: $D \cap \{t'<..<t\} \neq \{\}$ **if** $t' \in \{t_0..<t\}$ **for** t' **using** *that* **by** (*intro D*(*2*)) *fastforce+*

show *?thesis*
proof (*rule measurable-cong*[*THEN iffD2*])
show $T\ \omega < t \longleftrightarrow (\exists\, r\in D \cap \{t_0..<t\}.\ T\ \omega \leq r)$ **if** $\omega \in space\ (F\ t)$ **for** ω
using $**$[*of* $T\ \omega$] *that less-imp-le stopping-time-ge-zero asm* **by** *fastforce*
show *Measurable.pred* $(F\ t)$ $(\lambda w.\ \exists\, r\in D \cap \{t_0..<t\}.\ T\ w \leq r)$
using *stopping-time-measurable-le asm D* **by** (*intro measurable-pred-countable*) *auto*
qed
next
assume $\neg\ (\forall\, t'\in\{t_0..<t\}.\ \{t'<..<t\} \neq \{\})$
then obtain t' **where** t': $t' \in \{t_0..<t\}$ $\{t'<..<t\} = \{\}$ **by** *blast*
show *?thesis*
proof (*rule measurable-cong*[*THEN iffD2*])
show $T\ \omega < t \longleftrightarrow T\ \omega \leq t'$ **for** ω **using** t' **by** (*metis atLeastLessThan-iff emptyE greaterThanLessThan-iff linorder-not-less order.strict-trans1*)
show *Measurable.pred* $(F\ t)$ $(\lambda w.\ T\ w \leq t')$ **using** t' **by** (*intro stopping-time-measurable-le*[*OF asm*(*1*)]) *auto*
qed
qed
qed
thus *?thesis*

using *assms sets-F-mono*[*of - t*] **by** (*auto simp add*: *pred-def*)
qed

lemma *stopping-time-measurable-ge*:
assumes *stopping-time* $T\ s \geq t_0\ t \geq s$
shows *Measurable.pred* $(F\ t)\ (\lambda\omega.\ T\ \omega \geq s)$
by (*auto simp*: *not-less*[*symmetric*] *intro*: *stopping-time-measurable-less*[*OF assms*] *Measurable.pred-intros-logic*)

lemma *stopping-time-measurable-gr*:
assumes *stopping-time* $T\ s \geq t_0\ t \geq s$
shows *Measurable.pred* $(F\ t)\ (\lambda x.\ s < T\ x)$
by (*auto simp add*: *not-le*[*symmetric*] *intro*: *stopping-time-measurable-le*[*OF assms*] *Measurable.pred-intros-logic*)

lemma *stopping-time-measurable-eq*:
assumes *stopping-time* $T\ s \geq t_0\ t \geq s$
shows *Measurable.pred* $(F\ t)\ (\lambda\omega.\ T\ \omega = s)$
unfolding *eq-iff* **using** *stopping-time-measurable-le*[*OF assms*] *stopping-time-measurable-ge*[*OF assms*]
by (*intro pred-intros-logic*)

lemma *stopping-time-less-stopping-time*:
assumes *stopping-time T stopping-time S*
shows *Measurable.pred* (*pre-sigma T*) $(\lambda\omega.\ T\ \omega < S\ \omega)$
proof (*rule pred-pre-sigmaI*)
fix *t* **assume** *asm*: $t \geq t_0$
obtain D :: *'b set* **where** *D*: *countable D* **and** *semidense-D*: $\bigwedge x\ y.\ x < y \Longrightarrow (\exists\, b{\in}D.\ x \leq b \wedge b < y)$
using *countable-separating-set-linorder2* **by** *auto*
show *Measurable.pred* $(F\ t)\ (\lambda\omega.\ T\ \omega < S\ \omega \wedge T\ \omega \leq t)$
proof (*rule measurable-cong*[*THEN iffD2*])
let $?f = \lambda\omega.$ *if* $T\ \omega = t$ *then* $\neg\ S\ \omega \leq t$ *else* $\exists\, s{\in}D \cap \{t_0..t\}.\ T\ \omega \leq s \wedge \neg\ (S\ \omega \leq s)$
{
fix ω **assume** $\omega \in$ *space* $(F\ t)\ T\ \omega \leq t\ T\ \omega \neq t\ T\ \omega < S\ \omega$
hence $t_0 \leq T\ \omega\ T\ \omega <$ *min* $t\ (S\ \omega)$ **using** *asm stopping-time-ge-zero*[*OF assms*(*1*)] **by** *auto*
then obtain r **where** $r \in D\ t_0 \leq r\ T\ \omega \leq r\ r <$ *min* $t\ (S\ \omega)$ **using** *semidense-D order-trans* **by** *blast*
hence $\exists\, s{\in}D \cap \{t_0..t\}.\ T\ \omega \leq s \wedge s < S\ \omega$ **by** *auto*
}
thus $(T\ \omega < S\ \omega \wedge T\ \omega \leq t) = ?f\ \omega$ **if** $\omega \in$ *space* $(F\ t)$ **for** ω **using** *that* **by** *force*
show *Measurable.pred* $(F\ t)$ *?f*
using *assms asm D*
by (*intro pred-intros-logic measurable-If measurable-pred-countable countable-Collect*
stopping-time-measurable-le predE stopping-time-measurable-eq) *auto*

qed
qed (*intro assms*)

end

lemma (**in** *enat-filtered-measure*) *stopping-time-SUP-enat*:
fixes T :: *nat* ⇒ ('*a* ⇒ *enat*)
shows ($\bigwedge i.$ *stopping-time* (T i)) ⟹ *stopping-time* (*SUP* i. T i)
unfolding *stopping-time-def SUP-apply SUP-le-iff* **by** (*auto intro*!: *pred-intros-countable*)

lemma (**in** *enat-filtered-measure*) *stopping-time-Inf-enat*:
assumes $\bigwedge i.$ *Measurable.pred* (F i) (P i)
shows *stopping-time* ($\lambda\omega$. *Inf* {i. P i ω})
proof −
{
fix t :: *enat* **assume** *asm*: $t \neq \infty$
moreover
{
fix i ω **assume** *Inf* {i. P i ω} $\le t$
moreover have $a <$ *eSuc* $b \longleftrightarrow (a \le b \wedge a \neq \infty)$ **for** a b **by** (*cases* a) *auto*
ultimately have ($\exists i{\le}t$. P i ω) **using** *asm* **unfolding** *Inf-le-iff* **by** (*cases* t) (*auto elim*!: *allE*[*of* - *eSuc* t])
}
ultimately have ∗: $\bigwedge\omega$. *Inf* {i. P i ω} $\le t \longleftrightarrow$ ($\exists i{\in}\{..t\}$. P i ω) **by** (*auto intro*!: *Inf-lower2*)
have *Measurable.pred* (F t) ($\lambda\omega$. *Inf* {i. P i ω} $\le t$) **unfolding** ∗ **using** *sets-F-mono assms* **by** (*intro pred-intros-countable-bounded*) (*auto simp*: *pred-def*)
}
moreover have *Measurable.pred* (F t) ($\lambda\omega$. *Inf* {i. P i ω} $\le t$) **if** $t = \infty$ **for** t **using** *that* **by** *simp*
ultimately show *?thesis* **by** (*blast intro*: *stopping-timeI*[*OF i0-lb*])
qed

lemma (**in** *nat-filtered-measure*) *stopping-time-Inf-nat*:
assumes $\bigwedge i.$ *Measurable.pred* (F i) (P i)
$\bigwedge i\ \omega.\ \omega \in$ *space* $M \Longrightarrow \exists n.\ P\ n\ \omega$
shows *stopping-time* ($\lambda\omega$. *Inf* {i. P i ω})
proof (*rule stopping-time-cong*[*THEN iffD2*])
show *stopping-time* (λx. *LEAST* n. P n x)
proof
fix t
have ((*LEAST* n. P n ω) $\le t$) = ($\exists i{\le}t$. P i ω) **if** $\omega \in$ *space* M **for** ω **by** (*rule LeastI2-wellorder-ex*[*OF assms*(*2*)[*OF that*]]) *auto*
moreover have *Measurable.pred* (F t) (λw. $\exists i{\in}\{..t\}$. P i w) **using** *sets-F-mono*[*of* - t] *assms* **by** (*intro pred-intros-countable-bounded*) (*auto simp*: *pred-def*)
ultimately show *Measurable.pred* (F t) ($\lambda\omega$. (*LEAST* n. P n ω) $\le t$) **by** (*subst measurable-cong*[*of* F t]) *auto*
qed (*simp*)
qed (*simp add*: *Inf-nat-def*)

definition *stopped-value* :: $('b \Rightarrow 'a \Rightarrow 'c) \Rightarrow ('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'c)$ **where**
stopped-value $X\ \tau\ \omega = X\ (\tau\ \omega)\ \omega$

3.3 Hitting Time

Given a stochastic process X and a borel set A, *hitting-time* $X\ A\ s\ t$ is the first time X is in A after time s and before time t. If X does not hit A after time s and before t then the hitting time is simply t. The definition presented here coincides with the definition of hitting times in mathlib [1].

context *linearly-filtered-measure*
begin

definition *hitting-time* :: $('b \Rightarrow 'a \Rightarrow 'c) \Rightarrow 'c\ set \Rightarrow 'b \Rightarrow 'b \Rightarrow ('a \Rightarrow 'b)$ **where**
hitting-time $X\ A\ s\ t = (\lambda\omega.\ if\ \exists\, i{\in}\{s..t\} \cap \{t_0..\}.\ X\ i\ \omega \in A\ then\ Inf\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})\ else\ max\ t_0\ t)$

lemma *hitting-time-def'*:
hitting-time $X\ A\ s\ t = (\lambda\omega.\ Inf\ (insert\ (max\ t_0\ t)\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})))$
proof *cases*
assume *asm*: $t_0 \le s \wedge s \le t$
hence $\{s..t\} \cap \{t_0..\} = \{s..t\}$ **by** *simp*
{
fix ω
assume $*$: $\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\} \neq \{\}$
then obtain i **where** $i \in \{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}$ **by** *blast*
hence $Inf\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}) \le t$ **by** (*intro cInf-lower*[*of i*, *THEN order-trans*]) *auto*
hence $Inf\ (insert\ (max\ t_0\ t)\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})) = Inf\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})$ **using** *asm* $*$ *inf-absorb2* **by** (*subst cInf-insert-If*) *force+*
also have ... = *hitting-time* $X\ A\ s\ t\ \omega$ **using** $*$ **unfolding** *hitting-time-def* **by** *auto*
finally have *hitting-time* $X\ A\ s\ t\ \omega = Inf\ (insert\ (max\ t_0\ t)\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}))$ **by** *argo*
}
moreover
{
fix ω
assume $\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\} = \{\}$
hence *hitting-time* $X\ A\ s\ t\ \omega = Inf\ (insert\ (max\ t_0\ t)\ (\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}))$ **unfolding** *hitting-time-def* **by** *auto*
}
ultimately show *?thesis* **by** *fast*
next
assume $\neg\ (t_0 \le s \wedge s \le t)$
moreover
{

assume *asm*: $s < t_0$ $t \geq t_0$
hence $\{s..t\} \cap \{t_0..\} = \{t_0..t\}$ **by** *simp*
{
fix ω
assume $*$: $\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\} \neq \{\}$
then obtain i **where** $i \in \{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}$ **by** *blast*
hence *Inf* $(\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\}) \leq t$ **by** (*intro cInf-lower*[*of i*, *THEN order-trans*]) *auto*
hence *Inf* (*insert* (*max* t_0 t) $(\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})$) = *Inf* $(\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})$ **using** *asm* $*$ *inf-absorb2* **by** (*subst cInf-insert-If*) *force*+

also have ... = *hitting-time X A s t* ω **using** $*$ **unfolding** *hitting-time-def* **by** *auto*
finally have *hitting-time X A s t* ω = *Inf* (*insert* (*max* t_0 t) $(\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})$) **by** *argo*
}
moreover
{
fix ω
assume $\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\} = \{\}$
hence *hitting-time X A s t* ω = *Inf* (*insert* (*max* t_0 t) $(\{s..t\} \cap \{t_0..\} \cap \{i.\ X\ i\ \omega \in A\})$) **unfolding** *hitting-time-def* **by** *auto*
}
ultimately have *?thesis* **by** *fast*
}
moreover have *?thesis* **if** $s < t_0$ $t < t_0$ **using** *that* **unfolding** *hitting-time-def* **by** *auto*
moreover have *?thesis* **if** $s > t$ **using** *that* **unfolding** *hitting-time-def* **by** *auto*
ultimately show *?thesis* **by** *fastforce*
qed

— The following lemma provides a sufficient condition for an injective function to preserve a hitting time.

lemma *hitting-time-inj-on*:
assumes *inj-on f S* $\bigwedge\omega\ t.\ t \geq t_0 \Longrightarrow X\ t\ \omega \in S$ $A \subseteq S$
shows *hitting-time X A* = *hitting-time* $(\lambda t\ \omega.\ f\ (X\ t\ \omega))$ (*f* ‘ *A*)
proof −
have $X\ t\ \omega \in A \longleftrightarrow f\ (X\ t\ \omega) \in f$ ‘ A **if** $t \geq t_0$ **for** t ω **using** *assms that inj-on-image-mem-iff* **by** *meson*
hence $\{t_0..\} \cap \{i.\ X\ i\ \omega \in A\} = \{t_0..\} \cap \{i.\ f\ (X\ i\ \omega) \in f$ ‘ $A\}$ **for** ω **by** *blast*
thus *?thesis* **unfolding** *hitting-time-def′ Int-assoc* **by** *presburger*
qed

lemma *hitting-time-translate*:
fixes *c* :: - :: *ab-group-add*
shows *hitting-time X A* = *hitting-time* $(\lambda n\ \omega.\ X\ n\ \omega + c)$ (((+) *c*) ‘ *A*)
by (*subst hitting-time-inj-on*[*OF inj-on-add*, *of - UNIV*]) (*simp add*: *add.commute*)+

lemma *hitting-time-le*:
assumes $t \geq t_0$
shows *hitting-time X A s t ω ≤ t*
unfolding *hitting-time-def′* **using** *assms*
by (*intro cInf-lower*[*of max* t_0 *t*, *THEN order-trans*]) *auto*

lemma *hitting-time-ge*:
assumes $t \geq t_0$ $s \leq t$
shows *s ≤ hitting-time X A s t ω*
unfolding *hitting-time-def′* **using** *assms*
by (*intro le-cInf-iff*[*THEN iffD2*]) *auto*

lemma *hitting-time-mono*:
assumes $t \geq t_0$ $s \leq s'$ $t \leq t'$
shows *hitting-time X A s t ω ≤ hitting-time X A s′ t′ ω*
unfolding *hitting-time-def′* **using** *assms* **by** (*fastforce intro*!: *cInf-mono*)

end

context *nat-filtered-measure*
begin

— Hitting times are stopping times for adapted processes.

lemma *stopping-time-hitting-time*:
assumes *adapted-process M F 0 X A ∈ borel*
shows *stopping-time* (*hitting-time X A s t*)
proof −
interpret *adapted-process M F 0 X* **by** (*rule assms*)
have *insert t* ({*s*..*t*} ∩ {*i*. *X i ω ∈ A*}) = {*i*. *i* = *t* ∨ *i* ∈ ({*s*..*t*} ∩ {*i*. *X i ω ∈ A*})} **for** *ω* **by** *blast*
hence *hitting-time X A s t* = (*λω*. *Inf* {*i*. *i* = *t* ∨ *i* ∈ ({*s*..*t*} ∩ {*i*. *X i ω ∈ A*})})
unfolding *hitting-time-def′* **by** *simp*
thus *?thesis* **using** *assms* **by** (*auto intro*: *stopping-time-Inf-nat*)
qed

lemma *stopping-time-hitting-time′*:
assumes *adapted-process M F 0 X A ∈ borel stopping-time s* ⋀*ω*. *s ω ≤ t*
shows *stopping-time* (*λω*. *hitting-time X A* (*s ω*) *t ω*)
proof −
interpret *adapted-process M F 0 X* **by** (*rule assms*)
{
fix *n*
have *s ω ≤ hitting-time X A* (*s ω*) *t ω* **if** *s ω > n* **for** *ω* **using** *hitting-time-ge*[*OF* - *assms*(*4*)] **by** *simp*
hence (⋃*i*∈{*n*<..}. {*ω*. *s ω* = *i*} ∩ {*ω*. *hitting-time X A i t ω ≤ n*}) = {} **by** *fastforce*
hence ∗: (*λω*. *hitting-time X A* (*s ω*) *t ω ≤ n*) = (*λω*. ∃*i*≤*n*. *s ω* = *i* ∧ *hitting-time X A i t ω ≤ n*) **by** *force*

have *Measurable.pred* $(F\ n)$ $(\lambda\omega.\ s\ \omega = i \wedge$ *hitting-time* $X\ A\ i\ t\ \omega \leq n)$ **if** $i \leq n$ **for** i
proof −
have *Measurable.pred* $(F\ i)$ $(\lambda\omega.\ s\ \omega = i)$ **using** *stopping-time-measurable-eq assms* **by** *blast*
hence *Measurable.pred* $(F\ n)$ $(\lambda\omega.\ s\ \omega = i)$ **by** (*meson less-eq-nat.simps measurable-from-subalg subalgebra-F that*)
moreover have *Measurable.pred* $(F\ n)$ $(\lambda\omega.$ *hitting-time* $X\ A\ i\ t\ \omega \leq n)$ **using** *stopping-timeD*[*OF stopping-time-hitting-time*, *OF assms*(*1*,*2*)] **by** *blast*
ultimately show *?thesis* **by** *auto*
qed
hence *Measurable.pred* $(F\ n)$ $(\lambda\omega.\ \exists\, i{\leq}n.\ s\ \omega = i \wedge$ *hitting-time* $X\ A\ i\ t\ \omega \leq n)$ **by** (*intro pred-intros-countable*) *auto*
hence *Measurable.pred* $(F\ n)$ $(\lambda\omega.$ *hitting-time* $X\ A\ (s\ \omega)\ t\ \omega \leq n)$ **using** $*$ **by** *argo*
}
thus *?thesis* **by** (*intro stopping-timeI*) *auto*
qed

— If X hits A at time $j \in \{s..t\}$, then the stopped value of X at the hitting time of A in the interval $\{s..t\}$ is an element of A.

lemma *stopped-value-hitting-time-mem*:
assumes $j \in \{s..t\}$ $X\ j\ \omega \in A$
shows *stopped-value* X (*hitting-time* $X\ A\ s\ t)\ \omega \in A$
proof −
have $\exists\, i{\in}\{s..t\} \cap \{0..\}.\ X\ i\ \omega \in A$ **using** *assms* **by** *blast*
moreover have *Inf* $(\{s..t\} \cap \{i.\ X\ i\ \omega \in A\}) \in \{s..t\} \cap \{i.\ X\ i\ \omega \in A\}$ **using** *assms* **by** (*blast intro*!: *Inf-nat-def1*)
ultimately show *?thesis* **unfolding** *hitting-time-def stopped-value-def* **by** *simp*
qed

lemma *hitting-time-le-iff*:
assumes $i < t$
shows *hitting-time* $X\ A\ s\ t\ \omega \leq i \longleftrightarrow (\exists\, j \in \{s..i\}.\ X\ j\ \omega \in A)$ (**is** *?lhs* = *?rhs*)
proof
assume *?lhs*
moreover have *hitting-time* $X\ A\ s\ t\ \omega \in$ *insert* t $(\{s..t\} \cap \{i.\ X\ i\ \omega \in A\})$ **by** (*metis hitting-time-def′ Int-atLeastAtMostR2 inf-sup-aci*(*1*) *insertI1 max-0L wellorder-InfI*)
ultimately have *hitting-time* $X\ A\ s\ t\ \omega \in \{s..i\} \cap \{i.\ X\ i\ \omega \in A\}$ **using** *assms* **by** *force*
thus *?rhs* **by** *blast*
next
assume *?rhs*
then obtain j **where** j: $j \in \{s..i\}$ $X\ j\ \omega \in A$ **by** *blast*
hence *hitting-time* $X\ A\ s\ t\ \omega \leq j$ **unfolding** *hitting-time-def′* **using** *assms* **by** (*auto intro*: *cInf-lower*)

thus *?lhs* **using** *j* **by** *simp*
qed

lemma *hitting-time-less-iff*:
assumes $i \leq t$
shows *hitting-time* $X\ A\ s\ t\ \omega < i \longleftrightarrow (\exists j \in \{s..<i\}.\ X\ j\ \omega \in A)$ (**is** *?lhs* = *?rhs*)
proof
assume *?lhs*
moreover have *hitting-time* $X\ A\ s\ t\ \omega \in$ *insert* t $(\{s..t\} \cap \{i.\ X\ i\ \omega \in A\})$ **by** (*metis hitting-time-def' Int-atLeastAtMostR2 inf-sup-aci*(*1*) *insertI1 max-0L wellorder-InfI*)
ultimately have *hitting-time* $X\ A\ s\ t\ \omega \in \{s..<i\} \cap \{i.\ X\ i\ \omega \in A\}$ **using** *assms* **by** *force*
thus *?rhs* **by** *blast*
next
assume *?rhs*
then obtain *j* **where** *j*: $j \in \{s..<i\}$ $X\ j\ \omega \in A$ **by** *blast*
hence *hitting-time* $X\ A\ s\ t\ \omega \leq j$ **unfolding** *hitting-time-def'* **using** *assms* **by** (*auto intro*: *cInf-lower*)
thus *?lhs* **using** *j* **by** *simp*
qed

— If X already hits A in the interval $\{s..t\}$, then *hitting-time* $X\ A\ s\ t$ = *hitting-time* $X\ A\ s\ t'$ for $t \leq t'$.

lemma *hitting-time-eq-hitting-time*:
assumes $t \leq t'$ $j \in \{s..t\}$ $X\ j\ \omega \in A$
shows *hitting-time* $X\ A\ s\ t\ \omega$ = *hitting-time* $X\ A\ s\ t'\ \omega$ (**is** *?lhs* = *?rhs*)
proof −
have *hitting-time* $X\ A\ s\ t\ \omega \in \{s..t'\}$ **using** *hitting-time-le*[*THEN order-trans*, *of* $t\ t'\ X\ A\ s$] *hitting-time-ge*[*of* $t\ s\ X\ A$] *assms* **by** *auto*
moreover have *stopped-value* X (*hitting-time* $X\ A\ s\ t$) $\omega \in A$ **by** (*blast intro*: *stopped-value-hitting-time-mem assms*)
ultimately have *hitting-time* $X\ A\ s\ t'\ \omega \leq$ *hitting-time* $X\ A\ s\ t\ \omega$ **by** (*fastforce simp add*: *hitting-time-def'*[**where** *t*=*t'*] *stopped-value-def intro*!: *cInf-lower*)
thus *?thesis* **by** (*blast intro*: *le-antisym hitting-time-mono*[*OF - order-refl assms*(*1*)])
qed

end

end

4 Doob's Upcrossing Inequality and Martingale Convergence Theorems

In this section we formalize upcrossings and downcrossings. Following this, we prove Doob's upcrossing inequality and first martingale convergence theorem.

theory *Upcrossing*
  imports *Stopping-Time*
begin

lemma *real-embedding-borel-measurable*: *real* ∈ *borel-measurable borel* **by** (*auto intro*: *borel-measurable-continuous-onI*)

lemma *limsup-lower-bound*:
  fixes *u*:: *nat* ⇒ *ereal*
  assumes *limsup u* > *l*
  shows ∃ *N*>*k*. *u N* > *l*
proof −
  have *limsup u* = − *liminf* (λ*n*. − *u n*) **using** *liminf-ereal-cminus*[*of 0 u*] **by** *simp*
  hence *liminf* (λ*n*. − *u n*) < − *l* **using** *assms ereal-less-uminus-reorder* **by** *presburger*
  hence ∃ *N*>*k*. − *u N* < − *l* **using** *liminf-upper-bound* **by** *blast*
  thus *?thesis* **using** *ereal-less-uminus-reorder* **by** *simp*
qed

lemma *ereal-abs-max-min*: |*c*| = *max 0 c* − *min 0 c* **for** *c* :: *ereal*
  by (*cases c* ≥ *0*) *auto*

4.1 Upcrossings and Downcrossings

Given a stochastic process X, real values a and b, and some point in time N, we would like to define a notion of "upcrossings" of X across the band $\{a..b\}$ which counts the number of times any realization of X crosses from below a to above b before time N. To make this heuristic rigorous, we inductively define the following hitting times.

context *nat-filtered-measure*
begin

context
  fixes *X* :: *nat* ⇒ *′a* ⇒ *real*
    and *a b* :: *real*
    and *N* :: *nat*
begin

primrec *upcrossing* :: *nat* ⇒ *′a* ⇒ *nat* **where**
  upcrossing 0 = (λω. *0*) |
  upcrossing (*Suc n*) = (λω. *hitting-time X* {*b*..} (*hitting-time X* {..*a*} (*upcrossing n* ω) *N* ω) *N* ω)

definition *downcrossing* :: *nat* ⇒ *′a* ⇒ *nat* **where**
  downcrossing n = (λω. *hitting-time X* {..*a*} (*upcrossing n* ω) *N* ω)

lemma *upcrossing-simps*:
upcrossing 0 = (λω. *0*)
upcrossing (*Suc n*) = (λω. *hitting-time X* {*b*..} (*downcrossing n* ω) *N* ω)
by (*auto simp add*: *downcrossing-def*)

lemma *downcrossing-simps*:
downcrossing 0 = *hitting-time X* {..*a*} *0 N*
downcrossing n = (λω. *hitting-time X* {..*a*} (*upcrossing n* ω) *N* ω)
by (*auto simp add*: *downcrossing-def*)

declare *upcrossing.simps*[*simp del*]

lemma *upcrossing-le*: *upcrossing n* ω ≤ *N*
by (*cases n*) (*auto simp add*: *upcrossing-simps hitting-time-le*)

lemma *downcrossing-le*: *downcrossing n* ω ≤ *N*
by (*cases n*) (*auto simp add*: *downcrossing-simps hitting-time-le*)

lemma *upcrossing-le-downcrossing*: *upcrossing n* ω ≤ *downcrossing n* ω
unfolding *downcrossing-simps* **by** (*auto intro*: *hitting-time-ge upcrossing-le*)

lemma *downcrossing-le-upcrossing-Suc*: *downcrossing n* ω ≤ *upcrossing* (*Suc n*) ω
unfolding *upcrossing-simps* **by** (*auto intro*: *hitting-time-ge downcrossing-le*)

lemma *upcrossing-mono*:
assumes *n* ≤ *m*
shows *upcrossing n* ω ≤ *upcrossing m* ω
using *order-trans*[*OF upcrossing-le-downcrossing downcrossing-le-upcrossing-Suc*] *assms*
by (*rule lift-Suc-mono-le*)

lemma *downcrossing-mono*:
assumes *n* ≤ *m*
shows *downcrossing n* ω ≤ *downcrossing m* ω
using *order-trans*[*OF downcrossing-le-upcrossing-Suc upcrossing-le-downcrossing*] *assms*
by (*rule lift-Suc-mono-le*)

— The following lemmas help us make statements about when an upcrossing (resp. downcrossing) occurs, and the value that the process takes at that instant.

lemma *stopped-value-upcrossing*:
assumes *upcrossing* (*Suc n*) ω ≠ *N*
shows *stopped-value X* (*upcrossing* (*Suc n*)) ω ≥ *b*
proof −
have ∗: *upcrossing* (*Suc n*) ω < *N* **using** *le-neq-implies-less upcrossing-le assms* **by** *presburger*
have ∃ *j* ∈ {*downcrossing n* ω..*upcrossing* (*Suc n*) ω}. *X j* ω ∈ {*b*..}

using *hitting-time-le-iff*[*THEN iffD1*, *OF* ∗] *upcrossing-simps* **by** *fastforce*
then obtain *j* **where** *j*: $j \in \{downcrossing\ n\ \omega..N\}$ $X\ j\ \omega \in \{b..\}$ **using** ∗ **by** (*meson atLeastatMost-subset-iff le-refl subsetD upcrossing-le*)
thus *?thesis* **using** *stopped-value-hitting-time-mem*[*of j - - X*] **unfolding** *upcrossing-simps stopped-value-def* **by** *blast*
qed

lemma *stopped-value-downcrossing*:
assumes $downcrossing\ n\ \omega \neq N$
shows $stopped\text{-}value\ X\ (downcrossing\ n)\ \omega \leq a$
proof −
have ∗: $downcrossing\ n\ \omega < N$ **using** *le-neq-implies-less downcrossing-le assms* **by** *presburger*
have $\exists j \in \{upcrossing\ n\ \omega..downcrossing\ n\ \omega\}.\ X\ j\ \omega \in \{..a\}$
using *hitting-time-le-iff*[*THEN iffD1*, *OF* ∗] *downcrossing-simps* **by** *fastforce*
then obtain *j* **where** *j*: $j \in \{upcrossing\ n\ \omega..N\}$ $X\ j\ \omega \in \{..a\}$ **using** ∗ **by** (*meson atLeastatMost-subset-iff le-refl subsetD downcrossing-le*)
thus *?thesis* **using** *stopped-value-hitting-time-mem*[*of j - - X*] **unfolding** *downcrossing-simps stopped-value-def* **by** *blast*
qed

lemma *upcrossing-less-downcrossing*:
assumes $a < b$ $downcrossing\ (Suc\ n)\ \omega \neq N$
shows $upcrossing\ (Suc\ n)\ \omega < downcrossing\ (Suc\ n)\ \omega$
proof −
have $upcrossing\ (Suc\ n)\ \omega \neq N$ **using** *assms* **by** (*metis le-antisym downcrossing-le upcrossing-le-downcrossing*)
hence $stopped\text{-}value\ X\ (downcrossing\ (Suc\ n))\ \omega < stopped\text{-}value\ X\ (upcrossing\ (Suc\ n))\ \omega$
using *assms stopped-value-downcrossing stopped-value-upcrossing* **by** *force*
hence $downcrossing\ (Suc\ n)\ \omega \neq upcrossing\ (Suc\ n)\ \omega$ **unfolding** *stopped-value-def* **by** *force*
thus *?thesis* **using** *upcrossing-le-downcrossing* **by** (*simp add*: *le-neq-implies-less*)
qed

lemma *downcrossing-less-upcrossing*:
assumes $a < b$ $upcrossing\ (Suc\ n)\ \omega \neq N$
shows $downcrossing\ n\ \omega < upcrossing\ (Suc\ n)\ \omega$
proof −
have $downcrossing\ n\ \omega \neq N$ **using** *assms* **by** (*metis le-antisym upcrossing-le downcrossing-le-upcrossing-Suc*)
hence $stopped\text{-}value\ X\ (downcrossing\ n)\ \omega < stopped\text{-}value\ X\ (upcrossing\ (Suc\ n))\ \omega$
using *assms stopped-value-downcrossing stopped-value-upcrossing* **by** *force*
hence $downcrossing\ n\ \omega \neq upcrossing\ (Suc\ n)\ \omega$ **unfolding** *stopped-value-def* **by** *force*
thus *?thesis* **using** *downcrossing-le-upcrossing-Suc* **by** (*simp add*: *le-neq-implies-less*)
qed

lemma *upcrossing-less-Suc*:
assumes $a < b$ *upcrossing* $n\ \omega \neq N$
shows *upcrossing* $n\ \omega <$ *upcrossing* (*Suc* n) ω
by (*metis assms upcrossing-le-downcrossing downcrossing-less-upcrossing order-le-less-trans le-neq-implies-less upcrossing-le*)

lemma *upcrossing-eq-bound*:
assumes $a < b$ $n \geq N$
shows *upcrossing* $n\ \omega = N$
proof −
have ∗: *upcrossing* $N\ \omega = N$
proof −
{
assume ∗: *upcrossing* $N\ \omega \neq N$
hence *asm*: *upcrossing* $n\ \omega < N$ **if** $n \leq N$ **for** n **using** *upcrossing-mono upcrossing-le that* **by** (*metis le-antisym le-neq-implies-less*)
{
fix *i j*
assume $i \leq N$ $i < j$
hence *upcrossing* $i\ \omega \neq$ *upcrossing* $j\ \omega$ **by** (*metis Suc-leI asm assms*(*1*) *leD upcrossing-less-Suc upcrossing-mono*)
}
moreover
{
fix *j*
assume $j \leq N$
hence *upcrossing* $j\ \omega \leq$ *upcrossing* $N\ \omega$ **using** *upcrossing-mono* **by** *blast*
hence *upcrossing* (*Suc* N) $\omega \neq$ *upcrossing* $j\ \omega$ **using** *upcrossing-less-Suc*[*OF assms*(*1*) ∗] **by** *simp*
}
ultimately have *inj-on* ($\lambda n.$ *upcrossing* $n\ \omega$) {..*Suc* N} **unfolding** *inj-on-def* **by** (*metis atMost-iff le-SucE linorder-less-linear*)
hence *card* (($\lambda n.$ *upcrossing* $n\ \omega$) ‘ {..*Suc* N}) = *Suc* (*Suc* N) **by** (*simp add*: *inj-on-iff-eq-card*[*THEN iffD1*])
moreover have ($\lambda n.$ *upcrossing* $n\ \omega$) ‘ {..*Suc* N} ⊆ {..N} **using** *upcrossing-le* **by** *blast*
moreover have *card* (($\lambda n.$ *upcrossing* $n\ \omega$) ‘ {..*Suc* N}) ≤ *Suc* N **using** *card-mono*[*OF - calculation*(*2*)] **by** *simp*
ultimately have *False* **by** *linarith*
}
thus *?thesis* **by** *blast*
qed
thus *?thesis* **using** *upcrossing-mono*[*OF assms*(*2*), *of* ω] *upcrossing-le*[*of n* ω] **by** *simp*
qed

lemma *downcrossing-eq-bound*:

assumes *a* < *b* *n* ≥ *N*
shows *downcrossing n ω* = *N*
using *upcrossing-le-downcrossing*[*of n ω*] *downcrossing-le*[*of n ω*] *upcrossing-eq-bound*[*OF assms*] **by** *simp*

lemma *stopping-time-crossings*:
assumes *adapted-process M F 0 X*
shows *stopping-time* (*upcrossing n*) *stopping-time* (*downcrossing n*)
proof −
have *stopping-time* (*upcrossing n*) ∧ *stopping-time* (*downcrossing n*)
proof (*induction n*)
case *0*
then show *?case* **unfolding** *upcrossing-simps downcrossing-simps*
using *stopping-time-const stopping-time-hitting-time*[*OF assms*] **by** *simp*
next
case (*Suc n*)
have *stopping-time* (*upcrossing* (*Suc n*)) **unfolding** *upcrossing-simps*
using *assms Suc downcrossing-le* **by** (*intro stopping-time-hitting-time′*) *auto*
moreover have *stopping-time* (*downcrossing* (*Suc n*)) **unfolding** *downcrossing-simps*
using *assms calculation upcrossing-le* **by** (*intro stopping-time-hitting-time′*) *auto*
ultimately show *?case* **by** *blast*
qed
thus *stopping-time* (*upcrossing n*) *stopping-time* (*downcrossing n*) **by** *blast+*
qed

lemmas *stopping-time-upcrossing* = *stopping-time-crossings*(*1*)
lemmas *stopping-time-downcrossing* = *stopping-time-crossings*(*2*)

— We define *upcrossings-before* as the number of upcrossings which take place strictly before time *N*.

definition *upcrossings-before* :: ′*a* ⇒ *nat* **where**
upcrossings-before = (*λω*. *Sup* {*n*. *upcrossing n ω* < *N*})

lemma *upcrossings-before-bdd-above*:
assumes *a* < *b*
shows *bdd-above* {*n*. *upcrossing n ω* < *N*}
proof −
have {*n*. *upcrossing n ω* < *N*} ⊆ {..<*N*} **unfolding** *lessThan-def Collect-mono-iff*
using *upcrossing-eq-bound*[*OF assms*] *linorder-not-less order-less-irrefl* **by** *metis*
thus *?thesis* **by** (*meson bdd-above-Iio bdd-above-mono*)
qed

lemma *upcrossings-before-less*:
assumes *a* < *b* *0* < *N*
shows *upcrossings-before ω* < *N*
proof −

have ∗: {*n*. *upcrossing n ω* < *N*} ⊆ {..<*N*} **unfolding** *lessThan-def Collect-mono-iff*
using *upcrossing-eq-bound*[*OF assms*(*1*)] *linorder-not-less order-less-irrefl* **by** *metis*
have *upcrossing 0 ω* < *N* **unfolding** *upcrossing-simps* **by** (*rule assms*)
moreover have *Sup* {..<*N*} < *N* **unfolding** *Sup-nat-def* **using** *assms* **by** *simp*
ultimately show *?thesis* **unfolding** *upcrossings-before-def* **using** *cSup-subset-mono*[*OF* - - ∗] **by** *force*
qed

lemma *upcrossings-before-less-implies-crossing-eq-bound*:
assumes *a* < *b upcrossings-before ω* < *n*
shows *upcrossing n ω* = *N*
downcrossing n ω = *N*
proof −
have ¬ *upcrossing n ω* < *N* **using** *assms upcrossings-before-bdd-above*[*of ω*] *upcrossings-before-def bdd-above-nat finite-Sup-less-iff* **by** *fastforce*
thus *upcrossing n ω* = *N* **using** *upcrossing-le*[*of n ω*] **by** *simp*
thus *downcrossing n ω* = *N* **using** *upcrossing-le-downcrossing*[*of n ω*] *downcrossing-le*[*of n ω*] **by** *simp*
qed

lemma *upcrossings-before-le*:
assumes *a* < *b*
shows *upcrossings-before ω* ≤ *N*
using *upcrossings-before-less assms less-le-not-le upcrossings-before-def*
by (*cases N*) *auto*

lemma *upcrossings-before-mem*:
assumes *a* < *b 0* < *N*
shows *upcrossings-before ω* ∈ {*n*. *upcrossing n ω* < *N*} ∩ {..<*N*}
proof −
have *upcrossing 0 ω* < *N* **using** *assms* **unfolding** *upcrossing-simps* **by** *simp*
hence {*n*. *upcrossing n ω* < *N*} ≠ {} **by** *blast*
moreover have *finite* {*n*. *upcrossing n ω* < *N*} **using** *upcrossings-before-bdd-above*[*OF assms*(*1*)] **by** (*simp add*: *bdd-above-nat*)
ultimately show *?thesis* **using** *Max-in upcrossings-before-less*[*OF assms*(*1*,*2*)] *Sup-nat-def upcrossings-before-def* **by** *auto*
qed

lemma *upcrossing-less-of-le-upcrossings-before*:
assumes *a* < *b 0* < *N n* ≤ *upcrossings-before ω*
shows *upcrossing n ω* < *N*
using *upcrossings-before-mem*[*OF assms*(*1*,*2*), *of ω*] *upcrossing-mono*[*OF assms*(*3*), *of ω*] **by** *simp*

lemma *upcrossings-before-sum-def*:
assumes *a* < *b*
shows *upcrossings-before ω* = (∑ *k*∈{*1*..*N*}. *indicator* {*n*. *upcrossing n ω* < *N*}

k)
proof (*cases N*)
  case *0*
  then show *?thesis* **unfolding** *upcrossings-before-def* **by** *simp*
next
  case (*Suc N'*)
  have *upcrossing 0 ω < N* **using** *assms Suc* **unfolding** *upcrossing-simps* **by** *simp*
  hence {*n. upcrossing n ω < N*} ≠ {} **by** *blast*
  hence ∗: ¬ *upcrossing n ω < N* **if** *n* ∈ {*upcrossings-before ω <..N*} **for** *n*
    using *finite-Sup-less-iff*[*THEN iffD1*, *OF bdd-above-nat*[*THEN iffD1*, *OF upcrossings-before-bdd-above*], *of ω n*]
    by (*metis that assms greaterThanAtMost-iff less-not-refl mem-Collect-eq upcrossings-before-def*)
  have ∗∗: *upcrossing n ω < N* **if** *n* ∈ {*1..upcrossings-before ω*} **for** *n*
   using *assms that Suc* **by** (*intro upcrossing-less-of-le-upcrossings-before*) *auto*
  have *upcrossings-before ω < N* **using** *upcrossings-before-less Suc assms* **by** *simp*
  hence {*1..N*} − {*1..upcrossings-before ω*} = {*upcrossings-before ω<..N*}
     {*1..N*} ∩ {*1..upcrossings-before ω*} = {*1..upcrossings-before ω*} **by** *force+*
  hence (∑ *k*∈{*1..N*}. *indicator* {*n. upcrossing n ω < N*} *k*) =
     (∑ *k*∈{*1..upcrossings-before ω*}. *indicator* {*n. upcrossing n ω < N*} *k*) + (∑ *k*∈{*upcrossings-before ω <..N*}. *indicator* {*n. upcrossing n ω < N*} *k*)
    using *sum.Int-Diff*[*OF finite-atLeastAtMost*, *of - 1 N* {*1..upcrossings-before ω*}] **by** *metis*
  also have ... = *upcrossings-before ω* **using** ∗ ∗∗ **by** *simp*
  finally show *?thesis* **by** *argo*
qed

lemma *upcrossings-before-measurable*:
  assumes *adapted-process M F 0 X a < b*
  shows *upcrossings-before* ∈ *borel-measurable M*
  unfolding *upcrossings-before-sum-def*[*OF assms*(*2*)]
  using *stopping-time-measurable*[*OF stopping-time-crossings*(*1*), *OF assms*(*1*)] **by** *simp*

lemma *upcrossings-before-measurable'*:
  assumes *adapted-process M F 0 X a < b*
  shows (*λω. real* (*upcrossings-before ω*)) ∈ *borel-measurable M*
  using *real-embedding-borel-measurable upcrossings-before-measurable*[*OF assms*] **by** *simp*

end

lemma *crossing-eq-crossing*:
  assumes *N ≤ N'*
    and *downcrossing X a b N n ω < N*
   shows *upcrossing X a b N n ω = upcrossing X a b N' n ω*
     downcrossing X a b N n ω = downcrossing X a b N' n ω
proof −
  have *upcrossing X a b N n ω = upcrossing X a b N' n ω* ∧ *downcrossing X a b*

N n ω = *downcrossing X a b N′ n ω* **using** *assms*(*2*)
proof (*induction n*)
case *0*
show *?case* **by** (*metis* (*no-types*, *lifting*) *upcrossing-simps*(*1*) *assms atLeast-0 bot-nat-0*.*extremum hitting-time-def hitting-time-eq-hitting-time inf-top*.*right-neutral leD downcrossing-mono downcrossing-simps*(*1*) *max-nat*.*left-neutral*)
next
case (*Suc n*)
hence *upper-less*: *upcrossing X a b N* (*Suc n*) *ω* < *N* **using** *upcrossing-le-downcrossing Suc order*.*strict-trans1* **by** *blast*
hence *lower-less*: *downcrossing X a b N n ω* < *N* **using** *downcrossing-le-upcrossing-Suc order*.*strict-trans1* **by** *blast*

obtain *j* **where** *j* ∈ {*downcrossing X a b N n ω*..<*N*} *X j ω* ∈ {*b*..}
using *hitting-time-less-iff*[*THEN iffD1*, *OF order-refl*] *upper-less* **by** (*force simp add*: *upcrossing-simps*)
hence *upper-eq*: *upcrossing X a b N* (*Suc n*) *ω* = *upcrossing X a b N′* (*Suc n*) *ω*
using *Suc*(*1*)[*OF lower-less*] *assms*(*1*)
by (*auto simp add*: *upcrossing-simps intro*!: *hitting-time-eq-hitting-time*)
obtain *j* **where** *j*: *j* ∈ {*upcrossing X a b N* (*Suc n*) *ω*..<*N*} *X j ω* ∈ {..*a*}
using *Suc*(*2*) *hitting-time-less-iff*[*THEN iffD1*, *OF order-refl*] **by** (*force simp add*: *downcrossing-simps*)
thus *?case* **unfolding** *downcrossing-simps upper-eq* **by** (*force intro*: *hitting-time-eq-hitting-time assms*)
qed
thus *upcrossing X a b N n ω* = *upcrossing X a b N′ n ω downcrossing X a b N n ω* = *downcrossing X a b N′ n ω* **by** *auto*
qed

lemma *crossing-eq-crossing′*:
assumes *N* ≤ *N′*
and *upcrossing X a b N* (*Suc n*) *ω* < *N*
shows *upcrossing X a b N* (*Suc n*) *ω* = *upcrossing X a b N′* (*Suc n*) *ω*
downcrossing X a b N n ω = *downcrossing X a b N′ n ω*
proof −
show *lower-eq*: *downcrossing X a b N n ω* = *downcrossing X a b N′ n ω*
using *downcrossing-le-upcrossing-Suc*[*THEN order*.*strict-trans1*] *crossing-eq-crossing assms* **by** *fast*
have ∃ *j*∈{*downcrossing X a b N n ω*..<*N*}. *X j ω* ∈ {*b*..} **using** *assms*(*2*) **by** (*intro hitting-time-less-iff*[*OF order-refl*, *THEN iffD1*]) (*simp add*: *upcrossing-simps lower-eq*)
then obtain *j* **where** *j*∈{*downcrossing X a b N n ω*..*N*} *X j ω* ∈ {*b*..} **by** *fastforce*
thus *upcrossing X a b N* (*Suc n*) *ω* = *upcrossing X a b N′* (*Suc n*) *ω*
unfolding *upcrossing-simps stopped-value-def* **using** *hitting-time-eq-hitting-time*[*OF assms*(*1*)] *lower-eq* **by** *metis*
qed

lemma *upcrossing-eq-upcrossing*:

assumes $N \leq N'$
and *upcrossing X a b N n ω < N*
shows *upcrossing X a b N n ω = upcrossing X a b N′ n ω*
using *crossing-eq-crossing′[OF assms(1)] assms(2) upcrossing-simps*
by (*cases n*) (*presburger*, *fast*)

lemma *upcrossings-before-zero*: *upcrossings-before X a b 0 ω = 0*
unfolding *upcrossings-before-def* **by** *simp*

lemma *upcrossings-before-less-exists-upcrossing*:
assumes $a < b$
and *upcrossing*: $N \leq L$ $X\ L\ \omega < a$ $L \leq U$ $b < X\ U\ \omega$
shows *upcrossings-before X a b N ω < upcrossings-before X a b (Suc U) ω*
proof −
have *upcrossing X a b (Suc U) (upcrossings-before X a b N ω) ω ≤ L*
using *assms upcrossing-le[THEN order-trans, OF upcrossing(1)]*
by (*cases 0 < N*, *subst upcrossing-eq-upcrossing[of N Suc U, symmetric, OF - upcrossing-less-of-le-upcrossings-before]*)
(*auto simp add: upcrossings-before-zero upcrossing-simps*)
hence *downcrossing X a b (Suc U) (upcrossings-before X a b N ω) ω ≤ U*
unfolding *downcrossing-simps* **using** *upcrossing* **by** (*force intro: hitting-time-le-iff[THEN iffD2]*)
hence *upcrossing X a b (Suc U) (Suc (upcrossings-before X a b N ω)) ω < Suc U*
unfolding *upcrossing-simps* **using** *upcrossing* **by** (*force intro: hitting-time-less-iff[THEN iffD2]*)
thus *?thesis* **using** *cSup-upper[OF - upcrossings-before-bdd-above[OF assms(1)]] upcrossings-before-def* **by** *fastforce*
qed

lemma *crossings-translate*:
upcrossing X a b N = upcrossing (λn ω. (X n ω + c)) (a + c) (b + c) N
downcrossing X a b N = downcrossing (λn ω. (X n ω + c)) (a + c) (b + c) N
proof −
have *upper*: *upcrossing X a b N n = upcrossing (λn ω. (X n ω + c)) (a + c) (b + c) N n* **for** *n*
proof (*induction n*)
case *0*
then show *?case* **by** (*simp only: upcrossing.simps*)
next
case (*Suc n*)
have ((+) *c* ‘ {..*a*}) = {..*a* + *c*} **by** *simp*
moreover have ((+) *c* ‘ {*b*..}) = {*b* + *c*..} **by** *simp*
ultimately show *?case* **unfolding** *upcrossing.simps* **using** *hitting-time-translate[of X {b..} c] hitting-time-translate[of X {..a} c] Suc* **by** *presburger*
qed
thus *upcrossing X a b N = upcrossing (λn ω. (X n ω + c)) (a + c) (b + c) N* **by** *blast*
have ((+) *c* ‘ {..*a*}) = {..*a* + *c*} **by** *simp*

thus *downcrossing X a b N* = *downcrossing* ($\lambda n\ \omega$. (*X n* ω + *c*)) (*a* + *c*) (*b* + *c*) *N* **using** *upper downcrossing-simps hitting-time-translate*[*of X* {..*a*} *c*] **by** *presburger*
qed

lemma *upcrossings-before-translate*:
upcrossings-before X a b N = *upcrossings-before* ($\lambda n\ \omega$. (*X n* ω + *c*)) (*a* + *c*) (*b* + *c*) *N*
using *upcrossings-before-def crossings-translate* **by** *simp*

lemma *crossings-pos-eq*:
assumes *a* < *b*
shows *upcrossing X a b N* = *upcrossing* ($\lambda n\ \omega$. *max 0* (*X n* ω − *a*)) *0* (*b* − *a*) *N*
downcrossing X a b N = *downcrossing* ($\lambda n\ \omega$. *max 0* (*X n* ω − *a*)) *0* (*b* − *a*) *N*
proof −
have ∗: *max 0* (*x* − *a*) ∈ {..*0*} ⟷ *x* − *a* ∈ {..*0*} *max 0* (*x* − *a*) ∈ {*b* − *a*..} ⟷ *x* − *a* ∈ {*b* − *a*..} **for** *x* **using** *assms* **by** *auto*
have *upcrossing X a b N* = *upcrossing* ($\lambda n\ \omega$. *X n* ω − *a*) *0* (*b* − *a*) *N* **using** *crossings-translate*[*of X a b N* − *a*] **by** *simp*
thus *upper*: *upcrossing X a b N* = *upcrossing* ($\lambda n\ \omega$. *max 0* (*X n* ω − *a*)) *0* (*b* − *a*) *N* **unfolding** *upcrossing-def hitting-time-def′* **using** ∗ **by** *presburger*

thus *downcrossing X a b N* = *downcrossing* ($\lambda n\ \omega$. *max 0* (*X n* ω − *a*)) *0* (*b* − *a*) *N*
unfolding *downcrossing-simps hitting-time-def′* **using** *upper* ∗ **by** *simp*
qed

lemma *upcrossings-before-mono*:
assumes *a* < *b N* ≤ *N′*
shows *upcrossings-before X a b N* ω ≤ *upcrossings-before X a b N′* ω
proof (*cases N*)
case *0*
then show *?thesis* **unfolding** *upcrossings-before-def* **by** *simp*
next
case (*Suc N′*)
hence *upcrossing X a b N 0* ω < *N* **unfolding** *upcrossing-simps* **by** *simp*
thus *?thesis* **unfolding** *upcrossings-before-def* **using** *upcrossings-before-bdd-above upcrossing-eq-upcrossing assms* **by** (*intro cSup-subset-mono*) *auto*
qed

lemma *upcrossings-before-pos-eq*:
assumes *a* < *b*
shows *upcrossings-before X a b N* = *upcrossings-before* ($\lambda n\ \omega$. *max 0* (*X n* ω − *a*)) *0* (*b* − *a*) *N*
using *upcrossings-before-def crossings-pos-eq*[*OF assms*] **by** *simp*

— We define *upcrossings* to be the total number of upcrossings a stochastic process completes as $N \longrightarrow \infty$.

definition *upcrossings* :: $(nat \Rightarrow 'a \Rightarrow real) \Rightarrow real \Rightarrow real \Rightarrow 'a \Rightarrow ennreal$ **where**
upcrossings X a b = ($\lambda\omega$. (*SUP N. ennreal* (*upcrossings-before X a b N* ω)))

lemma *upcrossings-measurable*:
assumes *adapted-process M F 0 X a* < *b*
shows *upcrossings X a b* ∈ *borel-measurable M*
unfolding *upcrossings-def*
using *upcrossings-before-measurable'*[*OF assms*] **by** (*auto intro*!: *borel-measurable-SUP*)

end

lemma (**in** *nat-finite-filtered-measure*) *integrable-upcrossings-before*:
assumes *adapted-process M F 0 X a* < *b*
shows *integrable M* ($\lambda\omega$. *real* (*upcrossings-before X a b N* ω))
proof −
have ($\int^+$ *x. ennreal* (*norm* (*real* (*upcrossings-before X a b N x*))) ∂M) ≤ ($\int^+$ *x. ennreal N* ∂M) **using** *upcrossings-before-le*[*OF assms*(*2*)] **by** (*intro nn-integral-mono*) *simp*
also have ... = *ennreal N* ∗ *emeasure M* (*space M*) **by** *simp*
also have ... < ∞ **by** (*metis emeasure-real ennreal-less-top ennreal-mult-less-top infinity-ennreal-def*)
finally show *?thesis* **by** (*intro integrableI-bounded upcrossings-before-measurable' assms*)
qed

4.2 Doob's Upcrossing Inequality

Doob's upcrossing inequality provides a bound on the expected number of upcrossings a submartingale completes before some point in time. The proof follows the proof presented in the paper *A Formalization of Doob's Martingale Convergence Theorems in mathlib* [1] [2].

context *nat-finite-filtered-measure*
begin

theorem *upcrossing-inequality*:
fixes *a b* :: *real* **and** *N* :: *nat*
assumes *submartingale M F 0 X*
shows (*b* − *a*) ∗ ($\int\omega$. *real* (*upcrossings-before X a b N* ω) ∂M) ≤ ($\int\omega$. *max 0* (*X N* ω − *a*) ∂M)
proof −
interpret *submartingale-linorder M F 0 X* **unfolding** *submartingale-linorder-def* **by** (*intro assms*)
show *?thesis*
proof (*cases a* < *b*)
case *True*
— We show the statement first for *X 0* non-negative and *X N* greater than or equal to *a*.

have ∗: (*b* − *a*) ∗ (∫*ω*. *real* (*upcrossings-before X a b N ω*) ∂*M*) ≤ (∫*ω*. *X N ω* ∂*M*)
if *asm*: *submartingale M F 0 X a* < *b* ⋀*ω*. *X 0 ω* ≥ *0* ⋀*ω*. *X N ω* ≥ *a*
for *a b X*
proof −
interpret *subm*: *submartingale M F 0 X* **by** (*intro asm*)
define *C* :: *nat* ⇒ *'a* ⇒ *real* **where** *C* = (*λn ω*. ∑*k* < *N*. *indicator* {*downcrossing X a b N k ω*..<*upcrossing X a b N* (*Suc k*) *ω*} *n*)
have *C-values*: *C n ω* ∈ {*0*, *1*} **for** *n ω*
proof (*cases* ∃*j* < *N*. *n* ∈ {*downcrossing X a b N j ω*..<*upcrossing X a b N* (*Suc j*) *ω*})
case *True*
then obtain *j* **where** *j*: *j* ∈ {..<*N*} *n* ∈ {*downcrossing X a b N j ω*..<*upcrossing X a b N* (*Suc j*) *ω*} **by** *blast*
{
fix *k l* :: *nat* **assume** *k-less-l*: *k* < *l*
hence *Suc-k-le-l*: *Suc k* ≤ *l* **by** *simp*
have {*downcrossing X a b N k ω*..<*upcrossing X a b N* (*Suc k*) *ω*} ∩ {*downcrossing X a b N l ω*..<*upcrossing X a b N* (*Suc l*) *ω*} =
{*downcrossing X a b N l ω*..<*upcrossing X a b N* (*Suc k*) *ω*}
using *k-less-l upcrossing-mono downcrossing-mono* **by** *simp*
moreover have *upcrossing X a b N* (*Suc k*) *ω* ≤ *downcrossing X a b N l ω*
using *upcrossing-le-downcrossing downcrossing-mono*[*OF Suc-k-le-l*] *order-trans* **by** *blast*
ultimately have {*downcrossing X a b N k ω*..<*upcrossing X a b N* (*Suc k*) *ω*} ∩ {*downcrossing X a b N l ω*..<*upcrossing X a b N* (*Suc l*) *ω*} = {} **by** *simp*
}
hence *disjoint-family-on* (*λk*. {*downcrossing X a b N k ω*..<*upcrossing X a b N* (*Suc k*) *ω*}) {..<*N*}
unfolding *disjoint-family-on-def*
by (*metis Int-commute linorder-less-linear*)
hence *C n ω* = *1* **unfolding** *C-def* **using** *sum-indicator-disjoint-family*[**where** *?f*=*λ-*. *1*] *j* **by** *fastforce*
thus *?thesis* **by** *blast*
next
case *False*
hence *C n ω* = *0* **unfolding** *C-def* **by** *simp*
thus *?thesis* **by** *simp*
qed
hence *C-interval*: *C n ω* ∈ {*0*..*1*} **for** *n ω* **by** (*metis atLeastAtMost-iff empty-iff insert-iff order.refl zero-less-one-class.zero-le-one*)

— We consider the discrete stochastic integral of *C* and *λn ω*. *1* − *C n ω*.
define *C'* **where** *C'* = (*λn ω*. ∑*k* < *n*. *C k ω* $*_R$ (*X* (*Suc k*) *ω* − *X k ω*))
define *one-minus-C'* **where** *one-minus-C'* = (*λn ω*. ∑*k* < *n*. (*1* − *C k ω*) $*_R$ (*X* (*Suc k*) *ω* − *X k ω*))

— We use the fact that the crossing times are stopping times to show that C is predictable.

have *adapted-C*: *adapted-process M F 0 C*
proof
fix *i*
have (*λω. indicat-real* {*downcrossing X a b N k ω..<upcrossing X a b N* (*Suc k*) *ω*} *i*) ∈ *borel-measurable* (*F i*) **for** *k*
unfolding *indicator-def*
using *stopping-time-upcrossing*[*OF subm.adapted-process-axioms*, *THEN stopping-time-measurable-gr*]
stopping-time-downcrossing[*OF subm.adapted-process-axioms*, *THEN stopping-time-measurable-le*]
by *force*
thus *C i* ∈ *borel-measurable* (*F i*) **unfolding** *C-def* **by** *simp*
qed
hence *adapted-process M F 0* (*λn ω. 1* − *C n ω*) **by** (*intro adapted-process.diff-adapted adapted-process-const*)
hence *submartingale-one-minus-C′*: *submartingale M F 0 one-minus-C′* **unfolding** *one-minus-C′-def* **using** *C-interval*
by (*intro submartingale-partial-sum-scaleR*[*of - - 1*] *submartingale-linorder.intro asm*) *auto*

have *C n* ∈ *borel-measurable M* **for** *n*
using *adapted-C adapted-process.adapted measurable-from-subalg subalg* **by** *blast*

have *integrable-C′*: *integrable M* (*C′ n*) **for** *n* **unfolding** *C′-def* **using** *C-interval*
by (*intro submartingale-partial-sum-scaleR*[*THEN submartingale.integrable*] *submartingale-linorder.intro adapted-C asm*) *auto*

— We show the following inequality, by using the fact that *one-minus-C′* is a submartingale.
have $integral^L$ *M* (*C′ n*) ≤ $integral^L$ *M* (*X n*) **for** *n*
proof −
interpret *subm′*: *submartingale-linorder M F 0 one-minus-C′* **unfolding** *submartingale-linorder-def* **by** (*rule submartingale-one-minus-C′*)
have *0* ≤ $integral^L$ *M* (*one-minus-C′ n*)
using *subm′.set-integral-le*[*OF sets.top*, **where** *i=0* **and** *j=n*] *space-F subm′.integrable* **by** (*fastforce simp add*: *set-integral-space one-minus-C′-def*)
moreover have *one-minus-C′ n ω* = (∑*k* < *n. X* (*Suc k*) *ω* − *X k ω*) − *C′ n ω* **for** *ω*
unfolding *one-minus-C′-def C′-def* **by** (*simp only*: *scaleR-diff-left sum-subtractf scale-one*)
ultimately have *0* ≤ (*LINT ω*|*M.* (∑*k* < *n. X* (*Suc k*) *ω* − *X k ω*)) − $integral^L$ *M* (*C′ n*)
using *subm.integrable integrable-C′*
by (*subst Bochner-Integration.integral-diff*[*symmetric*]) (*auto simp add*: *one-minus-C′-def*)
moreover have (*LINT ω*|*M.* (∑*k*<*n. X* (*Suc k*) *ω* − *X k ω*)) ≤ (*LINT ω*|*M. X n ω*) **using** *asm sum-lessThan-telescope*[*of λi. X i - n*] *subm.integrable*

by (*intro integral-mono*) *auto*
ultimately show *?thesis* **by** *linarith*
qed
moreover have $(b - a) * (\int \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M) \leq integral^L\ M\ (C'\ N)$
proof (*cases N*)
case *0*
then show *?thesis* **using** *C′-def upcrossings-before-zero* **by** *simp*
next
case (*Suc N′*)
{
fix ω
have *dc-not-N*: $downcrossing\ X\ a\ b\ N\ k\ \omega \neq N$ **if** $k < upcrossings\text{-}before\ X\ a\ b\ N\ \omega$ **for** *k*
by (*metis Suc Suc-leI asm*(*2*) *downcrossing-le-upcrossing-Suc leD that upcrossing-less-of-le-upcrossings-before zero-less-Suc*)
have *uc-not-N*:$upcrossing\ X\ a\ b\ N\ (Suc\ k)\ \omega \neq N$ **if** $k < upcrossings\text{-}before\ X\ a\ b\ N\ \omega$ **for** *k*
by (*metis Suc Suc-leI asm*(*2*) *order-less-irrefl that upcrossing-less-of-le-upcrossings-before zero-less-Suc*)

have *subset-lessThan-N*: $\{downcrossing\ X\ a\ b\ N\ i\ \omega..<upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega\} \subseteq \{..<N\}$ **if** $i < N$ **for** *i* **using** *that*
by (*simp add*: *lessThan-atLeast0 upcrossing-le*)

— First we rewrite the sum as follows:

have $C'\ N\ \omega = (\sum k<N.\ \sum i<N.\ indicator\ \{downcrossing\ X\ a\ b\ N\ i\ \omega..<upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega\}\ k * (X\ (Suc\ k)\ \omega - X\ k\ \omega))$
unfolding *C′-def C-def* **by** (*simp add*: *sum-distrib-right*)
also have $... = (\sum i<N.\ \sum k<N.\ indicator\ \{downcrossing\ X\ a\ b\ N\ i\ \omega..<upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega\}\ k * (X\ (Suc\ k)\ \omega - X\ k\ \omega))$
using *sum.swap* **by** *fast*
also have $... = (\sum i<N.\ \sum k\in\{..<N\} \cap \{downcrossing\ X\ a\ b\ N\ i\ \omega..<upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega\}.\ X\ (Suc\ k)\ \omega - X\ k\ \omega)$
by (*subst Indicator-Function.sum-indicator-mult*) *simp+*
also have $... = (\sum i<N.\ \sum k\in\{downcrossing\ X\ a\ b\ N\ i\ \omega..<upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega\}.\ X\ (Suc\ k)\ \omega - X\ k\ \omega)$
using *subset-lessThan-N*[*THEN Int-absorb1*] **by** *simp*
also have $... = (\sum i<N.\ X\ (upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega)\ \omega - X\ (downcrossing\ X\ a\ b\ N\ i\ \omega)\ \omega)$
by (*subst sum-Suc-diff′*[*OF downcrossing-le-upcrossing-Suc*]) *blast*
finally have $*$: $C'\ N\ \omega = (\sum i<N.\ X\ (upcrossing\ X\ a\ b\ N\ (Suc\ i)\ \omega)\ \omega - X\ (downcrossing\ X\ a\ b\ N\ i\ \omega)\ \omega)$ **.**

— For $k \leq N$, we consider three cases:
— 1. If $k < upcrossings\text{-}before\ X\ a\ b\ N\ \omega$, then $X\ (upcrossing\ X\ a\ b\ N\ (Suc\ k)\ \omega)\ \omega - X\ (downcrossing\ X\ a\ b\ N\ k\ \omega)\ \omega \geq b - a$
— 2. If $upcrossings\text{-}before\ X\ a\ b\ N\ \omega < k$, then X (*upcrossing X a b N*

(*Suc k*) ω) ω = X (*downcrossing X a b N k* ω) ω
— 3. If k = *upcrossings-before X a b N* ω, then X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω ≥ *0*

have *summand-zero-if*: X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω = *0* **if** k > *upcrossings-before X a b N* ω **for** k
using *that upcrossings-before-less-implies-crossing-eq-bound*[*OF asm*(*2*)] **by** *simp*

have *summand-nonneg-if*: X (*upcrossing X a b N* (*Suc* (*upcrossings-before X a b N* ω)) ω) ω − X (*downcrossing X a b N* (*upcrossings-before X a b N* ω) ω) ω ≥ *0*
using *upcrossings-before-less-implies-crossing-eq-bound*(*1*)[*OF asm*(*2*) *lessI*]
stopped-value-downcrossing[*of X a b N - ω*, *THEN order-trans*, *OF - asm*(*4*)[*of* ω]]
by (*cases downcrossing X a b N* (*upcrossings-before X a b N* ω) ω ≠ N) (*simp add*: *stopped-value-def*)+

have *interval*: {*upcrossings-before X a b N* ω..<N} − {*upcrossings-before X a b N* ω} = {*upcrossings-before X a b N* ω<..<N}
using *Diff-insert atLeastSucLessThan-greaterThanLessThan lessThan-Suc lessThan-minus-lessThan* **by** *metis*

have (b − a) ∗ *real* (*upcrossings-before X a b N* ω) = (∑ -<*upcrossings-before X a b N* ω. b − a) **by** *simp*
also have ... ≤ (∑ k<*upcrossings-before X a b N* ω. *stopped-value X* (*upcrossing X a b N* (*Suc k*)) ω − *stopped-value X* (*downcrossing X a b N k*) ω)
using *stopped-value-downcrossing*[*OF dc-not-N*] *stopped-value-upcrossing*[*OF uc-not-N*] **by** (*force intro*!: *sum-mono*)
also have ... = (∑ k<*upcrossings-before X a b N* ω. X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω) **unfolding** *stopped-value-def* **by** *blast*
also have ... ≤ (∑ k<*upcrossings-before X a b N* ω. X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω)
+ (∑ k∈{*upcrossings-before X a b N* ω}. X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω)
+ (∑ k∈{*upcrossings-before X a b N* ω<..<N}. X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω)
using *summand-zero-if summand-nonneg-if* **by** *auto*
also have ... = (∑ k<N. X (*upcrossing X a b N* (*Suc k*) ω) ω − X (*downcrossing X a b N k* ω) ω)
using *upcrossings-before-le*[*OF asm*(*2*)]
by (*subst sum.subset-diff*[**where** A={..<N} **and** B={..<*upcrossings-before X a b N* ω}], *simp*, *simp*,
subst sum.subset-diff[**where** A={..<N} − {..<*upcrossings-before X a b N* ω} **and** B={*upcrossings-before X a b N* ω}])
(*simp add*: *Suc asm*(*2*) *upcrossings-before-less*, *simp*, *simp add*: *interval*)
finally have (b − a) ∗ *real* (*upcrossings-before X a b N* ω) ≤ C′ N ω

using $*$ **by** *presburger*
}
thus *?thesis* **using** *integrable-upcrossings-before subm.adapted-process-axioms asm integrable-C'*
by (*subst integral-mult-right-zero*[*symmetric*], *intro integral-mono*) *auto*
qed
ultimately show *?thesis* **using** *order-trans* **by** *blast*
qed

have $(b - a) * (\int \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M) = (b - a) * (\int \omega.\ real\ (upcrossings\text{-}before\ (\lambda n\ \omega.\ max\ 0\ (X\ n\ \omega - a))\ 0\ (b - a)\ N\ \omega)\ \partial M)$
using *upcrossings-before-pos-eq*[*OF True*] **by** *simp*
also have ... $\le (\int \omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M)$
using $*$[*OF submartingale-linorder.max-0*[*OF submartingale-linorder.intro*, *OF submartingale.diff*, *OF assms supermartingale-const*], *of 0 b − a a*] *True* **by** *simp*
finally show *?thesis* **.**
next
case *False*
have $0 \le (\int \omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M)$ **by** *simp*
moreover have $0 \le (\int \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M)$ **by** *simp*
moreover have $b - a \le 0$ **using** *False* **by** *simp*
ultimately show *?thesis* **using** *mult-nonpos-nonneg order-trans* **by** *meson*
qed
qed

theorem *upcrossing-inequality-Sup*:
fixes *a b* :: *real*
assumes *submartingale M F 0 X*
shows $(b - a) * (\int^{+} \omega.\ upcrossings\ X\ a\ b\ \omega\ \partial M) \le (SUP\ N.\ (\int^{+} \omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M))$
proof −
interpret *submartingale M F 0 X* **by** (*intro assms*)
show *?thesis*
proof (*cases* $a < b$)
case *True*
have $(\int^{+} \omega.\ upcrossings\ X\ a\ b\ \omega\ \partial M) = (SUP\ N.\ (\int^{+} \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M))$
unfolding *upcrossings-def*
using *upcrossings-before-mono True upcrossings-before-measurable'*[*OF adapted-process-axioms*]
by (*auto intro*: *nn-integral-monotone-convergence-SUP simp add*: *mono-def le-funI*)
hence $(b - a) * (\int^{+} \omega.\ upcrossings\ X\ a\ b\ \omega\ \partial M) = (SUP\ N.\ (b - a) * (\int^{+} \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M))$
by (*simp add*: *SUP-mult-left-ennreal*)
moreover
{
fix *N*
have $(\int^{+} \omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M) = (\int \omega.\ real\ (upcrossings\text{-}before$

$X\ a\ b\ N\ \omega)\ \partial M)$
by (*force intro*!: *nn-integral-eq-integral integrable-upcrossings-before True adapted-process-axioms*)
moreover have $(\int^+\omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M) = (\int\omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M)$
using *Bochner-Integration.integrable-diff*[*OF integrable integrable-const*]
by (*force intro*!: *nn-integral-eq-integral*)
ultimately have $(b - a) * (\int^+\omega.\ real\ (upcrossings\text{-}before\ X\ a\ b\ N\ \omega)\ \partial M) \le (\int^+\omega.\ max\ 0\ (X\ N\ \omega - a)\ \partial M)$
using *upcrossing-inequality*[*OF assms*, *of b a N*] *True ennreal-mult'*[*symmetric*] **by** *simp*
}
ultimately show *?thesis* **by** (*force intro*!: *Sup-mono*)
qed (*simp add*: *ennreal-neg*)
qed

end

end

5 Doob's First Martingale Convergence Theorem

theory *Doob-Convergence*
imports *Upcrossing*
begin

context *nat-finite-filtered-measure*
begin

Doob's martingale convergence theorem states that, if we have a submartingale where the supremum over the mean of the positive parts is finite, then the limit process exists almost surely and is integrable. Furthermore, the limit process is measurable with respect to the smallest σ-algebra containing all of the σ-algebras in the filtration. The argumentation below is taken mostly from [3].

theorem *submartingale-convergence-AE*:
fixes $X :: nat \Rightarrow {'a} \Rightarrow real$
assumes *submartingale M F 0 X*
and $\bigwedge n.\ (\int\omega.\ max\ 0\ (X\ n\ \omega)\ \partial M) \le C$
obtains X_{lim} **where** $AE\ \omega\ in\ M.\ (\lambda n.\ X\ n\ \omega) \longrightarrow X_{lim}\ \omega$
integrable $M\ X_{lim}$
$X_{lim} \in borel\text{-}measurable\ (F_\infty)$
proof –
interpret *submartingale-linorder M F 0 X* **unfolding** *submartingale-linorder-def* **by** (*rule assms*)

— We first show that the number of upcrossings has to be finite using the upcrossing inequality we proved above.

have *finite-upcrossings*: *AE* ω *in* M. *upcrossings X a b* $\omega \neq \infty$ **if** $a < b$ **for** *a b*
proof –
have *C-nonneg*: $C \geq 0$ **using** *assms*(*2*) **by** (*meson Bochner-Integration.integral-nonneg linorder-not-less max.cobounded1 order-less-le-trans*)
{
fix *n*
have $(\int^+\omega.\ max\ 0\ (X\ n\ \omega - a)\ \partial M) \leq (\int^+\omega.\ max\ 0\ (X\ n\ \omega) + |a|\ \partial M)$ **by** (*fastforce intro*: *nn-integral-mono ennreal-leI*)
also have ... $= (\int^+\omega.\ max\ 0\ (X\ n\ \omega)\ \partial M) + |a| * emeasure\ M\ (space\ M)$ **by** (*simp add*: *nn-integral-add*)
also have ... $= (\int\omega.\ max\ 0\ (X\ n\ \omega)\ \partial M) + |a| * emeasure\ M\ (space\ M)$ **using** *integrable* **by** (*simp add*: *nn-integral-eq-integral*)
also have ... $\leq C + |a| * emeasure\ M\ (space\ M)$ **using** *assms*(*2*) *ennreal-leI* **by** *simp*
finally have $(\int^+\omega.\ max\ 0\ (X\ n\ \omega - a)\ \partial M) \leq C + |a| * enn2real\ (emeasure\ M\ (space\ M))$ **using** *finite-emeasure-space C-nonneg* **by** (*simp add*: *ennreal-enn2real-if ennreal-mult*)
}
hence $(SUP\ N.\ \int^+ x.\ ennreal\ (max\ 0\ (X\ N\ x - a))\ \partial M) / (b - a) \leq ennreal\ (C + |a| * enn2real\ (emeasure\ M\ (space\ M))) / (b - a)$ **by** (*fast intro*: *divide-right-mono-ennreal Sup-least*)
moreover have $ennreal\ (C + |a| * enn2real\ (emeasure\ M\ (space\ M))) / (b - a) < \infty$ **using** *that C-nonneg* **by** (*subst divide-ennreal*) *auto*
moreover have $integral^N\ M\ (upcrossings\ X\ a\ b) \leq (SUP\ N.\ \int^+ x.\ ennreal\ (max\ 0\ (X\ N\ x - a))\ \partial M) / (b - a)$
using *upcrossing-inequality-Sup*[*OF assms*(*1*), *of b a*, *THEN divide-right-mono-ennreal*, *of b* – *a*]
ennreal-mult-divide-eq mult.commute[*of ennreal* (*b* – *a*)] *that* **by** *simp*
ultimately show *?thesis* **using** *upcrossings-measurable adapted-process-axioms that* **by** (*intro nn-integral-noteq-infinite*) *auto*
qed

— Since the number of upcrossings are finite, limsup and liminf have to agree almost everywhere. To show this we consider the following countable set, which has zero measure.

define *S* **where** $S = ((\lambda(a :: real, b).\ \{\omega \in space\ M.\ liminf\ (\lambda n.\ ereal\ (X\ n\ \omega)) < ereal\ a \wedge ereal\ b < limsup\ (\lambda n.\ ereal\ (X\ n\ \omega))\})\ `\ \{(a, b) \in \mathbb{Q} \times \mathbb{Q}.\ a < b\})$

have $(0, 1) \in \{(a :: real, b).\ (a, b) \in \mathbb{Q} \times \mathbb{Q} \wedge a < b\}$ **unfolding** *Rats-def* **by** *simp*
moreover have *countable* $\{(a, b).\ (a, b) \in \mathbb{Q} \times \mathbb{Q} \wedge a < b\}$ **by** (*blast intro*: *countable-subset*[*OF - countable-SIGMA*[*OF countable-rat countable-rat*]])
ultimately have *from-nat-into-S*: *range* (*from-nat-into S*) = *S from-nat-into S n* $\in$ *S* **for** *n*
unfolding *S-def*
by (*auto intro*!: *range-from-nat-into from-nat-into simp only*: *Rats-def*)
{

fix a b :: *real*
assume *a-less-b*: $a < b$
then obtain N **where** N: $x \in$ *space* $M - N \Longrightarrow$ *upcrossings* X a b $x \neq \infty$ $N \in$ *null-sets* M **for** x **using** *AE-E3*[*OF finite-upcrossings*] **by** *blast*
{
fix ω
assume *liminf-limsup*: *liminf* (λn. X n ω) $<$ a b $<$ *limsup* (λn. X n ω)
have *upcrossings* X a b $\omega = \infty$
proof −
{
fix n
have $\exists m$. *upcrossings-before* X a b m $\omega \geq n$
proof (*induction* n)
case *0*
have *Sup* {n. *upcrossing* X a b *0* n ω $<$ *0*} = *0* **by** *simp*
then show *?case* **unfolding** *upcrossings-before-def* **by** *blast*
next
case (*Suc* n)
then obtain m **where** m: $n \leq$ *upcrossings-before* X a b m ω **by** *blast*
obtain l **where** l: $l \geq m$ X l $\omega < a$ **using** *liminf-upper-bound*[*OF liminf-limsup*(*1*), *of m*] *nless-le* **by** *auto*
obtain u **where** u: $u \geq l$ X u $\omega > b$ **using** *limsup-lower-bound*[*OF liminf-limsup*(*2*), *of l*] *nless-le* **by** *auto*
show *?case* **using** *upcrossings-before-less-exists-upcrossing*[*OF a-less-b*, **where** *?X=X*, *OF l u*] m **by** (*metis Suc-leI le-neq-implies-less*)
qed
}
thus *?thesis* **unfolding** *upcrossings-def* **by** (*simp add*: *ennreal-SUP-eq-top*)
qed
}
hence {$\omega \in$ *space* M. *liminf* (λn. *ereal* (X n ω)) $<$ *ereal* a $\wedge$ *ereal* b $<$ *limsup* (λn. *ereal* (X n ω))} $\subseteq N$ **using** N **by** *blast*
moreover have {$\omega \in$ *space* M. *liminf* (λn. *ereal* (X n ω)) $<$ *ereal* a $\wedge$ *ereal* b $<$ *limsup* (λn. *ereal* (X n ω))} $\cap N \in$ *null-sets* M **by** (*force intro*: *null-set-Int1*[*OF N*(*2*)])
ultimately have *emeasure* M {$\omega \in$ *space* M. *liminf* (λn. *ereal* (X n ω)) $< a$ $\wedge$ $b <$ *limsup* (λn. *ereal* (X n ω))} = *0* **by** (*simp add*: *Int-absorb1 Int-commute null-setsD1*)
}
hence *emeasure* M (*from-nat-into* S n) = *0* **for** n **using** *from-nat-into-S*(*2*)[*of n*] **unfolding** *S-def* **by** *force*
moreover have $S \subseteq M$ **unfolding** *S-def* **by** *force*
ultimately have *emeasure* M ($\bigcup$ (*range* (*from-nat-into* S))) = *0* **using** *from-nat-into-S* **by** (*intro emeasure-UN-eq-0*) *auto*
moreover have ($\bigcup$ S) = {$\omega \in$ *space* M. *liminf* (λn. *ereal* (X n ω)) $\neq$ *limsup* (λn. *ereal* (X n ω))} (**is** *?L* = *?R*)
proof −
{
fix ω

assume *asm*: $\omega \in$ *?L*
then obtain *a b* :: *real* **where** $a < b$ *liminf* (λn. *ereal* ($X\ n\ \omega$)) $<$ *ereal a* $\wedge$ *ereal b* $<$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **unfolding** *S-def* **by** *blast*
hence *liminf* (λn. *ereal* ($X\ n\ \omega$)) $\neq$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **using** *ereal-less-le order.asym* **by** *fastforce*
hence $\omega \in$ *?R* **using** *asm* **unfolding** *S-def* **by** *blast*
}
moreover
{
fix ω
assume *asm*: $\omega \in$ *?R*
hence *liminf* (λn. *ereal* ($X\ n\ \omega$)) $<$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **using** *Liminf-le-Limsup*[*of sequentially*] *less-eq-ereal-def* **by** *auto*
then obtain a' **where** a': *liminf* (λn. *ereal* ($X\ n\ \omega$)) $<$ *ereal* a' *ereal* $a' <$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **using** *ereal-dense2* **by** *blast*
then obtain b' **where** b': *ereal* $a' <$ *ereal* b' *ereal* $b' <$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **using** *ereal-dense2* **by** *blast*
hence $a' < b'$ **by** *simp*
then obtain a **where** a: $a \in \mathbb{Q}$ $a' < a$ $a < b'$ **using** *Rats-dense-in-real* **by** *blast*
then obtain b **where** b: $b \in \mathbb{Q}$ $a < b$ $b < b'$ **using** *Rats-dense-in-real* **by** *blast*
have *liminf* (λn. *ereal* ($X\ n\ \omega$)) $<$ *ereal a* **using** *a* a' *le-ereal-less order-less-imp-le* **by** *meson*
moreover have *ereal b* $<$ *limsup* (λn. *ereal* ($X\ n\ \omega$)) **using** *b* b' *order-less-imp-le ereal-less-le* **by** *meson*
ultimately have $\omega \in$ *?L* **unfolding** *S-def* **using** *a b asm* **by** *blast*
}
ultimately show *?thesis* **by** *blast*
qed
ultimately have *emeasure M* $\{\omega \in$ *space M*. *liminf* (λn. *ereal* ($X\ n\ \omega$)) $\neq$ *limsup* (λn. *ereal* ($X\ n\ \omega$))$\}$ $=$ *0* **using** *from-nat-into-S* **by** *argo*
hence *liminf-limsup-AE*: *AE* ω *in M*. *liminf* (λn. $X\ n\ \omega$) $=$ *limsup* (λn. $X\ n\ \omega$) **by** (*intro AE-iff-measurable*[*THEN iffD2*, *OF - refl*]) *auto*
hence *convergent-AE*: *AE* ω *in M*. *convergent* (λn. *ereal* ($X\ n\ \omega$)) **using** *convergent-ereal* **by** *fastforce*

— Hence the limit exists almost everywhere.

have *bounded-pos-part*: *ennreal* ($\int \omega$. *max 0* ($X\ n\ \omega$) ∂M) $\leq$ *ennreal C* **for** *n* **using** *assms*(*2*) *ennreal-leI* **by** *blast*

— Integral of positive part is $< \infty$.

{
fix ω
assume *asm*: *convergent* (λn. *ereal* ($X\ n\ \omega$))
hence (λn. *max 0* (*ereal* ($X\ n\ \omega$))) $\longrightarrow$ *max 0* (*lim* (λn. *ereal* ($X\ n\ \omega$)))
using *convergent-LIMSEQ-iff isCont-tendsto-compose continuous-max continuous-const continuous-ident continuous-at-e2ennreal*

by *fast*

hence $(\lambda n.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega)))) \longrightarrow e2ennreal\ (max\ 0\ (lim\ (\lambda n.\ ereal\ (X\ n\ \omega))))$

using *isCont-tendsto-compose continuous-at-e2ennreal* **by** *blast*

moreover have $lim\ (\lambda n.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega)))) = e2ennreal\ (max\ 0\ (lim\ (\lambda n.\ ereal\ (X\ n\ \omega))))$ **using** *limI calculation* **by** *blast*

ultimately have $e2ennreal\ (max\ 0\ (liminf\ (\lambda n.\ ereal\ (X\ n\ \omega)))) = liminf\ (\lambda n.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega))))$ **using** *convergent-liminf-cl* **by** (*metis asm convergent-def limI*)

}

hence $(\int^{+}\omega.\ e2ennreal\ (max\ 0\ (liminf\ (\lambda n.\ ereal\ (X\ n\ \omega))))\ \partial M) = (\int^{+}\omega.\ liminf\ (\lambda n.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega))))\ \partial M)$ **using** *convergent-AE* **by** (*fast intro*: *nn-integral-cong-AE*)

moreover have $(\int^{+}\omega.\ liminf\ (\lambda n.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega))))\ \partial M) \leq liminf\ (\lambda n.\ (\int^{+}\omega.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega)))\ \partial M))$

by (*intro nn-integral-liminf*) *auto*

moreover have $(\int^{+}\omega.\ e2ennreal\ (max\ 0\ (ereal\ (X\ n\ \omega)))\ \partial M) = ennreal\ (\int \omega.\ max\ 0\ (X\ n\ \omega)\ \partial M)$ **for** *n*

using *e2ennreal-ereal ereal-max-0*

by (*subst nn-integral-eq-integral*[*symmetric*]) (*fastforce intro*!: *nn-integral-cong integrable* | *presburger*)+

moreover have *liminf-pos-part-finite*: $liminf\ (\lambda n.\ ennreal\ (\int \omega.\ max\ 0\ (X\ n\ \omega)\ \partial M)) < \infty$

unfolding *liminf-SUP-INF*

using *Inf-lower2*[*OF - bounded-pos-part*]

by (*intro order.strict-trans1*[*OF Sup-least*, *of - ennreal C*]) (*metis* (*mono-tags*, *lifting*) *atLeast-iff imageE image-eqI order.refl*, *simp*)

ultimately have *pos-part-finite*: $(\int^{+}\omega.\ e2ennreal\ (max\ 0\ (liminf\ (\lambda n.\ ereal\ (X\ n\ \omega))))\ \partial M) < \infty$ **by** *force*

— Integral of negative part is $< \infty$.

{

fix ω

assume *asm*: $convergent\ (\lambda n.\ ereal\ (X\ n\ \omega))$

hence $(\lambda n.\ -\ min\ 0\ (ereal\ (X\ n\ \omega))) \longrightarrow -\ min\ 0\ (lim\ (\lambda n.\ ereal\ (X\ n\ \omega)))$

using *convergent-LIMSEQ-iff isCont-tendsto-compose continuous-min continuous-const continuous-ident continuous-at-e2ennreal*

by *fast*

hence $(\lambda n.\ e2ennreal\ (-\ min\ 0\ (ereal\ (X\ n\ \omega)))) \longrightarrow e2ennreal\ (-\ min\ 0\ (lim\ (\lambda n.\ ereal\ (X\ n\ \omega))))$

using *isCont-tendsto-compose continuous-at-e2ennreal* **by** *blast*

moreover have $lim\ (\lambda n.\ e2ennreal\ (-\ min\ 0\ (ereal\ (X\ n\ \omega)))) = e2ennreal\ (-\ min\ 0\ (lim\ (\lambda n.\ ereal\ (X\ n\ \omega))))$ **using** *limI calculation* **by** *blast*

ultimately have $e2ennreal\ (-\ min\ 0\ (liminf\ (\lambda n.\ ereal\ (X\ n\ \omega)))) = liminf\ (\lambda n.\ e2ennreal\ (-\ min\ 0\ (ereal\ (X\ n\ \omega))))$ **using** *convergent-liminf-cl* **by** (*metis asm convergent-def limI*)

}

hence (∫⁺ω. *e2ennreal* (− *min 0* (*liminf* (λ*n*. *ereal* (*X n* ω)))) ∂*M*) = (∫⁺ω. *liminf* (λ*n*. *e2ennreal* (− *min 0* (*ereal* (*X n* ω)))) ∂*M*) **using** *convergent-AE* **by** (*fast intro*: *nn-integral-cong-AE*)
moreover have (∫⁺ω. *liminf* (λ*n*. *e2ennreal* (− *min 0* (*ereal* (*X n* ω)))) ∂*M*) ≤ *liminf* (λ*n*. (∫⁺ω. *e2ennreal* (− *min 0* (*ereal* (*X n* ω))) ∂*M*))
by (*intro nn-integral-liminf*) *auto*
moreover have (∫⁺ω. *e2ennreal* (− *min 0* (*ereal* (*X n* ω))) ∂*M*) = (∫ω. *max 0* (*X n* ω) ∂*M*) − (∫ω. *X n* ω ∂*M*) **for** *n*
proof −
have ∗: (− *min 0 c*) = *max 0 c* − *c* **if** *c* ≠ ∞ **for** *c* :: *ereal* **using** *that* **by** (*cases c* ≥ *0*) *auto*
hence (∫⁺ω. *e2ennreal* (− *min 0* (*ereal* (*X n* ω))) ∂*M*) = (∫⁺ω. *e2ennreal* (*max 0* (*ereal* (*X n* ω)) − (*ereal* (*X n* ω))) ∂*M*) **by** *simp*
also have ... = (∫⁺ ω. *ennreal* (*max 0* (*X n* ω) − (*X n* ω)) ∂*M*) **using** *e2ennreal-ereal ereal-max-0 ereal-minus*(*1*) **by** (*intro nn-integral-cong*) *presburger*
also have ... = (∫ω. *max 0* (*X n* ω) − (*X n* ω) ∂*M*) **using** *integrable* **by** (*intro nn-integral-eq-integral*) *auto*
finally show *?thesis* **using** *Bochner-Integration.integral-diff integrable* **by** *simp*
qed
moreover have *liminf* (λ*n*. *ennreal* ((∫ω. *max 0* (*X n* ω) ∂*M*) − (∫ω. *X n* ω ∂*M*))) < ∞
proof −
{
fix *n A*
assume *asm*: *ennreal* ((∫ω. *max 0* (*X n* ω) ∂*M*) − (∫ω. *X n* ω ∂*M*)) ∈ *A*
have (∫ω. *X 0* ω ∂*M*) ≤ (∫ω. *X n* ω ∂*M*) **using** *set-integral-le*[*OF sets.top order-refl*, *of n*] *space-F* **by** (*simp add*: *integrable set-integral-space*)
hence (∫ω. *max 0* (*X n* ω) ∂*M*) − (∫ω. *X n* ω ∂*M*) ≤ *C* − (∫ω. *X 0* ω ∂*M*) **using** *assms*(*2*)[*of n*] **by** *argo*
hence *ennreal* ((∫ω. *max 0* (*X n* ω) ∂*M*) − (∫ω. *X n* ω ∂*M*)) ≤ *ennreal* (*C* − (∫ω. *X 0* ω ∂*M*)) **using** *ennreal-leI* **by** *blast*
hence *Inf A* ≤ *ennreal* (*C* − (∫ω. *X 0* ω ∂*M*)) **by** (*rule Inf-lower2*[*OF asm*])
}
thus *?thesis*
unfolding *liminf-SUP-INF*
by (*intro order.strict-trans1*[*OF Sup-least*, *of - ennreal* (*C* − (∫ω. *X 0* ω ∂*M*))]) (*metis* (*no-types*, *lifting*) *atLeast-iff imageE image-eqI order.refl order-trans*, *simp*)
qed
ultimately have *neg-part-finite*: (∫⁺ ω. *e2ennreal* (− (*min 0* (*liminf* (λ*n*. *ereal* (*X n* ω))))) ∂*M*) < ∞ **by** *simp*

— Putting it all together now to show that the limit is integrable and < ∞ a.e.

have *e2ennreal* |*liminf* (λ*n*. *ereal* (*X n* ω))| = *e2ennreal* (*max 0* (*liminf* (λ*n*. *ereal* (*X n* ω)))) + *e2ennreal* (− (*min 0* (*liminf* (λ*n*. *ereal* (*X n* ω))))) **for** ω
unfolding *ereal-abs-max-min*
by (*simp add*: *eq-onp-same-args max-def plus-ennreal.abs-eq*)
hence (∫⁺ ω. *e2ennreal* |*liminf* (λ*n*. *ereal* (*X n* ω))| ∂*M*) = (∫⁺ ω. *e2ennreal*

(*max 0* (*liminf* (λn. *ereal* (*X n* ω)))) ∂M) + ($\int^+$ ω. *e2ennreal* (− (*min 0* (*liminf* (λn. *ereal* (*X n* ω))))) ∂M) **by** (*auto intro*: *nn-integral-add*)
  hence *nn-integral-finite*: ($\int^+$ ω. *e2ennreal* |*liminf* (λn. *ereal* (*X n* ω))| ∂M) $\neq$ ∞ **using** *pos-part-finite neg-part-finite* **by** *auto*
  hence *finite-AE*: *AE* ω *in M*. *e2ennreal* |*liminf* (λn. *ereal* (*X n* ω))| $\neq \infty$ **by** (*intro nn-integral-noteq-infinite*) *auto*
  moreover
  {
    fix ω
    assume *asm*: *liminf* (λn. *X n* ω) = *limsup* (λn. *X n* ω) |*liminf* (λn. *ereal* (*X n* ω))| $\neq \infty$
    hence (λn. *X n* ω) $\longrightarrow$ *real-of-ereal* (*liminf* (λn. *X n* ω)) **using** *limsup-le-liminf-real ereal-real′* **by** *simp*
  }
  ultimately have *converges*: *AE* ω *in M*. (λn. *X n* ω) $\longrightarrow$ *real-of-ereal* (*liminf* (λn. *X n* ω)) **using** *liminf-limsup-AE* **by** *fastforce*

  {
    fix ω
    assume *e2ennreal* |*liminf* (λn. *ereal* (*X n* ω))| $\neq \infty$
    hence |*liminf* (λn. *ereal* (*X n* ω))| $\neq \infty$ **by** *force*
    hence *e2ennreal* |*liminf* (λn. *ereal* (*X n* ω))| = *ennreal* (*norm* (*real-of-ereal* (*liminf* (λn. *ereal* (*X n* ω))))) **by** *fastforce*
  }
  hence ($\int^+$ ω. *e2ennreal* |*liminf* (λn. *ereal* (*X n* ω))| ∂M) = ($\int^+$ ω. *ennreal* (*norm* (*real-of-ereal* (*liminf* (λn. *ereal* (*X n* ω))))) ∂M) **using** *finite-AE* **by** (*fast intro*: *nn-integral-cong-AE*)
  hence ($\int^+$ ω. *ennreal* (*norm* (*real-of-ereal* (*liminf* (λn. *ereal* (*X n* ω))))) ∂M) < ∞ **using** *nn-integral-finite* **by** (*simp add*: *order-less-le*)
  hence *integrable M* ($\lambda\omega$. *real-of-ereal* (*liminf* (λn. *X n* ω))) **by** (*intro integrableI-bounded*) *auto*
  moreover have ($\lambda\omega$. *real-of-ereal* (*liminf* (λn. *X n* ω))) $\in$ *borel-measurable* F_∞ **using** *borel-measurable-liminf*[*OF F-infinity-measurableI*] *adapted* **by** *measurable*
  ultimately show *?thesis* **using** *that converges* **by** *presburger*
qed

— We state the theorem again for martingales and supermartingales.

corollary *supermartingale-convergence-AE*:
  fixes *X* :: *nat* $\Rightarrow$ *′a* $\Rightarrow$ *real*
  assumes *supermartingale M F 0 X*
    and $\bigwedge n$. ($\int \omega$. *max 0* (− *X n* ω) ∂M) $\leq C$
  obtains X_{lim} **where** *AE* ω *in M*. (λn. *X n* ω) $\longrightarrow$ X_{lim} ω
    integrable M X_{lim}
    $X_{lim} \in$ *borel-measurable* (F_∞)
proof −
  obtain *Y* **where** $*$: *AE* ω *in M*. (λn. − *X n* ω) $\longrightarrow$ *Y* ω *integrable M Y Y* $\in$ *borel-measurable* (F_∞)
    using *supermartingale.uminus*[*OF assms*(*1*), *THEN submartingale-convergence-AE*]

assms(*2*) **by** *auto*
hence $AE\ \omega\ in\ M.\ (\lambda n.\ X\ n\ \omega) \longrightarrow (-\ Y)\ \omega$ *integrable* $M\ (-\ Y)$ $-\ Y \in$ *borel-measurable* (F_∞)
using *isCont-tendsto-compose*[*OF isCont-minus*, *OF continuous-ident*] *integrable-minus borel-measurable-uminus* **unfolding** *fun-Compl-def* **by** *fastforce+*
thus *?thesis* **using** *that*[*of* $-\ Y$] **by** *blast*
qed

corollary *martingale-convergence-AE*:
fixes $X :: nat \Rightarrow 'a \Rightarrow real$
assumes *martingale M F 0 X*
and $\bigwedge n.\ (\int \omega.\ |X\ n\ \omega|\ \partial M) \le C$
obtains X_{lim} **where** $AE\ \omega\ in\ M.\ (\lambda n.\ X\ n\ \omega) \longrightarrow X_{lim}\ \omega$
integrable $M\ X_{lim}$
$X_{lim} \in$ *borel-measurable* (F_∞)
proof −
interpret *martingale-linorder M F 0 X* **unfolding** *martingale-linorder-def* **by** (*rule assms*)
have $max\ 0\ (X\ n\ \omega) \le |X\ n\ \omega|$ **for** $n\ \omega$ **by** *linarith*
hence $(\int \omega.\ max\ 0\ (X\ n\ \omega)\ \partial M) \le C$ **for** n **using** *assms*(*2*)[*THEN dual-order.trans*, *OF integral-mono*, *OF integrable-max*] *integrable* **by** *fast*
thus *?thesis* **using** *that submartingale-convergence-AE*[*OF submartingale-axioms*] **by** *blast*
qed

corollary *martingale-nonneg-convergence-AE*:
fixes $X :: nat \Rightarrow 'a \Rightarrow real$
assumes *martingale M F 0 X* $\bigwedge n.\ AE\ \omega\ in\ M.\ X\ n\ \omega \ge 0$
obtains X_{lim} **where** $AE\ \omega\ in\ M.\ (\lambda n.\ X\ n\ \omega) \longrightarrow X_{lim}\ \omega$
integrable $M\ X_{lim}$
$X_{lim} \in$ *borel-measurable* (F_∞)
proof −
interpret *martingale-linorder M F 0 X* **unfolding** *martingale-linorder-def* **by** (*rule assms*)
have $AE\ \omega\ in\ M.\ max\ 0\ (-\ X\ n\ \omega) = 0$ **for** n **using** *assms*(*2*)[*of n*] **by** *force*
hence $(\int \omega.\ max\ 0\ (-\ X\ n\ \omega)\ \partial M) \le 0$ **for** n **by** (*simp add*: *integral-eq-zero-AE*)
thus *?thesis* **using** *that supermartingale-convergence-AE*[*OF supermartingale-axioms*] **by** *blast*
qed

end

end

References

[1] R. Degenne and K. Ying. A Formalization of Doob's Martingale Convergence Theorems in mathlib. In *Proceedings of the 12th ACM SIG-*

PLAN International Conference on Certified Programs and Proofs. Association for Computing Machinery, New York, United States, 2022. arXiv:2212.05578.

[2] R. Degenne and K. Ying. probability.martingale.basic - mathlib, 2022. https://leanprover-community.github.io/mathlib_docs/probability/martingale/basic.html, Last Accessed: 15 Feb 2024.

[3] R. Durrett. *Probability: Theory and Examples*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. URL: https://books.google.de/books?id=b22MDwAAQBAJ.

[4] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, 2020.

[5] J. Hölzl and A. Heller. Three Chapters of Measure Theory in Isabelle/HOL. In M. C. J. D. van Eekelen, H. Geuvers, J. Schmaltz, and F. Wiedijk, editors, *Interactive Theorem Proving (ITP 2011)*, volume 6898 of *LNCS*, pages 135–151, 2011. doi:10.1007/978-3-642-22863-6_12.

[6] A. Keskin. A Formalization of Martingales in Isabelle/HOL. Bachelor's thesis, Technical University of Munich, 2023. arXiv:2311.06188.

[7] A. Keskin. Martingales. *Archive of Formal Proofs*, November 2023. https://isa-afp.org/entries/Martingales.html, Formal proof development.