

Negatively Associated Random Variables

Emin Karayel

February 6, 2026

Abstract

Negative Association is a generalization of independence for random variables, that retains some of the key properties of independent random variables. In particular closure properties, such as composition with monotone functions, as well as, the well-known Chernoff-Hoeffding bounds.

This entry introduces the concept and verifies the most important closure properties, as well as, the concentration inequalities. It also verifies the FKG inequality, which is a generalization of Chebyshev's sum inequality for distributive lattices and a key tool for establishing negative association, but has also many applications beyond the context of negative association, in particular, statistical physics and graph theory.

As an example, permutation distributions are shown to be negatively associated, from which many more sets of negatively random variables can be derived, such as, e.g., n -subsets, or the the balls-into-bins process.

Finally, the entry derives a correct false-positive rate for Bloom filters using the library.

Contents

1	Preliminary Definitions and Lemmas	2
2	Definition	10
3	Chernoff-Hoeffding Bounds	31
4	The FKG inequality	46
5	Preliminary Results on Lattices	54
6	Permutation Distributions	62
7	Application: Bloom Filters	90

1 Preliminary Definitions and Lemmas

theory *Negative-Association-Util*

imports

Concentration-Inequalities.Concentration-Inequalities-Preliminary

Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF

begin

abbreviation (*input*) *flip* :: $\langle 'a \Rightarrow 'b \Rightarrow 'c \rangle \Rightarrow 'b \Rightarrow 'a \Rightarrow 'c$ **where**
 $\langle \text{flip } f \ x \ y \equiv f \ y \ x \rangle$

Additional introduction rules for boundedness:

lemma *bounded-const-min*:

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

assumes *bdd-below* ($f \ 'M$)

shows *bounded* $((\lambda x. \min c (f \ x)) \ 'M)$

proof –

obtain h **where** $\bigwedge x. x \in M \implies f \ x \geq h$ **using** *assms(1)* **unfolding** *bdd-below-def*
by *auto*

thus *?thesis* **by** (*intro boundedI*[**where** $B = \max |c| \ |-h|$]) *force*
qed

lemma *bounded-prod*:

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

assumes *finite* I

assumes $\bigwedge i. i \in I \implies \text{bounded } (f \ i \ 'T)$

shows *bounded* $((\lambda x. (\prod i \in I. f \ i \ x)) \ 'T)$

using *assms* **by** (*induction* I) (*auto intro:bounded-mult-comp bounded-const*)

lemma *bounded-vec-mult-comp*:

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

assumes *bounded* ($f \ 'T$) *bounded* ($g \ 'T$)

shows *bounded* $((\lambda x. (f \ x) *_{\mathbb{R}} (g \ x)) \ 'T)$

using *bounded-mult-comp[OF assms]* **by** *simp*

lemma *bounded-max*:

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

assumes *bounded* $((\lambda x. f \ x) \ 'T)$

shows *bounded* $((\lambda x. \max c (f \ x)) \ 'T)$

proof –

obtain m **where** $\text{norm } (f \ x) \leq m$ **if** $x \in T$ **for** x
using *assms* **unfolding** *bounded-iff* **by** *auto*

thus *?thesis* **by** (*intro boundedI*[**where** $B = \max m \ c$]) *fastforce*
qed

lemma *bounded-of-bool*: *bounded* (*range of-bool*) **by** *auto*

lemma *bounded-range-imp*:
assumes *bounded (range f)*
shows *bounded (($\lambda\omega. f (h \omega)$) 'S)*
by (*intro bounded-subset[OF assms]*) *auto*

The following allows to state integrability and conditions about the integral simultaneously, e.g. *has-int-that M f ($\lambda x. x \leq c$)* says *f* is integrable on *M* and the integral smaller or equal to *c*.

definition *has-int-that where*
has-int-that M f P = (integrable M f \wedge (P ($\int \omega. f \omega \partial M$)))

lemma *true-eq-iff*: *P \implies True = P* **by** *auto*

lemma *le-trans*: *y \leq z \implies x \leq y \longrightarrow x \leq (z :: 'a :: order)* **by** *auto*

lemma *has-int-that-mono*:
assumes $\bigwedge x. P x \longrightarrow Q x$
shows *has-int-that M f P \leq has-int-that M f Q*
using *assms unfolding has-int-that-def* **by** *auto*

lemma *has-int-thatD*:
assumes *has-int-that M f P*
shows *integrable M f P (integral^L M f)*
using *assms has-int-that-def* **by** *auto*

This is useful to specify which components a functional depends on.

definition *depends-on :: (('a \Rightarrow 'b) \Rightarrow 'c) \Rightarrow 'a set \Rightarrow bool*
where *depends-on f I = ($\forall x y. \text{restrict } x I = \text{restrict } y I \longrightarrow f x = f y$)*

lemma *depends-onI*:
assumes $\bigwedge x. f x = f (\lambda i. \text{if } i \in I \text{ then } (x i) \text{ else undefined})$
shows *depends-on f I*

proof –

have *f x = f y* **if** *restrict x I = restrict y I* **for** *x y*

proof –

have *f x = f (restrict x I)* **using** *assms unfolding restrict-def* **by** *simp*

also have *... = f (restrict y I)* **using** *that* **by** *simp*

also have *... = f y* **using** *assms unfolding restrict-def* **by** *simp*

finally show *?thesis* **by** *simp*

qed

thus *?thesis* **unfolding** *depends-on-def* **by** *blast*

qed

lemma *depends-on-comp*:
assumes *depends-on f I*
shows *depends-on (g \circ f) I*
using *assms unfolding depends-on-def* **by** (*metis o-apply*)

lemma *depends-on-comp-2*:
assumes *depends-on f I*

shows *depends-on* $(\lambda x. g (f x)) I$
using *assms unfolding depends-on-def by metis*

lemma *depends-onD*:
assumes *depends-on f I*
shows $f \omega = f (\lambda i \in I. (\omega i))$
using *assms unfolding depends-on-def by (metis extensional-restrict restrict-extensional)*

lemma *depends-onD2*:
assumes *depends-on f I restrict x I = restrict y I*
shows $f x = f y$
using *assms unfolding depends-on-def by metis*

lemma *depends-on-empty*:
assumes *depends-on f {}*
shows $f \omega = f \text{undefined}$
by (*intro depends-onD2[OF assms]*) *auto*

lemma *depends-on-mono*:
assumes $I \subseteq J$ *depends-on f I*
shows *depends-on f J*
using *assms unfolding depends-on-def by (metis restrict-restrict Int-absorb1)*

abbreviation *square-integrable M f* \equiv *integrable M ((power2 :: real \Rightarrow real) \circ f)*

There are many results in the field of negative association, where a statement is true for simultaneously monotone or anti-monotone functions. With the below construction, we introduce a mechanism where we can parameterize on the direction of a relation:

datatype *RelDirection* = *Fwd* | *Rev*

definition *dir-le* :: *RelDirection* \Rightarrow ($'d :: \text{order}$) \Rightarrow ($'d :: \text{order}$) \Rightarrow *bool* (**infixl** $\leq_{\geq 1} 60$)
where *dir-le* η = (*if* $\eta = \text{Fwd}$ *then* (\leq) *else* (\geq))

lemma *dir-le[simp]*:
 $(\leq_{\text{Fwd}}) = (\leq)$
 $(\leq_{\text{Rev}}) = (\geq)$
by (*auto simp:dir-le-def*)

definition *dir-sign* :: *RelDirection* \Rightarrow $'a :: \{\text{one}, \text{uminus}\}$ (± 1)
where *dir-sign* η = (*if* $\eta = \text{Fwd}$ *then* 1 *else* (-1))

lemma *dir-le-refl*: $x \leq_{\eta} x$
by (*cases* η) *auto*

lemma *dir-sign[simp]*:
 $(\pm_{\text{Fwd}}) = (1)$
 $(\pm_{\text{Rev}}) = (-1)$

by (auto simp:dir-sign-def)

lemma conv-rel-to-sign:

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

shows monotone (\leq) $(\leq_{\geq \eta}) f = mono ((*)(\pm_{\eta}) \circ f)$

by (cases η) (simp-all add:monotone-def)

instantiation RelDirection :: times

begin

definition times-RelDirection :: RelDirection \Rightarrow RelDirection \Rightarrow RelDirection **where**

times-RelDirection-def: times-RelDirection $x y = (if x = y then Fwd else Rev)$

instance by standard

end

lemmas rel-dir-mult[simp] = times-RelDirection-def

lemma dir-mult-hom: $(\pm_{\sigma} * \tau) = (\pm_{\sigma}) * ((\pm_{\tau})::real)$

unfolding dir-sign-def times-RelDirection-def **by** (cases σ , auto intro:RelDirection.exhaust)

Additional lemmas about clamp for the specific case on reals.

lemma clamp-eqI2:

assumes $x \in \{a..b::real\}$

shows $x = clamp a b x$

using assms **unfolding** clamp-def **by** simp

lemma clamp-eqI:

assumes $|x| \leq (a::real)$

shows $x = clamp (-a) a x$

using assms **by** (intro clamp-eqI2) auto

lemma clamp-real-def:

fixes $x :: real$

shows $clamp a b x = max a (min x b)$

proof –

consider (i) $x < a$ | (ii) $x \geq a$ $x \leq b$ | (iii) $x > b$ **by** linarith

thus ?thesis **unfolding** clamp-def **by** (cases) auto

qed

lemma clamp-range:

assumes $a \leq b$

shows $\bigwedge x. clamp a b x \geq a \bigwedge x. clamp a b x \leq b$ range (clamp a b) $\subseteq \{a..b::real\}$

using assms **by** (auto simp: clamp-real-def)

lemma clamp-abs-le:

assumes $a \geq (0::real)$

shows $|clamp (-a) a x| \leq |x|$

using assms **unfolding** clamp-real-def **by** simp

```

lemma bounded-clamp:
  fixes a b :: real
  shows bounded ((clamp a b o f) ' S)
proof (cases a ≤ b)
  case True
  show ?thesis using clamp-range[OF True] bounded-closed-interval bounded-subset
    by (metis image-comp image-mono subset-UNIV)
  next
  case False
  hence clamp a b (f x) = a for x unfolding clamp-def by (simp add: max-def)
  hence (clamp a b o f) ' S ⊆ {a..a} by auto
  thus ?thesis using bounded-subset bounded-closed-interval by metis
qed

```

```

lemma bounded-clamp-alt:
  fixes a b :: real
  shows bounded ((λx. clamp a b (f x)) ' S)
  using bounded-clamp by (auto simp: comp-def)

```

```

lemma clamp-borel[measurable]:
  fixes a b :: 'a::{euclidean-space, second-countable-topology}
  shows clamp a b ∈ borel-measurable borel
  unfolding clamp-def by measurable

```

```

lemma monotone-clamp:
  assumes monotone (≤) (≤η) f
  shows monotone (≤) (≤η) (λω. clamp a (b::real) (f ω))
  using assms unfolding monotone-def clamp-real-def by (cases η) force+

```

This part introduces the term *KL-div* as the Kullback-Leibler divergence between a pair of Bernoulli random variables. The expression is useful to express some of the Chernoff bounds more concisely [12, Th. 1].

```

lemma radon-nikodym-pmf:
  assumes set-pmf p ⊆ set-pmf q
  defines f ≡ (λx. ennreal (pmf p x / pmf q x))
  shows
    AE x in measure-pmf q. RN-deriv q p x = f x (is ?R1)
    AE x in measure-pmf p. RN-deriv q p x = f x (is ?R2)
proof –
  have pmf p x = 0 if pmf q x = 0 for x
    using assms(1) that by (meson pmf-eq-0-set-pmf subset-iff)
  hence a:(pmf q x * (pmf p x / pmf q x)) = pmf p x for x by simp
  have emeasure (density q f) A = emeasure p A (is ?L = ?R) for A
proof –
  have ?L = set-nn-integral (measure-pmf q) A f
    by (subst emeasure-density) auto
  also have ... = (∫+ x∈A. ennreal (pmf q x) * f x ∂count-space UNIV)
    by (simp add: ac-simps nn-integral-measure-pmf)
  also have ... = (∫+ x∈A. ennreal (pmf p x) ∂count-space UNIV)

```

using a **unfolding** f -def by (subst ennreal-mult^[symmetric]) simp-all
 also have ... = emeasure (bind-pmf p return-pmf) A
 unfolding emeasure-bind-pmf nn-integral-measure-pmf by simp
 also have ... = ?R by simp
 finally show ?thesis by simp
qed
 hence density (measure-pmf q) f = measure-pmf p by (intro measure-eqI) auto
 hence AE x in measure-pmf q. f x = RN-deriv q p x by (intro measure-pmf.RN-deriv-unique)
 simp
 thus ?R1 **unfolding** AE-measure-pmf-iff by auto
 thus ?R2 using assms **unfolding** AE-measure-pmf-iff by auto
qed

lemma *KL-divergence-pmf*:
 assumes set-pmf q \subseteq set-pmf p
 shows *KL-divergence* b (measure-pmf p) (measure-pmf q) = ($\int x. \log b$ (pmf q x
 / pmf p x) ∂ q)
 unfolding *KL-divergence-def* *entropy-density-def*
 by (intro integral-cong-AE AE-mp[OF radon-nikodym-pmf(2)][OF assms(1)] AE-I2)
 auto

definition *KL-div* :: real \Rightarrow real \Rightarrow real **where**
KL-div p q = *KL-divergence* (exp 1) (bernoulli-pmf q) (bernoulli-pmf p)

lemma *KL-div-eq*:
 assumes q \in {0<.. <1 } p \in {0..1}
 shows *KL-div* p q = p * ln (p/q) + (1-p) * ln ((1-p)/(1-q)) (is ?L = ?R)
proof –
 have set-pmf (bernoulli-pmf p) \subseteq set-pmf (bernoulli-pmf q)
 using assms(1) set-pmf-bernoulli by auto
 hence ?L = ($\int x. \ln$ (pmf (bernoulli-pmf p) x / pmf (bernoulli-pmf q) x)
 ∂ bernoulli-pmf p)
 unfolding *KL-div-def* by (subst *KL-divergence-pmf*) (simp-all add:log-ln[symmetric])
 also have ... = ?R
 using assms(1,2) by (subst integral-bernoulli-pmf) auto
 finally show ?thesis by simp
qed

lemma *KL-div-extreme-cases*:
 assumes p \in {0,1}
 shows *KL-div* p p = 0 (is ?L = ?R)
proof –
 have ?L = ($\int x. \ln$ (pmf (bernoulli-pmf p) x / pmf (bernoulli-pmf p) x) ∂ bernoulli-pmf
 p)
 unfolding *KL-div-def* by (subst *KL-divergence-pmf*) (simp-all add:log-ln[symmetric])
 also have ... = 0 using assms
 by (cases p = 0) auto
 finally show ?thesis by simp
qed

lemma *KL-div-eq'*:
assumes $q \in \{0..1\} \ p \in \{0..1\} \ q > 0 \vee p = 0 \ q < 1 \vee p = 1$
shows $KL\text{-div } p \ q = p * \ln (p/q) + (1-p) * \ln ((1-p)/(1-q))$ (is ?L = ?R)
proof (cases $q \in \{0,1\}$)
 case *True*
 hence $0:p = q$ **using** *assms* **by** *auto*
 hence $KL\text{-div } p \ q = 0$ **using** *True* **unfolding** 0 **by** (*intro KL-div-extreme-cases*)
 auto
 also have $\dots = p * \ln (p/q) + (1-p) * \ln ((1-p)/(1-q))$ **unfolding** 0 **using**
 True **by** *simp*
 finally show *?thesis* **by** *simp*
 next
 case *False*
 then show *?thesis* **using** *assms* **by** (*intro KL-div-eq*) *auto*
qed

lemma *KL-div-swap-gen*:
assumes $q \in \{0..1\} \ p \in \{0..1\} \ q > 0 \vee p = 0 \ q < 1 \vee p = 1$
shows $KL\text{-div } p \ q = KL\text{-div } (1-p) \ (1-q)$
using *assms* **by** (*subst* $(1 \ 2)$ *KL-div-eq'*) *auto*

lemma *KL-div-swap*:
assumes $q \in \{0 < .. < 1\} \ p \in \{0..1\}$
shows $KL\text{-div } p \ q = KL\text{-div } (1-p) \ (1-q)$
using *assms* **by** (*intro KL-div-swap-gen*) *auto*

A few results about independent random variables:

lemma (in *prob-space*) *indep-vars-const*:
assumes $\bigwedge i. i \in I \implies c \ i \in \text{space } (N \ i)$
shows *indep-vars* $N \ (\lambda i -. c \ i) \ I$
proof –
 have *rv*: *random-variable* $(N \ i) \ (\lambda -. c \ i)$ **if** $i \in I$ **for** i **using** *assms*[*OF that*]
by *simp*
 have *b*:*indep-sets* $(\lambda i. \{\text{space } M, \{\}\}) \ I$
 proof (*intro indep-setsI, goal-cases*)
 case $(1 \ i)$ **thus** *?case* **by** *simp*
 next
 case $(2 \ A \ J)$
 show *?case*
 proof (cases $\forall j \in J. A \ j = \text{space } M$)
 case *True* **thus** *?thesis* **using** $2(1)$ **by** (*simp add:prob-space*)
 next
 case *False*
 then obtain i **where** $i:A \ i = \{\}$ $i \in J$ **using** 2 **by** *auto*
 hence $\text{prob } (\bigcap (A \ ' J)) = \text{prob } \{\}$ **by** (*intro arg-cong[where f=prob]*) *auto*
 also have $\dots = 0$ **by** *simp*
 also have $\dots = (\prod_{j \in J. \text{prob } (A \ j)})$
 using i **by** (*intro prod-zero[symmetric] 2 bexI[where x=i]*) *auto*

```

    finally show ?thesis by simp
  qed
qed
have  $\{(\lambda-. c i) - 'A \cap \text{space } M \mid A. A \in \text{sets } (N i)\} = \{\text{space } M, \{\}\}$  (is ?L = ?R) if  $i \in I$  for  $i$ 
proof
  show ?L  $\subseteq$  ?R by auto
next
  have  $(\lambda A. (\lambda-. c i) - 'A \cap \text{space } M) \{\} = \{\} \{\} \in N i$  by auto
  hence  $\{\} \in ?L$  unfolding image-Collect[symmetric] by blast
  moreover have  $(\lambda A. (\lambda-. c i) - 'A \cap \text{space } M) (\text{space } (N i)) = \text{space } M \text{ space } (N i) \in N i$ 
  using assms[OF that] by auto
  hence  $\text{space } M \in ?L$  unfolding image-Collect[symmetric] by blast
  ultimately show ?R  $\subseteq$  ?L by simp
qed
hence indep-sets  $(\lambda i. \{(\lambda-. c i) - 'A \cap \text{space } M \mid A. A \in \text{sets } (N i)\}) I$ 
  using iffD2[OF indep-sets-cong b] b by simp
thus ?thesis unfolding indep-vars-def2 by (intro conjI rv ballI)
qed

```

lemma *indep-vars-map-pmf*:

```

  assumes prob-space.indep-vars (measure-pmf p) ( $\lambda-. \text{discrete}$ ) ( $\lambda i. X i \circ f$ ) I
  shows prob-space.indep-vars (map-pmf f p) ( $\lambda-. \text{discrete}$ ) X I
  using assms unfolding map-pmf-rep-eq by (intro measure-pmf.indep-vars-distr)
auto

```

lemma *indep-var-pair-pmf*:

```

  fixes  $x y :: 'a \text{ pmf}$ 
  shows prob-space.indep-var (pair-pmf x y) discrete fst discrete snd
proof -

```

```

  have split-bool-univ: UNIV = insert True {False} by auto

```

```

  have pair-prod: pair-pmf x y = map-pmf ( $\lambda \omega. (\omega \text{ True}, \omega \text{ False})$ ) (prod-pmf UNIV (case-bool x y))

```

```

  unfolding split-bool-univ by (subst Pi-pmf-insert)
  (simp-all add: map-pmf-comp Pi-pmf-singleton pair-map-pmf2 case-prod-beta)

```

```

  have case-bool-eq: case-bool discrete discrete = ( $\lambda-. \text{discrete}$ )
  by (intro ext) (simp add: bool.case-eq-if)

```

```

  have prob-space.indep-vars (prod-pmf UNIV (case-bool x y)) ( $\lambda-. \text{discrete}$ ) ( $\lambda i \omega. \omega i$ ) UNIV

```

```

  by (intro indep-vars-Pi-pmf) auto

```

```

  moreover have  $(\lambda i. (\text{case-bool fst snd } i) \circ (\lambda \omega. ((\omega \text{ True})::'a, \omega \text{ False}))) = (\lambda i \omega. \omega i)$ 

```

```

  by (auto intro!: ext split:bool.splits)

```

```

  ultimately show ?thesis

```

```

  unfolding prob-space.indep-var-def[OF prob-space-measure-pmf] pair-prod case-bool-eq

```

```

    by (intro indep-vars-map-pmf) simp
qed

```

```

lemma measure-pair-pmf: measure (pair-pmf p q) (A × B) = measure p A *
measure q B (is ?L = ?R)

```

```

proof –

```

```

    have ?L = measure (pair-pmf p q) ((A ∩ set-pmf p) × (B ∩ set-pmf q))

```

```

    by (intro measure-eq-AE AE-pmfI) auto

```

```

    also have ... = measure p (A ∩ set-pmf p) * measure q (B ∩ set-pmf q)

```

```

    by (intro measure-pmf-prob-product) auto

```

```

    also have ... = ?R by (intro arg-cong2[where f=(*)] measure-eq-AE AE-pmfI)
auto

```

```

    finally show ?thesis by simp

```

```

qed

```

```

instance bool :: second-countable-topology

```

```

proof

```

```

    show ∃ B::bool set set. countable B ∧ open = generate-topology B

```

```

    by (intro exI[of - range lessThan ∪ range greaterThan]) (auto simp: open-bool-def)

```

```

qed

```

```

end

```

2 Definition

This section introduces the concept of negatively associated random variables (RVs). The definition follows, as closely as possible, the original description by Joag-Dev and Proschan [13].

However, the following modifications have been made:

Singleton and empty sets of random variables are considered negatively associated. This is useful because it simplifies many of the induction proofs. The second modification is that the RV's don't have to be real valued. Instead the range can be into any linearly ordered space with the borel σ -algebra. This is a major enhancement compared to the original work, as well as results by following authors [6, 7, 8, 14, 17].

```

theory Negative-Association-Definition

```

```

imports

```

```

    Concentration-Inequalities.Bienaymes-Identity

```

```

    Negative-Association-Util

```

```

begin

```

```

context prob-space

```

```

begin

```

```

definition neg-assoc :: ('i ⇒ 'a ⇒ 'c :: {linorder-topology}) ⇒ 'i set ⇒ bool

```

```

    where neg-assoc X I = (

```

$(\forall i \in I. \text{random-variable borel } (X i)) \wedge$
 $(\forall (f::\text{nat} \Rightarrow ('i \Rightarrow 'c) \Rightarrow \text{real}) J. J \subseteq I \wedge$
 $(\forall \iota < 2. \text{bounded } (\text{range } (f \iota)) \wedge \text{mono}(f \iota) \wedge \text{depends-on } (f \iota) ([J, I-J]! \iota) \wedge$
 $f \iota \in \text{PiM } ([J, I-J]! \iota) (\lambda \cdot. \text{borel}) \rightarrow_M \text{borel}) \longrightarrow$
 $\text{covariance } (f 0 \circ \text{flip } X) (f 1 \circ \text{flip } X) \leq 0))$

lemma *neg-assocI*:

assumes $\bigwedge i. i \in I \Longrightarrow \text{random-variable borel } (X i)$
assumes $\bigwedge f g J. J \subseteq I$
 $\Longrightarrow \text{depends-on } f J \Longrightarrow \text{depends-on } g (I-J)$
 $\Longrightarrow \text{mono } f \Longrightarrow \text{mono } g$
 $\Longrightarrow \text{bounded } (\text{range } f::\text{real set}) \Longrightarrow \text{bounded } (\text{range } g)$
 $\Longrightarrow f \in \text{PiM } J (\lambda \cdot. \text{borel}) \rightarrow_M \text{borel} \Longrightarrow g \in \text{PiM } (I-J) (\lambda \cdot. \text{borel}) \rightarrow_M \text{borel}$
 $\Longrightarrow \text{covariance } (f \circ \text{flip } X) (g \circ \text{flip } X) \leq 0$
shows *neg-assoc X I*
using *assms unfolding neg-assoc-def* **by** (*auto simp:numeral-eq-Suc All-less-Suc*)

lemma *neg-assocI2*:

assumes *[measurable]*: $\bigwedge i. i \in I \Longrightarrow \text{random-variable borel } (X i)$
assumes $\bigwedge f g J. J \subseteq I$
 $\Longrightarrow \text{depends-on } f J \Longrightarrow \text{depends-on } g (I-J)$
 $\Longrightarrow \text{mono } f \Longrightarrow \text{mono } g$
 $\Longrightarrow \text{bounded } (\text{range } f) \Longrightarrow \text{bounded } (\text{range } g)$
 $\Longrightarrow f \in \text{PiM } J (\lambda \cdot. \text{borel}) \rightarrow_M (\text{borel} :: \text{real measure})$
 $\Longrightarrow g \in \text{PiM } (I-J) (\lambda \cdot. \text{borel}) \rightarrow_M (\text{borel} :: \text{real measure})$
 $\Longrightarrow (\int \omega. f(\lambda i. X i \omega) * g(\lambda i. X i \omega) \partial M) \leq (\int \omega. f(\lambda i. X i \omega) \partial M) * (\int \omega. g(\lambda i. X i \omega) \partial M)$
shows *neg-assoc X I*
proof (*rule neg-assocI, goal-cases*)
case (1 *i*) **thus** ?*case* **using** *assms(1)* **by** *auto*
next
case (2 *f g J*)

note *[measurable]* = 2(8,9)

note *bounded* = *integrable-bounded bounded-intros*

have *[measurable]*: *random-variable borel* $(\lambda \omega. f (\lambda i. X i \omega))$
using *subsetD[OF 2(1)]* **by** (*subst depends-onD[OF 2(2)]*) *measurable*
moreover have *[measurable]*: *random-variable borel* $(\lambda \omega. g (\lambda i. X i \omega))$
by (*subst depends-onD[OF 2(3)]*) *measurable*
moreover have *integrable M* $(\lambda \omega. ((f \circ (\lambda x y. X y x)) \omega)^2)$
unfolding comp-def **by** (*intro bounded bounded-subset[OF 2(6)]*) *auto*
moreover have *integrable M* $(\lambda \omega. ((g \circ (\lambda x y. X y x)) \omega)^2)$
unfolding comp-def **by** (*intro bounded bounded-subset[OF 2(7)]*) *auto*
ultimately show ?*case* **using** *assms(2)[OF 2(1-9)]*
by (*subst covariance-eq*) (*auto simp:comp-def*)

qed

lemma *neg-assoc-empty*:

```

neg-assoc X {}
proof (intro neg-assocI2, goal-cases)
  case (1 i)
  then show ?case by simp
next
  case (2 f g J)
  define fc gc where fc:fc = f undefined and gc:gc = g undefined

  have depends-on f {} depends-on g {} using 2 by auto
  hence fg-simps: f = ( $\lambda x. fc$ ) g = ( $\lambda x. gc$ ) unfolding fc gc using depends-on-empty
  by auto
  then show ?case unfolding fg-simps by (simp add:prob-space)
qed

```

```

lemma neg-assoc-singleton:
  assumes random-variable borel (X i)
  shows neg-assoc X {i}
proof (rule neg-assocI2, goal-cases)
  case (1 i)
  then show ?case using assms by auto
next
  case (2 f g J)
  show ?case
  proof (cases J = {})
    case True
    define fc where fc = f undefined
    have f:f = ( $\lambda-. fc$ )
      unfolding fc-def by (intro ext depends-onD2[OF 2(2)]) (auto simp:True)
    then show ?thesis unfolding f by (simp add:prob-space)
  next
    case False
    hence J: J = {i} using 2(1) by auto
    define gc where gc = g undefined
    have g:g = ( $\lambda-. gc$ )
      unfolding gc-def by (intro ext depends-onD2[OF 2(3)]) (auto simp:J)
    then show ?thesis unfolding g by (simp add:prob-space)
  qed
qed

```

```

lemma neg-assoc-imp-measurable:
  assumes neg-assoc X I
  assumes  $i \in I$ 
  shows random-variable borel (X i)
  using assms unfolding neg-assoc-def by auto

```

Even though the assumption was that defining property is true for pairs of monotone functions over the random variables, it is also true for pairs of anti-monotone functions.

```

lemma neg-assoc-imp-mult-mono-bounded:

```

fixes $f g :: ('i \Rightarrow 'c::\text{linorder-topology}) \Rightarrow \text{real}$
assumes $\text{neg-assoc } X I$
assumes $J \subseteq I$
assumes $\text{bounded (range } f) \text{ bounded (range } g)$
assumes $\text{monotone } (\leq) (\leq_{\geq \eta}) f \text{ monotone } (\leq) (\leq_{\geq \eta}) g$
assumes $\text{depends-on } f J \text{ depends-on } g (I-J)$
assumes $[\text{measurable}]: f \in \text{borel-measurable } (Pi_M J (\lambda-. \text{borel}))$
assumes $[\text{measurable}]: g \in \text{borel-measurable } (Pi_M (I-J) (\lambda-. \text{borel}))$
shows
 $\text{covariance } (f \circ \text{flip } X) (g \circ \text{flip } X) \leq 0$
 $(f \omega. f (\lambda i. X i \omega) * g (\lambda i. X i \omega) \partial M) \leq \text{expectation } (\lambda x. f(\lambda y. X y x)) * \text{expectation } (\lambda x. g(\lambda y. X y x))$
 $(\text{is } ?L \leq ?R)$
proof –
define q **where** $q \iota = (\text{if } \iota = 0 \text{ then } f \text{ else } g)$ **for** $\iota :: \text{nat}$
define h **where** $h \iota = ((* (\pm \eta)) \circ (q \iota))$ **for** $\iota :: \text{nat}$

note $[\text{measurable}] = \text{neg-assoc-imp-measurable}[OF \text{ assms}(1)]$
note $\text{bounded} = \text{integrable-bounded bounded-intros}$

have $1:\text{bounded (range } ((* (\pm \eta)) \circ q \iota) \text{ depends-on } (q \iota) ([J, I-J]! \iota)$
 $q \iota \in PiM ([J, I-J]! \iota) (\lambda-. \text{borel}) \rightarrow_M \text{borel mono } ((* (\pm \eta)) \circ q \iota)$ **if** $\iota \in \{0, 1\}$
for ι
using $\text{that assms unfolding } q\text{-def conv-rel-to-sign by (auto intro:bounded-mult-comp)}$

have $2: ((* (\pm \eta::\text{real})) \in \text{borel} \rightarrow_M \text{borel by simp}$

have $3:\forall \iota < Suc (Suc 0). \text{bounded (range } (h \iota)) \wedge \text{mono}(h \iota) \wedge \text{depends-on } (h \iota)$
 $([J, I-J]! \iota) \wedge$
 $h \iota \in PiM ([J, I-J]! \iota) (\lambda-. \text{borel}) \rightarrow_M \text{borel}$ **unfolding** $\text{All-less-Suc } h\text{-def}$
by $(\text{intro conjI } 1 \text{ depends-on-comp measurable-comp}[OF - 2]) \text{ auto}$

have $\text{covariance } (f \circ \text{flip } X) (g \circ \text{flip } X) = \text{covariance } (q 0 \circ \text{flip } X) (q 1 \circ \text{flip } X)$
unfolding $q\text{-def by simp}$
also have $\dots = \text{covariance } (h 0 \circ \text{flip } X) (h 1 \circ \text{flip } X)$
unfolding $h\text{-def covariance-def comp-def by (cases } \eta) (\text{auto simp:algebra-simps})$
also have $\dots \leq 0$ **using** $3 \text{ assms}(1,2) \text{ numeral-2-eq-2 unfolding neg-assoc-def}$
by metis
finally show $\text{covariance } (f \circ \text{flip } X) (g \circ \text{flip } X) \leq 0$ **by simp**

moreover have $m\text{-f: random-variable borel } (\lambda x. f(\lambda i. X i x))$
using $\text{subsetD}[OF \text{ assms}(2)]$ **by** $(\text{subst depends-onD}[OF \text{ assms}(7)]) \text{ measurable}$
moreover have $m\text{-g: random-variable borel } (\lambda x. g(\lambda i. X i x))$
by $(\text{subst depends-onD}[OF \text{ assms}(8)]) \text{ measurable}$
moreover have $\text{integrable } M (\lambda \omega. ((f \circ (\lambda x y. X y x)) \omega)^2)$ **unfolding comp-def**
by $(\text{intro bounded bounded-subset}[OF \text{ assms}(3)] \text{ measurable-compose}[OF m-f])$
 auto
moreover have $\text{integrable } M (\lambda \omega. ((g \circ (\lambda x y. X y x)) \omega)^2)$ **unfolding comp-def**

by (intro bounded bounded-subset[OF assms(4)] measurable-compose[OF m-g])
 auto

ultimately show $?L \leq ?R$ by (subst (asm) covariance-eq) (auto simp:comp-def)
 qed

lemma lim-min-n: $(\lambda n. \min (\text{real } n) x) \longrightarrow x$

proof -

define m where $m = \text{nat } \lceil |x| \rceil$

have $\min (\text{real } (n+m)) x = x$ for n unfolding m-def by (intro min-absorb2)
 linarith

hence $(\lambda n. \min (\text{real } (n+m)) x) \longrightarrow x$ by simp

thus ?thesis using LIMSEQ-offset[where k=m] by fast

qed

lemma lim-clamp-n: $(\lambda n. \text{clamp } (-\text{real } n) (\text{real } n) x) \longrightarrow x$

proof -

define m where $m = \text{nat } \lceil |x| \rceil$

have $\text{clamp } (-\text{real } (n+m)) (\text{real } (n+m)) x = x$ for n unfolding m-def

by (intro clamp-eqI[symmetric]) linarith

hence $(\lambda n. \text{clamp } (-\text{real } (n+m)) (\text{real } (n+m)) x) \longrightarrow x$ by simp

thus ?thesis using LIMSEQ-offset[where k=m] by fast

qed

lemma neg-assoc-imp-mult-mono:

fixes $f g :: ('i \Rightarrow 'c::\text{linorder-topology}) \Rightarrow \text{real}$

assumes neg-assoc X I

assumes $J \subseteq I$

assumes square-integrable M (f o flip X) square-integrable M (g o flip X)

assumes monotone $(\leq) (\leq \geq \eta)$ f monotone $(\leq) (\leq \geq \eta)$ g

assumes depends-on f J depends-on g (I-J)

assumes [measurable]: $f \in \text{borel-measurable } (Pi_M J (\lambda \cdot. \text{borel}))$

assumes [measurable]: $g \in \text{borel-measurable } (Pi_M (I-J) (\lambda \cdot. \text{borel}))$

shows $(\int \omega. f (\lambda i. X i \omega) * g (\lambda i. X i \omega) \partial M) \leq (\int x. f(\lambda y. X y x) \partial M) * (\int x. g(\lambda y. X y x) \partial M)$

(is $?L \leq ?R$)

proof -

let $?cf = \lambda n x. \text{clamp } (-\text{real } n) (\text{real } n) (f x)$

let $?cg = \lambda n x. \text{clamp } (-\text{real } n) (\text{real } n) (g x)$

note [measurable] = neg-assoc-imp-measurable[OF assms(1)]

have m-f: random-variable borel $(\lambda x. f(\lambda i. X i x))$

using subsetD[OF assms(2)] by (subst depends-onD[OF assms(7)]) measurable

have m-g: random-variable borel $(\lambda x. g(\lambda i. X i x))$

by (subst depends-onD[OF assms(8)]) measurable

note intro-rules = borel-measurable-times measurable-compose[OF - clamp-borel]

AE-I2

measurable-compose[OF - borel-measurable-norm] *lim-clamp-n m-f m-g*

have $a: (\lambda n. (\int \omega. ?cf\ n\ (\lambda i. X\ i\ \omega) * ?cg\ n\ (\lambda i. X\ i\ \omega)\ \partial M)) \longrightarrow ?L$ **using** *assms*(3,4)

by (*intro integral-dominated-convergence*[**where** $w = \lambda \omega. norm\ (f(\lambda i. X\ i\ \omega)) * norm\ (g(\lambda i. X\ i\ \omega))$]

intro-rules tendsto-mult cauchy-schwartz(1)[**where** $M = M$]

(*auto intro!*: *clamp-abs-le mult-mono simp add:comp-def abs-mult*)

have $(\lambda n. (\int x. ?cf\ n\ (\lambda y. X\ y\ x)\ \partial M)) \longrightarrow (\int x. f(\lambda y. X\ y\ x)\ \partial M)$

using *square-integrable-imp-integrable*[OF m-f] *assms*(3) **unfolding** *comp-def*

by (*intro integral-dominated-convergence*[**where** $w = \lambda \omega. norm\ (f(\lambda i. X\ i\ \omega))$]

intro-rules)

(*simp-all add:clamp-abs-le*)

moreover have $(\lambda n. (\int x. ?cg\ n\ (\lambda y. X\ y\ x)\ \partial M)) \longrightarrow (\int x. g(\lambda y. X\ y\ x)\ \partial M)$

using *square-integrable-imp-integrable*[OF m-g] *assms*(4) **unfolding** *comp-def*

by (*intro integral-dominated-convergence*[**where** $w = \lambda \omega. norm\ (g(\lambda i. X\ i\ \omega))$]

intro-rules)

(*simp-all add:clamp-abs-le*)

ultimately have $b: (\lambda n. (\int x. ?cf\ n\ (\lambda y. X\ y\ x)\ \partial M) * (\int x. ?cg\ n\ (\lambda y. X\ y\ x)\ \partial M)) \longrightarrow ?R$

by (*rule tendsto-mult*)

show *?thesis*

by (*intro lim-mono*[OF - a b, **where** $N = 0$] *bounded-clamp-alt assms*(5,6,9,10) *monotone-clamp*

neg-assoc-imp-mult-mono-bounded[OF *assms*(1,2), **where** $\eta = \eta$] *depends-on-comp-2*[OF *assms*(7)]

measurable-compose[OF - clamp-borel] *depends-on-comp-2*[OF *assms*(8)])

qed

Property P4 [13]

lemma *neg-assoc-subset*:

assumes $J \subseteq I$

assumes *neg-assoc X I*

shows *neg-assoc X J*

proof (*rule neg-assocI,goal-cases*)

case (1 i)

then show *?case using neg-assoc-imp-measurable*[OF *assms*(2)] *assms*(1) **by** *auto*

next

case (2 f g K)

have $a: K \subseteq I$ **using** 2 *assms*(1) **by** *auto*

have $g = g \circ (\lambda m. restrict\ m\ (J - K))$

using 2 *depends-onD unfolding comp-def by (intro ext) auto*

also have ... \in *borel-measurable* ($Pi_M (I - K) (\lambda-. \text{borel})$)
using 2 *assms(1)* **by** (*intro measurable-comp*[*OF measurable-restrict-subset*])
auto
finally have $g \in$ *borel-measurable* ($Pi_M (I - K) (\lambda-. \text{borel})$) **by** *simp*
moreover have *depends-on* $g (I-K)$ **using** *depends-on-mono* *assms(1)* 2
by (*metis Diff-mono dual-order.eq-iff*)
ultimately show *covariance* ($f \circ \text{flip } X$) ($g \circ \text{flip } X$) ≤ 0
using 2 **by** (*intro neg-assoc-imp-mult-mono-bounded*[*OF assms(2)*] *a*, **where**
 $\eta = \text{Fwd}$) *simp-all*
qed

lemma *neg-assoc-imp-mult-mono-nonneg*:

fixes $f g :: ('i \Rightarrow 'c::\text{linorder-topology}) \Rightarrow \text{real}$
assumes *neg-assoc* $X I J \subseteq I$
assumes *range* $f \subseteq \{0..\}$ *range* $g \subseteq \{0..\}$
assumes *integrable* $M (f \circ \text{flip } X)$ *integrable* $M (g \circ \text{flip } X)$
assumes *monotone* (\leq) ($\leq \geq \eta$) f *monotone* (\leq) ($\leq \geq \eta$) g
assumes *depends-on* $f J$ *depends-on* $g (I-J)$
assumes $f \in$ *borel-measurable* ($Pi_M J (\lambda-. \text{borel})$) $g \in$ *borel-measurable* (Pi_M
 $(I-J) (\lambda-. \text{borel})$)

shows *has-int-that* $M (\lambda\omega. f (\text{flip } X \omega) * g (\text{flip } X \omega))$
 $(\lambda r. r \leq \text{expectation } (f \circ \text{flip } X) * \text{expectation } (g \circ \text{flip } X))$

proof –

let $?cf = (\lambda n x. \text{min } (\text{real } n) (f x))$
let $?cg = (\lambda n x. \text{min } (\text{real } n) (g x))$

define u **where** $u = (\lambda\omega. f (\lambda i. X i \omega) * g (\lambda i. X i \omega))$
define h **where** $h n \omega = ?cf n (\lambda i. X i \omega) * ?cg n (\lambda i. X i \omega)$ **for** $n \omega$
define x **where** $x = (\text{SUP } n. \text{expectation } (h n))$

note *borel-intros* = *borel-measurable-times* *borel-measurable-const* *borel-measurable-min*
borel-measurable-power

note *bounded-intros'* = *integrable-bounded* *bounded-intros* *bounded-const-min*

have *f-meas*: *random-variable* *borel* ($\lambda x. f (\lambda i. X i x)$)
using *borel-measurable-integrable*[*OF assms(5)*] **by** (*simp add: comp-def*)
have *g-meas*: *random-variable* *borel* ($\lambda x. g (\lambda i. X i x)$)
using *borel-measurable-integrable*[*OF assms(6)*] **by** (*simp add: comp-def*)

have *h-int*: *integrable* $M (h n)$ **for** n
unfolding *h-def* **using** *assms(3,4)* **by** (*intro bounded-intros'* *borel-intros* *f-meas*
g-meas) *fast+*

have *exp-h-le-R*: *expectation* ($h n$) \leq *expectation* ($f \circ \text{flip } X$) * *expectation* ($g \circ \text{flip}$
 X) **for** n

proof –

have *square-integrable* $M ((\lambda a. \text{min } (\text{real } n) (f a)) \circ (\lambda x y. X y x))$
using *assms(3)* **unfolding** *comp-def*
by (*intro bounded-intros'* *bdd-belowI*[**where** $m=0$] *borel-intros* *f-meas*) *auto*

moreover have *square-integrable* M $((\lambda a. \min (\text{real } n) (g a)) \circ (\lambda x y. X y x))$
using *assms(4)* **unfolding** *comp-def*
by $(\text{intro } \text{bounded-intros}' \text{ bdd-belowI}[\text{where } m=0] \text{ borel-intros } g\text{-meas}) \text{ auto}$
moreover have *monotone* $(\leq) (\leq_{\geq \eta}) ((\lambda a. \min (\text{real } n) (f a)))$
using *monotoneD[OF assms(7)]* **unfolding** *comp-def min-mult-distrib-left*
by $(\text{intro } \text{monotoneI}) (\text{cases } \eta, \text{fastforce+})$
moreover have *monotone* $(\leq) (\leq_{\geq \eta}) ((\lambda a. \min (\text{real } n) (g a)))$
using *monotoneD[OF assms(8)]* **unfolding** *comp-def min-mult-distrib-left*
by $(\text{intro } \text{monotoneI}) (\text{cases } \eta, \text{fastforce+})$
ultimately have *expectation* $(h n) \leq \text{expectation } (?cf \text{ n}\circ\text{flip } X) * \text{expectation}$
 $(?cg \text{ n}\circ\text{flip } X)$
unfolding *h-def comp-def*
by $(\text{intro } \text{neg-assoc-imp-mult-mono}[OF \text{ assms}(1-2)] \text{ borel-intros } \text{assms}(11,12)$
 $\text{depends-on-comp-2}[OF \text{ assms}(10)] \text{ depends-on-comp-2}[OF \text{ assms}(9)]) (\text{auto}$
simp:comp-def)
also have $\dots \leq \text{expectation } (f\circ\text{flip } X) * \text{expectation } (g\circ\text{flip } X)$
using *assms(3,4)* **by** $(\text{intro } \text{mult-mono } \text{integral-nonneg-AE } \text{AE-I2 } \text{inte-}$
 $\text{gral-mono}' \text{ assms}(5,6)) \text{ auto}$
finally show *?thesis* **by** *simp*
qed

have *h-mono-ptw*: $\text{AE } \omega \text{ in } M. \text{ mono } (\lambda n. h n \omega)$
using *assms(3,4)* **unfolding** *h-def* **by** $(\text{intro } \text{AE-I2 } \text{monoI } \text{mult-mono}) \text{ auto}$
have *h-mono*: $\text{mono } (\lambda n. \text{expectation } (h n))$
by $(\text{intro } \text{monoI } \text{integral-mono-AE } \text{AE-mp}[OF \text{ h-mono-ptw } \text{AE-I2}] \text{ h-int}) (\text{simp}$
 $\text{add:mono-def})$

have *random-variable borel* u **using** *f-meas g-meas* **unfolding** *u-def* **by** $(\text{intro}$
 $\text{borel-intros})$
moreover have $\text{AE } \omega \text{ in } M. (\lambda n. h n \omega) \longrightarrow u \omega$
unfolding *h-def u-def* **by** $(\text{intro } \text{tendsto-mult } \text{lim-min-n } \text{AE-I2})$
moreover have *bdd-above* $(\text{range } (\lambda n. \text{expectation } (h n)))$
using *exp-h-le-R* **by** $(\text{intro } \text{bdd-aboveI}) \text{ auto}$
hence $(\lambda n. \text{expectation } (h n)) \longrightarrow x$
using *LIMSEQ-incseq-SUP[OF - h-mono]* **unfolding** *x-def* **by** *simp*
ultimately have *has-bochner-integral* $M u x$ **using** *h-int h-mono-ptw*
by $(\text{intro } \text{has-bochner-integral-monotone-convergence}[\text{where } f=h])$
moreover have $x \leq \text{expectation } (f\circ\text{flip } X) * \text{expectation } (g\circ\text{flip } X)$
unfolding *x-def* **by** $(\text{intro } \text{cSUP-least } \text{exp-h-le-R}) \text{ simp}$
ultimately show *?thesis* **unfolding** *has-bochner-integral-iff u-def has-int-that-def*
by *auto*
qed

Property P2 [13]

lemma *neg-assoc-imp-prod-mono*:
fixes $f :: 'i \Rightarrow ('c::\text{linorder-topology}) \Rightarrow \text{real}$
assumes *finite* I
assumes *neg-assoc* $X I$
assumes $\bigwedge i. i \in I \implies \text{integrable } M (\lambda \omega. f i (X i \omega))$

assumes $\bigwedge i. i \in I \implies \text{monotone } (\leq) (\leq_{\geq \eta}) (f i)$
assumes $\bigwedge i. i \in I \implies \text{range } (f i) \subseteq \{0..\}$
assumes $\bigwedge i. i \in I \implies f i \in \text{borel-measurable borel}$
shows *has-int-that* $M (\lambda \omega. (\prod i \in I. f i (X i \omega))) (\lambda r. r \leq (\prod i \in I. \text{expectation } (\lambda \omega. f i (X i \omega))))$
using *assms*
proof (*induction I rule:finite-induct*)
case *empty then show ?case by (simp add:has-int-that-def)*
next
case (*insert x F*)

define *g* **where** $g v = f x (v x)$ **for** v
define *h* **where** $h v = (\prod i \in F. f i (v i))$ **for** v

have $0: \{x\} \subseteq \text{insert } x F$ **by** *auto*

have *ran-g: range g* $\subseteq \{0..\}$ **using** *insert(7) unfolding g-def by auto*

have $\text{True} = \text{has-int-that } M (\lambda \omega. \prod i \in F. f i (X i \omega)) (\lambda r. r \leq (\prod i \in F. \text{expectation } (\lambda \omega. f i (X i \omega))))$
by (*intro true-eq-iff insert neg-assoc-subset[OF - insert(4)] auto*)
also have $\dots = \text{has-int-that } M (h \circ \text{flip } X) (\lambda r. r \leq (\prod i \in F. \text{expectation } (\lambda \omega. f i (X i \omega))))$
unfolding *h-def by (intro arg-cong2[where f=has-int-that M] refl)(simp add:comp-def)*
finally have $2: \text{has-int-that } M (h \circ \text{flip } X) (\lambda r. r \leq (\prod i \in F. \text{expectation } (\lambda \omega. f i (X i \omega))))$
by *simp*

have $(\prod i \in F. f i (v i)) \geq 0$ **for** v **using** *insert(7) by (intro prod-nonneg) auto*
hence $\text{range } h \subseteq \{0..\}$ **unfolding** *h-def by auto*
moreover have *integrable M (g o flip X) unfolding g-def using insert(5) by (auto simp:comp-def)*
moreover have $3: \text{monotone } (\leq) (\leq_{\geq \eta}) (f x)$ **using** *insert(6) by simp*
have $\text{monotone } (\leq) (\leq_{\geq \eta}) g$ **using** *monotoneD[OF 3]*
unfolding *g-def by (intro monotoneI) (auto simp:comp-def le-fun-def)*
moreover have $4: \text{monotone } (\leq) (\leq_{\geq \eta}) (f i) \wedge x. f i x \geq 0$ **if** $i \in F$ **for** i
using *that insert(6,7) by force+*
hence $\text{monotone } (\leq) (\leq_{\geq \eta}) h$ **using** *monotoneD[OF 4(1)] unfolding h-def*
by (*intro monotoneI (cases η, auto intro:prod-mono simp:comp-def le-fun-def)*)
moreover have *depends-on g {x} unfolding g-def by (intro depends-onI) force*
moreover have *depends-on h F*
unfolding *h-def by (intro depends-onI prod.cong refl) force*
hence *depends-on h (F - {x}) using insert(2) by simp*
moreover have $g \in \text{borel-measurable } (Pi_M \{x\}) (\lambda-. \text{borel})$ **unfolding** *g-def*
by (*intro measurable-compose[OF - insert(8)] measurable-component-singleton*)
auto
moreover have $h \in \text{borel-measurable } (Pi_M F (\lambda-. \text{borel}))$
unfolding *h-def by (intro borel-measurable-prod measurable-compose[OF - in-*

$sert(8)$
measurable-component-singleton *auto*
hence $h \in \text{borel-measurable } (Pi_M (F - \{x\}) (\lambda-. \text{borel}))$ **using** *insert(2)* **by**
simp
ultimately have $\text{True} = \text{has-int-that } M (\lambda\omega. g (\text{flip } X \omega) * h (\text{flip } X \omega))$
 $(\lambda r. r \leq \text{expectation } (g \circ \text{flip } X) * \text{expectation } (h \circ \text{flip } X))$
by (*intro true-eq-iff neg-assoc-imp-mult-mono-nonneg[OF insert(4)] 0*, **where**
 $\eta = \eta$)
ran-g has-int-thatD[OF 2] *simp-all*
also have $\dots = \text{has-int-that } M (\lambda\omega. (\prod_{i \in \text{insert } x F. f i (X i \omega))})$
 $(\lambda r. r \leq \text{expectation } (g \circ \text{flip } X) * \text{expectation } (h \circ \text{flip } X))$
unfolding *g-def h-def* **using** *insert(1,2)* **by** (*intro arg-cong2[where f=has-int-that*
M] refl) *simp*
also have $\dots \leq \text{has-int-that } M (\lambda\omega. (\prod_{i \in \text{insert } x F. f i (X i \omega))})$
 $(\lambda r. r \leq \text{expectation } (g \circ \text{flip } X) * (\prod_{i \in F. \text{expectation } (f i \circ X i))})$
using *ran-g has-int-thatD[OF 2]* **by** (*intro has-int-that-mono le-trans mult-left-mono*
integral-nonneg-AE AE-I2) (*auto simp: comp-def*)
also have $\dots = \text{has-int-that } M$
 $(\lambda\omega. \prod_{i \in \text{insert } x F. f i (X i \omega)) (\lambda r. r \leq (\prod_{i \in \text{insert } x F. \text{expectation } (f i \circ$
 $X i))$)
using *insert(1,2)* **by** (*intro arg-cong2[where f=has-int-that M] refl*) (*simp*
add:g-def comp-def)
finally show *?case* **using** *le-boolE* **by** (*simp add:comp-def*)
qed

Property P5 [13]

lemma *neg-assoc-compose*:

fixes $f :: 'j \Rightarrow ('i \Rightarrow ('c :: \text{linorder-topology})) \Rightarrow ('d :: \text{linorder-topology})$
assumes *finite I*
assumes *neg-assoc X I*
assumes $\bigwedge j. j \in J \implies \text{deps } j \subseteq I$
assumes $\bigwedge j1 j2. j1 \in J \implies j2 \in J \implies j1 \neq j2 \implies \text{deps } j1 \cap \text{deps } j2 = \{\}$
assumes $\bigwedge j. j \in J \implies \text{monotone } (\leq) (\leq_{\geq \eta}) (f j)$
assumes $\bigwedge j. j \in J \implies \text{depends-on } (f j) (\text{deps } j)$
assumes $\bigwedge j. j \in J \implies f j \in \text{borel-measurable } (Pi_M (\text{deps } j) (\lambda-. \text{borel}))$
shows *neg-assoc* $(\lambda j \omega. f j (\lambda i. X i \omega)) J$
proof (*rule neg-assocI, goal-cases*)
case (1 *i*)
note [*measurable*] = *neg-assoc-imp-measurable[OF assms(2)] assms(7)[OF 1]*
note $\mathcal{B} = \text{assms}(3)[OF 1]$
have 2: $f i (\lambda i. X i \omega) = f i (\lambda i \in \text{deps } i. X i \omega)$ **for** ω
using \mathcal{B} **by** (*intro depends-onD2[OF assms(6)] 1 fastforce*)
show *?case* **unfolding** 2 **by** *measurable (rule subsetD[OF 3])*
next
case (2 *g h K*)

let $?g = (\lambda\omega. g (\lambda j. f j \omega))$
let $?h = (\lambda\omega. h (\lambda j. f j \omega))$

note $dep-f = depends-onD[OF depends-on-mono[OF - assms(6)],symmetric]$

have $g-alt-1: ?g = (\lambda\omega. g (\lambda j \in J. f j \omega))$
using $2(1)$ **by** $(intro\ ext\ depends-onD[OF\ depends-on-mono[OF - 2(2)]])\ auto$

have $g-alt-2: ?g = (\lambda\omega. g (\lambda j \in K. f j \omega))$
by $(intro\ ext\ depends-onD[OF\ 2(2)])$

have $g-alt-3: ?g = (\lambda\omega. g (\lambda j \in K. f j (restrict\ \omega\ (deps\ j))))$ **unfolding** $g-alt-2$
using $2(1)$
by $(intro\ restrict-ext\ ext\ arg-cong[where\ f=g]\ depends-onD[OF\ assms(6)])\ auto$

have $h-alt-1: ?h = (\lambda\omega. h (\lambda j \in J. f j \omega))$
by $(intro\ ext\ depends-onD[OF\ depends-on-mono[OF - 2(3)]])\ auto$

have $h-alt-2: ?h = (\lambda\omega. h (\lambda j \in J-K. f j \omega))$
by $(intro\ ext\ depends-onD[OF\ 2(3)])$

have $h-alt-3: ?h = (\lambda\omega. h (\lambda j \in J-K. f j (restrict\ \omega\ (deps\ j))))$ **unfolding** $h-alt-2$
by $(intro\ restrict-ext\ ext\ arg-cong[where\ f=h]\ depends-onD[OF\ assms(6)])\ auto$

have $3: \bigcup (deps\ ' (J-K)) \subseteq I - \bigcup (deps\ ' K)$ **using** $assms(3,4)$ $2(1)$ **by** $blast$

have $\bigcup (deps\ ' K) \subseteq I$ **using** $2(1)$ $assms(3)$ **by** $auto$

moreover **have** $bounded\ (range\ ?g)\ bounded\ (range\ ?h)$
using $2(6,7)$ **by** $(auto\ intro: bounded-subset)$

moreover **have** $monotone\ (\le)\ (\leq_{\eta})\ ?g$
unfolding $g-alt-1$ **using** $monotoneD[OF\ assms(5)]$
by $(intro\ monotoneI)\ (cases\ \eta,\ auto\ intro!:monoD[OF\ 2(4)]\ le-funI)$

moreover **have** $monotone\ (\le)\ (\leq_{\eta})\ ?h$
unfolding $h-alt-1$ **using** $monotoneD[OF\ assms(5)]$
by $(intro\ monotoneI)\ (cases\ \eta,\ auto\ intro!:monoD[OF\ 2(5)]\ le-funI)$

moreover **have** $depends-on\ ?g\ (\bigcup (deps\ ' K))$
using $2(1)$ **unfolding** $g-alt-2$
by $(intro\ depends-onI\ arg-cong[where\ f=g]\ restrict-ext\ depends-onD2[OF\ assms(6)])\ auto$

moreover **have** $depends-on\ ?h\ (\bigcup (deps\ ' (J-K)))$
unfolding $h-alt-2$
by $(intro\ depends-onI\ arg-cong[where\ f=h]\ restrict-ext\ depends-onD2[OF\ assms(6)])\ auto$

hence $depends-on\ ?h\ (I - \bigcup (deps\ ' K))$ **using** $depends-on-mono[OF\ 3]$ **by** $auto$

moreover **have** $?g \in borel-measurable\ (Pi_M\ (\bigcup (deps\ ' K))\ (\lambda-. borel))$
unfolding $g-alt-3$ **using** $2(1)$
by $(intro\ measurable-compose[OF - 2(8)]\ measurable-compose[OF - assms(7)]\ measurable-restrict\ measurable-component-singleton)\ auto$

moreover **have** $?h \in borel-measurable\ (Pi_M\ (I - \bigcup (deps\ ' K))\ (\lambda-. borel))$
unfolding $h-alt-3$ **using** 3
by $(intro\ measurable-compose[OF - 2(9)]\ measurable-compose[OF - assms(7)]\ measurable-restrict\ measurable-component-singleton)\ auto$

ultimately **have** $covariance\ (?g \circ flip\ X)\ (?h \circ flip\ X) \leq 0$

by (rule neg-assoc-imp-mult-mono-bounded[OF assms(2), where $J = \bigcup (\text{deps } K)$ and $\eta = \eta$])
 thus covariance $(g \circ (\lambda x y. f y (\lambda i. X i x))) (h \circ (\lambda x y. f y (\lambda i. X i x))) \leq 0$
 by (simp add: comp-def)
 qed

lemma neg-assoc-compose-simple:

fixes $f :: 'i \Rightarrow ('c :: \text{linorder-topology}) \Rightarrow ('d :: \text{linorder-topology})$
 assumes finite I
 assumes neg-assoc $X I$
 assumes $\bigwedge i. i \in I \implies \text{monotone } (\leq) (\leq_{\eta}) (f i)$
 assumes [measurable]: $\bigwedge i. i \in I \implies f i \in \text{borel-measurable borel}$
 shows neg-assoc $(\lambda i \omega. f i (X i \omega)) I$
proof –
 have depends-on $(\lambda \omega. f i (\omega i)) \{i\}$ if $i \in I$ for i
 by (intro depends-onI) auto
 moreover have monotone $(\leq) (\leq_{\eta}) (\lambda \omega. f i (\omega i))$ if $i \in I$ for i
 using monotoneD[OF assms(3)[OF that]] by (intro monotoneI) (cases η , auto dest: le-funE)
 ultimately show ?thesis
 by (intro neg-assoc-compose[OF assms(1,2), where $\text{deps} = \lambda i. \{i\}$ and $\eta = \eta$])
 simp-all
 qed

lemma covariance-distr:

fixes $f g :: 'b \Rightarrow \text{real}$
 assumes [measurable]: $\varphi \in M \rightarrow_M N$ $f \in \text{borel-measurable } N$ $g \in \text{borel-measurable } N$
 shows prob-space.covariance $(\text{distr } M N \varphi) f g = \text{covariance } (f \circ \varphi) (g \circ \varphi)$ (is ?L = ?R)
proof –
 let ?M' = $\text{distr } M N \varphi$
 have ps-distr: prob-space ?M' by (intro prob-space-distr) measurable
 interpret p2: prob-space ?M'
 using ps-distr by auto
 have ?L = expectation $(\lambda x. (f(\varphi x) - \text{expectation } (\lambda x. f(\varphi x))) * (g(\varphi x) - \text{expectation } (\lambda x. g(\varphi x))))$
 unfolding p2.covariance-def by (subst (1 2 3) integral-distr) measurable
 also have ... = ?R
 unfolding covariance-def comp-def by simp
 finally show ?thesis by simp
 qed

lemma neg-assoc-iff-distr:

assumes [measurable]: $\bigwedge i. i \in I \implies X i \in \text{borel-measurable } M$
 shows neg-assoc $X I \iff$
 prob-space.neg-assoc $(\text{distr } M (PiM I (\lambda-. \text{borel})) (\lambda \omega. \lambda i \in I. X i \omega)) (\text{flip id}) I$
 (is ?L \iff ?R)
proof

```

let ?M' = distr M (Pi_M I (λ-. borel)) (λω. λi∈I. X i ω)
have ps-distr: prob-space ?M'
  by (intro prob-space-distr) measurable

interpret p2: prob-space ?M'
  using ps-distr by auto

show ?R if ?L
proof (rule p2.neg-assocI, goal-cases)
  case (1 i)
  thus ?case using assms that unfolding id-def by measurable
next
  case (2 f g J)

  have dep-I: depends-on f I depends-on g I
    using depends-on-mono[OF Diff-subset[of I J]] depends-on-mono[OF 2(1)]
  2(2-3) by auto

  have f-meas[measurable]: (λx. f x) ∈ borel-measurable (Pi_M I (λ-. borel))
    by (subst depends-onD[OF 2(2)]) (intro 2 measurable-compose[OF measurable-restrict-subset])

  have g-meas[measurable]: (λx. g x) ∈ borel-measurable (Pi_M I (λ-. borel))
    by (subst depends-onD[OF 2(3)])
    (intro 2 measurable-compose[OF measurable-restrict-subset], auto)

  have covariance (f ∘ id ∘ (λω. λi∈I. X i ω)) (g ∘ id ∘ (λω. λi∈I. X i ω)) =
    covariance (f ∘ flip X) (g ∘ flip X)
  using depends-onD[OF dep-I(2)] depends-onD[OF dep-I(1)] by (simp add:comp-def)
  also have ... ≤ 0
  using 2 by (intro neg-assoc-imp-mult-mono-bounded[OF that 2(1,6,7), where
  η=Fwd]) simp-all
  finally have covariance (f ∘ id ∘ (λω. λi∈I. X i ω)) (g ∘ id ∘ (λω. λi∈I. X i
  ω)) ≤ 0 by simp
  thus ?case by (subst covariance-distr) measurable
qed

show ?L if ?R
proof (rule neg-assocI, goal-cases)
  case (1 i)
  then show ?case by measurable
next
  case (2 f g J)

  have dep-I: depends-on f I depends-on g I
    using depends-on-mono[OF Diff-subset[of I J]] depends-on-mono[OF 2(1)]
  2(2-3) by auto

  have f-meas[measurable]: (λx. f x) ∈ borel-measurable (Pi_M I (λ-. borel))

```

by (*subst depends-onD*[*OF 2(2)*]) (*intro 2 measurable-compose*[*OF measurable-restrict-subset*])

have *g-meas*[*measurable*]: $(\lambda x. g x) \in \text{borel-measurable } (Pi_M I (\lambda-. \text{borel}))$
by (*subst depends-onD*[*OF 2(3)*])
(*intro 2 measurable-compose*[*OF measurable-restrict-subset*], *auto*)

note [*measurable*] = *2(8,9)*

have *covariance* ($f \circ (\lambda x y. X y x)$) ($g \circ (\lambda x y. X y x)$) =
covariance ($f \circ (\lambda \omega. \lambda i \in I. X i \omega)$) ($g \circ (\lambda \omega. \lambda i \in I. X i \omega)$)

using *depends-onD*[*OF dep-I(2)*] *depends-onD*[*OF dep-I(1)*] **by** (*simp add:comp-def*)

also have ... = *p2.covariance* ($f \circ id$) ($g \circ id$) **by** (*subst covariance-distr*)

measurable

also have ... ≤ 0

using 2 by (*intro p2.neg-assoc-imp-mult-mono-bounded*[*OF that 2(1), where $\eta = Fwd$*])

(*simp-all add:comp-def*)

finally show ?*case* **by** *simp*

qed

qed

lemma *neg-assoc-cong*:

assumes *finite I*

assumes [*measurable*]: $\bigwedge i. i \in I \implies Y i \in \text{borel-measurable } M$

assumes *neg-assoc X I* $\bigwedge i. i \in I \implies AE \omega \text{ in } M. X i \omega = Y i \omega$

shows *neg-assoc Y I*

proof –

have [*measurable*]: $\bigwedge i. i \in I \implies X i \in \text{borel-measurable } M$

using *neg-assoc-imp-measurable*[*OF assms(3)*] **by** *auto*

let ?*B* = $(\lambda-. \text{borel})$

have *a:AE x in M. ($\forall i \in I. (X i x = Y i x)$)* **by** (*intro AE-finite-allI assms*)

have *AE x in M. ($\lambda i \in I. X i x = \lambda i \in I. Y i x$)* **by** (*intro AE-mp*[*OF a AE-I2*])

auto

hence *b:distr M (PiM I ?B) ($\lambda \omega. \lambda i \in I. X i \omega = \text{distr } M (PiM I ?B) (\lambda \omega. \lambda i \in I. Y i \omega)$)*

by (*intro distr-cong-AE refl*) *measurable*

have *prob-space.neg-assoc (distr M (PiM I ($\lambda-. \text{borel}$)) ($\lambda \omega. \lambda i \in I. X i \omega$)) (flip id) I*

using *assms(2,3)* **by** (*intro iffD1*[*OF neg-assoc-iff-distr*]) *measurable*

thus ?*thesis* **unfolding** *b* **using** *assms(2)*

by (*intro iffD2*[*OF neg-assoc-iff-distr*[**where** $I=I$]]) *auto*

qed

lemma *neg-assoc-reindex-aux*:

assumes *inj-on h I*

assumes *neg-assoc X (h ' I)*

shows *neg-assoc ($\lambda k. X (h k)$) I*

proof (*rule neg-assocI, goal-cases*)

```

case (1 i) thus ?case using neg-assoc-imp-measurable[OF assms(2)] by simp
next
case (2 f g J)
let ?f = (λω. f (compose J ω h))
let ?g = (λω. g (compose (I-J) ω h))

note neg-assoc-imp-mult-mono-intros =
  neg-assoc-imp-mult-mono-bounded(1)[OF assms(2), where J=h'J and η=Fwd]
  measurable-compose[OF - 2(8)] measurable-compose[OF - 2(9)]
  measurable-compose[OF - measurable-finmap-compose]
  bounded-range-imp[OF 2(6)] bounded-range-imp[OF 2(7)]

have [simp]:h ' I - h ' J = h ' (I-J)
using assms(1) 2(1) by (simp add: inj-on-image-set-diff)

have covariance (f◦(λx y. X(h y)x)) (g◦(λx y. X(h y)x)) = covariance (?f ◦ flip
X) (?g ◦ flip X)
unfolding comp-def
by (intro arg-cong2[where f=covariance] ext depends-onD2[OF 2(2)] de-
pends-onD2[OF 2(3)])
  (auto simp:compose-def)
also have ... ≤ 0 using 2(1)
by (intro neg-assoc-imp-mult-mono-intros monotoneI depends-onI) (auto intro!:
monoD[OF 2(4)] monoD[OF 2(5)] simp:le-fun-def compose-def restrict-def
cong:if-cong)
finally show ?case by simp
qed

lemma neg-assoc-reindex:
assumes inj-on h I finite I
shows neg-assoc X (h ' I) ⟷ neg-assoc (λk. X (h k)) I (is ?L ⟷ ?R)
proof
assume ?L
thus ?R using neg-assoc-reindex-aux[OF assms(1)] by blast
next
note inv-h-inj = inj-on-the-inv-into[OF assms(1)]
assume a: ?R
hence b:neg-assoc (λk. X (h (the-inv-into I h k))) (h ' I)
using the-inv-into-onto[OF assms(1)] by (intro neg-assoc-reindex-aux[OF inv-h-inj])
auto
show ?L
using f-the-inv-into-f[OF assms(1)] neg-assoc-imp-measurable[OF a] assms(2)
by (intro neg-assoc-cong[OF - - b]) auto
qed

lemma measurable-compose-merge-1:
assumes depends-on h K
assumes h ∈ PiM K M' →M N K ⊆ I ∪ J
assumes (λx. restrict (fst (f x)) (K ∩ I)) ∈ A →M PiM (K ∩ I) M'

```

assumes $(\lambda x. \text{restrict } (\text{snd } (f x)) (K \cap J)) \in A \rightarrow_M \text{PiM } (K \cap J) M'$
shows $(\lambda x. h(\text{merge } I J (f x))) \in A \rightarrow_M N$
proof –
let $?f1 = \lambda x. \text{fst } (f x)$
let $?f2 = \lambda x. \text{snd } (f x)$
let $?g1 = \lambda x. \text{restrict } (\text{fst } (f x)) (K \cap I)$
let $?g2 = \lambda x. \text{restrict } (\text{snd } (f x)) (K \cap J)$

have $a1: (\lambda x. \text{merge } I J (?g1 x, ?g2 x) i) \in A \rightarrow_M M' i$ **if** $i \in K \cap I$ **for** i
using *that measurable-compose[OF assms(4) measurable-component-singleton[OF that]]*
by (*simp add:merge-def*)

have $a2: (\lambda x. \text{merge } I J (?g1 x, ?g2 x) i) \in A \rightarrow_M M' i$ **if** $i \in K \cap J$ **if** $i \notin I$ **for** i
using *that measurable-compose[OF assms(5) measurable-component-singleton[OF that(1)]]*
by (*simp add:merge-def*)

have $a: (\lambda x. \text{merge } I J (?g1 x, ?g2 x) i) \in A \rightarrow_M M' i$ **if** $i \in K$ **for** i
using *assms(3) a1 a2 that by auto*

have $(\lambda x. h(\text{merge } I J (f x))) = (\lambda x. h(\text{merge } I J (?f1 x, ?f2 x)))$ **by** *simp*
also have $\dots = (\lambda x. h(\lambda i \in K. \text{merge } I J (?f1 x, ?f2 x) i))$
using *depends-onD[OF assms(1)] by simp*
also have $\dots = (\lambda x. h(\lambda i \in K. \text{merge } I J (?g1 x, ?g2 x) i))$
by (*intro ext arg-cong[where f=h] (auto simp:comp-def restrict-def merge-def case-prod-beta)*)
also have $\dots \in A \rightarrow_M N$
by (*intro measurable-compose[OF - assms(2)] measurable-restrict a*)
finally show *?thesis by simp*
qed

lemma *measurable-compose-merge-2:*

assumes *depends-on* $h K h \in \text{PiM } K M' \rightarrow_M N K \subseteq I \cup J$
assumes $(\lambda x. \text{restrict } (f x) (K \cap I)) \in A \rightarrow_M \text{PiM } (K \cap I) M'$
assumes $(\lambda x. \text{restrict } (g x) (K \cap J)) \in A \rightarrow_M \text{PiM } (K \cap J) M'$
shows $(\lambda x. h(\text{merge } I J (f x, g x))) \in A \rightarrow_M N$
using *assms by (intro measurable-compose-merge-1[OF assms(1–3)]) simp-all*

lemma *neg-assoc-combine:*

fixes $I I1 I2 :: 'i \text{ set}$
fixes $X :: 'i \Rightarrow 'a \Rightarrow ('b::\text{linorder-topology})$
assumes *finite* $I I1 \cup I2 = I I1 \cap I2 = \{\}$
assumes *indep-var* $(\text{PiM } I1 (\lambda \cdot. \text{borel})) (\lambda \omega. \lambda i \in I1. X i \omega) (\text{PiM } I2 (\lambda \cdot. \text{borel}))$
 $(\lambda \omega. \lambda i \in I2. X i \omega)$
assumes *neg-assoc* $X I1$
assumes *neg-assoc* $X I2$
shows *neg-assoc* $X I$
proof –

define X' **where** $X' i = (if\ i \in I\ then\ X\ i\ else\ (\lambda\ -. \text{undefined}))$ **for** i

have X -measurable: random-variable borel $(X\ i)$ **if** $i \in I$ **for** i
using that $assms(2)$ neg-assoc-imp-measurable[$OF\ assms(5)$]
neg-assoc-imp-measurable[$OF\ assms(6)$] **by** auto

have $rv[measurable]$: random-variable borel $(X' i)$ **for** i
unfolding X' -def **using** X -measurable **by** auto

have $na-I1$: neg-assoc $X' I1$ **using** neg-assoc-cong
unfolding X' -def **using** $assms(1,2)$ neg-assoc-imp-measurable[$OF\ assms(5)$]
by (intro neg-assoc-cong[$OF - - assms(5)$] $AE-I2$) auto

have $na-I2$: neg-assoc $X' I2$ **using** neg-assoc-cong
unfolding X' -def **using** $assms(1,2)$ neg-assoc-imp-measurable[$OF\ assms(6)$]
by (intro neg-assoc-cong[$OF - - assms(6)$] $AE-I2$) auto

have iv : indep-var($PiM\ I1\ (\lambda\ -. \text{borel})$)($\lambda\omega. \lambda i \in I1. X' i\ \omega$)($PiM\ I2\ (\lambda\ -. \text{borel})$)($\lambda\omega. \lambda i \in I2. X' i\ \omega$)
using $assms(2,4)$ **unfolding** indep-var-def X' -def **by** (auto simp add: restrict-def cong: if-cong)

let $?N = PiM\ I1\ (\lambda\ -. \text{borel}) \otimes_M PiM\ I2\ (\lambda\ -. \text{borel})$
let $?A = distr\ M\ (PiM\ I1\ (\lambda\ -. \text{borel}))\ (\lambda\omega. \lambda i \in I1. X' i\ \omega)$
let $?B = distr\ M\ (PiM\ I2\ (\lambda\ -. \text{borel}))\ (\lambda\omega. \lambda i \in I2. X' i\ \omega)$
let $?H = distr\ M\ ?N\ (\lambda\omega. (\lambda i \in I1. X' i\ \omega, \lambda i \in I2. X' i\ \omega))$

have indep: $?H = (?A \otimes_M ?B)$
and rvs : random-variable ($PiM\ I1\ (\lambda\ -. \text{borel})$)($\lambda\omega. \lambda i \in I1. X' i\ \omega$)
random-variable ($PiM\ I2\ (\lambda\ -. \text{borel})$)($\lambda\omega. \lambda i \in I2. X' i\ \omega$)
using iffD1[$OF\ indep\text{-var}\text{-distribution}\text{-eq}\ iv$] **by** auto

interpret pa : prob-space $?A$ **by** (intro prob-space-distr rvs)
interpret pb : prob-space $?B$ **by** (intro prob-space-distr rvs)
interpret pair-sigma-finite $?A\ ?B$
using pa .sigma-finite-measure pb .sigma-finite-measure **by** (intro pair-sigma-finite.intro)

interpret pab : prob-space ($?A \otimes_M ?B$)
by (intro prob-space-pair pa .prob-space-axioms pb .prob-space-axioms)

have pa -na: pa .neg-assoc ($\lambda x\ y. y\ x$) $I1$
using $assms(2)$ iffD1[$OF\ neg\text{-assoc}\text{-iff}\text{-distr}\ na\text{-}I1$] **by** fastforce

have pb -na: pb .neg-assoc ($\lambda x\ y. y\ x$) $I2$
using $assms(2)$ iffD1[$OF\ neg\text{-assoc}\text{-iff}\text{-distr}\ na\text{-}I2$] **by** fastforce

have na - X' : neg-assoc $X' I$
proof (rule neg-assocI2, goal-cases)
case (1 i) **thus** $?case$ **by** measurable

next
case (2 f g K)

note *bounded-intros* =
bounded-range-imp[OF 2(6)] *bounded-range-imp*[OF 2(7)] *pa.integrable-bounded*
pb.integrable-bounded *pab.integrable-bounded* *bounded-intros* *pb.finite-measure-axioms*

have [*measurable*]:
restrict x I ∈ *space* (*Pi*_M I (λ-. *borel*)) **for** x :: ('i ⇒ 'b) **and** I **by** (*simp*
add:space-PiM)

have a: K ⊆ I1 ∪ I2 **using** 2 *assms*(2) **by** *auto*
have b: I - K ⊆ I1 ∪ I2 **using** *assms*(2) **by** *auto*

note *merge-1* = *measurable-compose-merge-2*[OF 2(2,8) a] *measurable-compose-merge-2*[OF
2(3,9) b]
note *merge-2* = *measurable-compose-merge-1*[OF 2(2,8) a] *measurable-compose-merge-1*[OF
2(3,9) b]

have *merge-mono*:
merge I1 I2 (w, y) ≤ *merge* I1 I2 (x, z) **if** w ≤ x y ≤ z **for** w x y z :: 'i ⇒ 'b
using *le-funD*[OF *that*(1)] *le-funD*[OF *that*(2)] **unfolding** *merge-def* **by**
(*intro le-funI*) *auto*

have *split-h*: h ∘ *flip* X' = (λω. h (*merge* I1 I2 (λi∈I1. X' i ω, λi∈I2. X' i
ω)))
if *depends-on* h I **for** h :: - ⇒ *real*
using *assms*(2) **unfolding** *comp-def*
by (*intro ext depends-onD2*[OF *that*]) (*auto simp:restrict-def merge-def*)

have *depends-on* f I *depends-on* g I
using 2(1) **by** (*auto intro:depends-on-mono*[OF - 2(2)] *depends-on-mono*[OF
- 2(3)])
note *split* = *split-h*[OF *this*(1)] *split-h*[OF *this*(2)]

have *step-1*: (∫ y. f (*merge* I1 I2 (x, y)) * g (*merge* I1 I2 (x, y)) ∂?B) ≤
(∫ y. f (*merge* I1 I2 (x, y)) ∂ ?B) * (∫ y. g (*merge* I1 I2 (x, y)) ∂?B) (**is** ?L1
≤ ?R1)
for x
proof -
have *step1-1*: *monotone* (≤) (≤_{Fwd}) (λa. f (*merge* I1 I2 (x, a)))
unfolding *dir-le* **by** (*intro monoI monoD*[OF 2(4)] *merge-mono*) *simp*
have *step1-2*: *monotone* (≤) (≤_{Fwd}) (λa. g (*merge* I1 I2 (x, a)))
unfolding *dir-le* **by** (*intro monoI monoD*[OF 2(5)] *merge-mono*) *simp*
have *step1-3*: *depends-on* (λa. f (*merge* I1 I2 (x, a))) (K ∩ I2)
by (*subst depends-onD*[OF 2(2)])
(*auto intro:depends-onI simp:merge-def restrict-def cong:if-cong*)
have *step1-4*: *depends-on* (λa. g (*merge* I1 I2 (x, a))) (I2 - K ∩ I2)
by (*subst depends-onD*[OF 2(3)])

(auto intro:depends-onI simp:merge-def restrict-def cong:if-cong)
show ?thesis
by (intro pb.neg-assoc-imp-mult-mono-bounded(2)[OF pb-na, **where** $\eta = Fwd$
and $J = K \cap I2$]
 bounded-intros merge-1 step1-1 step1-2 step1-3 step1-4) measurable
qed

have step2-1: monotone $(\leq) (\leq_{Fwd}) (\lambda a. pb.expectation (\lambda y. f (merge I1 I2 (a,y))))$
unfolding dir-le
by (intro monoI integral-mono bounded-intros merge-1 monoD[OF 2(4)]
 merge-mono) measurable

have step2-2: monotone $(\leq) (\leq_{Fwd}) (\lambda a. pb.expectation (\lambda y. g (merge I1 I2 (a,y))))$
unfolding dir-le
by (intro monoI integral-mono bounded-intros merge-1 monoD[OF 2(5)]
 merge-mono) measurable

have step2-3: depends-on $(\lambda a. pb.expectation (\lambda y. f (merge I1 I2 (a, y)))) (K \cap I1)$
by (subst depends-onD[OF 2(2)])
 (auto intro:depends-onI simp:merge-def restrict-def cong:if-cong)

have step2-4: depends-on $(\lambda a. pb.expectation (\lambda y. g (merge I1 I2 (a, y)))) (I1 - K \cap I1)$
by (subst depends-onD[OF 2(3)])
 (auto intro:depends-onI simp:merge-def restrict-def cong:if-cong)

have $(\int \omega. (f \circ flip X') \omega * (g \circ flip X') \omega \partial M) = (\int \omega. f (merge I1 I2 \omega) * g(merge I1 I2 \omega) \partial ?H)$
unfolding split **by** (intro integral-distr[symmetric] merge-2 borel-measurable-times)
 measurable
also have $\dots = (\int \omega. f(merge I1 I2 \omega) * g(merge I1 I2 \omega) \partial (?A \otimes_M ?B))$
unfolding indep **by** simp
also have $\dots = (\int x. (\int y. f(merge I1 I2 (x,y)) * g(merge I1 I2 (x,y)) \partial ?B) \partial ?A)$
by (intro integral-fst'[symmetric] bounded-intros merge-2 borel-measurable-times)
 measurable
also have $\dots \leq (\int x. (\int y. f(merge I1 I2 (x,y)) \partial ?B) * (\int y. g(merge I1 I2 (x,y)) \partial ?B) \partial ?A)$
by (intro integral-mono-AE bounded-intros step-1 AE-I2 pb.borel-measurable-lebesgue-integral
 borel-measurable-times iffD2[OF measurable-split-conv] merge-2) measurable
also have $\dots \leq (\int x. (\int y. f(merge I1 I2 (x,y)) \partial ?B) \partial ?A) * (\int x. (\int y. g(merge I1 I2 (x,y)) \partial ?B) \partial ?A)$
by (intro pa.neg-assoc-imp-mult-mono-bounded[OF pa-na, **where** $\eta = Fwd$ **and**
 $J = K \cap I1$]
 bounded-intros pb.borel-measurable-lebesgue-integral iffD2[OF measurable-split-conv])

```

merge-2 step2-1 step2-2 step2-3 step2-4) measurable
also have ... = (∫ ω. f(merge I1 I2 ω) ∂(?A ⊗M ?B)) * (∫ ω. g(merge I1 I2
ω) ∂(?A ⊗M ?B))
by (intro arg-cong2[where f=(*)] integral-fst' merge-2 bounded-intros) mea-
surable
also have ... = (∫ ω. f(merge I1 I2 ω) ∂?H) * (∫ ω. g(merge I1 I2 ω) ∂?H)
unfolding indep by simp
also have ... = (∫ ω. (f◦flip X') ω ∂M) * (∫ ω. (g◦flip X') ω ∂M)
unfolding split by (intro arg-cong2[where f=(*)] integral-distr merge-2)
measurable
finally show ?case by (simp add:comp-def)
qed
show ?thesis by (intro neg-assoc-cong[OF assms(1) X-measurable na-X']) (simp-all
add:X'-def)
qed

```

Property P7 [13]

lemma *neg-assoc-union*:

```

fixes I :: 'i set
fixes p :: 'j ⇒ 'i set
fixes X :: 'i ⇒ 'a ⇒ ('b::linorder-topology)
assumes finite I ∪ (p ' J) = I
assumes indep-vars (λj. PiM (p j) (λ-. borel)) (λj ω. λi ∈ p j. X i ω) J
assumes ∧j. j ∈ J ⇒ neg-assoc X (p j)
assumes disjoint-family-on p J
shows neg-assoc X I

```

proof –

```

let ?B = (λ-. borel)
define T where T = {j ∈ J. p j ≠ {}}

```

```

define g where g i = (THE j. j ∈ J ∧ i ∈ p j) for i
have g: g i = j if i ∈ p j j ∈ J for i j unfolding g-def
proof (rule the1-equality)

```

```

show ∃!j. j ∈ J ∧ i ∈ p j
using assms(5) that unfolding bex1-def disjoint-family-on-def by auto
show j ∈ J ∧ i ∈ p j using that by auto

```

qed

```

have ran-T: T ⊆ J unfolding T-def by simp
hence disjoint-family-on p T using assms(5) disjoint-family-on-mono by metis
moreover have finite (∪ (p ' T)) using ran-T assms(1,2)
by (meson Union-mono finite-subset image-mono)
moreover have ∧i. i ∈ T ⇒ p i ≠ {} unfolding T-def by auto
ultimately have fin-T: finite T using infinite-disjoint-family-imp-infinite-UNION
by auto

```

```

have neg-assoc X (∪ (p ' T))
using fin-T ran-T
proof (induction T rule:finite-induct)

```

case empty thus ?case using neg-assoc-empty by simp
next
case (insert x F)

note $r = \text{indep-var-compose}[OF \text{ indep-var-restrict}[OF \text{ assms}(3)], \text{ where } A=F$
and } B=\{x\}] -]

have $a: (\lambda\omega. \lambda i \in \bigcup(p'F). X i \omega) = (\lambda\omega. \lambda i \in \bigcup(p'F). \omega (g i) i) \circ (\lambda\omega. \lambda i \in F.$
 $\lambda i \in p i. X i \omega)$
using $\text{insert}(4)$ **g by** $(\text{intro restrict-ext ext}) \text{ auto}$
have $b: (\lambda\omega. \lambda i \in p x. X i \omega) = (\lambda\omega i. \omega x i) \circ (\lambda\omega. \lambda i \in \{x\}. \lambda i \in p i. X i \omega)$
by $(\text{simp add:comp-def restrict-def})$

have $c: (\lambda x. x (g i) i) \in \text{borel-measurable } (Pi_M F (\lambda j. Pi_M (p j) ?B))$ **if** $i \in$
 $(\bigcup(p'F))$ **for** i
proof –
have $h: i \in p (g i)$ **and** $q: g i \in F$ **using** g **that** $\text{insert}(4)$ **by** auto
thus $?thesis$
by $(\text{intro measurable-compose}[OF \text{ measurable-component-singleton}[OF q]])$
 measurable
qed

have $\text{finite } (\bigcup (p ' \text{insert } x F))$ **using** $\text{assms}(1,2)$ $\text{insert}(4)$
by $(\text{meson Sup-subset-mono image-mono infinite-super})$
moreover **have** $\bigcup (p ' F) \cup p x = \bigcup (p ' \text{insert } x F)$ **by** auto
moreover **have** $\bigcup (p ' F) \cap p x = \{\}$
using $\text{assms}(5)$ $\text{insert}(2,4)$ **unfolding** $\text{disjoint-family-on-def}$ **by** fast
moreover **have**
 $\text{indep-var } (Pi_M (\bigcup(p'F)) ?B) (\lambda\omega. \lambda i \in \bigcup(p'F). X i \omega) (Pi_M (p x) ?B) (\lambda\omega.$
 $\lambda i \in p x. X i \omega)$
unfolding $a b$ **using** $\text{insert}(1,2,4)$ **by** $(\text{intro } r \text{ measurable-restrict } c) \text{ simp-all}$
moreover **have** $\text{neg-assoc } X (\bigcup (p ' F))$ **using** $\text{insert}(4)$ **by** $(\text{intro } \text{insert}(3))$
 auto
moreover **have** $\text{neg-assoc } X (p x)$ **using** $\text{insert}(4)$ **by** $(\text{intro } \text{assms}(4)) \text{ auto}$
ultimately **show** $?case$ **by** $(\text{rule } \text{neg-assoc-combine})$
qed
moreover **have** $(\bigcup(p ' T)) = I$ **using** $\text{assms}(2)$ **unfolding** $T\text{-def}$ **by** auto
ultimately **show** $?thesis$ **by** auto
qed

Property P5 [13]

lemma $\text{indep-imp-neg-assoc}$:

assumes $\text{finite } I$

assumes $\text{indep-vars } (\lambda-. \text{borel}) X I$

shows $\text{neg-assoc } X I$

proof –

have $a: \text{neg-assoc } X \{i\}$ **if** $i \in I$ **for** i

using $\text{that } \text{assms}(2)$ **unfolding** indep-vars-def

by $(\text{intro } \text{neg-assoc-singleton}) \text{ auto}$

have $b: (\bigcup_{j \in I}. \{j\}) = I$ **by** *auto*
have $c: \text{indep-vars } (\lambda j. \text{Pi}_M \{j\} (\lambda-. \text{borel})) (\lambda j \omega. \lambda i \in \{j\}. X j \omega) I$
by (*intro indep-vars-compose2[OF assms(2)] measurable*)
have $d: \text{indep-vars } (\lambda j. \text{Pi}_M \{j\} (\lambda-. \text{borel})) (\lambda j \omega. \lambda i \in \{j\}. X i \omega) I$
by (*intro iffD2[OF indep-vars-cong c] restrict-ext ext*) *auto*
show *?thesis* **by** (*intro neg-assoc-union[OF assms(1) b d a]*) (*auto simp: disjoint-family-on-def*)
qed

end

lemma *neg-assoc-map-pmf*:

shows $\text{measure-pmf.neg-assoc } (\text{map-pmf } f \ p) \ X \ I = \text{measure-pmf.neg-assoc } p \ (\lambda i \omega. X i (f \ \omega)) \ I$
(is $?L \longleftrightarrow ?R$ **)**

proof –

let $?d1 = \text{distr } (\text{measure-pmf } (\text{map-pmf } f \ p)) \ (\text{Pi}_M \ I \ (\lambda-. \text{borel})) \ (\lambda \omega. \lambda i \in I. X i \omega)$
let $?d2 = \text{distr } (\text{measure-pmf } p) \ (\text{Pi}_M \ I \ (\lambda-. \text{borel})) \ (\lambda \omega. \lambda i \in I. X i (f \ \omega))$

have $\text{emeasure } ?d1 \ A = \text{emeasure } ?d2 \ A$ **if** $A \in \text{sets } (\text{Pi}_M \ I \ (\lambda-. \text{borel}))$ **for** A

proof –

have $\text{emeasure } ?d1 \ A = \text{emeasure } (\text{measure-pmf } p) \ \{x. (\lambda i \in I. X i (f \ x)) \in A\}$
using *that* **by** (*subst emeasure-distr*) (*simp-all add: vimage-def space-PiM*)
also have $\dots = \text{emeasure } ?d2 \ A$
using *that* **by** (*subst emeasure-distr*) (*simp-all add: space-PiM vimage-def*)
finally show *?thesis* **by** *simp*

qed

hence $a: ?d1 = ?d2$ **by** (*intro measure-eqI*) *auto*

have $?L \longleftrightarrow \text{prob-space.neg-assoc } ?d1 \ (\lambda x \ y. y \ x) \ I$
by (*subst measure-pmf.neg-assoc-iff-distr*) *auto*
also have $\dots \longleftrightarrow \text{prob-space.neg-assoc } ?d2 \ (\lambda x \ y. y \ x) \ I$
unfolding a **by** *simp*
also have $\dots \longleftrightarrow ?R$
by (*subst measure-pmf.neg-assoc-iff-distr*) *auto*
finally show *?thesis* **by** *simp*

qed

end

3 Chernoff-Hoeffding Bounds

This section shows that all the well-known Chernoff-Hoeffding bounds hold also for negatively associated random variables. The proofs follow the derivations by Hoeffding [11], as well as, Motwani and Raghavan [16, Ch. 4], with the modification that the crucial steps, where the classic proofs use independence, are replaced with the application of Property P2 for negatively

associated RV's.

theory *Negative-Association-Chernoff-Bounds*

imports

Negative-Association-Definition

Concentration-Inequalities.McDiarmid-Inequality

Weighted-Arithmetic-Geometric-Mean. Weighted-Arithmetic-Geometric-Mean

begin

context *prob-space*

begin

context

fixes $I :: 'i$ set

fixes $X :: 'i \Rightarrow 'a \Rightarrow \text{real}$

assumes $na-X$: *neg-assoc* $X I$

assumes $fin-I$: *finite* I

begin

private lemma *transfer-to-clamped-vars*:

assumes $(\forall i \in I. AE \ \omega \text{ in } M. X \ i \ \omega \in \{a \ i..b \ i\} \wedge a \ i \leq b \ i)$

assumes \mathcal{X} -def: $\mathcal{X} = (\lambda i. \text{clamp} \ (a \ i) \ (b \ i) \circ X \ i)$

shows *neg-assoc* $\mathcal{X} I$ (**is** ?A)

and $\bigwedge i. i \in I \implies \text{expectation} \ (\mathcal{X} \ i) = \text{expectation} \ (X \ i)$

and $\mathcal{P}(\omega \text{ in } M. (\sum i \in I. X \ i \ \omega) \leq_{\geq} \eta \ c) = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} \ i \ \omega) \leq_{\geq} \eta \ c)$

(is ?C)

and $\bigwedge i \ \omega. i \in I \implies \mathcal{X} \ i \ \omega \in \{a \ i..b \ i\}$

and $\bigwedge i \ S. i \in I \implies \text{bounded} \ (\mathcal{X} \ i \ 'S)$

and $\bigwedge i. i \in I \implies \text{expectation} \ (\mathcal{X} \ i) \in \{a \ i..b \ i\}$

proof –

note $[\text{measurable}] = \text{clamp-borel}$

note $rv-X = \text{neg-assoc-imp-measurable}[OF \ na-X]$

hence $rv-\mathcal{X}$: *random-variable borel* $(\mathcal{X} \ i)$ **if** $i \in I$ **for** i

unfolding \mathcal{X} -def **using** $rv-X[OF \ that]$ **by** *measurable*

have $a:AE \ x \text{ in } M. \mathcal{X} \ i \ x = X \ i \ x$ **if** $i \in I$ **for** i

unfolding \mathcal{X} -def **using** *clamp-eqI2* **by** (*intro* AE -mp[*OF* *bspec*[*OF* *assms*(1) *that*] AE -I2]) *auto*

hence $b:AE \ x \text{ in } M. (\forall i \in I. \mathcal{X} \ i \ x = X \ i \ x)$

by (*intro* AE -finite-all[*OF* $fin-I$]) *simp*

show ?A

using a **by** (*intro* *neg-assoc-cong*[*OF* $fin-I \ rv-\mathcal{X} \ na-X$]) *force+*

show $\text{expectation} \ (\mathcal{X} \ i) = \text{expectation} \ (X \ i)$ **if** $i \in I$ **for** i

by (*intro* *integral-cong-AE* $a \ rv-X \ rv-\mathcal{X}$ *that*)

have $\{\omega \in \text{space } M. (\sum i \in I. X \ i \ \omega) \leq_{\geq} \eta \ c\} \in \text{events}$ **using** $rv-X$ **by** (*cases* η)

simp-all

moreover have $\{\omega \in \text{space } M. (\sum_{i \in I} \mathcal{X} \ i \ \omega) \leq_{\geq \eta} c\} \in \text{events}$ **using** *rv- \mathcal{X}*
by (*cases η*) *simp-all*
ultimately show $?C$ **by** (*intro measure-eq-AE AE-mp[OF b AE-I2]*) *auto*

show $c: \mathcal{X} \ i \ \omega \in \{a \ i..b \ i\}$ **if** $i \in I$ **for** $\omega \ i$
unfolding *\mathcal{X} -def comp-def* **using** *assms(1) clamp-range* **that by** *simp*

show $d: \text{bounded} (\mathcal{X} \ i \ 'S)$ **if** $i \in I$ **for** $S \ i$
using *c[OF that] assms(2) bounded-clamp* **by** *blast*

show $\text{expectation} (\mathcal{X} \ i) \in \{a \ i..b \ i\}$ **if** $i \in I$ **for** i
unfolding *atLeastAtMost-iff* **using** *c[OF that] rv- \mathcal{X} [OF that]*
by (*intro conjI integral-ge-const integral-le-const AE-I2 integrable-bounded d[OF that]*) *auto*
qed

lemma *ln-one-plus-x-lower-bound:*

assumes $x \geq (0::\text{real})$
shows $2*x/(2+x) \leq \ln (1 + x)$

proof –

define v **where** $v \ x = \ln(1+x) - 2 * x / (2+x)$ **for** $x :: \text{real}$
define v' **where** $v' \ x = 1/(1+x) - 4/(2+x)^2$ **for** $x :: \text{real}$

have $v\text{-deriv: } (v \ \text{has-real-derivative} (v' \ x)) \ \text{at } x$ **if** $x \geq 0$ **for** x
using *that unfolding v-def v'-def power2-eq-square* **by** (*auto intro!: derivative-eq-intros*)
have $v\text{-deriv-nonneg: } v' \ x \geq 0$ **if** $x \geq 0$ **for** x
using *that unfolding v'-def*
by (*simp add: divide-simps power2-eq-square*) (*simp add: algebra-simps*)

have $v\text{-mono: } v \ x \leq v \ y$ **if** $x \leq y \ x \geq 0$ **for** $x \ y$
using *v-deriv v-deriv-nonneg that order-trans*
by (*intro DERIV-nonneg-imp-nondecreasing[OF that(1)]*) *blast*

have $0 = v \ 0$ **unfolding** *v-def* **by** *simp*
also have $\dots \leq v \ x$ **using** *v-mono assms* **by** *auto*
finally have $v \ x \geq 0$ **by** *simp*
thus $?thesis$ **unfolding** *v-def* **by** *simp*

qed

Based on Theorem 4.1 by Motwani and Raghavan [16].

theorem *multiplicative-chernoff-bound-upper:*

assumes $\delta > 0$
assumes $\bigwedge i. i \in I \implies \text{AE } \omega \ \text{in } M. X \ i \ \omega \in \{0..1\}$
defines $\mu \equiv (\sum_{i \in I} \text{expectation} (X \ i))$
shows $\mathcal{P}(\omega \ \text{in } M. (\sum_{i \in I} X \ i \ \omega) \geq (1+\delta) * \mu) \leq (\text{exp } \delta / ((1+\delta) \ \text{powr } (1+\delta)))$
 $\text{powr } \mu$ (**is** $?L \leq ?R$)
and $\mathcal{P}(\omega \ \text{in } M. (\sum_{i \in I} X \ i \ \omega) \geq (1+\delta) * \mu) \leq \text{exp } (-\delta^2) * \mu / (2+\delta)$
(**is** $\leq ?R1$)

proof –

define \mathcal{X} **where** $\mathcal{X} = (\lambda i. \text{clamp } 0 \ 1 \circ X \ i)$

have $\text{transfer-to-clamped-vars-assms}$: $(\forall i \in I. \text{AE } \omega \text{ in } M. X \ i \ \omega \in \{0 \ .. \ 1\} \wedge 0 \leq (1::\text{real}))$

using $\text{assms}(2)$ **by** auto

note $\text{ttcv} = \text{transfer-to-clamped-vars}[OF \ \text{transfer-to-clamped-vars-assms} \ \mathcal{X}\text{-def}]$

note $[\text{measurable}] = \text{neg-assoc-imp-measurable}[OF \ \text{ttcv}(1)]$

define t **where** $t = \ln \ (1+\delta)$

have $t\text{-gt-0}$: $t > 0$ **using** $\text{assms}(1)$ **unfolding** $t\text{-def}$ **by** simp

let $?h = (\lambda x. 1 + (\exp \ t - 1) * x)$

note $\text{bounded}' = \text{integrable-bounded} \ \text{bounded-prod} \ \text{bounded-vec-mult-comp} \ \text{bounded-intros} \ \text{ttcv}(5)$

have int : $\text{integrable } M \ (\mathcal{X} \ i)$ **if** $i \in I$ **for** i

using that **by** $(\text{intro} \ \text{bounded}') \ \text{simp-all}$

have $2*\delta \leq (2+\delta)* \ln \ (1 + \delta)$

using $\text{assms}(1)$ $\text{ln-one-plus-x-lower-bound}[OF \ \text{less-imp-le}[OF \ \text{assms}(1)]]$ **by** $(\text{simp} \ \text{add:field-simps})$

hence $(1+\delta)*(2*\delta) \leq (1 + \delta) *(2+\delta)* \ln \ (1 + \delta)$ **using** $\text{assms}(1)$ **by** simp

hence $a:(\delta - (1 + \delta) * \ln \ (1 + \delta)) \leq -(\delta^2)/(2+\delta)$

using $\text{assms}(1)$ **by** $(\text{simp} \ \text{add:field-simps} \ \text{power2-eq-square})$

have $\mu\text{-ge-0}$: $\mu \geq 0$ **unfolding** $\mu\text{-def}$ **using** $\text{ttcv}(2,6)$ **by** $(\text{intro} \ \text{sum-nonneg}) \ \text{auto}$

note $\mathcal{X}\text{-prod-mono} = \text{has-int-thatD}(2)[OF \ \text{neg-assoc-imp-prod-mono}[OF \ \text{fin-I} \ \text{ttcv}(1), \ \text{where} \ \eta = \text{Fwd}]]$

have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} \ i \ \omega) \geq (1+\delta) * \mu)$ **using** $\text{ttcv}(3)$ **[where** $\eta = \text{Rev}]$ **by** simp

also have $\dots = \mathcal{P}(\omega \text{ in } M. (\prod i \in I. \exp \ (t * \mathcal{X} \ i \ \omega)) \geq \exp \ (t * (1+\delta) * \mu))$

using $t\text{-gt-0}$ **by** $(\text{simp} \ \text{add:sum-distrib-left}[\text{symmetric}] \ \text{exp-sum}[OF \ \text{fin-I}, \ \text{symmetric}])$

also have $\dots \leq \text{expectation} \ (\lambda \omega. (\prod i \in I. \exp \ (t * \mathcal{X} \ i \ \omega))) / \exp \ (t*(1+\delta)*\mu)$

by $(\text{intro} \ \text{integral-Markov-inequality-measure}[\text{where} \ A = \{\}]) \ \text{bounded}' \ \text{AE-I2} \ \text{prod-nonneg} \ \text{fin-I})$

simp-all

also have $\dots \leq (\prod i \in I. \text{expectation} \ (\lambda \omega. \exp \ (t*\mathcal{X} \ i \ \omega))) / \exp \ (t*(1+\delta)*\mu)$

using $t\text{-gt-0}$ **by** $(\text{intro} \ \text{divide-right-mono} \ \mathcal{X}\text{-prod-mono} \ \text{bounded}' \ \text{image-subsetI} \ \text{monotoneI}) \ \text{simp-all}$

also have $\dots = (\prod i \in I. \text{expectation} \ (\lambda \omega. \exp \ ((1-\mathcal{X} \ i \ \omega) *_R \ 0 + \mathcal{X} \ i \ \omega *_R \ t))) / \exp \ (t*(1+\delta)*\mu)$

by $(\text{simp} \ \text{add:ac-simps})$

also have $\dots \leq (\prod i \in I. \text{expectation} \ (\lambda \omega. (1-\mathcal{X} \ i \ \omega) * \exp \ 0 + \mathcal{X} \ i \ \omega * \exp \ t)) / \exp \ (t*(1+\delta)*\mu)$

using $\text{ttcv}(4)$

by (*intro divide-right-mono prod-mono integral-mono conjI bounded' convex-onD[OF exp-convex]*)
simp-all
also have $\dots = (\prod i \in I. ?h (\text{expectation } (\mathcal{X} i))) / \exp (t*(1+\delta)*\mu)$
using *int by (simp add:algebra-simps prob-space cong:prod.cong)*
also have $\dots \leq (\prod i \in I. \exp((\exp t-1)* \text{expectation } (\mathcal{X} i))) / \exp (t*(1+\delta)*\mu)$
using *t-gt-0 ttcv(4)*
by (*intro divide-right-mono prod-mono exp-ge-add-one-self conjI add-nonneg-nonneg mult-nonneg-nonneg simp-all*)
also have $\dots = \exp ((\exp t-1)* \mu) / \exp (t*(1+\delta)*\mu)$
unfolding *exp-sum[OF fin-I, symmetric] μ -def* **by** (*simp add:ttcv(2) sum-distrib-left*)
also have $\dots = \exp (\delta * \mu) / \exp (\ln (1+\delta)*(1+\delta) * \mu)$
using *assms(1) unfolding μ -def t-def* **by** (*simp add:sum-distrib-left*)
also have $\dots = \exp \delta \text{ powr } \mu / \exp (\ln(1+\delta)*(1+\delta)) \text{ powr } \mu$
unfolding *powr-def* **by** (*simp add:ac-simps*)
also have $\dots = ?R$ **using** *assms(1) by (subst powr-divide) (simp-all add:powr-def)*
finally show $?L \leq ?R$ **by** *simp*
also have $\dots = \exp (\mu * \ln (\exp \delta / \exp ((1 + \delta) * \ln (1 + \delta))))$
using *assms unfolding powr-def* **by** *simp*
also have $\dots = \exp (\mu * (\delta - (1 + \delta) * \ln (1 + \delta)))$ **by** (*subst ln-div*) *simp-all*
also have $\dots \leq \exp (\mu * (-\delta^2)/(2+\delta))$
by (*intro iffD2[OF exp-le-cancel-iff] mult-left-mono a μ -ge-0*)
also have $\dots = ?R1$ **by** *simp*
finally show $?L \leq ?R1$ **by** *simp*
qed

lemma *ln-one-minus-x-lower-bound:*

assumes $x \in \{(0::\text{real})..<1\}$
shows $(x^2/2-x)/(1-x) \leq \ln (1 - x)$
proof –
define v **where** $v x = \ln(1-x) - (x^2/2-x) / (1-x)$ **for** $x :: \text{real}$
define v' **where** $v' x = -1/(1-x) - (-(x^2)/2+x-1)/((1-x)^2)$ **for** $x :: \text{real}$

have v -*deriv*: (v has-real-derivative ($v' x$)) (at x) **if** $x \in \{0..<1\}$ **for** x
using *that unfolding v-def v'-def power2-eq-square*
by (*auto intro!:derivative-eq-intros simp:algebra-simps*)
have v -*deriv-nonneg*: $v' x \geq 0$ **if** $x \geq 0$ **for** x
using *that unfolding v'-def by (simp add:divide-simps power2-eq-square)*

have v -*mono*: $v x \leq v y$ **if** $x \leq y$ $x \geq 0$ $y < 1$ **for** $x y$
using v -*deriv* v -*deriv-nonneg* *that unfolding atLeastLessThan-iff*
by (*intro DERIV-nonneg-imp-nondecreasing[OF that(1)]*)
(*metis (mono-tags, opaque-lifting) Ico-eq-Ico ivl-subset linorder-not-le order-less-irrefl*)

have $0 = v 0$ **unfolding** v -*def* **by** *simp*
also have $\dots \leq v x$ **using** v -*mono* *assms* **by** *auto*
finally have $v x \geq 0$ **by** *simp*
thus *thesis* **unfolding** v -*def* **by** *simp*
qed

Based on Theorem 4.2 by Motwani and Raghavan [16].

theorem *multiplicative-chernoff-bound-lower*:

assumes $\delta \in \{0 < \cdot < 1\}$

assumes $\bigwedge i. i \in I \implies AE \ \omega \text{ in } M. X \ i \ \omega \in \{0..1\}$

defines $\mu \equiv (\sum i \in I. expectation \ (X \ i))$

shows $\mathcal{P}(\omega \text{ in } M. (\sum i \in I. X \ i \ \omega) \leq (1-\delta)*\mu) \leq (exp \ (-\delta)/(1-\delta) \text{ powr } (1-\delta))$
 $\text{powr } \mu$ (**is** $?L \leq ?R$)

and $\mathcal{P}(\omega \text{ in } M. (\sum i \in I. X \ i \ \omega) \leq (1-\delta)*\mu) \leq (exp \ (-\delta^2)*\mu/2)$ (**is** $\leq ?R1$)

proof –

define \mathcal{X} **where** $\mathcal{X} = (\lambda i. clamp \ 0 \ 1 \ \circ \ X \ i)$

have *transfer-to-clamped-vars-assms*: $(\forall i \in I. AE \ \omega \text{ in } M. X \ i \ \omega \in \{0 .. 1\} \wedge 0 \leq (1::real))$

using *assms(2)* **by** *auto*

note *tccv* = *transfer-to-clamped-vars*[*OF transfer-to-clamped-vars-assms* \mathcal{X} -*def*]

note [*measurable*] = *neg-assoc-imp-measurable*[*OF tccv(1)*]

define t **where** $t = \ln \ (1-\delta)$

have *t-lt-0*: $t < 0$ **using** *assms(1)* **unfolding** *t-def* **by** *simp*

let $?h = (\lambda x. 1 + (exp \ t - 1) * x)$

note *bounded'* = *integrable-bounded bounded-prod bounded-vec-mult-comp bounded-intros tccv(5)*

have μ -*ge-0*: $\mu \geq 0$ **unfolding** μ -*def* **using** *tccv(2,6)* **by** (*intro sum-nonneg*) *auto*

have *int*: *integrable* $M \ (\mathcal{X} \ i)$ **if** $i \in I$ **for** i

using *that* **by** (*intro bounded'*) *simp-all*

note \mathcal{X} -*prod-mono* = *has-int-thatD(2)*[*OF neg-assoc-imp-prod-mono*[*OF fin-I tccv(1)*, **where** $\eta = Rev$]]

have 0 : $0 \leq 1 + (exp \ t - 1) * expectation \ (\mathcal{X} \ i)$ **if** $i \in I$ **for** i

proof –

have $0 \leq 1 + (exp \ t - 1) * 1$ **by** *simp*

also have $\dots \leq 1 + (exp \ t - 1) * expectation \ (\mathcal{X} \ i)$

using *t-lt-0 tccv(6)*[*OF that*] **by** (*intro add-mono mult-left-mono-neg*) *auto*

finally show *?thesis* **by** *simp*

qed

have $\delta \in \{0..<1\}$ **using** *assms(1)* **by** *simp*

from *ln-one-minus-x-lower-bound*[*OF this*]

have $\delta^2 / 2 - \delta \leq (1 - \delta) * \ln \ (1 - \delta)$ **using** *assms(1)* **by** (*simp add:field-simps*)

hence $1: -\delta - (1 - \delta) * \ln \ (1 - \delta) \leq -\delta^2 / 2$ **by** (*simp add:algebra-simps*)

have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} \ i \ \omega) \leq (1-\delta) * \mu)$ **using** *tccv(3)*[**where** $\eta = Fwd$] **by** *simp*

also have $\dots = \mathcal{P}(\omega \text{ in } M. (\prod i \in I. \exp (t * \mathcal{X} i \omega)) \geq \exp (t * (1-\delta) * \mu))$
using *t-lt-0* **by** (*simp add: sum-distrib-left[symmetric] exp-sum[OF fin-I, symmetric]*)
also have $\dots \leq \text{expectation} (\lambda \omega. (\prod i \in I. \exp (t * \mathcal{X} i \omega))) / \exp (t * (1-\delta) * \mu)$
by (*intro integral-Markov-inequality-measure[where A={}] bounded' AE-I2 prod-nonneg fin-I*)
simp-all
also have $\dots \leq (\prod i \in I. \text{expectation} (\lambda \omega. \exp (t * \mathcal{X} i \omega))) / \exp (t * (1-\delta) * \mu)$
using *t-lt-0* **by** (*intro divide-right-mono X-prod-mono bounded' image-subsetI monotoneI*) *simp-all*
also have $\dots = (\prod i \in I. \text{expectation} (\lambda \omega. \exp ((1-\mathcal{X} i \omega) *_{\mathbb{R}} 0 + \mathcal{X} i \omega *_{\mathbb{R}} t))) / \exp (t * (1-\delta) * \mu)$
by (*simp add: ac-simps*)
also have $\dots \leq (\prod i \in I. \text{expectation} (\lambda \omega. (1-\mathcal{X} i \omega) * \exp 0 + \mathcal{X} i \omega * \exp t)) / \exp (t * (1-\delta) * \mu)$
using *ttcv(4)*
by (*intro divide-right-mono prod-mono integral-mono conjI bounded' convex-onD[OF exp-convex]*)
simp-all
also have $\dots = (\prod i \in I. ?h (\text{expectation} (\mathcal{X} i))) / \exp (t * (1-\delta) * \mu)$
using *int* **by** (*simp add: algebra-simps prob-space cong: prod.cong*)
also have $\dots \leq (\prod i \in I. \exp ((\exp t - 1) * \text{expectation} (\mathcal{X} i))) / \exp (t * (1-\delta) * \mu)$
using *0* **by** (*intro divide-right-mono prod-mono exp-ge-add-one-self conjI*)
simp-all
also have $\dots = \exp ((\exp t - 1) * \mu) / \exp (t * (1-\delta) * \mu)$
unfolding *exp-sum[OF fin-I, symmetric] mu-def* **by** (*simp add: ttcv(2) sum-distrib-left*)
also have $\dots = \exp ((-\delta) * \mu) / \exp (\ln (1-\delta) * (1-\delta) * \mu)$
using *assms(1) unfolding mu-def t-def* **by** (*simp add: sum-distrib-left*)
also have $\dots = \exp (-\delta) \text{powr } \mu / \exp (\ln (1-\delta) * (1-\delta)) \text{powr } \mu$
unfolding *powr-def* **by** (*simp add: ac-simps*)
also have $\dots = ?R$ **using** *assms(1)* **by** (*subst powr-divide*) (*simp-all add: powr-def*)
finally show $?L \leq ?R$ **by** *simp*
also have $\dots = \exp (\mu * (-\delta - (1-\delta) * \ln (1-\delta)))$
using *assms(1) unfolding powr-def* **by** (*simp add: ln-div*)
also have $\dots \leq \exp (\mu * (-\delta^2) / 2)$
by (*intro iffD2[OF exp-le-cancel-iff] mult-left-mono mu-ge-0 1*)
finally show $?L \leq ?R1$ **by** (*simp add: ac-simps*)

qed

theorem *multiplicative-bernoulli-bound-two-sided:*

assumes $\delta \in \{0 < .. < 1\}$
assumes $\bigwedge i. i \in I \implies AE \omega \text{ in } M. X i \omega \in \{0..1\}$
defines $\mu \equiv (\sum i \in I. \text{expectation} (X i))$
shows $\mathcal{P}(\omega \text{ in } M. |(\sum i \in I. X i \omega) - \mu| \geq \delta * \mu) \leq 2 * (\exp (-\delta^2 * \mu / 3))$ (**is** $?L \leq ?R$)
proof –
define \mathcal{X} **where** $\mathcal{X} = (\lambda i. \text{clamp } 0 \ 1 \circ X i)$
have *transfer-to-clamped-vars-assms*: $(\forall i \in I. AE \omega \text{ in } M. X i \omega \in \{0..1\}) \wedge 0 \leq (1::\text{real})$
using *assms(2)* **by** *auto*

note $ttcv = transfer\text{-}to\text{-}clamped\text{-}vars[OF\ transfer\text{-}to\text{-}clamped\text{-}vars\text{-}assms\ \mathcal{X}\text{-}def]$
have $\mu\text{-}ge\text{-}0: \mu \geq 0$ **unfolding** $\mu\text{-}def$ **using** $ttcv(2,6)$ **by** $(intro\ sum\text{-}nonneg)$
auto
note $[measurable] = neg\text{-}assoc\text{-}imp\text{-}measurable[OF\ na\text{-}X]$
have $?L = \mathcal{P}(\omega\ in\ M. (\sum\ i \in I. X\ i\ \omega) \geq (1+\delta)*\mu \vee (\sum\ i \in I. X\ i\ \omega) \leq (1-\delta)*\mu)$
unfolding $abs\text{-}real\text{-}def$
by $(intro\ arg\text{-}cong[where\ f=prob]\ Collect\text{-}cong)$ $(auto\ simp:algebra\text{-}simps)$
also have $\dots = measure\ M(\{\omega \in space\ M. (\sum\ i \in I. X\ i\ \omega) \geq (1+\delta)*\mu\} \cup \{\omega \in space\ M. (\sum\ i \in I. X\ i\ \omega) \leq (1-\delta)*\mu\})$
by $(intro\ arg\text{-}cong[where\ f=prob])\ auto$
also have $\dots \leq \mathcal{P}(\omega\ in\ M. (\sum\ i \in I. X\ i\ \omega) \geq (1+\delta)*\mu) + \mathcal{P}(\omega\ in\ M. (\sum\ i \in I. X\ i\ \omega) \leq (1-\delta)*\mu)$
by $(intro\ measure\text{-}Un\text{-}le)\ measurable$
also have $\dots \leq exp\ (-(\delta^2)*\mu/(2+\delta)) + exp\ (-(\delta^2)*\mu/2)$
unfolding $\mu\text{-}def$ **using** $assms(1,2)$
by $(intro\ multiplicative\text{-}chernoff\text{-}bound\text{-}lower\ multiplicative\text{-}chernoff\text{-}bound\text{-}upper\ add\text{-}mono)\ auto$
also have $\dots \leq exp\ (-(\delta^2)*\mu/3) + exp\ (-(\delta^2)*\mu/3)$
using $assms(1)\ \mu\text{-}ge\text{-}0$ **by** $(intro\ iffD2[OF\ exp\text{-}le\text{-}cancel\text{-}iff])\ add\text{-}mono\ divide\text{-}left\text{-}mono\text{-}neg)\ auto$
also have $\dots = ?R$ **by** $simp$
finally show $?thesis$ **by** $simp$
qed
lemma $additive\text{-}chernoff\text{-}bound\text{-}upper\text{-}aux:$
assumes $\bigwedge i. i \in I \implies AE\ \omega\ in\ M. X\ i\ \omega \in \{0..1\}\ I \neq \{\}$
defines $\mu \equiv (\sum\ i \in I. expectation\ (X\ i)) / real\ (card\ I)$
assumes $\delta \in \{0 < .. < 1 - \mu\}\ \mu \in \{0 < .. < 1\}$
shows $\mathcal{P}(\omega\ in\ M. (\sum\ i \in I. X\ i\ \omega) \geq (\mu + \delta)*real\ (card\ I)) \leq exp\ (-real\ (card\ I) * KL\text{-}div\ (\mu + \delta)\ \mu)$
(is $?L \leq ?R)$
proof –
define \mathcal{X} **where** $\mathcal{X} = (\lambda i. clamp\ 0\ 1 \circ X\ i)$
have $transfer\text{-}to\text{-}clamped\text{-}vars\text{-}assms: (\forall\ i \in I. AE\ \omega\ in\ M. X\ i\ \omega \in \{0..1\}) \wedge 0 \leq (1::real)$
using $assms(1)$ **by** $auto$
note $ttcv = transfer\text{-}to\text{-}clamped\text{-}vars[OF\ transfer\text{-}to\text{-}clamped\text{-}vars\text{-}assms\ \mathcal{X}\text{-}def]$
note $[measurable] = neg\text{-}assoc\text{-}imp\text{-}measurable[OF\ ttcv(1)]$
define $t :: real$ **where** $t = ln\ ((\mu + \delta)/\mu) - ln\ ((1 - \mu - \delta)/(1 - \mu))$
let $?h = \lambda x. 1 + (exp\ t - 1) * x$
let $?n = real\ (card\ I)$
have $n\text{-}gt\text{-}0: ?n > 0$ **using** $assms(2)\ fin\text{-}I$ **by** $auto$
have $a: (1 - \mu - \delta) > 0\ \mu > 0\ 1 - \mu > 0\ \mu + \delta > 0$

using *assms(4,5)* **by** *auto*

have $\ln((1 - \mu - \delta) / (1 - \mu)) < 0$ **using** *a assms(4)* **by** (*intro ln-less-zero*)
auto

moreover have $\ln((\mu + \delta) / \mu) > 0$ **using** *a assms(4)* **by** (*intro ln-gt-zero*)
auto

ultimately have *t-gt-0: t > 0* **unfolding** *t-def* **by** *simp*

note *bounded' = integrable-bounded bounded-prod bounded-vec-mult-comp bounded-intros*
ttcv(5)

note \mathcal{X} -*prod-mono = has-int-thatD(2)[OF neg-assoc-imp-prod-mono[OF fin-I*
ttcv(1), where $\eta = Fwd$]

have *int: integrable M (X i) if i ∈ I for i*
using *that* **by** (*intro bounded'*) *simp-all*

have $0: 0 \leq 1 + (\exp t - 1) * \text{expectation } (\mathcal{X} i)$ **if** $i \in I$ **for** i
using *t-gt-0 ttcv(6)[OF that]* **by** (*intro add-nonneg-nonneg mult-nonneg-nonneg*)
auto

have $1 + (\exp t - 1) * \mu = 1 + ((\mu + \delta) * (1 - \mu) / (\mu * (1 - \mu - \delta))) - 1$
 $* \mu$
using *a* **unfolding** *t-def exp-diff* **by** *simp*
also have $\dots = 1 + (\delta / (\mu * (1 - \mu - \delta))) * \mu$
using *a* **by** (*subst divide-diff-eq-iff*) (*simp, simp add:algebra-simps*)
also have $\dots = (1 - \mu - \delta) / (1 - \mu - \delta) + (\delta / (1 - \mu - \delta))$ **using** *a* **by** *simp*
also have $\dots = (1 - \mu) / (1 - \mu - \delta)$
unfolding *add-divide-distrib[symmetric]* **by** (*simp add:algebra-simps*)
also have $\dots = \text{inverse } ((1 - \mu - \delta) / (1 - \mu))$ **using** *a* **by** *simp*
also have $\dots = \exp(\ln(\text{inverse } ((1 - \mu - \delta) / (1 - \mu))))$ **using** *a* **by** *simp*
also have $\dots = \exp(-\ln((1 - \mu - \delta) / (1 - \mu)))$ **using** *a* **by** (*subst ln-inverse*)
(simp-all)
finally have $1: 1 + (\exp t - 1) * \mu = \exp(-\ln((1 - \mu - \delta) / (1 - \mu)))$ **by** *simp*

have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} i \omega) \geq (\mu + \delta) * ?n)$ **using** *ttcv(3)[where*
 $\eta = Rev]$ **by** *simp*
also have $\dots = \mathcal{P}(\omega \text{ in } M. (\prod i \in I. \exp(t * \mathcal{X} i \omega)) \geq \exp(t * (\mu + \delta) * ?n))$
using *t-gt-0* **by** (*simp add: sum-distrib-left[symmetric] exp-sum[OF fin-I, symmetric]*)
also have $\dots \leq \text{expectation } (\lambda \omega. (\prod i \in I. \exp(t * \mathcal{X} i \omega))) / \exp(t * (\mu + \delta) * ?n)$
by (*intro integral-Markov-inequality-measure[where A={}] bounded' AE-I2*
prod-nonneg fin-I)
simp-all
also have $\dots \leq (\prod i \in I. \text{expectation } (\lambda \omega. \exp(t * \mathcal{X} i \omega))) / \exp(t * (\mu + \delta) * ?n)$
using *t-gt-0* **by** (*intro divide-right-mono X-prod-mono bounded' image-subsetI*
monotoneI) *simp-all*
also have $\dots = (\prod i \in I. \text{expectation } (\lambda \omega. \exp((1 - \mathcal{X} i \omega) *_{\mathbb{R}} 0 + \mathcal{X} i \omega *_{\mathbb{R}}$

$t)) / \exp(t * (\mu + \delta) * ?n)$
by (*simp add:ac-simps*)
also have $\dots \leq (\prod_{i \in I}. \text{expectation } (\lambda \omega. (1 - \mathcal{X} i \omega) * \exp 0 + \mathcal{X} i \omega * \exp t))$
 $/ \exp(t * (\mu + \delta) * ?n)$
using *ttcv(4)*
by (*intro divide-right-mono prod-mono integral-mono conjI bounded' convex-onD[OF exp-convex]*)
simp-all
also have $\dots = (\prod_{i \in I}. ?h(\text{expectation } (\mathcal{X} i))) / \exp(t * (\mu + \delta) * ?n)$
using *int by (simp add:algebra-simps prob-space cong:prod.cong)*
also have $\dots = (\text{root } (\text{card } I) (\prod_{i \in I}. 1 + (\exp t - 1) * \text{expectation } (\mathcal{X} i))) \wedge (\text{card } I) / \exp(t * (\mu + \delta) * ?n)$
using *n-gt-0*
by (*intro arg-cong2[where f=(/)] real-root-pow-pos2[symmetric] prod-nonneg refl 0) auto*
also have $\dots \leq ((\sum_{i \in I}. 1 + (\exp t - 1) * \text{expectation } (\mathcal{X} i)) / ?n)^{\wedge} (\text{card } I) / \exp(t * (\mu + \delta) * ?n)$
by (*intro divide-right-mono power-mono arithmetic-geometric-mean[OF fn-I] real-root-ge-zero*)
prod-nonneg 0) simp-all
also have $\dots \leq ((\sum_{i \in I}. 1 + (\exp t - 1) * \text{expectation } (\mathcal{X} i)) / ?n)^{\text{powr } ?n} / \exp(t * (\mu + \delta) * ?n)$
using *n-gt-0 0 by (subst powr-realpow') (auto intro!:sum-nonneg divide-nonneg-pos 0)*
also have $\dots \leq ((\sum_{i \in I}. 1 + (\exp t - 1) * \text{expectation } (\mathcal{X} i)) / ?n)^{\text{powr } ?n} / \exp(t * (\mu + \delta) * ?n)$
using *ttcv(2) by (simp cong:sum.cong)*
also have $\dots = (1 + (\exp t - 1) * \mu)^{\text{powr } ?n} / \exp(t * (\mu + \delta) * ?n)$
using *n-gt-0 unfolding μ -def sum.distrib sum-distrib-left[symmetric] by (simp add:divide-simps)*
also have $\dots = (1 + (\exp t - 1) * \mu)^{\text{powr } ?n} / \exp(t * (\mu + \delta)) \text{ powr } ?n$
unfolding *powr-def by simp*
also have $\dots = ((1 + (\exp t - 1) * \mu) / \exp(t * (\mu + \delta)))^{\text{powr } ?n}$
using *a t-gt-0 by (auto intro: powr-divide[symmetric] add-nonneg-nonneg mult-nonneg-nonneg)*
also have $\dots = (\exp(-\ln((1 - \mu - \delta) / (1 - \mu))) * \exp(-(t * (\mu + \delta))))^{\text{powr } ?n}$
unfolding *1 exp-minus inverse-eq-divide by simp*
also have $\dots = \exp(-\ln((1 - \mu - \delta) / (1 - \mu)) - t * (\mu + \delta))^{\text{powr } ?n}$
unfolding *exp-add[symmetric] by simp*
also have $\dots = \exp(-\ln((1 - \mu - \delta) / (1 - \mu)) - (\ln((\mu + \delta) / \mu) - \ln((1 - \mu - \delta) / (1 - \mu))) * (\mu + \delta))^{\text{powr } ?n}$
using *a unfolding t-def by (simp add:divide-simps)*
also have $\dots = \exp(-\text{KL-div } (\mu + \delta) \mu)^{\text{powr } ?n}$
using *a by (subst KL-div-eq) (simp-all add:field-simps)*
also have $\dots = ?R$ **unfolding** *powr-def by simp*
finally show *?thesis by simp*
qed

lemma *additive-chernoff-bound-upper-aux-2:*

assumes $\bigwedge i. i \in I \implies AE \ \omega \text{ in } M. X \ i \ \omega \in \{0..1\} \ I \neq \{\}$
defines $\mu \equiv (\sum i \in I. \text{expectation } (X \ i)) / \text{real } (\text{card } I)$
assumes $\mu \in \{0 <..< 1\}$
shows $\mathcal{P}(\omega \text{ in } M. (\sum i \in I. X \ i \ \omega) \geq \text{real } (\text{card } I)) \leq \text{exp } (-\text{real } (\text{card } I) * KL\text{-div } 1 \ \mu)$
(is ?L ≤ ?R)
proof –
define \mathcal{X} **where** $\mathcal{X} = (\lambda i. \text{clamp } 0 \ 1 \circ X \ i)$
have *transfer-to-clamped-vars-assms*: $(\forall i \in I. AE \ \omega \text{ in } M. X \ i \ \omega \in \{0..1\} \wedge 0 \leq (1 :: \text{real}))$
using *assms(1)* **by** *auto*
note *ttcv* = *transfer-to-clamped-vars*[*OF transfer-to-clamped-vars-assms* \mathcal{X} -*def*]
note [*measurable*] = *neg-assoc-imp-measurable*[*OF ttcv(1)*]

let $?n = \text{real } (\text{card } I)$

have *n-gt-0*: $?n > 0$ **using** *assms(2)* *fin-I* **by** *auto*

note *bounded'* = *integrable-bounded bounded-prod bounded-vec-mult-comp bounded-intros ttcv(5) bounded-max*

note \mathcal{X} -*prod-mono* = *has-int-thatD(2)*[*OF neg-assoc-imp-prod-mono*[*OF fin-I ttcv(1)*], **where** $\eta = \text{Fwd}$]

have *a2*: $(\prod i \in I. \text{max } 0 \ (X \ i \ \omega)) \geq 1$ **if** $(\sum i \in I. X \ i \ \omega) \geq ?n$ **for** ω
proof –
have $(\sum i \in I. 1 - X \ i \ \omega) \leq 0$ **using** *that* **by** (*simp add:sum-subtractf*)
moreover **have** $(\sum i \in I. 1 - X \ i \ \omega) \geq 0$ **using** *ttcv(4)* **by** (*intro sum-nonneg*)
simp
ultimately **have** $(\sum i \in I. 1 - X \ i \ \omega) = 0$ **by** *simp*
with *iffD1*[*OF sum-nonneg-eq-0-iff*[*OF fin-I*] *this*]
have $\forall i \in I. 1 - X \ i \ \omega = 0$ **using** *ttcv(4)* **by** *simp*
hence $X \ i \ \omega = 1$ **if** $i \in I$ **for** i **using** *that* **by** *auto*
thus *?thesis* **by** (*intro prod-ge-1*) *fastforce*
qed

have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. X \ i \ \omega) \geq ?n)$ **using** *ttcv(3)*[**where** $\eta = \text{Rev}$] **by** *simp*
also **have** $\dots \leq \mathcal{P}(\omega \text{ in } M. (\prod i \in I. \text{max } 0 \ (X \ i \ \omega)) \geq 1)$
using *a2* **by** (*intro finite-measure-mono*) *auto*
also **have** $\dots \leq \text{expectation } (\lambda \omega. (\prod i \in I. \text{max } 0 \ (X \ i \ \omega))) / 1$
by (*intro integral-Markov-inequality-measure*[**where** $A = \{\}$] *bounded' AE-I2 prod-nonneg fin-I*)
auto
also **have** $\dots \leq (\prod i \in I. \text{expectation } (\lambda \omega. \text{max } 0 \ (X \ i \ \omega))) / 1$
by (*intro divide-right-mono* \mathcal{X} -*prod-mono* *bounded' image-subsetI monotoneI*)
simp-all
also **have** $\dots \leq (\prod i \in I. \text{expectation } (X \ i))$ **using** *ttcv(4)* **by** *simp*

also have $\dots = (\text{root } (\text{card } I) (\prod_{i \in I} \text{expectation } (\mathcal{X} \ i))) \wedge (\text{card } I)$
using $n\text{-gt-0}$ $\text{ttcv}(6)$ **by** $(\text{intro } \text{real-root-pow-pos2}[\text{symmetric}] \ \text{prod-nonneg refl})$
auto
also have $\dots \leq ((\sum_{i \in I} \text{expectation } (\mathcal{X} \ i)) / ?n) \wedge (\text{card } I)$
using $\text{ttcv}(6)$ **by** $(\text{intro } \text{power-mono arithmetic-geometric-mean}[\text{OF } \text{fin-I}] \ \text{real-root-ge-zero}$
 $\text{prod-nonneg})$ *auto*
also have $\dots \leq ((\sum_{i \in I} \text{expectation } (\mathcal{X} \ i)) / ?n) \ \text{powr } ?n$
using $n\text{-gt-0}$ $\text{ttcv}(6)$ **by** $(\text{subst } \text{powr-realpow}^\wedge) \ (\text{auto } \text{intro}!: \text{sum-nonneg di-}$
 $\text{vide-nonneg-pos})$
also have $\dots \leq \mu \ \text{powr } ?n$ **using** $\text{ttcv}(2)$ **unfolding** $\mu\text{-def}$ **by** simp
also have $\dots = ?R$ **using** $\text{assms}(4)$ **unfolding** powr-def **by** $(\text{subst } \text{KL-div-eq})$
 $(\text{auto } \text{simp}:\text{ln-div})$
finally show $?thesis$ **by** simp
qed

Based on Theorem 1 by Hoeffding [11].

lemma *additive-chernoff-bound-upper:*

assumes $\bigwedge i. i \in I \implies AE \ \omega \ \text{in } M. \ X \ i \ \omega \in \{0..1\} \ I \neq \{\}$
defines $\mu \equiv (\sum_{i \in I} \text{expectation } (X \ i)) / \text{real } (\text{card } I)$
assumes $\delta \in \{0..1-\mu\} \ \mu \in \{0 < .. < 1\}$
shows $\mathcal{P}(\omega \ \text{in } M. (\sum_{i \in I} X \ i \ \omega) \geq (\mu + \delta) * \text{real } (\text{card } I)) \leq \exp (-\text{real } (\text{card } I)$
 $* \text{KL-div } (\mu + \delta) \ \mu)$
 $(\text{is } ?L \leq ?R)$
proof –
note $[\text{measurable}] = \text{neg-assoc-imp-measurable}[\text{OF } \text{na-X}]$

let $?n = \text{real } (\text{card } I)$
have $n\text{-gt-0}: ?n > 0$ **using** $\text{assms } \text{fin-I}$ **by** *auto*

note $X\text{-prod-mono} = \text{has-int-thatD}(2)[\text{OF } \text{neg-assoc-imp-prod-mono}[\text{OF } \text{fin-I}$
 $\text{na-X, where } \eta = \text{Fwd}]]$

have $b: AE \ x \ \text{in } M. (\forall i \in I. X \ i \ x \in \{0..1\})$
using $\text{assms}(1)$ **by** $(\text{intro } \text{AE-finite-allI}[\text{OF } \text{fin-I}]) \ \text{simp}$
hence $c: AE \ x \ \text{in } M. (\sum_{i \in I} 1 - X \ i \ x) \geq 0$
by $(\text{intro } \text{AE-mp}[\text{OF } b \ \text{AE-I2}] \ \text{impI } \text{sum-nonneg})$ *auto*

consider $(i) \ \delta = 0 \mid (ii) \ \delta \in \{0 < .. < 1 - \mu\} \mid (iii) \ 1 - \mu = \delta$ **using** $\text{assms}(4)$ **by**
fastforce

thus $?thesis$

proof (cases)

case i

hence $\text{KL-div } (\mu + \delta) \ \mu = 0$ **using** $\text{assms}(4,5)$ **by** $(\text{subst } \text{KL-div-eq})$ *auto*

thus $?thesis$ **by** simp

next

case ii

thus $?thesis$ **unfolding** $\mu\text{-def}$ **using** assms **by** $(\text{intro } \text{additive-chernoff-bound-upper-ax})$

auto

next

case *iii*
hence $a:\mu+\delta=1$ **by** *simp*
thus *?thesis unfolding a mult-1 unfolding μ -def using assms*
by (*intro additive-chernoff-bound-upper-aux-2*) *auto*
qed
qed

Based on Theorem 2 by Hoeffding [11].

lemma *hoeffding-bound-upper*:

assumes $\bigwedge i. i \in I \implies a\ i \leq b\ i$
assumes $\bigwedge i. i \in I \implies AE\ \omega\ in\ M. X\ i\ \omega \in \{a\ i..b\ i\}$
defines $n \equiv real\ (card\ I)$
defines $\mu \equiv (\sum\ i \in I. expectation\ (X\ i))$
assumes $\delta \geq 0\ (\sum\ i \in I. (b\ i - a\ i)^2) > 0$
shows $\mathcal{P}(\omega\ in\ M. (\sum\ i \in I. X\ i\ \omega) \geq \mu + \delta * n) \leq exp\ (-2*(n*\delta)^2 / (\sum\ i \in I. (b\ i - a\ i)^2))$
(is $?L \leq ?R$ **)**

proof (*cases $\delta=0$*)

case *True* **thus** *?thesis by simp*

next

case *False*

define \mathcal{X} **where** $\mathcal{X} = (\lambda i. clamp\ (a\ i)\ (b\ i) \circ X\ i)$

have *transfer-to-clamped-vars-assms*: $(\forall i \in I. AE\ \omega\ in\ M. X\ i\ \omega \in \{a\ i..b\ i\} \wedge a\ i \leq b\ i)$

using *assms(1,2)* **by** *auto*

note *ttcv = transfer-to-clamped-vars[OF transfer-to-clamped-vars-assms \mathcal{X} -def]*

note [*measurable*] = *neg-assoc-imp-measurable[OF ttcv(1)]*

define s **where** $s = (\sum\ i \in I. (b\ i - a\ i)^2)$

have *s-gt-0*: $s > 0$ **using** *assms unfolding s-def by auto*

have *I-ne*: $I \neq \{\}$ **using** *assms(6) by auto*

have *n-gt-0*: $n > 0$ **using** *I-ne fin-I unfolding n-def by auto*

define t **where** $t = 4 * \delta * n / s$

have *t-gt-0*: $t > 0$ **unfolding** *t-def using False n-gt-0 s-gt-0 assms by auto*

note *bounded' = integrable-bounded bounded-prod bounded-vec-mult-comp bounded-intros ttcv(5)*

note *\mathcal{X} -prod-mono = has-int-thatD(2)[OF neg-assoc-imp-prod-mono[OF fin-I ttcv(1), where $\eta=Fwd$]]*

have *int*: *integrable* $M\ (\mathcal{X}\ i)$ **if** $i \in I$ **for** i

using *that by (intro bounded') simp-all*

define ν **where** $\nu\ i = expectation\ (X\ i)$ **for** i

have *1*: *expectation* $(\lambda x. \mathcal{X}\ i\ x - \nu\ i) = 0$ **if** $i \in I$ **for** i

unfolding ν -def **using** *int[OF that] ttcv(2)[OF that]* **by** (*simp add:prob-space*)
have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} i \omega) \geq \mu + \delta * n)$ **using** *ttcv(3)[where $\eta = Rev$]*
by *simp*
also have $\dots = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. \mathcal{X} i \omega - \nu i) \geq \delta * n)$
using *n-gt-0 unfolding μ -def ν -def by (simp add:algebra-simps sum-subtractf)*
also have $\dots = \mathcal{P}(\omega \text{ in } M. (\prod i \in I. \exp(t * (\mathcal{X} i \omega - \nu i))) \geq \exp(t * \delta * n))$
using *t-gt-0 by (simp add: sum-distrib-left[symmetric] exp-sum[OF fin-I,symmetric])*
also have $\dots \leq \text{expectation}(\lambda \omega. (\prod i \in I. \exp(t * (\mathcal{X} i \omega - \nu i)))) / \exp(t * \delta * n)$
by (*intro integral-Markov-inequality-measure[where $A = \{\}$] bounded' AE-I2 prod-nonneg fin-I*)
simp-all
also have $\dots \leq (\prod i \in I. \text{expectation}(\lambda \omega. \exp(t * (\mathcal{X} i \omega - \nu i)))) / \exp(t * \delta * n)$
using *t-gt-0 by (intro divide-right-mono \mathcal{X} -prod-mono bounded' image-subsetI monotoneI) simp-all*
also have $\dots \leq (\prod i \in I. \exp(t^2 * ((b i - \nu i) - (a i - \nu i))^2 / 8)) / \exp(t * \delta * n)$
using *ttcv(4) 1*
by (*intro divide-right-mono prod-mono conjI Hoeffdings-lemma-bochner t-gt-0 AE-I2) simp-all*
also have $\dots = (\prod i \in I. \exp(t^2 * (b i - a i)^2 / 8)) / \exp(t * \delta * n)$ **by** *simp*
also have $\dots = \exp((t^2/8) * (\sum i \in I. (b i - a i)^2)) / \exp(t * \delta * n)$
unfolding *exp-sum[OF fin-I, symmetric]* **by** (*simp add:algebra-simps sum-distrib-left*)
also have $\dots = \exp((t^2/8) * s - t * \delta * n)$
unfolding *exp-diff s-def* **by** *simp*
also have $\dots = \exp(-2 * (n * \delta)^2 / s)$
using *s-gt-0 unfolding t-def by (simp add:divide-simps power2-eq-square)*
also have $\dots = ?R$ **unfolding** *s-def* **by** *simp*
finally show $?thesis$ **by** *simp*
qed

end

Dual and two-sided versions of Theorem 1 and 2 by Hoeffding [11].

lemma *additive-chernoff-bound-lower:*

assumes *neg-assoc X I finite I*
assumes $\bigwedge i. i \in I \implies AE \omega \text{ in } M. X i \omega \in \{0..1\} I \neq \{\}$
defines $\mu \equiv (\sum i \in I. \text{expectation}(X i)) / \text{real}(\text{card } I)$
assumes $\delta \in \{0.. \mu\} \mu \in \{0 < .. < 1\}$
shows $\mathcal{P}(\omega \text{ in } M. (\sum i \in I. X i \omega) \leq (\mu - \delta) * \text{real}(\text{card } I)) \leq \exp(-\text{real}(\text{card } I) * KL\text{-div}(\mu - \delta) \mu)$
(is $?L \leq ?R$ **)**

proof –

note [*measurable*] = *neg-assoc-imp-measurable[OF assms(1)]*

have *int[simp]: integrable M (X i) if $i \in I$ for i*

using that by (intro integrable-const-bound[where B=1] AE-mp[OF assms(3)][OF that] AE-I2]) auto
have n-gt-0: real (card I) > 0 **using** assms **by** auto

hence 0: $(1-\mu) = (\sum_{i \in I}. \text{expectation } (\lambda \omega. 1 - X i \omega)) / \text{real } (\text{card } I)$
unfolding μ -def **by** (simp add:prob-space sum-subtractf divide-simps)
have 1: neg-assoc $(\lambda i \omega. 1 - X i \omega) I$
by (intro neg-assoc-compose-simple[OF assms(2,1), where $\eta = \text{Rev}$]) (auto intro:antimonoI)

have 2: $\delta \leq (1 - (1 - \mu)) \delta \geq 0$ **using** assms **by** auto
have 3: $1-\mu \in \{0 < .. < 1\}$ **using** assms **by** auto
have ?L = $\mathcal{P}(\omega \text{ in } M. (\sum_{i \in I}. 1 - X i \omega) \geq ((1-\mu)+\delta)*\text{real } (\text{card } I))$
by (simp add:sum-subtractf algebra-simps)
also have ... $\leq \exp(-\text{real } (\text{card } I) * \text{KL-div } ((1-\mu)+\delta) (1-\mu))$
using assms(3) 1 2 3 **unfolding** 0 **by** (intro additive-chernoff-bound-upper assms(2,4)) auto
also have ... = $\exp(-\text{real } (\text{card } I) * \text{KL-div } (1-(\mu-\delta)) (1-\mu))$ **by** (simp add:algebra-simps)
also have ... = ?R **using** assms(6,7) **by** (subst KL-div-swap) (simp-all add:algebra-simps)
finally show ?thesis **by** simp
qed

lemma hoeffding-bound-lower:

assumes neg-assoc X I finite I
assumes $\bigwedge i. i \in I \implies a i \leq b i$
assumes $\bigwedge i. i \in I \implies AE \omega \text{ in } M. X i \omega \in \{a i..b i\}$
defines $n \equiv \text{real } (\text{card } I)$
defines $\mu \equiv (\sum_{i \in I}. \text{expectation } (X i))$
assumes $\delta \geq 0$ $(\sum_{i \in I}. (b i - a i)^2) > 0$
shows $\mathcal{P}(\omega \text{ in } M. (\sum_{i \in I}. X i \omega) \leq \mu - \delta * n) \leq \exp(-2*(n*\delta)^2 / (\sum_{i \in I}. (b i - a i)^2))$
(is ?L \leq ?R)

proof -

have 0: $-\mu = (\sum_{i \in I}. \text{expectation } (\lambda \omega. - X i \omega))$ **unfolding** μ -def **by** (simp add:sum-negf)
have 1: neg-assoc $(\lambda i \omega. - X i \omega) I$
by (intro neg-assoc-compose-simple[OF assms(2,1), where $\eta = \text{Rev}$]) (auto intro:antimonoI)

have ?L = $\mathcal{P}(\omega \text{ in } M. (\sum_{i \in I}. -X i \omega) \geq (-\mu)+\delta*n)$ **by** (simp add:algebra-simps sum-negf)

also have ... $\leq \exp(-2*(n*\delta)^2 / (\sum_{i \in I}. ((-a i) - (-b i))^2))$
using assms(3,4,8) **unfolding** 0 n-def **by** (intro hoeffding-bound-upper[OF 1] assms(2,4,7)) auto

also have ... = ?R **by** simp
finally show ?thesis **by** simp
qed

lemma *hoeffding-bound-two-sided*:

assumes *neg-assoc* X I *finite* I

assumes $\bigwedge i. i \in I \implies a \leq b$

assumes $\bigwedge i. i \in I \implies AE \omega \text{ in } M. X i \omega \in \{a..b\}$ $I \neq \{\}$

defines $n \equiv \text{real } (\text{card } I)$

defines $\mu \equiv (\sum i \in I. \text{expectation } (X i))$

assumes $\delta \geq 0$ $(\sum i \in I. (b - a)^2) > 0$

shows $\mathcal{P}(\omega \text{ in } M. |(\sum i \in I. X i \omega) - \mu| \geq \delta * n) \leq 2 * \exp(-2 * (n * \delta)^2 / (\sum i \in I. (b - a)^2))$

(**is** $?L \leq ?R$)

proof –

note $[\text{measurable}] = \text{neg-assoc-imp-measurable}[OF \text{ assms}(1)]$

have $?L = \mathcal{P}(\omega \text{ in } M. (\sum i \in I. X i \omega) \geq \mu + \delta * n \vee (\sum i \in I. X i \omega) \leq \mu - \delta * n)$

unfolding *abs-real-def* **by** (*intro arg-cong*[**where** $f = \text{prob}$] *Collect-cong*) *auto*

also have $\dots = \text{measure } M (\{\omega \in \text{space } M. (\sum i \in I. X i \omega) \geq \mu + \delta * n\} \cup \{\omega \in \text{space } M. (\sum i \in I. X i \omega) \leq \mu - \delta * n\})$

by (*intro arg-cong*[**where** $f = \text{prob}$] *auto*)

also have $\dots \leq \mathcal{P}(\omega \text{ in } M. (\sum i \in I. X i \omega) \geq \mu + \delta * n) + \mathcal{P}(\omega \text{ in } M. (\sum i \in I. X i \omega) \leq \mu - \delta * n)$

by (*intro measure-Un-le*) *measurable*

also have $\dots \leq \exp(-2 * (n * \delta)^2 / (\sum i \in I. (b - a)^2)) + \exp(-2 * (n * \delta)^2 / (\sum i \in I. (b - a)^2))$

unfolding *n-def* *μ-def* **by** (*intro hoeffding-bound-lower hoeffding-bound-upper add-mono assms*)

also have $\dots = ?R$ **by** *simp*

finally show $?thesis$ **by** *simp*

qed

end

end

4 The FKG inequality

The FKG inequality [9] is a generalization of Chebyshev's less known other inequality. It is sometimes referred to as Chebyshev's sum inequality. Although there is also a continuous version, which can be stated as:

$$E[fg] \geq E[f]E[g]$$

where f, g are continuous simultaneously monotone or simultaneously antimonotone functions on the Lebesgue probability space $[a, b] \subseteq \mathbb{R}$. (Ef denotes the expectation of the function.)

Note that the inequality is also true for totally ordered discrete probability spaces, for example: $\{1, \dots, n\}$ with uniform probabilities.

The FKG inequality is essentially a generalization of the above to not necessarily totally ordered spaces, but finite distributive lattices.

The proof follows the derivation in the book by Alon and Spencer [2, Ch. 6].

theory *Negative-Association-FKG-Inequality*

imports

Negative-Association-Util

Birkhoff-Finite-Distributive-Lattices.Birkhoff-Finite-Distributive-Lattices

begin

theorem *four-functions-helper:*

fixes $\varphi :: \text{nat} \Rightarrow 'a \text{ set} \Rightarrow \text{real}$

assumes *finite I*

assumes $\bigwedge i. i \in \{0..3\} \implies \varphi i \in \text{Pow } I \rightarrow \{0..\}$

assumes $\bigwedge A B. A \subseteq I \implies B \subseteq I \implies \varphi 0 A * \varphi 1 B \leq \varphi 2 (A \cup B) * \varphi 3 (A \cap B)$

shows $(\sum A \in \text{Pow } I. \varphi 0 A) * (\sum B \in \text{Pow } I. \varphi 1 B) \leq (\sum C \in \text{Pow } I. \varphi 2 C) * (\sum D \in \text{Pow } I. \varphi 3 D)$

using *assms*

proof (*induction I arbitrary:φ rule:finite-induct*)

case *empty*

then show *?case using empty by auto*

next

case (*insert x I*)

define φ' **where** $\varphi' i A = \varphi i A + \varphi i (A \cup \{x\})$ **for** $i A$

have $a: (\sum A \in \text{Pow } (\text{insert } x I). \varphi i A) = (\sum A \in \text{Pow } I. \varphi' i A)$ (**is** *?L1 = ?R1*)
for i

proof –

have $?L1 = (\sum A \in \text{Pow } I. \varphi i A) + (\sum A \in \text{insert } x I. \varphi i A)$

using *insert(1,2) unfolding Pow-insert by (intro sum.union-disjoint) auto*

also have $\dots = (\sum A \in \text{Pow } I. \varphi i A) + (\sum A \in \text{Pow } I. \varphi i (\text{insert } x A))$

using *insert(2) by (subst sum.reindex) (auto intro!:inj-onI)*

also have $\dots = ?R1$ **using** *insert(1) unfolding φ'-def sum.distrib by simp*

finally show *?thesis by simp*

qed

have $\varphi\text{-int}: \varphi 0 A * \varphi 1 B \leq \varphi 2 C * \varphi 3 D$

if $C = A \cup B$ $D = A \cap B$ $A \subseteq \text{insert } x I$ $B \subseteq \text{insert } x I$ **for** $A B C D$

using *that insert(5) by auto*

have $\varphi\text{-nonneg}: \varphi i A \geq 0$ **if** $A \subseteq \text{insert } x I$ $i \in \{0..3\}$ **for** $i A$

using *that insert(4) by auto*

have $\varphi' 0 A * \varphi' 1 B \leq \varphi' 2 (A \cup B) * \varphi' 3 (A \cap B)$ **if** $A \subseteq I$ $B \subseteq I$ **for** $A B$

proof –

define $a0 a1$ **where** $a: a0 = \varphi 0 A$ $a1 = \varphi 0 (\text{insert } x A)$

define $b0 b1$ **where** $b: b0 = \varphi 1 B$ $b1 = \varphi 1 (\text{insert } x B)$

```

define c0 c1 where c: c0 =  $\varphi$  2 (A  $\cup$  B) c1 =  $\varphi$  2 (insert x (A  $\cup$  B))
define d0 d1 where d: d0 =  $\varphi$  3 (A  $\cap$  B) d1 =  $\varphi$  3 (insert x (A  $\cap$  B))

have 0:a0 * b0  $\leq$  c0 * d0 using that unfolding a b c d by (intro  $\varphi$ -int) auto
have 1:a0 * b1  $\leq$  c1 * d0 using that insert(2) unfolding a b c d by (intro
 $\varphi$ -int) auto
have 2:a1 * b0  $\leq$  c1 * d0 using that insert(2) unfolding a b c d by (intro
 $\varphi$ -int) auto
have 3:a1 * b1  $\leq$  c1 * d1 using that insert(2) unfolding a b c d by (intro
 $\varphi$ -int) auto
have 4:a0  $\geq$  0 a1  $\geq$  0 b0  $\geq$  0 b1  $\geq$  0 using that unfolding a b by (auto
intro!:  $\varphi$ -nonneg)
have 5:c0  $\geq$  0 c1  $\geq$  0 d0  $\geq$  0 d1  $\geq$  0 using that unfolding c d by (auto
intro!:  $\varphi$ -nonneg)

consider (a) c1 = 0 | (b) d0 = 0 | (c) c1 > 0 d0 > 0 using 4 5 by argo

then have (a0 + a1) * (b0 + b1)  $\leq$  (c0 + c1) * (d0 + d1)
proof (cases)
  case a
    hence a0 * b1 = 0 a1 * b0 = 0 a1 * b1 = 0
      using 1 2 3 by (intro order-antisym mult-nonneg-nonneg 4 5;simp-all)+
    then show ?thesis unfolding distrib-left distrib-right
      using 0 4 5 by (metis add-mono mult-nonneg-nonneg)
  next
    case b
      hence a0 * b0 = 0 a0 * b1 = 0 a1 * b0 = 0
        using 0 1 2 by (intro order-antisym mult-nonneg-nonneg 4 5;simp-all)+
      then show ?thesis unfolding distrib-left distrib-right
        using 3 4 5 by (metis add-mono mult-nonneg-nonneg)
  next
    case c
      have 0  $\leq$  (c1*d0 - a0*b1) * (c1*d0 - a1*b0)
        using 1 2 by (intro mult-nonneg-nonneg) auto
      hence (a0 + a1) * (b0 + b1)*d0*c1  $\leq$  (a0*b0 + c1*d0) * (c1*d0 + a1*b1)
        by (simp add:algebra-simps)
      hence (a0 + a1) * (b0 + b1)  $\leq$  ((a0*b0)/d0 + c1) * (d0 + (a1*b1)/c1)
        using c 4 5 by (simp add:field-simps)
      also have ...  $\leq$  (c0 + c1) * (d0 + d1)
        using 0 3 c 4 5 by (intro mult-mono add-mono order.refl) (simp add:field-simps)+
      finally show ?thesis by simp
    qed

thus ?thesis unfolding  $\varphi'$ -def a b c d by auto
qed

moreover have  $\varphi'$  i  $\in$  Pow I  $\rightarrow$  {0..} if i  $\in$  {0..3} for i
  using insert(4)[OF that] unfolding  $\varphi'$ -def by (auto intro!:add-nonneg-nonneg)
ultimately show ?case unfolding a by (intro insert(3)) auto

```

qed

The following is the Ahlswede-Daykin inequality [1] also referred to by Alon and Spencer as the four functions theorem [2, Th. 6.1.1].

theorem *four-functions:*

fixes $\alpha \beta \gamma \delta :: 'a \text{ set} \Rightarrow \text{real}$

assumes *finite I*

assumes $\alpha \in \text{Pow } I \rightarrow \{0..\} \beta \in \text{Pow } I \rightarrow \{0..\} \gamma \in \text{Pow } I \rightarrow \{0..\} \delta \in \text{Pow } I \rightarrow \{0..\}$

assumes $\bigwedge A B. A \subseteq I \implies B \subseteq I \implies \alpha A * \beta B \leq \gamma (A \cup B) * \delta (A \cap B)$

assumes $M \subseteq \text{Pow } I \ N \subseteq \text{Pow } I$

shows $(\sum A \in M. \alpha A) * (\sum B \in N. \beta B) \leq (\sum C \mid \exists A \in M. \exists B \in N. C = A \cup B. \gamma C) * (\sum D \mid \exists A \in M. \exists B \in N. D = A \cap B. \delta D)$

(**is** ?L ≤ ?R)

proof –

define α' **where** $\alpha' A = (\text{if } A \in M \text{ then } \alpha A \text{ else } 0)$ **for** A

define β' **where** $\beta' B = (\text{if } B \in N \text{ then } \beta B \text{ else } 0)$ **for** B

define γ' **where** $\gamma' C = (\text{if } \exists A \in M. \exists B \in N. C = A \cup B \text{ then } \gamma C \text{ else } 0)$ **for** C

define δ' **where** $\delta' D = (\text{if } \exists A \in M. \exists B \in N. D = A \cap B \text{ then } \delta D \text{ else } 0)$ **for** D

define φ **where** $\varphi i = [\alpha', \beta', \gamma', \delta'] ! i$ **for** i

have *list-all* $(\lambda i. \varphi i \in \text{Pow } I \rightarrow \{0..\}) [0..<4]$

unfolding φ -def α' -def β' -def γ' -def δ' -def **using** *assms(2–5)*

by (*auto simp add:numeral-eq-Suc*)

hence φ -nonneg: $\varphi i \in \text{Pow } I \rightarrow \{0..\}$ **if** $i \in \{0..3\}$ **for** i

unfolding *list.pred-set* **using** *that* **by** *auto*

have $0: \varphi 0 A * \varphi 1 B \leq \varphi 2 (A \cup B) * \varphi 3 (A \cap B)$ (**is** ?L1 ≤ ?R1) **if** $A \subseteq I \ B \subseteq I$ **for** $A \ B$

proof (*cases* $A \in M \wedge B \in N$)

case *True*

have ?L1 = $\alpha A * \beta B$ **using** *True* **unfolding** φ -def α' -def β' -def **by** *auto*

also have $\dots \leq \gamma (A \cup B) * \delta (A \cap B)$ **by** (*intro that assms(6)*)

also have $\dots = ?R1$ **using** *True* **unfolding** γ' -def δ' -def φ -def **by** *auto*

finally show *?thesis* **by** *simp*

next

case *False*

hence ?L1 = 0 **unfolding** α' -def β' -def φ -def **by** *auto*

also have $\dots \leq ?R1$ **using** φ -nonneg[*of 2*] φ -nonneg[*of 3*] **that**

by (*intro mult-nonneg-nonneg*) *auto*

finally show *?thesis* **by** *simp*

qed

have *fin-pow*: *finite* ($\text{Pow } I$) **using** *assms(1)* **by** *simp*

have ?L = $(\sum A \in \text{Pow } I. \alpha' A) * (\sum B \in \text{Pow } I. \beta' B)$

unfolding α' -def β' -def **using** *assms(1,7,8)* **by** (*simp add: sum.If-cases Int-absorb1*)

also have $\dots = (\sum A \in Pow\ I. \varphi\ 0\ A) * (\sum A \in Pow\ I. \varphi\ 1\ A)$ **unfolding**
 φ -def **by simp**
also have $\dots \leq (\sum A \in Pow\ I. \varphi\ 2\ A) * (\sum A \in Pow\ I. \varphi\ 3\ A)$
by (intro four-functions-helper assms(1) φ -nonneg 0) **auto**
also have $\dots = (\sum A \in Pow\ I. \gamma'\ A) * (\sum B \in Pow\ I. \delta'\ B)$ **unfolding** φ -def
by simp
also have $\dots = ?R$
unfolding γ' -def δ' -def *sum.If-cases[OF fin-pow] sum.neutral-const add-0-right*
using *assms(7,8)*
by (intro arg-cong2[**where** $f=(*)$] *sum.cong refl*) **auto**
finally show *?thesis* **by simp**
qed

Using Birkhoff's Representation Theorem [3, 5] it is possible to generalize the previous to finite distributive lattices [2, Cor. 6.1.2].

lemma *four-functions-in-lattice:*

fixes $\alpha\ \beta\ \gamma\ \delta :: 'a :: finite-distrib-lattice \Rightarrow real$
assumes $range\ \alpha \subseteq \{0..\}$ $range\ \beta \subseteq \{0..\}$ $range\ \gamma \subseteq \{0..\}$ $range\ \delta \subseteq \{0..\}$
assumes $\bigwedge x\ y. \alpha\ x * \beta\ y \leq \gamma\ (x \sqcup y) * \delta\ (x \sqcap y)$
shows $(\sum x \in M. \alpha\ x) * (\sum y \in N. \beta\ y) \leq (\sum c \mid \exists x \in M. \exists y \in N. c = x \sqcup y. \gamma\ c) * (\sum d \mid \exists x \in M. \exists y \in N. d = x \sqcap y. \delta\ d)$
(is $?L \leq ?R$ **)**

proof –

let $?e = \lambda x :: 'a. \{ \ x \ }$
let $?f = the-inv\ ?e$

have *ran-e: range ?e = $\mathcal{O}\mathcal{J}$* **by** (rule *bij-betw-imp-surj-on[OF birkhoffs-theorem]*)
have *inj-e: inj ?e* **by** (rule *bij-betw-imp-inj-on[OF birkhoffs-theorem]*)

define $conv :: ('a \Rightarrow real) \Rightarrow 'a\ set \Rightarrow real$
where $conv\ \varphi\ I = (if\ I \in \mathcal{O}\mathcal{J}\ then\ \varphi\ (?f\ I)\ else\ 0)$ **for** $\varphi\ I$

define $\alpha'\ \beta'\ \gamma'\ \delta'$ **where** *prime-def:* $\alpha' = conv\ \alpha\ \beta' = conv\ \beta\ \gamma' = conv\ \gamma\ \delta' = conv\ \delta$

have $1: conv\ \varphi \in Pow\ \mathcal{J} \rightarrow \{0..\}$ **if** $range\ \varphi \subseteq \{(0::real)..\}$ **for** φ
using *that* **unfolding** *conv-def* **by** (intro *Pi-I*) **auto**

have $0: \alpha'\ A * \beta'\ B \leq \gamma'\ (A \cup B) * \delta'\ (A \cap B)$ **if** $A \subseteq \mathcal{J}\ B \subseteq \mathcal{J}$ **for** $A\ B$

proof (cases $A \in \mathcal{O}\mathcal{J} \wedge B \in \mathcal{O}\mathcal{J}$)

case *True*

define $x\ y$ **where** $xy: x = ?f\ A\ y = ?f\ B$

have $p0: ?e\ (x \sqcup y) = A \cup B$

using *True ran-e* **unfolding** *join-irreducibles-join-homomorphism xy*

by (*subst (1 2) f-the-inv-into-f[OF inj-e]*) **auto**

hence $p1: A \cup B \in \mathcal{O}\mathcal{J}$ **using** *ran-e* **by auto**

have $p2: ?e\ (x \sqcap y) = A \cap B$

using *True ran-e unfolding join-irreducibles-meet-homomorphism xy*
 by *(subst (1 2) f-the-inv-into-f[OF inj-e]) auto*
 hence $p3:A \cap B \in \mathcal{OJ}$ using *ran-e by auto*

have $\alpha' A * \beta' B = \alpha (?f A) * \beta (?f B)$ using *True unfolding prime-def conv-def by simp*

also have $\dots \leq \gamma (?f A \sqcup ?f B) * \delta (?f A \sqcap ?f B)$ by *(intro assms(5))*

also have $\dots = \gamma (x \sqcup y) * \delta (x \sqcap y)$ **unfolding** *xy by simp*

also have $\dots = \gamma (?f (?e (x \sqcup y))) * \delta (?f (?e (x \sqcap y)))$ by *(simp add: the-inv-f-f[OF inj-e])*

also have $\dots = \gamma (?f (A \cup B)) * \delta (?f (A \cap B))$ **unfolding** *p0 p2 by auto*

also have $\dots = \gamma' (A \cup B) * \delta' (A \cap B)$ using *p1 p3 unfolding prime-def conv-def by auto*

finally show *?thesis by simp*

next

case *False*

hence $\alpha' A * \beta' B = 0$ **unfolding** *prime-def conv-def by simp*

also have $\dots \leq \gamma' (A \cup B) * \delta' (A \cap B)$ **unfolding** *prime-def*

using *1 that assms(3,4) by (intro mult-nonneg-nonneg) auto*

finally show *?thesis by simp*

qed

define *M'* where $M' = (\lambda x. \{ x \}) ' M$

define *N'* where $N' = (\lambda x. \{ x \}) ' N$

have *ran1*: $M' \subseteq \mathcal{OJ}$ $N' \subseteq \mathcal{OJ}$ **unfolding** *M'-def N'-def using ran-e by auto*

hence *ran2*: $M' \subseteq \text{Pow } \mathcal{J}$ $N' \subseteq \text{Pow } \mathcal{J}$ **unfolding** *down-irreducibles-def by auto*

have $?f \in ?e ' S \rightarrow S$ **for** *S* using *inj-e by (simp add: Pi-iff the-inv-f-f)*

hence *bij-betw*: *bij-betw* $?f (?e ' S)$ **for** *S* :: 'a set

by *(intro bij-betwI[where g=?e] the-inv-f-f f-the-inv-into-f inj-e) auto*

have *a*: $\{C. \exists A \in M'. \exists B \in N'. C = A \cup B\} = ?e ' \{c. \exists x \in M. \exists y \in N. c = x \sqcup y\}$

unfolding *M'-def N'-def Set.bex-simps join-irreducibles-join-homomorphism[symmetric]*

by *auto*

have *b*: $\{D. \exists A \in M'. \exists B \in N'. D = A \cap B\} = ?e ' \{c. \exists x \in M. \exists y \in N. c = x \sqcap y\}$

unfolding *M'-def N'-def Set.bex-simps join-irreducibles-meet-homomorphism[symmetric]*

by *auto*

have *M'-N'-un-ran*: $\{C. \exists A \in M'. \exists B \in N'. C = A \cup B\} \subseteq \mathcal{OJ}$

unfolding *a using ran-e by auto*

have *M'-N'-int-ran*: $\{C. \exists A \in M'. \exists B \in N'. C = A \cap B\} \subseteq \mathcal{OJ}$

unfolding *b using ran-e by auto*

have *?L* = $(\sum A \in M'. \alpha (?f A)) * (\sum A \in N'. \beta (?f A))$

unfolding *M'-def N'-def*

by *(intro arg-cong2[where f=(*)] sum.reindex-bij-betw[symmetric] bij-betw)*

also have $\dots = (\sum A \in M'. \alpha' A) * (\sum A \in N'. \beta' A)$

unfolding *prime-def conv-def* **using** *ran1* **by** (*intro arg-cong2*[**where** $f=(*)$]
sum.cong refl) *auto*
also have $\dots \leq (\sum C \mid \exists A \in M'. \exists B \in N'. C = A \cup B. \gamma' C) * (\sum D \mid \exists A \in M'. \exists B \in N'. D = A \cap B. \delta' D)$
using *ran2* **by** (*intro four-functions*[**where** $I=\mathcal{J}$] *0*) (*auto intro!:*1 *assms simp:prime-def*)
also have $\dots = (\sum C \mid \exists A \in M'. \exists B \in N'. C = A \cup B. \gamma(?f C)) * (\sum D \mid \exists A \in M'. \exists B \in N'. D = A \cap B. \delta(?f D))$
using *M'-N'-un-ran M'-N'-int-ran* **unfolding** *prime-def conv-def*
by (*intro arg-cong2*[**where** $f=(*)$] *sum.cong refl*) *auto*
also have $\dots = ?R$
unfolding *a b* **by** (*intro arg-cong2*[**where** $f=(*)$] *sum.reindex-bij-betw bij-betw*)
finally show *?thesis* **by** *simp*
qed

theorem *fk-g-inequality*:

fixes $\mu :: 'a :: \text{finite-distrib-lattice} \Rightarrow \text{real}$
assumes $\text{range } \mu \subseteq \{0..\}$ $\text{range } f \subseteq \{0..\}$ $\text{range } g \subseteq \{0..\}$
assumes $\bigwedge x y. \mu x * \mu y \leq \mu (x \sqcup y) * \mu (x \sqcap y)$
assumes *mono f mono g*
shows $(\sum x \in \text{UNIV}. \mu x * f x) * (\sum x \in \text{UNIV}. \mu x * g x) \leq (\sum x \in \text{UNIV}. \mu x * f x * g x) * \text{sum } \mu \text{ UNIV}$
(is $?L \leq ?R$ **)**

proof –

define α **where** $\alpha x = \mu x * f x$ **for** x
define β **where** $\beta x = \mu x * g x$ **for** x
define γ **where** $\gamma x = \mu x * f x * g x$ **for** x
define δ **where** $\delta x = \mu x$ **for** x

have $0 : f x \geq 0$ **if** $\text{range } f \subseteq \{0..\}$ **for** $f :: 'a \Rightarrow \text{real}$ **and** x
using *that* **by** *auto*

note $\mu f g \text{-nonneg} = 0[\text{OF } \text{assms}(1)] \ 0[\text{OF } \text{assms}(2)] \ 0[\text{OF } \text{assms}(3)]$

have $1 : \alpha x * \beta y \leq \gamma (x \sqcup y) * \delta (x \sqcap y)$ **(is** $?L1 \leq ?R1$ **)** **for** $x y$

proof –

have $?L1 = (\mu x * \mu y) * (f x * g y)$ **unfolding** $\alpha\text{-def } \beta\text{-def}$ **by** (*simp add:ac-simps*)

also have $\dots \leq (\mu (x \sqcup y) * \mu (x \sqcap y)) * (f x * g y)$

using *assms(2,3)* **by** (*intro mult-right-mono assms(4) mult-nonneg-nonneg*)

auto

also have $\dots \leq (\mu (x \sqcup y) * \mu (x \sqcap y)) * (f (x \sqcup y) * g (x \sqcup y))$

using $\mu f g \text{-nonneg}$

by (*intro mult-left-mono mult-mono monoD*[*OF assms(5)*] *monoD*[*OF assms(6)*] *mult-nonneg-nonneg*)

simp-all

also have $\dots = ?R1$ **unfolding** $\gamma\text{-def } \delta\text{-def}$ **by** *simp*

finally show *?thesis* **by** *simp*

qed

have $?L = (\sum x \in UNIV. \alpha x) * (\sum y \in UNIV. \beta y)$ **unfolding** α -def β -def **by** *simp*
also have $\dots \leq (\sum c \mid \exists x \in UNIV. \exists y \in UNIV. c = x \sqcup y. \gamma c) * (\sum d \mid \exists x \in UNIV. \exists y \in UNIV. d = x \sqcap y. \delta d)$
using μ fg-nonneg **by** (*intro four-functions-in-lattice 1*) (*auto simp: α -def β -def γ -def δ -def*)
also have $\dots = (\sum x \in UNIV. \gamma x) * (\sum x \in UNIV. \delta x)$
using *sup.idem*[**where** $'a = 'a$] *inf.idem*[**where** $'a = 'a$]
by (*intro arg-cong2*[**where** $f = (*)$] *sum.cong refl UNIV-eq-I*[*symmetric*] *CollectI*) (*metis UNIV-I*)
also have $\dots = ?R$ **unfolding** γ -def δ -def **by** *simp*
finally show $?thesis$ **by** *simp*
qed

theorem *fkq-inequality-gen*:

fixes $\mu :: 'a :: \text{finite-distrib-lattice} \Rightarrow \text{real}$
assumes $\text{range } \mu \subseteq \{0..\}$
assumes $\bigwedge x y. \mu x * \mu y \leq \mu (x \sqcup y) * \mu (x \sqcap y)$
assumes *monotone* (\leq) (\leq_{τ}) *f* *monotone* (\leq) (\leq_{σ}) *g*
shows $(\sum x \in UNIV. \mu x * f x) * (\sum x \in UNIV. \mu x * g x) \leq_{\tau * \sigma} (\sum x \in UNIV. \mu x * f x * g x) * \text{sum } \mu UNIV$
(is $?L \leq_{?x} ?R$)

proof –

define a **where** $a = \text{max} (\text{MAX } x. -f x * (\pm_{\tau})) (\text{MAX } x. -g x * (\pm_{\sigma}))$

define f' **where** $f' x = a + f x * (\pm_{\tau})$ **for** x

define g' **where** $g' x = a + g x * (\pm_{\sigma})$ **for** x

have f' -mono: *mono* f' **unfolding** f' -def **using** *monotoneD*[*OF assms(3)*]

by (*intro monoI add-mono order.refl*) (*cases* τ , *auto simp: comp-def ac-simps*)

have g' -mono: *mono* g' **unfolding** g' -def **using** *monotoneD*[*OF assms(4)*]

by (*intro monoI add-mono order.refl*) (*cases* σ , *auto simp: comp-def ac-simps*)

have f' -nonneg: $f' x \geq 0$ **for** x

unfolding f' -def a -def *max-add-distrib-left*

by (*intro max.coboundedI1*) (*auto intro!: Max.coboundedI simp: algebra-simps real-0-le-add-iff*)

have g' -nonneg: $g' x \geq 0$ **for** x

unfolding g' -def a -def *max-add-distrib-left*

by (*intro max.coboundedI2*) (*auto intro!: Max.coboundedI simp: algebra-simps real-0-le-add-iff*)

let $?M = (\sum x \in UNIV. \mu x)$

let $?sum = (\lambda f. (\sum x \in UNIV. \mu x * f x))$

have $(\pm_{\tau * \sigma}) * ?L = ?sum (\lambda x. f x * (\pm_{\tau})) * ?sum (\lambda x. g x * (\pm_{\sigma}))$

by (simp add:ac-simps sum-distrib-left[symmetric] dir-mult-hom del:rel-dir-mult)
 also have ... = (?sum (λx. (f x*(±τ)+a))-?M*a) * (?sum (λx. (g x*(±σ)+a))-?M*a)
 by (simp add:algebra-simps sum.distrib sum-distrib-left)
 also have ... = (?sum f')*(?sum g') - ?M*a*?sum f'- ?M*a*?sum g' +
 ?M*?M*a*a
 unfolding f'-def g'-def by (simp add:algebra-simps)
 also have ... ≤ ((∑ x∈UNIV. μ x*f' x*g' x)*?M) - ?M*a*?sum f'- ?M*a*?sum
 g' + ?M*?M*a*a
 using f'-nonneg g'-nonneg
 by (intro diff-mono add-mono order.refl fkg-inequality assms(1,2) f'-mono
 g'-mono) auto
 also have ... = ?sum (λx. (f x*(±τ))*(g x*(±σ)))*?M
 unfolding f'-def g'-def by (simp add:algebra-simps sum.distrib sum-distrib-left[symmetric])
 also have ... = (±τ*σ) * ?R
 by (simp add:ac-simps sum.distrib sum-distrib-left[symmetric] dir-mult-hom
 del:rel-dir-mult)
 finally have (±τ*σ) * ?L ≤ (±τ*σ) * ?R by simp
 thus ?thesis by (cases τ*σ, auto)
 qed

theorem fkg-inequality-pmf:

fixes M :: ('a :: finite-distrib-lattice) pmf
 fixes f g :: 'a ⇒ real
 assumes ∧x y. pmf M x * pmf M y ≤ pmf M (x ⊔ y) * pmf M (x ⊓ y)
 assumes monotone (≤) (≤≥τ) f monotone (≤) (≤≥σ) g
 shows (∫ x. f x ∂M) * (∫ x. g x ∂M) ≤≥τ * σ (∫ x. f x * g x ∂M)
 (is ?L ≤≥. ?R)
 proof –
 have 0: ?L = (∑ a∈UNIV. pmf M a * f a) * (∑ a∈UNIV. pmf M a * g a)
 by (subst (1 2) integral-measure-pmf-real[**where** A=UNIV]) (auto simp:ac-simps)
 have ?R = ?R * (∫ x. 1 ∂M) by simp
 also have ... = (∑ a∈UNIV. pmf M a*f a*g a) * sum (pmf M) UNIV
 by (subst (1 2) integral-measure-pmf-real[**where** A=UNIV]) (auto simp:ac-simps)
 finally have 1: ?R = (∑ a∈UNIV. pmf M a*f a*g a) * sum (pmf M) UNIV by
 simp
 thus ?thesis unfolding 0 1
 by (intro fkg-inequality-gen assms) auto
 qed

end

5 Preliminary Results on Lattices

This entry establishes a few missing lemmas for the set-based theory of lattices from “HOL-Algebra”. In particular, it introduces the sublocale for distributive lattices.

More crucially, a transfer theorem which can be used in conjunction with the Types-To-Sets mechanism to be able to work with locally defined finite

distributive lattices.

This is being needed for the verification of the negative association of permutation distributions in Section 6.

```
theory Negative-Association-More-Lattices
  imports HOL-Algebra.Lattice
begin
```

Lemma 1 Birkhoff Lattice Theory, p.8, L3

```
lemma (in lattice) meet-assoc-law:
  assumes  $x \in \text{carrier } L \ y \in \text{carrier } L \ z \in \text{carrier } L$ 
  shows  $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$ 
  using assms by (metis (full-types) eq-is-equal weak-meet-assoc)
```

Lemma 1 Birkhoff Lattice Theory, p.8, L3

```
lemma (in lattice) join-assoc-law:
  assumes  $x \in \text{carrier } L \ y \in \text{carrier } L \ z \in \text{carrier } L$ 
  shows  $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$ 
  using assms by (metis (full-types) eq-is-equal weak-join-assoc)
```

Lemma 1 Birkhoff Lattice Theory, p.8, L4

```
lemma (in lattice) absorbtion-law:
  assumes  $x \in \text{carrier } L \ y \in \text{carrier } L$ 
  shows  $x \sqcap (x \sqcup y) = x \ x \sqcup (x \sqcap y) = x$ 
proof –
  have  $x \sqsubseteq x \sqcup y$  using assms join-left by auto
  hence  $x = x \sqcap (x \sqcup y)$  using assms by (intro iffD1[OF le-iff-join]) auto
  thus  $x \sqcap (x \sqcup y) = x$  by simp

  have  $x \sqcap y \sqsubseteq x$  using assms meet-left by auto
  hence  $(x \sqcap y) \sqcup x = x$  using assms le-iff-meet by (intro iffD1[OF le-iff-meet])
auto
  thus  $x \sqcup (x \sqcap y) = x$  using join-comm by metis
qed
```

Theorem 9 Birkhoff Lattice Theory, p.11

```
lemma (in lattice) distrib-laws-equiv:
  defines meet-distrib  $\equiv (\forall x \ y \ z. \{x,y,z\} \subseteq \text{carrier } L \longrightarrow (x \sqcap (y \sqcup z)) = (x \sqcap y) \sqcup (x \sqcap z))$ 
  defines join-distrib  $\equiv (\forall x \ y \ z. \{x,y,z\} \subseteq \text{carrier } L \longrightarrow (x \sqcup (y \sqcap z)) = (x \sqcup y) \sqcap (x \sqcup z))$ 
  shows meet-distrib  $\longleftrightarrow$  join-distrib
proof
  assume a:meet-distrib
  have  $(x \sqcup y) \sqcap (x \sqcup z) = x \sqcup (y \sqcap z)$  (is  $?L = ?R$ ) if  $\{x,y,z\} \subseteq \text{carrier } L$  for  $x \ y \ z$ 
  proof –
  have  $?L = ((x \sqcup y) \sqcap x) \sqcup ((x \sqcup y) \sqcap z)$  using a that unfolding meet-distrib-def
by simp
```

also have $\dots = x \sqcup (z \sqcap (x \sqcup y))$ **using** *that absorbtion-law meet-comm by (metis insert-subset)*
also have $\dots = x \sqcup ((z \sqcap x) \sqcup (z \sqcap y))$ **using** *a that unfolding meet-distrib-def by simp*
also have $\dots = (x \sqcup (z \sqcap x)) \sqcup (z \sqcap y)$ **using** *that meet-assoc-law join-assoc-law by simp*
also have $\dots = x \sqcup (z \sqcap y)$ **using** *that absorbtion-law meet-comm by (metis insert-subset)*
also have $\dots = ?R$ **by** *(metis meet-comm)*
finally show *?thesis* **by** *simp*
qed
thus *join-distrib unfolding join-distrib-def by auto*
next
assume *a:join-distrib*
have $(x \sqcap y) \sqcup (x \sqcap z) = x \sqcap (y \sqcup z)$ **(is** $?L = ?R$ **if** $\{x, y, z\} \subseteq \text{carrier } L$ **for** x y z
proof $-$
have $?L = ((x \sqcap y) \sqcup x) \sqcap ((x \sqcap y) \sqcup z)$ **using** *a that unfolding join-distrib-def by simp*
also have $\dots = x \sqcap (z \sqcup (x \sqcap y))$ **using** *that absorbtion-law join-comm by (metis insert-subset)*
also have $\dots = x \sqcap ((z \sqcup x) \sqcap (z \sqcup y))$ **using** *a that unfolding join-distrib-def by simp*
also have $\dots = (x \sqcap (z \sqcup x)) \sqcap (z \sqcup y)$ **using** *that meet-assoc-law join-assoc-law by simp*
also have $\dots = x \sqcap (z \sqcup y)$ **using** *that absorbtion-law join-comm by (metis insert-subset)*
also have $\dots = ?R$ **by** *(metis join-comm)*
finally show *?thesis* **by** *simp*
qed
thus *meet-distrib unfolding meet-distrib-def by auto*
qed

lemma *(in lattice) lub-unique-set:*

assumes *is-lub* L z S

shows $z = \bigsqcup S$

proof $-$

have *a:is-lub* L z' $S \implies z = z'$ **for** z'

using *least-unique assms* **by** *simp*

show *?thesis*

unfolding *sup-def*

by *(rule someI2[where a=z], rule assms(1), rule a)*

qed

lemma *(in lattice) lub-unique:*

assumes *is-lub* L z $\{x, y\}$

shows $z = x \sqcup y$

using *lub-unique-set[OF assms]* **unfolding** *join-def* **by** *auto*

lemma (in *lattice*) *glb-unique-set*:

assumes *is-glb* L z S

shows $z = \bigcap S$

proof –

have $a:is-glb$ L z' $S \implies z = z'$ **for** z'

using *greatest-unique assms(1)* **by** *simp*

show *?thesis*

unfolding *meet-def inf-def*

by (*rule someI2[where a=z]*, *rule assms(1)*, *rule a*)

qed

lemma (in *lattice*) *glb-unique*:

assumes *is-glb* L z $\{x,y\}$

shows $z = x \sqcap y$

using *glb-unique-set[OF assms]* **unfolding** *meet-def* **by** *auto*

lemma (in *lattice*) *inf-lower*:

assumes $S \subseteq \text{carrier } L$ $s \in S$ *finite* S

shows $\bigcap S \sqsubseteq s$

proof –

have *is-glb* L $(\bigcap S)$ S **using** *assms(2)* **by** (*intro finite-inf-greatest assms(1,3)*) *auto*

hence $(\bigcap S) \in \text{Lower } L$ S **using** *greatest-mem* **by** *metis*

thus *?thesis* **using** *assms(1,2)* **by** *auto*

qed

lemma (in *lattice*) *sup-upper*:

assumes $S \subseteq \text{carrier } L$ $s \in S$ *finite* S

shows $s \sqsubseteq \bigcup S$

proof –

have *is-lub* L $(\bigcup S)$ S **using** *assms(2)* **by** (*intro finite-sup-least assms(1,3)*) *auto*

hence $(\bigcup S) \in \text{Upper } L$ S **using** *least-mem* **by** *metis*

thus *?thesis* **using** *assms(1,2)* **by** *auto*

qed

locale *distrib-lattice* = *lattice* +

assumes *max-distrib*:

$x \in \text{carrier } L \implies y \in \text{carrier } L \implies z \in \text{carrier } L \implies (x \sqcap (y \sqcup z)) = (x \sqcap y) \sqcup (x \sqcap z)$

begin

lemma *min-distrib*:

assumes $x \in \text{carrier } L$ $y \in \text{carrier } L$ $z \in \text{carrier } L$

shows $(x \sqcup (y \sqcap z)) = (x \sqcup y) \sqcap (x \sqcup z)$

proof –

have $a:\forall x y z. \{x, y, z\} \subseteq \text{carrier } L \longrightarrow x \sqcap (y \sqcup z) = x \sqcap y \sqcup x \sqcap z$ **using** *max-distrib* **by** *auto*

show *?thesis* **using** *iffD1[OF distrib-laws-equiv a]* *assms* **by** *simp*

qed

end

locale *finite-ne-distrib-lattice* = *distrib-lattice* +
assumes *non-empty-carrier*: $\text{carrier } L \neq \{\}$
assumes *finite-carrier*: *finite* (*carrier* *L*)
begin

lemma *bounded-lattice-axioms-1*: $\exists x. \text{least } L \ x \ (\text{carrier } L)$
proof –
have $\bigcap \text{carrier } L \in \text{Lower } L \ (\text{carrier } L)$
by (*intro greatest-mem*[**where** $L=L$] *finite-inf-greatest*[*OF* *finite-carrier* - *non-empty-carrier*])
auto
hence $\forall x \in \text{carrier } L. (\bigcap \text{carrier } L) \sqsubseteq x$ **unfolding** *Lower-def* **by** *auto*
moreover **have** $\bigcap \text{carrier } L \in \text{carrier } L$
using *finite-inf-closed*[*OF* *finite-carrier* - *non-empty-carrier*] **by** *auto*
ultimately **have** *least* *L* ($\bigcap \text{carrier } L$) (*carrier* *L*)
unfolding *least-def* **by** *auto*

thus *?thesis* **by** *auto*
qed

lemma *bounded-lattice-axioms-2*: $\exists x. \text{greatest } L \ x \ (\text{carrier } L)$
proof –
have $\bigcup \text{carrier } L \in \text{Upper } L \ (\text{carrier } L)$
by (*intro least-mem*[**where** $L=L$] *finite-sup-least*[*OF* *finite-carrier* - *non-empty-carrier*])
auto
hence $\forall x \in \text{carrier } L. x \sqsubseteq (\bigcup \text{carrier } L)$ **unfolding** *Upper-def* **by** *auto*
moreover **have** $\bigcup \text{carrier } L \in \text{carrier } L$
using *finite-sup-closed*[*OF* *finite-carrier* - *non-empty-carrier*] **by** *auto*
ultimately **have** *greatest* *L* ($\bigcup \text{carrier } L$) (*carrier* *L*)
unfolding *greatest-def* **by** *auto*

thus *?thesis* **by** *auto*
qed

sublocale *bounded-lattice*
using *bounded-lattice-axioms-1* *bounded-lattice-axioms-2*
by (*unfold-locales*) *auto*

lemma *inf-empty*: $\bigcap \{\} = \top$
proof –
have *is-glb* *L* $\top \ \{\}$ **using** *top-greatest* **by** *simp*
thus *?thesis* **using** *glb-unique-set* **by** *auto*
qed

lemma *inf-closed*: $S \subseteq \text{carrier } L \implies \bigcap S \in \text{carrier } L$
using *finite-carrier inf-empty top-closed finite-inf-closed*
by (*metis finite-subset*)

lemma *inf-insert*:

assumes $x \in \text{carrier } L$ $S \subseteq \text{carrier } L$

shows $\sqcap (\text{insert } x S) = x \sqcap (\sqcap S)$

proof –

have *fin-S*: *finite S* **using** *finite-carrier* *assms(2)* *finite-subset* **by** *metis*

have *inf-S-carr*: $\sqcap S \in \text{carrier } L$ **using** *inf-closed[OF assms(2)]* **by** *force*

have $x \sqcap (\sqcap S) \sqsubseteq s$ **if** $s \in S$ **for** s

proof –

have $\sqcap S \sqsubseteq s$ **using** *that fin-S* *assms(2)*

by (*metis empty-iff finite-inf-greatest greatest-Lower-below*)

moreover **have** $x \sqcap (\sqcap S) \sqsubseteq \sqcap S$ **using** *inf-S-carr* *assms(1)* *meet-right* **by**

metis

ultimately **show** *?thesis* **using** *inf-S-carr* *meet-closed*

by (*meson assms le-trans subsetD that*)

qed

moreover **have** $x \sqcap (\sqcap S) \sqsubseteq x$ **using** *inf-S-carr* *assms(1)* *meet-left* **by** *metis*

ultimately **have** $x \sqcap (\sqcap S) \in \text{Lower } L$ (*insert x S*)

using *assms(1)* *meet-closed* *inf-S-carr* **unfolding** *Lower-def* **by** *auto*

moreover **have** $y \sqsubseteq (x \sqcap (\sqcap S))$ **if** $y \in \text{Lower } L$ (*insert x S*) **for** y

proof–

have *y-carr*: $y \in \text{carrier } L$ **using** *that assms* **unfolding** *Lower-def* **by** *auto*

have *y-lb*: $y \sqsubseteq x$ **using** *that assms* **unfolding** *Lower-def* **by** *auto*

moreover **have** $y \in \text{Lower } L$ S **using** *that unfolding* *Lower-def* **by** *auto*

hence $y \sqsubseteq \sqcap S$ **using** *finite-inf-greatest[OF fin-S assms(2)]*

by (*metis greatest-le inf-empty top-higher y-carr*)

ultimately **show** *?thesis*

using *y-carr* *inf-S-carr* *assms(1)* *meet-le* **by** *simp*

qed

ultimately **have** *is-glb* L $(x \sqcap (\sqcap S))$ (*insert x S*) **by** (*simp add: greatest-def*)

thus *?thesis* **by** (*intro glb-unique-set[symmetric]*)

qed

lemma *sup-empty*: $\sqcup \{\} = \perp$

proof –

have *is-lub* L \perp $\{\}$ **using** *bottom-least* **by** *simp*

thus *?thesis* **using** *lub-unique-set* **by** *auto*

qed

lemma *sup-closed*: $S \subseteq \text{carrier } L \implies \sqcup S \in \text{carrier } L$

using *finite-carrier* *sup-empty* *bottom-closed* *finite-sup-closed*

by (*metis finite-subset*)

lemma *sup-insert*:

assumes $x \in \text{carrier } L$ $S \subseteq \text{carrier } L$

shows $\sqcup (\text{insert } x S) = x \sqcup (\sqcup S)$

proof –

have $fin-S$: *finite S* **using** *finite-carrier assms(2) finite-subset* **by** *metis*
have $sup-S-carr$: $\sqcup S \in carrier L$ **using** *sup-closed[OF assms(2)]* **by** *force*

have $s \sqsubseteq x \sqcup (\sqcup S)$ **if** $s \in S$ **for** s
proof –
have $s \sqsubseteq \sqcup S$ **using** *that fin-S assms(2)*
by (*metis empty-iff finite-sup-least least-Upper-above*)
moreover have $\sqcup S \sqsubseteq x \sqcup (\sqcup S)$ **using** *sup-S-carr assms(1) join-right* **by**
metis
ultimately show *?thesis* **using** *sup-S-carr join-closed assms*
by (*meson le-trans subsetD that*)
qed
moreover have $x \sqsubseteq x \sqcup (\sqcup S)$ **using** *sup-S-carr assms(1) join-left* **by** *metis*
ultimately have $x \sqcup (\sqcup S) \in Upper L$ (*insert x S*)
using *assms(1) sup-S-carr unfolding Upper-def* **by** *auto*
moreover have $x \sqcup (\sqcup S) \sqsubseteq y$ **if** $y \in Upper L$ (*insert x S*) **for** y
proof–
have $y-carr$: $y \in carrier L$ **using** *that assms unfolding Lower-def* **by** *auto*
have $y-lb$: $x \sqsubseteq y$ **using** *that assms* **by** *auto*

moreover have $y \in Upper L S$ **using** *that unfolding Upper-def* **by** *auto*
hence $\sqcup S \sqsubseteq y$ **using** *finite-sup-least[OF fin-S assms(2)]*
using *least-le sup-empty bottom-lower y-carr* **by** *metis*
ultimately show *?thesis*
using *y-carr sup-S-carr assms(1) join-le* **by** *simp*
qed
ultimately have *is-lub L (x \sqcup ($\sqcup S$)) (insert x S)* **by** (*simp add: least-def*)
thus *?thesis* **by** (*intro lub-unique-set[symmetric]*)
qed

lemma *inf-carrier*: $\sqcap (carrier L) = \perp$
proof –
have $\sqcap carrier L \in Lower L (carrier L)$
by (*intro greatest-mem[where L=L] finite-inf-greatest[OF finite-carrier - non-empty-carrier]*)
auto
hence $\forall x \in carrier L. (\sqcap carrier L) \sqsubseteq x$ **unfolding** *Lower-def* **by** *auto*
moreover have $\sqcap carrier L \in carrier L$
using *finite-inf-closed[OF finite-carrier - non-empty-carrier]* **by** *auto*
ultimately show *?thesis* **by** (*intro bottom-eq*) *auto*
qed

lemma *sup-carrier*: $\sqcup (carrier L) = \top$
proof –
have $\sqcup carrier L \in Upper L (carrier L)$
by (*intro least-mem[where L=L] finite-sup-least[OF finite-carrier - non-empty-carrier]*)
auto
hence $\forall x \in carrier L. x \sqsubseteq (\sqcup carrier L)$ **unfolding** *Upper-def* **by** *auto*
moreover have $\sqcup carrier L \in carrier L$
using *finite-sup-closed[OF finite-carrier - non-empty-carrier]* **by** *auto*

ultimately show *?thesis* by (intro top-eq) auto
qed

lemma *transfer-to-type*:

```

assumes finite (carrier L) type-definition Rep Abs (carrier L)
defines inf'  $\equiv (\lambda M. \text{Abs } (\sqcap \text{ Rep } 'M))$ 
defines sup'  $\equiv (\lambda M. \text{Abs } (\sqcup \text{ Rep } 'M))$ 
defines join'  $\equiv (\lambda x y. \text{Abs } (\text{Rep } x \sqcap \text{Rep } y))$ 
defines le'  $\equiv (\lambda x y. (\text{Rep } x \sqsubseteq \text{Rep } y))$ 
defines less'  $\equiv (\lambda x y. (\text{Rep } x \sqsubset \text{Rep } y))$ 
defines meet'  $\equiv (\lambda x y. (\text{Abs } (\text{Rep } x \sqcup \text{Rep } y)))$ 
defines bot'  $\equiv (\text{Abs } \perp :: 'c)$ 
defines top'  $\equiv \text{Abs } \top$ 
shows class.finite-distrib-lattice inf' sup' join' le' less' meet' bot' top'
proof –
  interpret type-definition Rep Abs (carrier L)
  using assms(2) by auto

```

note *defs* = *inf'-def sup'-def join'-def le'-def less'-def meet'-def bot'-def bot'-def top'-def*

note *td* = *Rep Rep-inverse Abs-inverse inf-closed sup-closed meet-closed join-closed Rep-range*

```

have class-lattice: class.lattice join' le' less' meet'
  unfolding defs using td
proof (unfold-locales, goal-cases)
  case 1 thus ?case unfolding lless-eq by auto
next
  case 2 thus ?case by (metis le-refl)
next
  case 3 thus ?case by (metis le-trans)
next
  case 4 thus ?case by (meson Rep-inject local.le-antisym)
next
  case 5 thus ?case by (metis meet-left)
next
  case 6 thus ?case by (metis meet-right)
next
  case 7 thus ?case by (metis meet-le)
next
  case 8 thus ?case by (metis join-left)
next
  case 9 thus ?case by (metis join-right)
next
  case 10 thus ?case by (metis join-le)
qed

```

have *class-distrib-lattice*: class.distrib-lattice join' le' less' meet'

```

unfolding class.distrib-lattice-def eqTrueI[OF class-lattice]
unfolding defs class.distrib-lattice-axioms-def using td
using min-distrib by auto

have class-finite: class.finite TYPE('c)
by (unfold-locales) (metis assms(1) Abs-image finite-imageI)

have class-finite-lattice: class.finite-lattice inf' sup' join' le' less' meet' bot' top'
unfolding class.finite-lattice-def eqTrueI[OF class-lattice] eqTrueI[OF class-finite]
unfolding defs class.distrib-lattice-axioms-def class.finite-lattice-axioms-def using td
ing td
proof (intro conjI TrueI, goal-cases)
  case 1 thus ?case using sup-carrier inf-empty by simp
next
  case 2 thus ?case unfolding image-insert by (metis inf-insert image-subsetI)
next
  case 3 thus ?case using inf-carrier sup-empty by simp
next
  case 4 thus ?case unfolding image-insert by (metis sup-insert image-subsetI)
next
  case 5 thus ?case using inf-carrier by simp
next
  case 6 thus ?case using sup-carrier by simp
qed

show ?thesis
using class-finite-lattice class-distrib-lattice
unfolding class.finite-distrib-lattice-def by auto
qed

end

end

```

6 Permutation Distributions

One of the fundamental examples for negatively associated random variables are permutation distributions.

Let x_1, \dots, x_n be n (not-necessarily) distinct values from a totally ordered set, then we choose a permutation $\sigma : \{0, \dots, n-1\} \rightarrow \{0, \dots, n-1\}$ uniformly at random. Then the random variables defined by $X_i(\sigma) = x_{\sigma(i)}$ are negatively associated.

An important special case is the case where x consists of 1 one and $(n-1)$ zeros, modelling randomly putting a ball into one of n bins. Of course the process can be repeated independently, the resulting distribution is also referred to as the balls into bins process. Because of the closure properties established before, it is possible to conclude that the number of hits of each

bin in such a process are also negatively associated random variables.

In this section, we will derive that permutation distributions are negatively associated. The proof follows Dubhashi [8, Th. 10] closely. A very short proof was presented in the work by Joag-Dev [13], which, however, is incomplete.

theory *Negative-Association-Permutation-Distributions*

imports

Negative-Association-Definition

Negative-Association-FKG-Inequality

Negative-Association-More-Lattices

Finite-Fields.Finite-Fields-More-PMF

HOL-Types-To-Sets.Types-To-Sets

Executable-Randomized-Algorithms.Randomized-Algorithm

Twelvefold-Way.Card-Bijections

begin

The following introduces a lattice for n -element subsets of a finite set (with size larger or equal to n .) A subset x is smaller or equal to y , if the smallest element of x is smaller or equal to the smallest element of y , the second smallest element of x is smaller or equal to the second smallest element of y , etc.)

The lattice is introduced without name by Dubhashi [7, Example 7].

definition *le-ordered-set-lattice* :: ('a::linorder) set \Rightarrow 'a set \Rightarrow bool

where *le-ordered-set-lattice* $S T = \text{list-all2 } (\leq) (\text{sorted-list-of-set } S) (\text{sorted-list-of-set } T)$

definition *ordered-set-lattice* :: ('a :: linorder) set \Rightarrow nat \Rightarrow 'a set gorder

where *ordered-set-lattice* $S n =$

(| *carrier* = { $T. T \subseteq S \wedge \text{finite } T \wedge \text{card } T = n$ },

eq = (=),

le = *le-ordered-set-lattice* |)

definition *osl-repr* :: ('a :: linorder) set \Rightarrow nat \Rightarrow 'a set \Rightarrow nat \Rightarrow 'a

where *osl-repr* $S n e = (\lambda i \in \{..<n\}. \text{sorted-list-of-set } e ! i)$

lemma *osl-carr-sorted-list-of-set*:

assumes *finite* $S n \leq \text{card } S$

assumes $s \in \text{carrier } (\text{ordered-set-lattice } S n)$

defines $t \equiv \text{sorted-list-of-set } s$

shows *finite* $s \text{ card } s = n \ s \subseteq S \ \text{length } t = n \ \text{set } t = s \ \text{sorted-wrt } (<) \ t$

using *assms* **unfolding** *ordered-set-lattice-def* **by** *auto*

lemma *ordered-set-lattice-carrier-intro*:

assumes *finite* $S n \leq \text{card } S$

assumes $\text{set } s \subseteq S \ \text{distinct } s \ \text{length } s = n$

shows $\text{set } s \in \text{carrier } (\text{ordered-set-lattice } S n)$

using *assms* *distinct-card* **unfolding** *ordered-set-lattice-def* **by** *auto*

lemma *osl-list-repr-inj*:
assumes *finite S n ≤ card S*
assumes *s ∈ carrier (ordered-set-lattice S n)*
assumes *t ∈ carrier (ordered-set-lattice S n)*
assumes $\bigwedge i. \text{osl-repr } S \ n \ s \ i = \text{osl-repr } S \ n \ t \ i$
shows $s = t$
proof –
note $c1 = \text{osl-carr-sorted-list-of-set}[OF \ \text{assms}(1,2,3)]$
note $c2 = \text{osl-carr-sorted-list-of-set}[OF \ \text{assms}(1,2,4)]$

have *sorted-list-of-set s ! i = sorted-list-of-set t ! i if i < n for i*
using *assms(5) that unfolding osl-repr-def lessThan-iff restrict-def by metis*
hence *sorted-list-of-set s = sorted-list-of-set t*
using $c1(4) \ c2(4)$ **by** *(intro nth-equalityI) auto*
thus $s = t$
using $c1(1) \ c2(1)$ *sorted-list-of-set-inject* **by** *auto*
qed

lemma *osl-leD*:
assumes *finite S n ≤ card S*
assumes *e ∈ carrier (ordered-set-lattice S n)*
assumes *f ∈ carrier (ordered-set-lattice S n)*
shows $e \sqsubseteq_{\text{ordered-set-lattice } S \ n} f \longleftrightarrow (\forall i. \text{osl-repr } S \ n \ e \ i \leq \text{osl-repr } S \ n \ f \ i)$ **(is ?L = ?R)**
proof –
note $c1 = \text{osl-carr-sorted-list-of-set}[OF \ \text{assms}(1,2,3)]$
note $c2 = \text{osl-carr-sorted-list-of-set}[OF \ \text{assms}(1,2,4)]$

have $?L = \text{list-all2 } (\leq) \ (\text{sorted-list-of-set } e) \ (\text{sorted-list-of-set } f)$
unfolding *ordered-set-lattice-def le-ordered-set-lattice-def* **by** *simp*
also have $\dots = ?R$ **using** $c1(4) \ c2(4)$ **unfolding** *list-all2-conv-all-nth osl-repr-def*
by *simp*
finally show *?thesis* **by** *simp*
qed

lemma *ordered-set-lattice-partial-order*:
fixes $S :: ('a :: \text{linorder}) \ \text{set}$
assumes *finite S n ≤ card S*
shows *partial-order (ordered-set-lattice S n)*
proof –
let $?L = \text{ordered-set-lattice } S \ n$

note $\text{osl-list-repr-inj} = \text{osl-list-repr-inj}[OF \ \text{assms}]$
note $\text{osl-leD} = \text{osl-leD}[OF \ \text{assms}]$

have $\text{ref}:x \sqsubseteq_{?L} x$ **if** $x \in \text{carrier } ?L$ **for** x
using *osl-leD* **that** **by** *auto*

```

have antisym: $x = y$  if  $x \sqsubseteq_{?L} y$   $y \sqsubseteq_{?L} x$   $x \in \text{carrier } ?L$   $y \in \text{carrier } ?L$  for  $x$   $y$ 
using osl-leD osl-list-repr-inj that by (metis order-antisym)

have trans: $x \sqsubseteq_{?L} z$ 
if  $x \sqsubseteq_{?L} y$   $y \sqsubseteq_{?L} z$   $x \in \text{carrier } ?L$   $y \in \text{carrier } ?L$   $z \in \text{carrier } ?L$  for  $x$   $y$   $z$ 
using osl-leD that by (meson order-trans)

have eq-eq:  $(.=_{?L}) = (=)$  unfolding ordered-set-lattice-def by simp

show partial-order ?L
using ref antisym trans eq-eq by (unfold-locales presburger+)
qed

lemma map2-max-mono:
fixes  $xs :: ('a :: \text{linorder}) \text{ list}$ 
assumes  $\text{length } xs = \text{length } ys$ 
assumes sorted-wrt  $(<)$   $xs$  sorted-wrt  $(<)$   $ys$ 
shows sorted-wrt  $(<)$   $(\text{map2 } \text{max } xs \ ys)$ 
using assms
proof (induction xs ys rule:list-induct2)
case Nil
then show ?case by simp
next
case (Cons  $x$   $xs$   $y$   $ys$ )
have  $\text{max } x \ y < \text{max } a \ b$  if  $(a,b) \in \text{set } (\text{zip } xs \ ys)$  for  $a$   $b$ 
proof –
have  $x < a$  using set-zip-leftD[OF that] Cons(3) by auto
moreover have  $y < b$  using set-zip-rightD[OF that] Cons(4) by auto
ultimately show ?thesis by (auto intro: max.strict-coboundedI1 max.strict-coboundedI2)
qed
moreover have sorted-wrt  $(<)$   $(\text{map2 } \text{max } xs \ ys)$ 
using Cons(3,4) by (intro Cons(2)) auto
ultimately show ?case by auto
qed

lemma map2-min-mono:
fixes  $xs :: ('a :: \text{linorder}) \text{ list}$ 
assumes  $\text{length } xs = \text{length } ys$ 
assumes sorted-wrt  $(<)$   $xs$  sorted-wrt  $(<)$   $ys$ 
shows sorted-wrt  $(<)$   $(\text{map2 } \text{min } xs \ ys)$ 
using assms
proof (induction xs ys rule:list-induct2)
case Nil
then show ?case by simp
next
case (Cons  $x$   $xs$   $y$   $ys$ )
have  $\text{min } x \ y < \text{min } a \ b$  if  $(a,b) \in \text{set } (\text{zip } xs \ ys)$  for  $a$   $b$ 
proof –
have  $x < a$  using set-zip-leftD[OF that] Cons(3) by auto

```

moreover have $y < b$ **using** *set-zip-rightD*[*OF that*] *Cons*(4) **by** *auto*
ultimately show *?thesis* **by** (*auto intro: min.strict-coboundedI1 min.strict-coboundedI2*)
qed
moreover have *sorted-wrt* ($<$) (*map2 min xs ys*)
using *Cons*(3,4) **by** (*intro Cons*(2)) *auto*
ultimately show *?case* **by** *auto*
qed

lemma *ordered-set-lattice-carrier-finite-ne*:
assumes *finite S n ≤ card S*
shows *carrier (ordered-set-lattice S n) ≠ {} finite (carrier (ordered-set-lattice S n))*
proof –
let $?C = \text{carrier } (\text{ordered-set-lattice } S \ n)$

have $0 < (\text{card } S \ \text{choose } n)$ **by** (*intro zero-less-binomial assms*(2))
also have $\dots = \text{card } \{T. T \subseteq S \wedge \text{card } T = n\}$ **unfolding** *n-subsets*[*OF assms*(1)] **by** *simp*
also have $\dots = \text{card } \{T. T \subseteq S \wedge \text{finite } T \wedge \text{card } T = n\}$
using *assms*(1) *finite-subset* **by** (*intro arg-cong*[**where** $f = \text{card}$] *Collect-cong*)
auto
also have $\dots = \text{card } ?C$ **unfolding** *ordered-set-lattice-def* **by** *simp*
finally have $\text{card } ?C > 0$ **by** *simp*
thus $?C \neq \{\}$ *finite ?C* **unfolding** *card-gt-0-iff* **by** *auto*
qed

lemma *ordered-set-lattice-lattice*:
fixes $S :: ('a :: \text{linorder}) \ \text{set}$
assumes *finite S n ≤ card S*
shows *finite-ne-distrib-lattice (ordered-set-lattice S n)*
proof –
let $?L = \text{ordered-set-lattice } S \ n$

note *osl-leD = osl-leD*[*OF assms*]
note *osl-list-repr-inj = osl-list-repr-inj*[*OF assms*]

interpret *partial-order ?L* **by** (*intro ordered-set-lattice-partial-order assms*)

define *lmax* **where** $\text{lmax } x \ y = \text{set } (\text{map2 } \text{max } (\text{sorted-list-of-set } x) (\text{sorted-list-of-set } y))$
for $x \ y :: 'a \ \text{set}$

define *lmin* **where** $\text{lmin } x \ y = \text{set } (\text{map2 } \text{min } (\text{sorted-list-of-set } x) (\text{sorted-list-of-set } y))$
for $x \ y :: 'a \ \text{set}$

have *lmax-1*:
 $\text{osl-repr } S \ n \ (\text{lmax } s \ t) \ i = \text{max } (\text{osl-repr } S \ n \ s \ i) (\text{osl-repr } S \ n \ t \ i)$ (**is** $?L1 = ?R1$)

$lmax\ s\ t \in carrier\ ?L$
if $s \in carrier\ ?L\ t \in carrier\ ?L$ **for** $s\ t\ i$
proof –
note $s\text{-carr} = osl\text{-carr}\text{-sorted}\text{-list}\text{-of}\text{-set}[OF\ assms\ that(1)]$
note $t\text{-carr} = osl\text{-carr}\text{-sorted}\text{-list}\text{-of}\text{-set}[OF\ assms\ that(2)]$

have $s\text{:sorted}\text{-wrt}\ (<) (map2\ max\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t))$
using $s\text{-carr}\ t\text{-carr}$ **by** $(intro\ map2\text{-max}\text{-mono})\ auto$
hence $?L1 = (\lambda i \in \{..<n\}. (map2\ max\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t)))\ !\ i)\ i$
unfolding $lmax\text{-def}\ osl\text{-repr}\text{-def}\ strict\text{-sorted}\text{-iff}$
by $(subst\ linorder\text{-class}\text{-sorted}\text{-list}\text{-of}\text{-set}\text{-idem}\text{-if}\text{-sorted}\text{-distinct})\ auto$
also have $\dots = (\lambda i \in \{..<n\}. max\ (sorted\text{-list}\text{-of}\text{-set}\ s\ !\ i)\ (sorted\text{-list}\text{-of}\text{-set}\ t\ !\ i))\ i$
using $s\text{-carr}\ t\text{-carr}$ **by** $simp$
also have $\dots = ?R1$ **unfolding** $osl\text{-repr}\text{-def}$ **by** $auto$
finally show $?L1 = ?R1$ **by** $simp$

have $set\ (zip\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t)) \subseteq S \times S$
using $s\text{-carr}(3,5)\ t\text{-carr}(3,5)$ **by** $(auto\ intro\text{:set}\text{-zip}\text{-leftD}\ set\text{-zip}\text{-rightD})$
hence $set\ (map2\ max\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t)) \subseteq S$
by $(auto\ simp\text{:max}\text{-def})$
thus $lmax\ s\ t \in carrier\ ?L$
using $s\text{-carr}\ t\text{-carr}\ s$ **unfolding** $lmax\text{-def}\ strict\text{-sorted}\text{-iff}$
by $(intro\ ordered\text{-set}\text{-lattice}\text{-carrier}\text{-intro}[OF\ assms])\ auto$
qed

have $lmin\text{-1}$:
 $osl\text{-repr}\ S\ n\ (lmin\ s\ t)\ i = min\ (osl\text{-repr}\ S\ n\ s\ i)\ (osl\text{-repr}\ S\ n\ t\ i)$ **(is** $?L1 = ?R1)$
 $lmin\ s\ t \in carrier\ ?L$
if $s \in carrier\ ?L\ t \in carrier\ ?L$ **for** $s\ t\ i$
proof –
note $s\text{-carr} = osl\text{-carr}\text{-sorted}\text{-list}\text{-of}\text{-set}[OF\ assms\ that(1)]$
note $t\text{-carr} = osl\text{-carr}\text{-sorted}\text{-list}\text{-of}\text{-set}[OF\ assms\ that(2)]$

have $s\text{:sorted}\text{-wrt}\ (<) (map2\ min\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t))$
using $s\text{-carr}\ t\text{-carr}$ **by** $(intro\ map2\text{-min}\text{-mono})\ auto$
hence $?L1 = (\lambda i \in \{..<n\}. (map2\ min\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t)))\ !\ i)\ i$
unfolding $lmin\text{-def}\ osl\text{-repr}\text{-def}\ strict\text{-sorted}\text{-iff}$
by $(subst\ linorder\text{-class}\text{-sorted}\text{-list}\text{-of}\text{-set}\text{-idem}\text{-if}\text{-sorted}\text{-distinct})\ auto$
also have $\dots = (\lambda i \in \{..<n\}. min\ (sorted\text{-list}\text{-of}\text{-set}\ s\ !\ i)\ (sorted\text{-list}\text{-of}\text{-set}\ t\ !\ i))\ i$
using $s\text{-carr}\ t\text{-carr}$ **by** $simp$
also have $\dots = ?R1$ **unfolding** $osl\text{-repr}\text{-def}$ **by** $auto$
finally show $?L1 = ?R1$ **by** $simp$

have $set\ (zip\ (sorted\text{-list}\text{-of}\text{-set}\ s)\ (sorted\text{-list}\text{-of}\text{-set}\ t)) \subseteq S \times S$

```

    using s-carr(3,5) t-carr(3,5) by (auto intro:set-zip-leftD set-zip-rightD)
  hence set (map2 min (sorted-list-of-set s) (sorted-list-of-set t))  $\subseteq$  S
    by (auto simp:min-def)
  thus lmin s t  $\in$  carrier ?L
    using s-carr t-carr s unfolding lmin-def strict-sorted-iff
    by (intro ordered-set-lattice-carrier-intro[OF assms]) auto
qed

have lmax: is-lub ?L (lmax x y) {x,y} if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
  using that lmax-1 osl-leD by (intro least-UpperI) (auto simp:Upper-def)
  hence  $\exists$  s. is-lub ?L s {x, y} if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
    using that by auto
  hence 1: upper-semilattice ?L by (unfold-locales) auto

have lmin: is-glb ?L (lmin x y) {x,y} if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
  using that lmin-1 osl-leD by (intro greatest-LowerI) (auto simp:Lower-def)
  hence  $\exists$  s. is-glb ?L s {x, y} if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
    using that by auto
  hence 2: lower-semilattice ?L by (unfold-locales) auto

have 4:lattice ?L using 1 2 unfolding lattice-def by auto
interpret lattice ?L using 4 by simp

have join-eq: x  $\sqcap_{?L}$  y = lmin x y if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
  by (intro glb-unique[symmetric] that lmin)

have meet-eq: x  $\sqcup_{?L}$  y = lmax x y if x  $\in$  carrier ?L y  $\in$  carrier ?L for x y
  by (intro lub-unique[symmetric] that lmax)

have (x  $\sqcap_{?L}$  (y  $\sqcup_{?L}$  z)) = (x  $\sqcap_{?L}$  y)  $\sqcup_{?L}$  (x  $\sqcap_{?L}$  z)
  if x  $\in$  carrier ?L y  $\in$  carrier ?L z  $\in$  carrier ?L for x y z
  proof -
    have osl-repr S n (lmin x (lmax y z)) i = osl-repr S n (lmax (lmin x y) (lmin
x z)) i for i
      using lmax-1 that lmin-1 by (simp add:min-max-distrib2)
    hence lmin x (lmax y z) = lmax (lmin x y) (lmin x z)
      by (intro osl-list-repr-inj lmax-1 lmin-1 that allI)
    thus ?thesis using that by (simp add: meet-eq join-eq lmax-1 lmin-1)
  qed
  thus ?thesis using 4 ordered-set-lattice-carrier-finite-ne[OF assms(1,2)] by (unfold-locales)
auto
qed

lemma insort-eq:
  fixes xs :: ('a :: linorder) list
  assumes sorted xs
  shows  $\exists$  ys zs. insort e xs = ys@e#zs  $\wedge$  ys@zs=xs  $\wedge$  set ys  $\subseteq$  {..e}  $\wedge$  set zs  $\subseteq$ 
{e..}
  proof -

```

```

let ?ys = takeWhile ( $\lambda x. x < e$ ) xs
let ?zs = dropWhile ( $\lambda x. x < e$ ) xs

have a:insort e xs = ?ys@e#?zs by (induction xs) auto

have sorted (?ys@e#?zs) unfolding a[symmetric] using assms sorted-insort by
auto
hence sorted ([e]@?zs) by (simp add: sorted-append)
hence set ?zs  $\subseteq$  {e..} unfolding sorted-append by auto
moreover have set ?ys  $\subseteq$  {.. $e$ } by (metis lessThan-iff set-takeWhileD sub-
set-eq)
moreover have ?ys @ ?zs = xs by simp
ultimately show ?thesis using a by blast
qed

```

lemma list-all2-insort:

```

fixes xs ys :: ('a :: linorder) list
assumes length xs = length ys sorted xs sorted ys
shows list-all2 ( $\leq$ ) xs ys  $\longleftrightarrow$  list-all2 ( $\leq$ ) (insort e xs) (insort e ys)
proof -

```

obtain x1 x3 **where** xs:

```

  xs = x1@x3 insort e xs = x1@e#x3 set x1  $\subseteq$  {.. $e$ } set x3  $\subseteq$  {e..}
  using insort-eq[OF assms(2)] by blast

```

obtain y1 y3 **where** ys: ys = y1@y3

```

  insort e ys = y1@e#y3 set y1  $\subseteq$  {.. $e$ } set y3  $\subseteq$  {e..}
  using insort-eq[OF assms(3)] by blast

```

```

have l: length y1 + length y3 = length x1 + length x3 using assms(1) xs(1)
ys(1) by simp

```

```

have list-all2 ( $\leq$ ) xs ys  $\longleftrightarrow$  list-all2 ( $\leq$ ) (x1@x3) (y1@y3) by (simp add: xs
ys)

```

also **have** ... \longleftrightarrow list-all2 (\leq) (x1@e#x3) (y1@e#y3) (**is** ?L \longleftrightarrow ?R)

proof (cases length x1 < length y1)

case True

have length x3 > 0 **using** l True **by** linarith

hence (x1@x3) ! length x1 \geq e

using xs(4) nth-mem in-mono **unfolding** nth-append **by** fastforce

moreover **have** (y1@y3) ! length x1 < e

using True ys(3) nth-mem **unfolding** nth-append **by** auto

moreover **have** length x1 < length (x1@x3) **using** l True **by** auto

ultimately **have** 1:?L = False

unfolding xs ys list-all2-conv-all-nth **by** (meson leD order.trans)

have (y1@e#y3) ! length x1 < e

using True ys(3) nth-mem **unfolding** nth-append **by** auto

moreover **have** (x1@e#x3) ! length x1 = e **by** simp

moreover **have** length x1 < length (x1@e#x3) **using** l True **by** auto

```

ultimately have ?R = False
  unfolding xs(2) ys(2) list-all2-conv-all-nth by (metis leD)

thus ?thesis using 1 by auto
next
case False
let ?x1 = take (length y1) x1
define x2 where [simp]: x2 = drop (length y1) x1

define y2 where [simp]: y2 = take (length x1 - length y1) y3
let ?y3 = drop (length x1 - length y1) y3

have l2: length x2 = length y2 using False l by simp
have set-x2: set x2  $\subseteq$  {.. $e$ }
  unfolding x2-def using xs(3) set-drop-subset subset-trans by metis
have set-y2: set y2  $\subseteq$  {e..}
  unfolding y2-def using ys(4) set-take-subset subset-trans by metis

have set (x2@[ $e$ ])  $\subseteq$  {.. $e$ } set (e#y2)  $\subseteq$  {e..}
  using set-x2 set-y2 by auto
hence a':list-all2 ( $\lambda x y. x \leq e \wedge e \leq y$ ) (x2@[ $e$ ]) (e#y2)
  using l2 set-zip-leftD set-zip-rightD by (intro list-all2I conjI ballI case-prodI2)
fastforce+
have a:list-all2 ( $\leq$ ) (x2@[ $e$ ]) (e#y2) by (intro list-all2-mono[OF a']) auto

have b':list-all2 ( $\lambda x y. x \leq e \wedge e \leq y$ ) x2 y2
  using l2 set-x2 set-y2 set-zip-leftD set-zip-rightD by (intro list-all2I conjI
ballI case-prodI2) fastforce+
have b:list-all2 ( $\leq$ ) x2 y2 by (intro list-all2-mono[OF b']) auto

have ?L  $\longleftrightarrow$  list-all2 ( $\leq$ ) ((?x1 @ x2) @ x3) (y1 @ y2 @ ?y3) by simp
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) (?x1 @ x2 @ x3) (y1 @ y2 @ ?y3) using append-assoc
by metis
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) ?x1 y1  $\wedge$  list-all2 ( $\leq$ ) (x2 @ x3) (y2 @ ?y3)
  using False by (intro list-all2-append) auto
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) ?x1 y1  $\wedge$  (list-all2 ( $\leq$ ) x2 y2  $\wedge$  list-all2 ( $\leq$ )
x3 ?y3)
  using l False by (intro arg-cong2[where f=( $\wedge$ )] refl list-all2-append) simp
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) ?x1 y1  $\wedge$  (list-all2 ( $\leq$ ) (x2@[ $e$ ]) (e#y2)  $\wedge$  list-all2
( $\leq$ ) x3 ?y3)
  using a b by simp
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) ?x1 y1  $\wedge$  (list-all2 ( $\leq$ ) ((x2@[ $e$ ]) @ x3) ((e#y2) @ ?y3))
  using l False by (intro arg-cong2[where f=( $\wedge$ )] refl list-all2-append[symmetric])
simp
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) (?x1 @ ((x2@[ $e$ ]) @ x3)) (y1 @ ((e#y2) @ ?y3))
  using False by (intro list-all2-append[symmetric]) auto
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) ((?x1 @ x2) @ (e#x3)) (y1 @ e#(y2 @ ?y3))
  using append-assoc by (intro arg-cong2[where f=list-all2 ( $\leq$ )] simp-all)
also have ...  $\longleftrightarrow$  ?R by simp

```

```

    finally show ?thesis by simp
qed
also have ...  $\longleftrightarrow$  list-all2 ( $\leq$ ) (insort e xs) (insort e ys) using xs ys by simp
finally show ?thesis by simp
qed

lemma le-ordered-set-lattice-diff:
  fixes x y :: ('a :: linorder) set
  assumes finite x finite y card x = card y
  shows le-ordered-set-lattice x y  $\longleftrightarrow$  le-ordered-set-lattice (x - y) (y - x)
proof -
  let ?le = le-ordered-set-lattice
  define u v S where vars: u = x - y v = y - x S = x  $\cap$  y

  have fins: finite S finite u finite v unfolding vars using assms by auto

  have disj: S  $\cap$  u = {} S  $\cap$  v = {} unfolding vars by auto

  have cards: card u = card v unfolding vars using assms
    by (simp add: card-le-sym-Diff order-antisym)

  have ?le x y = ?le (u  $\cup$  S) (v  $\cup$  S) unfolding vars by (intro arg-cong2[where
f=?le]) auto
  also have ... = ?le u v using fins(1) disj
  proof (induction S rule:finite-induct)
    case empty thus ?case by simp
  next
    case (insert x F)
    define us where us = sorted-list-of-set (u  $\cup$  F)
    define vs where vs = sorted-list-of-set (v  $\cup$  F)
    have card (u  $\cup$  F) = card u + card F using insert fins by (intro card-Un-disjoint)
    auto
    also have ... = card v + card F using cards by auto
    also have ... = card (v  $\cup$  F) using insert fins by (intro card-Un-disjoint[symmetric])
    auto
    finally have cards': card (u  $\cup$  F) = card (v  $\cup$  F) by simp

    have ?le (u  $\cup$  insert x F) (v  $\cup$  insert x F) = ?le (insert x (u  $\cup$  F)) (insert x
(v  $\cup$  F))
    by simp
    also have ... = list-all2 ( $\leq$ ) (insort x us) (insort x vs)
    unfolding le-ordered-set-lattice-def us-def vs-def using insert fins(2,3)
    by (intro arg-cong2[where f=list-all2 ( $\leq$ )] sorted-list-of-set-insert ) auto
    also have ... = list-all2 ( $\leq$ ) us vs
    using cards' by (intro list-all2-insort[symmetric]) (simp-all add:us-def vs-def)
    also have ... = ?le (u  $\cup$  F) (v  $\cup$  F)
    unfolding le-ordered-set-lattice-def us-def vs-def by simp
    also have ... = ?le u v using insert by (intro insert) auto
    finally show ?case by simp

```

qed
 also have $\dots = ?le (x-y) (y-x)$ **unfolding vars by simp**
 finally show $?thesis$ **by simp**
 qed

lemma *ordered-set-lattice-carrier*:
 assumes $T \in carrier (ordered-set-lattice S n)$
 shows *finite T card T = n T \subseteq S*
 using *assms* **unfolding ordered-set-lattice-def by auto**

lemma *ordered-set-lattice-dual*:
 assumes *finite S n \leq card S*
 defines $L \equiv ordered-set-lattice S n$
 defines $M \equiv ordered-set-lattice S (card S - n)$
 shows
 $\wedge x. x \in carrier L \implies (S-x) \in carrier M$
 $\wedge x. x \in carrier M \implies (S-x) \in carrier L$
 $\wedge x y. x \in carrier L \wedge y \in carrier L \implies x \sqsubseteq_L y \longleftrightarrow (S-y) \sqsubseteq_M (S-x)$
proof (*goal-cases*)
 case (1 x)
 thus $?case$ **using** *assms(1,2)* **unfolding** *ordered-set-lattice-def M-def L-def*
 by (*auto intro: card-Diff-subset*)
 next
 case (2 x)
 thus $?case$ **using** *assms(1,2)* **unfolding** *ordered-set-lattice-def M-def L-def*
 by (*auto simp: card-Diff-subset-Int Int-absorb1*)
 next
 case (3 x y)
 hence *a: finite x finite y card x = card y x \subseteq S y \subseteq S*
unfolding *ordered-set-lattice-def M-def L-def by auto*

have *b: card (S - m) = card S - card m if m \subseteq S for m*
 using *that assms(1) card-Diff-subset finite-subset[OF - assms(1)] by auto*

have *le-ordered-set-lattice x y = le-ordered-set-lattice (x-y) (y-x)*
 by (*intro le-ordered-set-lattice-diff a*)
 also have $\dots = le-ordered-set-lattice ((S-y)-(S-x)) ((S-x)-(S-y))$
 using *a* by (*intro arg-cong2[where f=le-ordered-set-lattice] auto*)
 also have $\dots = le-ordered-set-lattice (S - y) (S - x)$
 using *a b assms(1)* by (*intro le-ordered-set-lattice-diff[symmetric] auto*)
 finally have *le-ordered-set-lattice x y = le-ordered-set-lattice (S - y) (S - x)*
 by *simp*
 thus $?case$ **unfolding** *ordered-set-lattice-def M-def L-def by simp*
 qed

lemma *bij-betw-ord-set-lattice-pairs*:
 assumes *finite S n \leq card S*
 defines $L \equiv ordered-set-lattice S n$
 assumes $x \in carrier L y \in carrier L x \sqsubseteq_L y$

shows $\exists \varphi. \text{bij-betw } \varphi \ x \ y \wedge \text{strict-mono-on } x \ \varphi \wedge (\forall e. \varphi \ e \geq e)$
proof –
let $?xs = \text{sorted-list-of-set } x$
let $?ys = \text{sorted-list-of-set } y$

let $?p1 = \text{the-inv-into } \{..<n\} \ (\lambda i. ?xs \ ! \ i)$
let $?p2 = (\lambda i. ?ys \ ! \ i)$

have $x: \text{card } x = n \ \text{finite } x$ **using** $\text{assms}(4)$ **unfolding** $L\text{-def ordered-set-lattice-def}$
by auto
have $y: \text{card } y = n \ \text{finite } y$ **using** $\text{assms}(5)$ **unfolding** $L\text{-def ordered-set-lattice-def}$
by auto
have $l\text{-xs}: \text{length } ?xs = n$ **using** $\text{length-sorted-list-of-set } x$ **by** simp
have $l\text{-ys}: \text{length } ?ys = n$ **using** $\text{length-sorted-list-of-set } y$ **by** simp

have $le: ?xs \ ! \ i \leq ?ys \ ! \ i$ **if** $i \in \{..<n\}$ **for** i
using $\text{assms}(6)$ $l\text{-xs } l\text{-ys}$ **that** **unfolding** $L\text{-def ordered-set-lattice-def } le\text{-ordered-set-lattice-def}$
by $(\text{auto } \text{simp } \text{add:list-all2-conv-all-nth})$

have $xs\text{-strict-mono}: \text{strict-mono-on } \{..<n\} \ ((!) \ ?xs)$
using $\text{strict-sorted-list-of-set}$
by $(\text{metis } l\text{-xs } \text{lessThan-iff sorted-wrt-iff-nth-less strict-mono-onI})$

hence $\text{inj-xs}: \text{inj-on } ((!) \ ?xs) \ \{..<n\}$ **using** $\text{strict-mono-on-imp-inj-on}$ **by** auto
have $\text{set } ?xs = x$ **using** $\text{set-sorted-list-of-set } x$ **by** simp
hence $\text{ran-xs}: ((!) \ ?xs) \ ' \ \{..<n\} = x$ **using** set-conv-nth **unfolding** $l\text{-xs}[\text{symmetric}]$
by fast

have $\text{set } ?ys = y$ **using** $\text{set-sorted-list-of-set } y$ **by** simp
hence $\text{ran-ys}: ((!) \ ?ys) \ ' \ \{..<n\} = y$ **using** set-conv-nth **unfolding** $l\text{-ys}[\text{symmetric}]$
by fast

have $p1\text{-strict-mono}: \text{strict-mono-on } x \ ?p1$
proof $(\text{rule } \text{strict-mono-onI})$
fix $r \ s$ **assume** $a: r \in x \ s \in x \ r < s$
have $?p1 \ r \in \{..<n\}$ **using** $a \ \text{ran-xs}$ **by** $(\text{intro } \text{the-inv-into-into}[OF \ \text{inj-xs}])$
 auto
moreover **have** $?p1 \ s \in \{..<n\}$ **using** $a \ \text{ran-xs}$ **by** $(\text{intro } \text{the-inv-into-into}[OF \ \text{inj-xs}])$ auto
moreover **have** $?xs \ ! \ (?p1 \ r) = r$ **using** $a \ \text{ran-xs}$ **by** $(\text{intro } f\text{-the-inv-into-f}[OF \ \text{inj-xs}])$ auto
moreover **have** $?xs \ ! \ (?p1 \ s) = s$ **using** $a \ \text{ran-xs}$ **by** $(\text{intro } f\text{-the-inv-into-f}[OF \ \text{inj-xs}])$ auto
ultimately show $?p1 \ r < ?p1 \ s$ **using** $a(3)$ $\text{strict-mono-on-leD}[OF \ xs\text{-strict-mono}]$
by fastforce
qed

have $\text{ran-p1}: ?p1 \ ' \ x = \{..<n\}$ **using** ran-xs $\text{the-inv-into-onto}[OF \ \text{inj-xs}]$ **by** simp

```

have p2-strict-mono: strict-mono-on {.. $n$ } ?p2
  using strict-sorted-list-of-set
  by (metis l-ys lessThan-iff sorted-wrt-iff-nth-less strict-mono-onI)

define  $\varphi$  where  $\varphi = (\lambda e. \text{if } e \in x \text{ then } (?p2 (?p1 e)) \text{ else } e)$ 

have strict-mono-on  $x$  (?p2  $\circ$  ?p1)
proof (rule strict-mono-onI)
  fix  $r s$  assume  $a: r \in x \ s \in x \ r < s$ 
  have ?p1  $r < ?p1 s$  using a strict-mono-onD[OF p1-strict-mono] by auto
  moreover have ?p1  $r \in \{.. $n\}$  ?p1  $s \in \{.. $n\}$  using a ran-p1 by auto
  ultimately show (?p2  $\circ$  ?p1)  $r < (?p2 \circ ?p1) s$ 
    using strict-mono-onD[OF p2-strict-mono] by simp
qed

  hence  $\varphi$ -strict-mono: strict-mono-on  $x$   $\varphi$  unfolding  $\varphi$ -def strict-mono-on-def
by simp
  hence  $\varphi$ -inj: inj-on  $\varphi$   $x$  using strict-mono-on-imp-inj-on by auto

have  $\varphi$  '  $x \subseteq y$  using ran-p1 ran-ys unfolding  $\varphi$ -def by auto
hence  $\varphi$  '  $x = y$  using card-image[OF  $\varphi$ -inj]  $x y$  by (intro card-seteq) auto
hence bij-betw  $\varphi$   $x y$  using  $\varphi$ -inj unfolding bij-betw-def by auto

moreover have  $\varphi$   $e \geq e$  for  $e$ 
proof (cases  $e \in x$ )
  case True
  have  $e = ?xs ! (?p1 e)$ 
    using True ran-xs by (intro f-the-inv-into-f[symmetric] inj-xs) auto
  also have ...  $\leq ?p2 (?p1 e)$  using ran-p1 True by (intro le) auto
  also have ... =  $\varphi e$  using True by (simp add: $\varphi$ -def)
  finally show ?thesis by simp
next
  case False
  then show ?thesis unfolding  $\varphi$ -def by simp
qed

ultimately show ?thesis using  $\varphi$ -strict-mono by auto
qed

definition bij-pmf  $I F = \text{pmf-of-set } \{f. \text{bij-betw } f \ I \ F \wedge f \in \text{extensional } I\}$ 

lemma card-bijections':
  assumes finite  $A$  finite  $B$  card  $A = \text{card } B$ 
  shows card  $\{f. \text{bij-betw } f \ A \ B \wedge f \in \text{extensional } A\} = \text{fact } (\text{card } A)$  (is ?L = ?R)
proof -
  have ?L = card  $\{f \in A \rightarrow_E B. \text{bij-betw } f \ A \ B\}$ 
    using bij-betw-imp-surj-on[where  $A=A$  and  $B=B$ ]
    by (intro arg-cong[where  $f=\text{card}$ ] Collect-cong) (auto simp:PiE-def Pi-def)$$ 
```

also have ... = fact (card A) using card-bijections[OF assms] assms(3) by simp
 finally show ?thesis by simp
 qed

lemma bij-betw-non-empty-finite:
 assumes finite I finite F card I = card F
 shows
 finite {f. bij-betw f I F ∧ f ∈ extensional I} (is ?T1)
 {f. bij-betw f I F ∧ f ∈ extensional I} ≠ {} (is ?T2)
 proof –
 have fact (card I) > (0::nat) using fact-gt-zero by simp
 thus ?T1 ?T2
 using card-bijections'[OF assms] card-gt-0-iff by force+
 qed

lemma bij-pmf:
 assumes finite I finite F card I = card F
 shows
 set-pmf (bij-pmf I F) = {f. bij-betw f I F ∧ f ∈ extensional I}
 finite (set-pmf (bij-pmf I F))
 using bij-betw-non-empty-finite[OF assms] unfolding bij-pmf-def by auto

lemma expectation-ge-eval-at-point:
 assumes $\bigwedge y. y \in \text{set-pmf } p \implies f y \geq (0::\text{real})$
 assumes integrable p f
 shows $\text{pmf } p x * f x \leq (\int x. f x \partial p)$ (is ?L ≤ ?R)
 proof –
 have ?L = $(\sum a \in \{x\}. f a * \text{of-bool}(a=x) * \text{pmf } p a)$ by simp
 also have ... = $(\int a. f a * \text{of-bool}(a=x) \partial p)$
 by (intro integral-measure-pmf-real[symmetric]) auto
 also have ... ≤ ?R
 using assms by (intro integral-mono-AE' AE-pmfI) auto
 finally show ?thesis by simp
 qed

lemma split-bij-pmf:
 assumes finite I finite F card I = card F J ⊆ I
 shows bij-pmf I F =
 do {
 S ← pmf-of-set {S. card S = card J ∧ S ⊆ F};
 φ ← bij-pmf J S;
 ψ ← bij-pmf (I-J) (F-S);
 return-pmf (merge J (I-J) (φ, ψ))
 } (is ?L = ?R)
 proof (rule pmf-eq-iff-le)
 fix x
 let ?p1 = pmf-of-set {S. card S = card J ∧ S ⊆ F}

let $?p2 = \text{bij-pmf } J$
let $?p3 = (\lambda S. \text{bij-pmf } (I-J) (F-S))$

have $f0: \text{finite } J$ **using** $\text{finite-subset } \text{assms}(1,4)$ **by** metis
have $f1: \text{finite } (I-J)$ **using** $\text{finite-subset } \text{assms}(1,4)$ **by** force

note $\text{pos1} = \text{pmf-of-set}[OF \text{bij-betw-non-empty-finite}(2,1)[OF \text{assms}(1-3)]]$

show $\text{pmf } (\text{bij-pmf } I F) x \leq \text{pmf } ?R x$
proof ($\text{cases } x \in \text{set-pmf } ?L$)
case True
hence $a: \text{bij-betw } x I F x \in \text{extensional } I$
using $\text{bij-pmf}[OF \text{assms}(1-3)]$ **by** auto

define T **where** $T = x \text{ ' } J$
define y **where** $y = \text{restrict } x J$
define z **where** $z = \text{restrict } x (I-J)$

have $x\text{-on-compl}: x \text{ ' } (I-J) = (F-T)$ **using** $a \text{ assms}(4)$ **unfolding** $T\text{-def}$
 bij-betw-def
by ($\text{subst } \text{inj-on-image-set-diff}[\text{where } C=I]$) auto

have $T-F: T \subseteq F$ **using** $\text{bij-betw-imp-surj-on}[OF a(1)] \text{ assms}(4)$ **unfolding**
 $T\text{-def}$ **by** auto

have $f2: \text{finite } T$ **using** $\text{assms}(2) T-F \text{ finite-subset}$ **by** auto
have $f3: \text{finite } (F - T)$ **using** $\text{assms}(2) T-F \text{ finite-subset}$ **by** auto
have $c1: \text{card } J = \text{card } T$
unfolding $T\text{-def}$ **using** $\text{assms}(4) \text{ inj-on-subset } \text{bij-betw-imp-inj-on}[OF a(1)]$
by ($\text{intro } \text{card-image}[\text{symmetric}]$) auto
have $c2: \text{card } (I-J) = \text{card } (F-T)$
unfolding $x\text{-on-compl}[\text{symmetric}]$ **using** $\text{inj-on-subset } \text{bij-betw-imp-inj-on}[OF$
 $a(1)]$
by ($\text{intro } \text{card-image}[\text{symmetric}]$) force

have $\text{restrict } x (J \cup (I - J)) = \text{restrict } x I$ **using** $\text{assms}(4)$ **by** force
also have $\dots = x$ **using** $a \text{ extensional-restrict}$ **by** auto
finally have $b: \text{restrict } x (J \cup (I - J)) = x$ **by** simp

have $y: y \in \text{extensional } J \text{ bij-betw } y J T$
using $\text{assms}(4) \text{ inj-on-subset } a \text{ y-def}$ **unfolding** $\text{bij-betw-def } T\text{-def}$ **by** auto

have $z \text{ ' } (I-J) = (F-T)$ **using** $x\text{-on-compl}$ **unfolding** $z\text{-def}$ **by** auto
hence $z: z \in \text{extensional } (I-J) \text{ bij-betw } z (I-J) (F-T)$
using $a \text{ z-def}$ **unfolding** $\text{bij-betw-def } T\text{-def}$ **by** ($\text{auto } \text{intro:inj-on-diff}$)

have $\text{pos-assms2}: \{S. \text{card } S = \text{card } J \wedge S \subseteq F\} \neq \{\}$ $\text{finite } \{S. \text{card } S = \text{card}$
 $J \wedge S \subseteq F\}$
using $T-F c1$ **by** ($\text{auto } \text{intro!}: \text{finite-subset}[OF - \text{iffD2}[OF \text{finite-Pow-iff}$

$assms(2)]])$

note $pos3 =$

$pmf\text{-of-set}[OF\ bij\text{-betw}\text{-non}\text{-empty}\text{-finite}(2,1)[OF\ f0\ f2\ c1]]$

$pmf\text{-of-set}[OF\ bij\text{-betw}\text{-non}\text{-empty}\text{-finite}(2,1)[OF\ f1\ f3\ c2]]$

have $fin\text{-pmf}1: finite\ (set\text{-pmf}\ ?p1)$ **using** $pos\text{-assms}2\ set\text{-pmf}\text{-of}\text{-set}$ **by** $simp$

note $[simp] = integrable\text{-measure}\text{-pmf}\text{-finite}[OF\ fin\text{-pmf}1,$ **where** $'b=real]$

have $fin\text{-pmf}2: finite\ (set\text{-pmf}\ (?p2\ T))$ **by** $(intro\ bij\text{-pmf}[OF\ f0\ f2\ c1])$

note $[simp] = integrable\text{-measure}\text{-pmf}\text{-finite}[OF\ fin\text{-pmf}2,$ **where** $'b=real]$

have $fin\text{-pmf}3: finite\ (set\text{-pmf}\ (?p3\ T))$ **by** $(intro\ bij\text{-pmf}[OF\ f1\ f3\ c2])$

note $[simp] = integrable\text{-measure}\text{-pmf}\text{-finite}[OF\ fin\text{-pmf}3,$ **where** $'b=real]$

have $pmf\ ?L\ x = 1 / real\ (card\ \{f.\ bij\text{-betw}\ f\ I\ F \wedge f \in extensional\ I\})$

using $a\ pos1\ unfolding\ bij\text{-pmf}\text{-def}$ **by** $simp$

also have $\dots = 1 / real\ (fact\ (card\ I))$ **using** $assms$ **by** $(simp\ add: card\text{-bijections}'$

also have $\dots = 1 / real\ (fact\ (card\ J) * fact\ (card\ I - card\ J) * (card\ I\ choose\ card\ J))$

using $assms(1,4)\ card\text{-mono}$ **by** $(subst\ binomial\text{-fact}\text{-lemma})\ auto$

also have $\dots = 1 / real\ ((card\ F\ choose\ card\ J) * fact\ (card\ J) * fact\ (card\ (I - J)))$

using $assms(3)\ card\text{-Diff}\text{-subset}[OF\ f0\ assms(4)]$ **by** $simp$

also have $\dots = 1 / real\ (card\ \{S.\ S \subseteq F \wedge card\ S = card\ J\} * card\ \{f.\ bij\text{-betw}\ f\ J\ T \wedge f \in extensional\ J\} *$

$card\ \{f.\ bij\text{-betw}\ f\ (I - J)\ (F - T) \wedge f \in extensional\ (I - J)\})$

using $f0\ f1\ f2\ f3\ assms(2)\ c1\ c2$ **by** $(simp\ add: card\text{-bijections}'\ n\text{-subsets})$

also have $\dots = pmf\ ?p1\ T * pmf\ (?p2\ T)\ y * pmf\ (?p3\ T)\ z$

using $y\ z\ c1\ T\text{-F}\ unfolding\ bij\text{-pmf}\text{-def}\ pos3\ pmf\text{-of}\text{-set}[OF\ pos\text{-assms}2]$

by $(simp\ add: conj\text{-commute})$

also have $\dots = pmf\ ?p1\ T * (pmf\ (?p2\ T)\ y * (pmf\ (?p3\ T)\ z * of\text{-bool}(merge\ J\ (I - J)\ (y, z) = x)))$

unfolding $y\text{-def}\ z\text{-def}\ merge\text{-restrict}\ merge\text{-x}\text{-x}\text{-eq}\text{-restrict}\ b$ **by** $simp$

also have $\dots \leq pmf\ ?p1\ T * (pmf\ (?p2\ T)\ y * (\int\ \psi.\ of\text{-bool}(merge\ J\ (I - J)\ (y, \psi) = x)\ \partial^{?p3}\ T))$

by $(intro\ mult\text{-left}\text{-mono}\ expectation\text{-ge}\text{-eval}\text{-at}\text{-point}\ integral\text{-nonneg}\text{-AE}\ AE\text{-pmf}I)$ $simp\text{-all}$

also have $\dots \leq pmf\ ?p1\ T * (\int\ \varphi.\ (\int\ \psi.\ of\text{-bool}(merge\ J\ (I - J)\ (\varphi, \psi) = x)\ \partial^{?p3}\ T))\ \partial^{?p2}\ T)$

by $(intro\ mult\text{-left}\text{-mono}\ expectation\text{-ge}\text{-eval}\text{-at}\text{-point}\ integral\text{-nonneg}\text{-AE}\ AE\text{-pmf}I)$ $simp\text{-all}$

also have $\dots \leq (\int\ S.\ (\int\ \varphi.\ (\int\ \psi.\ of\text{-bool}(merge\ J\ (I - J)\ (\varphi, \psi) = x)\ \partial^{?p3}\ S))\ \partial^{?p2}\ S)\ \partial^{?p1}$

by $(intro\ expectation\text{-ge}\text{-eval}\text{-at}\text{-point}\ integral\text{-nonneg}\text{-AE}\ AE\text{-pmf}I)\ simp\text{-all}$

also have $\dots = pmf\ ?R\ x\ unfolding\ pmf\text{-bind}$ **by** $(simp\ add: indicator\text{-def})$

finally show $?thesis$ **by** $simp$

next

case $False$

hence $\text{pmf } ?L \ x = 0$ **by** (*simp add: set-pmf-iff*)
 also have $\dots \leq \text{pmf } ?R \ x$ **by** *simp*
 finally show *?thesis* **by** *simp*
qed
qed

lemma *map-bij-pmf*:
 assumes *finite I finite F card I = card F inj-on φ F*
 shows $\text{map-pmf } (\lambda f. (\lambda x \in I. \varphi(f \ x))) \ (\text{bij-pmf } I \ F) = \text{bij-pmf } I \ (\varphi \ ' \ F)$
proof –
 let *?h = the-inv-into F φ*

have *h-bij: bij-betw ?h ($\varphi \ ' \ F$) F*
 using *assms(4)* **by** (*simp add: bij-betw-the-inv-into inj-on-imp-bij-betw*)

have *bij-betw ($\lambda f. (\lambda x \in I. \varphi(f \ x))$)*
{f. bij-betw f I F $\wedge f \in \text{extensional } I$ } *{f. bij-betw f I ($\varphi \ ' \ F$) $\wedge f \in \text{extensional } I$ }*

proof (*intro bij-betwI[where $g = (\lambda f. (\lambda x \in I. ?h(f \ x)))$], goal-cases*)
 case 1 **thus** *?case*
 using *bij-betw-trans[OF - inj-on-imp-bij-betw[OF assms(4)], where $A = I$]*
 by (*auto simp: comp-def*)

next
 case 2 **thus** *?case*
 using *bij-betw-trans[OF - h-bij, where $A = I$]* **by** (*auto simp: comp-def*)

next
 case (3 *x*)
 hence $x \in I \rightarrow F \ x \in \text{extensional } I$ **using** *bij-betw-imp-surj-on* **by** *auto*
 hence $(\lambda \omega \in I. ?h ((\lambda y \in I. \varphi (x \ y)) \ \omega)) \ \omega = x \ \omega$ **for** ω
 by (*auto intro!: the-inv-into-f-f[OF assms(4)] simp: restrict-def extensional-def*)
thus *?case* **by** *auto*

next
 case (4 *y*)
 hence $y \in I \rightarrow (\varphi \ ' \ F) \ y \in \text{extensional } I$ **using** *bij-betw-imp-surj-on* **by** *blast+*
 hence $(\lambda x \in I. \varphi ((\lambda x \in I. \text{the-inv-into } F \ \varphi (y \ x)) \ x)) \ \omega = y \ \omega$ **for** ω
 by (*auto intro!: f-the-inv-into-f[OF assms(4)] simp: restrict-def extensional-def*)
thus *?case* **by** *auto*

qed
thus *?thesis*
unfolding *bij-pmf-def* **by** (*intro map-pmf-of-set-bij-betw bij-betw-non-empty-finite assms*)
qed

lemma *pmf-of-multiset-eq-pmf-of-setI*:
 assumes $c > 0 \ x \neq \{\#\}$
 assumes $\bigwedge i. i \in y \implies \text{count } x \ i = c$
 assumes $\bigwedge i. i \in \# \ x \implies i \in y$
 shows $\text{pmf-of-multiset } x = \text{pmf-of-set } y$
proof (*rule pmf-eqI*)

fix i
have $a:\text{set-mset } x = y$ **using** $\text{assms}(1,3,4)$ count-eq-zero-iff **by force**
hence $y\text{-ne}: y \neq \{\}$ $\text{finite } y$ **using** $\text{assms}(2)$ **by auto**

have $\text{size } x = \text{sum } (\text{count } x) y$ **unfolding** $\text{size-multiset-overloaded-eq } a$ **by simp**
also have $\dots = \text{sum } (\lambda\cdot. c) y$ **by** $(\text{intro } \text{sum.cong refl } \text{assms}(3))$ **auto**
also have $\dots = c * \text{card } y$ **using** $y\text{-ne}$ **by simp**
finally have $c * \text{card } y = \text{size } x$ **by simp**
hence $\text{rel}: \text{real } (\text{size } x) / \text{real } c = \text{real } (\text{card } y)$
using $\text{assms}(1)$ **by** $(\text{simp add:field-simps flip:of-nat-mult})$

have $\text{pmf } (\text{pmf-of-multiset } x) i = \text{real } (\text{count } x i) / \text{real } (\text{size } x)$
using $\text{assms}(2)$ **by simp**
also have $\dots = \text{real } c * \text{of-bool}(i \in y) / \text{real } (\text{size } x)$
using assms **by** $(\text{auto simp:of-bool-def count-eq-zero-iff})$
also have $\dots = \text{of-bool}(i \in y) / \text{real } (\text{card } y)$
unfolding $\text{rel}[\text{symmetric}]$ **by simp**
also have $\dots = \text{pmf } (\text{pmf-of-set } y) i$
using $y\text{-ne}$ **by simp**
finally show $\text{pmf } (\text{pmf-of-multiset } x) i = \text{pmf } (\text{pmf-of-set } y) i$ **by simp**
qed

lemma card-multi-bij :
assumes $\text{finite } J$
assumes $I = \bigcup (A \text{ ' } J) \text{ disjoint-family-on } A J$
assumes $\bigwedge j. j \in J \implies \text{finite } (A j) \wedge \text{finite } (B j) \wedge \text{card } (A j) = \text{card } (B j)$
shows $\text{card } \{f. (\forall j \in J. \text{bij-betw } f (A j) (B j)) \wedge f \in \text{extensional } I\} = (\prod_{i \in J. \text{fact } (\text{card } (A i))})$
(is $\text{card } ?L = ?R$ **)**
proof –
define g **where** $g i = (\text{THE } j. j \in J \wedge i \in A j)$ **for** i
have $g: g i = j$ **if** $i \in A j$ $j \in J$ **for** $i j$ **unfolding** $g\text{-def}$
proof $(\text{rule } \text{the1-equality})$
show $\exists !j. j \in J \wedge i \in A j$
using $\text{assms}(3)$ **that** **unfolding** $\text{bex1-def disjoint-family-on-def}$ **by auto**
show $j \in J \wedge i \in A j$ **using** **that** **by auto**
qed

have $\text{bij-betw } (\lambda\varphi. (\lambda i \in I. \varphi (g i) i))$
 $(\text{PiE } J (\lambda j. \{f. \text{bij-betw } f (A j) (B j) \wedge f \in \text{extensional } (A j)\})) ?L$
proof $(\text{intro } \text{bij-betwI}[\text{where } g = \lambda x. \lambda i \in J. \text{restrict } x (A i)] \text{ Pi-I, goal-cases})$
case $(1 x)$
have $\text{bij-betw } (\lambda i \in I. x (g i) i) (A j) (B j)$ **if** $j \in J$ **for** j
proof –
have $\text{last:bij-betw } (x j) (A j) (B j)$ **using** **that 1** **by auto**
have $A j \subseteq I$ **using** **that** $\text{assms}(2)$ **by auto**
thus $?thesis$ **using** g **that** **by** $(\text{intro } \text{iffD2}[\text{OF } \text{bij-betw-cong last}])$ **auto**
qed

```

    thus ?case using 1 by auto
next
  case (2 x)
  thus ?case by (intro iffD2[OF restrict-PiE-iff] ballI) simp
next
  case (3 x)
  have restrict (λi∈I. x (g i) i) (A j) = x j if j ∈ J for j
  proof -
    have A j ⊆ I using that assms(2) by auto
    moreover have x j ∈ extensional (A j) using that 3 by auto
    hence restrict (λi. x (g i) i) (A j) = x j
      using g that unfolding restrict-def extensional-def by auto
    ultimately show ?thesis unfolding restrict-restrict using Int-absorb1 by
metis
  qed
  thus ?case using 3 unfolding extensional-def PiE-def by auto
next
  case (4 y)
  have (λj∈J. restrict y (A j)) (g i) i = y i if that':i ∈ I for i
  proof -
    obtain j where i ∈ A j j ∈ J using that' assms(2) by auto
    thus ?thesis using g by simp
  qed
  thus ?case using 4 unfolding extensional-def by auto
qed

  hence card ?L = card (PiE J (λj. {f. bij-betw f (A j) (B j) ∧ f ∈ extensional (A
j)}))
  using bij-betw-same-card[symmetric] by auto
  also have ... = (∏ i∈J. card {f. bij-betw f (A i) (B i) ∧ f ∈ extensional (A
i)})
  unfolding card-PiE[OF assms(1)] by simp
  also have ... = (∏ i∈J. fact (card (A i)))
  using assms(4) by (intro prod.cong card-bijections') auto
  finally show ?thesis by simp
qed

lemma map-bij-pmf-non-inj:
  fixes I :: 'a set
  fixes F :: 'b set
  fixes φ :: 'b ⇒ 'c
  assumes finite I finite F card I = card F
  defines q ≡ {f. f ∈ extensional I ∧ {#f x. x ∈# mset-set I#} = {#φ x. x ∈#
mset-set F#}}
  shows map-pmf (λf. (λx∈I. φ(f x))) (bij-pmf I F) = pmf-of-set q (is ?L = -)
proof -
  let ?G = {# φ x. x ∈# mset-set F #}
  let ?G' = set-mset ?G
  define c :: nat where c = (∏ i ∈ set-mset ?G. fact (count ?G i))

```

note $ne = \text{bij-betw-non-empty-finite}[OF \text{ assms}(1-3)]$
note $cim = \text{count-image-mset-eq-card-vimage}$

have $c \geq 1$ **unfolding** $c\text{-def}$ **by** (intro prod-ge-1) **auto**
hence $c\text{-gt-0}: c > 0$ **by** simp

have $?L = \text{pmf-of-multiset } \{\#\lambda x \in I. \varphi(f x). f \in \# \text{ mset-set } \{f. \text{bij-betw } f I F \wedge f \in \text{extensional } I\} \# \}$
unfolding bij-pmf-def **by** $(\text{intro map-pmf-of-set}[OF ne])$
also have $\dots = \text{pmf-of-set } q$ **unfolding** $q\text{-def}$
proof $(\text{rule pmf-of-multiset-eq-pmf-of-setI}[OF c\text{-gt-0}], \text{goal-cases})$
case 1
have $\text{card } \{f. \text{bij-betw } f I F \wedge f \in \text{extensional } I\} > 0$ **using** ne **by** fastforce
thus $?case$ **by** $(\text{simp add:nonempty-has-size})$
next
case $(2 f)$

hence $a: \text{image-mset } f (\text{mset-set } I) = \text{image-mset } \varphi (\text{mset-set } F)$ **by** simp
hence $\text{card } \{x \in F. \varphi x = g\} = \text{card } \{x \in I. f x = g\}$ **for** g
using $cim[OF \text{ assms}(1)] cim[OF \text{ assms}(2)]$ **by** metis
hence $b: \text{card } (\varphi -' \{g\} \cap F) = \text{card } (f -' \{g\} \cap I)$ **for** g
by $(\text{auto simp add:Int-def conj-commute})$

have $c: \text{bij-betw } \omega I F \wedge (\lambda i \in I. \varphi(\omega i)) = f \iff (\forall g \in ?G'. \text{bij-betw } \omega (f -' \{g\} \cap I) (\varphi -' \{g\} \cap F))$
(is $?L1 = ?R1$) **for** ω
proof
assume $?L1$
hence $d: \text{bij-betw } \omega I F$ **and** $e: \forall i \in I. \varphi(\omega i) = f i$ **by** auto
have $\text{bij-betw } \omega (f -' \{g\} \cap I) (\varphi -' \{g\} \cap F)$ **if** $g \in ?G'$ **for** g
proof $-$
have $\text{card } (\varphi -' \{g\} \cap F) = \text{card } (\omega -' (f -' \{g\} \cap I))$
unfolding b **using** d
by $(\text{intro card-image[symmetric]}) (\text{simp add: bij-betw-imp-inj-on inj-on-Int})$
hence $\omega -' (f -' \{g\} \cap I) = \varphi -' \{g\} \cap F$
using $\text{assms}(2) e \text{ bij-betw-imp-surj-on}[OF d]$ **by** $(\text{intro card-seteq image-subsetI}) \text{auto}$
thus $?thesis$ **by** $(\text{intro bij-betw-subset}[OF d]) \text{auto}$
qed
thus $?R1$ **by** auto
next
assume $f: ?R1$

have $g: \varphi(\omega i) = f i$ **if** $i \in I$ **for** i
proof $-$
have $f i \in ?G'$ **unfolding** $a[\text{symmetric}]$ **using** $\text{that assms}(1)$ **by** auto
hence $\omega -' (f -' \{f i\} \cap I) = (\varphi -' \{f i\} \cap F)$
using $\text{bij-betw-imp-surj-on}$ **using** f **by** metis

```

    thus ?thesis using that by (auto simp add:vimage-def)
  qed
  have  $x = y$  if  $x \in I$   $y \in I$   $\omega x = \omega y$  for  $x y$ 
  proof -
    have  $f x \in ?G'$  unfolding a[symmetric] using that assms(1) by auto
    hence  $\text{inj-on } \omega (f - \{f x\} \cap I)$  using f bij-betw-imp-inj-on by blast
    moreover have  $f x = f y$  using that g by metis
    ultimately show  $x = y$  using that(1,2,3) inj-onD[where  $f=\omega$ ,  $OF$  -
that(3)] by fastforce
  qed
  hence  $h:\text{inj-on } \omega I$  by (rule inj-onI)

  have  $i: \omega^{-1} I \subseteq F$ 
  proof (rule image-subsetI)
    fix x assume  $x \in I$ 
    hence  $f x \in ?G'$   $x \in (f - \{f x\} \cap I)$  using assms(1) unfolding a[symmetric]
  by auto
    thus  $\omega x \in F$  using bij-betw-imp-surj-on f by fast
  qed
  have  $\text{bij-betw } \omega I F$ 
    using card-image[OF h] assms(3) unfolding bij-betw-def
    by (intro conjI card-seteq i h assms) auto
  thus ?L1 using g 2 unfolding restrict-def extensional-def by auto
  qed

  have  $j: f^{-1} I \subseteq \varphi^{-1} F$  using a
    by (metis assms(1,2) finite-set-mset-mset-set multiset.set-map set-eq-subset)

  have  $c = (\prod g \in ?G'. \text{fact } (\text{card } (f - \{g\} \cap I)))$ 
    unfolding b[symmetric] c-def cim[OF assms(2)]
    by (simp add:vimage-def Int-def conj-commute)
  also have  $\dots = \text{card } \{\omega. (\forall g \in ?G'. \text{bij-betw } \omega (f - \{g\} \cap I) (\varphi - \{g\} \cap F))$ 
 $\wedge \omega \in \text{extensional } I\}$ 
    using assms(1,2) j b
    by (intro card-multi-bij[symmetric]) (auto simp: vimage-def disjoint-family-on-def)
  also have  $\dots = \text{card } \{\omega. \text{bij-betw } \omega I F \wedge \omega \in \text{extensional } I \wedge (\lambda i \in I. \varphi (\omega$ 
 $i)) = f\}$ 
    using c by (intro arg-cong[where  $f=\text{card}$ ] Collect-cong) auto
  finally show ?case using ne by (subst count-image-mset-eq-card-vimage) auto
  next
  case (3 f)
  then obtain u where u-def:bij-betw u I F u  $\in \text{extensional } I$   $f = (\lambda x. \lambda xa \in I.$ 
 $\varphi (x xa)) u$ 
    using ne by auto

  have  $\text{image-mset } f (mset-set I) = \text{image-mset } \varphi (\text{image-mset } u (mset-set I))$ 
    using assms(1) unfolding u-def(3) multiset.map-comp by (intro image-mset-cong)
  auto
  also have  $\dots = \text{image-mset } \varphi (mset-set F)$  using image-mset-mset-set u-def(1)

```

unfolding *bij-betw-def* **by** (*intro arg-cong2*[**where** *f=image-mset*] *refl*) *auto*
finally have *image-mset f (mset-set I) = image-mset φ (mset-set F)* **by** *simp*

moreover have $f \in \text{extensional } I$ **unfolding** *u-def(3)* **by** *auto*
ultimately show *?case* **by** *simp*

qed
finally show *?thesis* **by** *simp*
qed

lemmas *fkf-inequality-pmf-internalized = fkf-inequality-pmf*[*unoverload-type 'a*]

lemma *permutation-distributions-are-neg-associated:*
fixes $F :: ('a :: \text{linorder-topology}) \text{ set}$
fixes $I :: 'b \text{ set}$
assumes *finite F finite I card I = card F*
shows *measure-pmf.neg-assoc (bij-pmf I F) ($\lambda i \omega. \omega i$) I*
proof (*rule measure-pmf.neg-assocI2, goal-cases*)
case (*1 i*) **thus** *?case* **by** *simp*
next
case (*2 f g J*)

have *fin-J: finite J* **using** *2(1) assms(2) finite-subset* **by** *metis*
have *fin-I-J: finite (I-J)* **using** *2(1) assms(2) finite-subset* **by** *blast*

define k **where** $k = \text{card } J$

have *k-le-F: $k \leq \text{card } F$* **unfolding** *k-def* **using** *2(1) assms(2,3) card-mono* **by**
force

let $?p0 = \text{bij-pmf } I \ F$
let $?p1 = \text{pmf-of-set } \{S. \text{card } S = \text{card } J \wedge S \subseteq F\}$
let $?p2 = \lambda S. \text{bij-pmf } J \ S$
let $?p3 = \lambda S. \text{bij-pmf } (I - J) \ (F - S)$

note *set-pmf-p0 = bij-pmf*[*OF assms(2,1,3)*]

note *integrable-p0*[*simp*] = *integrable-measure-pmf-finite*[*OF set-pmf-p0(2)*], **where**
'b=real]

note *dep-f = 2(2)*
note *dep-g = 2(3)*

have *bounded-f: bounded (f ' S)* **for** S **using** *bounded-subset*[*OF 2(6) image-mono*]
by *simp*
have *bounded-g: bounded (g ' S)* **for** S **using** *bounded-subset*[*OF 2(7) im-*
age-mono] **by** *simp*

note *mono-f = 2(4)*
note *mono-g = 2(5)*

let $?L = \text{ordered-set-lattice } F \ k$

define f' **where** $f' \ S = (\int \varphi. f \ \varphi \ \partial^?p2 \ S)$ **for** S
define g' **where** $g' \ S = (\int \varphi. g \ \varphi \ \partial^?p3 \ S)$ **for** S

interpret L : *finite-ne-distrib-lattice ordered-set-lattice* $F \ k$
by (*intro ordered-set-lattice-lattice assms(1) k-le-F*)

have carr-L-ne : *carrier* $?L \neq \{\}$ **and** fin-L : *finite* (*carrier* $?L$)
using *ordered-set-lattice-carrier-finite-ne*[*OF assms(1) k-le-F*] **by** *auto*

have mono-f' : *monotone-on* (*carrier* $?L$) ($\sqsubseteq_{?L}$) (\leq) f'
proof (*rule monotone-onI*)
fix $S \ T$
assume $a:S \sqsubseteq_{?L} T \ S \in \text{carrier } ?L \ T \in \text{carrier } ?L$
then obtain ϱ **where** $\varrho\text{-bij}$: *bij-betw* $\varrho \ S \ T$ **and** $\varrho\text{-inc}$: $\bigwedge e. \varrho \ e \geq e$
using *bij-betw-ord-set-lattice-pairs*[*OF assms(1) k-le-F*] **by** *blast*

note $S\text{-carr} = \text{ordered-set-lattice-carrier}[OF \ a(2)]$
have $c:\text{card } J = \text{card } S$ **using** $S\text{-carr}$ $k\text{-def}$ **by** *auto*

note $\text{set-pmf-p2} = \text{bij-pmf}[OF \ \text{fin-J } S\text{-carr}(1) \ c]$
note $\text{int} = \text{integrable-measure-pmf-finite}[OF \ \text{set-pmf-p2}(2)]$

have $f' \ S = (\int \varphi. f \ (\lambda\omega \in J. \varphi \ \omega) \ \partial^?p2 \ S)$ **unfolding** $f'\text{-def}$
using set-pmf-p2 *extensional-restrict* **by** (*intro integral-cong-AE AE-pmfI*)

force+
also have $\dots \leq (\int \varphi. f \ (\lambda\omega \in J. \varrho(\varphi \ \omega)) \ \partial^?p2 \ S)$ **unfolding** $f'\text{-def}$
using $\varrho\text{-inc}$ **unfolding** *restrict-def*
by (*intro integral-mono-AE AE-pmfI monoD*[*OF mono-f*] *int*) (*auto simp*:
 $le\text{-fun-def}$)

also have $\dots = (\int \varphi. f \ \varphi \ \partial(\text{map-pmf} \ (\lambda\varphi. (\lambda\omega \in J. \varrho(\varphi \ \omega))) \ (\partial^?p2 \ S)))$ **by** *simp*
also have $\dots = (\int \varphi. f \ \varphi \ \partial(\partial^?p2 \ (\varrho \ ' S)))$
using *ordered-set-lattice-carrier*[*OF a(2)*] $k\text{-def}$
by (*intro arg-cong2*[**where** $f = \text{measure-pmf.expectation}$] *map-bij-pmf refl*
 $\text{bij-betw-imp-inj-on}$ [*OF* $\varrho\text{-bij}$] *fin-J*) *auto*

also have $\dots = (\int \varphi. f \ \varphi \ \partial^?p2 \ T)$ **using** $\text{bij-betw-imp-surj-on}$ [*OF* $\varrho\text{-bij}$] **by**
 simp

finally show $f' \ S \leq f' \ T$ **unfolding** $f'\text{-def}$ **by** simp
qed

have mono-g' : *monotone-on* (*carrier* $?L$) ($\sqsubseteq_{?L}$) (\leq) ($(*)(-1) \circ g'$)
proof (*rule monotone-onI*)
fix $S \ T$
let $?M = \text{ordered-set-lattice } F \ (\text{card } F - k)$
assume $a:S \sqsubseteq_{?L} T \ S \in \text{carrier } ?L \ T \in \text{carrier } ?L$
hence $a': (F - T) \sqsubseteq_{?M} (F - S) \ (F - S) \in \text{carrier } ?M \ (F - T) \in \text{carrier } ?M$
using *ordered-set-lattice-dual*[*OF assms(1) k-le-F*] **by** *auto*

then obtain ϱ **where** ϱ -bij: bij-betw ϱ $(F-T)$ $(F-S)$ **and** ϱ -inc: $\bigwedge e. \varrho e \geq e$
using bij-betw-ord-set-lattice-pairs[OF assms(1)] k-le-F **by** (meson diff-le-self)
note T -carr = ordered-set-lattice-carrier[OF a'(3)]

have c : card $(I-J) = \text{card } (F-T)$
using assms ordered-set-lattice-carrier[OF a(3)] k-def 2(1) fin-J
by (simp add: card-Diff-subset)
note set-pmf-p3 = bij-pmf[OF fin-I-J T-carr(1) c]
note int = integrable-measure-pmf-finite[OF set-pmf-p3(2)]

have $g' T = (\int \varphi. g (\lambda\omega \in I-J. \varphi \omega) \partial?p3 T)$ **unfolding** g' -def
using set-pmf-p3 extensional-restrict **by** (intro integral-cong-AE AE-pmfI)

force+
also have $\dots \leq (\int \varphi. g (\lambda\omega \in I-J. \varrho(\varphi \omega)) \partial?p3 T)$ **unfolding** g' -def re-
strict-def **using** ϱ -inc
by (intro integral-mono-AE AE-pmfI monoD[OF mono-g] int) (auto simp:
le-fun-def)

also have $\dots = (\int \varphi. g \varphi \partial(\text{map-pmf } (\lambda\varphi. (\lambda\omega \in I-J. \varrho(\varphi \omega))) (?p3 T)))$ **by**
simp

also have $\dots = (\int \varphi. g \varphi \partial(\text{bij-pmf } (I - J) (\varrho '(F-T))))$ **using** assms
by (intro arg-cong2[**where** $f = \text{measure-pmf.expectation}$] map-bij-pmf refl
bij-betw-imp-inj-on[OF ϱ -bij] fin-J c) auto

also have $\dots = (\int \varphi. g \varphi \partial?p3 S)$ **using** bij-betw-imp-surj-on[OF ϱ -bij] **by**
simp

finally have $g' T \leq g' S$ **unfolding** g' -def **by** simp
thus $((*) (-1) \circ g') S \leq ((*) (-1) \circ g') T$ **by** simp
qed

have $(\int S. f' S * g' S \partial?p1) \leq (\int S. f' S \partial?p1) * (\int S. g' S \partial?p1)$
if td : $\exists (Rep :: 'x \Rightarrow 'a \text{ set}) \text{ Abs. type-definition } Rep \text{ Abs } (\text{carrier } ?L)$
proof –

obtain $Rep :: 'x \Rightarrow 'a \text{ set}$ **and** Abs **where** td :type-definition $Rep \text{ Abs } (\text{carrier } ?L)$
using td **by** auto
interpret type-definition $Rep \text{ Abs } \text{carrier } ?L$ **using** td **by** auto

have $\text{carr-L: carrier } ?L = \{S. \text{card } S = \text{card } J \wedge S \subseteq F\}$
using finite-subset[OF - assms(1)] **unfolding** ordered-set-lattice-def k-def
by (auto simp add:set-eq-iff)

have Rep -bij: bij-betw $Rep \text{ UNIV } \{S. \text{card } S = \text{card } J \wedge S \subseteq F\}$
using Rep-range Rep-inject carr-L **unfolding** bij-betw-def **by** (intro conjI
inj-onI) auto

have fin -UNIV: finite $(\text{UNIV} :: 'x \text{ set})$
using fin-L carr-L Rep-bij bij-betw-finite **by** metis

let $?p1' = \text{pmf-of-set } (\text{UNIV} :: 'x \text{ set})$
have rep -p1: $?p1 = \text{map-pmf } Rep ?p1'$

by (intro UNIV-not-empty map-pmf-of-set-bij-betw[symmetric] Rep-bij fin-UNIV)

note * = L.transfer-to-type[OF fin-L td]

note fkg = fkg-inequality-pmf-internalized[OF *]

have mono-rep-f': monotone ($\lambda S T. \text{Rep } S \sqsubseteq_{\text{?}L} \text{Rep } T$) (\leq) ($f' \circ \text{Rep}$)
 using mono-f' Rep unfolding monotone-on-def by simp

have mono-rep-g': monotone ($\lambda S T. \text{Rep } S \sqsubseteq_{\text{?}L} \text{Rep } T$) (\geq) ($g' \circ \text{Rep}$)
 using mono-g' Rep unfolding monotone-on-def by simp

have pmf-const: pmf ?p1' x = 1/(real (CARD('x))) for x
 by (subst pmf-of-set[OF - fin-UNIV]) auto

have ($\int S. f' S * g' S \partial^?p1$) = ($\int S. f' (\text{Rep } S) * g' (\text{Rep } S) \partial^?p1$)
 unfolding rep-p1 by simp

also have ... \leq ($\int S. f' (\text{Rep } S) \partial^?p1$) * ($\int S. g' (\text{Rep } S) \partial^?p1$)
 using mono-rep-f' mono-rep-g'
 by (intro fkg[where $\tau = \text{Fwd}$ and $\sigma = \text{Rev}$, simplified]) (simp-all add:comp-def pmf-const)

also have ... = ($\int S. f' S \partial^?p1$) * ($\int S. g' S \partial^?p1$)
 unfolding rep-p1 by simp

finally show ($\int S. f' S * g' S \partial^?p1$) \leq ($\int S. f' S \partial^?p1$) * ($\int S. g' S \partial^?p1$)
 by simp

qed

note core-result = this[cancel-type-definition, OF carr-L-ne]

note split-p0 = split-bij-pmf[OF assms(2,1,3) 2(1)]

have ($\int x. f x * g x \partial \text{bij-pmf } I F$) =
 ($\int S. (\int \varphi. (\int \psi. f (\text{merge } J (I-J) (\varphi, \psi)) * g (\text{merge } J (I-J) (\varphi, \psi)) \partial^?p3 S) \partial^?p2 S) \partial^?p1$)
 unfolding k-def by (simp add:split-p0 bounded-intros bounded-f bounded-g integral-bind-pmf)

also have ... = ($\int S. (\int \varphi. (\int \psi. f \varphi * g \psi \partial^?p3 S) \partial^?p2 S) \partial^?p1$)
 by (intro integral-cong-AE AE-pmfI arg-cong2[where $f = (*)$] depends-onD2[OF dep-f] depends-onD2[OF dep-g]) simp-all

also have ... = ($\int S. (\int \varphi. f \varphi \partial^?p2 S) * (\int \psi. g \psi \partial^?p3 S) \partial^?p1$) by simp

also have ... \leq ($\int S. (\int \varphi. f \varphi \partial^?p2 S) \partial^?p1$) * ($\int S. (\int \varphi. g \varphi \partial^?p3 S) \partial^?p1$)
 using core-result unfolding f'-def g'-def by simp

also have ... = ($\int S. (\int \varphi. (\int \psi. f \varphi \partial^?p3 S) \partial^?p2 S) \partial^?p1$) * ($\int S. (\int \varphi. (\int \psi. g \psi \partial^?p3 S) \partial^?p2 S) \partial^?p1$)
 by simp

also have ... =
 ($\int S. (\int \varphi. (\int \psi. f (\text{merge } J (I-J) (\varphi, \psi)) \partial^?p3 S) \partial^?p2 S) \partial^?p1$) *
 ($\int S. (\int \varphi. (\int \psi. g (\text{merge } J (I-J) (\varphi, \psi)) \partial^?p3 S) \partial^?p2 S) \partial^?p1$)
 by (intro arg-cong2[where $f = (*)$] integral-cong-AE AE-pmfI depends-onD2[OF dep-f])

*depends-on*D2[*OF dep-g*] *simp-all*
also have ... = $(\int x. f x \partial^?p0) * (\int x. g x \partial^?p0)$
unfolding *k-def* **by** (*simp add:split-p0 bounded-intros bounded-f bounded-g*
integral-bind-pmf)
finally show $(\int x. f x * g x \partial^?p0) \leq (\int x. f x \partial^?p0) * (\int x. g x \partial^?p0)$ **by** *simp*
qed

lemma *multiset-permutation-distributions-are-neg-associated:*

fixes *F* :: ('a :: *linorder-topology*) *multiset*

fixes *I* :: 'b *set*

assumes *finite I card I = size F*

defines *p* \equiv *pmf-of-set* { $\varphi. \varphi \in$ *extensional I* \wedge *image-mset* φ (*mset-set I*) = *F*}

shows *measure-pmf.neg-assoc p* ($\lambda i \omega. \omega i$) *I*

proof –

let *?xs* = *sorted-list-of-multiset F*

define α **where** $\alpha k = ?xs ! (\min k (\text{length } ?xs - 1))$ **for** *k*

let *?N* = {..*size F*}

let *?h* = ($\lambda f. (\lambda i \in I. \alpha (f i))$)

have *sorted-xs: sorted ?xs* **by** (*induction F, auto simp:sorted-insort*)

have *mono- α : mono α*

proof (*cases ?xs = []*)

case *True* **thus** *?thesis* **unfolding** α -*def* **by** *simp*

next

case *False* **thus** *?thesis* **unfolding** α -*def*

by (*intro monoI sorted-nth-mono[OF sorted-xs]*) (*simp-all add: min.strict-coboundedI2*)

qed

have *l-xs: length ?xs = size F* **by** (*metis mset-sorted-list-of-multiset size-mset*)

have *image-mset α (mset-set {..*size F*}) = image-mset (!) ?xs (mset-set {..*size F*})*

unfolding α -*def* *l-xs[symmetric]* **by** (*intro image-mset-cong*) *auto*

also have ... = *mset ?xs* **unfolding** *l-xs[symmetric]*

by (*metis map-nth mset-map mset-set-upto-eq-mset-upto*)

also have ... = *F* **by** *simp*

finally have *0:image-mset α (mset-set {..*size F*}) = F* **by** *simp*

have *map-pmf* ($\lambda f. (\lambda i \in I. \alpha (f i))$) (*bij-pmf I ?N*) =

pmf-of-set { $f \in$ *extensional I. image-mset* f (*mset-set I*) = *image-mset α*
(*mset-set {..*size F*})*}

using *assms* **by** (*intro map-bij-pmf-non-inj*) *auto*

also have ... = *p* **unfolding** *p-def 0* **by** *simp*

finally have *1:map-pmf* ($\lambda f. (\lambda i \in I. \alpha (f i))$) (*bij-pmf I ?N*) = *p* **by** *simp*

have *2:measure-pmf.neg-assoc* (*bij-pmf I {..*size F*})* ($\lambda i \omega. \omega i$) *I*

using *assms(1,2)* **by** (*intro permutation-distributions-are-neg-associated*) *auto*

have *measure-pmf.neg-assoc* (*bij-pmf* $I \{..<size\ F\}$) ($\lambda i\ \omega.$ *if* $i \in I$ *then* $\alpha(\omega\ i)$ *else* *undefined*) I
using *mono- α* **by** (*intro* *measure-pmf.neg-assoc.compose-simple*[*OF* *assms*(1) 2, **where** $\eta=Fwd$]
borel-measurable-continuous-onI) *simp-all*
hence *measure-pmf.neg-assoc* (*map-pmf* ($\lambda f. (\lambda i \in I. \alpha(f\ i))$) (*bij-pmf* $I \{..<size\ F\}$)) ($\lambda i\ \omega. \omega\ i$) I
by (*simp* *add:neg-assoc-map-pmf* *restrict-def* *if-distrib* *if-distribR*)
thus *?thesis* **unfolding** 1 **by** *simp*
qed

lemma *n-subsets-prob*:

assumes $d \leq card\ S$ *finite* $S\ s \in S$

shows

measure-pmf.prob (*pmf-of-set* $\{a. a \subseteq S \wedge card\ a = d\}$) $\{\omega. s \notin \omega\} = (1 - real\ d/card\ S)$

measure-pmf.prob (*pmf-of-set* $\{a. a \subseteq S \wedge card\ a = d\}$) $\{\omega. s \in \omega\} = real\ d/card\ S$

proof –

let $?C = \{a. a \subseteq S \wedge card\ a = d\}$

have $card\ ?C > 0$ **unfolding** *n-subsets*[*OF* *assms*(2)] **using** *zero-less-binomial*[*OF* *assms*(1)] **by** *simp*

hence $ne: ?C \neq \{\}$ *finite* $?C$ **using** *card-gt-0-iff* **by** *blast+*

have *card-S-gt-0*: $card\ S > 0$ **using** *assms*(2,3) *card-gt-0-iff* **by** *auto*

have *measure* (*pmf-of-set* $?C$) $\{x. s \notin x\} = real\ (card\ \{T. T \subseteq S \wedge card\ T = d \wedge s \notin T\}) / card\ ?C$

by (*subst* *measure-pmf-of-set*[*OF* *ne*]) (*simp-all* *add:Int-def*)

also **have** $\dots = real\ (card\ \{T. T \subseteq (S - \{s\}) \wedge card\ T = d\}) / card\ ?C$

by (*intro* *arg-cong2*[**where** $f=(\lambda x\ y. real\ (card\ x)/y)$] *Collect-cong*) *auto*

also **have** $\dots = real(card\ (S - \{s\})\ choose\ d) / real\ (card\ S\ choose\ d)$

using *assms*(1,2) **by** (*subst* (1 2) *n-subsets*) *auto*

also **have** $\dots = real((card\ S - 1)\ choose\ d) / real\ (card\ S\ choose\ d)$ **using** *assms* **by** *simp*

also **have** $\dots = real(card\ S * ((card\ S - 1)\ choose\ d)) / real\ (card\ S * (card\ S\ choose\ d))$

using *card-S-gt-0* **by** *simp*

also **have** $\dots = real\ (card\ S - d) / real\ (card\ S)$

unfolding *binomial-absorb-comp*[*symmetric*] **by** *simp*

also **have** $\dots = (real\ (card\ S) - real\ d) / real\ (card\ S)$

using *assms* **by** (*subst* *of-nat-diff*) *auto*

also **have** $\dots = (1 - real\ d/card\ S)$ **using** *card-S-gt-0* **by** (*simp* *add:field-simps*)

finally **show** *measure* (*pmf-of-set* $?C$) $\{x. s \notin x\} = (1 - real\ d/card\ S)$ **by** *simp*

hence $\langle 1 - measure\ (pmf-of-set\ ?C)\ \{x. s \notin x\} = real\ d/card\ S \rangle$ **by** *simp*

thus *measure-pmf.prob* (*pmf-of-set* $?C$) $\{\omega. s \in \omega\} = real\ d/card\ S$

by (subst (asm) measure-pmf.prob-compl[symmetric]) (auto simp:diff-eq Compl-eq)
qed

lemma *n-subsets-distribution-neg-assoc*:

assumes *finite S k ≤ card S*

defines $p \equiv \text{pmf-of-set } \{T. T \subseteq S \wedge \text{card } T = k\}$

shows *measure-pmf.neg-assoc p (∈) S*

proof –

define $F :: \text{bool multiset}$ **where** $F = \text{replicate-mset } k \text{ True} + \text{replicate-mset } (\text{card } S - k) \text{ False}$

let $?qset = \{ \varphi \in \text{extensional } S. \text{image-mset } \varphi (\text{mset-set } S) = F \}$

define q **where** $q = \text{pmf-of-set } ?qset$

have $a: \text{card } S = \text{size } F$ **unfolding** $F\text{-def}$ **using** $\text{assms}(2)$ **by** simp

have $b: \text{image-mset } \varphi (\text{mset-set } S) = F \longleftrightarrow \text{card } (\varphi -' \{ \text{True} \} \cap S) = k$

(is $?L \longleftrightarrow ?R$) **for** φ

proof –

have $de: \text{card } (\varphi -' \{ \text{False} \} \cap S) + \text{card } (\varphi -' \{ \text{True} \} \cap S) = \text{card } S$

using $\text{assms}(1)$ **by** (subst $\text{card-Un-disjoint[symmetric]}$) (auto intro:arg-cong[**where** $f=\text{card}$])

have $?L \longleftrightarrow (\forall i. \text{count } \{ \# \varphi x. x \in \# \text{mset-set } S \# \} i = \text{count } F i)$ **using** *multiset-eq-iff* **by** blast

also have $\dots \longleftrightarrow (\forall i. \text{card } (\varphi -' \{ i \} \cap S) = \text{count } F i)$

unfolding $\text{count-image-mset-eq-card-vimage[OF assms}(1)]$ $\text{vimage-def Int-def}$

by ($\text{simp add:conj-commute}$)

also have $\dots \longleftrightarrow \text{card } (\varphi -' \{ \text{True} \} \cap S) = k \wedge \text{card } (\varphi -' \{ \text{False} \} \cap S) = (\text{card } S - k)$

unfolding $F\text{-def}$ **using** $\text{assms}(1)$ **by** auto

also have $\dots \longleftrightarrow ?R$ **using** $\text{assms}(2)$ de **by** auto

finally show $?thesis$ **by** simp

qed

have $\text{bij-betw } (\lambda \omega. \lambda s \in S. s \in \omega) \{T. T \subseteq S \wedge \text{card } T = k\} ?qset$ **unfolding** b

by (intro bij-betwI [**where** $g = \lambda \varphi. \{x. x \in S \wedge \varphi x\}$]) $Pi\text{-I ext}$

(auto intro: arg-cong[**where** $f=\text{card}$]) $\text{simp:extensional-def vimage-def Int-def conj-commute}$)

moreover have $\text{card } \{T. T \subseteq S \wedge \text{card } T = k\} > 0$

unfolding $n\text{-subsets[OF assms}(1)]$ **by** (intro $\text{zero-less-binomial assms}(2)$)

hence $\{T. T \subseteq S \wedge \text{card } T = k\} \neq \{\} \wedge \text{finite } \{T. T \subseteq S \wedge \text{card } T = k\}$

using card-gt-0-iff **by** blast

ultimately have $c: \text{map-pmf } (\lambda \omega. \lambda s \in S. s \in \omega) p = q$

unfolding $p\text{-def } q\text{-def}$ **by** (intro $\text{map-pmf-of-set-bij-betw}$) auto

have $\text{measure-pmf.neg-assoc } (\text{map-pmf } (\lambda \omega. \lambda s \in S. s \in \omega) p) (\lambda i \omega. \omega i) S$

unfolding $c\text{-def}$ **by** (intro $\text{multiset-permutation-distributions-are-neg-associated}$ $a\text{ assms}(1)$)

hence $d: \text{measure-pmf.neg-assoc } p (\lambda s \omega. \text{if } s \in S \text{ then } (s \in \omega) \text{ else undefined}) S$

```

    unfolding neg-assoc-map-pmf by (simp add:restrict-def cong:if-cong)
    show ?thesis by (intro measure-pmf.neg-assoc-cong[OF assms(1) - d] AE-pmfI)
  auto
qed

end

```

7 Application: Bloom Filters

The false positive probability of Bloom Filters is a case where negative association is really useful. Traditionally it is derived only approximately. Bloom [4] first derives the expected number of bits set to true given the number of elements inserted, then the false positive probability is computed, pretending that the expected number of bits is the actual number of bits.

Both Blooms original derivation and Mitzenmacher and Upfal [15] use this method.

A more correct approach would be to derive a tail bound for the number of set bits and derive a false-positive probability based on that, which unfortunately leads to a complex formula.

An exact result has later been derived using combinatorial methods by Gopinathan and Sergey [10]. However their formula is less useful, as it consists of a sum with Stirling numbers and binomial coefficients.

It is however easy to see that the original bound derived by Bloom is a correct upper bound for the false positive probability using negative association. (This is pointed out by Bao et al. [?].)

In this section, we derive the same bound using this library as an example for the applicability of this library.

```

theory Negative-Association-Bloom-Filters
  imports Negative-Association-Permutation-Distributions
begin

```

```

fun bloom-filter-pmf where
  bloom-filter-pmf 0 d N = return-pmf {} |
  bloom-filter-pmf (Suc n) d N = do {
    h ← bloom-filter-pmf n d N;
    a ← pmf-of-set {a. a ⊆ {..N::nat} ∧ card a = d};
    return-pmf (a ∪ h)
  }

```

```

lemma bloom-filter-neg-assoc:
  assumes d ≤ N
  shows measure-pmf.neg-assoc (bloom-filter-pmf n d N) (λi ω. i ∈ ω) {..N}
proof (induction n)
  case 0

```

```

have a:measure-pmf.neg-assoc (bloom-filter-pmf 0 d N) ( $\lambda-$  False)  $\{..<N\}$ 
by (intro measure-pmf.indep-imp-neg-assoc measure-pmf.indep-vars-const) auto
show ?case by (intro measure-pmf.neg-assoc-cong[OF - - a] AE-pmfI) simp-all
next
case (Suc n)
let ?l = bloom-filter-pmf n d N
let ?r = pmf-of-set  $\{a. a \subseteq \{..<N\} \wedge \text{card } a = d\}$ 

define f where f j  $\omega = (\omega (\text{True},j) \vee \omega (\text{False},j))$  for  $\omega$  and j :: nat

have f-borel: f i  $\in$  borel-measurable (PiM (UNIV  $\times$   $\{i\}$ )) ( $\lambda-$  borel) (is ?L  $\in$ 
?R) for i
proof -
have f i = ( $\lambda\omega. \text{max}(\text{fst } \omega) (\text{snd } \omega)$ )  $\circ$  ( $\lambda\omega. (\omega (\text{True},i), \omega (\text{False},i))$ ) unfolding
f-def by auto
also have ...  $\in$  ?R by (intro measurable-comp[where N=borel  $\otimes_M$  borel])
measurable
finally show ?thesis by simp
qed

have 0: $\{\text{True}\} \times \{..<N\} \cup \{\text{False}\} \times \{..<N\} = \text{UNIV} \times \{..<N\}$  by auto

have s: $\{b\} \times \{..<N\} = \text{Pair } b \text{ ' } \{..<N\}$  for b :: bool by auto

have measure-pmf.neg-assoc (map-pmf snd (pair-pmf ?l ?r)) ( $\lambda i \omega. i \in \omega$ )
 $(\{..<N\})$ 
unfolding map-snd-pair-pmf using assms by (intro n-subsets-distribution-neg-assoc)
auto
hence na-l:
measure-pmf.neg-assoc (pair-pmf ?l ?r) ( $\lambda i \omega. \text{snd } i \in \text{case-bool fst snd } (\text{fst } i)$ )
 $\omega$   $(\{\text{False}\} \times \{..<N\})$ 
unfolding s neg-assoc-map-pmf by (subst measure-pmf.neg-assoc-reindex)
(auto intro:inj-onI)

have measure-pmf.neg-assoc (map-pmf fst (pair-pmf ?l ?r)) ( $\in$ )  $(\{..<N\})$ 
unfolding map-fst-pair-pmf using Suc by simp
hence na-r:
measure-pmf.neg-assoc (pair-pmf ?l ?r) ( $\lambda i \omega. \text{snd } i \in \text{case-bool fst snd } (\text{fst } i)$ )
 $\omega$   $(\{\text{True}\} \times \{..<N\})$ 
unfolding s neg-assoc-map-pmf by (subst measure-pmf.neg-assoc-reindex)
(auto intro:inj-onI)

have c: prob-space.indep-var (pair-pmf ?l ?r)
(PiM  $(\{\text{True}\} \times \{..<N\})$ ) ( $\lambda-$  borel) x (PiM  $(\{\text{False}\} \times \{..<N\})$ ) ( $\lambda-$  borel)
y
if x = ( $(\lambda\omega. \lambda i \in \{\text{True}\} \times \{..<N\}. \text{snd } i \in \omega)$ )  $\circ$  fst y = ( $(\lambda\omega. \lambda i \in \{\text{False}\} \times$ 
 $\{..<N\}. \text{snd } i \in \omega)$ )  $\circ$  snd
for x y
unfolding that by (intro prob-space.indep-var-compose[OF - indep-var-pair-pmf])

```

prob-space-measure-pmf
 (auto simp:space-PiM)

have *a:measure-pmf.neg-assoc* (*pair-pmf ?l ?r*) ($\lambda i \omega. \text{snd } i \in \text{case-bool fst snd (fst } i) \omega$) ($UNIV \times \{..<N\}$)
by (*intro measure-pmf.neg-assoc-combine*[*OF - 0*] *na-l na-r c*) (auto simp: *restrict-def mem-Times-iff*)
have *measure-pmf.neg-assoc* (*pair-pmf ?l ?r*) ($\lambda i \omega. f i (\lambda i. \text{snd } i \in \text{case-bool fst snd (fst } i) \omega)$) ($\{..<N\}$)
by (*intro measure-pmf.neg-assoc-compose*[*OF - a*, **where** *deps=* $\lambda j. UNIV \times \{j\}$ **and** *$\eta = Fwd$*]
monotoneI depends-onI f-borel) (auto simp:*f-def*)
hence *measure-pmf.neg-assoc* (*pair-pmf ?l ?r*) ($\lambda i \omega. i \in \text{fst } \omega \vee i \in \text{snd } \omega$) ($\{..<N\}$)
unfolding *f-def* **by** (*simp add:case-prod-beta'*)
hence *measure-pmf.neg-assoc* (*map-pmf* (*case-prod* (\cup)) (*pair-pmf ?l ?r*)) (\in) ($\{..<N\}$)
unfolding *neg-assoc-map-pmf* **by** (*simp add:case-prod-beta'*)
thus *?case* **by** (*simp add:pair-pmf-def map-bind-pmf Un-commute*)
qed

lemma *bloom-filter-cell-prob*:

assumes $d \leq N$ $i < N$
shows *measure* (*bloom-filter-pmf* n d N) $\{\omega. i \in \omega\} = 1 - (1 - \text{real } d/\text{real } N)^{\wedge n}$
proof –
have *measure* (*bloom-filter-pmf* n d N) $\{\omega. i \notin \omega\} = (1 - \text{real } d/\text{real } N)^{\wedge n}$
proof (*induction n*)
case 0 **thus** *?case* **by** *simp*
next
case (*Suc n*)
let *?p = pair-pmf* (*bloom-filter-pmf* n d N) (*pmf-of-set* $\{a. a \subseteq \{..<N\} \wedge \text{card } a = d\}$)
have *a*: $\{\omega. i \notin \text{fst } \omega \wedge i \notin \text{snd } \omega\} = (\{\omega. i \notin \omega\}) \times (\{\omega. i \notin \omega\})$ **by** *auto*
have *measure* *?p* $\{\omega. i \notin \text{fst } \omega \wedge i \notin \text{snd } \omega\} = (1 - \text{real } d/N)^{\wedge n} * (1 - \text{real } d/\text{card } \{..<N\})$
using *assms* **unfolding** *a measure-pair-pmf*
by (*intro Suc n-subsets-prob(1) arg-cong2[where f=(*)]*) *auto*
also **have** $\dots = (1 - \text{real } d/N)^{\wedge (n+1)}$ **by** *simp*
finally **have** *measure* *?p* $\{\omega. i \notin \text{fst } \omega \wedge i \notin \text{snd } \omega\} = (1 - \text{real } d/N)^{\wedge (n+1)}$
by *simp*
hence *measure* (*map-pmf* ($\lambda \omega. \text{snd } \omega \cup \text{fst } \omega$) *?p*) $\{\omega. i \notin \omega\} = (1 - \text{real } d/N)^{\wedge (n+1)}$
by (*simp add:disj-commute*)
thus *?case* **by** (*simp add:pair-pmf-def map-bind-pmf*)
qed
hence $1 - \text{measure} (\text{bloom-filter-pmf } n \text{ } d \text{ } N) \{\omega. i \in \omega\} = (1 - \text{real } d/\text{real } N)^{\wedge n}$

by (*subst measure-pmf.prob-compl[symmetric]*) (*auto simp:set-diff-eq*)
 thus *?thesis* by *simp*
 qed

lemma *bloom-filter-false-positive-prob*:
 assumes $d \leq N$ $T \subseteq \{..<N\}$ $\text{card } T = d$
 shows $\text{measure } (\text{bloom-filter-pmf } n \ d \ N) \ \{\omega. \ T \subseteq \omega\} \leq (1 - (1 - \text{real } d/\text{real } N)^n)^d$
 (is $?L \leq ?R$)
proof –
 let $?p = \text{bloom-filter-pmf } n \ d \ N$
 have *na*: *measure-pmf.neg-assoc* (*bloom-filter-pmf* $n \ d \ N$) ($\lambda i \ \omega. \ i \in \omega$) T
 by (*intro measure-pmf.neg-assoc-subset[OF assms(2) bloom-filter-neg-assoc]*
assms(1))

have *fin-T*: *finite* T **using** *assms(2)* *finite-subset* **by** *auto*
 hence *a*: $\text{of-bool } (T \subseteq y) = (\prod_{t \in T. \ \text{of-bool } (t \in y)}::\text{real})$ **for** y
 by (*induction* T) *auto*

have $?L = \text{measure } ?p \ (\{\omega. \ T \subseteq \omega\} \cap \text{space } ?p)$ **by** *simp*
 also have $\dots = (\int \omega. \ (\prod_{t \in T. \ \text{of-bool } (t \in \omega)}) \ \partial ?p)$
unfolding *Bochner-Integration.integral-indicator[symmetric]* *indicator-def*
using *a* **by** (*intro integral-cong-AE AE-pmfI*) *auto*
 also have $\dots \leq (\prod_{t \in T. \ (\int \omega. \ \text{of-bool } (t \in \omega) \ \partial ?p))$
by (*intro has-int-thatD(2)[OF measure-pmf.neg-assoc-imp-prod-mono[OF - na,*
where $\eta = Fwd]$)
integrable-bounded-pmf bounded-range-imp[OF bounded-of-bool] fin-T
borel-measurable-continuous-onI) (*auto intro:monoI*)
 also have $\dots = (\prod_{t \in T. \ \text{measure } ?p \ (\{\omega. \ t \in \omega\} \cap \text{space } ?p))$
unfolding *Bochner-Integration.integral-indicator[symmetric]* *indicator-def* **by**
simp
 also have $\dots = (\prod_{t \in T. \ \text{measure } ?p \ \{\omega. \ t \in \omega\})$ **by** *simp*
 also have $\dots = (\prod_{t \in T. \ 1 - (1 - \text{real } d/\text{real } N)^n)$
using *assms(1,2)* **by** (*intro prod.cong bloom-filter-cell-prob*) *auto*
 also have $\dots = ?R$ **using** *assms(3)* **by** *simp*
 finally show *?thesis* by *simp*
 qed

end

References

- [1] R. Ahlswede and D. E. Daykin. An inequality for the weights of two families of sets, their unions and intersections. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 43:183–185, 1978.
- [2] N. Alon and J. H. Spencer. *The Probabilistic Method, Second Edition*. John Wiley & Sons, Ltd, 2nd edition, 2000.

- [3] G. Birkhoff. *Lattice Theory*. AMS, 3rd edition, 1967.
- [4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422426, July 1970.
- [5] M. Doty. Birkhoff’s representation theorem for finite distributive lattices. *Archive of Formal Proofs*, December 2022. https://isa-afp.org/entries/Birkhoff_Finite_Distributive_Lattices.html, Formal proof development.
- [6] D. Dubhashi, J. Jonasson, and D. Ranjan. Positive influence and negative dependence. *Combinatorics, Probability and Computing*, 16(1):29–41, 2007.
- [7] D. Dubhashi and D. Ranjan. Balls and bins: A study in negative dependence. *Random Structures & Algorithms*, 13(2):99–124, 1998.
- [8] D. P. Dubhashi, V. Priebe, and D. Ranjan. Negative dependence through the fkg inequality. *BRICS Report Series*, 3, 1996.
- [9] C. Fortuin, P. Kastelyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Commun. Math. Phys.*, 22:89–103, jun 1971.
- [10] K. Gopinathan and I. Sergey. Certifying certainty and uncertainty in approximate membership query structures. In S. K. Lahiri and C. Wang, editors, *Computer Aided Verification*, pages 279–303, Cham, 2020. Springer International Publishing.
- [11] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.
- [12] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. In M. Serna, R. Shaltiel, K. Jansen, and J. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 617–631, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [13] K. Joag-Dev and F. Proschan. Negative association of random variables with applications. *Annals of Statistics*, 11:286–295, 1983.
- [14] S. Lisawadi and T.-C. Hu. On the negative association property for the dependent bootstrap random variables. *Lobachevskii Journal of Mathematics*, 32:32–38, 2011.
- [15] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, USA, 2nd edition, 2017.

- [16] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [17] R. Pemantle. Towards a theory of negative dependence. *Journal of Mathematical Physics*, 41(3):1371–1390, 03 2000.