

# Pricing in discrete financial models

Mnacho Echenim

March 17, 2025

## Contents

<b>1</b>	<b>Generated subalgebras</b>	<b>1</b>
1.1	Independence between a random variable and a subalgebra. . .	3
<b>2</b>	<b>Filtrations</b>	<b>4</b>
2.1	Basic definitions . . . . .	4
2.2	Stochastic processes . . . . .	5
2.2.1	Adapted stochastic processes . . . . .	6
2.2.2	Predictable stochastic processes . . . . .	7
2.3	Initially trivial filtrations . . . . .	9
2.4	Filtration-equivalent measure spaces . . . . .	10
<b>3</b>	<b>Martingales</b>	<b>11</b>
<b>4</b>	<b>Discrete Conditional Expectation</b>	<b>12</b>
4.1	Preliminary measurability results . . . . .	12
4.2	Definition of explicit conditional expectation . . . . .	14
<b>5</b>	<b>Infinite coin toss space</b>	<b>19</b>
5.1	Preliminary results . . . . .	19
5.2	Bernoulli streams . . . . .	21
5.3	Natural filtration on the infinite coin toss space . . . . .	22
5.3.1	The projection function . . . . .	22
5.3.2	Natural filtration locale . . . . .	25
5.3.3	Probability component . . . . .	30
5.3.4	Filtration equivalence for the natural filtration . . . . .	33
5.3.5	More results on the projection function . . . . .	34
5.3.6	Integrals and conditional expectations on the natural filtration . . . . .	36
5.4	Images of stochastic processes by prefixes of streams . . . . .	39
5.4.1	Definitions . . . . .	39
5.4.2	Induced filtration, relationship with filtration gener- ated by underlying stochastic process . . . . .	41

<b>6</b>	<b>Geometric random walk</b>	<b>45</b>
<b>7</b>	<b>Fair Prices</b>	<b>46</b>
7.1	Preliminary results . . . . .	46
7.1.1	On the almost everywhere filter . . . . .	46
7.1.2	On conditional expectations . . . . .	47
7.2	Financial formalizations . . . . .	48
7.2.1	Markets . . . . .	48
7.2.2	Quantity processes and portfolios . . . . .	49
7.2.3	Trading strategies . . . . .	52
7.2.4	Self-financing portfolios . . . . .	54
7.2.5	Replicating portfolios . . . . .	60
7.2.6	Arbitrages . . . . .	60
7.2.7	Fair prices . . . . .	62
7.3	Risk-neutral probability space . . . . .	65
7.3.1	risk-free rate and discount factor processes . . . . .	65
7.3.2	Discounted value of a stochastic process . . . . .	67
7.3.3	Results on risk-neutral probability spaces . . . . .	69
<b>8</b>	<b>The Cox Ross Rubinstein model</b>	<b>71</b>
8.1	Preliminary results on the market . . . . .	71
8.2	Risk-neutral probability space for the geometric random walk . . . . .	74
8.3	Existence of a replicating portfolio . . . . .	78
<b>9</b>	<b>Effective computation definitions and results</b>	<b>86</b>
9.1	Generation of lists of boolean elements . . . . .	86
9.2	Probability components for lists . . . . .	87
9.3	Geometric process applied to lists . . . . .	87
9.4	Effective computation of discounted values . . . . .	88
<b>10</b>	<b>Pricing results on options</b>	<b>88</b>
10.1	Call option . . . . .	88
10.2	Put option . . . . .	89
10.3	Lookback option . . . . .	89
10.4	Asian option . . . . .	91

## 1 Generated subalgebras

This section contains definitions and properties related to generated subalgebras.

**theory** *Generated-Subalgebra* **imports** *HOL-Probability.Probability*

**begin**

**definition** *gen-subalgebra* **where**  
*gen-subalgebra*  $M\ G = \text{sigma}(\text{space } M)\ G$

**lemma** *gen-subalgebra-space*:  
**shows**  $\text{space}(\text{gen-subalgebra } M\ G) = \text{space } M$   
<proof>

**lemma** *gen-subalgebra-sets*:  
**assumes**  $G \subseteq \text{sets } M$   
**and**  $A \in G$   
**shows**  $A \in \text{sets}(\text{gen-subalgebra } M\ G)$   
<proof>

**lemma** *gen-subalgebra-sig-sets*:  
**assumes**  $G \subseteq \text{Pow}(\text{space } M)$   
**shows**  $\text{sets}(\text{gen-subalgebra } M\ G) = \text{sigma-sets}(\text{space } M)\ G$  <proof>

**lemma** *gen-subalgebra-sigma-sets*:  
**assumes**  $G \subseteq \text{sets } M$   
**and** *sigma-algebra*  $(\text{space } M)\ G$   
**shows**  $\text{sets}(\text{gen-subalgebra } M\ G) = G$   
<proof>

**lemma** *gen-subalgebra-is-subalgebra*:  
**assumes** *sub*:  $G \subseteq \text{sets } M$   
**and** *sigal*: *sigma-algebra*  $(\text{space } M)\ G$   
**shows** *subalgebra*  $M(\text{gen-subalgebra } M\ G)$  (**is** *subalgebra*  $M\ ?N$ )  
<proof>

**definition** *fct-gen-subalgebra* ::  $'a\ \text{measure} \Rightarrow 'b\ \text{measure} \Rightarrow ('a \Rightarrow 'b) \Rightarrow 'a\ \text{measure}$  **where**  
 $\text{fct-gen-subalgebra } M\ N\ X = \text{gen-subalgebra } M(\text{sigma-sets}(\text{space } M)\ \{X - 'B \cap (\text{space } M) \mid B. B \in \text{sets } N\})$

**lemma** *fct-gen-subalgebra-sets*:  
**shows**  $\text{sets}(\text{fct-gen-subalgebra } M\ N\ X) = \text{sigma-sets}(\text{space } M)\ \{X - 'B \cap \text{space } M \mid B. B \in \text{sets } N\}$   
<proof>

**lemma** *fct-gen-subalgebra-space*:

**shows**  $\text{space } (\text{fct-gen-subalgebra } M N X) = \text{space } M$   
*<proof>*

**lemma** *fct-gen-subalgebra-eq-sets:*

**assumes**  $\text{sets } M = \text{sets } P$

**shows**  $\text{fct-gen-subalgebra } M N X = \text{fct-gen-subalgebra } P N X$

*<proof>*

**lemma** *fct-gen-subalgebra-sets-mem:*

**assumes**  $B \in \text{sets } N$

**shows**  $X - ' B \cap (\text{space } M) \in \text{sets } (\text{fct-gen-subalgebra } M N X)$  *<proof>*

**lemma** *fct-gen-subalgebra-is-subalgebra:*

**assumes**  $X \in \text{measurable } M N$

**shows**  $\text{subalgebra } M (\text{fct-gen-subalgebra } M N X)$

*<proof>*

**lemma** *fct-gen-subalgebra-fct-measurable:*

**assumes**  $X \in \text{space } M \rightarrow \text{space } N$

**shows**  $X \in \text{measurable } (\text{fct-gen-subalgebra } M N X) N$

*<proof>*

**lemma** *fct-gen-subalgebra-min:*

**assumes**  $\text{subalgebra } M P$

**and**  $f \in \text{measurable } P N$

**shows**  $\text{subalgebra } P (\text{fct-gen-subalgebra } M N f)$

*<proof>*

**lemma** *fct-preimage-sigma-sets:*

**assumes**  $X \in \text{space } M \rightarrow \text{space } N$

**shows**  $\text{sigma-sets } (\text{space } M) \{X - ' B \cap \text{space } M \mid B. B \in \text{sets } N\} = \{X - ' B \cap \text{space } M \mid B. B \in \text{sets } N\}$  (**is** ?L = ?R)

*<proof>*

**lemma** *fct-gen-subalgebra-sigma-sets:*

**assumes**  $X \in \text{space } M \rightarrow \text{space } N$

**shows**  $\text{sets } (\text{fct-gen-subalgebra } M N X) = \{X - ' B \cap \text{space } M \mid B. B \in \text{sets } N\}$

*<proof>*

**lemma** *fct-gen-subalgebra-info:*

**assumes**  $f \in \text{space } M \rightarrow \text{space } N$

**and**  $x \in \text{space } M$

**and**  $w \in \text{space } M$

**and**  $f x = f w$

**shows**  $\bigwedge A. A \in \text{sets } (\text{fct-gen-subalgebra } M N f) \implies (x \in A) = (w \in A)$

*<proof>*

## 1.1 Independence between a random variable and a subalgebra.

**definition** (in *prob-space*) *subalgebra-indep-var* :: ('a  $\Rightarrow$  real)  $\Rightarrow$  'a measure  $\Rightarrow$  bool **where**

*subalgebra-indep-var* X N  $\longleftrightarrow$   
X  $\in$  borel-measurable M &  
(subalgebra M N) &  
(indep-set (sigma-sets (space M) { X - ' A  $\cap$  space M | A. A  $\in$  sets borel}))  
(sets N))

**lemma** (in *prob-space*) *indep-set-mono*:

**assumes** indep-set A B  
**assumes** A'  $\subseteq$  A  
**assumes** B'  $\subseteq$  B  
**shows** indep-set A' B'

*<proof>*

**lemma** (in *prob-space*) *subalgebra-indep-var-indicator*:

**fixes** X :: 'a  $\Rightarrow$  real  
**assumes** subalgebra-indep-var X N  
**and** X  $\in$  borel-measurable M  
**and** A  $\in$  sets N  
**shows** indep-var borel X borel (indicator A)

*<proof>*

**lemma** *fct-gen-subalgebra-cong*:

**assumes** space M = space P  
**and** sets N = sets Q  
**shows** fct-gen-subalgebra M N X = fct-gen-subalgebra P Q X

*<proof>*

**end**

## 2 Filtrations

This theory introduces basic notions about filtrations, which permit to define adaptable processes and predictable processes in the case where the filtration is indexed by natural numbers.

**theory** *Filtration* **imports** *HOL-Probability.Probability*  
**begin**

## 2.1 Basic definitions

```

class linorder-bot = linorder + bot
instantiation nat::linorder-bot
begin
instance ⟨proof⟩
end

```

```

definition filtration :: 'a measure ⇒ ('i::linorder-bot ⇒ 'a measure) ⇒ bool where
filtration M F ←→
  (∀ t. subalgebra M (F t)) ∧
  (∀ s t. s ≤ t → subalgebra (F t) (F s))

```

```

lemma filtrationI:
  assumes ∀ t. subalgebra M (F t)
  and ∀ s t. s ≤ t → subalgebra (F t) (F s)
shows filtration M F ⟨proof⟩

```

```

lemma filtrationE1:
  assumes filtration M F
  shows subalgebra M (F t) ⟨proof⟩

```

```

lemma filtrationE2:
  assumes filtration M F
  shows s ≤ t ⇒ subalgebra (F t) (F s) ⟨proof⟩

```

```

locale filtrated-prob-space = prob-space +
  fixes F
  assumes filtration: filtration M F

```

```

lemma (in filtrated-prob-space) filtration-space:
  assumes s ≤ t
  shows space (F s) = space (F t) ⟨proof⟩

```

```

lemma (in filtrated-prob-space) filtration-measurable:
  assumes f ∈ measurable (F t) N
shows f ∈ measurable M N ⟨proof⟩

```

```

lemma (in filtrated-prob-space) increasing-measurable-info:
  assumes f ∈ measurable (F s) N
  and s ≤ t
  shows f ∈ measurable (F t) N
  ⟨proof⟩

```

```

definition disc-filtr :: 'a measure ⇒ (nat ⇒ 'a measure) ⇒ bool where
disc-filtr M F ←→

```

$(\forall n. \text{subalgebra } M (F n)) \wedge$   
 $(\forall n m. n \leq m \longrightarrow \text{subalgebra } (F m) (F n))$

**locale** *disc-filtr-prob-space* = *prob-space* +  
**fixes** *F*  
**assumes** *discrete-filtration: disc-filtr M F*

**lemma** (**in** *disc-filtr-prob-space*) *subalgebra-filtration:*  
**assumes** *subalgebra N M*  
**and** *filtration M F*  
**shows** *filtration N F*  
 $\langle \text{proof} \rangle$

**sublocale** *disc-filtr-prob-space*  $\subseteq$  *filtrated-prob-space*  
 $\langle \text{proof} \rangle$

## 2.2 Stochastic processes

Stochastic processes are collections of measurable functions. Those of a particular interest when there is a filtration are the adapted stochastic processes.

**definition** *stoch-procs where*  
 $\text{stoch-procs } M N = \{X. \forall t. (X t) \in \text{measurable } M N\}$

### 2.2.1 Adapted stochastic processes

**definition** *adapt-stoch-proc where*  
 $(\text{adapt-stoch-proc } F X N) \longleftrightarrow (\forall t. (X t) \in \text{measurable } (F t) N)$

**abbreviation** *borel-adapt-stoch-proc F X*  $\equiv$  *adapt-stoch-proc F X borel*

**lemma** (**in** *filtrated-prob-space*) *adapted-is-dsp:*  
**assumes** *adapt-stoch-proc F X N*  
**shows**  $X \in \text{stoch-procs } M N$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *filtrated-prob-space*) *adapt-stoch-proc-borel-measurable:*  
**assumes** *adapt-stoch-proc F X N*  
**shows**  $\forall n. (X n) \in \text{measurable } M N$   
 $\langle \text{proof} \rangle$

**lemma** (**in** *filtrated-prob-space*) *borel-adapt-stoch-proc-borel-measurable:*  
**assumes** *borel-adapt-stoch-proc F X*  
**shows**  $\forall n. (X n) \in \text{borel-measurable } M$

*<proof>*

**lemma** (in *filtrated-prob-space*) *constant-process-borel-adapted*:  
  **shows** *borel-adapt-stoch-proc F* ( $\lambda n w. c$ )  
*<proof>*

**lemma** (in *filtrated-prob-space*) *borel-adapt-stoch-proc-add*:  
  **fixes**  $X::'b \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, topological-monoid-add}\})$   
  **assumes** *borel-adapt-stoch-proc F X*  
  **and** *borel-adapt-stoch-proc F Y*  
**shows** *borel-adapt-stoch-proc F* ( $\lambda t w. X t w + Y t w$ ) *<proof>*

**lemma** (in *filtrated-prob-space*) *borel-adapt-stoch-proc-sum*:  
  **fixes**  $A::'d \Rightarrow 'b \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, topological-comm-monoid-add}\})$   
  **assumes**  $\bigwedge i. i \in S \implies \text{borel-adapt-stoch-proc F } (A i)$   
**shows** *borel-adapt-stoch-proc F* ( $\lambda t w. (\sum i \in S. A i t w)$ ) *<proof>*

**lemma** (in *filtrated-prob-space*) *borel-adapt-stoch-proc-times*:  
  **fixes**  $X::'b \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, real-normed-algebra}\})$   
  **assumes** *borel-adapt-stoch-proc F X*  
  **and** *borel-adapt-stoch-proc F Y*  
**shows** *borel-adapt-stoch-proc F* ( $\lambda t w. X t w * Y t w$ ) *<proof>*

**lemma** (in *filtrated-prob-space*) *borel-adapt-stoch-proc-prod*:  
  **fixes**  $A::'d \Rightarrow 'b \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, real-normed-field}\})$   
  **assumes**  $\bigwedge i. i \in S \implies \text{borel-adapt-stoch-proc F } (A i)$   
**shows** *borel-adapt-stoch-proc F* ( $\lambda t w. (\prod i \in S. A i t w)$ ) *<proof>*

## 2.2.2 Predictable stochastic processes

**definition** *predict-stoch-proc where*

(*predict-stoch-proc F X N*)  $\longleftrightarrow (X \ 0 \in \text{measurable } (F \ 0) \ N \wedge (\forall n. (X \ (\text{Suc } n)) \in \text{measurable } (F \ n) \ N))$

**abbreviation** *borel-predict-stoch-proc F X*  $\equiv \text{predict-stoch-proc F X borel}$

**lemma** (in *disc-filtr-prob-space*) *predict-imp-adapt*:  
  **assumes** *predict-stoch-proc F X N*  
  **shows** *adapt-stoch-proc F X N* *<proof>*

**lemma** (in *disc-filtr-prob-space*) *predictable-is-dsp*:  
  **assumes** *predict-stoch-proc F X N*  
  **shows**  $X \in \text{stoch-procs } M \ N$   
*<proof>*



**lemma** (in *disc-filtr-prob-space*) *borel-predict-stoch-proc-borel-measurable*:  
**assumes** *borel-predict-stoch-proc F X*  
**shows**  $\forall n. (X\ n) \in \text{borel-measurable } M$  *<proof>*

**lemma** (in *disc-filtr-prob-space*) *constant-process-borel-predictable*:  
**shows** *borel-predict-stoch-proc F*  $(\lambda\ n\ w.\ c)$   
*<proof>*

**lemma** (in *disc-filtr-prob-space*) *borel-predict-stoch-proc-add*:  
**fixes**  $X::\text{nat} \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, topological-monoid-add}\})$   
**assumes** *borel-predict-stoch-proc F X*  
**and** *borel-predict-stoch-proc F Y*  
**shows** *borel-predict-stoch-proc F*  $(\lambda\ t\ w.\ X\ t\ w + Y\ t\ w)$  *<proof>*

**lemma** (in *disc-filtr-prob-space*) *borel-predict-stoch-proc-sum*:  
**fixes**  $A::'d \Rightarrow \text{nat} \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, topological-comm-monoid-add}\})$   
**assumes**  $\bigwedge i. i \in S \implies \text{borel-predict-stoch-proc } F\ (A\ i)$   
**shows** *borel-predict-stoch-proc F*  $(\lambda\ t\ w.\ (\sum\ i \in S. A\ i\ t\ w))$  *<proof>*

**lemma** (in *disc-filtr-prob-space*) *borel-predict-stoch-proc-times*:  
**fixes**  $X::\text{nat} \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, real-normed-algebra}\})$   
**assumes** *borel-predict-stoch-proc F X*  
**and** *borel-predict-stoch-proc F Y*  
**shows** *borel-predict-stoch-proc F*  $(\lambda\ t\ w.\ X\ t\ w * Y\ t\ w)$  *<proof>*

**lemma** (in *disc-filtr-prob-space*) *borel-predict-stoch-proc-prod*:  
**fixes**  $A::'d \Rightarrow \text{nat} \Rightarrow 'a \Rightarrow ('c::\{\text{second-countable-topology, real-normed-field}\})$   
**assumes**  $\bigwedge i. i \in S \implies \text{borel-predict-stoch-proc } F\ (A\ i)$   
**shows** *borel-predict-stoch-proc F*  $(\lambda\ t\ w.\ (\prod\ i \in S. A\ i\ t\ w))$  *<proof>*

**definition** (in *prob-space*) *constant-image where*  
*constant-image f = (if  $\exists c::'b::\{t2\text{-space}\}.$   $\forall x \in \text{space } M. f\ x = c$  then*  
*SOME c.  $\forall x \in \text{space } M. f\ x = c$  else undefined)*

**lemma** (in *prob-space*) *constant-imageI*:  
**assumes**  $\exists c::'b::\{t2\text{-space}\}.$   $\forall x \in \text{space } M. f\ x = c$   
**shows**  $\forall x \in \text{space } M. f\ x = (\text{constant-image } f)$   
*<proof>*

**lemma** (in *prob-space*) *constant-image-pos*:

**assumes**  $\forall x \in \text{space } M. (0::\text{real}) < f x$   
**and**  $\exists c::\text{real}. \forall x \in \text{space } M. f x = c$   
**shows**  $0 < (\text{constant-image } f)$   
 $\langle \text{proof} \rangle$

**definition open-except where**  
 $\text{open-except } x y = (\text{if } x = y \text{ then } \{\} \text{ else } \text{SOME } A. \text{open } A \wedge x \in A \wedge y \notin A)$

**lemma open-exceptI:**  
**assumes**  $(x::'b::\{t1\text{-space}\}) \neq y$   
**shows**  $\text{open } (\text{open-except } x y)$  **and**  $x \in \text{open-except } x y$  **and**  $y \notin \text{open-except } x y$   
 $\langle \text{proof} \rangle$

**lemma open-except-set:**  
**assumes**  $\text{finite } A$   
**and**  $(x::'b::\{t1\text{-space}\}) \notin A$   
**shows**  $\exists U. \text{open } U \wedge x \in U \wedge U \cap A = \{\}$   
 $\langle \text{proof} \rangle$

**definition open-exclude-set where**  
 $\text{open-exclude-set } x A = (\text{if } (\exists U. \text{open } U \wedge U \cap A = \{x\}) \text{ then } \text{SOME } U. \text{open } U \wedge U \cap A = \{x\} \text{ else } \{\})$

**lemma open-exclude-setI:**  
**assumes**  $\exists U. \text{open } U \wedge U \cap A = \{x\}$   
**shows**  $\text{open } (\text{open-exclude-set } x A)$  **and**  $(\text{open-exclude-set } x A) \cap A = \{x\}$   
 $\langle \text{proof} \rangle$

**lemma open-exclude-finite:**  
**assumes**  $\text{finite } A$   
**and**  $(x::'b::\{t1\text{-space}\}) \in A$   
**shows**  $\text{open-set: open } (\text{open-exclude-set } x A)$  **and**  $\text{inter-}x:(\text{open-exclude-set } x A) \cap A = \{x\}$   
 $\langle \text{proof} \rangle$

## 2.3 Initially trivial filtrations

Intuitively, these are filtrations that can be used to denote the fact that there is no information at the start.

**definition init-triv-filt::'a measure  $\Rightarrow$  ('i::linorder-bot  $\Rightarrow$  'a measure)  $\Rightarrow$  bool where**  
 $\text{init-triv-filt } M F \longleftrightarrow \text{filtration } M F \wedge \text{sets } (F \text{ bot}) = \{\{\}, \text{space } M\}$

**lemma triv-measurable-cst:**  
**fixes**  $f::'a \Rightarrow 'b::\{t2\text{-space}\}$   
**assumes**  $\text{space } N = \text{space } M$   
**and**  $\text{space } M \neq \{\}$   
**and**  $\text{sets } N = \{\{\}, \text{space } M\}$   
**and**  $f \in \text{measurable } N \text{ borel}$

**shows**  $\exists c::'b. \forall x \in \text{space } N. f x = c$   
 $\langle \text{proof} \rangle$

**locale** *trivial-init-filtrated-prob-space* = *prob-space* +  
**fixes** *F*  
**assumes** *info-filtration: init-triv-filt M F*

**sublocale** *trivial-init-filtrated-prob-space*  $\subseteq$  *filtrated-prob-space*  
 $\langle \text{proof} \rangle$

**locale** *triv-init-disc-filtr-prob-space* = *prob-space* +  
**fixes** *F*  
**assumes** *info-disc-filtr: disc-filtr M F  $\wedge$  sets (F bot) = {{}}, space M*

**sublocale** *triv-init-disc-filtr-prob-space*  $\subseteq$  *trivial-init-filtrated-prob-space*  
 $\langle \text{proof} \rangle$

**sublocale** *triv-init-disc-filtr-prob-space*  $\subseteq$  *disc-filtr-prob-space*  
 $\langle \text{proof} \rangle$

**lemma** (**in** *triv-init-disc-filtr-prob-space*) *adapted-init:*  
**assumes** *borel-adapt-stoch-proc F x*  
**shows**  $\exists c. \forall w \in \text{space } M. ((x \ 0 \ w)::\text{real}) = c$   
 $\langle \text{proof} \rangle$

## 2.4 Filtration-equivalent measure spaces

This is a relaxation of the notion of equivalent probability spaces, where equivalence is tested modulo a filtration. Equivalent measure spaces agree on events that have a zero probability of occurring; here, filtration-equivalent measure spaces agree on such events when they belong to the filtration under consideration.

**definition** *filt-equiv where*  
*filt-equiv F M N*  $\longleftrightarrow$  *sets M = sets N  $\wedge$  filtration M F  $\wedge$  ( $\forall t A. A \in \text{sets } (F t)$*   
 $\longrightarrow$  (*emeasure M A = 0*)  $\longleftrightarrow$  (*emeasure N A = 0*)

**lemma** *filt-equiv-space:*  
**assumes** *filt-equiv F M N*  
**shows** *space M = space N*  $\langle \text{proof} \rangle$

**lemma** *filt-equiv-sets:*  
**assumes** *filt-equiv F M N*  
**shows** *sets M = sets N*  $\langle \text{proof} \rangle$

**lemma** *filt-equiv-filtration*:  
**assumes** *filt-equiv F M N*  
**shows** *filtration N F*  $\langle$ *proof* $\rangle$

**lemma** (**in** *filtrated-prob-space*) *AE-borel-eq*:  
**fixes**  $f::'a\Rightarrow\text{real}$   
**assumes**  $f\in\text{borel-measurable } (F\ t)$   
**and**  $g\in\text{borel-measurable } (F\ t)$   
**and** *AE w in M. f w = g w*  
**shows**  $\{w\in\text{space } M. f\ w \neq g\ w\} \in \text{sets } (F\ t) \wedge \text{emeasure } M\ \{w\in\text{space } M. f\ w \neq g\ w\} = 0$   
 $\langle$ *proof* $\rangle$

**lemma** (**in** *prob-space*) *filt-equiv-borel-AE-eq*:  
**fixes**  $f::'a\Rightarrow\text{real}$   
**assumes** *filt-equiv F M N*  
**and**  $f\in\text{borel-measurable } (F\ t)$   
**and**  $g\in\text{borel-measurable } (F\ t)$   
**and** *AE w in M. f w = g w*  
**shows** *AE w in N. f w = g w*  
 $\langle$ *proof* $\rangle$

**lemma** *filt-equiv-prob-space-subalgebra*:  
**assumes** *prob-space N*  
**and** *filt-equiv F M N*  
**and** *sigma-finite-subalgebra M G*  
**shows** *sigma-finite-subalgebra N G*  $\langle$ *proof* $\rangle$

**lemma** *filt-equiv-measurable*:  
**assumes** *filt-equiv F M N*  
**and**  $f\in\text{measurable } M\ P$   
**shows**  $f\in\text{measurable } N\ P$   $\langle$ *proof* $\rangle$

**lemma** *filt-equiv-imp-subalgebra*:  
**assumes** *filt-equiv F M N*  
**shows** *subalgebra N M*  $\langle$ *proof* $\rangle$

**end**

### 3 Martingales

**theory** *Martingale* **imports** *Filtration*  
**begin**

**definition** *martingale* **where**

*martingale*  $M F X \longleftrightarrow$   
 $(\text{filtration } M F) \wedge (\forall t. \text{integrable } M (X t)) \wedge (\text{borel-adapt-stoch-proc } F X) \wedge$   
 $(\forall t s. t \leq s \longrightarrow (\text{AE } w \text{ in } M. \text{real-cond-exp } M (F t) (X s) w = X t w))$

**lemma** *martingaleAE*:

**assumes** *martingale*  $M F X$   
**and**  $t \leq s$   
**shows**  $\text{AE } w \text{ in } M. \text{real-cond-exp } M (F t) (X s) w = (X t) w$  *<proof>*

**lemma** *martingale-add*:

**assumes** *martingale*  $M F X$   
**and** *martingale*  $M F Y$   
**and**  $\forall m. \text{sigma-finite-subalgebra } M (F m)$   
**shows** *martingale*  $M F (\lambda n w. X n w + Y n w)$  *<proof>*

**lemma** *disc-martingale-charact*:

**assumes**  $(\forall n. \text{integrable } M (X n))$   
**and** *filtration*  $M F$   
**and**  $\forall m. \text{sigma-finite-subalgebra } M (F m)$   
**and**  $\forall m. X m \in \text{borel-measurable } (F m)$   
**and**  $(\forall n. \text{AE } w \text{ in } M. \text{real-cond-exp } M (F n) (X (\text{Suc } n)) w = (X n) w)$   
**shows** *martingale*  $M F X$  *<proof>*

**lemma** *(in finite-measure) constant-martingale*:

**assumes**  $\forall t. \text{sigma-finite-subalgebra } M (F t)$   
**and** *filtration*  $M F$   
**shows** *martingale*  $M F (\lambda n w. c)$  *<proof>*

**end**

### 4 Discrete Conditional Expectation

**theory** *Disc-Cond-Expect* **imports** *HOL-Probability.Probability Generated-Subalgebra*  
**begin**

## 4.1 Preliminary measurability results

These are some useful results, in particular when working with functions that have a countable codomain.

**definition** *disc-fct* **where**

*disc-fct*  $f \equiv \text{countable } (\text{range } f)$

**definition** *point-measurable* **where**

*point-measurable*  $M S f \equiv (f^{-1}(\text{space } M) \subseteq S) \wedge (\forall r \in (\text{range } f) \cap S . f^{-1}\{r\} \cap (\text{space } M) \in \text{sets } M)$

**lemma** *singl-meas-if*:

**assumes**  $f \in \text{space } M \rightarrow \text{space } N$

**and**  $\forall r \in \text{range } f \cap \text{space } N . \exists A \in \text{sets } N . \text{range } f \cap A = \{r\}$

**shows** *point-measurable* (*fct-gen-subalgebra*  $M N f$ ) (*space*  $N$ )  $f$  *<proof>*

**lemma** *meas-single-meas*:

**assumes**  $f \in \text{measurable } M N$

**and**  $\forall r \in \text{range } f \cap \text{space } N . \exists A \in \text{sets } N . \text{range } f \cap A = \{r\}$

**shows** *point-measurable*  $M$  (*space*  $N$ )  $f$  *<proof>*

**definition** *countable-preimages* **where**

*countable-preimages*  $B Y = (\lambda n . \text{if } ((\text{infinite } B) \vee (\text{finite } B \wedge n < \text{card } B)) \text{ then } Y - \{(from\text{-nat-into } B) n\} \text{ else } \{\})$

**lemma** *count-pre-disj*:

**fixes**  $i :: \text{nat}$

**assumes** *countable*  $B$

**and**  $i \neq j$

**shows** (*countable-preimages*  $B Y$ )  $i \cap$  (*countable-preimages*  $B Y$ )  $j = \{\}$  *<proof>*

**lemma** *count-pre-surj*:

**assumes** *countable*  $B$

**and**  $w \in Y - \{B\}$

**shows**  $\exists i . w \in$  (*countable-preimages*  $B Y$ )  $i$  *<proof>*

**lemma** *count-pre-img*:

**assumes**  $x \in$  (*countable-preimages*  $B Y$ )  $n$

**shows**  $Y x =$  (*from-nat-into*  $B$ )  $n$

*<proof>*

**lemma** *count-pre-union-img*:  
**assumes** *countable B*  
**shows**  $Y - 'B = (\bigcup i. (\text{countable-preimages } B \ Y) \ i)$   
 $\langle \text{proof} \rangle$

**lemma** *count-pre-meas*:  
**assumes** *point-measurable M (space N) Y*  
**and**  $B \subseteq \text{space } N$   
**and** *countable B*  
**shows**  $\forall i. (\text{countable-preimages } B \ Y) \ i \cap \text{space } M \in \text{sets } M$   
 $\langle \text{proof} \rangle$

**lemma** *disct-fct-point-measurable*:  
**assumes** *disc-fct f*  
**and** *point-measurable M (space N) f*  
**shows**  $f \in \text{measurable } M \ N \ \langle \text{proof} \rangle$

**lemma** *set-point-measurable*:  
**assumes** *point-measurable M (space N) Y*  
**and**  $B \subseteq \text{space } N$   
**and** *countable B*  
**shows**  $(Y - 'B) \cap \text{space } M \in \text{sets } M$   
 $\langle \text{proof} \rangle$

## 4.2 Definition of explicit conditional expectation

This section is devoted to an explicit computation of a conditional expectation for random variables that have a countable codomain. More precisely, the computed random variable is almost everywhere equal to a conditional expectation of the random variable under consideration.

**definition** *img-dce where*  
 $\text{img-dce } M \ Y \ X = (\lambda y. \text{if } \text{measure } M ((Y - ' \{y\}) \cap \text{space } M) = 0 \text{ then } 0 \text{ else } ((\text{integral}^L M (\lambda w. ((X \ w) * (\text{indicator } ((Y - ' \{y\}) \cap \text{space } M) \ w)))) / (\text{measure } M ((Y - ' \{y\}) \cap \text{space } M))))$

**definition** *expl-cond-expect where*  
 $\text{expl-cond-expect } M \ Y \ X = (\text{img-dce } M \ Y \ X) \circ Y$

**lemma** *nn-expl-cond-expect-pos*:  
**assumes**  $\forall w \in \text{space } M. 0 \leq X \ w$   
**shows**  $\forall w \in \text{space } M. 0 \leq (\text{expl-cond-expect } M \ Y \ X) \ w$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-expect-const*:

**assumes**  $Y w = Y y$   
**shows**  $\text{expl-cond-expect } M Y X w = \text{expl-cond-expect } M Y X y$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-exp-cong*:  
**assumes**  $\forall w \in \text{space } M. X w = Z w$   
**shows**  $\forall w \in \text{space } M. \text{expl-cond-expect } M Y X w = \text{expl-cond-expect } M Y Z w$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-exp-add*:  
**assumes** *integrable*  $M X$   
**and** *integrable*  $M Z$   
**shows**  $\forall w \in \text{space } M. \text{expl-cond-expect } M Y (\lambda x. X x + Z x) w = \text{expl-cond-expect } M Y X w + \text{expl-cond-expect } M Y Z w$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-exp-diff*:  
**assumes** *integrable*  $M X$   
**and** *integrable*  $M Z$   
**shows**  $\forall w \in \text{space } M. \text{expl-cond-expect } M Y (\lambda x. X x - Z x) w = \text{expl-cond-expect } M Y X w - \text{expl-cond-expect } M Y Z w$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-expect-prop-sets*:  
**assumes** *disc-fct*  $Y$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**and**  $D = \{w \in \text{space } M. Y w \in \text{space } N \wedge (P (\text{expl-cond-expect } M Y X w))\}$   
**shows**  $D \in \text{sets } M$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-expect-prop-sets2*:  
**assumes** *disc-fct*  $Y$   
**and** *point-measurable*  $(\text{fct-gen-subalgebra } M N Y) (\text{space } N) Y$   
**and**  $D = \{w \in \text{space } M. Y w \in \text{space } N \wedge (P (\text{expl-cond-expect } M Y X w))\}$   
**shows**  $D \in \text{sets } (\text{fct-gen-subalgebra } M N Y)$   
 $\langle \text{proof} \rangle$

**lemma** *expl-cond-expect-disc-fct*:  
**assumes** *disc-fct*  $Y$   
**shows** *disc-fct*  $(\text{expl-cond-expect } M Y X)$   
 $\langle \text{proof} \rangle$



**lemma** *expl-cond-expect-point-meas*:  
**assumes** *disc-fct*  $Y$   
**and** *point-measurable*  $M$  (*space*  $N$ )  $Y$   
**shows** *point-measurable*  $M$  *UNIV* (*expl-cond-expect*  $M$   $Y$   $X$ )  
 $\langle$ *proof* $\rangle$

**lemma** *expl-cond-expect-borel-measurable*:  
**assumes** *disc-fct*  $Y$   
**and** *point-measurable*  $M$  (*space*  $N$ )  $Y$   
**shows** (*expl-cond-expect*  $M$   $Y$   $X$ )  $\in$  *borel-measurable*  $M$   $\langle$ *proof* $\rangle$

**lemma** *expl-cond-exp-borel*:  
**assumes**  $Y \in$  *space*  $M \rightarrow$  *space*  $N$   
**and** *disc-fct*  $Y$   
**and**  $\forall r \in$  *range*  $Y \cap$  *space*  $N$ .  $\exists A \in$  *sets*  $N$ . *range*  $Y \cap A = \{r\}$   
**shows** (*expl-cond-expect*  $M$   $Y$   $X$ )  $\in$  *borel-measurable* (*fct-gen-subalgebra*  $M$   $N$   $Y$ )  
 $\langle$ *proof* $\rangle$

**lemma** *expl-cond-expect-indic-borel-measurable*:  
**assumes** *disc-fct*  $Y$   
**and** *point-measurable*  $M$  (*space*  $N$ )  $Y$   
**and**  $B \subseteq$  *space*  $N$   
**and** *countable*  $B$   
**shows**  $(\lambda w$ . *expl-cond-expect*  $M$   $Y$   $X$   $w *$  *indicator* (*countable-preimages*  $B$   $Y$   $n$   
 $\cap$  *space*  $M$ )  $w) \in$  *borel-measurable*  $M$   
 $\langle$ *proof* $\rangle$

**lemma** (*in finite-measure*) *dce-prod*:  
**assumes** *point-measurable*  $M$  (*space*  $N$ )  $Y$   
**and** *integrable*  $M$   $X$   
**and**  $\forall w \in$  *space*  $M$ .  $0 \leq X$   $w$   
**shows**  $\forall w$ .  $(Y$   $w) \in$  *space*  $N \rightarrow$  (*expl-cond-expect*  $M$   $Y$   $X$ )  $w *$  *measure*  $M$  ( $(Y$   
 $- \{Y$   $w\}) \cap$  *space*  $M$ ) = *integral* <sup>$L$</sup>   $M$   $(\lambda y$ .  $(X$   $y) *$  (*indicator*  $((Y$   $- \{Y$   $w\}) \cap$  *space*  
 $M)$   $y))$   
 $\langle$ *proof* $\rangle$

**lemma** *expl-cond-expect-const-exp:*

**shows**  $\text{integral}^L M (\lambda y. \text{expl-cond-expect } M Y X w * (\text{indicator } (Y - \{Y w\} \cap \text{space } M)) y) =$   
 $\text{integral}^L M (\lambda y. \text{expl-cond-expect } M Y X y * (\text{indicator } (Y - \{Y w\} \cap \text{space } M)) y)$   
(*proof*)

**lemma** *nn-expl-cond-expect-const-exp:*

**assumes**  $\forall w \in \text{space } M. 0 \leq X w$   
**shows**  $\text{integral}^N M (\lambda y. \text{expl-cond-expect } M Y X w * (\text{indicator } (Y - \{Y w\} \cap \text{space } M)) y) =$   
 $\text{integral}^N M (\lambda y. \text{expl-cond-expect } M Y X y * (\text{indicator } (Y - \{Y w\} \cap \text{space } M)) y)$   
(*proof*)

**lemma** (*in finite-measure*) *nn-expl-cond-bounded:*

**assumes**  $\forall w \in \text{space } M. 0 \leq X w$   
**and** *integrable*  $M X$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**and**  $w \in \text{space } M$   
**and**  $Y w \in \text{space } N$   
**shows**  $\text{integral}^N M (\lambda y. \text{expl-cond-expect } M Y X y * (\text{indicator } (Y - \{Y w\} \cap \text{space } M)) y) < \infty$   
(*proof*)

**lemma** (*in finite-measure*) *count-prod:*

**fixes**  $Y :: 'a \Rightarrow 'b$   
**assumes**  $B \subseteq \text{space } N$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**and** *integrable*  $M X$   
**and**  $\forall w \in \text{space } M. 0 \leq X w$   
**shows**  $\forall i. \text{integral}^L M (\lambda y. (X y) * (\text{indicator } (\text{countable-preimages } B Y i \cap \text{space } M)) y) =$   
 $\text{integral}^L M (\lambda y. (\text{expl-cond-expect } M Y X y) * (\text{indicator } (\text{countable-preimages } B Y i \cap \text{space } M)) y)$   
(*proof*)

**lemma** (*in finite-measure*) *count-pre-integrable:*

**assumes** *point-measurable*  $M (\text{space } N) Y$   
**and** *disc-fct*  $Y$   
**and**  $B \subseteq \text{space } N$   
**and** *countable*  $B$   
**and** *integrable*  $M X$

**and**  $\forall w \in \text{space } M. 0 \leq X w$   
**shows** *integrable*  $M (\lambda w. \text{expl-cond-expect } M Y X w * \text{indicator } (\text{countable-preimages } B Y n \cap \text{space } M) w)$   
 ⟨*proof*⟩

**lemma** (in *finite-measure*) *nn-cond-expl-is-cond-exp-tmp*:  
**assumes**  $\forall w \in \text{space } M. 0 \leq X w$   
**and** *integrable*  $M X$   
**and** *disc-fct*  $Y$   
**and** *point-measurable*  $M (\text{space } M') Y$   
**shows**  $\forall A \in \text{sets } M'. \text{integrable } M (\lambda w. ((\text{expl-cond-expect } M Y X) w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)) w)) \wedge$   
 $\text{integral}^L M (\lambda w. (X w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)) w)) =$   
 $\text{integral}^L M (\lambda w. ((\text{expl-cond-expect } M Y X) w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)))) w)$   
 ⟨*proof*⟩

**lemma** (in *finite-measure*) *nn-expl-cond-exp-integrable*:  
**assumes**  $\forall w \in \text{space } M. 0 \leq X w$   
**and** *integrable*  $M X$   
**and** *disc-fct*  $Y$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**shows** *integrable*  $M (\text{expl-cond-expect } M Y X)$   
 ⟨*proof*⟩

**lemma** (in *finite-measure*) *nn-cond-expl-is-cond-exp*:  
**assumes**  $\forall w \in \text{space } M. 0 \leq X w$   
**and** *integrable*  $M X$   
**and** *disc-fct*  $Y$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**shows**  $\forall A \in \text{sets } N. \text{integral}^L M (\lambda w. (X w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)) w)) =$   
 $\text{integral}^L M (\lambda w. ((\text{expl-cond-expect } M Y X) w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)))) w)$   
 ⟨*proof*⟩

**lemma** (in *finite-measure*) *expl-cond-exp-integrable*:  
**assumes** *integrable*  $M X$   
**and** *disc-fct*  $Y$   
**and** *point-measurable*  $M (\text{space } N) Y$   
**shows** *integrable*  $M (\text{expl-cond-expect } M Y X)$   
 ⟨*proof*⟩

**lemma** (in *finite-measure*) *is-cond-exp*:  
**assumes** *integrable M X*  
**and** *disc-fct Y*  
**and** *point-measurable M (space N) Y*  
**shows**  $\forall A \in \text{sets } N. \text{integral}^L M (\lambda w. (X w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)) w)) =$   
 $\text{integral}^L M (\lambda w. ((\text{expl-cond-expect } M Y X) w) * (\text{indicator } ((Y - 'A) \cap (\text{space } M)))) w)$   
*<proof>*

**lemma** (in *finite-measure*) *charact-cond-exp*:  
**assumes** *disc-fct Y*  
**and** *integrable M X*  
**and** *point-measurable M (space N) Y*  
**and**  $Y \in \text{space } M \rightarrow \text{space } N$   
**and**  $\forall r \in \text{range } Y \cap \text{space } N. \exists A \in \text{sets } N. \text{range } Y \cap A = \{r\}$   
**shows**  $AE w \text{ in } M. \text{real-cond-exp } M (\text{fct-gen-subalgebra } M N Y) X w = \text{expl-cond-expect } M Y X w$   
*<proof>*

**lemma** (in *finite-measure*) *charact-cond-exp'*:  
**assumes** *disc-fct Y*  
**and** *integrable M X*  
**and**  $Y \in \text{measurable } M N$   
**and**  $\forall r \in \text{range } Y \cap \text{space } N. \exists A \in \text{sets } N. \text{range } Y \cap A = \{r\}$   
**shows**  $AE w \text{ in } M. \text{real-cond-exp } M (\text{fct-gen-subalgebra } M N Y) X w = \text{expl-cond-expect } M Y X w$   
*<proof>*

**end**

## 5 Infinite coin toss space

This section contains the formalization of the infinite coin toss space, i.e., the probability space constructed on infinite sequences of independent coin tosses.

**theory** *Infinite-Coin-Toss-Space* **imports** *Filtration Generated-Subalgebra Disc-Cond-Expect*

**begin**

### 5.1 Preliminary results

**lemma** *decompose-init-prod*:

**fixes**  $n::nat$   
**shows**  $(\prod i \in \{0..n\}. f i) = f 0 * (\prod i \in \{1..n\}. f i)$   
 $\langle proof \rangle$

**lemma** *Inter-nonempty-distrib*:  
**assumes**  $A \neq \{\}$   
**shows**  $\bigcap A \cap B = (\bigcap C \in A. (C \cap B))$   
 $\langle proof \rangle$

**lemma** *enn2real-sum*: **shows**  $finite A \implies (\bigwedge a. a \in A \implies f a < top) \implies enn2real (sum f A) = (\sum a \in A. enn2real (f a))$   
 $\langle proof \rangle$

**lemma** *ennreal-sum*: **shows**  $finite A \implies (\bigwedge a. 0 \leq f a) \implies (\sum a \in A. ennreal (f a)) = ennreal (\sum a \in A. f a)$   
 $\langle proof \rangle$

**lemma** *stake-snth*:  
**assumes**  $stake n w = stake n x$   
**shows**  $Suc i \leq n \implies snth w i = snth x i$   
 $\langle proof \rangle$

**lemma** *stake-snth-charact*:  
**assumes**  $stake n w = stake n x$   
**shows**  $\forall i < n. snth w i = snth x i$   
 $\langle proof \rangle$

**lemma** *stake-snth'*:  
**shows**  $(\bigwedge i. Suc i \leq n \implies snth w i = snth x i) \implies stake n w = stake n x$   
 $\langle proof \rangle$

**lemma** *stake-inter-snth*:  
**fixes**  $x$   
**assumes**  $Suc 0 \leq n$   
**shows**  $\{w \in space M. (stake n w = stake n x)\} = (\bigcap i \in \{0.. n-1\}. \{w \in space M. (snth w i = snth x i)\})$   
 $\langle proof \rangle$

**lemma** *streams-stake-set*:  
**shows**  $pw \in streams A \implies set (stake n pw) \subseteq A$   
 $\langle proof \rangle$

**lemma** *stake-finite-universe-induct*:  
**assumes**  $finite A$

**and**  $A \neq \{\}$   
**shows**  $(\text{stake } (\text{Suc } n) \text{ `streams } A)) = \{s\#w \mid s \text{ w. } s \in A \wedge w \in (\text{stake } n \text{ `streams } A))\}$  **(is ?L = ?R)**  
 <proof>

**lemma** *stake-finite-universe-finite*:  
**assumes** *finite A*  
**and**  $A \neq \{\}$   
**shows** *finite (stake n `streams A)*  
 <proof>

**lemma** *diff-streams-only-if*:  
**assumes**  $w \neq x$   
**shows**  $\exists n. \text{snth } w \ n \neq \text{snth } x \ n$   
 <proof>

**lemma** *diff-streams-if*:  
**assumes**  $\exists n. \text{snth } w \ n \neq \text{snth } x \ n$   
**shows**  $w \neq x$   
 <proof>

**lemma** *sigma-set-union-count*:  
**assumes**  $\forall y \in A. B \ y \in \text{sigma-sets } X \ Y$   
**and** *countable A*  
**shows**  $(\bigcup_{y \in A} B \ y) \in \text{sigma-sets } X \ Y$   
 <proof>

**lemma** *sigma-set-inter-init*:  
**assumes**  $\bigwedge i. i \leq (n::\text{nat}) \implies A \ i \in \text{sigma-sets } sp \ B$   
**and**  $B \subseteq \text{Pow } sp$   
**shows**  $(\bigcap_{i \in \{m..n\}} A \ i) \in \text{sigma-sets } sp \ B$   
 <proof>

**lemma** *adapt-sigma-sets*:  
**assumes**  $\bigwedge i. i \leq n \implies (X \ i) \in \text{measurable } M \ N$   
**shows** *sigma-algebra (space M) (sigma-sets (space M) ( $\bigcup_{i \in \{m..n\}} \{X \ i - `$*   
 $A \cap \text{space } M \ |A. A \in \text{sets } N\}))$   
 <proof>

## 5.2 Bernoulli streams

Bernoulli streams represent the formal definition of the infinite coin toss space. The parameter  $p$  represents the probability of obtaining a head after a coin toss.

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

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

**lemma** *bernoulli-stream-space:*  
assumes  $N = \text{bernoulli-stream } p$   
shows  $\text{space } N = \text{streams UNIV}::\text{bool}$   
*<proof>*

**lemma** *bernoulli-stream-preimage:*  
assumes  $N = \text{bernoulli-stream } p$   
shows  $f^{-1} A \cap (\text{space } N) = f^{-1} A$   
*<proof>*

**lemma** *bernoulli-stream-component-probability:*  
assumes  $N = \text{bernoulli-stream } p$  and  $0 \leq p$  and  $p \leq 1$   
shows  $\forall n. \text{emeasure } N \{w \in \text{space } N. (\text{snth } w \ n)\} = p$   
*<proof>*

**lemma** *bernoulli-stream-component-probability-compl:*  
assumes  $N = \text{bernoulli-stream } p$  and  $0 \leq p$  and  $p \leq 1$   
shows  $\forall n. \text{emeasure } N \{w \in \text{space } N. \neg(\text{snth } w \ n)\} = 1 - p$   
*<proof>*

**lemma** *bernoulli-stream-sets:*  
assumes  $0 < q$   
and  $q < 1$   
and  $0 < p$   
and  $p < 1$   
shows  $\text{sets } (\text{bernoulli-stream } p) = \text{sets } (\text{bernoulli-stream } q)$  *<proof>*

**locale** *infinite-coin-toss-space =*  
fixes  $p::\text{real}$  and  $M::\text{bool}$  stream measure  
assumes  $p\text{-gt-0}: 0 \leq p$   
and  $p\text{-lt-1}: p \leq 1$   
and *bernoulli:*  $M = \text{bernoulli-stream } p$

**sublocale** *infinite-coin-toss-space*  $\subseteq$  *prob-space*  
*<proof>*

### 5.3 Natural filtration on the infinite coin toss space

The natural filtration on the infinite coin toss space is the discrete filtration  $F$  such that  $F \ n$  represents the restricted measure space in which the outcome of the first  $n$  coin tosses is known.

### 5.3.1 The projection function

Intuitively, the restricted measure space in which the outcome of the first  $n$  coin tosses is known can be defined by any measurable function that maps all infinite sequences that agree on the first  $n$  coin tosses to the same element.

**definition** (in *infinite-coin-toss-space*) *pseudo-proj-True*::  $\text{nat} \Rightarrow \text{bool stream} \Rightarrow \text{bool stream}$  **where**

$$\text{pseudo-proj-True } n = (\lambda w. \text{shift } (\text{stake } n \ w) \ (\text{sconst } \text{True}))$$

**definition** (in *infinite-coin-toss-space*) *pseudo-proj-False*::  $\text{nat} \Rightarrow \text{bool stream} \Rightarrow \text{bool stream}$  **where**

$$\text{pseudo-proj-False } n = (\lambda w. \text{shift } (\text{append } (\text{stake } n \ w) \ [\text{False}]) \ (\text{sconst } \text{True}))$$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-neq-True*:

**shows**  $\text{pseudo-proj-False } n \ w \neq \text{pseudo-proj-True } n \ w$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-measurable*:

**shows**  $\text{pseudo-proj-False } n \in \text{measurable } (\text{bernoulli-stream } p) \ (\text{bernoulli-stream } p)$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-stake*:

**shows**  $\text{stake } n \ (\text{pseudo-proj-True } n \ w) = \text{stake } n \ w \ \langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-stake*:

**shows**  $\text{stake } n \ (\text{pseudo-proj-False } n \ w) = \text{stake } n \ w \ \langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-stake-image*:

**assumes**  $\text{stake } n \ w = \text{stake } n \ x$

**shows**  $\text{pseudo-proj-True } n \ w = \text{pseudo-proj-True } n \ x \ \langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-prefix*:

**assumes**  $n \leq m$

**and**  $\text{pseudo-proj-True } m \ x = \text{pseudo-proj-True } m \ y$

**shows**  $\text{pseudo-proj-True } n \ x = \text{pseudo-proj-True } n \ y$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-measurable*:

**shows**  $\text{pseudo-proj-True } n \in \text{measurable } (\text{bernoulli-stream } p) \ (\text{bernoulli-stream } p)$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-finite-image*:

**shows**  $\text{finite } (\text{range } (\text{pseudo-proj-True } n))$

$\langle \text{proof} \rangle$



**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-finite-image*:

**shows** *finite* (*range* (*pseudo-proj-False* *n*))

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-proj*:

**shows** *pseudo-proj-True* *n* (*pseudo-proj-True* *n* *w*) = *pseudo-proj-True* *n* *w*

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-Suc-False-proj*:

**shows** *pseudo-proj-True* (*Suc* *n*) (*pseudo-proj-False* *n* *w*) = *pseudo-proj-False* *n*

*w*

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-Suc-proj*:

**shows** *pseudo-proj-True* (*Suc* *n*) (*pseudo-proj-True* *n* *w*) = *pseudo-proj-True* *n*

*w*

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-proj-Suc*:

**shows** *pseudo-proj-True* *n* (*pseudo-proj-True* (*Suc* *n*) *w*) = *pseudo-proj-True* *n*

*w*

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-shift*:

**shows** *length* *l* = *n*  $\implies$  *pseudo-proj-True* *n* (*shift* *l* (*sconst* *True*)) = *shift* *l* (*sconst* *True*)

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-suc-img*:

**shows** *pseudo-proj-True* (*Suc* *n*) *w*  $\in$  {*pseudo-proj-True* *n* *w*, *pseudo-proj-False* *n* *w*}

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *measurable-snth-count-space*:

**shows** ( $\lambda w. \text{snth } w \ n$ )  $\in$  *measurable* (*bernoulli-stream* *p*) (*count-space* (*UNIV::bool set*))

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-same-img*:

**assumes** *pseudo-proj-True* *n* *w* = *pseudo-proj-True* *n* *x*

**shows**  $stake\ n\ w = stake\ n\ x$   $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-snth*:  
**assumes**  $pseudo\text{-}proj\text{-}True\ n\ x = pseudo\text{-}proj\text{-}True\ n\ w$   
**shows**  $\bigwedge i. Suc\ i \leq n \implies snth\ x\ i = snth\ w\ i$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-snth'*:  
**assumes**  $(\bigwedge i. Suc\ i \leq n \implies snth\ w\ i = snth\ x\ i)$   
**shows**  $pseudo\text{-}proj\text{-}True\ n\ w = pseudo\text{-}proj\text{-}True\ n\ x$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-preimage*:  
**assumes**  $w = pseudo\text{-}proj\text{-}True\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}True\ n) - \{w\} = (\bigcap_{i \in \{m. Suc\ m \leq n\}} (\lambda w. snth\ w\ i) - \{snth\ w\ i\})$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-preimage*:  
**assumes**  $w = pseudo\text{-}proj\text{-}False\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}False\ n) - \{w\} = (\bigcap_{i \in \{m. Suc\ m \leq n\}} (\lambda w. snth\ w\ i) - \{snth\ w\ i\})$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-preimage-stake*:  
**assumes**  $w = pseudo\text{-}proj\text{-}True\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}True\ n) - \{w\} = \{x. stake\ n\ x = stake\ n\ w\}$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-preimage-stake*:  
**assumes**  $w = pseudo\text{-}proj\text{-}False\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}False\ n) - \{w\} = \{x. stake\ n\ x = stake\ n\ w\}$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-preimage-stake-space*:  
**assumes**  $w = pseudo\text{-}proj\text{-}True\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}True\ n) - \{w\} \cap space\ M = \{x \in space\ M. stake\ n\ x = stake\ n\ w\}$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-singleton*:  
**assumes**  $w = pseudo\text{-}proj\text{-}True\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}True\ n) - \{w\} \cap (space\ (bernoulli\text{-}stream\ p)) \in sets\ (bernoulli\text{-}stream\ p)$

$p$ )  
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-False-singleton*:  
**assumes**  $w = pseudo\text{-}proj\text{-}False\ n\ w$   
**shows**  $(pseudo\text{-}proj\text{-}False\ n) - \{w\} \cap (space\ (bernoulli\text{-}stream\ p)) \in sets\ (bernoulli\text{-}stream\ p)$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-inverse-induct*:  
**assumes**  $w \in range\ (pseudo\text{-}proj\text{-}True\ n)$   
**shows**  $(pseudo\text{-}proj\text{-}True\ n) - \{w\} =$   
 $(pseudo\text{-}proj\text{-}True\ (Suc\ n)) - \{w\} \cup (pseudo\text{-}proj\text{-}True\ (Suc\ n)) - \{pseudo\text{-}proj\text{-}False\ n\ w\}$   
 $\langle proof \rangle$

### 5.3.2 Natural filtration locale

This part is mainly devoted to the proof that the projection function defined above indeed permits to obtain a filtration on the infinite coin toss space, and that this filtration is initially trivial.

**definition** (in *infinite-coin-toss-space*) *nat-filtration::nat  $\Rightarrow$  bool stream measure*  
**where**

$$nat\text{-}filtration\ n = fct\text{-}gen\text{-}subalgebra\ M\ M\ (pseudo\text{-}proj\text{-}True\ n)$$

**locale** *infinite-cts-filtration* = *infinite-coin-toss-space* +  
**fixes**  $F$   
**assumes** *natural-filtration:  $F = nat\text{-}filtration$*

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-space*:  
**shows**  $space\ (nat\text{-}filtration\ n) = UNIV$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-sets*:  
**shows**  $sets\ (nat\text{-}filtration\ n) =$   
 $sigma\text{-}sets\ (space\ (bernoulli\text{-}stream\ p))$   
 $\{pseudo\text{-}proj\text{-}True\ n - \{B \cap space\ M \mid B. B \in sets\ (bernoulli\text{-}stream\ p)\}\}$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-singleton*:  
**assumes**  $pseudo\text{-}proj\text{-}True\ n\ w = w$   
**shows**  $pseudo\text{-}proj\text{-}True\ n - \{w\} \in sets\ (nat\text{-}filtration\ n)$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-pseudo-proj-True-measurable*:  
**shows** *pseudo-proj-True n* ∈ *measurable (nat-filtration n) M* ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-comp-measurable*:  
**assumes** *f* ∈ *measurable M N*  
**and** *f* ∘ *pseudo-proj-True n* = *f*  
**shows** *f* ∈ *measurable (nat-filtration n) N*  
 ⟨*proof*⟩

**definition** (in *infinite-coin-toss-space*) *set-discriminating where*  
*set-discriminating n f N* ≡ (∀ *w*. *f w* ≠ *f (pseudo-proj-True n w)* →  
 (∃ *A* ∈ *sets N*. (*f w* ∈ *A*) = (*f (pseudo-proj-True n w)* ∉ *A*)))

**lemma** (in *infinite-coin-toss-space*) *set-discriminating-if*:  
**fixes** *f*::*bool stream* ⇒ '*b*::{*t0-space*}  
**assumes** *f* ∈ *borel-measurable (nat-filtration n)*  
**shows** *set-discriminating n f borel* ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-not-borel-info*:  
**assumes** *f* ∈ *measurable (nat-filtration n) N*  
**and** *set-discriminating n f N*  
**shows** *f* ∘ *pseudo-proj-True n* = *f*  
 ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-info*:  
**fixes** *f*::*bool stream* ⇒ '*b*::{*t0-space*}  
**assumes** *f* ∈ *borel-measurable (nat-filtration n)*  
**shows** *f* ∘ *pseudo-proj-True n* = *f*  
 ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-not-borel-info'*:  
**assumes** *f* ∈ *measurable (nat-filtration n) N*  
**and** *set-discriminating n f N*  
**shows** *f* ∘ *pseudo-proj-False n* = *f*  
 ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-info'*:

**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } (\text{nat-filtration } n)$   
**shows**  $f \circ \text{pseudo-proj-False } n = f$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-filtration-borel-measurable-characterization:*  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } M$   
**shows**  $f \in \text{borel-measurable } (\text{nat-filtration } n) \longleftrightarrow f \circ \text{pseudo-proj-True } n = f$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-filtration-borel-measurable-init:*  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } (\text{nat-filtration } 0)$   
**shows**  $f = (\lambda w. f (\text{sconst True}))$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-filtration-Suc-sets:*  
**shows**  $\text{sets } (\text{nat-filtration } n) \subseteq \text{sets } (\text{nat-filtration } (\text{Suc } n))$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-filtration-subalgebra:*  
**shows**  $\text{subalgebra } M (\text{nat-filtration } n) \langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-discrete-filtration:*  
**shows**  $\text{filtration } M \text{ nat-filtration}$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-info-filtration:*  
**shows**  $\text{init-triv-filt } M \text{ nat-filtration} \langle \text{proof} \rangle$

**sublocale**  $\text{infinite-cts-filtration} \subseteq \text{triv-init-disc-filtr-prob-space}$   
 $\langle \text{proof} \rangle$

**lemma** (*in infinite-coin-toss-space*) *nat-filtration-vimage-finite:*

**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t2\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable (nat-filtration } n)$   
**shows**  $\text{finite (f(space } M)) \langle \text{proof} \rangle$

**lemma (in infinite-coin-toss-space) nat-filtration-borel-measurable-simple:**  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t2\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable (nat-filtration } n)$   
**shows**  $\text{simple-function } M f$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-coin-toss-space) nat-filtration-singleton-range-set:**  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t2\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable (nat-filtration } n)$   
**shows**  $\exists A \in \text{sets borel. range } f \cap A = \{f x\}$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-coin-toss-space) nat-filtration-borel-measurable-singleton:**  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t2\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable (nat-filtration } n)$   
**shows**  $f - \{f x\} \in \text{sets (nat-filtration } n)$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) borel-adapt-nat-filtration-info:**  
**fixes**  $X::\text{nat} \Rightarrow \text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $\text{borel-adapt-stoch-proc } F X$   
**and**  $m \leq n$   
**shows**  $X m (\text{pseudo-proj-True } n w) = X m w$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-coin-toss-space) nat-filtration-borel-measurable-integrable:**  
**assumes**  $f \in \text{borel-measurable (nat-filtration } n)$   
**shows**  $\text{integrable } M f$   
 $\langle \text{proof} \rangle$

**definition (in infinite-coin-toss-space) spick:: bool stream  $\Rightarrow$  nat  $\Rightarrow$  bool  $\Rightarrow$  bool stream where**  
 $\text{spick } w n v = \text{shift (stake } n w) (v\#\#\text{ sconst True)}$

**lemma (in infinite-coin-toss-space) spickI:**  
**shows**  $\text{stake } n (\text{spick } w n v) = \text{stake } n w \wedge \text{snth (spick } w n v) n = v$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-coin-toss-space) spick-eq-pseudo-proj-True:**  
**shows**  $\text{spick } w n \text{True} = \text{pseudo-proj-True } n w \langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *spick-eq-pseudo-proj-False*:  
**shows**  $spick\ w\ n\ False = pseudo-proj-False\ n\ w$   $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *spick-pseudo-proj*:  
**shows**  $spick\ (pseudo-proj-True\ (Suc\ n)\ w)\ n\ v = spick\ w\ n\ v$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *spick-pseudo-proj-gen*:  
**shows**  $m < n \implies spick\ (pseudo-proj-True\ n\ w)\ m\ v = spick\ w\ m\ v$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *spick-nat-filtration-measurable*:  
**shows**  $(\lambda w. spick\ w\ n\ v) \in measurable\ (nat-filtration\ n)\ M$   
 $\langle proof \rangle$

**definition** (in *infinite-coin-toss-space*) *proj-rep-set*:  
 $proj-rep-set\ n = range\ (pseudo-proj-True\ n)$

**lemma** (in *infinite-coin-toss-space*) *proj-rep-set-finite*:  
**shows**  $finite\ (proj-rep-set\ n)$   $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *set-filt-contain*:  
**assumes**  $A \in sets\ (nat-filtration\ n)$   
**and**  $w \in A$   
**shows**  $pseudo-proj-True\ n - \{pseudo-proj-True\ n\ w\} \subseteq A$   
 $\langle proof \rangle$

**lemma** (in *infinite-cts-filtration*) *measurable-range-rep*:  
**fixes**  $f::bool\ stream \Rightarrow 'b::\{t0-space\}$   
**assumes**  $f \in borel-measurable\ (nat-filtration\ n)$   
**shows**  $range\ f = (\bigcup\ r \in (proj-rep-set\ n). \{f(r)\})$   
 $\langle proof \rangle$

**lemma** (in *infinite-coin-toss-space*) *borel-measurable-stake*:  
**fixes**  $f::bool\ stream \Rightarrow 'b::\{t0-space\}$   
**assumes**  $f \in borel-measurable\ (nat-filtration\ n)$   
**and**  $stake\ n\ w = stake\ n\ y$   
**shows**  $f\ w = f\ y$   
 $\langle proof \rangle$

### 5.3.3 Probability component

The probability component permits to compute measures of subspaces in a straightforward way.

**definition** *prob-component* **where**

*prob-component* ( $p::\text{real}$ )  $w\ n = (\text{if } (\text{snth } w\ n) \text{ then } p \text{ else } 1-p)$

**lemma** *prob-component-neq-zero*:

**assumes**  $0 < p$

**and**  $p < 1$

**shows** *prob-component*  $p\ w\ n \neq 0$  *<proof>*

**lemma** *prob-component-measure*:

**fixes**  $x::\text{bool stream}$

**assumes**  $0 \leq p$

**and**  $p \leq 1$

**shows** *emeasure* (*measure-pmf* (*bernoulli-pmf*  $p$ ))  $\{\text{snth } x\ i\} = \text{prob-component } p\ x\ i$  *<proof>*

**lemma** *stake-preimage-measurable*:

**fixes**  $x::\text{bool stream}$

**assumes**  $\text{Suc } 0 \leq n$  **and**  $M = \text{bernoulli-stream } p$

**shows**  $\{w \in \text{space } M. (\text{stake } n\ w = \text{stake } n\ x)\} \in \text{sets } M$   
*<proof>*

**lemma** *snth-as-fct*:

**fixes**  $b$

**assumes**  $M = \text{bernoulli-stream } p$

**shows** *to-stream*  $-\{w \in \text{space } M. \text{snth } w\ i = b\} = \{X::\text{nat} \Rightarrow \text{bool}. X\ i = b\}$   
*<proof>*

**lemma** *stake-as-fct*:

**assumes**  $\text{Suc } 0 \leq n$  **and**  $M = \text{bernoulli-stream } p$

**shows** *to-stream*  $-\{w \in \text{space } M. (\text{stake } n\ w = \text{stake } n\ x)\} = \{X::\text{nat} \Rightarrow \text{bool}. \forall i. 0 \leq i \wedge i \leq n-1 \longrightarrow X\ i = \text{snth } x\ i\}$   
*<proof>*

**lemma** *bernoulli-stream-npref-prob*:

**fixes**  $x$

**assumes**  $M = \text{bernoulli-stream } p$

**shows** *emeasure*  $M\ \{w \in \text{space } M. (\text{stake } 0\ w = \text{stake } 0\ x)\} = 1$   
*<proof>*

**lemma** *bernoulli-stream-pref-prob*:

**fixes**  $x$



**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $0 \leq p$  **and**  $p \leq 1$   
**shows**  $n \geq \text{Suc } 0 \implies \text{emeasure } M \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} =$   
 $(\prod_{i \in \{0..n-1\}}. \text{prob-component } p \ x \ i)$   
 $\langle \text{proof} \rangle$

**lemma** *bernoulli-stream-pref-prob'*:  
**fixes**  $x$   
**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $p \leq 1$  **and**  $0 \leq p$   
**shows**  $\text{emeasure } M \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} = (\prod_{i \in \{0..<n\}}. \text{prob-component } p \ x \ i)$   
 $\langle \text{proof} \rangle$

**lemma** *bernoulli-stream-stake-prob*:  
**fixes**  $x$   
**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $p \leq 1$  **and**  $0 \leq p$   
**shows**  $\text{measure } M \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} = (\prod_{i \in \{0..<n\}}. \text{prob-component } p \ x \ i)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *bernoulli-stream-pseudo-prob*:  
**fixes**  $x$   
**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $p \leq 1$  **and**  $0 \leq p$   
**and**  $w \in \text{range } (\text{pseudo-proj-True } n)$   
**shows**  $\text{measure } M (\text{pseudo-proj-True } n - \{w\} \cap \text{space } M) = (\prod_{i \in \{0..<n\}}. \text{prob-component } p \ w \ i)$   
 $\langle \text{proof} \rangle$

**lemma** *bernoulli-stream-element-prob-rec*:  
**fixes**  $x$   
**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $0 \leq p$  **and**  $p \leq 1$   
**shows**  $\bigwedge n. \text{emeasure } M \{w \in \text{space } M. (\text{stake } (\text{Suc } n) \ w = \text{stake } (\text{Suc } n) \ x)\} =$   
 $(\text{emeasure } M \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} * \text{prob-component } p \ x \ n)$   
 $\langle \text{proof} \rangle$

**lemma** *bernoulli-stream-element-prob-rec'*:  
**fixes**  $x$   
**assumes**  $M = \text{bernoulli-stream } p$   
**and**  $0 \leq p$  **and**  $p \leq 1$   
**shows**  $\bigwedge n. \text{measure } M \{w \in \text{space } M. (\text{stake } (\text{Suc } n) \ w = \text{stake } (\text{Suc } n) \ x)\} =$   
 $(\text{measure } M \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} * \text{prob-component } p \ x \ n)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *bernoulli-stream-pseudo-prob-rec'*:  
**fixes**  $x$   
**assumes**  $\text{pseudo-proj-True } n \ x = x$   
**shows**  $\text{measure } M \ (\text{pseudo-proj-True } (\text{Suc } n) - \{x\}) =$   
 $(\text{measure } M \ (\text{pseudo-proj-True } n - \{x\}) * \text{prob-component } p \ x \ n)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *bernoulli-stream-pref-prob-pos*:  
**fixes**  $x$   
**assumes**  $0 < p$   
**and**  $p < 1$   
**shows**  $\text{emeasure } M \ \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} > 0$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *bernoulli-stream-pref-prob-neq-zero*:  
**fixes**  $x$   
**assumes**  $0 < p$   
**and**  $p < 1$   
**shows**  $\text{emeasure } M \ \{w \in \text{space } M. (\text{stake } n \ w = \text{stake } n \ x)\} \neq 0$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-element-prob-pref*:  
**assumes**  $w \in \text{range } (\text{pseudo-proj-True } n)$   
**shows**  $\text{emeasure } M \ \{y \in \text{space } M. \exists x \in (\text{pseudo-proj-True } n - \{w\}). y = c \#\#$   
 $x\} =$   
 $\text{prob-component } p \ (c \#\# w) \ 0 * \text{emeasure } M \ ((\text{pseudo-proj-True } n) - \{w\} \cap$   
 $\text{space } M)$   
 $\langle \text{proof} \rangle$

### 5.3.4 Filtration equivalence for the natural filtration

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-null-set*:  
**assumes**  $A \in \text{sets } (\text{nat-filtration } n)$   
**and**  $0 < p$   
**and**  $p < 1$   
**and**  $\text{emeasure } M \ A = 0$   
**shows**  $A = \{\}$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-AE-zero*:  
**fixes**  $f :: \text{bool } \text{stream} \Rightarrow \text{real}$   
**assumes**  $A \in \text{sets } (\text{nat-filtration } n)$   
**and**  $f \in \text{borel-measurable } (\text{nat-filtration } n)$   
**and**  $0 < p$   
**and**  $p < 1$   
**shows**  $\forall w. f \ w = 0$

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-AE-eq*:

fixes  $f :: \text{bool stream} \Rightarrow \text{real}$

assumes  $AE\ w\ \text{in}\ M.\ f\ w = g\ w$

and  $0 < p$

and  $p < 1$

and  $f \in \text{borel-measurable}\ (\text{nat-filtration}\ n)$

and  $g \in \text{borel-measurable}\ (\text{nat-filtration}\ n)$

shows  $f\ w = g\ w$

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *bernoulli-stream-equiv*:

assumes  $N = \text{bernoulli-stream}\ q$

and  $0 < p$

and  $p < 1$

and  $0 < q$

and  $q < 1$

shows *filt-equiv nat-filtration*  $M\ N$  *<proof>*

**lemma** (in *infinite-coin-toss-space*) *bernoulli-nat-filtration*:

assumes  $N = \text{bernoulli-stream}\ q$

and  $0 < q$

and  $q < 1$

and  $0 < p$

and  $p < 1$

shows *infinite-cts-filtration*  $q\ N$  *nat-filtration*

*<proof>*

### 5.3.5 More results on the projection function

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-Suc-prefix*:

shows *pseudo-proj-True*  $(\text{Suc}\ n)\ w = (w!!0)\#\#\ \text{pseudo-proj-True}\ n\ (\text{stl}\ w)$

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-img*:

assumes *pseudo-proj-True*  $n\ w = w$

shows  $w \in \text{range}\ (\text{pseudo-proj-True}\ n)$

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *sconst-if*:

assumes  $\bigwedge n.\ \text{snth}\ w\ n = \text{True}$

shows  $w = \text{sconst}\ \text{True}$

*<proof>*

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-suc-img-pref*:

**shows**  $\text{range} (\text{pseudo-proj-True} (\text{Suc } n)) = \{y. \exists w \in \text{range} (\text{pseudo-proj-True } n). y = \text{True} \#\# w\} \cup \{y. \exists w \in \text{range} (\text{pseudo-proj-True } n). y = \text{False} \#\# w\}$   
 ⟨proof⟩

**lemma** (in *infinite-coin-toss-space*) *reindex-pseudo-proj*:  
**shows**  $(\sum_{w \in \text{range} (\text{pseudo-proj-True } n)}. f (c \#\# w)) = (\sum_{y \in \{y. \exists w \in \text{range} (\text{pseudo-proj-True } n). y = c \#\# w\}}. f y)$   
 ⟨proof⟩

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-True-imp-False*:  
**assumes**  $\text{pseudo-proj-True } n w = \text{pseudo-proj-True } n x$   
**shows**  $\text{pseudo-proj-False } n w = \text{pseudo-proj-False } n x$   
 ⟨proof⟩

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-Suc-prefix*:  
**assumes**  $\text{pseudo-proj-True } n w = \text{pseudo-proj-True } n x$   
**shows**  $\text{pseudo-proj-True} (\text{Suc } n) w \in \{\text{pseudo-proj-True } n x, \text{pseudo-proj-False } n x\}$   
 ⟨proof⟩

**lemma** (in *infinite-coin-toss-space*) *pseudo-proj-Suc-preimage*:  
**shows**  $\text{range} (\text{pseudo-proj-True} (\text{Suc } n)) \cap (\text{pseudo-proj-True } n) - \{ \text{pseudo-proj-True } n x \} = \{ \text{pseudo-proj-True } n x, \text{pseudo-proj-False } n x \}$   
 ⟨proof⟩

**lemma** (in *infinite-cts-filtration*) *f-borel-Suc-preimage*:  
**assumes**  $f \in \text{measurable} (F n) N$   
**and** *set-discriminating*  $n f N$   
**shows**  $\text{range} (\text{pseudo-proj-True} (\text{Suc } n)) \cap f - \{f x\} = (\text{pseudo-proj-True } n) - \{f - \{f x\}\} \cup (\text{pseudo-proj-False } n) - \{f - \{f x\}\}$   
 ⟨proof⟩

**lemma** (in *infinite-cts-filtration*) *pseudo-proj-preimage*:  
**assumes**  $g \in \text{measurable} (F n) N$   
**and** *set-discriminating*  $n g N$   
**shows**  $\text{pseudo-proj-True } n - \{g - \{g z\}\} = \text{pseudo-proj-True } n - \{(\text{pseudo-proj-True } n - \{g - \{g z\}\})\}$   
 ⟨proof⟩

**lemma** (in *infinite-cts-filtration*) *borel-pseudo-proj-preimage*:

**fixes**  $g::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $g \in \text{borel-measurable } (F\ n)$   
**shows**  $\text{pseudo-proj-True } n - ' (g - ' \{g\ z\}) = \text{pseudo-proj-True } n - ' (\text{pseudo-proj-True } n - ' (g - ' \{g\ z\}))$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) pseudo-proj-False-preimage:**  
**assumes**  $g \in \text{measurable } (F\ n)\ N$   
**and**  $\text{set-discriminating } n\ g\ N$   
**shows**  $\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}) = \text{pseudo-proj-False } n - ' (\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}))$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) borel-pseudo-proj-False-preimage:**  
**fixes**  $g::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $g \in \text{borel-measurable } (F\ n)$   
**shows**  $\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}) = \text{pseudo-proj-False } n - ' (\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}))$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) pseudo-proj-preimage':**  
**assumes**  $g \in \text{measurable } (F\ n)\ N$   
**and**  $\text{set-discriminating } n\ g\ N$   
**shows**  $\text{pseudo-proj-True } n - ' (g - ' \{g\ z\}) = g - ' \{g\ z\}$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) borel-pseudo-proj-preimage':**  
**fixes**  $g::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $g \in \text{borel-measurable } (F\ n)$   
**shows**  $\text{pseudo-proj-True } n - ' (g - ' \{g\ z\}) = g - ' \{g\ z\}$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) pseudo-proj-False-preimage':**  
**assumes**  $g \in \text{measurable } (F\ n)\ N$   
**and**  $\text{set-discriminating } n\ g\ N$   
**shows**  $\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}) = g - ' \{g\ z\}$   
 $\langle \text{proof} \rangle$

**lemma (in infinite-cts-filtration) borel-pseudo-proj-False-preimage':**  
**fixes**  $g::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $g \in \text{borel-measurable } (F\ n)$   
**shows**  $\text{pseudo-proj-False } n - ' (g - ' \{g\ z\}) = g - ' \{g\ z\}$   
 $\langle \text{proof} \rangle$

### 5.3.6 Integrals and conditional expectations on the natural filtration

**lemma** (in *infinite-cts-filtration*) *cst-integral*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$   
**assumes**  $f \in \text{borel-measurable } (F \ 0)$   
**and**  $f \ (\text{sconst } \text{True}) = c$

**shows**  $\text{has-bochner-integral } M \ f \ c$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *cst-nn-integral*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$   
**assumes**  $f \in \text{borel-measurable } (F \ 0)$   
**and**  $\bigwedge w. 0 \leq f \ w$   
**and**  $f \ (\text{sconst } \text{True}) = c$

**shows**  $\text{integral}^N \ M \ f = \text{ennreal } c \ \langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *suc-measurable*:

**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } (F \ (\text{Suc } n))$   
**shows**  $(\lambda w. f \ (c \ \#\# \ w)) \in \text{borel-measurable } (F \ n)$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *F-n-nn-integral-pos*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$   
**shows**  $\bigwedge f. (\forall x. 0 \leq f \ x) \Longrightarrow f \in \text{borel-measurable } (F \ n) \Longrightarrow \text{integral}^N \ M \ f =$   
 $(\sum w \in \text{range } (\text{pseudo-proj-True } n). (\text{emeasure } M \ ((\text{pseudo-proj-True } n) - \{w\}$   
 $\cap \text{space } M)) * \text{ennreal } (f \ w))$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *F-n-integral-pos*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$   
**assumes**  $f \in \text{borel-measurable } (F \ n)$   
**and**  $\forall w. 0 \leq f \ w$   
**shows**  $\text{has-bochner-integral } M \ f$   
 $(\sum w \in \text{range } (\text{pseudo-proj-True } n). (\text{measure } M \ ((\text{pseudo-proj-True } n) - \{w\}$   
 $\cap \text{space } M)) * (f \ w))$

$\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *F-n-integral*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$   
**assumes**  $f \in \text{borel-measurable } (F \ n)$   
**shows**  $\text{has-bochner-integral } M \ f$   
 $(\sum w \in \text{range } (\text{pseudo-proj-True } n). (\text{measure } M \ ((\text{pseudo-proj-True } n) - \{w\}$   
 $\cap \text{space } M)) * (f \ w))$

*<proof>*

**lemma** (in *infinite-cts-filtration*) *F-n-integral-prob-comp*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$

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

**shows** *has-bochner-integral*  $M \ f$

$(\sum w \in \text{range } (\text{pseudo-proj-True } n). (\text{prod } (\text{prob-component } p \ w) \ \{0..<n\}) * (f \ w))$

*<proof>*

**lemma** (in *infinite-cts-filtration*) *expect-prob-comp*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$

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

**shows** *expectation*  $f =$

$(\sum w \in \text{range } (\text{pseudo-proj-True } n). (\text{prod } (\text{prob-component } p \ w) \ \{0..<n\}) * (f \ w))$

*<proof>*

**lemma** *sum-union-disjoint'*:

**assumes** *finite*  $A$

**and** *finite*  $B$

**and**  $A \cap B = \{\}$

**and**  $A \cup B = C$

**shows**  $\text{sum } g \ C = \text{sum } g \ A + \text{sum } g \ B$

*<proof>*

**lemma** (in *infinite-cts-filtration*) *borel-Suc-expectation*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$

**assumes**  $f \in \text{borel-measurable } (F \ (\text{Suc } n))$

**and**  $g \in \text{measurable } (F \ n) \ N$

**and** *set-discriminating*  $n \ g \ N$

**and**  $g - \{g \ z\} \in \text{sets } (F \ n)$

**and**  $\forall y \ z. (g \ y = g \ z \wedge \text{snth } y \ n = \text{snth } z \ n) \longrightarrow f \ y = f \ z$

**shows** *expectation*  $(\lambda x. f \ x * \text{indicator } (g - \{g \ z\}) \ x) =$

$\text{prob } (g - \{g \ z\}) * (p * f \ (\text{pseudo-proj-True } n \ z) +$

$(1 - p) * f \ (\text{pseudo-proj-False } n \ z))$

*<proof>*

**lemma** (in *infinite-cts-filtration*) *borel-Suc-expectation-pseudo-proj*:

**fixes**  $f::\text{bool stream} \Rightarrow \text{real}$

**assumes**  $f \in \text{borel-measurable } (F \ (\text{Suc } n))$

**shows** *expectation*  $(\lambda x. f \ x * \text{indicator } (\text{pseudo-proj-True } n - \{\text{pseudo-proj-True } n \ z\}) \ x) =$

$\text{prob } (\text{pseudo-proj-True } n - \{\text{pseudo-proj-True } n \ z\}) *$

$(p * (f \ (\text{pseudo-proj-True } n \ z)) + (1 - p) * (f \ (\text{pseudo-proj-False } n \ z)))$

*<proof>*

**lemma** (in *infinite-cts-filtration*) *f-borel-Suc-expl-cond-expect*:  
**assumes**  $f \in \text{borel-measurable } (F \text{ (Suc } n))$   
**and**  $g \in \text{measurable } (F \text{ } n) \text{ } N$   
**and** *set-discriminating*  $n \text{ } g \text{ } N$   
**and**  $g - \{g \text{ } w\} \in \text{sets } (F \text{ } n)$   
**and**  $\forall y \text{ } z. (g \text{ } y = g \text{ } z \wedge \text{snth } y \text{ } n = \text{snth } z \text{ } n) \longrightarrow f \text{ } y = f \text{ } z$   
**and**  $0 < p$   
**and**  $p < 1$   
**shows**  $\text{expl-cond-expect } M \text{ } g \text{ } f \text{ } w = p * f \text{ (pseudo-proj-True } n \text{ } w) + (1 - p) * f$   
 $\text{(pseudo-proj-False } n \text{ } w)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *f-borel-Suc-real-cond-exp*:  
**assumes**  $f \in \text{borel-measurable } (F \text{ (Suc } n))$   
**and**  $g \in \text{measurable } (F \text{ } n) \text{ } N$   
**and** *set-discriminating*  $n \text{ } g \text{ } N$   
**and**  $\forall w. g - \{g \text{ } w\} \in \text{sets } (F \text{ } n)$   
**and**  $\forall r \in \text{range } g \cap \text{space } N. \exists A \in \text{sets } N. \text{range } g \cap A = \{r\}$   
**and**  $\forall y \text{ } z. (g \text{ } y = g \text{ } z \wedge \text{snth } y \text{ } n = \text{snth } z \text{ } n) \longrightarrow f \text{ } y = f \text{ } z$   
**and**  $0 < p$   
**and**  $p < 1$   
**shows** *AE*  $w$  in  $M. \text{real-cond-exp } M \text{ (fct-gen-subalgebra } M \text{ } N \text{ } g) \text{ } f \text{ } w = p * f$   
 $\text{(pseudo-proj-True } n \text{ } w) + (1 - p) * f \text{ (pseudo-proj-False } n \text{ } w)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *f-borel-Suc-real-cond-exp-proj*:  
**assumes**  $f \in \text{borel-measurable } (F \text{ (Suc } n))$   
**and**  $0 < p$   
**and**  $p < 1$   
**shows** *AE*  $w$  in  $M. \text{real-cond-exp } M \text{ (fct-gen-subalgebra } M \text{ } M \text{ (pseudo-proj-True } n)) \text{ } f \text{ } w =$   
 $p * f \text{ (pseudo-proj-True } n \text{ } w) + (1 - p) * f \text{ (pseudo-proj-False } n \text{ } w)$   
 $\langle \text{proof} \rangle$

## 5.4 Images of stochastic processes by prefixes of streams

We define a function that, given a stream of coin tosses and a stochastic process, returns a stream of the values of the stochastic process up to a given time. This function will be used to characterize the smallest filtration that, at any time  $n$ , makes each random variable of a given stochastic process measurable up to time  $n$ .

### 5.4.1 Definitions

**primrec** *smap-stoch-proc* **where**  
 $\text{smap-stoch-proc } 0 \text{ } f \text{ } k \text{ } w = []$



|  $\text{smap-stoch-proc } (\text{Suc } n) f k w = (f k w) \# (\text{smap-stoch-proc } n f (\text{Suc } k) w)$

**lemma** *smap-stoch-proc-length*:

**shows**  $\text{length } (\text{smap-stoch-proc } n f k w) = n$   
*<proof>*

**lemma** *smap-stoch-proc-nth*:

**shows**  $\text{Suc } p \leq \text{Suc } n \implies \text{nth } (\text{smap-stoch-proc } (\text{Suc } n) f k w) p = f (k+p) w$   
*<proof>*

**definition** *proj-stoch-proc where*

$\text{proj-stoch-proc } f n = (\lambda w. \text{shift } (\text{smap-stoch-proc } n f 0 w) (\text{sconst } (f n w)))$

**lemma** *proj-stoch-proc-component*:

**shows**  $k < n \implies (\text{snth } (\text{proj-stoch-proc } f n w) k) = f k w$   
**and**  $n \leq k \implies (\text{snth } (\text{proj-stoch-proc } f n w) k) = f n w$   
*<proof>*

**lemma** *proj-stoch-proc-component'*:

**assumes**  $k \leq n$   
**shows**  $f k x = \text{snth } (\text{proj-stoch-proc } f n x) k$   
*<proof>*

**lemma** *proj-stoch-proc-eq-snth*:

**assumes**  $\text{proj-stoch-proc } f n x = \text{proj-stoch-proc } f n y$   
**and**  $k \leq n$   
**shows**  $f k x = f k y$   
*<proof>*

**lemma** *proj-stoch-measurable-if-adapted*:

**assumes** *filtration*  $M F$   
**and** *adapt-stoch-proc*  $F f N$   
**shows**  $\text{proj-stoch-proc } f n \in \text{measurable } M (\text{stream-space } N)$   
*<proof>*

**lemma** *proj-stoch-adapted-if-adapted*:

**assumes** *filtration*  $M F$   
**and** *adapt-stoch-proc*  $F f N$   
**shows**  $\text{proj-stoch-proc } f n \in \text{measurable } (F n) (\text{stream-space } N)$   
*<proof>*

**lemma** *proj-stoch-adapted-if-adapted'*:

**assumes** *filtration*  $M F$   
**and** *adapt-stoch-proc*  $F f N$   
**shows** *adapt-stoch-proc*  $F (\text{proj-stoch-proc } f) (\text{stream-space } N)$  *<proof>*

**lemma** (in *infinite-cts-filtration*) *proj-stoch-proj-invariant*:  
**fixes**  $X::nat \Rightarrow bool$  *stream*  $\Rightarrow 'b::\{t0\text{-space}\}$   
**assumes** *borel-adapt-stoch-proc*  $F X$   
**shows** *proj-stoch-proc*  $X n w = \text{proj-stoch-proc } X n (\text{pseudo-proj-True } n w)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *proj-stoch-set-finite-range*:  
**fixes**  $X::nat \Rightarrow bool$  *stream*  $\Rightarrow 'b::\{t0\text{-space}\}$   
**assumes** *borel-adapt-stoch-proc*  $F X$   
**shows** *finite* (*range* (*proj-stoch-proc*  $X n$ ))  
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *proj-stoch-set-discriminating*:  
**fixes**  $X::nat \Rightarrow bool$  *stream*  $\Rightarrow 'b::\{t0\text{-space}\}$   
**assumes** *borel-adapt-stoch-proc*  $F X$   
**shows** *set-discriminating*  $n$  (*proj-stoch-proc*  $X n$ )  $N$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *proj-stoch-preimage*:  
**assumes** *borel-adapt-stoch-proc*  $F X$   
**shows** (*proj-stoch-proc*  $X n$ )  $-'$   $\{\text{proj-stoch-proc } X n w\} = (\bigcap_{i \in \{m. m \leq n\}} (X i) -' \{X i w\})$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *proj-stoch-singleton-set*:  
**fixes**  $X::nat \Rightarrow bool$  *stream*  $\Rightarrow ('b::t2\text{-space})$   
**assumes** *borel-adapt-stoch-proc*  $F X$   
**shows** (*proj-stoch-proc*  $X n$ )  $-'$   $\{\text{proj-stoch-proc } X n w\} \in \text{sets } (F n)$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *finite-range-stream-space*:  
**fixes**  $f::'a \Rightarrow 'b::t1\text{-space}$   
**assumes** *finite* (*range*  $f$ )  
**shows**  $(\lambda w. \text{snth } w i) -' (\text{open-exclude-set } (f x) (\text{range } f)) \in \text{sets } (\text{stream-space borel})$   
 $\langle \text{proof} \rangle$

**lemma** (in *infinite-cts-filtration*) *proj-stoch-range-singleton*:  
**fixes**  $X::nat \Rightarrow bool$  *stream*  $\Rightarrow ('b::t2\text{-space})$   
**assumes** *borel-adapt-stoch-proc*  $F X$   
**and**  $r \in \text{range } (\text{proj-stoch-proc } X n)$   
**shows**  $\exists A \in \text{sets } (\text{stream-space borel}). \text{range } (\text{proj-stoch-proc } X n) \cap A = \{r\}$   
 $\langle \text{proof} \rangle$

**definition** (in *infinite-cts-filtration*) *stream-space-single* **where**

*stream-space-single*  $X r = (\text{if } (\exists U. U \in \text{sets } (\text{stream-space borel}) \wedge U \cap (\text{range } X) = \{r\})$

*then SOME*  $U. U \in \text{sets } (\text{stream-space borel}) \wedge U \cap (\text{range } X) = \{r\} \text{ else } \{\}$ )

**lemma** (in *infinite-cts-filtration*) *stream-space-singleI*:

**assumes**  $\exists U. U \in \text{sets } (\text{stream-space borel}) \wedge U \cap (\text{range } X) = \{r\}$

**shows** *stream-space-single*  $X r \in \text{sets } (\text{stream-space borel}) \wedge \text{stream-space-single } X r \cap (\text{range } X) = \{r\}$

*<proof>*

**lemma** (in *infinite-cts-filtration*)

**fixes**  $X::\text{nat} \Rightarrow \text{bool stream} \Rightarrow ('b::t2\text{-space})$

**assumes** *borel-adapt-stoch-proc*  $F X$

**and**  $r \in \text{range } (\text{proj-stoch-proc } X n)$

**shows** *stream-space-single-set*: *stream-space-single*  $(\text{proj-stoch-proc } X n) r \in \text{sets } (\text{stream-space borel})$

**and** *stream-space-single-preimage*: *stream-space-single*  $(\text{proj-stoch-proc } X n) r \cap \text{range } (\text{proj-stoch-proc } X n) = \{r\}$

*<proof>*

## 5.4.2 Induced filtration, relationship with filtration generated by underlying stochastic process

**definition** *comp-proj-i where*

*comp-proj-i*  $X n i y = \{z \in \text{range } (\text{proj-stoch-proc } X n). \text{snth } z i = y\}$

**lemma** (in *infinite-cts-filtration*) *comp-proj-i-finite*:

**fixes**  $X::\text{nat} \Rightarrow \text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$

**assumes** *borel-adapt-stoch-proc*  $F X$

**shows** *finite*  $(\text{comp-proj-i } X n i y)$

*<proof>*

**lemma** *stoch-proc-comp-proj-i-preimage*:

**assumes**  $i \leq n$

**shows**  $(X i) - \{X i x\} = (\bigcup z \in \text{comp-proj-i } X n i (X i x). (\text{proj-stoch-proc } X n) - \{z\})$

*<proof>*

**definition** *comp-proj where*

*comp-proj*  $X n y = \{z \in \text{range } (\text{proj-stoch-proc } X n). \text{snth } z n = y\}$

**lemma** (in *infinite-cts-filtration*) *comp-proj-finite*:

**fixes**  $X::\text{nat} \Rightarrow \text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$

**assumes** *borel-adapt-stoch-proc*  $F X$

**shows** *finite*  $(\text{comp-proj } X n y)$

*<proof>*

**lemma** *stoch-proc-comp-proj-preimage:*

**shows**  $(X\ n) - \{X\ n\ x\} = (\bigcup_{z \in \text{comp-proj } X\ n} (X\ n\ x). (\text{proj-stoch-proc } X\ n) - \{z\})$   
*<proof>*

**lemma** *comp-proj-stoch-proc-preimage:*

**shows**  $(\text{proj-stoch-proc } X\ n) - \{\text{proj-stoch-proc } X\ n\ x\} = (\bigcap_{i \in \{m. m \leq n\}} (X\ i) - \{X\ i\ x\})$   
*<proof>*

**definition** *stoch-proc-filt where*

*stoch-proc-filt*  $M\ X\ N\ (n::\text{nat}) = \text{gen-subalgebra } M\ (\text{sigma-sets } (\text{space } M) (\bigcup_{i \in \{m. m \leq n\}} \{(X\ i - 'A) \cap (\text{space } M) \mid A. A \in \text{sets } N\}))$

**lemma** *stoch-proc-filt-space:*

**shows**  $\text{space } (\text{stoch-proc-filt } M\ X\ N\ n) = \text{space } M$  *<proof>*

**lemma** *stoch-proc-filt-sets:*

**assumes**  $\bigwedge i. i \leq n \implies (X\ i) \in \text{measurable } M\ N$

**shows**  $\text{sets } (\text{stoch-proc-filt } M\ X\ N\ n) = (\text{sigma-sets } (\text{space } M) (\bigcup_{i \in \{m. m \leq n\}} \{(X\ i - 'A) \cap (\text{space } M) \mid A. A \in \text{sets } N\}))$   
*<proof>*

**lemma** *stoch-proc-filt-adapt:*

**assumes**  $\bigwedge n. X\ n \in \text{measurable } M\ N$

**shows**  $\text{adapt-stoch-proc } (\text{stoch-proc-filt } M\ X\ N) X\ N$  *<proof>*

**lemma** *stoch-proc-filt-disc-filtr:*

**assumes**  $\bigwedge i. (X\ i) \in \text{measurable } M\ N$

**shows**  $\text{disc-filtr } M\ (\text{stoch-proc-filt } M\ X\ N)$  *<proof>*

**lemma** *gen-subalgebra-eq-space-sets:*

**assumes**  $\text{space } M = \text{space } N$

**and**  $P = Q$

**and**  $P \subseteq \text{Pow } (\text{space } M)$

**shows**  $sets (gen-subalgebra M P) = sets (gen-subalgebra N Q)$   $\langle proof \rangle$

**lemma** *stoch-proc-filt-eq-sets*:

**assumes**  $space M = space N$

**shows**  $sets (stoch-proc-filt M X P n) = sets (stoch-proc-filt N X P n)$   $\langle proof \rangle$

**lemma** (**in** *infinite-cts-filtration*) *stoch-proc-filt-triv-init*:

**fixes**  $X::nat \Rightarrow bool \ stream \Rightarrow real$

**assumes** *borel-adapt-stoch-proc nat-filtration X*

**shows** *init-triv-filt M (stoch-proc-filt M X borel)*  $\langle proof \rangle$

**lemma** (**in** *infinite-cts-filtration*) *stream-space-borel-union*:

**fixes**  $X::nat \Rightarrow bool \ stream \Rightarrow ('b::t2-space)$

**assumes** *borel-adapt-stoch-proc F X*

**and**  $i \leq n$

**and**  $A \in sets \ borel$

**shows**  $\forall y \in A \cap range (X i). X i - \{y\} = (proj-stoch-proc X n) - \{ \bigcup z \in comp-proj-i X n \ i \ y.$

$(stream-space-single (proj-stoch-proc X n) z)$

$\langle proof \rangle$

**lemma** (**in** *infinite-cts-filtration*) *proj-stoch-pre-borel*:

**fixes**  $X::nat \Rightarrow bool \ stream \Rightarrow ('b::t2-space)$

**assumes** *borel-adapt-stoch-proc F X*

**shows**  $proj-stoch-proc X n - \{proj-stoch-proc X n x\} \in sets (stoch-proc-filt M X borel n)$

$\langle proof \rangle$

**lemma** (**in** *infinite-cts-filtration*) *stoch-proc-filt-gen*:

**fixes**  $X::nat \Rightarrow bool \ stream \Rightarrow ('b::t2-space)$

**assumes** *borel-adapt-stoch-proc F X*

**shows**  $stoch-proc-filt M X borel n = fct-gen-subalgebra M (stream-space borel) (proj-stoch-proc X n)$

$\langle proof \rangle$

**lemma** (**in** *infinite-coin-toss-space*) *stoch-proc-subalg-nat-filt*:

**assumes** *borel-adapt-stoch-proc nat-filtration X*

**shows** *subalgebra (nat-filtration n) (stoch-proc-filt M X borel n)*  $\langle proof \rangle$

**lemma** (**in** *infinite-coin-toss-space*)

**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 \leq q$   
**and**  $q \leq 1$   
**and**  $0 < p$   
**and**  $p < 1$   
**and**  $\text{filt-equiv nat-filtration } M N$   
**shows**  $\text{filt-equiv-sgt}: 0 < q$  **and**  $\text{filt-equiv-slt}: q < 1$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{stoch-proc-filt-filt-equiv}$ :  
**assumes**  $\text{filt-equiv } F M N$   
**shows**  $\text{stoch-proc-filt } M f P n = \text{stoch-proc-filt } N f P n$   $\langle \text{proof} \rangle$

**lemma**  $\text{filt-equiv-filt}$ :  
**assumes**  $\text{filt-equiv } F M N$   
**and**  $\text{filtration } M G$   
**shows**  $\text{filtration } N G$   $\langle \text{proof} \rangle$

**lemma**  $\text{filt-equiv-borel-AE-eq-iff}$ :  
**fixes**  $f::'a \Rightarrow \text{real}$   
**assumes**  $\text{filt-equiv } F M N$   
**and**  $f \in \text{borel-measurable } (F t)$   
**and**  $g \in \text{borel-measurable } (F t)$   
**and**  $\text{prob-space } N$   
**and**  $\text{prob-space } M$   
**shows**  $(AE w \text{ in } M. f w = g w) \longleftrightarrow (AE w \text{ in } N. f w = g w)$   
 $\langle \text{proof} \rangle$

**lemma** (**in**  $\text{infinite-coin-toss-space}$ )  $\text{filt-equiv-triv-init}$ :  
**assumes**  $\text{filt-equiv } F M N$   
**and**  $\text{init-triv-filt } M G$   
**shows**  $\text{init-triv-filt } N G$   $\langle \text{proof} \rangle$

**lemma** (**in**  $\text{infinite-coin-toss-space}$ )  $\text{fct-gen-subalgebra-meas-info}$ :  
**assumes**  $\forall w. f (g w) = f w$   
**and**  $f \in \text{space } M \rightarrow \text{space } N$   
**and**  $g \in \text{space } M \rightarrow \text{space } M$   
**shows**  $g \in \text{measurable } (\text{fct-gen-subalgebra } M N f) (\text{fct-gen-subalgebra } M N f)$   
 $\langle \text{proof} \rangle$

**end**  
**theory**  $\text{Geometric-Random-Walk}$  **imports**  $\text{Infinite-Coin-Toss-Space}$

**begin**

## 6 Geometric random walk

A geometric random walk is a stochastic process that can, at each time, move upwards or downwards, depending on the outcome of a coin toss.

**fun** (in *infinite-coin-toss-space*) *geom-rand-walk*:: *real*  $\Rightarrow$  *real*  $\Rightarrow$  *real*  $\Rightarrow$  (*nat*  $\Rightarrow$  *bool stream*  $\Rightarrow$  *real*) **where**

*base*: (*geom-rand-walk* *u d v*) 0 = ( $\lambda w. v$ )  
*step*: (*geom-rand-walk* *u d v*) (*Suc n*) = ( $\lambda w. ((\lambda True \Rightarrow u \mid False \Rightarrow d) (snth w n)) * (geom-rand-walk u d v) n w$ )

**locale** *prob-grw* = *infinite-coin-toss-space* +  
**fixes** *geom-proc*::*nat*  $\Rightarrow$  *bool stream*  $\Rightarrow$  *real* **and** *u*::*real* **and** *d*::*real* **and** *init*::*real*  
**assumes** *geometric-process*:*geom-proc* = *geom-rand-walk* *u d init*

**lemma** (in *prob-grw*) *geom-rand-walk-borel-measurable*:  
**shows** (*geom-proc* *n*)  $\in$  *borel-measurable M*  
*<proof>*

**lemma** (in *prob-grw*) *geom-rand-walk-pseudo-proj-True*:  
**shows** *geom-proc* *n* = *geom-proc* *n*  $\circ$  *pseudo-proj-True* *n*  
*<proof>*

**lemma** (in *prob-grw*) *geom-rand-walk-pseudo-proj-False*:  
**shows** *geom-proc* *n* = *geom-proc* *n*  $\circ$  *pseudo-proj-False* *n*  
*<proof>*

**lemma** (in *prob-grw*) *geom-rand-walk-borel-adapted*:  
**shows** *borel-adapt-stoch-proc* *nat-filtration* *geom-proc*  
*<proof>*

**lemma** (in *prob-grw*) *grw-succ-img*:  
**assumes** (*geom-proc* *n*) - ' {*x*}  $\neq$  {}  
**shows** (*geom-proc* (*Suc n*)) - ' ((*geom-proc* *n*) - ' {*x*}) = {*u\*x*, *d\*x*}  
*<proof>*

**lemma** (in *prob-grw*) *geom-rand-walk-strictly-positive*:  
**assumes** 0 < *init*  
**and** 0 < *d*  
**and** *d* < *u*  
**shows**  $\forall n w. 0 < \text{geom-proc } n w$   
*<proof>*

**lemma** (in *prob-grw*) *geom-rand-walk-diff-induct*:  
**shows**  $\bigwedge w. (\text{geom-proc } (\text{Suc } n) (\text{spick } w \ n \ \text{True}) - \text{geom-proc } (\text{Suc } n) (\text{spick } w \ n \ \text{False})) = (\text{geom-proc } n \ w * (u - d))$   
*<proof>*

**end**

## 7 Fair Prices

This section contains the formalization of financial notions, such as markets, price processes, portfolios, arbitrages, fair prices, etc. It also defines risk-neutral probability spaces, and proves the main result about the fair price of a derivative in a risk-neutral probability space, namely that this fair price is equal to the expectation of the discounted value of the derivative's payoff.

**theory** *Fair-Price* **imports** *Filtration Martingale Geometric-Random-Walk*  
**begin**

### 7.1 Preliminary results

#### 7.1.1 On the almost everywhere filter

**lemma** *AE-eq-trans*[*trans*]:  
**assumes** *AE x in M. A x = B x*  
**and** *AE x in M. B x = C x*  
**shows** *AE x in M. A x = C x*  
*<proof>*

**abbreviation** *AEeq* **where** *AEeq M X Y*  $\equiv$  *AE w in M. X w = Y w*

**lemma** *AE-add*:  
**assumes** *AE w in M. f w = g w*  
**and** *AE w in M. f' w = g' w*  
**shows** *AE w in M. f w + f' w = g w + g' w* *<proof>*

**lemma** *AE-sum*:  
**assumes** *finite I*  
**and**  $\forall i \in I. \text{AE } w \text{ in } M. f \ i \ w = g \ i \ w$   
**shows** *AE w in M.  $(\sum i \in I. f \ i \ w) = (\sum i \in I. g \ i \ w)$*  *<proof>*

**lemma** *AE-eq-cst*:  
**assumes** *AE w in M.  $(\lambda w. c) \ w = (\lambda w. d) \ w$*



**and** *emeasure*  $M$  (*space*  $M$ )  $\neq 0$   
**shows**  $c = d$   
*<proof>*

### 7.1.2 On conditional expectations

**lemma** (*in prob-space*) *subalgebra-sigma-finite*:  
**assumes** *subalgebra*  $M N$   
**shows** *sigma-finite-subalgebra*  $M N$  *<proof>*

**lemma** (*in prob-space*) *trivial-subalg-cond-expect-AE*:  
**assumes** *subalgebra*  $M N$   
**and** *sets*  $N = \{\{\}, \text{space } M\}$   
**and** *integrable*  $M f$   
**shows** *AE*  $x$  *in*  $M$ . *real-cond-exp*  $M N f x = (\lambda x. \text{expectation } f) x$   
*<proof>*

**lemma** (*in prob-space*) *triv-subalg-borel-eq*:  
**assumes** *subalgebra*  $M F$   
**and** *sets*  $F = \{\{\}, \text{space } M\}$   
**and** *AE*  $x$  *in*  $M$ .  $f x = (c::'b::\{t2\text{-space}\})$   
**and**  $f \in \text{borel-measurable } F$   
**shows**  $\forall x \in \text{space } M. f x = c$   
*<proof>*

**lemma** (*in prob-space*) *trivial-subalg-cond-expect-eq*:  
**assumes** *subalgebra*  $M N$   
**and** *sets*  $N = \{\{\}, \text{space } M\}$   
**and** *integrable*  $M f$   
**shows**  $\forall x \in \text{space } M. \text{real-cond-exp } M N f x = \text{expectation } f$   
*<proof>*

**lemma** (*in sigma-finite-subalgebra*) *real-cond-exp-cong'*:  
**assumes**  $\forall w \in \text{space } M. f w = g w$   
**and**  $f \in \text{borel-measurable } M$   
**shows** *AE*  $w$  *in*  $M$ . *real-cond-exp*  $M F f w = \text{real-cond-exp } M F g w$   
*<proof>*

**lemma** (*in sigma-finite-subalgebra*) *real-cond-exp-bsum* :  
**fixes**  $f::'b \Rightarrow 'a \Rightarrow \text{real}$   
**assumes** [*measurable*]:  $\bigwedge i. i \in I \implies \text{integrable } M (f i)$   
**shows** *AE*  $x$  *in*  $M$ . *real-cond-exp*  $M F (\lambda x. \sum i \in I. f i x) x = (\sum i \in I. \text{real-cond-exp } M F (f i) x)$

*<proof>*

## 7.2 Financial formalizations

### 7.2.1 Markets

**definition** *stk-strict-subs*::'c set  $\Rightarrow$  bool **where**  
*stk-strict-subs*  $S \longleftrightarrow S \neq UNIV$

**typedef** ('a,'c) *discrete-market* = {(s::('c set), a::'c  $\Rightarrow$  (nat  $\Rightarrow$  'a  $\Rightarrow$  real)). *stk-strict-subs* s} *<proof>*

**definition** *prices* **where**  
*prices*  $Mkt = (snd (Rep-discrete-market Mkt))$

**definition** *assets* **where**

*assets*  $Mkt = UNIV$

**definition** *stocks* **where**  
*stocks*  $Mkt = (fst (Rep-discrete-market Mkt))$

**definition** *discrete-market-of*  
**where**  
*discrete-market-of*  $S A =$   
*Abs-discrete-market* (if (*stk-strict-subs*  $S$ ) then  $S$  else {},  $A$ )

**lemma** *prices-of*:  
**shows** *prices* (*discrete-market-of*  $S A$ ) =  $A$   
*<proof>*

**lemma** *stocks-of*:  
**assumes**  $UNIV \neq S$   
**shows** *stocks* (*discrete-market-of*  $S A$ ) =  $S$   
*<proof>*

**lemma** *mkt-stocks-assets*:  
**shows** *stk-strict-subs* (*stocks*  $Mkt$ ) *<proof>*

### 7.2.2 Quantity processes and portfolios

These are functions that assign quantities to assets; each quantity is a stochastic process. Basic operations are defined on these processes.

**Basic operations** **definition** *qty-empty* **where**  
*qty-empty* = ( $\lambda (x::'a) (n::nat) w. 0::real$ )

**definition** *qty-single* **where**

$qty\text{-}single\ asset\ qt\text{-}proc = (qty\text{-}empty(asset := qt\text{-}proc))$

**definition**  $qty\text{-}sum::('b \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow ('b \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow ('b \Rightarrow nat \Rightarrow 'a \Rightarrow real)$  **where**  
 $qty\text{-}sum\ pf1\ pf2 = (\lambda x\ n\ w.\ pf1\ x\ n\ w + pf2\ x\ n\ w)$

**definition**  $qty\text{-}mult\text{-}comp::('b \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow (nat \Rightarrow 'a \Rightarrow real) \Rightarrow ('b \Rightarrow nat \Rightarrow 'a \Rightarrow real)$  **where**  
 $qty\text{-}mult\text{-}comp\ pf1\ qty = (\lambda x\ n\ w.\ (pf1\ x\ n\ w) * (qty\ n\ w))$

**definition**  $qty\text{-}rem\text{-}comp::('b \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow 'b \Rightarrow ('b \Rightarrow nat \Rightarrow 'a \Rightarrow real)$  **where**  
 $qty\text{-}rem\text{-}comp\ pf1\ x = pf1(x := (\lambda n\ w.\ 0))$

**definition**  $qty\text{-}replace\text{-}comp$  **where**

$qty\text{-}replace\text{-}comp\ pf1\ x\ pf2 = qty\text{-}sum\ (qty\text{-}rem\text{-}comp\ pf1\ x)\ (qty\text{-}mult\text{-}comp\ pf2\ (pf1\ x))$

**Support sets** If  $p\ x\ n\ w$  is different from 0, this means that this quantity is held on interval  $]n-1, n]$ .

**definition**  $support\text{-}set::('b \Rightarrow nat \Rightarrow 'a \Rightarrow real) \Rightarrow 'b\ set$  **where**  
 $support\text{-}set\ p = \{x.\ \exists\ n\ w.\ p\ x\ n\ w \neq 0\}$

**lemma**  $qty\text{-}empty\text{-}support\text{-}set:$

**shows**  $support\text{-}set\ qty\text{-}empty = \{\}$   $\langle proof \rangle$

**lemma**  $sum\text{-}support\text{-}set:$

**shows**  $support\text{-}set\ (qty\text{-}sum\ pf1\ pf2) \subseteq (support\text{-}set\ pf1) \cup (support\text{-}set\ pf2)$   
 $\langle proof \rangle$

**lemma**  $mult\text{-}comp\text{-}support\text{-}set:$

**shows**  $support\text{-}set\ (qty\text{-}mult\text{-}comp\ pf1\ qty) \subseteq (support\text{-}set\ pf1)$   
 $\langle proof \rangle$

**lemma**  $remove\text{-}comp\text{-}support\text{-}set:$

**shows**  $support\text{-}set\ (qty\text{-}rem\text{-}comp\ pf1\ x) \subseteq ((support\text{-}set\ pf1) - \{x\})$   
 $\langle proof \rangle$

**lemma**  $replace\text{-}comp\text{-}support\text{-}set:$

**shows**  $support\text{-}set\ (qty\text{-}replace\text{-}comp\ pf1\ x\ pf2) \subseteq (support\text{-}set\ pf1 - \{x\}) \cup support\text{-}set\ pf2$   
 $\langle proof \rangle$

**lemma**  $single\text{-}comp\text{-}support:$

**shows**  $support\text{-}set\ (qty\text{-}single\ asset\ qty) \subseteq \{asset\}$   
 $\langle proof \rangle$

**lemma**  $single\text{-}comp\text{-}nz\text{-}support:$

**assumes**  $\exists\ n\ w.\ qty\ n\ w \neq 0$

**shows** *support-set* (*qty-single asset qty*) = {*asset*}  
⟨*proof*⟩

**Portfolios** **definition** *portfolio where*  
*portfolio p*  $\longleftrightarrow$  *finite* (*support-set p*)

**definition** *stock-portfolio* :: ('a, 'b) *discrete-market*  $\Rightarrow$  ('b  $\Rightarrow$  nat  $\Rightarrow$  'a  $\Rightarrow$  real)  $\Rightarrow$   
*bool* **where**  
*stock-portfolio Mkt p*  $\longleftrightarrow$  *portfolio p*  $\wedge$  *support-set p*  $\subseteq$  *stocks Mkt*

**lemma** *sum-portfolio*:  
**assumes** *portfolio pf1*  
**and** *portfolio pf2*  
**shows** *portfolio (qty-sum pf1 pf2)* ⟨*proof*⟩

**lemma** *sum-basic-support-set*:  
**assumes** *stock-portfolio Mkt pf1*  
**and** *stock-portfolio Mkt pf2*  
**shows** *stock-portfolio Mkt (qty-sum pf1 pf2)* ⟨*proof*⟩

**lemma** *mult-comp-portfolio*:  
**assumes** *portfolio pf1*  
**shows** *portfolio (qty-mult-comp pf1 qty)* ⟨*proof*⟩

**lemma** *mult-comp-basic-support-set*:  
**assumes** *stock-portfolio Mkt pf1*  
**shows** *stock-portfolio Mkt (qty-mult-comp pf1 qty)* ⟨*proof*⟩

**lemma** *remove-comp-portfolio*:  
**assumes** *portfolio pf1*  
**shows** *portfolio (qty-rem-comp pf1 x)* ⟨*proof*⟩

**lemma** *remove-comp-basic-support-set*:  
**assumes** *stock-portfolio Mkt pf1*  
**shows** *stock-portfolio Mkt (qty-mult-comp pf1 qty)* ⟨*proof*⟩

**lemma** *replace-comp-portfolio*:  
**assumes** *portfolio pf1*  
**and** *portfolio pf2*  
**shows** *portfolio (qty-replace-comp pf1 x pf2)* ⟨*proof*⟩

**lemma** *replace-comp-stocks*:  
**assumes** *support-set pf1*  $\subseteq$  *stocks Mkt*  $\cup$  {*x*}  
**and** *support-set pf2*  $\subseteq$  *stocks Mkt*

**shows**  $\text{support-set } (\text{qty-replace-comp } pf1 \ x \ pf2) \subseteq \text{stocks } Mkt$   
 $\langle \text{proof} \rangle$

**lemma** *single-comp-portfolio*:  
**shows**  $\text{portfolio } (\text{qty-single asset qty})$   
 $\langle \text{proof} \rangle$

**Value processes** **definition** *val-process* **where**  
 $\text{val-process } Mkt \ p = (\text{if } (\neg (\text{portfolio } p)) \text{ then } (\lambda \ n \ w. \ 0)$   
 $\text{else } (\lambda \ n \ w. (\text{sum } (\lambda x. ((\text{prices } Mkt) \ x \ n \ w) * (p \ x \ (\text{Suc } n) \ w))) (\text{support-set } p))))$

**lemma** *subset-val-process'*:  
**assumes**  $\text{finite } A$   
**and**  $\text{support-set } p \subseteq A$   
**shows**  $\text{val-process } Mkt \ p \ n \ w = (\text{sum } (\lambda x. ((\text{prices } Mkt) \ x \ n \ w) * (p \ x \ (\text{Suc } n) \ w))) A$   
 $\langle \text{proof} \rangle$

**lemma** *sum-val-process*:  
**assumes**  $\text{portfolio } pf1$   
**and**  $\text{portfolio } pf2$   
**shows**  $\forall n \ w. \text{val-process } Mkt \ (\text{qty-sum } pf1 \ pf2) \ n \ w = (\text{val-process } Mkt \ pf1) \ n \ w$   
 $+ (\text{val-process } Mkt \ pf2) \ n \ w$   
 $\langle \text{proof} \rangle$

**lemma** *mult-comp-val-process*:  
**assumes**  $\text{portfolio } pf1$   
**shows**  $\forall n \ w. \text{val-process } Mkt \ (\text{qty-mult-comp } pf1 \ qty) \ n \ w = ((\text{val-process } Mkt \ pf1) \ n \ w) * (\text{qty } (\text{Suc } n) \ w)$   
 $\langle \text{proof} \rangle$

**lemma** *remove-comp-values*:  
**assumes**  $x \neq y$   
**shows**  $\forall n \ w. \text{pf1 } x \ n \ w = (\text{qty-rem-comp } pf1 \ y) \ x \ n \ w$   
 $\langle \text{proof} \rangle$

**lemma** *remove-comp-val-process:*

**assumes** *portfolio pf1*

**shows**  $\forall n w. \text{val-process } Mkt \text{ (qty-rem-comp pf1 y) } n w = ((\text{val-process } Mkt \text{ pf1}) n w) - (\text{prices } Mkt \text{ y } n w) * (\text{pf1 y (Suc n) } w)$   
*<proof>*

**lemma** *replace-comp-val-process:*

**assumes**  $\forall n w. \text{prices } Mkt \text{ x } n w = \text{val-process } Mkt \text{ pf2 } n w$

**and** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall n w. \text{val-process } Mkt \text{ (qty-replace-comp pf1 x pf2) } n w = \text{val-process } Mkt \text{ pf1 } n w$   
*<proof>*

**lemma** *qty-single-val-process:*

**shows**  $\text{val-process } Mkt \text{ (qty-single asset qty) } n w = \text{prices } Mkt \text{ asset } n w * \text{qty (Suc n) } w$   
*<proof>*

### 7.2.3 Trading strategies

**locale** *disc-equity-market = triv-init-disc-filtr-prob-space +*  
**fixes** *Mkt::('a,'b) discrete-market*

#### Discrete predictable processes

**Trading strategy definition** (in *disc-filtr-prob-space*) *trading-strategy*

**where**

*trading-strategy p*  $\longleftrightarrow$  *portfolio p*  $\wedge (\forall \text{asset} \in \text{support-set } p. \text{borel-predict-stoch-proc } F \text{ (p asset)})$

**definition** (in *disc-filtr-prob-space*) *support-adapt:: ('a, 'b) discrete-market*  $\Rightarrow$  (*'b*  $\Rightarrow$  *nat*  $\Rightarrow$  *'a*  $\Rightarrow$  *real*)  $\Rightarrow$  *bool* **where**

*support-adapt Mkt pf*  $\longleftrightarrow (\forall \text{asset} \in \text{support-set } pf. \text{borel-adapt-stoch-proc } F \text{ (prices } Mkt \text{ asset)})$

**lemma** (in *disc-filtr-prob-space*) *quantity-adapted:*

**assumes**  $\forall \text{asset} \in \text{support-set } p. p \text{ asset (Suc n)} \in \text{borel-measurable } (F \text{ n})$

$\forall \text{asset} \in \text{support-set } p. \text{prices } Mkt \text{ asset } n \in \text{borel-measurable } (F \text{ n})$

**shows**  $\text{val-process } Mkt \text{ p } n \in \text{borel-measurable } (F \text{ n})$   
*<proof>*

**lemma** (in *disc-filtr-prob-space*) *trading-strategy-adapted:*

**assumes** *trading-strategy p*  
**and** *support-adapt Mkt p*  
**shows** *borel-adapt-stoch-proc F (val-process Mkt p) <proof>*

**lemma** (**in** *disc-equity-market*) *ats-val-process-adapted*:  
**assumes** *trading-strategy p*  
**and** *support-adapt Mkt p*  
**shows** *borel-adapt-stoch-proc F (val-process Mkt p) <proof>*

**lemma** (**in** *disc-equity-market*) *trading-strategy-init*:  
**assumes** *trading-strategy p*  
**and** *support-adapt Mkt p*  
**shows**  $\exists c. \forall w \in \text{space } M. \text{val-process } Mkt\ p\ 0\ w = c$  *<proof>*

**definition** (**in** *disc-equity-market*) *initial-value where*  
*initial-value pf = constant-image (val-process Mkt pf 0)*

**lemma** (**in** *disc-equity-market*) *initial-valueI*:  
**assumes** *trading-strategy pf*  
**and** *support-adapt Mkt pf*  
**shows**  $\forall w \in \text{space } M. \text{val-process } Mkt\ pf\ 0\ w = \text{initial-value } pf$  *<proof>*

**lemma** (**in** *disc-equity-market*) *inc-predict-support-trading-strat*:  
**assumes** *trading-strategy pf1*  
**shows**  $\forall \text{asset} \in \text{support-set } pf1 \cup B. \text{borel-predict-stoch-proc } F\ (pf1\ \text{asset})$   
*<proof>*

**lemma** (**in** *disc-equity-market*) *inc-predict-support-trading-strat'*:  
**assumes** *trading-strategy pf1*  
**and** *asset*  $\in$  *support-set pf1*  $\cup$  *B*  
**shows** *borel-predict-stoch-proc F (pf1 asset)*  
*<proof>*

**lemma** (**in** *disc-equity-market*) *inc-support-trading-strat*:  
**assumes** *trading-strategy pf1*  
**shows**  $\forall \text{asset} \in \text{support-set } pf1 \cup B. \text{borel-adapt-stoch-proc } F\ (pf1\ \text{asset})$  *<proof>*

**lemma** (**in** *disc-equity-market*) *qty-empty-trading-strat*:  
**shows** *trading-strategy qty-empty* *<proof>*

**lemma** (in *disc-equity-market*) *sum-trading-strat*:  
**assumes** *trading-strategy pf1*  
**and** *trading-strategy pf2*  
**shows** *trading-strategy (qty-sum pf1 pf2)*  
 ⟨*proof*⟩

**lemma** (in *disc-equity-market*) *mult-comp-trading-strat*:  
**assumes** *trading-strategy pf1*  
**and** *borel-predict-stoch-proc F qty*  
**shows** *trading-strategy (qty-mult-comp pf1 qty)*  
 ⟨*proof*⟩

**lemma** (in *disc-equity-market*) *remove-comp-trading-strat*:  
**assumes** *trading-strategy pf1*  
**shows** *trading-strategy (qty-rem-comp pf1 x)*  
 ⟨*proof*⟩

**lemma** (in *disc-equity-market*) *replace-comp-trading-strat*:  
**assumes** *trading-strategy pf1*  
**and** *trading-strategy pf2*  
**shows** *trading-strategy (qty-replace-comp pf1 x pf2)* ⟨*proof*⟩

#### 7.2.4 Self-financing portfolios

**Closing value process** **fun** *up-cl-proc* **where**  
*up-cl-proc Mkt p 0 = val-process Mkt p 0 |*  
*up-cl-proc Mkt p (Suc n) = (λw. ∑<sub>x∈support-set p. prices Mkt x (Suc n) w \* p</sub>*  
*x (Suc n) w)*

**definition** *cls-val-process* **where**  
*cls-val-process Mkt p = (if (¬ (portfolio p)) then (λ n w. 0)*  
*else (λ n w . up-cl-proc Mkt p n w))*

**lemma** (in *disc-filtr-prob-space*) *quantity-updated-borel*:  
**assumes**  $\forall n. \forall \text{asset} \in \text{support-set } p. p \text{ asset } (\text{Suc } n) \in \text{borel-measurable } (F \ n)$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } p. \text{prices Mkt asset } n \in \text{borel-measurable } (F \ n)$   
**shows**  $\forall n. \text{cls-val-process Mkt p } n \in \text{borel-measurable } (F \ n)$   
 ⟨*proof*⟩

**lemma** (in *disc-equity-market*) *cls-val-process-adapted*:  
**assumes** *trading-strategy p*  
**and** *support-adapt Mkt p*  
**shows** *borel-adapt-stoch-proc F (cls-val-process Mkt p)*  
 ⟨*proof*⟩



**lemma** *subset-cls-val-process*:

**assumes** *finite A*

**and** *support-set p ⊆ A*

**shows**  $\forall n w. \text{cls-val-process Mkt } p \text{ (Suc } n) w = (\text{sum } (\lambda x. ((\text{prices Mkt}) x \text{ (Suc } n) w) * (p x \text{ (Suc } n) w)) A)$   
*<proof>*

**lemma** *subset-cls-val-process'*:

**assumes** *finite A*

**and** *support-set p ⊆ A*

**shows**  $\text{cls-val-process Mkt } p \text{ (Suc } n) w = (\text{sum } (\lambda x. ((\text{prices Mkt}) x \text{ (Suc } n) w) * (p x \text{ (Suc } n) w)) A)$   
*<proof>*

**lemma** *sum-cls-val-process-Suc*:

**assumes** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall n w. \text{cls-val-process Mkt (qty-sum pf1 pf2) (Suc } n) w = (\text{cls-val-process Mkt pf1) (Suc } n) w + (\text{cls-val-process Mkt pf2) (Suc } n) w$   
*<proof>*

**lemma** *sum-cls-val-process0*:

**assumes** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall w. \text{cls-val-process Mkt (qty-sum pf1 pf2) 0 } w = (\text{cls-val-process Mkt pf1) 0 } w + (\text{cls-val-process Mkt pf2) 0 } w$  *<proof>*

**lemma** *sum-cls-val-process*:

**assumes** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall n w. \text{cls-val-process Mkt (qty-sum pf1 pf2) } n w = (\text{cls-val-process Mkt pf1) } n w + (\text{cls-val-process Mkt pf2) } n w$   
*<proof>*

**lemma** *mult-comp-cls-val-process0*:

**assumes** *portfolio pf1*

**shows**  $\forall w. \text{cls-val-process Mkt (qty-mult-comp pf1 qty) 0 } w = ((\text{cls-val-process Mkt pf1) 0 } w) * (\text{qty (Suc 0) } w)$  *<proof>*

**lemma** *mult-comp-cls-val-process-Suc*:

**assumes** *portfolio pf1*

**shows**  $\forall n w. \text{cls-val-process Mkt (qty-mult-comp pf1 qty) (Suc } n) w = ((\text{cls-val-process Mkt pf1) (Suc } n) w) * (\text{qty (Suc } n) w)$   
*<proof>*

**lemma** *remove-comp-cls-val-process0*:

**assumes** *portfolio pf1*

**shows**  $\forall w. \text{cls-val-process } Mkt \text{ (qty-rem-comp pf1 y) } 0 \ w =$

$((\text{cls-val-process } Mkt \text{ pf1) } 0 \ w) - (\text{prices } Mkt \text{ y } 0 \ w) * (\text{pf1 y (Suc 0) w}) \langle \text{proof} \rangle$

**lemma** *remove-comp-cls-val-process-Suc*:

**assumes** *portfolio pf1*

**shows**  $\forall n \ w. \text{cls-val-process } Mkt \text{ (qty-rem-comp pf1 y) (Suc n) } w =$

$((\text{cls-val-process } Mkt \text{ pf1) (Suc n) } w) - (\text{prices } Mkt \text{ y (Suc n) } w) * (\text{pf1 y (Suc n) } w) \langle \text{proof} \rangle$

**lemma** *replace-comp-cls-val-process0*:

**assumes**  $\forall w. \text{prices } Mkt \ x \ 0 \ w = \text{cls-val-process } Mkt \ \text{pf2 } 0 \ w$

**and** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall w. \text{cls-val-process } Mkt \text{ (qty-replace-comp pf1 x pf2) } 0 \ w = \text{cls-val-process } Mkt \ \text{pf1 } 0 \ w$

$\langle \text{proof} \rangle$

**lemma** *replace-comp-cls-val-process-Suc*:

**assumes**  $\forall n \ w. \text{prices } Mkt \ x \ (\text{Suc } n) \ w = \text{cls-val-process } Mkt \ \text{pf2 } (\text{Suc } n) \ w$

**and** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall n \ w. \text{cls-val-process } Mkt \text{ (qty-replace-comp pf1 x pf2) (Suc n) } w = \text{cls-val-process } Mkt \ \text{pf1 } (\text{Suc } n) \ w$

$\langle \text{proof} \rangle$

**lemma** *replace-comp-cls-val-process*:

**assumes**  $\forall n \ w. \text{prices } Mkt \ x \ n \ w = \text{cls-val-process } Mkt \ \text{pf2 } n \ w$

**and** *portfolio pf1*

**and** *portfolio pf2*

**shows**  $\forall n \ w. \text{cls-val-process } Mkt \text{ (qty-replace-comp pf1 x pf2) } n \ w = \text{cls-val-process } Mkt \ \text{pf1 } n \ w$

$\langle \text{proof} \rangle$

**lemma** *qty-single-updated*:

**shows**  $\text{cls-val-process } Mkt \text{ (qty-single asset qty) (Suc n) } w =$

$\text{prices } Mkt \ \text{asset } (\text{Suc } n) \ w * \text{qty } (\text{Suc } n) \ w$

$\langle \text{proof} \rangle$

**Self-financing definition** *self-financing* where

*self-financing*  $Mkt\ p \longleftrightarrow (\forall n. \text{val-process } Mkt\ p\ (Suc\ n) = \text{cls-val-process } Mkt\ p\ (Suc\ n))$

**lemma** *self-financingE*:

**assumes** *self-financing*  $Mkt\ p$

**shows**  $\forall n. \text{val-process } Mkt\ p\ n = \text{cls-val-process } Mkt\ p\ n$

*<proof>*

**lemma** *static-portfolio-self-financing*:

**assumes**  $\forall x \in \text{support-set } p. (\forall w\ i. p\ x\ i\ w = p\ x\ (Suc\ i)\ w)$

**shows** *self-financing*  $Mkt\ p$

*<proof>*

**lemma** *sum-self-financing*:

**assumes** *portfolio*  $pf1$

**and** *portfolio*  $pf2$

**and** *self-financing*  $Mkt\ pf1$

**and** *self-financing*  $Mkt\ pf2$

**shows** *self-financing*  $Mkt\ (\text{qty-sum } pf1\ pf2)$

*<proof>*

**lemma** *mult-time-constant-self-financing*:

**assumes** *portfolio*  $pf1$

**and** *self-financing*  $Mkt\ pf1$

**and**  $\forall n\ w. \text{qty } n\ w = \text{qty } (Suc\ n)\ w$

**shows** *self-financing*  $Mkt\ (\text{qty-mult-comp } pf1\ \text{qty})$

*<proof>*

**lemma** *replace-comp-self-financing*:

**assumes**  $\forall n\ w. \text{prices } Mkt\ x\ n\ w = \text{cls-val-process } Mkt\ pf2\ n\ w$

**and** *portfolio*  $pf1$

**and** *portfolio*  $pf2$

**and** *self-financing*  $Mkt\ pf1$

**and** *self-financing*  $Mkt\ pf2$

**shows** *self-financing*  $Mkt\ (\text{qty-replace-comp } pf1\ x\ pf2)$

*<proof>*

**Make a portfolio self-financing** **fun** *remaining-qty* **where**

*init*: *remaining-qty*  $Mkt\ v\ pf\ \text{asset } 0 = (\lambda w. 0) \mid$

*first*: *remaining-qty*  $Mkt\ v\ pf\ \text{asset } (Suc\ 0) = (\lambda w. (v - \text{val-process } Mkt\ pf\ 0$

$w)/(prices\ Mkt\ asset\ 0\ w)) \mid$   
*step: remaining-qty Mkt v pf asset (Suc (Suc n)) =  $(\lambda w. (remaining-qty\ Mkt\ v\ pf\ asset\ (Suc\ n)\ w) +$*   
 *$(cls-val-process\ Mkt\ pf\ (Suc\ n)\ w - val-process\ Mkt\ pf\ (Suc\ n)\ w)/(prices\ Mkt\ asset\ (Suc\ n)\ w))$*

**lemma** (in *disc-equity-market*) *remaining-qty-predict'*:  
**assumes** *borel-adapt-stoch-proc F (prices Mkt asset)*  
**and** *trading-strategy pf*  
**and** *support-adapt Mkt pf*  
**shows** *remaining-qty Mkt v pf asset (Suc n)  $\in$  borel-measurable (F n)*  
 *$\langle$ proof $\rangle$*

**lemma** (in *disc-equity-market*) *remaining-qty-predict*:  
**assumes** *borel-adapt-stoch-proc F (prices Mkt asset)*  
**and** *trading-strategy pf*  
**and** *support-adapt Mkt pf*  
**shows** *borel-predict-stoch-proc F (remaining-qty Mkt v pf asset)  $\langle$ proof $\rangle$*

**lemma** (in *disc-equity-market*) *remaining-qty-adapt*:  
**assumes** *borel-adapt-stoch-proc F (prices Mkt asset)*  
**and** *trading-strategy pf*  
**and** *support-adapt Mkt pf*  
**shows** *remaining-qty Mkt v pf asset n  $\in$  borel-measurable (F n)*  
 *$\langle$ proof $\rangle$*

**lemma** (in *disc-equity-market*) *remaining-qty-adapted*:  
**assumes** *borel-adapt-stoch-proc F (prices Mkt asset)*  
**and** *trading-strategy pf*  
**and** *support-adapt Mkt pf*  
**shows** *borel-adapt-stoch-proc F (remaining-qty Mkt v pf asset)  $\langle$ proof $\rangle$*

**definition** *self-finance where*  
*self-finance Mkt v pf (asset::'a) = qty-sum pf (qty-single asset (remaining-qty Mkt v pf asset))*

**lemma** *self-finance-portfolio*:  
**assumes** *portfolio pf*  
**shows** *portfolio (self-finance Mkt v pf asset)  $\langle$ proof $\rangle$*

**lemma** *self-finance-init*:  
**assumes**  $\forall w. prices\ Mkt\ asset\ 0\ w \neq 0$   
**and** *portfolio pf*  
**shows** *val-process Mkt (self-finance Mkt v pf asset) 0 w = v*

$\langle proof \rangle$

**lemma** *self-finance-succ*:

**assumes** *prices Mkt asset (Suc n) w  $\neq$  0*

**and** *portfolio pf*

**shows** *val-process Mkt (self-finance Mkt v pf asset) (Suc n) w = prices Mkt asset (Suc n) w \* remaining-qty Mkt v pf asset (Suc n) w + cls-val-process Mkt pf (Suc n) w*

$\langle proof \rangle$

**lemma** *self-finance-updated*:

**assumes** *prices Mkt asset (Suc n) w  $\neq$  0*

**and** *portfolio pf*

**shows** *cls-val-process Mkt (self-finance Mkt v pf asset) (Suc n) w = cls-val-process Mkt pf (Suc n) w + prices Mkt asset (Suc n) w \* (remaining-qty Mkt v pf asset) (Suc n) w*

$\langle proof \rangle$

**lemma** *self-finance-charact*:

**assumes**  $\forall n w. \text{prices Mkt asset (Suc n) w} \neq 0$

**and** *portfolio pf*

**shows** *self-financing Mkt (self-finance Mkt v pf asset)*

$\langle proof \rangle$

### 7.2.5 Replicating portfolios

**definition** (in *disc-filtr-prob-space*) *price-structure::('a  $\Rightarrow$  real)  $\Rightarrow$  nat  $\Rightarrow$  real  $\Rightarrow$  (nat  $\Rightarrow$  'a  $\Rightarrow$  real)  $\Rightarrow$  bool* **where**

*price-structure pyf T  $\pi$  pr  $\longleftrightarrow$  (( $\forall w \in \text{space } M. \text{pr } 0 \text{ } w = \pi$ )  $\wedge$  (AE w in M. pr T w = pyf w)  $\wedge$  (pr T  $\in$  borel-measurable (F T)))*

**lemma** (in *disc-filtr-prob-space*) *price-structure-init*:

**assumes** *price-structure pyf T  $\pi$  pr*

**shows**  $\forall w \in \text{space } M. \text{pr } 0 \text{ } w = \pi$   $\langle proof \rangle$

**lemma** (in *disc-filtr-prob-space*) *price-structure-borel-measurable*:

**assumes** *price-structure pyf T  $\pi$  pr*

**shows** *pr T  $\in$  borel-measurable (F T)*  $\langle proof \rangle$

**lemma** (in *disc-filtr-prob-space*) *price-structure-maturity*:

**assumes** *price-structure pyf T  $\pi$  pr*

**shows** *AE w in M. pr T w = pyf w*  $\langle proof \rangle$

**definition** (in *disc-equity-market*) *replicating-portfolio* **where**

*replicating-portfolio pf der matur  $\longleftrightarrow$  (stock-portfolio Mkt pf)  $\wedge$  (trading-strategy pf)  $\wedge$  (self-financing Mkt pf)  $\wedge$*

*(AE w in M. cls-val-process Mkt pf matur w = der w)*

**definition** (in *disc-equity-market*) *is-attainable* **where**  
*is-attainable der matur*  $\longleftrightarrow (\exists \text{ pf. replicating-portfolio pf der matur})$

**lemma** (in *disc-equity-market*) *replicating-price-process*:  
**assumes** *replicating-portfolio pf der matur*  
**and** *support-adapt Mkt pf*  
**shows** *price-structure der matur (initial-value pf) (cls-val-process Mkt pf)*  
*<proof>*

### 7.2.6 Arbitrages

**definition** (in *disc-filtr-prob-space*) *arbitrage-process*  
**where**  
*arbitrage-process Mkt p*  $\longleftrightarrow (\exists m. (\text{self-financing Mkt p}) \wedge (\text{trading-strategy p})$   
 $\wedge$   
 $(\forall w \in \text{space } M. \text{val-process Mkt p } 0 \ w = 0) \wedge$   
 $(AE \ w \ \text{in } M. 0 \leq \text{cls-val-process Mkt p } m \ w) \wedge$   
 $0 < \mathcal{P}(w \ \text{in } M. \text{cls-val-process Mkt p } m \ w > 0))$

**lemma** (in *disc-filtr-prob-space*) *arbitrage-processE*:  
**assumes** *arbitrage-process Mkt p*  
**shows**  $(\exists m. (\text{self-financing Mkt p}) \wedge (\text{trading-strategy p}) \wedge$   
 $(\forall w \in \text{space } M. \text{cls-val-process Mkt p } 0 \ w = 0) \wedge$   
 $(AE \ w \ \text{in } M. 0 \leq \text{cls-val-process Mkt p } m \ w) \wedge$   
 $0 < \mathcal{P}(w \ \text{in } M. \text{cls-val-process Mkt p } m \ w > 0))$   
*<proof>*

**lemma** (in *disc-filtr-prob-space*) *arbitrage-processI*:  
**assumes**  $(\exists m. (\text{self-financing Mkt p}) \wedge (\text{trading-strategy p}) \wedge$   
 $(\forall w \in \text{space } M. \text{cls-val-process Mkt p } 0 \ w = 0) \wedge$   
 $(AE \ w \ \text{in } M. 0 \leq \text{cls-val-process Mkt p } m \ w) \wedge$   
 $0 < \mathcal{P}(w \ \text{in } M. \text{cls-val-process Mkt p } m \ w > 0))$   
**shows** *arbitrage-process Mkt p* *<proof>*

**definition** (in *disc-filtr-prob-space*) *viable-market*  
**where**  
*viable-market Mkt*  $\longleftrightarrow (\forall p. \text{stock-portfolio Mkt p} \longrightarrow \neg \text{arbitrage-process Mkt p})$

**lemma** (in *disc-filtr-prob-space*) *arbitrage-val-process*:  
**assumes** *arbitrage-process Mkt pf1*  
**and** *self-financing Mkt pf2*  
**and** *trading-strategy pf2*  
**and**  $\forall n \ w. \text{cls-val-process Mkt pf1 } n \ w = \text{cls-val-process Mkt pf2 } n \ w$   
**shows** *arbitrage-process Mkt pf2*

*<proof>*

**definition** *coincides-on where*

*coincides-on Mkt Mkt2 A*  $\longleftrightarrow$  (*stocks Mkt* = *stocks Mkt2*  $\wedge$  ( $\forall x. x \in A \longrightarrow$  *prices Mkt*  $x$  = *prices Mkt2*  $x$ ))

**lemma** *coincides-val-process:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set pf*  $\subseteq A$

**shows**  $\forall n w. \text{val-process } Mkt \text{ pf } n \ w = \text{val-process } Mkt2 \text{ pf } n \ w$

*<proof>*

**lemma** *coincides-cls-val-process':*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set pf*  $\subseteq A$

**shows**  $\forall n w. \text{cls-val-process } Mkt \text{ pf } (Suc \ n) \ w = \text{cls-val-process } Mkt2 \text{ pf } (Suc \ n) \ w$

*w*

*<proof>*

**lemma** *coincides-cls-val-process:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set pf*  $\subseteq A$

**shows**  $\forall n w. \text{cls-val-process } Mkt \text{ pf } n \ w = \text{cls-val-process } Mkt2 \text{ pf } n \ w$

*<proof>*

**lemma** (**in** *disc-filtr-prob-space*) *coincides-on-self-financing:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set p*  $\subseteq A$

**and** *self-financing Mkt p*

**shows** *self-financing Mkt2 p*

*<proof>*

**lemma** (**in** *disc-filtr-prob-space*) *coincides-on-arbitrage:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set p*  $\subseteq A$

**and** *arbitrage-process Mkt p*

**shows** *arbitrage-process Mkt2 p*

*<proof>*

**lemma** (**in** *disc-filtr-prob-space*) *coincides-on-stocks-viable:*

**assumes** *coincides-on Mkt Mkt2 (stocks Mkt)*

**and** *viable-market Mkt*

**shows** *viable-market Mkt2* *<proof>*

**lemma** *coincides-stocks-val-process:*

**assumes** *stock-portfolio Mkt pf*

**and** *coincides-on Mkt Mkt2 (stocks Mkt)*

**shows**  $\forall n w. \text{val-process } Mkt \text{ pf } n w = \text{val-process } Mkt2 \text{ pf } n w$  *<proof>*

**lemma** *coincides-stocks-cls-val-process:*

**assumes** *stock-portfolio Mkt pf*

**and** *coincides-on Mkt Mkt2 (stocks Mkt)*

**shows**  $\forall n w. \text{cls-val-process } Mkt \text{ pf } n w = \text{cls-val-process } Mkt2 \text{ pf } n w$  *<proof>*

**lemma** (**in** *disc-filtr-prob-space*) *coincides-on-adapted-val-process:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set p  $\subseteq$  A*

**and** *borel-adapt-stoch-proc F (val-process Mkt p)*

**shows** *borel-adapt-stoch-proc F (val-process Mkt2 p) <proof>*

**lemma** (**in** *disc-filtr-prob-space*) *coincides-on-adapted-cls-val-process:*

**assumes** *coincides-on Mkt Mkt2 A*

**and** *support-set p  $\subseteq$  A*

**and** *borel-adapt-stoch-proc F (cls-val-process Mkt p)*

**shows** *borel-adapt-stoch-proc F (cls-val-process Mkt2 p) <proof>*

## 7.2.7 Fair prices

**definition** (**in** *disc-filtr-prob-space*) *fair-price where*

*fair-price Mkt  $\pi$  pyf matur  $\longleftrightarrow$*

*( $\exists$  pr. price-structure pyf matur  $\pi$  pr  $\wedge$*

*( $\forall$  x Mkt2 p. (x  $\notin$  stocks Mkt  $\longrightarrow$*

*((coincides-on Mkt Mkt2 (stocks Mkt))  $\wedge$  (prices Mkt2 x = pr)  $\wedge$  portfolio p*

*$\wedge$  support-set p  $\subseteq$  stocks Mkt  $\cup$  {x}  $\longrightarrow$*

*$\neg$  arbitrage-process Mkt2 p))))*

**lemma** (**in** *disc-filtr-prob-space*) *fair-priceI:*

**assumes** *fair-price Mkt  $\pi$  pyf matur*

**shows** *( $\exists$  pr. price-structure pyf matur  $\pi$  pr  $\wedge$*

*( $\forall$  x. (x  $\notin$  stocks Mkt  $\longrightarrow$*

*( $\forall$  Mkt2 p. (coincides-on Mkt Mkt2 (stocks Mkt))  $\wedge$  (prices Mkt2 x = pr)  $\wedge$*

*portfolio p  $\wedge$  support-set p  $\subseteq$  stocks Mkt  $\cup$  {x}  $\longrightarrow$*

*$\neg$  arbitrage-process Mkt2 p))))* *<proof>*

**Existence when replicating portfolio** **lemma** (**in** *disc-equity-market*) *replicating-fair-price:*

**assumes** *viable-market Mkt*

**and** *replicating-portfolio pf der matur*

**and** *support-adapt Mkt pf*

**shows** *fair-price Mkt (initial-value pf) der matur*

*<proof>*



**Uniqueness when replicating portfolio** The proof of uniqueness requires the existence of a stock that always takes strictly positive values.

**locale** *disc-market-pos-stock* = *disc-equity-market* +  
**fixes** *pos-stock*  
**assumes** *in-stock*: *pos-stock*  $\in$  *stocks Mkt*  
**and** *positive*:  $\forall n w. \text{prices } Mkt \text{ pos-stock } n w > 0$   
**and** *readable*:  $\forall \text{asset} \in \text{stocks } Mkt. \text{borel-adapt-stoch-proc } F (\text{prices } Mkt \text{ asset})$

**lemma** (**in** *disc-market-pos-stock*) *pos-stock-borel-adapted*:  
**shows** *borel-adapt-stoch-proc* *F* (*prices Mkt pos-stock*)  
 $\langle \text{proof} \rangle$

**definition** *static-quantities where*  
*static-quantities*  $p \longleftrightarrow (\forall \text{asset} \in \text{support-set } p. \exists c::\text{real}. p \text{ asset} = (\lambda n w. c))$

**lemma** (**in** *disc-filtr-prob-space*) *static-quantities-trading-strat*:  
**assumes** *static-quantities*  $p$   
**and** *finite* (*support-set*  $p$ )  
**shows** *trading-strategy*  $p \langle \text{proof} \rangle$

**lemma** *two-component-support-set*:  
**assumes**  $\exists n w. a n w \neq 0$   
**and**  $\exists n w. b n w \neq 0$   
**and**  $x \neq y$   
**shows** *support-set*  $((\lambda (x::'b) (n::\text{nat}) (w::'a). 0::\text{real})(x:= a, y:= b)) = \{x, y\}$   
 $\langle \text{proof} \rangle$

**lemma** *two-component-val-process*:  
**assumes** *arb-pf* =  $((\lambda (x::'b) (n::\text{nat}) (w::'a). 0::\text{real})(x:= a, y:= b))$   
**and** *portfolio* *arb-pf*  
**and**  $x \neq y$   
**and**  $\exists n w. a n w \neq 0$   
**and**  $\exists n w. b n w \neq 0$   
**shows** *val-process* *Mkt* *arb-pf*  $n w =$   
 $\text{prices } Mkt y n w * b (\text{Suc } n) w + \text{prices } Mkt x n w * a (\text{Suc } n) w$   
 $\langle \text{proof} \rangle$

**lemma** *quantity-update-support-set*:  
**assumes**  $\exists n w. pr n w \neq 0$   
**and**  $x \notin \text{support-set } p$   
**shows** *support-set*  $(p(x:=pr)) = \text{support-set } p \cup \{x\}$   
 $\langle \text{proof} \rangle$

**lemma** *fix-asset-price*:  
**shows**  $\exists x \text{ Mkt2}. x \notin \text{stocks Mkt} \wedge$   
*coincides-on Mkt Mkt2 (stocks Mkt)  $\wedge$*   
*prices Mkt2 x = pr*  
*<proof>*

**lemma** (*in disc-market-pos-stock*) *arbitrage-portfolio-properties*:  
**assumes** *price-structure der matur  $\pi$  pr*  
**and** *replicating-portfolio pf der matur*  
**and** (*coincides-on Mkt Mkt2 (stocks Mkt)*)  
**and** (*prices Mkt2 x = pr*)  
**and** *x  $\notin$  stocks Mkt*  
**and** *diff-inv = ( $\pi - \text{initial-value pf}$ ) / constant-image (prices Mkt pos-stock 0)*  
**and** *diff-inv  $\neq$  0*  
**and** *arb-pf = ( $\lambda (x::'b) (n::\text{nat}) (w::'a). 0::\text{real})(x := (\lambda n w. -1), \text{pos-stock} :=$*   
*( $\lambda n w. \text{diff-inv}$ ))*  
**and** *contr-pf = qty-sum arb-pf pf*  
**shows** *self-financing Mkt2 contr-pf*  
**and** *trading-strategy contr-pf*  
**and**  $\forall w \in \text{space } M. \text{cls-val-process Mkt2 contr-pf } 0 \ w = 0$   
**and**  $0 < \text{diff-inv} \longrightarrow (\text{AE } w \text{ in } M. 0 < \text{cls-val-process Mkt2 contr-pf matur } w)$   
**and**  $\text{diff-inv} < 0 \longrightarrow (\text{AE } w \text{ in } M. 0 > \text{cls-val-process Mkt2 contr-pf matur } w)$   
**and** *support-set arb-pf = {x, pos-stock}*  
**and** *portfolio contr-pf*  
*<proof>*

**lemma** (*in disc-equity-market*) *mult-comp-cls-val-process-measurable'*:  
**assumes** *cls-val-process Mkt2 pf n  $\in$  borel-measurable (F n)*  
**and** *portfolio pf*  
**and** *qty n  $\in$  borel-measurable (F n)*  
**and**  $0 \neq n$   
**shows** *cls-val-process Mkt2 (qty-mult-comp pf qty) n  $\in$  borel-measurable (F n)*  
*<proof>*

**lemma** (*in disc-equity-market*) *mult-comp-cls-val-process-measurable*:  
**assumes**  $\forall n. \text{cls-val-process Mkt2 pf } n \in \text{borel-measurable (F } n)$   
**and** *portfolio pf*  
**and**  $\forall n. \text{qty (Suc } n) \in \text{borel-measurable (F } n)$   
**shows**  $\forall n. \text{cls-val-process Mkt2 (qty-mult-comp pf qty) } n \in \text{borel-measurable (F } n)$   
*<proof>*

**lemma** (in *disc-equity-market*) *mult-comp-val-process-measurable*:  
**assumes** *val-process Mkt2 pf n ∈ borel-measurable (F n)*  
**and** *portfolio pf*  
**and** *qty (Suc n) ∈ borel-measurable (F n)*  
**shows** *val-process Mkt2 (qty-mult-comp pf qty) n ∈ borel-measurable (F n)*  
 ⟨*proof*⟩

**lemma** (in *disc-market-pos-stock*) *repl-fair-price-unique*:  
**assumes** *replicating-portfolio pf der matur*  
**and** *fair-price Mkt π der matur*  
**shows**  $\pi = \text{initial-value } pf$   
 ⟨*proof*⟩

## 7.3 Risk-neutral probability space

### 7.3.1 risk-free rate and discount factor processes

**fun** *disc-rfr-proc*::  $real \Rightarrow nat \Rightarrow 'a \Rightarrow real$   
**where**  
*rfr-base*:  $(disc-rfr-proc\ r)\ 0\ w = 1$   
*rfr-step*:  $(disc-rfr-proc\ r)\ (Suc\ n)\ w = (1+r) * (disc-rfr-proc\ r)\ n\ w$

**lemma** *disc-rfr-proc-borel-measurable*:  
**shows**  $(disc-rfr-proc\ r)\ n \in borel-measurable\ M$   
 ⟨*proof*⟩

**lemma** *disc-rfr-proc-nonrandom*:  
**fixes**  $r::real$   
**shows**  $\bigwedge n. disc-rfr-proc\ r\ n \in borel-measurable\ (F\ 0)$  ⟨*proof*⟩

**lemma** (in *disc-equity-market*) *disc-rfr-constant-time*:  
**shows**  $\exists c. \forall w \in space\ (F\ 0). (disc-rfr-proc\ r\ n)\ w = c$   
 ⟨*proof*⟩

**lemma** (in *disc-filtr-prob-space*) *disc-rfr-proc-borel-adapted*:  
**shows** *borel-adapt-stoch-proc F (disc-rfr-proc r)*  
 ⟨*proof*⟩

**lemma** *disc-rfr-proc-positive*:  
**assumes**  $-1 < r$   
**shows**  $\bigwedge n\ w. 0 < disc-rfr-proc\ r\ n\ w$   
 ⟨*proof*⟩

**lemma** (in *prob-space*) *disc-rfr-constant-time-pos*:  
**assumes**  $-1 < r$   
**shows**  $\exists c > 0. \forall w \in \text{space } M. (\text{disc-rfr-proc } r \ n) \ w = c$   
*<proof>*

**lemma** *disc-rfr-proc-Suc-div*:  
**assumes**  $-1 < r$   
**shows**  $\bigwedge w. \text{disc-rfr-proc } r \ (\text{Suc } n) \ w / \text{disc-rfr-proc } r \ n \ w = 1+r$   
*<proof>*

**definition** *discount-factor where*  
 $\text{discount-factor } r \ n = (\lambda w. \text{inverse } (\text{disc-rfr-proc } r \ n \ w))$

**lemma** *discount-factor-times-rfr*:  
**assumes**  $-1 < r$   
**shows**  $(1+r) * \text{discount-factor } r \ (\text{Suc } n) \ w = \text{discount-factor } r \ n \ w$  *<proof>*

**lemma** *discount-factor-borel-measurable*:  
**shows**  $\text{discount-factor } r \ n \in \text{borel-measurable } M$  *<proof>*

**lemma** *discount-factor-init*:  
**shows**  $\text{discount-factor } r \ 0 = (\lambda w. 1)$  *<proof>*

**lemma** *discount-factor-nonrandom*:  
**shows**  $\text{discount-factor } r \ n \in \text{borel-measurable } M$  *<proof>*

**lemma** *discount-factor-positive*:  
**assumes**  $-1 < r$   
**shows**  $\bigwedge n \ w. 0 < \text{discount-factor } r \ n \ w$  *<proof>*

**lemma** (in *prob-space*) *discount-factor-constant-time-pos*:  
**assumes**  $-1 < r$   
**shows**  $\exists c > 0. \forall w \in \text{space } M. (\text{discount-factor } r \ n) \ w = c$  *<proof>*

**locale** *rsk-free-asset* =  
**fixes** *Mkt r risk-free-asset*  
**assumes** *acceptable-rate*:  $-1 < r$   
**and** *rf-price*:  $\text{prices } Mkt \ \text{risk-free-asset} = \text{disc-rfr-proc } r$   
**and** *rf-stock*:  $\text{risk-free-asset} \in \text{stocks } Mkt$

**locale** *rfr-disc-equity-market* = *disc-equity-market* + *rsk-free-asset* +  
**assumes** *rd*:  $\forall \text{asset} \in \text{stocks } Mkt. \text{borel-adapt-stoch-proc } F \ (\text{prices } Mkt \ \text{asset})$

**sublocale** *rfr-disc-equity-market*  $\subseteq$  *disc-market-pos-stock* - - - *risk-free-asset*  
 ⟨proof⟩

### 7.3.2 Discounted value of a stochastic process

**definition** *discounted-value* **where**

*discounted-value*  $r X = (\lambda n w. \text{discount-factor } r n w * X n w)$

**lemma** (in *rfr-disc-equity-market*) *discounted-rfr*:

**shows** *discounted-value*  $r$  (*prices Mkt risk-free-asset*)  $n w = 1$  ⟨proof⟩

**lemma** *discounted-init*:

**shows**  $\forall w. \text{discounted-value } r X 0 w = X 0 w$  ⟨proof⟩

**lemma** *discounted-mult*:

**shows**  $\forall n w. \text{discounted-value } r (\lambda m x. X m x * Y m x) n w = X n w * (\text{discounted-value } r Y) n w$   
 ⟨proof⟩

**lemma** *discounted-mult'*:

**shows** *discounted-value*  $r (\lambda m x. X m x * Y m x) n w = X n w * (\text{discounted-value } r Y) n w$   
 ⟨proof⟩

**lemma** *discounted-mult-times-rfr*:

**assumes**  $-1 < r$

**shows** *discounted-value*  $r (\lambda m w. (1+r) * X w) (Suc n) w = \text{discounted-value } r (\lambda m w. X w) n w$   
 ⟨proof⟩

**lemma** *discounted-cong*:

**assumes**  $\forall n w. X n w = Y n w$

**shows**  $\forall n w. \text{discounted-value } r X n w = \text{discounted-value } r Y n w$

⟨proof⟩

**lemma** *discounted-cong'*:

**assumes**  $X n w = Y n w$

**shows** *discounted-value*  $r X n w = \text{discounted-value } r Y n w$

⟨proof⟩

**lemma** *discounted-AE-cong*:

**assumes** *AE*  $w$  in  $N$ .  $X n w = Y n w$

**shows** *AE*  $w$  in  $N$ . *discounted-value*  $r X n w = \text{discounted-value } r Y n w$   
 ⟨proof⟩

**lemma** *discounted-sum*:

**assumes** *finite I*

**shows**  $\forall n w. (\sum_{i \in I}. (\text{discounted-value } r (\lambda m x. f i m x)) n w) = (\text{discounted-value } r (\lambda m x. (\sum_{i \in I}. f i m x)) n w)$   
*<proof>*

**lemma** *discounted-adapted*:

**assumes** *borel-adapt-stoch-proc F X*

**shows** *borel-adapt-stoch-proc F (discounted-value r X) <proof>*

**lemma** *discounted-measurable*:

**assumes** *X ∈ borel-measurable N*

**shows** *discounted-value r (λm. X) m ∈ borel-measurable N <proof>*

**lemma** (*in prob-space*) *discounted-integrable*:

**assumes** *integrable N (X n)*

**and**  $-1 < r$

**and** *space N = space M*

**shows** *integrable N (discounted-value r X n) <proof>*

### 7.3.3 Results on risk-neutral probability spaces

**definition** (*in rfr-disc-equity-market*) *risk-neutral-prob* **where**

*risk-neutral-prob N*  $\longleftrightarrow$  (*prob-space N*)  $\wedge$  ( $\forall$  *asset*  $\in$  *stocks Mkt. martingale N* *F (discounted-value r (prices Mkt asset))*)

**lemma** *integrable-val-process*:

**assumes**  $\forall$  *asset*  $\in$  *support-set pf. integrable M (λw. prices Mkt asset n w \* pf asset (Suc n) w)*

**shows** *integrable M (val-process Mkt pf n)*

*<proof>*

**lemma** *integrable-self-fin-uwp*:

**assumes**  $\forall$  *asset*  $\in$  *support-set pf. integrable M (λw. prices Mkt asset n w \* pf asset (Suc n) w)*

**and** *self-financing Mkt pf*

**shows** *integrable M (cls-val-process Mkt pf n)*

*<proof>*

**lemma** (*in rfr-disc-equity-market*) *stocks-portfolio-risk-neutral*:

**assumes** *risk-neutral-prob N*

**and** *trading-strategy pf*

**and** *subalgebra*  $N M$   
**and** *support-set*  $pf \subseteq \text{stocks } Mkt$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{integrable } N (\lambda w. \text{prices } Mkt \text{ asset } (Suc\ n) w$   
 $* \text{pf asset } (Suc\ n) w)$   
**shows**  $\forall x \in \text{support-set } pf. AE\ w\ \text{in } N.$   
 $(\text{real-cond-exp } N (F\ n) (\text{discounted-value } r (\lambda m\ y. \text{prices } Mkt\ x\ m\ y * \text{pf } x$   
 $m\ y) (Suc\ n))) w =$   
 $\text{discounted-value } r (\lambda m\ y. \text{prices } Mkt\ x\ m\ y * \text{pf } x (Suc\ m) y) n w$   
 $\langle \text{proof} \rangle$

**lemma** (in *rfr-disc-equity-market*) *self-fin-trad-strat-mart*:

**assumes** *risk-neutral-prob*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *trading-strategy*  $pf$   
**and** *self-financing*  $Mkt\ pf$   
**and** *stock-portfolio*  $Mkt\ pf$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{integrable } N (\lambda w. \text{prices } Mkt \text{ asset } n\ w * \text{pf}$   
 $\text{asset } (Suc\ n) w)$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{integrable } N (\lambda w. \text{prices } Mkt \text{ asset } (Suc\ n) w$   
 $* \text{pf asset } (Suc\ n) w)$   
**shows** *martingale*  $N F (\text{discounted-value } r (\text{cls-val-process } Mkt\ pf))$   
 $\langle \text{proof} \rangle$

**lemma** (in *disc-filtr-prob-space*) *finite-integrable-vp*:

**assumes**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{finite } (\text{prices } Mkt \text{ asset } n \text{ '}(space\ M))$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{finite } (\text{pf asset } n \text{ '}(space\ M))$   
**and** *prob-space*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *trading-strategy*  $pf$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{prices } Mkt \text{ asset } n \in \text{borel-measurable } M$   
**shows**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{integrable } N (\lambda w. \text{prices } Mkt \text{ asset } n\ w * \text{pf}$   
 $\text{asset } (Suc\ n) w)$   
 $\langle \text{proof} \rangle$

**lemma** (in *disc-filtr-prob-space*) *finite-integrable-uvp*:

**assumes**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{finite } (\text{prices } Mkt \text{ asset } n \text{ '}(space\ M))$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{finite } (\text{pf asset } n \text{ '}(space\ M))$   
**and** *prob-space*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *trading-strategy*  $pf$   
**and**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{prices } Mkt \text{ asset } n \in \text{borel-measurable } M$   
**shows**  $\forall n. \forall \text{asset} \in \text{support-set } pf. \text{integrable } N (\lambda w. \text{prices } Mkt \text{ asset } (Suc\ n) w$   
 $* \text{pf asset } (Suc\ n) w)$   
 $\langle \text{proof} \rangle$

**lemma** (in *rfr-disc-equity-market*) *self-fin-trad-strat-mart-finite*:

**assumes** *risk-neutral-prob*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *trading-strategy*  $pf$   
**and** *self-financing*  $Mkt pf$   
**and** *support-set*  $pf \subseteq stocks Mkt$   
**and**  $\forall n. \forall asset \in support-set pf. finite (prices Mkt asset n \text{ '(space } M))$   
**and**  $\forall n. \forall asset \in support-set pf. finite (pf asset n \text{ '(space } M))$   
**and**  $\forall asset \in stocks Mkt. borel-adapt-stoch-proc F (prices Mkt asset)$   
**shows** *martingale*  $N F (discounted-value r (cls-val-process Mkt pf))$   
 $\langle proof \rangle$

**lemma** (in *rfr-disc-equity-market*) *replicating-expectation*:

**assumes** *risk-neutral-prob*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *replicating-portfolio*  $pf pyf matur$   
**and**  $\forall n. \forall asset \in support-set pf. integrable N (\lambda w. prices Mkt asset n w * pf asset (Suc n) w)$   
**and**  $\forall n. \forall asset \in support-set pf. integrable N (\lambda w. prices Mkt asset (Suc n) w * pf asset (Suc n) w)$   
**and** *viable-market*  $Mkt$   
**and** *sets*  $(F 0) = \{\{\}, space M\}$   
**and**  $pyf \in borel-measurable (F matur)$   
**shows** *fair-price*  $Mkt (prob-space.expectation N (discounted-value r (\lambda m. pyf) matur))$   
 $pyf matur$   
 $\langle proof \rangle$

**lemma** (in *rfr-disc-equity-market*) *replicating-expectation-finite*:

**assumes** *risk-neutral-prob*  $N$   
**and** *filt-equiv*  $F M N$   
**and** *replicating-portfolio*  $pf pyf matur$   
**and**  $\forall n. \forall asset \in support-set pf. finite (prices Mkt asset n \text{ '(space } M))$   
**and**  $\forall n. \forall asset \in support-set pf. finite (pf asset n \text{ '(space } M))$   
**and** *viable-market*  $Mkt$   
**and** *sets*  $(F 0) = \{\{\}, space M\}$   
**and**  $pyf \in borel-measurable (F matur)$   
**shows** *fair-price*  $Mkt (prob-space.expectation N (discounted-value r (\lambda m. pyf) matur))$   
 $pyf matur$   
 $\langle proof \rangle$

**end**



## 8 The Cox Ross Rubinstein model

This section defines the Cox-Ross-Rubinstein model of a financial market, and characterizes a risk-neutral probability space for this market. This, together with the proof that every derivative is attainable, permits to obtain a formula to explicitly compute the fair price of any derivative.

**theory** *CRR-Model* **imports** *Fair-Price*

**begin**

**locale** *CRR-hyps* = *prob-grw* + *rsk-free-asset* +  
  **fixes** *stk*  
**assumes** *stocks*: *stocks Mkt* = {*stk*, *risk-free-asset*}  
  **and** *stk-price*: *prices Mkt stk* = *geom-proc*  
  **and** *S0-positive*:  $0 < \text{init}$   
  **and** *down-positive*:  $0 < d$  **and** *down-lt-up*:  $d < u$   
  **and** *psgt*:  $0 < p$   
  **and** *pslt*:  $p < 1$

**locale** *CRR-market* = *CRR-hyps* +  
  **fixes** *G*  
**assumes** *stock-filtration*: *G* = *stoch-proc-filt M geom-proc borel*

### 8.1 Preliminary results on the market

**lemma** (in *CRR-market*) *case-asset*:  
  **assumes** *asset*  $\in$  *stocks Mkt*  
  **shows** *asset* = *stk*  $\vee$  *asset* = *risk-free-asset*  
*<proof>*

**lemma** (in *CRR-market*)  
  **assumes** *N* = *bernoulli-stream q*  
  **and**  $0 < q$   
  **and**  $q < 1$   
  **shows** *bernoulli-gen-filtration*: *filtration N G*  
  **and** *bernoulli-sigma-finite*:  $\forall n.$  *sigma-finite-subalgebra N (G n)*  
*<proof>*

**sublocale** *CRR-market*  $\subseteq$  *rfr-disc-equity-market - G*  
*<proof>*

**lemma** (in *CRR-market*) *two-stocks*:  
**shows** *stk*  $\neq$  *risk-free-asset*  
*<proof>*

**lemma** (in *CRR-market*) *stock-pf-vp-expand*:

**assumes** *stock-portfolio Mkt pf*

**shows**  $\text{val-process } Mkt \text{ pf } n \ w = \text{geom-proc } n \ w * \text{pf } stk \ (Suc \ n) \ w +$   
 $\text{disc-rfr-proc } r \ n \ w * \text{pf } \text{risk-free-asset} \ (Suc \ n) \ w$

*<proof>*

**lemma** (in *CRR-market*) *stock-pf-uvp-expand*:

**assumes** *stock-portfolio Mkt pf*

**shows**  $\text{cls-val-process } Mkt \text{ pf } \ (Suc \ n) \ w = \text{geom-proc} \ (Suc \ n) \ w * \text{pf } stk \ (Suc \ n)$   
 $w +$

$\text{disc-rfr-proc } r \ (Suc \ n) \ w * \text{pf } \text{risk-free-asset} \ (Suc \ n) \ w$

*<proof>*

**lemma** (in *CRR-market*) *pos-pf-neg-uwp*:

**assumes** *stock-portfolio Mkt pf*

**and**  $d < 1+r$

**and**  $0 < \text{pf } stk \ (Suc \ n) \ (\text{spick } w \ n \ False)$

**and**  $\text{val-process } Mkt \text{ pf } n \ (\text{spick } w \ n \ False) \leq 0$

**shows**  $\text{cls-val-process } Mkt \text{ pf } \ (Suc \ n) \ (\text{spick } w \ n \ False) < 0$

*<proof>*

**lemma** (in *CRR-market*) *neg-pf-neg-uwp*:

**assumes** *stock-portfolio Mkt pf*

**and**  $1+r < u$

**and**  $\text{pf } stk \ (Suc \ n) \ (\text{spick } w \ n \ True) < 0$

**and**  $\text{val-process } Mkt \text{ pf } n \ (\text{spick } w \ n \ True) \leq 0$

**shows**  $\text{cls-val-process } Mkt \text{ pf } \ (Suc \ n) \ (\text{spick } w \ n \ True) < 0$

*<proof>*

**lemma** (in *CRR-market*) *zero-pf-neg-uwp*:

**assumes** *stock-portfolio Mkt pf*

**and**  $\text{pf } stk \ (Suc \ n) \ w = 0$

**and**  $\text{pf } \text{risk-free-asset} \ (Suc \ n) \ w \neq 0$

**and**  $\text{val-process } Mkt \text{ pf } n \ w \leq 0$

**shows**  $\text{cls-val-process } Mkt \text{ pf } \ (Suc \ n) \ w < 0$

*<proof>*

**lemma** (in *CRR-market*) *neg-pf-exists*:

**assumes** *stock-portfolio Mkt pf*

**and** *trading-strategy pf*  
**and**  $1+r < u$   
**and**  $d < 1+r$   
**and** *val-process Mkt pf n w  $\leq 0$*   
**and** *pf stk (Suc n) w  $\neq 0 \vee$  pf risk-free-asset (Suc n) w  $\neq 0$*   
**shows**  $\exists y. \text{cls-val-process Mkt pf (Suc n) } y < 0$   
*<proof>*

**lemma (in CRR-market) non-zero-components:**  
**assumes** *val-process Mkt pf n y  $\neq 0$*   
**and** *stock-portfolio Mkt pf*  
**shows** *pf stk (Suc n) y  $\neq 0 \vee$  pf risk-free-asset (Suc n) y  $\neq 0$*   
*<proof>*

**lemma (in CRR-market) neg-pf-Suc:**  
**assumes** *stock-portfolio Mkt pf*  
**and** *trading-strategy pf*  
**and** *self-financing Mkt pf*  
**and**  $1+r < u$   
**and**  $d < 1+r$   
**and** *cls-val-process Mkt pf n w  $< 0$*   
**shows**  $n \leq m \implies \exists y. \text{cls-val-process Mkt pf m } y < 0$   
*<proof>*

**lemma (in CRR-market) viable-if:**  
**assumes**  $1+r < u$   
**and**  $d < 1+r$   
**shows** *viable-market Mkt* *<proof>*

**lemma (in CRR-market) viable-only-if-d:**  
**assumes** *viable-market Mkt*  
**shows**  $d < 1+r$   
*<proof>*

**lemma (in CRR-market) viable-only-if-u:**  
**assumes** *viable-market Mkt*  
**shows**  $1+r < u$   
*<proof>*

**lemma (in CRR-market) viable-iff:**  
**shows** *viable-market Mkt*  $\longleftrightarrow (d < 1+r \wedge 1+r < u)$  *<proof>*

## 8.2 Risk-neutral probability space for the geometric random walk

**lemma** (in *CRR-market*) *stock-price-borel-measurable*:  
**shows** *borel-adapt-stoch-proc G (prices Mkt stk)*  
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *risk-free-asset-martingale*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *martingale N G (discounted-value r (prices Mkt risk-free-asset))*  
 ⟨*proof*⟩

**lemma** (in *infinite-coin-toss-space*) *nat-filtration-from-eq-sets*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *sets (infinite-coin-toss-space.nat-filtration N n) = sets (nat-filtration n)*  
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *geom-proc-integrable*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 \leq q$   
**and**  $q \leq 1$   
**shows** *integrable N (geom-proc n)*  
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *CRR-infinite-cts-filtration*:  
**shows** *infinite-cts-filtration p M nat-filtration*  
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *proj-stoch-proc-geom-disc-fct*:  
**shows** *disc-fct (proj-stoch-proc geom-proc n) ⟨proof⟩*

**lemma** (in *CRR-market*) *proj-stoch-proc-geom-rng*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**shows** *proj-stoch-proc geom-proc n ∈ N →<sub>M</sub> stream-space borel*  
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *proj-stoch-proc-geom-open-set*:  
**shows**  $\forall r \in \text{range (proj-stoch-proc geom-proc n)} \cap \text{space (stream-space borel)}$ .  
 $\exists A \in \text{sets (stream-space borel)}$ .  $\text{range (proj-stoch-proc geom-proc n)} \cap A = \{r\}$   
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *bernoulli-AE-cond-exp*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and** *integrable*  $N X$   
**shows** *AE*  $w$  in  $N$ . *real-cond-exp*  $N$  (*fct-gen-subalgebra*  $N$  (*stream-space borel*)  
(*proj-stoch-proc geom-proc*  $n$ ))  $X w =$   
*expl-cond-expect*  $N$  (*proj-stoch-proc geom-proc*  $n$ )  $X w$   
⟨*proof*⟩

**lemma** (in *CRR-market*) *geom-proc-cond-exp*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *AE*  $w$  in  $N$ . *real-cond-exp*  $N$  (*fct-gen-subalgebra*  $N$  (*stream-space borel*)  
(*proj-stoch-proc geom-proc*  $n$ )) (*geom-proc* (*Suc*  $n$ ))  $w =$   
*expl-cond-expect*  $N$  (*proj-stoch-proc geom-proc*  $n$ ) (*geom-proc* (*Suc*  $n$ ))  $w$   
⟨*proof*⟩

**lemma** (in *CRR-market*) *expl-cond-eq-sets*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**shows** *expl-cond-expect*  $N$  (*proj-stoch-proc geom-proc*  $n$ )  $X \in$   
*borel-measurable* (*fct-gen-subalgebra*  $N$  (*stream-space borel*) (*proj-stoch-proc*  
*geom-proc*  $n$ ))  
⟨*proof*⟩

**lemma** (in *CRR-market*) *bernoulli-real-cond-exp-AE*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and** *integrable*  $N X$   
**shows** *real-cond-exp*  $N$  (*fct-gen-subalgebra*  $N$  (*stream-space borel*) (*proj-stoch-proc*  
*geom-proc*  $n$ ))  
 $X w = \text{expl-cond-expect } N$  (*proj-stoch-proc geom-proc*  $n$ )  $X w$   
⟨*proof*⟩

**lemma** (in *CRR-market*) *geom-proc-real-cond-exp-AE*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *real-cond-exp*  $N$  (*fct-gen-subalgebra*  $N$  (*stream-space borel*) (*proj-stoch-proc*  
*geom-proc*  $n$ ))  
(*geom-proc* (*Suc*  $n$ ))  $w = \text{expl-cond-expect } N$  (*proj-stoch-proc geom-proc*  $n$ )  
(*geom-proc* (*Suc*  $n$ ))  $w$   
⟨*proof*⟩

**lemma** (in *CRR-market*) *geom-proc-stoch-proc-filt*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *stoch-proc-filt*  $N$  *geom-proc borel*  $n = \text{fct-gen-subalgebra } N$  (*stream-space borel*) (*proj-stoch-proc geom-proc*  $n$ )  
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *bernoulli-cond-exp*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and** *integrable*  $N X$   
**shows** *real-cond-exp*  $N$  (*stoch-proc-filt*  $N$  *geom-proc borel*  $n$ )  $X w = \text{expl-cond-expect}$   $N$  (*proj-stoch-proc geom-proc*  $n$ )  $X w$   
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *stock-cond-exp*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *real-cond-exp*  $N$  (*stoch-proc-filt*  $N$  *geom-proc borel*  $n$ ) (*geom-proc* (*Suc*  $n$ ))  $w = \text{expl-cond-expect}$   $N$  (*proj-stoch-proc geom-proc*  $n$ ) (*geom-proc* (*Suc*  $n$ ))  $w$   
 $\langle \text{proof} \rangle$

**lemma** (in *prob-space*) *discount-factor-real-cond-exp*:  
**assumes** *integrable*  $M X$   
**and** *subalgebra*  $M G$   
**and**  $-1 < r$   
**shows** *AE*  $w$  in  $M$ . *real-cond-exp*  $M G$  ( $\lambda x. \text{discount-factor } r \ n \ x * X \ x$ )  $w = \text{discount-factor } r \ n \ w * (\text{real-cond-exp } M G X) \ w$   
 $\langle \text{proof} \rangle$

**lemma** (in *prob-space*) *discounted-value-real-cond-exp*:  
**assumes** *integrable*  $M X$   
**and**  $-1 < r$   
**and** *subalgebra*  $M G$   
**shows** *AE*  $w$  in  $M$ . *real-cond-exp*  $M G$  ( $(\text{discounted-value } r \ (\lambda m. X)) \ n$ )  $w = \text{discounted-value } r \ (\lambda m. (\text{real-cond-exp } M G X)) \ n \ w$   $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*)  
**assumes**  $q = (1 + r - d)/(u - d)$   
**and** *viable-market*  $Mkt$

**shows** *gt-param*:  $0 < q$   
**and** *lt-param*:  $q < 1$   
**and** *risk-neutral-param*:  $u * q + d * (1 - q) = 1 + r$   
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *bernoulli-expl-cond-expect-adapt*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *expl-cond-expect*  $N$  (*proj-stoch-proc geom-proc*  $n$ )  $f \in \text{borel-measurable } (G$   
 $n)$   
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *real-cond-exp-discount-stock*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *AE*  $w$  in  $N$ . *real-cond-exp*  $N$  ( $G$   $n$ )  
 (*discounted-value*  $r$  (*prices Mkt stk*) (*Suc*  $n$ ))  $w =$   
 $\text{discounted-value } r (\lambda m w. (q * u + (1 - q) * d) * \text{prices Mkt stk } n$   
 $w) (\text{Suc } n) w$   
 ⟨*proof*⟩

**lemma** (in *CRR-market*) *risky-asset-martingale-only-if*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and** *martingale*  $N$   $G$  (*discounted-value*  $r$  (*prices Mkt stk*))  
**shows**  $q = (1 + r - d) / (u - d)$   
 ⟨*proof*⟩

**locale** *CRR-market-viable* = *CRR-market* +  
**assumes** *CRR-viable*: *viable-market*  $Mkt$

**lemma** (in *CRR-market-viable*) *real-cond-exp-discount-stock-q-const*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $q = (1 + r - d) / (u - d)$   
**shows** *AE*  $w$  in  $N$ . *real-cond-exp*  $N$  ( $G$   $n$ )  
 (*discounted-value*  $r$  (*prices Mkt stk*) (*Suc*  $n$ ))  $w =$   
 $\text{discounted-value } r (\text{prices Mkt stk } n) w$   
 ⟨*proof*⟩

**lemma** (in *CRR-market-viable*) *risky-asset-martingale-if*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $q = (1 + r - d) / (u - d)$   
**shows** *martingale*  $N$   $G$  (*discounted-value*  $r$  (*prices*  $Mkt$   $stk$ ))  
*<proof>*

**lemma** (in *CRR-market-viable*) *risk-neutral-iff'*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 \leq q$   
**and**  $q \leq 1$   
**and** *filt-equiv nat-filtration*  $M$   $N$   
**shows** *rfr-disc-equity-market.risk-neutral-prob*  $G$   $Mkt$   $r$   $N \longleftrightarrow q = (1 + r - d) / (u - d)$   
*<proof>*

**lemma** (in *CRR-market-viable*) *risk-neutral-iff*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows** *rfr-disc-equity-market.risk-neutral-prob*  $G$   $Mkt$   $r$   $N \longleftrightarrow q = (1 + r - d) / (u - d)$   
*<proof>*

### 8.3 Existence of a replicating portfolio

**fun** (in *CRR-market*) *rn-rev-price* **where**  
*rn-rev-price*  $N$  *der matur*  $0$   $w = \text{der } w \mid$   
*rn-rev-price*  $N$  *der matur*  $(Suc\ n)$   $w = \text{discount-factor } r$   $(Suc\ 0)$   $w * \text{expl-cond-expect } N$  (*proj-stoch-proc geom-proc* (*matur*  $- Suc\ n$ )) (*rn-rev-price*  $N$  *der matur*  $n$ )  $w$

**lemma** (in *CRR-market*) *stock-filtration-eq*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $G$   $n = \text{stoch-proc-filt } N$  *geom-proc borel*  $n$   
*<proof>*

**lemma** (in *CRR-market*) *real-exp-eq*:  
**assumes** *der*  $\in$  *borel-measurable* ( $G$  *matur*)



**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $\text{real-cond-exp } N \text{ (stoch-proc-filt } N \text{ geom-proc borel } n) \text{ der } w =$   
 $\text{expl-cond-expect } N \text{ (proj-stoch-proc geom-proc } n) \text{ der } w$   
 ⟨proof⟩

**lemma (in CRR-market) rn-rev-price-rev-borel-adapt:**  
**assumes**  $\text{cash-flow} \in \text{borel-measurable } (G \text{ matur})$   
**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $(n \leq \text{matur}) \implies (\text{rn-rev-price } N \text{ cash-flow matur } n) \in \text{borel-measurable } (G$   
 $(\text{matur} - n))$   
 ⟨proof⟩

**lemma (in infinite-coin-toss-space) bernoulli-discounted-integrable:**  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and**  $\text{der} \in \text{borel-measurable } (\text{nat-filtration } n)$   
**and**  $-1 < r$   
**shows**  $\text{integrable } N \text{ (discounted-value } r \text{ } (\lambda m. \text{der}) m)$   
 ⟨proof⟩

**lemma (in CRR-market) rn-rev-expl-cond-expect:**  
**assumes**  $\text{der} \in \text{borel-measurable } (G \text{ matur})$   
**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $n \leq \text{matur} \implies \text{rn-rev-price } N \text{ der matur } n \text{ } w =$   
 $\text{expl-cond-expect } N \text{ (proj-stoch-proc geom-proc } (\text{matur} - n)) \text{ (discounted-value } r$   
 $(\lambda m. \text{der}) n) \text{ } w$   
 ⟨proof⟩

**definition (in CRR-market) rn-price where**  
 $\text{rn-price } N \text{ der matur } n \text{ } w = \text{expl-cond-expect } N \text{ (proj-stoch-proc geom-proc } n)$   
 $(\text{discounted-value } r \text{ } (\lambda m. \text{der}) (\text{matur} - n)) \text{ } w$

**definition (in CRR-market) rn-price-ind where**  
 $\text{rn-price-ind } N \text{ der matur } n \text{ } w = \text{rn-rev-price } N \text{ der matur } (\text{matur} - n) \text{ } w$

**lemma (in CRR-market) rn-price-eq:**  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$

**and**  $der \in \text{borel-measurable } (G \text{ matur})$   
**and**  $n \leq \text{matur}$   
**shows**  $\text{rn-price } N \text{ der matur } n \ w = \text{rn-price-ind } N \text{ der matur } n \ w$   $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *geom-proc-filt-info*:  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } (G \ n)$   
**shows**  $f \ w = f$  (*pseudo-proj-True*  $n \ w$ )  
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *geom-proc-filt-info'*:  
**fixes**  $f::\text{bool stream} \Rightarrow 'b::\{t0\text{-space}\}$   
**assumes**  $f \in \text{borel-measurable } (G \ n)$   
**shows**  $f \ w = f$  (*pseudo-proj-False*  $n \ w$ )  
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *rn-price-borel-adapt*:  
**assumes**  $\text{cash-flow} \in \text{borel-measurable } (G \ \text{matur})$   
**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**and**  $n \leq \text{matur}$   
**shows**  $(\text{rn-price } N \ \text{cash-flow} \ \text{matur } n) \in \text{borel-measurable } (G \ n)$   
 $\langle \text{proof} \rangle$

**definition** (in *CRR-market*) *delta-price where*  
 $\text{delta-price } N \ \text{cash-flow } T =$   
 $(\lambda \ n \ w. \ \text{if } (\text{Suc } n \leq T)$   
 $\ \ \ \ \ \text{then } (\text{rn-price } N \ \text{cash-flow } T \ (\text{Suc } n) \ (\text{pseudo-proj-True } n \ w) - \text{rn-price } N$   
 $\ \ \ \ \ \text{cash-flow } T \ (\text{Suc } n) \ (\text{pseudo-proj-False } n \ w)) /$   
 $\ \ \ \ \ \ (\text{geom-proc } (\text{Suc } n) \ (\text{spick } w \ n \ \text{True}) - \text{geom-proc } (\text{Suc } n) \ (\text{spick } w \ n \ \text{False}))$   
 $\ \ \ \ \ \ \text{else } 0)$

**lemma** (in *CRR-market*) *delta-price-eq*:  
**assumes**  $\text{Suc } n \leq T$   
**shows**  $\text{delta-price } N \ \text{cash-flow } T \ n \ w = (\text{rn-price } N \ \text{cash-flow } T \ (\text{Suc } n) \ (\text{spick}$   
 $\ w \ n \ \text{True}) - \text{rn-price } N \ \text{cash-flow } T \ (\text{Suc } n) \ (\text{spick } w \ n \ \text{False})) /$   
 $((\text{geom-proc } n \ w) * (u - d))$   
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market*) *geom-proc-spick*:

**shows**  $\text{geom-proc } (\text{Suc } n) (\text{spick } w \ n \ x) = (\text{if } x \ \text{then } u \ \text{else } d) * \text{geom-proc } n \ w$   
 ⟨proof⟩

**lemma** (in *CRR-market*) *spick-red-geom*:

**shows**  $(\lambda w. \text{spick } w \ n \ x) \in \text{measurable } (\text{fct-gen-subalgebra } M \ \text{borel } (\text{geom-proc } n)) (\text{fct-gen-subalgebra } M \ \text{borel } (\text{geom-proc } (\text{Suc } n)))$   
 ⟨proof⟩

**lemma** (in *CRR-market*) *geom-spick-Suc*:

**assumes**  $A \in \{(\text{geom-proc } (\text{Suc } n)) - 'B \mid B. B \in \text{sets borel}\}$   
**shows**  $(\lambda w. \text{spick } w \ n \ x) - 'A \in \{\text{geom-proc } n - 'B \mid B. B \in \text{sets borel}\}$   
 ⟨proof⟩

**lemma** (in *CRR-market*) *geom-spick-lt*:

**assumes**  $m < n$   
**shows**  $\text{geom-proc } m (\text{spick } w \ n \ x) = \text{geom-proc } m \ w$   
 ⟨proof⟩

**lemma** (in *CRR-market*) *geom-spick-eq*:

**shows**  $\text{geom-proc } m (\text{spick } w \ m \ x) = \text{geom-proc } m \ w$   
 ⟨proof⟩

**lemma** (in *CRR-market*) *spick-red-geom-filt*:

**shows**  $(\lambda w. \text{spick } w \ n \ x) \in \text{measurable } (G \ n) (G (\text{Suc } n))$  ⟨proof⟩

**lemma** (in *CRR-market*) *delta-price-adapted*:

**fixes**  $\text{cash-flow} :: \text{bool stream} \Rightarrow \text{real}$   
**assumes**  $\text{cash-flow} \in \text{borel-measurable } (G \ T)$   
**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $\text{borel-adapt-stoch-proc } G (\text{delta-price } N \ \text{cash-flow } T)$   
 ⟨proof⟩

**fun** (in *CRR-market*) *delta-predict where*

$\text{delta-predict } N \ \text{der} \ \text{matur } 0 = (\lambda w. \text{delta-price } N \ \text{der} \ \text{matur } 0 \ w) \mid$   
 $\text{delta-predict } N \ \text{der} \ \text{matur } (\text{Suc } n) = (\lambda w. \text{delta-price } N \ \text{der} \ \text{matur } n \ w)$

**lemma** (in *CRR-market*) *delta-predict-predict*:

**assumes**  $\text{der} \in \text{borel-measurable } (G \ \text{matur})$   
**and**  $N = \text{bernoulli-stream } q$   
**and**  $0 < q$   
**and**  $q < 1$   
**shows**  $\text{borel-predict-stoch-proc } G (\text{delta-predict } N \ \text{der} \ \text{matur})$  ⟨proof⟩

**definition (in CRR-market) delta-pf where**  
*delta-pf N der matur = qty-single stk (delta-predict N der matur)*

**lemma (in CRR-market) delta-pf-support:**  
*shows support-set (delta-pf N der matur)  $\subseteq$  {stk} <proof>*

**definition (in CRR-market) self-fin-delta-pf where**  
*self-fin-delta-pf N der matur v0 = self-finance Mkt v0 (delta-pf N der matur)*  
*risk-free-asset*

**lemma (in disc-equity-market) self-finance-trading-strat:**  
*assumes trading-strategy pf*  
*and portfolio pf*  
*and borel-adapt-stoch-proc F (prices Mkt asset)*  
*and support-adapt Mkt pf*  
*shows trading-strategy (self-finance Mkt v pf asset) <proof>*

**lemma (in CRR-market) self-fin-delta-pf-trad-strat:**  
*assumes der  $\in$  borel-measurable (G matur)*  
*and N = bernoulli-stream q*  
*and  $0 < q$*   
*and  $q < 1$*   
*shows trading-strategy (self-fin-delta-pf N der matur v0) <proof>*

**definition (in CRR-market) delta-hedging where**  
*delta-hedging N der matur = self-fin-delta-pf N der matur*  
*(prob-space.expectation N (discounted-value r ( $\lambda$ m. der) matur))*

**lemma (in CRR-market) geom-proc-eq-snth:**  
*shows ( $\bigwedge m. m \leq \text{Suc } n \implies \text{geom-proc } m \ x = \text{geom-proc } m \ y$ )  $\implies$*   
*( $\bigwedge m. m \leq n \implies \text{snth } x \ m = \text{snth } y \ m$ )*  
*<proof>*

**lemma (in CRR-market) geom-proc-eq-pseudo-proj-True:**  
*shows ( $\bigwedge m. m \leq n \implies \text{geom-proc } m \ x = \text{geom-proc } m \ y$ )  $\implies$*   
*(pseudo-proj-True (n) x = pseudo-proj-True (n) y)*  
*<proof>*

**lemma (in CRR-market) proj-stoch-eq-pseudo-proj-True:**  
*assumes proj-stoch-proc geom-proc m x = proj-stoch-proc geom-proc m y*  
*shows pseudo-proj-True m x = pseudo-proj-True m y*  
*<proof>*

**lemma (in CRR-market-viable) rn-rev-price-cond-expect:**  
*assumes N = bernoulli-stream q*

**and**  $0 < q$   
**and**  $q < 1$   
**and**  $der \in \text{borel-measurable } (G \text{ matur})$   
**and**  $Suc \ n \leq \text{matur}$   
**shows**  $\text{expl-cond-expect } N \ (\text{proj-stoch-proc geom-proc } n) \ (\text{rn-rev-price } N \ \text{der matur} \ (\text{matur} - \text{Suc } n)) \ w =$   
 $(q * \text{rn-rev-price } N \ \text{der matur} \ (\text{matur} - \text{Suc } n) \ (\text{pseudo-proj-True } n \ w) +$   
 $(1 - q) * \text{rn-rev-price } N \ \text{der matur} \ (\text{matur} - \text{Suc } n) \ (\text{pseudo-proj-False } n \ w))$   
 $\langle \text{proof} \rangle$

**lemma** (in *CRR-market-viable*) *rn-price-eq-ind*:  
**assumes**  $N = \text{bernoulli-stream } q$   
**and**  $n < \text{matur}$   
**and**  $0 < q$   
**and**  $q < 1$   
**and**  $der \in \text{borel-measurable } (G \text{ matur})$   
**shows**  $(1+r) * \text{rn-price } N \ \text{der matur } n \ w = q * \text{rn-price } N \ \text{der matur} \ (\text{Suc } n) \ (\text{pseudo-proj-True } n \ w) +$   
 $(1 - q) * \text{rn-price } N \ \text{der matur} \ (\text{Suc } n) \ (\text{pseudo-proj-False } n \ w)$   
 $\langle \text{proof} \rangle$

**lemma** *self-finance-updated-suc-suc*:  
**assumes** *portfolio pf*  
**and**  $\forall n. \text{prices } Mkt \ \text{asset } n \ w \neq 0$   
**shows**  $\text{cls-val-process } Mkt \ (\text{self-finance } Mkt \ v \ \text{pf } \text{asset}) \ (\text{Suc } (\text{Suc } n)) \ w =$   
 $\text{cls-val-process } Mkt \ \text{pf} \ (\text{Suc } (\text{Suc } n)) \ w +$   
 $(\text{prices } Mkt \ \text{asset} \ (\text{Suc } (\text{Suc } n)) \ w / (\text{prices } Mkt \ \text{asset} \ (\text{Suc } n) \ w)) *$   
 $(\text{cls-val-process } Mkt \ (\text{self-finance } Mkt \ v \ \text{pf } \text{asset}) \ (\text{Suc } n) \ w -$   
 $\text{val-process } Mkt \ \text{pf} \ (\text{Suc } n) \ w)$   
 $\langle \text{proof} \rangle$

**lemma** *self-finance-updated-suc-0*:  
**assumes** *portfolio pf*  
**and**  $\forall n \ w. \text{prices } Mkt \ \text{asset } n \ w \neq 0$   
**shows**  $\text{cls-val-process } Mkt \ (\text{self-finance } Mkt \ v \ \text{pf } \text{asset}) \ (\text{Suc } 0) \ w = \text{cls-val-process}$   
 $Mkt \ \text{pf} \ (\text{Suc } 0) \ w +$   
 $(\text{prices } Mkt \ \text{asset} \ (\text{Suc } 0) \ w / (\text{prices } Mkt \ \text{asset} \ 0 \ w)) *$   
 $(\text{val-process } Mkt \ (\text{self-finance } Mkt \ v \ \text{pf } \text{asset}) \ 0 \ w -$   
 $\text{val-process } Mkt \ \text{pf} \ 0 \ w)$   
 $\langle \text{proof} \rangle$

**lemma** *self-finance-updated-ind*:  
**assumes** *portfolio pf*  
**and**  $\forall n \ w. \text{prices } Mkt \ \text{asset } n \ w \neq 0$

**shows**  $\text{cls-val-process Mkt (self-finance Mkt v pf asset) (Suc n) w} = \text{cls-val-process Mkt pf (Suc n) w} +$   
 $(\text{prices Mkt asset (Suc n) w} / (\text{prices Mkt asset n w})) *$   
 $(\text{val-process Mkt (self-finance Mkt v pf asset) n w} -$   
 $\text{val-process Mkt pf n w})$   
 ⟨proof⟩

**lemma (in rfr-disc-equity-market) self-finance-risk-free-update-ind:**  
**assumes** *portfolio pf*  
**shows**  $\text{cls-val-process Mkt (self-finance Mkt v pf risk-free-asset) (Suc n) w} =$   
 $\text{cls-val-process Mkt pf (Suc n) w} +$   
 $(1 + r) * (\text{val-process Mkt (self-finance Mkt v pf risk-free-asset) n w} - \text{val-process Mkt pf n w})$   
 ⟨proof⟩

**lemma (in CRR-market) delta-pf-portfolio:**  
**shows** *portfolio (delta-pf N der matur) ⟨proof⟩*

**lemma (in CRR-market) delta-pf-updated:**  
**shows**  $\text{cls-val-process Mkt (delta-pf N der matur) (Suc n) w} =$   
 $\text{geom-proc (Suc n) w} * \text{delta-price N der matur n w} \langle \text{proof} \rangle$

**lemma (in CRR-market) delta-pf-val-process:**  
**shows**  $\text{val-process Mkt (delta-pf N der matur) n w} =$   
 $\text{geom-proc n w} * \text{delta-price N der matur n w} \langle \text{proof} \rangle$

**lemma (in CRR-market) delta-hedging-cls-val-process:**  
**shows**  $\text{cls-val-process Mkt (delta-hedging N der matur) (Suc n) w} =$   
 $\text{geom-proc (Suc n) w} * \text{delta-price N der matur n w} +$   
 $(1 + r) * (\text{val-process Mkt (delta-hedging N der matur) n w} - \text{geom-proc n w}$   
 $* \text{delta-price N der matur n w})$   
 ⟨proof⟩

**lemma (in CRR-market-viable) delta-hedging-eq-derivative-price:**  
**fixes**  $\text{der}::\text{bool stream} \Rightarrow \text{real}$  **and**  $\text{matur}::\text{nat}$   
**assumes**  $N = \text{bernoulli-stream } ((1 + r - d) / (u - d))$   
**and**  $\text{der} \in \text{borel-measurable } (G \text{ matur})$   
**shows**  $\bigwedge n w. n \leq \text{matur} \implies$   
 $\text{val-process Mkt (delta-hedging N der matur) n w} =$   
 $(\text{rn-price N der matur) n w}$

*<proof>*

**lemma** (in *CRR-market-viable*) *delta-hedging-same-cash-flow*:

**assumes**  $der \in \text{borel-measurable } (G \text{ matur})$

**and**  $N = \text{bernoulli-stream } ((1 + r - d) / (u - d))$

**shows**  $\text{cls-val-process Mkt } (\text{delta-hedging } N \text{ der matur}) \text{ matur } w =$   
 $\text{der } w$

*<proof>*

**lemma** (in *CRR-market*) *delta-hedging-trading-strat*:

**assumes**  $N = \text{bernoulli-stream } q$

**and**  $0 < q$

**and**  $q < 1$

**and**  $der \in \text{borel-measurable } (G \text{ matur})$

**shows** *trading-strategy* (delta-hedging  $N$  der matur) *<proof>*

**lemma** (in *CRR-market*) *delta-hedging-self-financing*:

**shows** *self-financing Mkt* (delta-hedging  $N$  der matur) *<proof>*

**lemma** (in *CRR-market-viable*) *delta-hedging-replicating*:

**assumes**  $der \in \text{borel-measurable } (G \text{ matur})$

**and**  $N = \text{bernoulli-stream } ((1 + r - d) / (u - d))$

**shows** *replicating-portfolio* (delta-hedging  $N$  der matur) der matur

*<proof>*

**definition** (in *disc-equity-market*) *complete-market where*

*complete-market*  $\longleftrightarrow (\forall \text{ matur. } \forall der \in \text{borel-measurable } (F \text{ matur}). (\exists p. \text{replicat-}$   
 $\text{ing-portfolio } p \text{ der matur}))$

**lemma** (in *CRR-market-viable*) *CRR-market-complete*:

**shows** *complete-market* *<proof>*

**lemma** *subalgebras-filtration*:

**assumes** *filtration*  $M F$

**and**  $\forall t. \text{subalgebra } (F t) (G t)$

**and**  $\forall s t. s \leq t \longrightarrow \text{subalgebra } (G t) (G s)$

**shows** *filtration*  $M G$  *<proof>*

**lemma** *subfilt-filt-equiv*:

**assumes** *filt-equiv*  $F M N$

**and**  $\forall t. \text{subalgebra } (F t) (G t)$

**and**  $\forall s t. s \leq t \longrightarrow \text{subalgebra } (G t) (G s)$

**shows** *filt-equiv*  $G M N$  *<proof>*

**lemma** (in *CRR-market-viable*) *CRR-market-fair-price*:

```

assumes  $pyf \in \text{borel-measurable } (G \text{ matur})$ 
shows fair-price Mkt
   $(\sum w \in \text{range } (\text{pseudo-proj-True } \text{matur}). (\text{prod } (\text{prob-component } ((1 + r - d) / (u - d)) w) \{0..<\text{matur}\}) * ((\text{discounted-value } r (\lambda m. pyf) \text{matur}) w))$ 
   $pyf \text{ matur}$ 
 $\langle \text{proof} \rangle$ 

```

```

end
theory Option-Price-Examples imports CRR-Model

```

```

begin

```

This file contains pricing results for four options in the Cox-Ross-Rubinstein model. The first section contains results relating some functions to the more abstract counterparts that were used to prove fairness and completeness results. The second section contains the pricing results for a few options; some path-dependent and others not.

## 9 Effective computation definitions and results

### 9.1 Generation of lists of boolean elements

The function `gener-bool-list` permits to generate lists of boolean elements. It is used to generate a list representative of the range of boolean streams by the function `pseudo-proj-True`.

```

fun gener-bool-list where
  gener-bool-list 0 = {}
  | gener-bool-list (Suc n) = {True # w | w. w ∈ gener-bool-list n} ∪ {False # w | w. w ∈ gener-bool-list n}

```

**lemma** *gener-bool-list-elem-length*:

```

shows  $\bigwedge x. x \in \text{gener-bool-list } n \implies \text{length } x = n$ 
 $\langle \text{proof} \rangle$ 

```

**lemma** (in *infinite-coin-toss-space*) *stake-gener-bool-list*:

```

shows stake n'streams (UNIV::bool set) = gener-bool-list n
 $\langle \text{proof} \rangle$ 

```

**lemma** (in *infinite-coin-toss-space*) *pseudo-range-stake*:

```

assumes  $\bigwedge w. f w = g (\text{stake } n w)$ 
shows  $(\sum w \in \text{range } (\text{pseudo-proj-True } n). f w) = (\sum y \in (\text{gener-bool-list } n). g y)$ 
 $\langle \text{proof} \rangle$ 

```

### 9.2 Probability components for lists

```

fun lprob-comp where

```



*lprob-comp* (*p::real*) [] = 1  
 | *lprob-comp* *p* (*x # xs*) = (if *x* then *p* else (1-*p*)) \* *lprob-comp* *p* *xs*

**lemma** *lprob-comp-last*:

**shows** *lprob-comp* *p* (*xs @ [x]*) = (*lprob-comp* *p* *xs*) \* (if *x* then *p* else (1 - *p*))  
 <proof>

**lemma** (in *infinite-coin-toss-space*) *lprob-comp-stake*:

**shows** (prod (prob-component *pr* *w*) {0..< *matur*}) = *lprob-comp* *pr* (stake *matur* *w*)  
 <proof>

### 9.3 Geometric process applied to lists

**fun** *lrev-geom* **where**

*lrev-geom* *u d v* [] = *v*  
 | *lrev-geom* *u d v* (*x#xs*) = (if *x* then *u* else *d*) \* *lrev-geom* *u d v* *xs*

**fun** *lgeom-proc* **where** *lgeom-proc* *u d v l* = *lrev-geom* *u d v* (rev *l*)

**lemma** (in *infinite-coin-toss-space*) *geom-lgeom*:

**shows** *geom-rand-walk* *u d v n w* = *lgeom-proc* *u d v* (stake *n* *w*)  
 <proof>

**lemma** *lgeom-proc-take*:

**assumes** *i* ≤ *n*  
**shows** *lgeom-proc* *u d* *init* (stake *i* *w*) = *lgeom-proc* *u d* *init* (take *i* (stake *n* *w*))  
 <proof>

### 9.4 Effective computation of discounted values

**fun** *det-discount* **where**

*det-discount* (*r::real*) 0 = 1  
 | *det-discount* *r* (Suc *n*) = (inverse (1+*r*)) \* (*det-discount* *r* *n*)

**lemma** *det-discounted*:

**shows** *discounted-value* *r* *X n w* = (*det-discount* *r* *n*) \* (*X n w*) <proof>

## 10 Pricing results on options

### 10.1 Call option

A call option is parameterized by a strike *K* and maturity *T*. If *S* denotes the price of the (unique) risky asset at time *T*, then the option pays max(*S* - *K*, 0) at that time.

**definition** (in *CRR-market*) *call-option* **where**

$call-option (T::nat) (K::real) = (\lambda w. max (prices Mkt stk T w - K) 0)$

**lemma** (in *CRR-market*) *call-borel*:

**shows**  $call-option T K \in borel-measurable (G T)$  *<proof>*

**lemma** (in *CRR-market-viable*) *call-option-lgeom*:

**shows**  $call-option T K w = max ((lgeom-proc u d init (stake T w)) - K) 0$   
*<proof>*

**lemma** (in *CRR-market-viable*) *disc-call-option-lgeom*:

**shows**  $(discounted-value r (\lambda m. (call-option T K)) T w) =$   
 $(det-discount r T) * (max ((lgeom-proc u d init (stake T w)) - K) 0)$   
*<proof>*

**lemma** (in *CRR-market-viable*) *call-effect-compute*:

**shows**  $(\sum_{w \in range (pseudo-proj-True matur).} (prod (prob-component pr w)$   
 $\{0..<matur\}) * (discounted-value r (\lambda m. (call-option matur K)) matur w)) =$   
 $(\sum_{y \in (gener-bool-list matur).} lprob-comp pr y * (det-discount r matur) * (max ((lgeom-proc u d init (take matur y)) - K) 0))$   
*<proof>*

**fun** *call-price where*

$call-price u d init r matur K = (\sum_{y \in (gener-bool-list matur).} lprob-comp ((1 + r - d) / (u - d)) y * (det-discount r matur) * (max ((lgeom-proc u d init (take matur (take matur y))) - K) 0))$

Evaluating the function above returns the fair price of a call option.

**lemma** (in *CRR-market-viable*) *call-price*:

**shows** *fair-price Mkt*  
 $(call-price u d init r matur K)$   
 $(call-option matur K) matur$   
*<proof>*

## 10.2 Put option

A put option is also parameterized by a strike K and maturity T. If S denotes the price of the (unique) risky asset at time T, then the option pays  $\max(K - S, 0)$  at that time.

**definition** (in *CRR-market*) *put-option where*

$put-option (T::nat) (K::real) = (\lambda w. max (K - prices Mkt stk T w) 0)$

**lemma** (in *CRR-market*) *put-borel*:

**shows**  $put-option T K \in borel-measurable (G T)$  *<proof>*

**lemma** (in *CRR-market-viable*) *put-option-lgeom*:

**shows**  $put-option T K w = max (K - (lgeom-proc u d init (stake T w))) 0$   
*<proof>*

**lemma** (in *CRR-market-viable*) *disc-put-option-lgeom*:  
**shows** (*discounted-value*  $r$  ( $\lambda m.$  (*put-option*  $T$   $K$ ))  $T$   $w$ ) =  
(*det-discount*  $r$   $T$ ) \* (*max* ( $K -$  (*lgeom-proc*  $u$   $d$  *init* (*stake*  $T$   $w$ )))  $0$ )  
 $\langle$ *proof* $\rangle$

**lemma** (in *CRR-market-viable*) *put-effect-compute*:  
**shows** ( $\sum_{w \in \text{range } (\text{pseudo-proj-True } \text{matur})}$ . (*prod* (*prob-component*  $pr$   $w$ )  
 $\{0..<\text{matur}\}$ ) \*  
(*discounted-value*  $r$  ( $\lambda m.$  (*put-option*  $\text{matur}$   $K$ ))  $\text{matur}$   $w$ )) =  
( $\sum_{y \in (\text{gener-bool-list } \text{matur})}$ . *lprob-comp*  $pr$   $y$  \* (*det-discount*  $r$   $\text{matur}$ ) \*  
(*max* ( $K -$  (*lgeom-proc*  $u$   $d$  *init* (*take*  $\text{matur}$   $y$ )))  $0$ ))  
 $\langle$ *proof* $\rangle$

**fun** *put-price* **where**  
*put-price*  $u$   $d$  *init*  $r$   $\text{matur}$   $K$  = ( $\sum_{y \in (\text{gener-bool-list } \text{matur})}$ . *lprob-comp* (( $1 +$   
 $r - d$ ) / ( $u - d$ ))  $y$  \* (*det-discount*  $r$   $\text{matur}$ ) \*  
(*max* ( $K -$  (*lgeom-proc*  $u$   $d$  *init* (*take*  $\text{matur}$  (*take*  $\text{matur}$   $y$ ))))  $0$ ))

Evaluating the function above returns the fair price of a put option.

**lemma** (in *CRR-market-viable*) *put-price*:  
**shows** *fair-price*  $Mkt$   
(*put-price*  $u$   $d$  *init*  $r$   $\text{matur}$   $K$ )  
(*put-option*  $\text{matur}$   $K$ )  $\text{matur}$   
 $\langle$ *proof* $\rangle$

### 10.3 Lookback option

A lookback option is parameterized by a maturity  $T$ . If  $S_n$  denotes the price of the (unique) risky asset at time  $n$ , then the option pays  $\max(S_n, 0 \leq n \leq T) - S_T$  at that time.

**definition** (in *CRR-market*) *lbk-option* **where**  
*lbk-option* ( $T::\text{nat}$ ) = ( $\lambda w.$  *Max* (( $\lambda i.$  (*prices*  $Mkt$   $stk$ )  $i$   $w$ )) $\{0 .. T\}$ ) - (*prices*  $Mkt$   $stk$   $T$   $w$ ))

**lemma** *borel-measurable-Max-finite*:  
**fixes**  $f::'a \Rightarrow 'b \Rightarrow 'c::\{\text{second-countable-topology, linorder-topology}\}$   
**assumes**  $0 < (n::\text{nat})$   
**shows**  $\bigwedge A.$   $\text{card } A = n \implies \forall a \in A. f a \in \text{borel-measurable } M \implies (\lambda w.$  *Max*  
( $(\lambda a. f a w)$ ' $A$ ))  $\in \text{borel-measurable } M$   $\langle$ *proof* $\rangle$

**lemma** (in *CRR-market*) *lbk-borel*:  
**shows** *lbk-option*  $T \in \text{borel-measurable } (G$   $T)$   $\langle$ *proof* $\rangle$

**lemma** (in *CRR-market-viable*) *lbk-option-lgeom*:  
**shows** *lbk-option*  $T$   $w$  = *Max* (( $\lambda i.$  (*lgeom-proc*  $u$   $d$  *init* (*stake*  $i$   $w$ ))) $\{0 .. T\}$ )  
- (*lgeom-proc*  $u$   $d$  *init* (*stake*  $T$   $w$ ))

*<proof>*

**lemma** (in *CRR-market-viable*) *disc-lbk-option-lgeom*:  
**shows** (*discounted-value*  $r$  ( $\lambda m.$  (*lbk-option*  $T$ ))  $T$   $w$ ) =  
(*det-discount*  $r$   $T$ ) \* (*Max* (( $\lambda i.$  (*lgeom-proc*  $u$   $d$  *init* (*take*  $i$  (*stake*  $T$   $w$ )))) { $0$   
..  $T$ }) - (*lgeom-proc*  $u$   $d$  *init* (*stake*  $T$   $w$ )))  
*<proof>*

**lemma** (in *CRR-market-viable*) *lbk-effect-compute*:  
**shows** ( $\sum_{w \in \text{range } (\text{pseudo-proj-True } \text{matur})}.$  (*prod* (*prob-component*  $pr$   $w$ )  
{ $0..<\text{matur}$ >} \*  
(*discounted-value*  $r$  ( $\lambda m.$  (*lbk-option*  $\text{matur}$ ))  $\text{matur}$   $w$ )) =  
( $\sum_{y \in (\text{gener-bool-list } \text{matur}).}$  *lprob-comp*  $pr$   $y$  \* (*det-discount*  $r$   $\text{matur}$ ) \*  
(*Max* (( $\lambda i.$  (*lgeom-proc*  $u$   $d$  *init* (*take*  $i$   $y$ )))) { $0$  ..  $\text{matur}$ }) - (*lgeom-proc*  $u$   $d$   
*init*  $y$ )))  
*<proof>*

**fun** *lbk-price* **where**  
*lbk-price*  $u$   $d$  *init*  $r$   $\text{matur}$  = ( $\sum_{y \in (\text{gener-bool-list } \text{matur}).}$  *lprob-comp* (( $1 + r -$   
 $d$ ) / ( $u - d$ )  $y$  \* (*det-discount*  $r$   $\text{matur}$ ) \*  
(*Max* (( $\lambda i.$  (*lgeom-proc*  $u$   $d$  *init* (*take*  $i$   $y$ )))) { $0$  ..  $\text{matur}$ }) - (*lgeom-proc*  $u$   $d$   
*init*  $y$ )))

Evaluating the function above returns the fair price of a lookback option.

**lemma** (in *CRR-market-viable*) *lbk-price*:  
**shows** *fair-price*  $Mkt$   
(*lbk-price*  $u$   $d$  *init*  $r$   $\text{matur}$ )  
(*lbk-option*  $\text{matur}$ )  $\text{matur}$   
*<proof>*

**value** *lbk-price* 1.2 0.8 10 0.03 2

## 10.4 Asian option

An asian option is parameterized by a maturity  $T$ . This option pays the average price of the risky asset at time  $T$ .

**definition** (in *CRR-market*) *asian-option* **where**  
*asian-option* ( $T::\text{nat}$ ) = ( $\lambda w.$  ( $\sum_{i \in \{1.. T\}.$  *prices*  $Mkt$  *stk*  $i$   $w$ )/ $T$ )

**lemma** (in *CRR-market*) *asian-borel*:  
**shows** *asian-option*  $T \in \text{borel-measurable } (G$   $T)$  *<proof>*

**lemma** (in *CRR-market-viable*) *asian-option-lgeom*:  
**shows** *asian-option*  $T$   $w$  = ( $\sum_{i \in \{1.. T\}.$  *lgeom-proc*  $u$   $d$  *init* (*stake*  $i$   $w$ ))/  $T$   
*<proof>*

**lemma** (in *CRR-market-viable*) *disc-asian-option-lgeom*:  
**shows** (*discounted-value*  $r$  ( $\lambda m.$  (*asian-option*  $T$ ))  $T$   $w$ ) =  
 $(\text{det-discount } r \ T) * (\sum_{i \in \{1.. T\}} \text{lgeom-proc } u \ d \ \text{init } (\text{take } i \ (\text{stake } T \ w))) /$   
 $T$   
*<proof>*

**lemma** (in *CRR-market-viable*) *asian-effect-compute*:  
**shows** ( $\sum_{w \in \text{range } (\text{pseudo-proj-True } \text{matur})} (\text{prod } (\text{prob-component } \text{pr } w) \{0..<\text{matur}\}) * (\text{discounted-value } r \ (\lambda m. (\text{asian-option } \text{matur})) \ \text{matur } w) = (\sum_{y \in (\text{gener-bool-list } \text{matur})} \text{lprob-comp } \text{pr } y * (\text{det-discount } r \ \text{matur}) * (\sum_{i \in \{1.. \text{matur}\}} \text{lgeom-proc } u \ d \ \text{init } (\text{take } i \ y)) / \text{matur})$ )  
*<proof>*

**fun** *asian-price* **where**  
*asian-price*  $u \ d \ \text{init } r \ \text{matur} = (\sum_{y \in (\text{gener-bool-list } \text{matur})} \text{lprob-comp } ((1 + r - d) / (u - d)) \ y * (\text{det-discount } r \ \text{matur}) * (\sum_{i \in \{1.. \text{matur}\}} \text{lgeom-proc } u \ d \ \text{init } (\text{take } i \ y)) / \text{matur})$

Evaluating the function above returns the fair price of an asian option.

**lemma** (in *CRR-market-viable*) *asian-price*:  
**shows** *fair-price*  $Mkt$   
 $(\text{asian-price } u \ d \ \text{init } r \ \text{matur})$   
 $(\text{asian-option } \text{matur}) \ \text{matur}$   
*<proof>*

**end**