

S-Finite Measure Monad on Quasi-Borel Spaces

Michikazu Hirata, Yasuhiko Minamide

April 21, 2024

Abstract

The s-finite measure monad on quasi-Borel spaces provides a suitable denotational model for higher-order probabilistic programs with conditioning. This entry is a formalization of the s-finite measure monad and related notions, including s-finite measures, s-finite kernels, and a proof automation for quasi-Borel spaces which is an extension of our previous entry *quasi-Borel spaces*. We also implement several examples of probabilistic programs in previous works and prove their property.

This work is a part of the work by Hirata, Minamide, and Sato, *Semantic Foundations of Higher-Order Probabilistic Programs in Isabelle/HOL* which will be presented at the 14th Conference on Interactive Theorem Proving (ITP2023).

Contents

1	Lemmas	3
2	Kernels	11
2.1	S-Finite Measures	11
2.2	Measure Kernel	45
2.3	Finite Kernel	49
2.4	Sub-Probability Kernel	50
2.5	Probability Kernel	53
2.6	S-Finite Kernel	54
3	Quasi-Borel Spaces	84
3.1	Definitions	84
3.1.1	Quasi-Borel Spaces	84
3.1.2	Empty Space	88
3.1.3	Unit Space	88
3.1.4	Sub-Spaces	89
3.1.5	Image Spaces	89
3.1.6	Binary Product Spaces	90
3.1.7	Binary Coproduct Spaces	92

3.1.8	Product Spaces	101
3.1.9	Coproduct Spaces	102
3.1.10	List Spaces	105
3.1.11	Option Spaces	111
3.1.12	Function Spaces	111
3.1.13	Ordering on Quasi-Borel Spaces	112
3.2	Morphisms of Quasi-Borel Spaces	117
3.3	Relation to Measurable Spaces	126
3.3.1	The Functor R	126
3.3.2	The Functor L	127
3.3.3	The Adjunction	136
3.3.4	Morphism Pred	157
3.3.5	The Adjunction w.r.t. Ordering	160
4	The S-Finite Measure Monad	164
4.1	The S-Finite Measure Monad	164
4.1.1	Space of S-Finite Measures	164
4.1.2	The S-Finite Measure Monad	171
4.1.3	l	175
4.1.4	Return	178
4.1.5	Bind	179
4.1.6	The Functorial Action	184
4.1.7	Join	185
4.1.8	Strength	186
4.1.9	The Probability Monad	197
4.1.10	Almost Everywhere	206
4.1.11	Integral	208
4.1.12	Binary Product Measures	224
4.1.13	The Inverse Function of l	232
4.1.14	PMF and SPMF	237
4.1.15	Density	238
4.1.16	Normalization	242
4.1.17	Product Measures	245
4.2	Measures	247
4.2.1	The Lebesgue Measure	247
4.2.2	Counting Measure	248
4.2.3	Normal Distribution	248
4.2.4	Uniform Distribution	249
4.2.5	Bernoulli Distribution	251
5	Examples	252
5.1	Montecarlo Approximation	252
5.2	Query	257
5.2.1	<code>twoUs</code>	262

5.2.2	Two Dice	264
5.2.3	Gaussian Mean Learning	268
5.2.4	Continuous Distributions	279
5.2.5	Normal Distribution	280
5.2.6	Half Normal Distribution	281
5.2.7	Erlang Distribution	284
5.2.8	Uniform Distribution on $(0, 1) \times (0, 1)$.	285
5.2.9	If then else	286
5.2.10	Weekend	287
5.2.11	Whattime	289
5.2.12	Distributions on Functions	292

For the terminology of s-finite measures/kernels, we refer to the work by Staton [4]. For the definition of the s-finite measure monad, we refer to the lecture note by Yang [6]. The construction of the s-finite measure monad is based on the detailed pencil-and-paper proof by Tetsuya Sato.

1 Lemmas

theory *Lemmas-S-Finite-Measure-Monad*

imports *HOL-Probability.Probability Standard-Borel-Spaces.StandardBorel*

begin

lemma *integrable-mono-measure*:

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

assumes $[\text{measurable-cong, measurable}]: \text{sets } M = \text{sets } N \ M \leq N \ \text{integrable } N \ f$

shows *integrable* $M \ f$

using *assms(3) nn-integral-mono-measure[OF assms(1,2), of $\lambda x. \text{ennreal } (\text{norm } (f \ x))$]*

by *(auto simp: integrable-iff-bounded)*

lemma *AE-mono-measure*:

assumes $\text{sets } M = \text{sets } N \ M \leq N \ \text{AE } x \ \text{in } N. \ P \ x$

shows *AE* $x \ \text{in } M. \ P \ x$

by *(metis (no-types, lifting) AE-E Collect-cong assms eventually-ae-filter le-measure le-zero-eq null-setsI sets-eq-imp-space-eq)*

lemma *finite-measure-return:finite-measure* (return $M \ x$)

by *(auto intro!: finite-measureI) (metis ennreal-top-neq-one ennreal-zero-neq-top indicator-eq-0-iff indicator-eq-1-iff)*

lemma *nn-integral-return'*:

assumes $x \notin \text{space } M$

shows $(\int^+ x. \ g \ x \ \partial \text{return } M \ x) = 0$

proof –

have *emeasure* (return $M \ x$) $A = 0$ **for** A

by(cases $A \in \text{sets } M$, insert *assms*) (auto *simp*: indicator-def emeasure-notin-sets
dest: sets.sets-into-space)
thus ?thesis
by(auto *simp*: nn-integral-def simple-integral-def) (meson SUP-least le-zero-eq)
qed

lemma pair-measure-return: return M $l \otimes_M$ return N $r =$ return $(M \otimes_M N)$
 (l, r)

proof(safe intro!: measure-eqI)

fix A
assume $A \in \text{sets } (\text{return } M$ $l \otimes_M$ return N $r)$
then have $A[\text{measurable}]$: $A \in \text{sets } (M \otimes_M N)$ **by** *simp*
note [*measurable-cong*] = sets-return[of M] sets-return[of N]
interpret finite-measure return N r **by**(*simp* add: finite-measure-return)
consider $l \notin \text{space } M \mid r \notin \text{space } N \mid l \in \text{space } M$ $r \in \text{space } N$ **by** auto
then show emeasure (return M $l \otimes_M$ return N r) $A =$ emeasure (return $(M \otimes_M N)$ (l, r)) A (**is** ?lhs = ?rhs)
by(cases, insert sets.sets-into-space[OF A]) (auto *simp*: emeasure-pair-measure
nn-integral-return' space-pair-measure nn-integral-return, auto *simp*: indicator-def)
qed *simp-all*

lemma null-measure-distr: distr (null-measure M) N $f =$ null-measure N
by(auto intro!: measure-eqI *simp*: distr-def emeasure-sigma)

lemma integral-measurable-subprob-algebra2:

fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
assumes [*measurable*]: $(\lambda(x, y). f$ x $y) \in \text{borel-measurable } (M \otimes_M N)$ $L \in$
measurable M (*subprob-algebra* N)
shows $(\lambda x. \text{integral}^L (L$ $x) (f$ $x)) \in \text{borel-measurable } M$
proof –
note *integral-measurable-subprob-algebra*[*measurable*]
note *measurable-distr2*[*measurable*]
have $(\lambda x. \text{integral}^L (\text{distr } (L$ $x) (M \otimes_M N) (\lambda y. (x, y))) (\lambda(x, y). f$ x $y)) \in$
borel-measurable M
by *measurable*
then show $(\lambda x. \text{integral}^L (L$ $x) (f$ $x)) \in \text{borel-measurable } M$
by (*rule* *measurable-cong*[*THEN iffD1, rotated*])
(*simp* add: *integral-distr*)

qed

lemma distr-id':

assumes sets $N =$ sets M
and $\bigwedge x. x \in \text{space } N \implies f$ $x = x$
shows distr N M $f = N$
by(*simp* add: distr-cong[OF refl refl, of N f id, *simplified*, OF *assms*(2), *simplified*]
distr-id2[OF *assms*(1)][*symmetric*] id-def)

lemma measure-density-times:

assumes [*measurable*]: $S \in \text{sets } M$ $X \in \text{sets } M$ $r \neq \infty$

shows $\text{measure (density } M (\lambda x. \text{indicator } S x * r)) X = \text{enn2real } r * \text{measure } M (S \cap X)$
proof –
have $[\text{simp}]: \text{density } M (\lambda x. \text{indicator } S x * r) = \text{density (density } M (\text{indicator } S)) (\lambda \cdot. r)$
by $(\text{simp add: density-density-eq})$
show $?thesis$
by $(\text{simp add: measure-density-const}[OF - \text{assms}(3)] \text{measure-restricted})$
qed

lemma complete-the-square:

fixes $a b c x :: \text{real}$
assumes $a \neq 0$
shows $a * x^2 + b * x + c = a * (x + (b / (2 * a)))^2 - ((b^2 - 4 * a * c) / (4 * a))$
using $\text{assms by (simp add: comm-semiring-1-class.power2-sum power2-eq-square[of } b / (2 * a)] \text{ring-class.ring-distrib}(1) \text{division-ring-class.diff-divide-distrib power2-eq-square[of } b])$

lemma complete-the-square2':

fixes $a b c x :: \text{real}$
assumes $a \neq 0$
shows $a * x^2 - 2 * b * x + c = a * (x - (b / a))^2 - ((b^2 - a * c) / a)$
using $\text{complete-the-square}[OF \text{assms, where } b = -2 * b \text{ and } x = x \text{ and } c = c]$
by $(\text{simp add: division-ring-class.diff-divide-distrib assms})$

lemma normal-density-mu-x-swap:

$\text{normal-density } \mu \sigma x = \text{normal-density } x \sigma \mu$
by $(\text{simp add: normal-density-def power2-commute})$

lemma normal-density-plus-shift: $\text{normal-density } \mu \sigma (x + y) = \text{normal-density } (\mu - x) \sigma y$

by $(\text{simp add: normal-density-def add commute diff-diff-eq2})$

lemma normal-density-times:

assumes $\sigma > 0 \sigma' > 0$
shows $\text{normal-density } \mu \sigma x * \text{normal-density } \mu' \sigma' x = (1 / \text{sqrt } (2 * \text{pi} * (\sigma^2 + \sigma'^2))) * \text{exp } (- (\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2))) * \text{normal-density } ((\mu * \sigma'^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt } (\sigma^2 + \sigma'^2)) x$
(is $?lhs = ?rhs)$

proof –

have $\text{non0: } 2 * \sigma^2 \neq 0 \ 2 * \sigma'^2 \neq 0 \ \sigma^2 + \sigma'^2 \neq 0$
using assms by auto
have $?lhs = \text{exp } (- ((x - \mu)^2 / (2 * \sigma^2))) * \text{exp } (- ((x - \mu')^2 / (2 * \sigma'^2))) / (\text{sqrt } (2 * \text{pi} * \sigma^2) * \text{sqrt } (2 * \text{pi} * \sigma'^2))$
by $(\text{simp add: normal-density-def})$
also have $\dots = \text{exp } (- ((x - \mu)^2 / (2 * \sigma^2)) - ((x - \mu')^2 / (2 * \sigma'^2))) / (\text{sqrt } (2 * \text{pi} * \sigma^2) * \text{sqrt } (2 * \text{pi} * \sigma'^2))$
by $(\text{simp add: exp-add[of } - ((x - \mu)^2 / (2 * \sigma^2)) - ((x - \mu')^2 / (2 * \sigma'^2)), \text{simplified add-uminus-conv-diff}]$

also have ... = $\exp\left(-\frac{(x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \sqrt{(\sigma^2 + \sigma'^2)^2}) - (\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2)))}{(2 * \pi * \sigma^2) * \sqrt{2 * \pi * \sigma'^2}}\right)$

proof –

have $((x - \mu)^2 / (2 * \sigma^2)) + ((x - \mu')^2 / (2 * \sigma'^2)) = (x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \sqrt{(\sigma^2 + \sigma'^2)^2}) + (\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2)))$

(is ?lhs' = ?rhs')

proof –

have $?lhs' = (2 * ((x - \mu)^2 * \sigma'^2) + 2 * ((x - \mu')^2 * \sigma^2)) / (4 * (\sigma^2 * \sigma'^2))$

by(simp add: field-class.add-frac-eq[OF non0(1,2)])

also have ... = $((x - \mu)^2 * \sigma'^2 + (x - \mu')^2 * \sigma^2) / (2 * (\sigma^2 * \sigma'^2))$

by(simp add: power2-eq-square division-ring-class.add-divide-distrib)

also have ... = $((\sigma^2 + \sigma'^2) * x^2 - 2 * (\mu * \sigma^2 + \mu' * \sigma^2) * x + (\mu^2 * \sigma^2 + \mu'^2 * \sigma'^2)) / (2 * (\sigma^2 * \sigma'^2))$

by(simp add: comm-ring-1-class.power2-diff ring-class.left-diff-distrib semiring-class.distrib-right)

also have ... = $((\sigma^2 + \sigma'^2) * (x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 - ((\mu * \sigma^2 + \mu' * \sigma^2)^2 - (\sigma^2 + \sigma'^2) * (\mu^2 * \sigma^2 + \mu'^2 * \sigma'^2)) / (\sigma^2 + \sigma'^2)) / (2 * (\sigma^2 * \sigma'^2))$

by(simp only: complete-the-square2[OF non0(3), of x (\mu * \sigma^2 + \mu' * \sigma^2) (\mu^2 * \sigma^2 + \mu'^2 * \sigma'^2)])

also have ... = $((\sigma^2 + \sigma'^2) * (x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2) / (2 * (\sigma^2 * \sigma'^2)) - (((\mu * \sigma^2 + \mu' * \sigma^2)^2 - (\sigma^2 + \sigma'^2) * (\mu^2 * \sigma^2 + \mu'^2 * \sigma'^2)) / (\sigma^2 + \sigma'^2)) / (2 * (\sigma^2 * \sigma'^2))$

by(simp add: division-ring-class.diff-divide-distrib)

also have ... = $(x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * ((\sigma * \sigma') / \sqrt{(\sigma^2 + \sigma'^2)^2}) - (((\mu * \sigma^2 + \mu' * \sigma^2)^2 - (\sigma^2 + \sigma'^2) * (\mu^2 * \sigma^2 + \mu'^2 * \sigma'^2)) / (\sigma^2 + \sigma'^2)) / (2 * (\sigma^2 * \sigma'^2)))$

by(simp add: monoid-mult-class.power2-eq-square[of (\sigma * \sigma') / \sqrt{(\sigma^2 + \sigma'^2)}] ab-semigroup-mult-class.mult commute[of \sigma^2 + \sigma'^2])

(simp add: monoid-mult-class.power2-eq-square[of \sigma] monoid-mult-class.power2-eq-square[of \sigma])

also have ... = $(x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \sqrt{(\sigma^2 + \sigma'^2)^2}) - ((\mu * \sigma^2)^2 + (\mu' * \sigma^2)^2 + 2 * (\mu * \sigma^2) * (\mu' * \sigma^2) - (\sigma^2 * (\mu^2 * \sigma^2) + \sigma'^2 * (\mu'^2 * \sigma'^2))) / ((\sigma^2 + \sigma'^2) * (2 * (\sigma^2 * \sigma'^2)))$

by(simp add: comm-semiring-1-class.power2-sum[of \mu * \sigma^2 \mu' * \sigma^2] semiring-class.distrib-right[of \sigma^2 \sigma'^2 \mu^2 * \sigma^2 + \mu'^2 * \sigma'^2])

(simp add: semiring-class.distrib-left[of - \mu^2 * \sigma^2 \mu'^2 * \sigma'^2])

also have ... = $(x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \sqrt{(\sigma^2 + \sigma'^2)^2}) + ((\sigma^2 * \sigma'^2) * \mu^2 + (\sigma^2 * \sigma'^2) * \mu'^2 - (\sigma^2 * \sigma'^2) * 2 * (\mu * \mu')) / ((\sigma^2 + \sigma'^2) * (2 * (\sigma^2 * \sigma'^2)))$

by(simp add: monoid-mult-class.power2-eq-square division-ring-class.minus-divide-left)

also have ... = $(x - (\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \sqrt{(\sigma^2 + \sigma'^2)^2}) + (\mu^2 + \mu'^2 - 2 * (\mu * \mu')) / ((\sigma^2 + \sigma'^2) * 2))$

using assms **by**(simp add: division-ring-class.add-divide-distrib division-ring-class.diff-divide-distrib)

also have ... = ?rhs'

by(simp add: comm-ring-1-class.power2-diff ab-semigroup-mult-class.mult commute[of

2])
finally show *?thesis* .
qed
thus *?thesis*
by *simp*
qed
also have ... = $(\exp(-(\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2))) / (\text{sqrt}(2 * \text{pi} * \sigma^2) * \text{sqrt}(2 * \text{pi} * \sigma'^2))) * \text{sqrt}(2 * \text{pi} * (\sigma * \sigma' / \text{sqrt}(\sigma^2 + \sigma'^2))^2) * \text{normal-density}((\mu * \sigma'^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt}(\sigma^2 + \sigma'^2)) x$
by(*simp add: exp-add[of - (x - (\mu * \sigma'^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2))^2 / (2 * (\sigma * \sigma' / \text{sqrt}(\sigma^2 + \sigma'^2))^2) - (\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2))],simplified*) *normal-density-def*)
also have ... = *?rhs*
proof -
have $\exp(-(\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2))) / (\text{sqrt}(2 * \text{pi} * \sigma^2) * \text{sqrt}(2 * \text{pi} * \sigma'^2)) * \text{sqrt}(2 * \text{pi} * (\sigma * \sigma' / \text{sqrt}(\sigma^2 + \sigma'^2))^2) = 1 / \text{sqrt}(2 * \text{pi} * (\sigma^2 + \sigma'^2)) * \exp(-(\mu - \mu')^2 / (2 * (\sigma^2 + \sigma'^2)))$
using *assms* **by**(*simp add: real-sqrt-mult*)
thus *?thesis*
by *simp*
qed
finally show *?thesis* .
qed

lemma *KL-normal-density*:

assumes [*arith*]: $b > 0 \ d > 0$

shows *KL-divergence* (*exp 1*) (*density lborel (normal-density a b)*) (*density lborel (normal-density c d)*) = $\ln(b / d) + (d^2 + (c - a)^2) / (2 * b^2) - 1 / 2$ (**is** *?lhs* = *?rhs*)

proof -

have *?lhs* = $(\int x. \text{normal-density } c \ d \ x * \ln(\text{normal-density } c \ d \ x / \text{normal-density } a \ b \ x) \ \partial \text{lborel})$

by(*unfold log-ln,rule lborel.KL-density-density*) (*use order.strict-implies-not-eq[OF normal-density-pos[of b a]]* **in** *auto*)

also have ... = $(\int x. \text{normal-density } c \ d \ x * \ln(\text{normal-density } c \ d \ x) - \text{normal-density } c \ d \ x * \ln(\text{normal-density } a \ b \ x) \ \partial \text{lborel})$

by(*simp add: ln-div[OF normal-density-pos[OF assms(2)] normal-density-pos[OF assms(1)]]* *right-diff-distrib*)

also have ... = $(\int x. \text{normal-density } c \ d \ x * \ln(\exp(-(x - c)^2 / (2 * d^2)) / \text{sqrt}(2 * \text{pi} * d^2)) - \text{normal-density } c \ d \ x * \ln(\exp(-(x - a)^2 / (2 * b^2)) / \text{sqrt}(2 * \text{pi} * b^2)) \ \partial \text{lborel})$

by(*simp add: normal-density-def*)

also have ... = $(\int x. \text{normal-density } c \ d \ x * (-(x - c)^2 / (2 * d^2) - \ln(\text{sqrt}(2 * \text{pi} * d^2))) - (\text{normal-density } c \ d \ x * (-(x - a)^2 / (2 * b^2) - \ln(\text{sqrt}(2 * \text{pi} * b^2)))) \ \partial \text{lborel})$

by(*simp add: ln-div*)

also have ... = $(\int x. \text{normal-density } c \ d \ x * (\ln(\text{sqrt}(2 * \text{pi} * b^2)) - \ln(\text{sqrt}(2 * \text{pi} * d^2))) + (\text{normal-density } c \ d \ x * ((x - a)^2 / (2 * b^2)) - \text{normal-density } c \ d \ x * ((x - c)^2 / (2 * d^2))) \ \partial \text{lborel})$

by(*auto intro!: Bochner-Integration.integral-cong simp: right-diff-distrib*)

also have ... = (∫ x. normal-density c d x * (ln (sqrt (2 * pi * b²)) - ln (sqrt (2 * pi * d²))) + (normal-density c d x * ((x - c)² / (2 * b²) + (2 * x * (c - a) + a² - c²) / (2 * b²)) - normal-density c d x * ((x - c)² / (2 * d²))) ∂lborel)
by(auto intro!: Bochner-Integration.integral-cong simp: add-divide-distrib[symmetric] power2-diff) (simp add: right-diff-distrib)
also have ... = (∫ x. (ln (sqrt (2 * pi * b²)) - ln (sqrt (2 * pi * d²))) * normal-density c d x + (((1 / (2 * b²) * (normal-density c d x * (x - c)²) + (2 * (c - a) / (2 * b²) * (normal-density c d x * x) + (a² - c²) / (2 * b²) * (normal-density c d x)) - 1 / (2 * d²) * (normal-density c d x * (x - c)²)) ∂lborel)
by(auto intro!: Bochner-Integration.integral-cong simp: add-divide-distrib[symmetric] ring-distrib)
also have ... = (∫ x. (ln (sqrt (2 * pi * b²)) - ln (sqrt (2 * pi * d²))) * normal-density c d x ∂lborel) + (((∫ x. 1 / (2 * b²) * (normal-density c d x * (x - c)²) ∂lborel) + (∫ x. (2 * (c - a) / (2 * b²) * (normal-density c d x * x) ∂lborel) + (∫ x. (a² - c²) / (2 * b²) * (normal-density c d x) ∂lborel)) - (∫ x. 1 / (2 * d²) * (normal-density c d x * (x - c)²) ∂lborel))
using integrable-normal-moment-nz-1[OF assms(2)] integrable-normal-moment[OF assms(2),where k=2] **by** simp
also have ... = ln (sqrt (2 * pi * b²)) - ln (sqrt (2 * pi * d²)) + 1 / (2 * b²) * d² + (2 * c - 2 * a) / (2 * b²) * c + (a² - c²) / (2 * b²) - 1 / (2 * d²) * d²
by(simp add: integral-normal-moment-even[OF assms(2),of - 1,simplified] integral-normal-moment-nz-1[OF assms(2)] del: times-divide-eq-left)
also have ... = ln (b / d) + 1 / (2 * b²) * d² + (2 * c - 2 * a) / (2 * b²) * c + (a² - c²) / (2 * b²) - 1 / (2 * d²) * d²
by(simp add: ln-sqrt ln-mult power2-eq-square diff-divide-distrib[symmetric] ln-div)
also have ... = ?rhs
by(auto simp: add-divide-distrib[symmetric] power2-diff left-diff-distrib) (simp add: power2-eq-square)
finally show ?thesis .
qed

lemma count-space-prod:count-space (UNIV :: ('a :: countable) set) ⊗_M count-space (UNIV :: ('b :: countable) set) = count-space UNIV
by(auto simp: pair-measure-countable)

lemma measure-pair-pmf:
fixes p :: ('a :: countable) pmf **and** q :: ('b :: countable) pmf
shows measure-pmf p ⊗_M measure-pmf q = measure-pmf (pair-pmf p q) (**is** ?lhs = ?rhs)
proof -
interpret pair-prob-space measure-pmf p measure-pmf q
by standard
have ?lhs = measure-pmf p ≫ (λx. measure-pmf q ≫ (λy. return (measure-pmf p ⊗_M measure-pmf q) (x, y)))
by(rule pair-measure-eq-bind)
also have ... = ?rhs
by(simp add: measure-pmf-bind pair-pmf-def return-pmf.rep-eq cong: return-cong[OF

sets-pair-measure-cong[*OF sets-measure-pmf-count-space*[*of p*] *sets-measure-pmf-count-space*[*of q*],*simplified count-space-prod*]]

finally show *?thesis* .

qed

lemma *distr-PiM-distr*:

assumes *finite I* $\bigwedge i. i \in I \implies$ *sigma-finite-measure* (*distr* (*M i*) (*N i*) (*f i*))

and $\bigwedge i. i \in I \implies$ *f i* \in *M i* \rightarrow_M *N i*

shows *distr* ($\prod_M i \in I. M i$) ($\prod_M i \in I. N i$) ($\lambda xi. \lambda i \in I. f i (xi i)$) = ($\prod_M i \in I. \text{distr } (M i) (N i) (f i)$)

proof –

define *M'* **where** *M' \equiv ($\lambda i. \text{if } i \in I \text{ then } M i \text{ else null-measure } (M i)$)*

have *f*[*measurable*]: $\bigwedge i. i \in I \implies$ *f i* \in *M' i* \rightarrow_M *N i* **and** [*measurable-cong*]:

$\bigwedge i. \text{sets } (M' i) = \text{sets } (M i)$ **and** [*simp*]: $\bigwedge i. i \in I \implies M' i = M i$

by(*auto simp: M'-def assms*)

interpret *product-sigma-finite* $\lambda i. \text{distr } (M' i) (N i) (f i)$

by(*auto simp: product-sigma-finite-def M'-def assms(2)*) (*auto intro!: finite-measure.sigma-finite-measure finite-measureI simp: null-measure-distr*)

interpret *ps: product-sigma-finite* *M'*

by(*auto simp: product-sigma-finite-def M'-def intro!: finite-measure.sigma-finite-measure[of null-measure -] finite-measureI sigma-finite-measure-distr[OF assms(2)]*)

have *distr* ($\prod_M i \in I. M i$) ($\prod_M i \in I. N i$) ($\lambda xi. \lambda i \in I. f i (xi i)$) = *distr* ($\prod_M i \in I. M' i$) ($\prod_M i \in I. N i$) ($\lambda xi. \lambda i \in I. f i (xi i)$)

by(*simp cong: PiM-cong*)

also have ... = ($\prod_M i \in I. \text{distr } (M' i) (N i) (f i)$)

proof(*rule PiM-eqI[OF assms(1)]*)

fix *A*

assume $\bigwedge i. i \in I \implies A i \in \text{sets } (\text{distr } (M' i) (N i) (f i))$

hence *h*[*measurable*]: $\bigwedge i. i \in I \implies A i \in \text{sets } (N i)$

by *simp*

have [*simp*]: ($\lambda xi. \lambda i \in I. f i (xi i)$) – ‘(*Pi_E* *I A*) \cap *space* (*Pi_M* *I M'*) = ($\prod_{i \in I. f i - ' A i \cap \text{space } (M' i)}$)

by(*auto simp: space-PiM*)

show *emeasure* (*distr* (*Pi_M* *I M'*) (*Pi_M* *I N*) ($\lambda xi. \lambda i \in I. f i (xi i)$)) (*Pi_E* *I A*) = ($\prod i \in I. \text{emeasure } (\text{distr } (M' i) (N i) (f i)) (A i)$)

by(*auto simp: emeasure-distr assms(1) ps.emeasure-PiM[OF assms(1)]*)

qed(*simp-all cong: sets-PiM-cong*)

also have ... = ($\prod_M i \in I. \text{distr } (M i) (N i) (f i)$)

by(*auto cong: PiM-cong*)

finally show *?thesis* .

qed

lemma *distr-PiM-distr-prob*:

assumes $\bigwedge i. i \in I \implies$ *prob-space* (*M i*)

and $\bigwedge i. i \in I \implies$ *f i* \in *M i* \rightarrow_M *N i*

shows *distr* ($\prod_M i \in I. M i$) ($\prod_M i \in I. N i$) ($\lambda xi. \lambda i \in I. f i (xi i)$) = ($\prod_M i \in I. \text{distr } (M i) (N i) (f i)$)

proof –

define *M'* **where** *M' \equiv ($\lambda i. \text{if } i \in I \text{ then } M i \text{ else return } (\text{count-space UNIV})$)*

```

undefined)
  define N' where N' ≡ (λi. if i ∈ I then N i else return (count-space UNIV)
undefined)
  interpret p: product-prob-space λi. distr (M' i) (N' i) (f i)
  by(auto simp: product-prob-space-def product-prob-space-axioms-def product-sigma-finite-def
M'-def prob-space-return N'-def assms intro!: prob-space-imp-sigma-finite prob-space.prob-space-distr)
  interpret p': product-prob-space M'
  by(auto simp: product-prob-space-def product-prob-space-axioms-def product-sigma-finite-def
M'-def prob-space-return assms intro!: prob-space-imp-sigma-finite)
  have f[measurable]: ∧i. i ∈ I ⇒ f i ∈ M' i →M N' i
  by(auto simp: assms M'-def N'-def)
  have [simp]: p.emb I = prod-emb I N'
  by standard (auto simp: prod-emb-def)
  have distr (ΠM i∈I. M i) (ΠM i∈I. N i) (λxi. λi∈I. f i (xi i)) = distr (ΠM
i∈I. M' i) (ΠM i∈I. N' i) (λxi. λi∈I. f i (xi i))
  by(simp add: M'-def N'-def cong: PiM-cong)
  also have ... = (ΠM i∈I. distr (M' i) (N' i) (f i))
  proof(rule p.PiM-eq)
    fix J F
    assume h[measurable]: finite J J ⊆ I ∧j. j ∈ J ⇒ F j ∈ p.M.events j
    then have [measurable]: ∧j. j ∈ J ⇒ F j ∈ sets (N' j) by simp
    show emeasure (distr (PiM I M') (PiM I N') (λxi. λi∈I. f i (xi i))) (p.emb
I J (PiE J F)) = (∏j∈J. emeasure (distr (M' j) (N' j) (f j)) (F j)) (is ?lhs =
?rhs)
    proof –
      have ?lhs = emeasure (PiM I M') ((λxi. λi∈I. f i (xi i)) -' (prod-emb I N'
J (PiE J F)) ∩ space (PiM I M'))
      by(simp add: emeasure-distr h)
      also have ... = emeasure (PiM I M') (prod-emb I M' J (ΠE i∈J. f i -' (F
i) ∩ space (M' i)))
      proof –
        have [simp]:(λxi. λi∈I. f i (xi i)) -' (prod-emb I N' J (PiE J F)) ∩ space
(PiM I M') = prod-emb I M' J (ΠE i∈J. f i -' (F i) ∩ space (M' i))
        using measurable-space[OF f] h(1,2,3)
        by(fastforce simp: space-PiM prod-emb-def PiE-def extensional-def Pi-def
M'-def N'-def)
        show ?thesis by simp
      qed
      also have ... = (∏i∈J. emeasure (M' i) (f i -' (F i) ∩ space (M' i)))
      by(rule p'.emeasure-PiM-emb,insert h(2)) (auto simp: h(1))
      also have ... = ?rhs
      using h(2) by(auto simp: emeasure-distr intro!: comm-monoid-mult-class.prod.cong)
      finally show ?thesis .
    qed
  qed (simp cong: sets-PiM-cong)
  also have ... = (ΠM i∈I. distr (M i) (N i) (f i))
  by(simp add: M'-def N'-def cong: distr-cong PiM-cong)
  finally show ?thesis .
qed

```

end

2 Kernels

theory *Kernels*
imports *Lemmas-S-Finite-Measure-Monad*
begin

2.1 S-Finite Measures

locale *s-finite-measure* =
fixes $M :: 'a \text{ measure}$
assumes *s-finite-sum*: $\exists Mi :: nat \Rightarrow 'a \text{ measure. } (\forall i. \text{sets } (Mi \ i) = \text{sets } M) \wedge$
 $(\forall i. \text{finite-measure } (Mi \ i)) \wedge (\forall A \in \text{sets } M. M \ A = (\sum i. Mi \ i \ A))$

lemma(in *sigma-finite-measure*) *s-finite-measure*: *s-finite-measure* M

proof

obtain $A :: nat \Rightarrow -$ where $A: \text{range } A \subseteq \text{sets } M \cup (\text{range } A) = \text{space } M \wedge i.$
emeasure $M \ (A \ i) \neq \infty$ *disjoint-family* A

by(*metis sigma-finite-disjoint*)

define *Mi* where $Mi \equiv (\lambda i. \text{measure-of } (\text{space } M) (\text{sets } M) (\lambda a. M \ (a \cap A \ i)))$

have *emeasure-Mi*: $Mi \ i \ a = M \ (a \cap A \ i)$ if $a \in \text{sets } M$ for $i \ a$

proof -

have *positive* (*sets* $(Mi \ i)$) $(\lambda a. M \ (a \cap A \ i))$ *countably-additive* (*sets* $(Mi \ i)$)
 $(\lambda a. M \ (a \cap A \ i))$

unfolding *positive-def countably-additive-def*

proof *safe*

fix $B :: nat \Rightarrow -$

assume *range* $B \subseteq \text{sets } (Mi \ i)$ *disjoint-family* B

with $A(1)$ have *range* $(\lambda j. B \ j \cap A \ i) \subseteq \text{sets } M$ *disjoint-family* $(\lambda j. B \ j \cap A \ i)$

by(*fastforce simp: Mi-def disjoint-family-on-def*)+

thus $(\sum j. M \ (B \ j \cap A \ i)) = M \ (\bigcup (\text{range } B) \cap A \ i)$

by (*metis UN-extend-simps(4) suminf-emeasure*)

qed *simp*

from *emeasure-measure-of[OF - - this]* that **show** *?thesis*

by(*auto simp add: Mi-def sets.space-closed*)

qed

have *sets-Mi*: *sets* $(Mi \ i) = \text{sets } M$ for i by(*simp add: Mi-def*)

show $\exists Mi. (\forall i. \text{sets } (Mi \ i) = \text{sets } M) \wedge (\forall i. \text{finite-measure } (Mi \ i)) \wedge (\forall A \in \text{sets}$
 $M. \text{emeasure } M \ A = (\sum i. \text{emeasure } (Mi \ i) \ A))$

proof(*safe intro!*: *exI[where x=Mi]*)

fix i

show *finite-measure* $(Mi \ i)$

using A by(*auto intro!*: *finite-measureI simp: sets-eq-imp-space-eq[OF sets-Mi]*
emeasure-Mi)

next

fix B

```

assume  $B:B \in \text{sets } M$ 
with  $A(1,4)$  have  $\text{range } (\lambda i. B \cap A i) \subseteq \text{sets } M \text{ disjoint-family } (\lambda i. B \cap A i)$ 
  by(auto simp: disjoint-family-on-def)
then show  $M B = (\sum i. (Mi i) B)$ 
  by(simp add: emeasure-Mi[OF B] suminf-emeasure A(2) B)
qed(simp-all add: sets-Mi)
qed

```

lemmas(*in finite-measure*) *s-finite-measure-finite-measure = s-finite-measure*

lemmas(*in subprob-space*) *s-finite-measure-subprob = s-finite-measure*

lemmas(*in prob-space*) *s-finite-measure-prob = s-finite-measure*

sublocale *sigma-finite-measure* \subseteq *s-finite-measure*
by(*rule s-finite-measure*)

lemma *s-finite-measureI*:
assumes $\bigwedge i. \text{sets } (Mi i) = \text{sets } M \wedge i. \text{finite-measure } (Mi i) \wedge A. A \in \text{sets } M \implies$
 $M A = (\sum i. Mi i A)$
shows *s-finite-measure* M
by *standard (use assms in blast)*

lemma *s-finite-measure-prodI*:
assumes $\bigwedge i j. \text{sets } (Mij i j) = \text{sets } M \wedge i j. Mij i j \text{ (space } M) < \infty \wedge A. A \in$
 $\text{sets } M \implies M A = (\sum i. (\sum j. Mij i j A))$
shows *s-finite-measure* M

proof –

```

define  $Mi'$  where  $Mi' \equiv (\lambda n. \text{case-prod } Mij \text{ (prod-decode } n))$ 
have sets-Mi'[measurable-cong]:  $\bigwedge i. \text{sets } (Mi' i) = \text{sets } M$ 
  using assms(1) by(simp add: Mi'-def split-beta')
have  $Mi'\text{-finite}$ :  $\bigwedge i. \text{finite-measure } (Mi' i)$ 
  using assms(2) sets-eq-imp-space-eq[OF sets-Mi'[symmetric]] top.not-eq-extremum
  by(fastforce intro!: finite-measureI simp: Mi'-def split-beta')
show ?thesis
proof (safe intro!: s-finite-measureI[where Mi=Mi'] sets-Mi' Mi'-finite)
  fix  $A$ 
  show  $A \in \text{sets } M \implies M A = (\sum i. Mi' i A)$ 
  by(simp add: assms(3) suminf-ennreal-2dimen[where f= $\lambda(x,y). Mij x y A$ ,  

simplified,OF refl,symmetric] Mi'-def split-beta')
qed
qed

```

corollary *s-finite-measure-s-finite-sumI*:
assumes $\bigwedge i. \text{sets } (Mi i) = \text{sets } M \wedge i. \text{s-finite-measure } (Mi i) \wedge A. A \in \text{sets } M$
 $\implies M A = (\sum i. Mi i A)$
shows *s-finite-measure* M
proof –
from *s-finite-measure.s-finite-sum[OF assms(2)]*

obtain M_{ij} **where** $M_{ij}[\text{measurable}]$: $\bigwedge i j. \text{sets } (M_{ij} \ i \ j) = \text{sets } M \ \bigwedge i j. \text{finite-measure } (M_{ij} \ i \ j) \ \bigwedge i j \ A. A \in \text{sets } M \implies M_{ij} \ i \ A = (\sum j. M_{ij} \ i \ j \ A)$
by (*metis* *assms*(1))
show *?thesis*
using *finite-measure.emeasure-finite*[*OF* M_{ij} (2)]
by(*auto* *intro!*: *s-finite-measure-prodI*[**where** $M_{ij} = M_{ij}$] *simp*: *assms*(3) M_{ij} *top.not-eq-extremum*)
qed

lemma *s-finite-measure-finite-sumI*:

assumes *finite* $I \ \bigwedge i. i \in I \implies \text{s-finite-measure } (M_{ij} \ i) \ \bigwedge i. i \in I \implies \text{sets } (M_{ij} \ i) = \text{sets } M$

and $\bigwedge A. A \in \text{sets } M \implies M \ A = (\sum i \in I. M_{ij} \ i \ A)$

shows *s-finite-measure* M

proof –

let $?Mi = \lambda n. \text{if } n < \text{card } I \text{ then } M_{ij} \ (\text{from-nat-into } I \ n) \text{ else null-measure } M$

show *?thesis*

proof(*rule* *s-finite-measure-s-finite-sumI*[*of* $?Mi$])

show $\bigwedge i. \text{s-finite-measure } (?Mi \ i)$

by (*metis* (*full-types*) *assms*(2) *bot-nat-0.extremum-strict* *card.empty* *emeasure-null-measure* *ennreal-top-neq-zero* *finite-measure.s-finite-measure-finite-measure* *finite-measureI* *from-nat-into* *infinity-ennreal-def*)

next

show $\bigwedge i. \text{sets } (?Mi \ i) = \text{sets } M$

by (*metis* (*full-types*) *assms*(3) *card-gt-0-iff* *dual-order.strict-trans2* *from-nat-into* *less-zeroE* *linorder-le-less-linear* *sets-null-measure*)

next

fix A

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

then have $M \ A = (\sum i \in I. M_{ij} \ i \ A)$

by *fact*

also have $\dots = (\sum n < \text{card } I. M_{ij} \ (\text{from-nat-into } I \ n) \ A)$

by(*rule* *sum.card-from-nat-into*[*symmetric*])

also have $\dots = (\sum n < \text{card } I. ?Mi \ n \ A)$

by *simp*

also have $\dots = (\sum n. ?Mi \ n \ A)$

by(*rule* *suminf-finite*[*symmetric*]) *auto*

finally show $M \ A = (\sum n. ?Mi \ n \ A)$.

qed

qed

lemma *countable-space-s-finite-measure*:

assumes *countable* (*space* M) *sets* $M = \text{Pow } (\text{space } M)$

shows *s-finite-measure* M

proof –

define M_i **where** $M_i \equiv (\lambda i. \text{measure-of } (\text{space } M) \ (\text{sets } M) \ (\lambda A. \text{emeasure } M \ (A \cap \{\text{from-nat-into } (\text{space } M) \ i\})))$

have *sets-Mi*[*measurable-cong*,*simp*]: *sets* $(M_i \ i) = \text{sets } M$ **for** i

by(*auto* *simp*: *Mi-def*)

have *emeasure-Mi*: $\text{emeasure } (Mi\ i)\ A = \text{emeasure } M\ (A \cap \{\text{from-nat-into } (\text{space } M)\ i\})$ **if** [*measurable*]: $A \in \text{sets } M$ **and** $i: i \in \text{to-nat-on } (\text{space } M)\ '(\text{space } M)$ **for** $i\ A$

proof –

have *from-nat-into* (*space* M) $i \in \text{space } M$
by (*simp add: from-nat-into-def i inv-into-into*)

hence [*measurable*]: $\{\text{from-nat-into } (\text{space } M)\ i\} \in \text{sets } M$
using *assms(2)* **by** *auto*

have $1: \text{countably-additive } (\text{sets } M)\ (\lambda A. \text{emeasure } M\ (A \cap \{\text{from-nat-into } (\text{space } M)\ i\}))$
(space M) i)

unfolding *countably-additive-def*

proof *safe*

fix $B :: \text{nat} \Rightarrow -$

assume $\text{range } B \subseteq \text{sets } M$ *disjoint-family* B

then have [*measurable*]: $\bigwedge i. B\ i \in \text{sets } M$ **and** *disjoint-family* $(\lambda j. B\ j \cap \{\text{from-nat-into } (\text{space } M)\ i\})$

by (*auto simp: disjoint-family-on-def*)

then have $(\sum j. \text{emeasure } M\ (B\ j \cap \{\text{from-nat-into } (\text{space } M)\ i\})) = \text{emeasure } M\ (\bigcup (\text{range } (\lambda j. B\ j \cap \{\text{from-nat-into } (\text{space } M)\ i\})))$

by (*intro suminf-emeasure*) *auto*

thus $(\sum j. \text{emeasure } M\ (B\ j \cap \{\text{from-nat-into } (\text{space } M)\ i\})) = \text{emeasure } M\ (\bigcup (\text{range } B) \cap \{\text{from-nat-into } (\text{space } M)\ i\})$

by *simp*

qed

have $2: \text{positive } (\text{sets } M)\ (\lambda A. \text{emeasure } M\ (A \cap \{\text{from-nat-into } (\text{space } M)\ i\}))$
by (*auto simp: positive-def*)

show *?thesis*

by (*simp add: Mi-def emeasure-measure-of-sigma[OF sets.sigma-algebra-axioms 2 1]*)

qed

define Mi' **where** $Mi' \equiv (\lambda i. \text{if } i \in \text{to-nat-on } (\text{space } M)\ '(\text{space } M)\ \text{then } Mi\ i\ \text{else } \text{null-measure } M)$

have [*measurable-cong, simp*]: $\text{sets } (Mi'\ i) = \text{sets } M$ **for** i

by (*auto simp: Mi'-def*)

show *?thesis*

proof (*rule s-finite-measure-s-finite-sumI[where Mi=Mi']*)

fix A

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

show $\text{emeasure } M\ A = (\sum i. \text{emeasure } (Mi'\ i)\ A)$ (**is** *?lhs = ?rhs*)

proof –

have $?lhs = (\int^+ x. \text{emeasure } M\ \{x\}\ \partial \text{count-space } A)$

using *sets.sets-into-space[OF A]* **by** (*auto intro!: emeasure-countable-singleton simp: assms(2) countable-subset[OF - assms(1)]*)

also have $\dots = (\int^+ x. \text{emeasure } (Mi\ (\text{to-nat-on } (\text{space } M)\ x))\ A\ \partial \text{count-space } A)$

proof (*safe intro!: nn-integral-cong*)

fix x

assume $x \in \text{space } (\text{count-space } A)$

then have $1: x \in A$ **by** *simp*

```

hence 2:to-nat-on (space M) x ∈ to-nat-on (space M) ‘ (space M)
  using A assms(2) by auto
have [simp]: from-nat-into (space M) (to-nat-on (space M) x) = x
  by (metis 1 2 A assms(1) eq-from-nat-into-iff in-mono sets.sets-into-space)
show emeasure M {x} = emeasure (Mi (to-nat-on (space M) x)) A
  using 1 by(simp add: emeasure-Mi[OF A 2])
qed
also have ... = (∫+ x∈A. emeasure (Mi (to-nat-on (space M) x)) A
∂count-space UNIV)
  by (simp add: nn-integral-count-space-indicator)
also have ... = (∫+ i∈to-nat-on (space M) ‘ A. emeasure (Mi i) A ∂count-space
UNIV)
  by(rule nn-integral-count-compose-inj[OF inj-on-subset[OF inj-on-to-nat-on[OF
assms(1)] sets.sets-into-space[OF A]]])
also have ... = (∫+ i∈to-nat-on (space M) ‘ A. emeasure (Mi' i) A
∂count-space UNIV)
proof –
  {
    fix x
    assume x ∈ A
    then have to-nat-on (space M) x ∈ to-nat-on (space M) ‘ (space M)
      using sets.sets-into-space[OF A] by auto
    hence emeasure (Mi (to-nat-on (space M) x)) A = emeasure (Mi' (to-nat-on
(space M) x)) A
      by(auto simp: Mi'-def)
  }
thus ?thesis
  by(auto intro!: nn-integral-cong simp: indicator-def)
qed
also have ... = (∫+ i. emeasure (Mi' i) A ∂count-space UNIV)
proof –
  {
    fix i
    assume i:i ∉ to-nat-on (space M) ‘ A
    have from-nat-into (space M) i ∉ A if i ∈ to-nat-on (space M) ‘ (space
M)
      by (metis i image-eqI that to-nat-on-from-nat-into)
    with emeasure-Mi have emeasure (Mi' i) A = 0
      by(auto simp: Mi'-def)
  }
thus ?thesis
  by(auto intro!: nn-integral-cong simp: indicator-def)
qed
also have ... = ?rhs
  by(rule nn-integral-count-space-nat)
finally show ?thesis .
qed
next
fix i

```

```

show s-finite-measure (Mi' i)
proof -
{
  fix x
  assume h: x ∈ space M i = to-nat-on (space M) x
  then have i: i ∈ to-nat-on (space M) ' space M
    by blast
  have x: from-nat-into (space M) i = x
    using h by (simp add: assms(1))
  consider M {x} = 0 | M {x} ≠ 0 M {x} < ∞ | M {x} = ∞
    using top.not-eq-extremum by fastforce
  hence s-finite-measure (Mi (to-nat-on (space M) x))
  proof cases
    case 1
    then have [simp]: Mi i = null-measure M
      by (auto intro!: measure-eqI simp: emeasure-Mi[OF - i] x Int-insert-right)
    show ?thesis
    by (auto simp: h(2)[symmetric] intro!: finite-measure.s-finite-measure-finite-measure
finite-measureI)
    next
    case 2
    then show ?thesis
      unfolding h(2)[symmetric]
    by (auto intro!: finite-measure.s-finite-measure-finite-measure finite-measureI
simp: sets-eq-imp-space-eq[OF sets-Mi] emeasure-Mi[OF - i] x h(1))
    next
    case 3
    show ?thesis
      unfolding h(2)[symmetric] s-finite-measure-def
    proof (safe intro!: exI[where x=λj. return M x] prob-space.finite-measure
prob-space-return h(1))
      fix A
      assume A ∈ sets (Mi i)
      then have [measurable]: A ∈ sets M
        by (simp add: Mi-def)
      thus emeasure (Mi i) A = (∑ i. emeasure (return M x) A)
        by (simp add: emeasure-Mi[OF - i] x) (cases x ∈ A, auto simp: 3
nn-integral-count-space-nat[symmetric])
      qed (auto simp: Mi-def)
    qed
  }
  thus ?thesis
  by (auto simp: Mi'-def) (auto intro!: finite-measure.s-finite-measure-finite-measure
finite-measureI)
  qed
  qed simp
  qed

```

lemma s-finite-measure-subprob-space:

$s\text{-finite-measure } M \iff (\exists Mi :: \text{nat} \Rightarrow 'a \text{ measure. } (\forall i. \text{sets } (Mi \ i) = \text{sets } M) \wedge (\forall i. (Mi \ i) (\text{space } M) \leq 1) \wedge (\forall A \in \text{sets } M. M \ A = (\sum i. Mi \ i \ A)))$

proof

assume $\exists Mi. (\forall i. \text{sets } (Mi \ i) = \text{sets } M) \wedge (\forall i. \text{emeasure } (Mi \ i) (\text{space } M) \leq 1) \wedge (\forall A \in \text{sets } M. M \ A = (\sum i. (Mi \ i) \ A))$

then obtain Mi **where** $1 : \bigwedge i. \text{sets } (Mi \ i) = \text{sets } M \wedge i. \text{emeasure } (Mi \ i) (\text{space } M) \leq 1 (\forall A \in \text{sets } M. M \ A = (\sum i. (Mi \ i) \ A))$

by *auto*

thus $s\text{-finite-measure } M$

by(*auto simp: s-finite-measure-def sets-eq-imp-space-eq[OF 1(1)] intro!: finite-measureI exI[where x=Mi]*) (*metis ennreal-one-less-top linorder-not-le*)

next

assume $s\text{-finite-measure } M$

then obtain Mi' **where** $Mi' : \bigwedge i. \text{sets } (Mi' \ i) = \text{sets } M \wedge i. \text{finite-measure } (Mi' \ i) \wedge A. A \in \text{sets } M \implies M \ A = (\sum i. Mi' \ i \ A)$

by (*metis s-finite-measure.s-finite-sum*)

obtain u **where** $u : \bigwedge i. u \ i < \infty \wedge i. Mi' \ i (\text{space } M) = u \ i$

using $Mi'(2)$ *finite-measure.emeasure-finite top.not-eq-extremum* **by** *fastforce*

define Mmn **where** $Mmn \equiv (\lambda(m,n). \text{if } n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil \text{ then scale-measure } (1 / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u \ m) \rceil)) (Mi' \ m) \text{ else } (\text{sigma } (\text{space } M) (\text{sets } M)))$

have $\text{sets-}Mmn : \text{sets } (Mmn \ k) = \text{sets } M$ **for** k **by**(*simp add: Mmn-def split-beta Mi'*)

have $\text{emeasure-}Mmn : (Mmn \ (m, \ n)) \ A = (Mi' \ m \ A) / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u \ m) \rceil)$ **if** $n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil$ **and** $A \in \text{sets } M$ **for** $n \ m \ A$

by(*auto simp: Mmn-def that ennreal-divide-times*)

have $\text{emeasure-}Mmn\text{-less1} : (Mmn \ (m, \ n)) \ A \leq 1$ **for** $m \ n \ A$

proof (*cases* $n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil \wedge A \in \text{sets } M$)

case $h : \text{True}$

have $(Mi' \ m) \ A \leq \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u \ m) \rceil)$

by(*rule order.trans[OF emeasure-mono[OF sets.sets-into-space sets.top]]*)

(*insert* $u(1) \ h, \text{auto simp: } u(2)[\text{symmetric}] \text{enn2real-le top.not-eq-extremum sets-eq-imp-space-eq[OF } Mi'(1)] \ Mi'(1)$)

with h **show** *?thesis*

by(*simp add: emeasure-}Mmn*) (*metis divide-le-posI-ennreal dual-order.eq-iff ennreal-zero-divide mult.right-neutral not-gr-zero zero-le*)

qed(*auto simp: Mmn-def emeasure-sigma emeasure-notin-sets Mi'*)

have $Mi'\text{-sum} : Mi' \ m \ A = (\sum n. Mmn \ (m, \ n) \ A)$ **if** $A \in \text{sets } M$ **for** $m \ A$

proof –

have $(\sum n. Mmn \ (m, \ n) \ A) = (\sum n. Mmn \ (m, \ n + \text{nat } \lceil \text{enn2real } (u \ m) \rceil) \ A) + (\sum n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil. Mmn \ (m, \ n) \ A)$

by(*simp add: suminf-offset[where f= $\lambda n. Mmn \ (m, \ n) \ A$]*)

also have $\dots = (\sum n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil. Mmn \ (m, \ n) \ A)$

by(*simp add: emeasure-sigma Mmn-def*)

also have $\dots = (\sum n < \text{nat } \lceil \text{enn2real } (u \ m) \rceil. (Mi' \ m \ A) / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u \ m) \rceil))$

by(*rule Finite-Cartesian-Product.sum-cong-aux*) (*auto simp: emeasure-}Mmn that*)

also have $\dots = Mi' \ m \ A$

```

proof (cases nat [enn2real (u m)])
  case 0
  with u[of m] show ?thesis
    by simp (metis Mi'(1) emeasure-mono enn2real-positive-iff less-le-not-le
linorder-less-linear not-less-zero sets.sets-into-space sets.top that)
  next
  case (Suc n')
  then have ennreal (real-of-int [enn2real (u m)]) > 0
    using ennreal-less-zero-iff by fastforce
  with u(1)[of m] have of_nat (nat [enn2real (u m)]) / ennreal (real-of-int
[enn2real (u m)]) = 1
    by (simp add: ennreal-eq-0-iff ennreal-of-nat-eq-real-of-nat)
  thus ?thesis
    by (simp add: ennreal-divide-times[symmetric])
  qed
  finally show ?thesis ..
qed
define Mi where Mi  $\equiv$  ( $\lambda i.$  Mmn (prod-decode i))
show  $\exists Mi. (\forall i. \text{sets } (Mi\ i) = \text{sets } M) \wedge (\forall i. \text{emeasure } (Mi\ i) (\text{space } M) \leq 1)$ 
 $\wedge (\forall A \in \text{sets } M. M\ A = (\sum i. (Mi\ i)\ A))$ 
  by (auto intro!: exI[where x=Mi] simp: Mi-def sets-Mmn suminf-ennreal-2dimen[OF
Mi'-sum] Mi'(3)) (metis emeasure-Mmn-less1 prod-decode-aux.cases)
qed

```

```

lemma(in s-finite-measure) finite-measures:
  obtains Mi where  $\bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \wedge i. (Mi\ i) (\text{space } M) \leq 1 \wedge A. M$ 
 $A = (\sum i. Mi\ i\ A)$ 
proof -
  obtain Mi where  $Mi: \bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \wedge i. (Mi\ i) (\text{space } M) \leq 1 \wedge A.$ 
 $A \in \text{sets } M \implies M\ A = (\sum i. Mi\ i\ A)$ 
    using s-finite-measure-axioms by (metis s-finite-measure-subprob-space)
  hence  $M\ A = (\sum i. Mi\ i\ A)$  for A
    by (cases A  $\in$  sets M) (auto simp: emeasure-notin-sets)
  with Mi(1,2) show ?thesis
    using that by blast
qed

```

```

lemma(in s-finite-measure) finite-measures-ne:
  assumes space M  $\neq$  {}
  obtains Mi where  $\bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \wedge i. \text{subprob-space } (Mi\ i) \wedge A. M$ 
 $A = (\sum i. Mi\ i\ A)$ 
  by (metis assms finite-measures sets-eq-imp-space-eq subprob-spaceI)

```

```

lemma(in s-finite-measure) finite-measures':
  obtains Mi where  $\bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \wedge i. \text{finite-measure } (Mi\ i) \wedge A. M$ 
 $A = (\sum i. Mi\ i\ A)$ 
  by (metis ennreal-top-neq-one finite-measureI finite-measures infinity-ennreal-def
sets-eq-imp-space-eq top.extremum-uniqueI)

```

lemma(in *s-finite-measure*) *s-finite-measure-distr*:
assumes $f[\text{measurable}] : f \in M \rightarrow_M N$
shows *s-finite-measure* (*distr* $M N f$)
proof –
obtain M_i **where** $M_i[\text{measurable-cong}] : \bigwedge i. \text{sets } (M_i i) = \text{sets } M \bigwedge i. \text{finite-measure } (M_i i) \bigwedge A. M A = (\sum i. M_i i A)$
by(*metis finite-measures'*)
show *?thesis*
by(*auto intro!*: *s-finite-measureI*[**where** $M_i = (\lambda i. \text{distr } (M_i i) N f)$] *finite-measure.finite-measure-distr*[*OF* $M_i(2)$] *simp*: *emeasure-distr* $M_i(3)$ *sets-eq-imp-space-eq*[*OF* $M_i(1)$])
qed

lemma *nn-integral-measure-suminf*:
assumes [*measurable-cong*] : $\bigwedge i. \text{sets } (M_i i) = \text{sets } M$ **and** $\bigwedge A. A \in \text{sets } M \implies M A = (\sum i. M_i i A)$ $f \in \text{borel-measurable } M$
shows $(\sum i. \int^+ x. f x \partial(M_i i)) = (\int^+ x. f x \partial M)$
using *assms(3)*
proof *induction*
case (*cong* $f g$)
then show *?case*
by (*metis* (*no-types*, *lifting*) *assms(1)* *nn-integral-cong* *sets-eq-imp-space-eq* *suminf-cong*)
next
case (*set* A)
then show *?case*
using *assms(1,2)* **by** *simp*
next
case (*mult* $u c$)
then show *?case*
by(*simp add*: *nn-integral-cmult*)
next
case (*add* $u v$)
then show *?case*
by(*simp add*: *nn-integral-add* *suminf-add*[*symmetric*])
next
case *ih*:(*seq* U)
have $(\sum i. \text{integral}^N (M_i i) (\bigsqcup \text{range } U)) = (\sum i. \int^+ x. (\bigsqcup j. U j x) \partial(M_i i))$
by(*auto intro!*: *suminf-cong*) (*metis* *SUP-apply*)
also have $\dots = (\sum i. \bigsqcup j. \int^+ x. U j x \partial(M_i i))$
using *ih* **by**(*auto intro!*: *suminf-cong* *nn-integral-monotone-convergence-SUP*)
also have $\dots = (\bigsqcup j. (\sum i. \int^+ x. U j x \partial(M_i i)))$
using *ih(3)* **by**(*auto intro!*: *ennreal-suminf-SUP-eq* *incseq-nn-integral*)
also have $\dots = (\bigsqcup j. \text{integral}^N M (U j))$
by(*simp add*: *ih*)
also have $\dots = (\int^+ x. (\bigsqcup j. U j x) \partial M)$
using *ih* **by**(*auto intro!*: *nn-integral-monotone-convergence-SUP*[*symmetric*])
also have $\dots = \text{integral}^N M (\bigsqcup \text{range } U)$
by(*metis* *SUP-apply*)
finally show *?case* .

qed

A density $M f$ of s -finite measure M and $f \in \text{borel-measurable } M$ is again s -finite. We do not require additional assumption, unlike σ -finite measures.

lemma(in s -finite-measure) s -finite-measure-density:

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

shows s -finite-measure (density $M f$)

proof –

obtain M_i **where** $M_i[\text{measurable-cong}] : \bigwedge i. \text{sets } (M_i i) = \text{sets } M \bigwedge i. \text{finite-measure } (M_i i) \bigwedge A. M A = (\sum i. M_i i A)$

by(metis finite-measures[^])

show ?thesis

proof(rule s -finite-measure- s -finite-sumI[**where** $M_i = \lambda i. \text{density } (M_i i) f$])

show s -finite-measure (density $(M_i i) f$) **for** i

proof –

define M_{ij} **where** $M_{ij} = (\lambda j :: \text{nat}. \text{if } j = 0 \text{ then density } (M_i i) (\lambda x. \infty * \text{indicator } \{x \in \text{space } M. f x = \infty\} x)$

$\text{else if } j = 1 \text{ then density } (M_i i) (\lambda x. f x * \text{indicator } \{x \in \text{space } M. f x < \infty\} x)$

$\text{else null-measure } M)$

have $\text{sets-}M_{ij}[\text{measurable-cong}] : \text{sets } (M_{ij} j) = \text{sets } M$ **for** j

by(auto simp: M_{ij} -def M_i)

have $\text{emeasure-}M_i : \text{density } (M_i i) f A = (\sum j. M_{ij} j A)$ (**is** ?lhs = ?rhs) **if** $A[\text{measurable}] : A \in \text{sets } M$ **for** A

proof –

have ?lhs = $(\int^+ x \in A. f x \partial M_i i)$

by(simp add: emeasure-density)

also have ... = $(\int^+ x. \infty * \text{indicator } \{x \in \text{space } M. f x = \infty\} x * \text{indicator } A x + f x * \text{indicator } \{x \in \text{space } M. f x < \infty\} x * \text{indicator } A x \partial M_i i)$

by(auto intro!: nn-integral-cong simp: sets-eq-imp-space-eq[OF $M_i(1)$] indicator-def) (simp add: top.not-eq-extremum)

also have ... = $\text{density } (M_i i) (\lambda x. \infty * \text{indicator } \{x \in \text{space } M. f x = \infty\} x) A + \text{density } (M_i i) (\lambda x. f x * \text{indicator } \{x \in \text{space } M. f x < \infty\} x) A$

by(simp add: nn-integral-add emeasure-density)

also have ... = ?rhs

using suminf-finite [of $\{.. < \text{Suc } (\text{Suc } 0)\} \lambda j. M_{ij} j A$] **by**(auto simp: M_{ij} -def)

finally show ?thesis .

qed

show ?thesis

proof(rule s -finite-measure- s -finite-sumI[OF - - $\text{emeasure-}M_i$])

fix $j :: \text{nat}$

consider $j = 0 \mid j = 1 \mid j \neq 0 \mid j \neq 1$ **by** auto

then show s -finite-measure ($M_{ij} j$)

proof cases

case 1

have $2 : M_{ij} j A = (\sum k. \text{density } (M_i i) (\text{indicator } \{x \in \text{space } M. f x = \top\}) A)$ **if** $A[\text{measurable}] : A \in \text{sets } M$ **for** A

by(auto simp: M_{ij} -def 1 emeasure-density nn-integral-suminf[symmetric] sets-eq-imp-space-eq[OF $M_i(1)$] indicator-def intro!: nn-integral-cong) (simp add:

```

nn-integral-count-space-nat[symmetric])
  show ?thesis
    by(auto simp: s-finite-measure-def 2 Mi(1)[of i] sets-Mij[of j] in-
intro!: exI[where x=λk. density (Mi i) (indicator {x∈space M. f x = ∞})] fi-
nite-measure.finite-measure-restricted Mi(2))
  next
    case 2
    show ?thesis
    by(auto intro!: sigma-finite-measure.s-finite-measure AE-mono-measure[OF
Mi(1)] sum-le-suminf[where I={i} and f=λi. Mi i -,simplified] simp: sigma-finite-measure.sigma-finite-iff-de-
finite-measure.sigma-finite-measure[OF Mi(2)[of i]] le-measure[OF Mi(1)] Mi in-
dicator-def 2 Mij-def) auto
  next
    case 3
    then show ?thesis
    by(auto simp: Mij-def intro!: finite-measure.s-finite-measure-finite-measure
finite-measureI)
  qed
qed(auto simp: sets-Mij Mi)
qed
qed(auto simp: emeasure-density nn-integral-measure-suminf[OF Mi(1,3)] Mi(1))
qed

```

lemma

```

fixes f :: 'a ⇒ 'b::{banach, second-countable-topology}
assumes [measurable-cong]:∧i. sets (Mi i) = sets M and ∧A. A∈sets M ⇒ M
A = (∑ i. Mi i A) integrable M f
shows lebesgue-integral-measure-suminf:(∑ i. ∫ x. f x ∂(Mi i)) = (∫ x. f x ∂M)
(is ?suminf)
and lebesgue-integral-measure-suminf-summable-norm: summable (λi. norm
(∫ x. f x ∂(Mi i))) (is ?summable2)
and lebesgue-integral-measure-suminf-summable-norm-in: summable (λi. ∫ x.
norm (f x) ∂(Mi i)) (is ?summable)
proof -
  have Mi:Mi i ≤ M for i
  using assms(2) ennreal-suminf-lessD linorder-not-le by(fastforce simp: assms(1))
le-measure[OF assms(1)])
  have sum2: summable (λi. norm (∫ x. g x ∂(Mi i))) if summable (λi. ∫ x. norm
(g x) ∂(Mi i)) for g :: 'a ⇒ 'b
  proof(rule summable-suminf-not-top)
    have (∑ i. ennreal (norm (∫ x. g x ∂(Mi i)))) ≤ (∑ i. ennreal (∫ x. norm (g
x) ∂(Mi i)))
    by(auto intro!: suminf-le)
    thus (∑ i. ennreal (norm (∫ x. g x ∂(Mi i)))) ≠ ⊤
    by (metis ennreal-suminf-neq-top[OF that] Bochner-Integration.integral-nonneg
neq-top-trans norm-ge-zero)
  qed simp
  have ?suminf ∧ ?summable
  using assms(3)

```

```

proof induction
  case  $h$ [measurable]:(base  $A$   $c$ )
  have  $Mi$ -fin: $Mi$   $i$   $A < \infty$  for  $i$ 
    by(rule order.strict-trans1[ $OF$  -  $h(2)$ ], auto simp: le-measureD3[ $OF$   $Mi$ 
  assms(1)])
  have 1: ( $\int x$ . (indicat-real  $A$   $x *_R c$ )  $\partial Mi$   $i$ ) = measure ( $Mi$   $i$ )  $A *_R c$  for  $i$ 
    using  $Mi$ -fin by simp
  have 2:summable ( $\lambda i$ .  $\int x$ . norm (indicat-real  $A$   $x *_R c$ )  $\partial Mi$   $i$ )
  proof(rule summable-suminf-not-top)
    show ( $\sum i$ . ennreal ( $\int x$ . norm (indicat-real  $A$   $x *_R c$ )  $\partial Mi$   $i$ ))  $\neq \top$  (is  $?l \neq$ 
  -)
  proof -
    have  $?l = (\sum i$ .  $Mi$   $i$   $A$ ) * norm  $c$ 
      using  $Mi$ -fin by(auto intro!: suminf-cong simp: measure-def enn2real-mult
  ennreal-mult)
    also have ... =  $M$   $A$  * norm  $c$ 
      by(simp add: assms(2))
    also have ...  $\neq \top$ 
      using  $h(2)$  by (simp add: ennreal-mult-less-top top.not-eq-extremum)
    finally show ?thesis .
  qed
qed simp
  have 3: ( $\sum i$ .  $\int x$ . indicat-real  $A$   $x *_R c$   $\partial Mi$   $i$ ) = ( $\int x$ . indicat-real  $A$   $x *_R c$ 
   $\partial M$ ) (is  $?l = ?r$ )
  proof -
    have [simp]: summable ( $\lambda i$ . enn2real ( $Mi$   $i$   $A$ ))
    using  $Mi$ -fin  $h$  by(auto intro!: summable-suminf-not-top simp: assms(2)[symmetric])
    have  $?l = (\sum i$ . measure ( $Mi$   $i$ )  $A$ ) *  $_R c$ 
      by(auto intro!: suminf-cong simp: 1 measure-def suminf-scaleR-left)
    also have ... =  $?r$ 
      using  $h(2)$   $Mi$ -fin by(simp add: ennreal-inj[where  $a=(\sum i$ . measure ( $Mi$   $i$ )
   $A$ ) and  $b=$ measure  $M$   $A$ , $OF$  suminf-nonneg measure-nonneg,symmetric,simplified
  measure-def] measure-def suminf-ennreal2[symmetric] assms(2)[symmetric])
    finally show ?thesis .
  qed
from 2 3 show ?case by simp
next
  case  $ih$ [measurable]:(add  $f$   $g$ )
  have 1:summable ( $\lambda i$ .  $\int x$ . norm ( $f$   $x$  +  $g$   $x$ )  $\partial Mi$   $i$ )
  proof(rule summable-suminf-not-top)
    show ( $\sum i$ . ennreal ( $\int x$ . norm ( $f$   $x$  +  $g$   $x$ )  $\partial Mi$   $i$ ))  $\neq \top$  (is  $?l \neq$  -)
  proof -
    have  $?l = (\sum i$ . ( $\int$   $^+x$ . ennreal (norm ( $f$   $x$  +  $g$   $x$ ))  $\partial Mi$   $i$ ))
      using  $ih$  by(auto intro!: suminf-cong nn-integral-eq-integral[symmetric]
  integrable-mono-measure[ $OF$  assms(1)  $Mi$ ])
    also have ...  $\leq (\sum i$ . ( $\int$   $^+x$ . ennreal (norm ( $f$   $x$ ) + norm ( $g$   $x$ ))  $\partial Mi$   $i$ ))
      by(auto intro!: suminf-le nn-integral-mono norm-triangle-ineq simp del:
  ennreal-plus)
    also have ... = ( $\sum i$ . ( $\int$   $^+x$ . ennreal (norm ( $f$   $x$ ))  $\partial Mi$   $i$ )) + ( $\sum i$ . ( $\int$   $^+x$ .

```

```

ennreal (norm (g x) ∂Mi i))
  by(auto intro!: suminf-cong simp: nn-integral-add suminf-add)
  also have ... = (∑ i. ennreal (∫ x. norm (f x) ∂Mi i)) + (∑ i. ennreal (∫ x.
norm (g x) ∂Mi i))
  using ih by(simp add: nn-integral-eq-integral integrable-mono-measure[OF
assms(1) Mi])
  also have ... < ⊤
  using ennreal-suminf-neq-top[OF conjunct2[OF ih(3)]] ennreal-suminf-neq-top[OF
conjunct2[OF ih(4)]]
  by (meson Bochner-Integration.integral-nonneg ennreal-add-eq-top norm-ge-zero
top.not-eq-extremum)
  finally show ?thesis
    using order.strict-iff-order by blast
qed
qed simp
with ih show ?case
  by(auto simp: Bochner-Integration.integral-add[OF integrable-mono-measure[OF
assms(1) Mi ih(1)]] integrable-mono-measure[OF assms(1) Mi ih(2)]] suminf-add[symmetric,OF
summable-norm-cancel[OF sum2[OF conjunct2[OF ih(3)]]] summable-norm-cancel[OF
sum2[OF conjunct2[OF ih(4)]]]])
next
  case ih[measurable]:(lim f fn)
  have 1:summable (λi. ∫ x. norm (f x) ∂(Mi i))
  proof(rule summable-suminf-not-top)
    show (∑ i. ennreal (∫ x. norm (f x) ∂(Mi i))) ≠ ⊤ (is ?lhs ≠ -)
  proof -
    have ?lhs = (∑ i. ∫+ x. ennreal (norm (f x) ∂Mi i))
    by(auto intro!: suminf-cong nn-integral-eq-integral[symmetric] integrable-mono-measure[OF
assms(1) Mi] simp: ih)
    also have ... = (∫+ x. ennreal (norm (f x) ∂M))
    by(simp add: nn-integral-measure-suminf[OF assms(1,2)])
    also have ... = ennreal (∫ x. norm (f x) ∂M)
    by(auto intro!: nn-integral-eq-integral ih(4))
    also have ... < ⊤ by simp
    finally show ?lhs ≠ ⊤
      using linorder-neq-iff by blast
  qed
qed simp
have (∑ i. ∫ x. f x ∂(Mi i)) = (∫ i. ∫ x. f x ∂(Mi i) ∂(count-space UNIV))
  by(rule integral-count-space-nat[symmetric]) (simp add: integrable-count-space-nat-iff
sum2[OF 1])
  also have ... = lim (λm. ∫ i. ∫ x. fn m x ∂(Mi i) ∂(count-space UNIV))
  proof(rule limI[OF integral-dominated-convergence[where w=λi. 2 * (∫ x.
norm (f x) ∂(Mi i)),symmetric],auto simp: AE-count-space integrable-count-space-nat-iff
1])
    show (λm. ∫ x. fn m x ∂(Mi i)) ⟶ ∫ x. f x ∂(Mi i) for i
    by(rule integral-dominated-convergence[where w=λx. 2 * norm (f x)],insert
ih) (auto intro!: integrable-mono-measure[OF assms(1) Mi] simp: sets-eq-imp-space-eq[OF
assms(1)])

```

```

next
  fix i j
  show norm (∫ x. fn j x ∂(Mi i)) ≤ 2 * (∫ x. norm (f x) ∂(Mi i)) (is ?l ≤ ?r)
  proof -
    have ?l ≤ (∫ x. norm (fn j x) ∂(Mi i))
      by simp
    also have ... ≤ (∫ x. 2 * norm (f x) ∂(Mi i))
      by(rule integral-mono,insert ih) (auto intro!: integrable-mono-measure[OF
assms(1) Mi] simp: sets-eq-imp-space-eq[OF assms(1)])
    finally show ?l ≤ ?r by simp
  qed
  qed
  also have ... = lim (λm. (∑ i. ∫ x. fn m x ∂(Mi i)))
  proof -
    have (∫ i. ∫ x. fn m x ∂(Mi i) ∂(count-space UNIV)) = (∑ i. ∫ x. fn m x
∂(Mi i)) for m
      by(auto intro!: integral-count-space-nat sum2 simp: integrable-count-space-nat-iff)
      (use ih(5) in auto)
    thus ?thesis by simp
  qed
  also have ... = lim (λm. ∫ x. fn m x ∂M)
    by(simp add: ih(5))
  also have ... = (∫ x. f x ∂M)
    using ih by(auto intro!: limI[OF integral-dominated-convergence[where
w=λx. 2 * norm (f x)])])
  finally show ?case
    using 1 by auto
  qed
  thus ?suminf ?summable ?summable2
    by(simp-all add: sum2)
  qed

```

```

lemma (in s-finite-measure) measurable-emeasure-Pair':
  assumes Q ∈ sets (N ⊗M M)
  shows (λx. emeasure M (Pair x -' Q)) ∈ borel-measurable N (is ?s Q ∈ -)
  proof -
    obtain Mi where Mi: ∧ i. sets (Mi i) = sets M ∧ i. finite-measure (Mi i) ∧ A.
M A = (∑ i. Mi i A)
      by(metis finite-measures')
    show ?thesis
      using Mi(1,2) assms finite-measure.finite-measure-cut-measurable[of Mi - Q N]
      by(simp add: Mi(3))
  qed

```

```

lemma (in s-finite-measure) measurable-emeasure'[measurable (raw)]:
  assumes space: ∧ x. x ∈ space N ⇒ A x ⊆ space M
  assumes A: {x ∈ space (N ⊗M M). snd x ∈ A (fst x)} ∈ sets (N ⊗M M)
  shows (λx. emeasure M (A x)) ∈ borel-measurable N

```


proof –
from *space* **have** $\bigwedge x. x \in \text{space } N \implies \text{Pair } x - \{x \in \text{space } (N \otimes_M M)\}. \text{snd}$
 $x \in A (\text{fst } x) = A x$
by (*auto simp: space-pair-measure*)
with *measurable-emeasure-Pair'*[*OF A*] **show** *?thesis*
by (*auto cong: measurable-cong*)
qed

lemma(*in s-finite-measure*) *emeasure-pair-measure'*:
assumes $X \in \text{sets } (N \otimes_M M)$
shows $\text{emeasure } (N \otimes_M M) X = (\int^+ x. \int^+ y. \text{indicator } X (x, y) \partial M \partial N)$
(*is - = ? μ X*)
proof (*rule emeasure-measure-of*[*OF pair-measure-def*])
show *positive* ($\text{sets } (N \otimes_M M)$) *? μ*
by (*auto simp: positive-def*)
have *eq[simp]*: $\bigwedge A x y. \text{indicator } A (x, y) = \text{indicator } (\text{Pair } x - A) y$
by (*auto simp: indicator-def*)
show *countably-additive* ($\text{sets } (N \otimes_M M)$) *? μ*
proof (*rule countably-additiveI*)
fix $F :: \text{nat} \Rightarrow ('b \times 'a) \text{ set}$ **assume** $F: \text{range } F \subseteq \text{sets } (N \otimes_M M)$ *dis-*
joint-family F
from F **have** $*$: $\bigwedge i. F i \in \text{sets } (N \otimes_M M)$ **by** *auto*
moreover **have** $\bigwedge x. \text{disjoint-family } (\lambda i. \text{Pair } x - F i)$
by (*intro disjoint-family-on-bisimulation*[*OF F(2)*]) *auto*
moreover **have** $\bigwedge x. \text{range } (\lambda i. \text{Pair } x - F i) \subseteq \text{sets } M$
using F **by** (*auto simp: sets-Pair1*)
ultimately **show** $(\sum n. ?\mu (F n)) = ?\mu (\bigcup i. F i)$
by (*auto simp add: nn-integral-suminf[symmetric] vimage-UN suminf-emeasure*
intro!: nn-integral-cong nn-integral-indicator[symmetric])
qed
show $\{a \times b \mid a \in \text{sets } N \wedge b \in \text{sets } M\} \subseteq \text{Pow } (\text{space } N \times \text{space } M)$
using *sets.space-closed*[*of N*] *sets.space-closed*[*of M*] **by** *auto*
qed fact

lemma (*in s-finite-measure*) *emeasure-pair-measure-alt'*:
assumes $X: X \in \text{sets } (N \otimes_M M)$
shows $\text{emeasure } (N \otimes_M M) X = (\int^+ x. \text{emeasure } M (\text{Pair } x - X) \partial N)$
proof –
have [*simp*]: $\bigwedge x y. \text{indicator } X (x, y) = \text{indicator } (\text{Pair } x - X) y$
by (*auto simp: indicator-def*)
show *?thesis*
using X **by** (*auto intro!: nn-integral-cong simp: emeasure-pair-measure' sets-Pair1*)
qed

proposition (*in s-finite-measure*) *emeasure-pair-measure-Times'*:
assumes $A: A \in \text{sets } N$ **and** $B: B \in \text{sets } M$
shows $\text{emeasure } (N \otimes_M M) (A \times B) = \text{emeasure } N A * \text{emeasure } M B$
proof –

have $\text{emeasure } (N \otimes_M M) (A \times B) = (\int^{+x}. \text{emeasure } M B * \text{indicator } A x \partial N)$
using $A B$ **by** (*auto intro!*: *nn-integral-cong simp: emeasure-pair-measure-alt'*)
also have $\dots = \text{emeasure } M B * \text{emeasure } N A$
using A **by** (*simp add: nn-integral-cmult-indicator*)
finally show *?thesis*
by (*simp add: ac-simps*)
qed

lemma(*in s-finite-measure*) *measure-times*:
assumes[*measurable*]: $A \in \text{sets } N B \in \text{sets } M$
shows $\text{measure } (N \otimes_M M) (A \times B) = \text{measure } N A * \text{measure } M B$
by(*auto simp: measure-def emeasure-pair-measure-Times' enn2real-mult*)

lemma *pair-measure-s-finite-measure-suminf*:
assumes Mi [*measurable-cong*]: $\bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \bigwedge i. \text{finite-measure } (Mi\ i) \bigwedge A. M A = (\sum i. Mi\ i\ A)$
and Ni [*measurable-cong*]: $\bigwedge i. \text{sets } (Ni\ i) = \text{sets } N \bigwedge i. \text{finite-measure } (Ni\ i) \bigwedge A. N A = (\sum i. Ni\ i\ A)$
shows $(M \otimes_M N) A = (\sum i\ j. (Mi\ i \otimes_M Ni\ j) A)$ (**is** *?lhs = ?rhs*)

proof –
interpret N : *s-finite-measure* N
by(*auto intro!*: *s-finite-measureI*[**where** $Mi=Mi$] *s-finite-measureI*[**where** $Mi=Ni$]
assms)

show *?thesis*

proof(*cases* $A \in \text{sets } (M \otimes_M N)$)

case [*measurable*]: *True*

show *?thesis*

proof –

have $?lhs = (\int^{+x}. N (Pair\ x - 'A) \partial M)$

by(*simp add: N.emeasure-pair-measure-alt'*)

also have $\dots = (\sum i. \int^{+x}. N (Pair\ x - 'A) \partial Mi\ i)$

using N .*measurable-emeasure-Pair'*[*of* A]

by(*auto intro!*: *nn-integral-measure-suminf*[*OF* $Mi(1,3)$,*symmetric*])

also have $\dots = (\sum i. \int^{+x}. (\sum j. Ni\ j (Pair\ x - 'A)) \partial Mi\ i)$

by(*simp add: Ni(3)*)

also have $\dots = (\sum i\ j. \int^{+x}. Ni\ j (Pair\ x - 'A) \partial Mi\ i)$

using *s-finite-measure.measurable-emeasure-Pair'*[*OF* *finite-measure.s-finite-measure-finite-measure*[*OF* $Ni(2)$],*of* A]

by(*auto simp: nn-integral-suminf intro!*: *suminf-cong*)

also have $\dots = ?rhs$

by(*auto intro!*: *suminf-cong simp: s-finite-measure.emeasure-pair-measure-alt'*[*OF* *finite-measure.s-finite-measure-finite-measure*[*OF* $Ni(2)$]])

finally show *?thesis* .

qed

next

case *False*

with $Mi(1)$ $Ni(1)$ **show** *?thesis*

by(*simp add: emeasure-notin-sets*)

qed
qed

lemma *pair-measure-s-finite-measure-suminf'*:

assumes Mi [measurable-cong]: $\bigwedge i. sets (Mi i) = sets M \wedge i. finite-measure (Mi i) \wedge A. M A = (\sum i. Mi i A)$

and Ni [measurable-cong]: $\bigwedge i. sets (Ni i) = sets N \wedge i. finite-measure (Ni i) \wedge A. N A = (\sum i. Ni i A)$

shows $(M \otimes_M N) A = (\sum i j. (Mi j \otimes_M Ni i) A)$ (is ?lhs = ?rhs)

proof –

interpret N : *s-finite-measure* N

by(*auto intro!*: *s-finite-measureI*[**where** $Mi=Mi$] *s-finite-measureI*[**where** $Mi=Ni$] *assms*)

show ?thesis

proof(*cases* $A \in sets (M \otimes_M N)$)

case [measurable]:*True*

show ?thesis

proof –

have ?lhs = $(\int^+ x. N (Pair x -' A) \partial M)$

by(*simp add*: *N.emeasure-pair-measure-alt'*)

also have ... = $(\int^+ x. (\sum i. Ni i (Pair x -' A)) \partial M)$

by(*auto intro!*: *nn-integral-cong simp*: Ni)

also have ... = $(\sum i. (\int^+ x. Ni i (Pair x -' A) \partial M))$

by(*auto intro!*: *nn-integral-suminf simp*: *finite-measure.finite-measure-cut-measurable*[*OF* $Ni(2)$])

also have ... = $(\sum i j. \int^+ x. Ni i (Pair x -' A) \partial Mi j)$

by(*auto intro!*: *suminf-cong nn-integral-measure-suminf*[*symmetric*] *simp*: *finite-measure.finite-measure-cut-measurable*[*OF* $Ni(2)$] Mi)

also have ... = ?rhs

by(*auto intro!*: *suminf-cong simp*: *s-finite-measure.emeasure-pair-measure-alt'*[*OF* *finite-measure.s-finite-measure-finite-measure*[*OF* $Ni(2)$]])

finally show ?thesis .

qed

next

case *False*

with $Mi(1)$ $Ni(1)$ **show** ?thesis

by(*simp add*: *emeasure-notin-sets*)

qed

qed

lemma *pair-measure-s-finite-measure*:

assumes *s-finite-measure* M **and** *s-finite-measure* N

shows *s-finite-measure* $(M \otimes_M N)$

proof –

obtain Mi **where** Mi [measurable-cong]: $\bigwedge i. sets (Mi i) = sets M \wedge i. finite-measure (Mi i) \wedge A. M A = (\sum i. Mi i A)$

by(*metis s-finite-measure.finite-measures'*[*OF* *assms*(1)])

obtain Ni **where** Ni [measurable-cong]: $\bigwedge i. sets (Ni i) = sets N \wedge i. finite-measure (Ni i) \wedge A. N A = (\sum i. Ni i A)$

by(metis s-finite-measure.finite-measures'[OF assms(2)])
show ?thesis
proof(rule s-finite-measure-prodI[**where** $M_{ij} = \lambda i j. M_i i \otimes_M N_i j$])
show emeasure $(M_i i \otimes_M N_i j)$ (space $(M \otimes_M N)$) $< \infty$ **for** $i j$
using finite-measure.emeasure-finite[OF $M_i(2)$][of i] finite-measure.emeasure-finite[OF $N_i(2)$][of j]]
by(auto simp: sets-eq-imp-space-eq[OF $M_i(1)$][of i],symmetric] sets-eq-imp-space-eq[OF $N_i(1)$][of j],symmetric] space-pair-measure s-finite-measure.emeasure-pair-measure-Times'[OF finite-measure.s-finite-measure-finite-measure[OF $N_i(2)$][of j]] ennreal-mult-less-top top.not-eq-extremum)
qed(auto simp: pair-measure-s-finite-measure-suminf $M_i N_i$)
qed

lemma(in s-finite-measure) borel-measurable-nn-integral-fst':
assumes [measurable]: $f \in \text{borel-measurable } (N \otimes_M M)$
shows $(\lambda x. \int^+ y. f(x, y) \partial M) \in \text{borel-measurable } N$
proof –
obtain M_i **where** $M_i[\text{measurable-cong}]: \bigwedge i. \text{sets } (M_i i) = \text{sets } M \bigwedge i. \text{finite-measure } (M_i i) \bigwedge A. M A = (\sum i. M_i i A)$
by(metis finite-measures^)
show ?thesis
by(rule measurable-cong[**where** $g = \lambda x. \sum i. \int^+ y. f(x, y) \partial M_i i$, THEN iffD2])
(auto simp: nn-integral-measure-suminf[OF $M_i(1,3)$] intro!: borel-measurable-suminf-order sigma-finite-measure.borel-measurable-nn-integral-fst[OF finite-measure.sigma-finite-measure[OF $M_i(2)$]])
qed

lemma (in s-finite-measure) nn-integral-fst':
assumes $f: f \in \text{borel-measurable } (M_1 \otimes_M M)$
shows $(\int^+ x. \int^+ y. f(x, y) \partial M \partial M_1) = \text{integral}^N (M_1 \otimes_M M) f$ (is ?I f = -)
using f **proof** induct
case (cong u v)
then have ?I u = ?I v
by (intro nn-integral-cong) (auto simp: space-pair-measure)
with cong **show** ?case
by (simp cong: nn-integral-cong)
qed (simp-all add: emeasure-pair-measure' nn-integral-cmult nn-integral-add nn-integral-monotone-convergence-SUP measurable-compose-Pair1 borel-measurable-nn-integral-fst' nn-integral-mono incseq-def le-fun-def image-comp cong: nn-integral-cong)

lemma (in s-finite-measure) borel-measurable-nn-integral'[measurable (raw)]:
case-prod $f \in \text{borel-measurable } (N \otimes_M M) \implies (\lambda x. \int^+ y. f x y \partial M) \in \text{borel-measurable } N$
using borel-measurable-nn-integral-fst'[of case-prod $f N$] **by** simp

lemma distr-pair-swap-s-finite:

assumes *s-finite-measure* $M1$ **and** *s-finite-measure* $M2$
shows $M1 \otimes_M M2 = \text{distr } (M2 \otimes_M M1) (M1 \otimes_M M2) (\lambda(x, y). (y, x))$ (**is**
 $?P = ?D$)
proof –
{
from *s-finite-measure.finite-measures'*[*OF assms(1)*] *s-finite-measure.finite-measures'*[*OF*
assms(2)]
obtain $Mi1\ Mi2$
where $Mi1:\wedge i. \text{sets } (Mi1\ i) = \text{sets } M1 \wedge i. \text{finite-measure } (Mi1\ i) \wedge A. M1$
 $A = (\sum i. Mi1\ i\ A)$
and $Mi2:\wedge i. \text{sets } (Mi2\ i) = \text{sets } M2 \wedge i. \text{finite-measure } (Mi2\ i) \wedge A. M2\ A$
 $= (\sum i. Mi2\ i\ A)$
by *metis*
fix A
assume $A[\text{measurable}]: A \in \text{sets } (M1 \otimes_M M2)$
have $\text{emeasure } (M1 \otimes_M M2)\ A = \text{emeasure } (M2 \otimes_M M1) ((\lambda(x, y). (y, x)))$
– ‘ $A \cap \text{space } (M2 \otimes_M M1)$)
proof –
{
fix $i\ j$
interpret *pair-sigma-finite* $Mi1\ i\ Mi2\ j$
by(*auto simp: pair-sigma-finite-def* $Mi1(2)\ Mi2(2)$ *finite-measure.sigma-finite-measure*)
have $\text{emeasure } (Mi1\ i \otimes_M Mi2\ j)\ A = \text{emeasure } (Mi2\ j \otimes_M Mi1\ i) ((\lambda(x,$
 $y). (y, x)) - 'A \cap \text{space } (M2 \otimes_M M1))$
using $Mi1(1)\ Mi2(1)$ **by**(*simp add: arg-cong[OF distr-pair-swap, of emea-*
sure] emeasure-distr sets-eq-imp-space-eq[OF sets-pair-measure-cong[OF Mi2(1) Mi1(1)]])
}
thus *?thesis*
by(*auto simp: pair-measure-s-finite-measure-suminf'[OF Mi2 Mi1] pair-measure-s-finite-measure-suminf[*
 $Mi1\ Mi2]$ *intro!: suminf-cong*)
qed
}
thus *?thesis*
by(*auto intro!: measure-eqI simp: emeasure-distr*)
qed

proposition *nn-integral-snd'*:

assumes *s-finite-measure* $M1$ *s-finite-measure* $M2$
and $f[\text{measurable}]: f \in \text{borel-measurable } (M1 \otimes_M M2)$
shows $(\int^+ y. (\int^+ x. f(x, y) \partial M1) \partial M2) = \text{integral}^N (M1 \otimes_M M2) f$
proof –
interpret $M1: s\text{-finite-measure } M1$ **by** *fact*
interpret $M2: s\text{-finite-measure } M2$ **by** *fact*
note *measurable-pair-swap[OF f]*
from $M1.\text{nn-integral-fst}'[OF\ \text{this}]$
have $(\int^+ y. (\int^+ x. f(x, y) \partial M1) \partial M2) = (\int^+ (x, y). f(y, x) \partial(M2 \otimes_M$
 $M1))$
by *simp*
also have $(\int^+ (x, y). f(y, x) \partial(M2 \otimes_M M1)) = \text{integral}^N (M1 \otimes_M M2) f$

by (subst distr-pair-swap-s-finite[OF assms(1,2)]) (auto simp add: nn-integral-distr intro!: nn-integral-cong)
 finally show ?thesis .
 qed

lemma (in *s-finite-measure*) *borel-measurable-lebesgue-integrable'*[*measurable (raw)*]:
 fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
 assumes [*measurable*]: case-prod $f \in \text{borel-measurable } (N \otimes_M M)$
 shows *Measurable.pred* N ($\lambda x. \text{integrable } M (f x)$)
proof –
 have [*simp*]: $\bigwedge x. x \in \text{space } N \implies \text{integrable } M (f x) \iff (\int^+ y. \text{norm } (f x y) \partial M) < \infty$
 unfolding *integrable-iff-bounded by simp*
 show ?thesis
 by (*simp cong: measurable-cong*)
 qed

lemma (in *s-finite-measure*) *measurable-measure'*[*measurable (raw)*]:
 $(\bigwedge x. x \in \text{space } N \implies A x \subseteq \text{space } M) \implies$
 $\{x \in \text{space } (N \otimes_M M). \text{snd } x \in A (\text{fst } x)\} \in \text{sets } (N \otimes_M M) \implies$
 $(\lambda x. \text{measure } M (A x)) \in \text{borel-measurable } N$
unfolding *measure-def by (intro measurable-emeasure' borel-measurable-enn2real)*
auto

proposition (in *s-finite-measure*) *borel-measurable-lebesgue-integral'*[*measurable (raw)*]:
 fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
 assumes $f[\text{measurable}]$: case-prod $f \in \text{borel-measurable } (N \otimes_M M)$
 shows $(\lambda x. \int y. f x y \partial M) \in \text{borel-measurable } N$
proof –
 from *borel-measurable-implies-sequence-metric*[*OF f, of 0*]
 obtain s where s : $\bigwedge i. \text{simple-function } (N \otimes_M M) (s i)$
 and $\forall x \in \text{space } (N \otimes_M M). (\lambda i. s i x) \longrightarrow (\text{case } x \text{ of } (x, y) \Rightarrow f x y)$
 and $\forall i. \forall x \in \text{space } (N \otimes_M M). \text{dist } (s i x) 0 \leq 2 * \text{dist } (\text{case } x \text{ of } (x, xa) \Rightarrow f x xa) 0$
 by *auto*
 then have *:
 $\bigwedge x y. x \in \text{space } N \implies y \in \text{space } M \implies (\lambda i. s i (x, y)) \longrightarrow f x y$
 $\bigwedge i x y. x \in \text{space } N \implies y \in \text{space } M \implies \text{norm } (s i (x, y)) \leq 2 * \text{norm } (f x y)$
 by (*auto simp: space-pair-measure*)
 have [*measurable*]: $\bigwedge i. s i \in \text{borel-measurable } (N \otimes_M M)$
 by (*rule borel-measurable-simple-function*) *fact*
 have $\bigwedge i. s i \in \text{measurable } (N \otimes_M M)$ (*count-space UNIV*)
 by (*rule measurable-simple-function*) *fact*
 define f' where [*abs-def*]: $f' i x =$
 (*if integrable* $M (f x)$ then *Bochner-Integration.simple-bochner-integral* $M (\lambda y. s i (x, y))$ else 0) **for** $i x$

```

{ fix i x assume x ∈ space N
  then have Bochner-Integration.simple-bochner-integral M (λy. s i (x, y)) =
    (∑ z ∈ s i ' (space N × space M). measure M {y ∈ space M. s i (x, y) = z}
  *R z)
  using s[THEN simple-functionD(1)]
  unfolding simple-bochner-integral-def
  by (intro sum.mono-neutral-cong-left)
    (auto simp: eq-commute space-pair-measure image-iff cong: conj-cong) }
note eq = this

show ?thesis
proof (rule borel-measurable-LIMSEQ-metric)
  fix i show f' i ∈ borel-measurable N
    unfolding f'-def by (simp-all add: eq cong: measurable-cong if-cong)
next
fix x assume x: x ∈ space N
{ assume int-f: integrable M (f x)
  have int-2f: integrable M (λy. 2 * norm (f x y))
    by (intro integrable-norm integrable-mult-right int-f)
  have (λi. integralL M (λy. s i (x, y))) → integralL M (f x)
  proof (rule integral-dominated-convergence)
    from int-f show f x ∈ borel-measurable M by auto
    show ∧i. (λy. s i (x, y)) ∈ borel-measurable M
      using x by simp
    show AE xa in M. (λi. s i (x, xa)) → f x xa
      using x * by auto
    show ∧i. AE xa in M. norm (s i (x, xa)) ≤ 2 * norm (f x xa)
      using x * by auto
  qed fact
moreover
{ fix i
  have Bochner-Integration.simple-bochner-integrable M (λy. s i (x, y))
  proof (rule simple-bochner-integrableI-bounded)
    have (λy. s i (x, y)) ' space M ⊆ s i ' (space N × space M)
      using x by auto
    then show simple-function M (λy. s i (x, y))
      using simple-functionD(1)[OF s(1), of i] x
      by (intro simple-function-borel-measurable)
        (auto simp: space-pair-measure dest: finite-subset)
    have (∫+ y. ennreal (norm (s i (x, y))) ∂M) ≤ (∫+ y. 2 * norm (f x y)
  ∂M)
      using x * by (intro nn-integral-mono) auto
    also have (∫+ y. 2 * norm (f x y) ∂M) < ∞
      using int-2f unfolding integrable-iff-bounded by simp
    finally show (∫+ xa. ennreal (norm (s i (x, xa))) ∂M) < ∞ .
  qed
  then have integralL M (λy. s i (x, y)) = Bochner-Integration.simple-bochner-integral
  M (λy. s i (x, y))

```

by (rule *simple-bochner-integrable-eq-integral[symmetric]*) }
 ultimately have ($\lambda i. \text{Bochner-Integration.simple-bochner-integral } M (\lambda y. s i$
 $(x, y)) \longrightarrow \text{integral}^L M (f x)$
 by *simp* }
 then
 show ($\lambda i. f' i x \longrightarrow \text{integral}^L M (f x)$
 unfolding *f'-def*
 by (cases *integrable M (f x)*) (*simp-all add: not-integrable-integral-eq*)
 qed
 qed

lemma *integrable-product-swap-s-finite*:
 fixes $f :: - \Rightarrow -::\{\text{banach, second-countable-topology}\}$
 assumes $M1:s\text{-finite-measure } M1$ and $M2:s\text{-finite-measure } M2$
 and *integrable* ($M1 \otimes_M M2$) f
 shows *integrable* ($M2 \otimes_M M1$) ($\lambda(x,y). f (y,x)$)
proof –
 have *: ($\lambda(x,y). f (y,x) = (\lambda x. f (\text{case } x \text{ of } (x,y) \Rightarrow (y,x)))$) by (auto *simp:*
fun-eq-iff)
 show ?thesis unfolding *
 by (rule *integrable-distr[OF measurable-pair-swap']*)
 (*simp add: distr-pair-swap-s-finite[OF M1 M2,symmetric] assms*)
 qed

lemma *integrable-product-swap-iff-s-finite*:
 fixes $f :: - \Rightarrow -::\{\text{banach, second-countable-topology}\}$
 assumes $M1:s\text{-finite-measure } M1$ and $M2:s\text{-finite-measure } M2$
 shows *integrable* ($M2 \otimes_M M1$) ($\lambda(x,y). f (y,x)$) \longleftrightarrow *integrable* ($M1 \otimes_M M2$)
 f
proof –
 from *integrable-product-swap-s-finite[OF M2 M1, of $\lambda(x,y). f (y,x)$]* *integrable-product-swap-s-finite[OF*
 $M1 M2, of f]$
 show ?thesis by auto
 qed

lemma *integral-product-swap-s-finite*:
 fixes $f :: - \Rightarrow -::\{\text{banach, second-countable-topology}\}$
 assumes $M1:s\text{-finite-measure } M1$ and $M2:s\text{-finite-measure } M2$
 and $f: f \in \text{borel-measurable } (M1 \otimes_M M2)$
 shows ($\int (x,y). f (y,x) \partial(M2 \otimes_M M1) = \text{integral}^L (M1 \otimes_M M2) f$)
proof –
 have *: ($\lambda(x,y). f (y,x) = (\lambda x. f (\text{case } x \text{ of } (x,y) \Rightarrow (y,x)))$) by (auto *simp:*
fun-eq-iff)
 show ?thesis unfolding *
 by (*simp add: integral-distr[symmetric, OF measurable-pair-swap' f] distr-pair-swap-s-finite[OF*
 $M1 M2, symmetric]$)
 qed

theorem(in *s-finite-measure*) *Fubini-integrable'*:

fixes $f :: - \Rightarrow -::\{\text{banach, second-countable-topology}\}$
assumes $f[\text{measurable}]$: $f \in \text{borel-measurable } (M1 \otimes_M M)$
and integ1 : $\text{integrable } M1 \ (\lambda x. \int y. \text{norm } (f \ (x, y)) \ \partial M)$
and integ2 : $\text{AE } x \text{ in } M1. \text{ integrable } M \ (\lambda y. f \ (x, y))$
shows $\text{integrable } (M1 \otimes_M M) f$
proof (rule *integrableI-bounded*)
have $(\int^+ p. \text{norm } (f p) \ \partial(M1 \otimes_M M)) = (\int^+ x. (\int^+ y. \text{norm } (f \ (x, y)) \ \partial M) \ \partial M1)$
by (simp add: *nn-integral-fst'[symmetric]*)
also have $\dots = (\int^+ x. |\int y. \text{norm } (f \ (x, y)) \ \partial M| \ \partial M1)$
proof(rule *nn-integral-cong-AE*)
show $\text{AE } x \text{ in } M1. (\int^+ y. \text{ennreal } (\text{norm } (f \ (x, y))) \ \partial M) = \text{ennreal } |LINT y|M. \text{norm } (f \ (x, y))|$
using *integ2*
proof *eventually-elim*
fix x **assume** $\text{integrable } M \ (\lambda y. f \ (x, y))$
then have f : $\text{integrable } M \ (\lambda y. \text{norm } (f \ (x, y)))$
by *simp*
then have $(\int^+ y. \text{ennreal } (\text{norm } (f \ (x, y))) \ \partial M) = \text{ennreal } (LINT y|M. \text{norm } (f \ (x, y)))$
by (rule *nn-integral-eq-integral*) *simp*
also have $\dots = \text{ennreal } |LINT y|M. \text{norm } (f \ (x, y))|$
using f **by** *simp*
finally show $(\int^+ y. \text{ennreal } (\text{norm } (f \ (x, y))) \ \partial M) = \text{ennreal } |LINT y|M. \text{norm } (f \ (x, y))|$.
qed
qed
also have $\dots < \infty$
using *integ1* **by** (simp add: *integrable-iff-bounded integral-nonneg-AE*)
finally show $(\int^+ p. \text{norm } (f p) \ \partial(M1 \otimes_M M)) < \infty$.
qed fact

lemma(in *s-finite-measure*) *emeasure-pair-measure-finite'*:
assumes A : $A \in \text{sets } (M1 \otimes_M M)$ **and** *finite*: $\text{emeasure } (M1 \otimes_M M) A < \infty$
shows $\text{AE } x \text{ in } M1. \text{ emeasure } M \ \{y \in \text{space } M. (x, y) \in A\} < \infty$
proof –
from *emeasure-pair-measure-alt'[OF A]* *finite*
have $(\int^+ x. \text{emeasure } M \ (\text{Pair } x - 'A) \ \partial M1) \neq \infty$
by *simp*
then have $\text{AE } x \text{ in } M1. \text{ emeasure } M \ (\text{Pair } x - 'A) \neq \infty$
by (rule *nn-integral-PInf-AE[rotated]*) (*intro measurable-emeasure-Pair' A*)
moreover have $\bigwedge x. x \in \text{space } M1 \implies \text{Pair } x - 'A = \{y \in \text{space } M. (x, y) \in A\}$
using *sets.sets-into-space[OF A]* **by** (*auto simp: space-pair-measure*)
ultimately show *?thesis* **by** (*auto simp: less-top*)
qed

lemma(in *s-finite-measure*) *AE-integrable-fst'''*:
fixes $f :: - \Rightarrow -::\{\text{banach, second-countable-topology}\}$
assumes $f[\text{measurable}]$: $\text{integrable } (M1 \otimes_M M) f$

shows $AE\ x\ in\ M1.\ integrable\ M\ (\lambda y.\ f\ (x,\ y))$
proof –
have $(\int^{+x}.\ (\int^{+y}.\ norm\ (f\ (x,\ y))\ \partial M)\ \partial M1) = (\int^{+x}.\ norm\ (f\ x)\ \partial(M1\ \otimes_M\ M))$
by $(rule\ nn-integral-fst')$ *simp*
also have $(\int^{+x}.\ norm\ (f\ x)\ \partial(M1\ \otimes_M\ M)) \neq \infty$
using f **unfolding** *integrable-iff-bounded* **by** *simp*
finally have $AE\ x\ in\ M1.\ (\int^{+y}.\ norm\ (f\ (x,\ y))\ \partial M) \neq \infty$
by $(intro\ nn-integral-PInf-AE\ borel-measurable-nn-integral')$
 $(auto\ simp:\ measurable-split-conv)$
with *AE-space* **show** *?thesis*
by *eventually-elim*
 $(auto\ simp:\ integrable-iff-bounded\ measurable-compose[OF\ -\ borel-measurable-integrable[OF\ f]])$ *less-top*
qed

lemma(*in s-finite-measure*) *integrable-fst-norm'*:

fixes $f :: - \Rightarrow - :: \{banach,\ second-countable-topology\}$
assumes $f[measurable]: integrable\ (M1\ \otimes_M\ M)\ f$
shows $integrable\ M1\ (\lambda x.\ \int y.\ norm\ (f\ (x,\ y))\ \partial M)$
unfolding *integrable-iff-bounded*

proof

show $(\lambda x.\ \int y.\ norm\ (f\ (x,\ y))\ \partial M) \in borel-measurable\ M1$
by $(rule\ borel-measurable-lebesgue-integral')$ *simp*
have $(\int^{+x}.\ ennreal\ (norm\ (\int y.\ norm\ (f\ (x,\ y))\ \partial M))\ \partial M1) = (\int^{+x}.\ (\int^{+y}.\ norm\ (f\ (x,\ y))\ \partial M)\ \partial M1)$
using *AE-integrable-fst'''[OF f]* **by** $(auto\ intro!:\ nn-integral-cong-AE\ simp:\ nn-integral-eq-integral)$
also have $(\int^{+x}.\ (\int^{+y}.\ norm\ (f\ (x,\ y))\ \partial M)\ \partial M1) = (\int^{+x}.\ norm\ (f\ x)\ \partial(M1\ \otimes_M\ M))$
by $(rule\ nn-integral-fst')$ *simp*
also have $(\int^{+x}.\ norm\ (f\ x)\ \partial(M1\ \otimes_M\ M)) < \infty$
using f **unfolding** *integrable-iff-bounded* **by** *simp*
finally show $(\int^{+x}.\ ennreal\ (norm\ (\int y.\ norm\ (f\ (x,\ y))\ \partial M))\ \partial M1) < \infty$.
qed

lemma(*in s-finite-measure*) *integrable-fst'''*:

fixes $f :: - \Rightarrow - :: \{banach,\ second-countable-topology\}$
assumes $f[measurable]: integrable\ (M1\ \otimes_M\ M)\ f$
shows $integrable\ M1\ (\lambda x.\ \int y.\ f\ (x,\ y)\ \partial M)$
by $(auto\ intro!:\ Bochner-Integration.integrable-bound[OF\ integrable-fst-norm'[OF\ f]])$

proposition(*in s-finite-measure*) *integral-fst'''*:

fixes $f :: - \Rightarrow - :: \{banach,\ second-countable-topology\}$
assumes $f: integrable\ (M1\ \otimes_M\ M)\ f$
shows $(\int x.\ (\int y.\ f\ (x,\ y)\ \partial M)\ \partial M1) = integral^L\ (M1\ \otimes_M\ M)\ f$
using f **proof** *induct*
case $(base\ A\ c)$

have A [*measurable*]: $A \in \text{sets } (M1 \otimes_M M)$ **by fact**

have eq : $\bigwedge x y. x \in \text{space } M1 \implies \text{indicator } A (x, y) = \text{indicator } \{y \in \text{space } M. (x, y) \in A\} y$
using *sets.sets-into-space[OF A]* **by** (*auto split: split-indicator simp: space-pair-measure*)

have $int-A$: *integrable* $(M1 \otimes_M M)$ (*indicator* $A :: - \implies \text{real}$)
using *base* **by** (*rule integrable-real-indicator*)

have $(\int x. \int y. \text{indicator } A (x, y) *_R c \partial M \partial M1) = (\int x. \text{measure } M \{y \in \text{space } M. (x, y) \in A\} *_R c \partial M1)$
proof (*intro integral-cong-AE*)
from *AE-integrable-fst''[OF int-A]* *AE-space*
show *AE* x *in* $M1$. $(\int y. \text{indicator } A (x, y) *_R c \partial M) = \text{measure } M \{y \in \text{space } M. (x, y) \in A\} *_R c$
by *eventually-elim (simp add: eq integrable-indicator-iff)*

qed *simp-all*

also have $\dots = \text{measure } (M1 \otimes_M M) A *_R c$
proof (*subst integral-scaleR-left*)
have $(\int^+ x. \text{ennreal } (\text{measure } M \{y \in \text{space } M. (x, y) \in A\}) \partial M1) = (\int^+ x. \text{emeasure } M \{y \in \text{space } M. (x, y) \in A\} \partial M1)$
using *emeasure-pair-measure-finite'[OF base]*
by (*intro nn-integral-cong-AE, eventually-elim*) (*simp add: emeasure-eq-ennreal-measure*)

also have $\dots = \text{emeasure } (M1 \otimes_M M) A$
using *sets.sets-into-space[OF A]*
by (*subst emeasure-pair-measure-alt'*)
(auto intro!: nn-integral-cong arg-cong[where f=emeasure M] simp: space-pair-measure)

finally have $*$: $(\int^+ x. \text{ennreal } (\text{measure } M \{y \in \text{space } M. (x, y) \in A\}) \partial M1) = \text{emeasure } (M1 \otimes_M M) A$.

from *base* * **show** *integrable* $M1$ $(\lambda x. \text{measure } M \{y \in \text{space } M. (x, y) \in A\})$
by (*simp add: integrable-iff-bounded*)

then have $(\int x. \text{measure } M \{y \in \text{space } M. (x, y) \in A\} \partial M1) = (\int^+ x. \text{ennreal } (\text{measure } M \{y \in \text{space } M. (x, y) \in A\}) \partial M1)$
by (*rule nn-integral-eq-integral[symmetric]*) *simp*

also note *

finally show $(\int x. \text{measure } M \{y \in \text{space } M. (x, y) \in A\} \partial M1) *_R c = \text{measure } (M1 \otimes_M M) A *_R c$
using *base* **by** (*simp add: emeasure-eq-ennreal-measure*)

qed

also have $\dots = (\int a. \text{indicator } A a *_R c \partial (M1 \otimes_M M))$
using *base* **by** *simp*

finally show *?case* .

next

case (*add f g*)

then have [*measurable*]: $f \in \text{borel-measurable } (M1 \otimes_M M)$ $g \in \text{borel-measurable } (M1 \otimes_M M)$
by *auto*

have $(\int x. \int y. f (x, y) + g (x, y) \partial M \partial M1) =$

```

    (∫ x. (∫ y. f (x, y) ∂M) + (∫ y. g (x, y) ∂M) ∂M1)
  apply (rule integral-cong-AE)
  apply simp-all
  using AE-integrable-fst'''[OF add(1)] AE-integrable-fst'''[OF add(3)]
  apply eventually-elim
  apply simp
  done
  also have ... = (∫ x. f x ∂(M1 ⊗M M)) + (∫ x. g x ∂(M1 ⊗M M))
    using integrable-fst'''[OF add(1)] integrable-fst'''[OF add(3)] add(2,4) by simp
  finally show ?case
    using add by simp
next
  case (lim f s)
  then have [measurable]: f ∈ borel-measurable (M1 ⊗M M) ∧ i. s i ∈ borel-measurable
(M1 ⊗M M)
    by auto

  show ?case
  proof (rule LIMSEQ-unique)
    show (λi. integralL (M1 ⊗M M) (s i)) → integralL (M1 ⊗M M) f
    proof (rule integral-dominated-convergence)
      show integrable (M1 ⊗M M) (λx. 2 * norm (f x))
        using lim(5) by auto
    qed (insert lim, auto)
  have (λi. ∫ x. ∫ y. s i (x, y) ∂M ∂M1) → ∫ x. ∫ y. f (x, y) ∂M ∂M1
  proof (rule integral-dominated-convergence)
    have AE x in M1. ∀ i. integrable M (λy. s i (x, y))
      unfolding AE-all-countable using AE-integrable-fst'''[OF lim(1)] ..
    with AE-space AE-integrable-fst'''[OF lim(5)]
    show AE x in M1. (λi. ∫ y. s i (x, y) ∂M) → ∫ y. f (x, y) ∂M
  proof eventually-elim
    fix x assume x: x ∈ space M1 and
      s: ∀ i. integrable M (λy. s i (x, y)) and f: integrable M (λy. f (x, y))
    show (λi. ∫ y. s i (x, y) ∂M) → ∫ y. f (x, y) ∂M
    proof (rule integral-dominated-convergence)
      show integrable M (λy. 2 * norm (f (x, y)))
        using f by auto
      show AE xa in M. (λi. s i (x, xa)) → f (x, xa)
        using x lim(3) by (auto simp: space-pair-measure)
      show ∧ i. AE xa in M. norm (s i (x, xa)) ≤ 2 * norm (f (x, xa))
        using x lim(4) by (auto simp: space-pair-measure)
    qed (insert x, measurable)
  qed
  show integrable M1 (λx. (∫ y. 2 * norm (f (x, y)) ∂M))
  by (intro integrable-mult-right integrable-norm integrable-fst''' lim)
  fix i show AE x in M1. norm (∫ y. s i (x, y) ∂M) ≤ (∫ y. 2 * norm (f (x,
y)) ∂M)
    using AE-space AE-integrable-fst'''[OF lim(1), of i] AE-integrable-fst'''[OF
lim(5)]

```

proof *eventually-elim*
fix x **assume** $x: x \in \text{space } M1$
and $s: \text{integrable } M (\lambda y. s \ i \ (x, y))$ **and** $f: \text{integrable } M (\lambda y. f \ (x, y))$
from s **have** $\text{norm } (\int y. s \ i \ (x, y) \ \partial M) \leq (\int^+ y. \text{norm } (s \ i \ (x, y)) \ \partial M)$
by *(rule integral-norm-bound-ennreal)*
also have $\dots \leq (\int^+ y. 2 * \text{norm } (f \ (x, y)) \ \partial M)$
using x **lim** **by** *(auto intro!: nn-integral-mono simp: space-pair-measure)*
also have $\dots = (\int y. 2 * \text{norm } (f \ (x, y)) \ \partial M)$
using f **by** *(intro nn-integral-eq-integral) auto*
finally show $\text{norm } (\int y. s \ i \ (x, y) \ \partial M) \leq (\int y. 2 * \text{norm } (f \ (x, y)) \ \partial M)$
by *simp*
qed
qed *simp-all*
then show $(\lambda i. \text{integral}^L (M1 \otimes_M M) (s \ i)) \longrightarrow \int x. \int y. f \ (x, y) \ \partial M$
 $\partial M1$
using *lim* **by** *simp*
qed
qed

lemma *(in s-finite-measure)*
fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
assumes $f: \text{integrable } (M1 \otimes_M M) \text{ (case-prod } f)$
shows *AE-integrable-fst''*: $AE \ x \ \text{in } M1. \text{integrable } M (\lambda y. f \ x \ y)$
and *integrable-fst''*: $\text{integrable } M1 (\lambda x. \int y. f \ x \ y \ \partial M)$
and *integrable-fst-norm*: $\text{integrable } M1 (\lambda x. \int y. \text{norm } (f \ x \ y) \ \partial M)$
and *integral-fst''*: $(\int x. (\int y. f \ x \ y \ \partial M) \ \partial M1) = \text{integral}^L (M1 \otimes_M M) (\lambda(x, y). f \ x \ y)$
using *AE-integrable-fst'''[OF f]* *integrable-fst'''[OF f]* *integral-fst'''[OF f]* *integrable-fst-norm'[OF f]* **by** *auto*

lemma
fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
assumes $M1: s\text{-finite-measure } M1$ **and** $M2: s\text{-finite-measure } M2$
and $f[\text{measurable}]: \text{integrable } (M1 \otimes_M M2) \text{ (case-prod } f)$
shows *AE-integrable-snd-s-finite*: $AE \ y \ \text{in } M2. \text{integrable } M1 (\lambda x. f \ x \ y)$ **(is ?AE)**
and *integrable-snd-s-finite*: $\text{integrable } M2 (\lambda y. \int x. f \ x \ y \ \partial M1)$ **(is ?INT)**
and *integrable-snd-norm-s-finite*: $\text{integrable } M2 (\lambda y. \int x. \text{norm } (f \ x \ y) \ \partial M1)$
(is ?INT2)
and *integral-snd-s-finite*: $(\int y. (\int x. f \ x \ y \ \partial M1) \ \partial M2) = \text{integral}^L (M1 \otimes_M M2) \text{ (case-prod } f)$ **(is ?EQ)**
proof –
interpret $Q: s\text{-finite-measure } M1$ **by** *fact*
have $Q\text{-int}: \text{integrable } (M2 \otimes_M M1) (\lambda(x, y). f \ y \ x)$
using f **unfolding** *integrable-product-swap-iff-s-finite[OF M1 M2, symmetric]*
by *simp*
show *?AE* **using** $Q.AE\text{-integrable-fst'''[OF } Q\text{-int}]$ **by** *simp*
show *?INT* **using** $Q.\text{integrable-fst'''[OF } Q\text{-int}]$ **by** *simp*
show *?INT2* **using** $Q.\text{integrable-fst-norm[OF } Q\text{-int}]$ **by** *simp*
show *?EQ* **using** $Q.\text{integral-fst'''[OF } Q\text{-int}]$

using *integral-product-swap-s-finite*[*OF M1 M2, of case-prod f*] by *simp*
qed

proposition *Fubini-integral'*:

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

assumes $M1:s\text{-finite-measure } M1$ **and** $M2:s\text{-finite-measure } M2$

and $f: \text{integrable } (M1 \otimes_M M2)$ (*case-prod f*)

shows $(\int y. (\int x. f x y \partial M1) \partial M2) = (\int x. (\int y. f x y \partial M2) \partial M1)$

unfolding *integral-snd-s-finite*[*OF assms*] *s-finite-measure.integral-fst'*[*OF assms(2,3)*]

..

locale *product-s-finite* =

fixes $M :: 'i \Rightarrow 'a \text{ measure}$

assumes *s-finite-measures*: $\bigwedge i. s\text{-finite-measure } (M i)$

sublocale *product-s-finite* $\subseteq M?$: *s-finite-measure* $M i$ **for** i

by (*rule s-finite-measures*)

locale *finite-product-s-finite* = *product-s-finite* M **for** $M :: 'i \Rightarrow 'a \text{ measure} +$

fixes $I :: 'i \text{ set}$

assumes *finite-index*: *finite* I

lemma (**in** *product-s-finite*) *emeasure-PiM*:

finite $I \implies (\bigwedge i. i \in I \implies A i \in \text{sets } (M i)) \implies \text{emeasure } (PiM I M) (PiE I A)$
 $= (\prod_{i \in I. \text{emeasure } (M i) (A i)}$

proof (*induct I arbitrary: A rule: finite-induct*)

case (*insert i I*)

interpret *finite-product-s-finite* $M I$ **by** *standard fact*

have *finite* (*insert i I*) **using** $\langle \text{finite } I \rangle$ **by** *auto*

interpret I' : *finite-product-s-finite* M *insert i I* **by** *standard fact*

let $?h = (\lambda(f, y). f(i := y))$

let $?P = \text{distr } (PiM I M \otimes_M M i) (PiM (\text{insert } i I) M) ?h$

let $?mu = \text{emeasure } ?P$

let $?I = \{j \in \text{insert } i I. \text{emeasure } (M j) (\text{space } (M j)) \neq 1\}$

let $?f = \lambda J E j. \text{if } j \in J \text{ then } \text{emeasure } (M j) (E j) \text{ else } \text{emeasure } (M j) (\text{space } (M j))$

have $\text{emeasure } (PiM (\text{insert } i I) M) (\text{prod-emb } (\text{insert } i I) M (\text{insert } i I) (PiE (\text{insert } i I) A)) =$

$(\prod_{i \in \text{insert } i I. \text{emeasure } (M i) (A i)}$

proof (*subst emeasure-extend-measure-Pair*[*OF PiM-def*])

fix $J E$ **assume** $(J \neq \{\} \vee \text{insert } i I = \{\}) \wedge \text{finite } J \wedge J \subseteq \text{insert } i I \wedge E \in$
 $(\prod_{j \in J. \text{sets } (M j)})$

then have $J: J \neq \{\} \text{ finite } J \wedge J \subseteq \text{insert } i I$ **and** $E: \forall j \in J. E j \in \text{sets } (M j)$

by *auto*

let $?p = \text{prod-emb } (\text{insert } i I) M J (PiE J E)$

let $?p' = \text{prod-emb } I M (J - \{i\}) (\prod_{E j \in J - \{i\}. E j}$

have $?mu ?p =$

$\text{emeasure } (Pi_M I M \otimes_M (M i))$ ($?h - ' ?p \cap \text{space } (Pi_M I M \otimes_M M i)$)
by (*intro emeasure-distr measurable-add-dim sets-PiM-I*) *fact+*
also have $?h - ' ?p \cap \text{space } (Pi_M I M \otimes_M M i) = ?p' \times$ (*if $i \in J$ then $E i$ else space $(M i)$*)
using $J E$ [*rule-format, THEN sets.sets-into-space*]
by (*force simp: space-pair-measure space-PiM prod-emb-iff PiE-def Pi-iff split: if-split-asm*)
also have $\text{emeasure } (Pi_M I M \otimes_M (M i))$ ($?p' \times$ (*if $i \in J$ then $E i$ else space $(M i)$*)) =
 $\text{emeasure } (Pi_M I M)$ $?p' * \text{emeasure } (M i)$ (*if $i \in J$ then $(E i)$ else space $(M i)$*)
using $J E$ **by** (*intro M.emeasure-pair-measure-Times' sets-PiM-I*) *auto*
also have $?p' = (\prod_{E j \in I. \text{if } j \in J - \{i\} \text{ then } E j \text{ else space } (M j)})$
using $J E$ [*rule-format, THEN sets.sets-into-space*]
by (*auto simp: prod-emb-iff PiE-def Pi-iff split: if-split-asm*) *blast+*
also have $\text{emeasure } (Pi_M I M)$ ($\prod_{E j \in I. \text{if } j \in J - \{i\} \text{ then } E j \text{ else space } (M j)}$) =
 $(\prod_{j \in I. \text{if } j \in J - \{i\} \text{ then emeasure } (M j) (E j) \text{ else emeasure } (M j) (\text{space } (M j))})$
using E **by** (*subst insert*) (*auto intro!: prod.cong*)
also have $(\prod_{j \in I. \text{if } j \in J - \{i\} \text{ then emeasure } (M j) (E j) \text{ else emeasure } (M j) (\text{space } (M j))}) * \text{emeasure } (M i)$ (*if $i \in J$ then $E i$ else space $(M i)$*) = $(\prod_{j \in \text{insert } i I. ?f J E j})$
using *insert* **by** (*auto simp: mult.commute intro!: arg-cong2[where f=(*)] prod.cong*)
also have $\dots = (\prod_{j \in J \cup I. ?f J E j})$
using *insert(1,2)* $J E$ **by** (*intro prod.mono-neutral-right*) *auto*
finally show $? \mu ?p = \dots$.

show *prod-emb* (*insert* $i I$) $M J (Pi_E J E) \in \text{Pow } (\prod_{E i \in \text{insert } i I. \text{space } (M i)})$
using $J E$ [*rule-format, THEN sets.sets-into-space*] **by** (*auto simp: prod-emb-iff PiE-def*)
next
show *positive* (*sets* $(Pi_M (\text{insert } i I) M))$ $? \mu$ *countably-additive* (*sets* $(Pi_M (\text{insert } i I) M))$ $? \mu$
using *emeasure-positive[of ?P] emeasure-countably-additive[of ?P]* **by** *simp-all*
next
show (*insert* $i I \neq \{\}$) \vee (*insert* $i I = \{\}$) \wedge *finite* (*insert* $i I$) \wedge
 $\text{insert } i I \subseteq \text{insert } i I \wedge A \in (\prod_{j \in \text{insert } i I. \text{sets } (M j)})$
using *insert* **by** *auto*
qed (*auto intro!: prod.cong*)
with *insert* **show** $? \text{case}$
by (*subst (asm) prod-emb-PiE-same-index*) (*auto intro!: sets.sets-into-space*)
qed *simp*

lemma (*in finite-product-s-finite*) *measure-times*:

$(\bigwedge i. i \in I \implies A \ i \in \text{sets } (M \ i)) \implies \text{emeasure } (Pi_M \ I \ M) (Pi_E \ I \ A) = (\prod_{i \in I}. \text{emeasure } (M \ i) (A \ i))$

using *emeasure-PiM[OF finite-index]* **by auto**

lemma (**in** *product-s-finite*) *nn-integral-empty*:

$0 \leq f \ (\lambda k. \text{undefined}) \implies \text{integral}^N (Pi_M \ \{\} \ M) \ f = f \ (\lambda k. \text{undefined})$

by (*simp add: PiM-empty nn-integral-count-space-finite*)

Every s-finite measure is represented as the push-forward measure of a σ -finite measure.

definition *Mi-to-NM* :: $(\text{nat} \Rightarrow 'a \ \text{measure}) \Rightarrow 'a \ \text{measure} \Rightarrow (\text{nat} \times 'a) \ \text{measure}$
where

Mi-to-NM *Mi* *M* $\equiv \text{measure-of } (\text{space } (\text{count-space } UNIV \otimes_M M)) (\text{sets } (\text{count-space } UNIV \otimes_M M)) (\lambda A. \sum i. \text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\lambda x. (i, x)) A)$

lemma

shows *sets-Mi-to-NM*[*measurable-cong, simp*]: $\text{sets } (Mi\text{-to-NM } Mi \ M) = \text{sets } (\text{count-space } UNIV \otimes_M M)$

and *space-Mi-to-NM*[*simp*]: $\text{space } (Mi\text{-to-NM } Mi \ M) = \text{space } (\text{count-space } UNIV \otimes_M M)$

by(*simp-all add: Mi-to-NM-def*)

context

fixes *M* :: $'a \ \text{measure}$ **and** *Mi* :: $\text{nat} \Rightarrow 'a \ \text{measure}$

assumes *sets-Mi*[*measurable-cong, simp*]: $\bigwedge i. \text{sets } (Mi \ i) = \text{sets } M$

and *emeasure-Mi*: $\bigwedge A. A \in \text{sets } M \implies M \ A = (\sum i. Mi \ i \ A)$

begin

lemma *emeasure-Mi-to-NM*:

assumes [*measurable*]: $A \in \text{sets } (\text{count-space } UNIV \otimes_M M)$

shows $\text{emeasure } (Mi\text{-to-NM } Mi \ M) \ A = (\sum i. \text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\lambda x. (i, x)) \ A)$

proof(*rule emeasure-measure-of*[**where** $\Omega = \text{space } (\text{count-space } UNIV \otimes_M M)$ **and** $A = \text{sets } (\text{count-space } UNIV \otimes_M M)$])

show *countably-additive* ($\text{sets } (Mi\text{-to-NM } Mi \ M)$) $(\lambda A. \sum i. \text{emeasure } (\text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\text{Pair } i)) \ A)$

unfolding *countably-additive-def*

proof *safe*

fix *A* :: $\text{nat} \Rightarrow (\text{nat} \times -) \ \text{set}$

assume $\text{range } A \subseteq \text{sets } (Mi\text{-to-NM } Mi \ M)$ **and** *dA*:*disjoint-family* *A*

hence [*measurable*]: $\bigwedge i. A \ i \in \text{sets } (\text{count-space } UNIV \otimes_M M)$

by auto

show $(\sum j \ i. \text{emeasure } (\text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\text{Pair } i)) (A \ j)) = (\sum i. \text{emeasure } (\text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\text{Pair } i)) (\bigcup (\text{range } A)))$ (**is** *?lhs = ?rhs*)

proof $-$

have *?lhs* = $(\sum i \ j. \text{emeasure } (\text{distr } (Mi \ i) (\text{count-space } UNIV \otimes_M M) (\text{Pair } i)) (A \ j))$

by(*auto simp: nn-integral-count-space-nat[symmetric] pair-sigma-finite-def*
sigma-finite-measure-count-space intro!: pair-sigma-finite.Fubini')
also have ... = ?*rhs*
proof(*rule suminf-cong*)
fix *n*
have [*simp*]:*Pair n - ' ∪ (range A) = (∪ (range (λj. Pair n - ' A j)))*
by *auto*
show ($\sum j. \text{emeasure } (\text{distr } (Mi\ n) \text{ (count-space UNIV } \otimes_M M) \text{ (Pair } n))$
 $(A\ j) = \text{emeasure } (\text{distr } (Mi\ n) \text{ (count-space UNIV } \otimes_M M) \text{ (Pair } n)) (\bigcup (\text{range } A))$)
using *dA by(fastforce intro!: suminf-emeasure simp: disjoint-family-on-def*
emeasure-distr)
qed
finally show ?*thesis* .
qed
qed(*auto simp: positive-def sets.space-closed Mi-to-NM-def*)

lemma *sigma-finite-Mi-to-NM-measure:*
assumes $\bigwedge i. \text{finite-measure } (Mi\ i)$
shows *sigma-finite-measure (Mi-to-NM Mi M)*
proof –
{
fix *n*
assume *emeasure (Mi-to-NM Mi M) ({n} × space M) = ⊤*
moreover have *emeasure (Mi-to-NM Mi M) ({n} × space M) = emeasure*
 $(Mi\ n) \text{ (space } M)$
by(*simp add: emeasure-Mi-to-NM emeasure-distr suminf-offset[of - Suc n]*)
ultimately have *False*
using *finite-measure.finite-emeasure-space[OF assms[of n]] by(auto simp:*
sets-eq-imp-space-eq[OF sets-Mi])
}
thus ?*thesis*
by(*auto intro!: exI[where x=∪ i. {{i} × space M}] simp: space-pair-measure*
sigma-finite-measure-def)
qed

lemma *distr-Mi-to-NM-M: distr (Mi-to-NM Mi M) M snd = M*
proof –
have [*simp*]:*Pair i - ' snd - ' A ∩ Pair i - ' space (count-space UNIV } \otimes_M M)*
 $= A$ **if** $A \in \text{sets } M$ **for** A **and** $i :: \text{nat}$
using *sets.sets-into-space[OF that] by(auto simp: space-pair-measure)*
show ?*thesis*
by(*auto intro!: measure-eqI simp: emeasure-distr emeasure-Mi-to-NM emeasure-Mi*)
qed
end

context

fixes $\mu :: 'a \text{ measure}$
assumes *standard-borel-ne*: *standard-borel-ne* μ
and *s-finite*: *s-finite-measure* μ

begin

interpretation $\mu : \text{s-finite-measure } \mu$ **by fact**

interpretation *n- μ* : *standard-borel-ne count-space* (*UNIV* :: *nat set*) $\otimes_M \mu$
by (*simp add*: *pair-standard-borel-ne standard-borel-ne*)

lemma *exists-push-forward*:

$\exists (\mu' :: \text{real measure}) f. f \in \text{borel} \rightarrow_M \mu \wedge \text{sets } \mu' = \text{sets borel} \wedge \text{sigma-finite-measure } \mu'$
 $\wedge \text{distr } \mu' \mu f = \mu$

proof –

obtain μi **where** $\mu i: \wedge i. \text{sets } (\mu i i) = \text{sets } \mu \wedge i. \text{finite-measure } (\mu i i) \wedge A. \mu A = (\sum i. \mu i i A)$

by (*metis* $\mu.\text{finite-measures}'$)

show *?thesis*

proof (*safe intro!*: *exI* [**where** $x = \text{distr } (Mi\text{-to-NM } \mu i \mu) \text{ borel } n\text{-}\mu.\text{to-real}$] *exI* [**where** $x = \text{snd} \circ n\text{-}\mu.\text{from-real}$])

have [*simp*]: $\text{distr } (\text{distr } (Mi\text{-to-NM } \mu i \mu) \text{ borel } n\text{-}\mu.\text{to-real}) (\text{count-space } UNIV \otimes_M \mu) n\text{-}\mu.\text{from-real} = Mi\text{-to-NM } \mu i \mu$

by (*auto simp*: *distr-distr comp-def intro!*: *distr-id'*)

show *sigma-finite-measure* ($\text{distr } (Mi\text{-to-NM } \mu i \mu) \text{ borel } n\text{-}\mu.\text{to-real}$)

by (*rule sigma-finite-measure-distr* [**where** $N = \text{count-space } UNIV \otimes_M \mu$ **and** $f = n\text{-}\mu.\text{from-real}$]) (*auto intro!*: *sigma-finite-Mi-to-NM-measure* μi)

next

have [*simp*]: $\text{distr } (Mi\text{-to-NM } \mu i \mu) \mu (\text{snd} \circ n\text{-}\mu.\text{from-real} \circ n\text{-}\mu.\text{to-real}) = \text{distr } (Mi\text{-to-NM } \mu i \mu) \mu \text{snd}$

by (*auto intro!*: *distr-cong* [*OF refl*])

show $\text{distr } (\text{distr } (Mi\text{-to-NM } \mu i \mu) \text{ borel } n\text{-}\mu.\text{to-real}) \mu (\text{snd} \circ n\text{-}\mu.\text{from-real}) = \mu$

by (*auto simp*: *distr-distr distr-Mi-to-NM-M* [*OF* $\mu i(1,3)$])

qed *auto*

qed

abbreviation $\mu'\text{-and-f} \equiv (\text{SOME } (\mu' :: \text{real measure}, f). f \in \text{borel} \rightarrow_M \mu \wedge \text{sets } \mu' = \text{sets borel} \wedge \text{sigma-finite-measure } \mu' \wedge \text{distr } \mu' \mu f = \mu)$

definition *sigma-pair- μ* $\equiv \text{fst } \mu'\text{-and-f}$

definition *sigma-pair-f* $\equiv \text{snd } \mu'\text{-and-f}$

lemma

shows *sigma-pair-f-measurable* : $\text{sigma-pair-f} \in \text{borel} \rightarrow_M \mu$ (**is** *?g1*)

and *sets-sigma-pair- μ* : $\text{sets } \text{sigma-pair-}\mu = \text{sets borel}$ (**is** *?g2*)

and *sigma-finite-sigma-pair- μ* : *sigma-finite-measure* *sigma-pair- μ* (**is** *?g3*)

and *distr-sigma-pair*: $\text{distr sigma-pair-}\mu \mu \text{ sigma-pair-f} = \mu$ (**is** ?g4)
proof –
have *case* μ' -and-f of $(\mu',f) \Rightarrow f \in \text{borel} \rightarrow_M \mu \wedge \text{sets } \mu' = \text{sets borel} \wedge$
sigma-finite-measure $\mu' \wedge \text{distr } \mu' \mu f = \mu$
by(*rule someI-ex*) (*use exists-push-forward in auto*)
then show ?g1 ?g2 ?g3 ?g4
by(*auto simp: sigma-pair-μ-def sigma-pair-f-def split-beta*)
qed

end

definition *s-finite-measure-algebra* :: 'a *measure* \Rightarrow 'a *measure measure* **where**
s-finite-measure-algebra $K =$
 $(\text{SUP } A \in \text{sets } K. \text{vimage-algebra } \{M. \text{s-finite-measure } M \wedge \text{sets } M = \text{sets } K\})$
 $(\lambda M. \text{emeasure } M A) \text{ borel}$

lemma *space-s-finite-measure-algebra*:
 $\text{space } (\text{s-finite-measure-algebra } K) = \{M. \text{s-finite-measure } M \wedge \text{sets } M = \text{sets } K\}$
by (*auto simp add: s-finite-measure-algebra-def space-Sup-eq-UN*)

lemma *s-finite-measure-algebra-cong*: $\text{sets } M = \text{sets } N \Longrightarrow \text{s-finite-measure-algebra } M = \text{s-finite-measure-algebra } N$
by (*simp add: s-finite-measure-algebra-def*)

lemma *measurable-emeasure-s-finite-measure-algebra*[*measurable*]:
 $a \in \text{sets } A \Longrightarrow (\lambda M. \text{emeasure } M a) \in \text{borel-measurable } (\text{s-finite-measure-algebra } A)$
by (*auto intro!: measurable-Sup1 measurable-vimage-algebra1 simp: s-finite-measure-algebra-def*)

lemma *measurable-measure-s-finite-measure-algebra*[*measurable*]:
 $a \in \text{sets } A \Longrightarrow (\lambda M. \text{measure } M a) \in \text{borel-measurable } (\text{s-finite-measure-algebra } A)$
unfolding *measure-def* **by** *measurable*

lemma *s-finite-measure-algebra-measurableD*:
assumes $N: N \in \text{measurable } M (\text{s-finite-measure-algebra } S)$ **and** $x: x \in \text{space } M$
shows $\text{space } (N x) = \text{space } S$
and $\text{sets } (N x) = \text{sets } S$
and $\text{measurable } (N x) K = \text{measurable } S K$
and $\text{measurable } K (N x) = \text{measurable } K S$
using *measurable-space[OF N x]*
by (*auto simp: space-s-finite-measure-algebra intro!: measurable-cong-sets dest: sets-eq-imp-space-eq*)

context

fixes $K M N$ **assumes** $K: K \in \text{measurable } M (\text{s-finite-measure-algebra } N)$
begin

lemma *s-finite-measure-algebra-kernel*: $a \in \text{space } M \Longrightarrow \text{s-finite-measure } (K a)$

using measurable-space[OF K] **by** (simp add: space-s-finite-measure-algebra)

lemma s-finite-measure-algebra-sets-kernel: $a \in \text{space } M \implies \text{sets } (K a) = \text{sets } N$
using measurable-space[OF K] **by** (simp add: space-s-finite-measure-algebra)

lemma measurable-emeasure-kernel-s-finite-measure-algebra[measurable]:
 $A \in \text{sets } N \implies (\lambda a. \text{emeasure } (K a) A) \in \text{borel-measurable } M$
using measurable-compose[OF K measurable-emeasure-s-finite-measure-algebra] .

end

lemma measurable-s-finite-measure-algebra:
 $(\bigwedge a. a \in \text{space } M \implies \text{s-finite-measure } (K a)) \implies$
 $(\bigwedge a. a \in \text{space } M \implies \text{sets } (K a) = \text{sets } N) \implies$
 $(\bigwedge A. A \in \text{sets } N \implies (\lambda a. \text{emeasure } (K a) A) \in \text{borel-measurable } M) \implies$
 $K \in \text{measurable } M \text{ (s-finite-measure-algebra } N)$
by (auto intro!: measurable-Sup2 measurable-vimage-algebra2 simp: s-finite-measure-algebra-def)

definition bind-kernel :: 'a measure \Rightarrow ('a \Rightarrow 'b measure) \Rightarrow 'b measure (**infixl**
 \ggg_k 54) **where**
bind-kernel M k = (if space M = {} then count-space {} else
let Y = k (SOME x. x \in space M) in
measure-of (space Y) (sets Y) ($\lambda B. \int^+ x. (k x B) \partial M$))

lemma bind-kernel-cong-All:
assumes $\bigwedge x. x \in \text{space } M \implies f x = g x$
shows $M \ggg_k f = M \ggg_k g$
proof(cases space M = {})
case 1:False
have (SOME x. x \in space M) \in space M
by (rule someI-ex) (use 1 **in** blast)
with assms **have** [simp]: f (SOME x. x \in space M) = g (SOME x. x \in space M)
by simp
have ($\lambda B. \int^+ x. \text{emeasure } (f x) B \partial M$) = ($\lambda B. \int^+ x. \text{emeasure } (g x) B \partial M$)
by standard (auto intro!: nn-integral-cong simp: assms)
thus ?thesis
by(auto simp: bind-kernel-def 1)
qed(simp add: bind-kernel-def)

lemma sets-bind-kernel:
assumes $\bigwedge x. x \in \text{space } M \implies \text{sets } (k x) = \text{sets } N \text{ space } M \neq \{\}$
shows $\text{sets } (M \ggg_k k) = \text{sets } N$
proof –
have $\text{sets } (k (\text{SOME } x. x \in \text{space } M)) = \text{sets } N$
by(rule someI2-ex) (use assms **in** auto)
with sets-eq-imp-space-eq[OF this] **show** ?thesis
by(simp add: bind-kernel-def assms(2))
qed

2.2 Measure Kernel

locale *measure-kernel* =

fixes $X :: 'a \text{ measure}$ **and** $Y :: 'b \text{ measure}$ **and** $\kappa :: 'a \Rightarrow 'b \text{ measure}$
assumes *kernel-sets*[*measurable-cong*]: $\bigwedge x. x \in \text{space } X \Longrightarrow \text{sets } (\kappa x) = \text{sets } Y$
and *emeasure-measurable*[*measurable*]: $\bigwedge B. B \in \text{sets } Y \Longrightarrow (\lambda x. \text{emeasure } (\kappa x) B) \in \text{borel-measurable } X$
and *Y-not-empty*: $\text{space } X \neq \{\} \Longrightarrow \text{space } Y \neq \{\}$
begin

lemma *kernel-space* : $\bigwedge x. x \in \text{space } X \Longrightarrow \text{space } (\kappa x) = \text{space } Y$
by (*meson kernel-sets sets-eq-imp-space-eq*)

lemma *measure-measurable*:

assumes $B \in \text{sets } Y$
shows $(\lambda x. \text{measure } (\kappa x) B) \in \text{borel-measurable } X$
using *emeasure-measurable*[*OF assms*] **by** (*simp add: Sigma-Algebra.measure-def*)

lemma *set-nn-integral-measure*:

assumes [*measurable-cong*]: $\text{sets } \mu = \text{sets } X$ **and** [*measurable*]: $A \in \text{sets } X \ B \in \text{sets } Y$

defines $\nu \equiv \text{measure-of } (\text{space } Y) (\text{sets } Y) (\lambda B. \int^{+x \in A. (\kappa x) B} \partial \mu)$
shows $\nu B = (\int^{+x \in A. (\kappa x) B} \partial \mu)$

proof -

have *nu-sets*[*measurable-cong*]: $\text{sets } \nu = \text{sets } Y$
by(*simp add: nu-def*)

have *positive* ($\text{sets } Y$) $(\lambda B. \int^{+x \in A. (\kappa x) B} \partial \mu)$
by(*simp add: positive-def*)

moreover have *countably-additive* ($\text{sets } Y$) $(\lambda B. \int^{+x \in A. (\kappa x) B} \partial \mu)$
unfolding *countably-additive-def*

proof *safe*

fix $C :: \text{nat} \Rightarrow -$

assume $h: \text{range } C \subseteq \text{sets } Y \text{ disjoint-family } C$

thus $(\sum i. \int^{+x \in A. (\kappa x) (C i)} \partial \mu) = (\int^{+x \in A. (\kappa x) (\bigcup (\text{range } C))} \partial \mu)$

by(*auto intro!: nn-integral-cong simp: sets-eq-imp-space-eq[OF assms(1)] kernel-sets suminf-emeasure nn-integral-suminf[symmetric]*)

qed

ultimately show *?thesis*

using *nu-def assms(3) emeasure-measure-of-sigma sets.sigma-algebra-axioms* **by**
blast

qed

corollary *nn-integral-measure*:

assumes $\text{sets } \mu = \text{sets } X \ B \in \text{sets } Y$

defines $\nu \equiv \text{measure-of } (\text{space } Y) (\text{sets } Y) (\lambda B. \int^{+x. (\kappa x) B} \partial \mu)$

shows $\nu B = (\int^{+x. (\kappa x) B} \partial \mu)$

using *set-nn-integral-measure*[*OF assms(1) sets.top assms(2)*]

by(*simp add: nu-def sets-eq-imp-space-eq[OF assms(1),symmetric]*)

lemma *distr-measure-kernel*:

assumes [measurable]: $f \in Y \rightarrow_M Z$
shows *measure-kernel* $X Z (\lambda x. \text{distr } (\kappa x) Z f)$
unfolding *measure-kernel-def*
proof *safe*
fix B
assume B [measurable]: $B \in \text{sets } Z$
show $(\lambda x. \text{emeasure } (\text{distr } (\kappa x) Z f) B) \in \text{borel-measurable } X$
by(*rule measurable-cong*[**where** $g = (\lambda x. \kappa x (f - ' B \cap \text{space } Y))$],*THEN iffD2*)
(*auto simp: emeasure-distr sets-eq-imp-space-eq*[*OF kernel-sets*])
next
show $\bigwedge x. \text{space } Z = \{\} \implies x \in \text{space } X \implies x \in \{\}$
by (*metis Y-not-empty assms measurable-empty-iff*)
qed *auto*

lemma *measure-kernel-comp*:
assumes [measurable]: $f \in W \rightarrow_M X$
shows *measure-kernel* $W Y (\lambda x. \kappa (f x))$
using *measurable-space*[*OF assms*] *kernel-sets Y-not-empty*
by(*auto simp: measure-kernel-def*)

lemma *emeasure-bind-kernel*:
assumes *sets* $\mu = \text{sets } X B \in \text{sets } Y \text{space } X \neq \{\}$
shows $(\mu \ggg_k \kappa) B = (\int^+ x. (\kappa x B) \partial\mu)$
proof –
have *sets* $(\kappa (\text{SOME } x. x \in \text{space } \mu)) = \text{sets } Y$
by(*rule someI2-ex*) (*use assms(3) kernel-sets sets-eq-imp-space-eq*[*OF assms(1)*])
in *auto*)
with *sets-eq-imp-space-eq*[*OF this*] **show** *?thesis*
by(*simp add: bind-kernel-def sets-eq-imp-space-eq*[*OF assms(1)*] *assms(3) nn-integral-measure*[*OF assms(1,2)*])
qed

lemma *measure-bind-kernel*:
assumes [measurable-cong]: *sets* $\mu = \text{sets } X$ **and** [measurable]: $B \in \text{sets } Y \text{space } X \neq \{\}$ $A E x \text{ in } \mu. \kappa x B < \infty$
shows *measure* $(\mu \ggg_k \kappa) B = (\int x. \text{measure } (\kappa x) B \partial\mu)$
using *assms(4)* **by**(*auto simp: emeasure-bind-kernel*[*OF assms(1-3)*] *measure-def integral-eq-nn-integral intro!: arg-cong*[*of - - enn2real*] *nn-integral-cong-AE*)

lemma *sets-bind-kernel*:
assumes *space* $X \neq \{\}$ *sets* $\mu = \text{sets } X$
shows *sets* $(\mu \ggg_k \kappa) = \text{sets } Y$
using *sets-bind-kernel*[*of* $\mu \kappa$, *OF kernel-sets,simplified sets-eq-imp-space-eq*[*OF assms(2)*]]
by(*auto simp: assms(1)*)

lemma *distr-bind-kernel*:
assumes *space* $X \neq \{\}$ **and** [measurable-cong]: *sets* $\mu = \text{sets } X$ **and** [measurable]: $f \in Y \rightarrow_M Z$

shows $\text{distr } (\mu \gg_k \kappa) Z f = \mu \gg_k (\lambda x. \text{distr } (\kappa x) Z f)$
proof –
{
 fix A
 assume $A[\text{measurable}]: A \in \text{sets } Z$
 have $\text{sets}[\text{measurable-cong}]: \text{sets } (\mu \gg_k \kappa) = \text{sets } Y$
 by($\text{rule sets-bind-kernel}[OF \text{ assms}(1,2)]$)
 have $\text{emeasure } (\text{distr } (\mu \gg_k \kappa) Z f) A = \text{emeasure } (\mu \gg_k (\lambda x. \text{distr } (\kappa x) Z f)) A$ (**is** $?lhs = ?rhs$)
 proof –
 have $?lhs = (\int^+ x. \text{emeasure } (\kappa x) (f - ' A \cap \text{space } Y) \partial\mu)$
 by($\text{simp add: emeasure-distr sets-eq-imp-space-eq}[OF \text{ sets}] \text{emeasure-bind-kernel}[OF \text{ assms}(2) - \text{assms}(1)]$)
 also have $\dots = (\int^+ x. \text{emeasure } (\text{distr } (\kappa x) Z f) A \partial\mu)$
 by($\text{auto simp: emeasure-distr sets-eq-imp-space-eq}[OF \text{ assms}(2)] \text{sets-eq-imp-space-eq}[OF \text{ kernel-sets}] \text{intro!}: \text{nn-integral-cong}$)
 also have $\dots = ?rhs$
 by($\text{simp add: measure-kernel.emeasure-bind-kernel}[OF \text{ distr-measure-kernel}[OF \text{ assms}(3)] \text{ assms}(2) - \text{assms}(1)]$)
 finally show $?thesis$.
 qed
}
thus $?thesis$
by($\text{auto intro!}: \text{measure-eqI simp: measure-kernel.sets-bind-kernel}[OF \text{ distr-measure-kernel}[OF \text{ assms}(3)] \text{ assms}(1,2)]$)
qed

lemma *bind-kernel-distr*:
assumes $[\text{measurable}]: f \in W \rightarrow_M X$ **and** $\text{space } W \neq \{\}$
shows $\text{distr } W X f \gg_k \kappa = W \gg_k (\lambda x. \kappa (f x))$
proof –
 have $X: \text{space } X \neq \{\}$
 using $\text{measurable-space}[OF \text{ assms}(1)] \text{ assms}(2)$ **by** *auto*
 show $?thesis$
 by($\text{rule measure-eqI, insert } X$) ($\text{auto simp: sets-bind-kernel}[OF X] \text{measure-kernel.sets-bind-kernel}[OF \text{ measure-kernel-comp}[OF \text{ assms}(1)] \text{ assms}(2) \text{ refl}] \text{emeasure-bind-kernel} \text{nn-integral-distr} \text{measure-kernel.emeasure-bind-kernel}[OF \text{ measure-kernel-comp}[OF \text{ assms}(1)] \text{ refl} - \text{assms}(2)]$)
qed

lemma *bind-kernel-return*:
assumes $x \in \text{space } X$
shows $\text{return } X x \gg_k \kappa = \kappa x$
proof –
 have $X: \text{space } X \neq \{\}$
 using assms **by** *auto*
 show $?thesis$
 by(rule measure-eqI) ($\text{auto simp: sets-bind-kernel}[OF X \text{ sets-return}] \text{kernel-sets}[OF \text{ assms}] \text{emeasure-bind-kernel}[OF \text{ sets-return, simplified, OF } - X] \text{nn-integral-return}[OF$

```

assms])
qed

lemma kernel-nn-integral-measurable:
  assumes f ∈ borel-measurable Y
  shows (λx. ∫+ y. f y ∂(κ x)) ∈ borel-measurable X
  using assms
proof induction
  case (cong f g)
  then show ?case
    by(auto intro!: measurable-cong[where f=λx. ∫+ y. f y ∂(κ x) and g=λx.
    ∫+ y. g y ∂(κ x), THEN iffD2] nn-integral-cong simp: sets-eq-imp-space-eq[OF ker-
    nel-sets])
  next
  case (set A)
  then show ?case
    by(auto intro!: measurable-cong[where f=λx. integralN (κ x) (indicator A)
    and g=λx. κ x A, THEN iffD2])
  next
  case (mult u c)
  then show ?case
    by(auto intro!: measurable-cong[where f=λx. ∫+ y. c * u y ∂κ x and g=λx.
    c * ∫+ y. u y ∂κ x, THEN iffD2] simp: nn-integral-cmult)
  next
  case (add u v)
  then show ?case
    by(auto intro!: measurable-cong[where f=λx. ∫+ y. v y + u y ∂κ x and g=λx.
    (∫+ y. v y ∂κ x) + (∫+ y. u y ∂κ x), THEN iffD2] simp: nn-integral-add)
  next
  case (seq U)
  then show ?case
    by(intro measurable-cong[where f=λx. integralN (κ x) (⊔ range U) and g=λx.
    ⊔ i. integralN (κ x) (U i), THEN iffD2])
    (auto simp: nn-integral-monotone-convergence-SUP[of U, simplified SUP-apply[symmetric]])
qed

```

```

lemma bind-kernel-measure-kernel:
  assumes measure-kernel Y Z k'
  shows measure-kernel X Z (λx. κ x ≫k k')
proof(cases space X = {})
  case True
  then show ?thesis
    by(auto simp: measure-kernel-def measurable-def)
  next
  case X:False
  then have Y: space Y ≠ {}
    by(simp add: Y-not-empty)
  interpret k': measure-kernel Y Z k' by fact
  show ?thesis

```


proof
fix B
assume $B \in \text{sets } Z$
with $k'. \text{emeasure-bind-kernel}[OF \text{ kernel-sets, of } - B]$ **show** $(\lambda x. \text{emeasure } (\kappa x \gg_k k') B) \in \text{borel-measurable } X$
by $(\text{auto intro!}: \text{measurable-cong}[\text{where } f = \lambda x. \text{emeasure } (\kappa x \gg_k k') B \text{ and } g = \lambda x. \int^+ y. \text{emeasure } (k' y) B \partial \kappa x, \text{THEN iffD2}] \text{ kernel-nn-integral-measurable simp: sets-eq-imp-space-eq}[OF \text{ kernel-sets}] Y)$
qed $(\text{use } k'. Y\text{-not-empty } Y \text{ } k'. \text{sets-bind-kernel}[OF Y \text{ kernel-sets}] \text{ in auto})$
qed

lemma *restrict-measure-kernel: measure-kernel (restrict-space X A) Y κ*

proof
fix B
assume $B \in \text{sets } Y$
from $\text{emeasure-measurable}[OF \text{ this}]$ **show** $(\lambda x. \text{emeasure } (\kappa x) B) \in \text{borel-measurable } (\text{restrict-space } X A)$
using *measurable-restrict-space1* **by** *blast*
qed $(\text{insert } Y\text{-not-empty, auto simp add: space-restrict-space kernel-sets})$

end

lemma *measure-kernel-cong-sets:*

assumes $\text{sets } X = \text{sets } X' \text{ sets } Y = \text{sets } Y'$
shows $\text{measure-kernel } X Y = \text{measure-kernel } X' Y'$
by *standard (simp add: measure-kernel-def measurable-cong-sets[OF assms(1) refl] sets-eq-imp-space-eq[OF assms(1)] assms(2) sets-eq-imp-space-eq[OF assms(2)])*

lemma *measure-kernel-pair-countble1:*

assumes $\text{countable } A \wedge i. i \in A \implies \text{measure-kernel } X Y (\lambda x. k (i, x))$
shows $\text{measure-kernel } (\text{count-space } A \otimes_M X) Y k$
using *assms* **by** $(\text{auto simp: measure-kernel-def space-pair-measure intro!: measurable-pair-measure-countable1})$

lemma *measure-kernel-empty-trivial:*

assumes $\text{space } X = \{\}$
shows $\text{measure-kernel } X Y k$
using *assms* **by** $(\text{auto simp: measure-kernel-def measurable-def})$

2.3 Finite Kernel

locale *finite-kernel* = *measure-kernel* +

assumes *finite-measure-spaces*: $\exists r < \infty. \forall x \in \text{space } X. \kappa x (\text{space } Y) < r$
begin

lemma *finite-measures:*

assumes $x \in \text{space } X$
shows *finite-measure* (κx)

proof –

obtain r **where** κx (*space* Y) $< r$
using *finite-measure-spaces* *assms* **by** *metis*
then show *?thesis*
by(*auto intro!*: *finite-measureI simp: sets-eq-imp-space-eq*[*OF kernel-sets* [*OF assms*]])
qed
end

lemma *finite-kernel-empty-trivial*: *space* $X = \{\}$ \implies *finite-kernel* $X Y f$
by(*auto simp: finite-kernel-def finite-kernel-axioms-def measure-kernel-empty-trivial intro!*: *exI*[**where** $x=1$])

lemma *finite-kernel-cong-sets*:
assumes *sets* $X = \text{sets } X'$ *sets* $Y = \text{sets } Y'$
shows *finite-kernel* $X Y = \text{finite-kernel } X' Y'$
by *standard* (*auto simp: measure-kernel-cong-sets*[*OF assms*] *finite-kernel-def finite-kernel-axioms-def sets-eq-imp-space-eq*[*OF assms*(1)] *sets-eq-imp-space-eq*[*OF assms*(2)])

2.4 Sub-Probability Kernel

locale *subprob-kernel* = *measure-kernel* +
assumes *subprob-spaces*: $\bigwedge x. x \in \text{space } X \implies \text{subprob-space } (\kappa x)$
begin
lemma *subprob-space*:
 $\bigwedge x. x \in \text{space } X \implies \kappa x$ (*space* Y) ≤ 1
by (*simp add: subprob-space.subprob-emeasure-le-1 subprob-spaces*)

lemma *subprob-measurable*[*measurable*]:
 $\kappa \in X \rightarrow_M \text{subprob-algebra } Y$
by(*auto intro!*: *measurable-subprob-algebra-generated*[*OF sets.sigma-sets-eq*[*symmetric*]
sets.Int-stable sets.space-closed] *simp: subprob-spaces kernel-sets emeasure-measurable*)

lemma *finite-kernel*: *finite-kernel* $X Y \kappa$
by(*auto simp: finite-kernel-def finite-kernel-axioms-def intro!*: *measure-kernel-axioms exI*[**where** $x=2$] *order.strict-trans1*[*OF subprob-space.subprob-emeasure-le-1* [*OF subprob-spaces*]])

sublocale *finite-kernel*
by (*rule finite-kernel*)

end

lemma *subprob-kernel-def'*:
subprob-kernel $X Y \kappa \longleftrightarrow \kappa \in X \rightarrow_M \text{subprob-algebra } Y$
by(*auto simp: subprob-kernel.subprob-measurable subprob-kernel-def subprob-kernel-axioms-def measure-kernel-def measurable-subprob-algebra measurable-empty-iff space-subprob-algebra-empty-iff*)
(*auto simp: subprob-measurableD*(2) *subprob-space-kernel*)

lemmas *subprob-kernelI* = *measurable-subprob-algebra*[*simplified subprob-kernel-def'*[*symmetric*]]

lemma *subprob-kernel-cong-sets*:

assumes *sets X = sets X' sets Y = sets Y'*
shows *subprob-kernel X Y = subprob-kernel X' Y'*
by *standard (auto simp: subprob-kernel-def' subprob-algebra-cong[OF assms(2)] measurable-cong-sets[OF assms(1) refl])*

lemma *subprob-kernel-empty-trivial*:

assumes *space X = {}*
shows *subprob-kernel X Y k*
using *assms by(auto simp: subprob-kernel-def subprob-kernel-axioms-def intro!: measure-kernel-empty-trivial)*

lemma *bind-kernel-bind*:

assumes *f ∈ M →_M subprob-algebra N*
shows *M ≫_k f = M ≫ f*
proof(*cases space M = {}*)
case *True*
then show *?thesis*
by(*simp add: bind-kernel-def bind-def*)
next
case *h:False*
interpret *subprob-kernel M N f*
using *assms(1) by(simp add: subprob-kernel-def')*
show *?thesis*
by(*rule measure-eqI,insert sets-kernel[OF assms] (auto simp: h sets-bind-kernel emeasure-bind-kernel[OF refl - h] emeasure-bind[OF h assms])*)
qed

lemma(**in** *measure-kernel*) *subprob-kernel-sum*:

assumes $\bigwedge x. x \in \text{space } X \implies \text{finite-measure } (\kappa x)$
obtains *ki* **where** $\bigwedge i. \text{subprob-kernel } X Y (ki\ i) \bigwedge A x. x \in \text{space } X \implies \kappa x A = (\sum i. ki\ i\ x\ A)$
proof –
obtain *u* **where** $\bigwedge x. x \in \text{space } X \implies u\ x < \infty \bigwedge x. x \in \text{space } X \implies u\ x = \kappa x (\text{space } Y)$
using *finite-measure.emeasure-finite[OF assms]*
by (*simp add: top.not-eq-extremum*)
have [*measurable*]: *u ∈ borel-measurable X*
by(*simp cong: measurable-cong add: u(2)*)
define *ki* **where** $ki \equiv (\lambda i x. \text{if } i < \text{nat } \lceil \text{enn2real } (u\ x) \rceil \text{ then } \text{scale-measure } (1 / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u\ x) \rceil)) (\kappa x) \text{ else } (\text{sigma } (\text{space } Y) (\text{sets } Y)))$
have $1: \bigwedge i x. x \in \text{space } X \implies \text{sets } (ki\ i\ x) = \text{sets } Y$
by(*auto simp: ki-def kernel-sets*)
have *subprob-kernel X Y (ki i)* **for** *i*
proof –
{

```

fix  $i B$ 
assume [measurable]:  $B \in \text{sets } Y$ 
have  $(\lambda x. \text{emeasure } (ki\ i\ x)\ B) = (\lambda x. \text{if } i < \text{nat } \lceil \text{enn2real } (u\ x) \rceil \text{ then } (1 / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u\ x) \rceil)) * \text{emeasure } (\kappa\ x)\ B \text{ else } 0)$ 
  by(auto simp: ki-def emeasure-sigma)
also have  $\dots \in \text{borel-measurable } X$ 
  by simp
finally have  $(\lambda x. \text{emeasure } (ki\ i\ x)\ B) \in \text{borel-measurable } X$  .
}
moreover {
  fix  $i x$ 
  assume  $x : x \in \text{space } X$ 
  have  $\text{emeasure } (ki\ i\ x)\ (\text{space } Y) \leq 1$ 
    by(cases u x = 0, auto simp: ki-def emeasure-sigma u(2)[OF x, symmetric])
    (metis u(1)[OF x, simplified] divide-ennreal-def divide-le-posI-ennreal enn2real-le-le-of-int-ceiling mult.commute mult.right-neutral not-gr-zero order.strict-iff-not)
    hence subprob-space  $(ki\ i\ x)$ 
    using  $x\ Y\ \text{not-empty}$  by(fastforce intro!: subprob-spaceI simp: sets-eq-imp-space-eq[OF 1[OF x]])
  }
  ultimately show ?thesis
    by(auto simp: subprob-kernel-def measure-kernel-def 1 Y-not-empty subprob-kernel-axioms-def)
qed
moreover have  $\kappa\ x\ A = (\sum i. ki\ i\ x\ A)$  if  $x : x \in \text{space } X$  for  $x\ A$ 
proof (cases A ∈ sets Y)
  case  $A[\text{measurable}]: \text{True}$ 
  have  $\text{emeasure } (\kappa\ x)\ A = (\sum i < \text{nat } \lceil \text{enn2real } (u\ x) \rceil. \text{emeasure } (ki\ i\ x)\ A)$ 
  proof(cases u x = 0)
    case True
    then show ?thesis
      using  $u(2)[OF\ that]$  by simp (metis A emeasure-eq-0 kernel-sets sets.sets-into-space sets.top x)
    next
      case  $u0:\text{False}$ 
      hence  $\text{real-of-int } \lceil \text{enn2real } (u\ x) \rceil > 0$ 
        by (metis enn2real-nonneg ennreal-0 ennreal-enn2real-if infinity-ennreal-def linorder-not-le nat-0-iff nle-le of-int-le-0-iff of-nat-eq-0-iff real-nat-ceiling-ge u(1) x)
        with  $u(1)[OF\ x]$  have  $\text{of-nat } (\text{nat } \lceil \text{enn2real } (u\ x) \rceil) / \text{ennreal } (\text{real-of-int } \lceil \text{enn2real } (u\ x) \rceil) = 1$ 
          by(simp add: ennreal-eq-0-iff ennreal-of-nat-eq-real-of-nat)
          thus ?thesis
          by(simp add: ki-def ennreal-divide-times[symmetric] mult.assoc[symmetric])
        qed
        then show ?thesis
          by(auto simp: suminf-offset[of λi. emeasure (ki i x) A nat ⌈enn2real (u x)⌉])
          (simp add: ki-def emeasure-sigma)
        next

```

```

    case False
  then show ?thesis
    using kernel-sets[OF x] 1[OF x]
    by(simp add: emeasure-notin-sets)
  qed
  ultimately show ?thesis
    using that by blast
  qed

```

2.5 Probability Kernel

```

locale prob-kernel = measure-kernel +
  assumes prob-spaces:  $\bigwedge x. x \in \text{space } X \implies \text{prob-space } (\kappa x)$ 
begin

```

```

lemma prob-space:
 $\bigwedge x. x \in \text{space } X \implies \kappa x (\text{space } Y) = 1$ 
  using kernel-space prob-space.emeasure-space-1 prob-spaces by fastforce

```

```

lemma prob-measurable[measurable]:
 $\kappa \in X \rightarrow_M \text{prob-algebra } Y$ 
  by(auto intro!: measurable-prob-algebra-generated[OF sets.sigma-sets-eq[symmetric]
sets.Int-stable sets.space-closed] simp: prob-spaces kernel-sets emeasure-measurable)

```

```

lemma subprob-kernel: subprob-kernel X Y  $\kappa$ 
  by (simp add: measurable-prob-algebraD subprob-kernel-def)

```

```

sublocale subprob-kernel
  by (simp add: subprob-kernel)

```

```

lemma restrict-probability-kernel:
  prob-kernel (restrict-space X A) Y  $\kappa$ 
  by(auto simp: prob-kernel-def restrict-measure-kernel prob-kernel-axioms-def space-restrict-space
  prob-spaces)

```

end

```

lemma prob-kernel-def':
  prob-kernel X Y  $\kappa \longleftrightarrow \kappa \in X \rightarrow_M \text{prob-algebra } Y$ 

```

proof

```

  assume h:  $\kappa \in X \rightarrow_M \text{prob-algebra } Y$ 
  show prob-kernel X Y  $\kappa$ 
    using subprob-measurableD(2)[OF measurable-prob-algebraD[OF h]] measurable-
    space[OF h] measurable-emeasure-kernel[OF measurable-prob-algebraD[OF h]]
    by(auto simp: prob-kernel-def measure-kernel-def prob-kernel-axioms-def space-prob-algebra
    ) (metis prob-space.not-empty sets-eq-imp-space-eq)
  qed(auto simp: prob-kernel.prob-measurable prob-kernel-def prob-kernel-axioms-def
  measure-kernel-def)

```

```

lemma bind-kernel-return'':
  assumes sets M = sets N
  shows  $M \gg_k \text{return } N = M$ 
proof(cases space M = {})
  case True
  then show ?thesis
    by(simp add: bind-kernel-def space-empty[symmetric])
next
  case False
  then have  $1: \text{space } N \neq \{\}$ 
    by(simp add: sets-eq-imp-space-eq[OF assms])
  interpret prob-kernel N N return N
    by(simp add: prob-kernel-def')
  show ?thesis
    by(rule measure-eqI) (auto simp: emeasure-bind-kernel[OF assms - 1] sets-bind-kernel[OF
  1 assms] assms)
qed

```

2.6 S-Finite Kernel

```

locale s-finite-kernel = measure-kernel +
  assumes s-finite-kernel-sum:  $\exists ki. (\forall i. \text{finite-kernel } X Y (ki\ i) \wedge (\forall x \in \text{space } X. \forall A \in \text{sets } Y. \kappa\ x\ A = (\sum i. ki\ i\ x\ A)))$ 

```

```

lemma s-finite-kernel-subI:
  assumes  $\bigwedge x. x \in \text{space } X \implies \text{sets } (\kappa\ x) = \text{sets } Y \bigwedge i. \text{subprob-kernel } X Y (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies A \in \text{sets } Y \implies \text{emeasure } (\kappa\ x)\ A = (\sum i. ki\ i\ x\ A)$ 
  shows s-finite-kernel X Y  $\kappa$ 
proof –
  interpret measure-kernel X Y  $\kappa$ 
  proof
    show  $B \in \text{sets } Y \implies (\lambda x. \text{emeasure } (\kappa\ x)\ B) \in \text{borel-measurable } X$  for  $B$ 
      using assms(2) by(simp add: assms(3) subprob-kernel-def' cong: measurable-cong)
    next
      show  $\text{space } X \neq \{\} \implies \text{space } Y \neq \{\}$ 
        using assms(2)[of 0] by(auto simp: subprob-kernel-def measure-kernel-def)
    qed fact
    show ?thesis
      by (auto simp: s-finite-kernel-def measure-kernel-axioms s-finite-kernel-axioms-def
      assms(2,3) intro!: exI[where x=ki] subprob-kernel.finite-kernel)
  qed

```

```

context s-finite-kernel
begin

```

```

lemma s-finite-kernels-fin:
  obtains ki where  $\bigwedge i. \text{finite-kernel } X Y (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies \kappa\ x\ A =$ 

```

$(\sum i. ki\ i\ x\ A)$

proof –

obtain ki **where** $ki: \bigwedge i. \text{finite-kernel } X\ Y\ (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies A \in \text{sets } Y \implies \kappa\ x\ A = (\sum i. ki\ i\ x\ A)$

by(*metis s-finite-kernel-sum*)

hence $\kappa\ x\ A = (\sum i. ki\ i\ x\ A)$ **if** $x \in \text{space } X$ **for** $x\ A$

by(*cases* $A \in \text{sets } Y$, *insert that kernel-sets[OF that]*) (*auto simp: finite-kernel-def measure-kernel-def emeasure-notin-sets*)

with ki **show** *?thesis*

using *that by auto*

qed

lemma *s-finite-kernels*:

obtains ki **where** $\bigwedge i. \text{subprob-kernel } X\ Y\ (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies \kappa\ x\ A = (\sum i. ki\ i\ x\ A)$

proof –

obtain ki **where** $ki: \bigwedge i. \text{finite-kernel } X\ Y\ (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies \kappa\ x\ A = (\sum i. ki\ i\ x\ A)$

by(*metis s-finite-kernels-fin*)

have $\exists kij. (\forall j. \text{subprob-kernel } X\ Y\ (kij\ j)) \wedge (\forall x\ A. x \in \text{space } X \longrightarrow ki\ i\ x\ A = (\sum j. kij\ j\ x\ A))$ **for** i

using *measure-kernel.subprob-kernel-sum[of X Y ki i, OF - finite-kernel.finite-measures[OF ki(1)[of i]] ki(1)[of i]* **by**(*metis finite-kernel-def*)

then obtain kij **where** $kij: \bigwedge i\ j. \text{subprob-kernel } X\ Y\ (kij\ i\ j) \bigwedge x\ A\ i. x \in \text{space } X \implies ki\ i\ x\ A = (\sum j. kij\ i\ j\ x\ A)$

by *metis*

have $\bigwedge i. \text{subprob-kernel } X\ Y\ (\text{case-prod } kij\ (\text{prod-decode } i))$

using $kij(1)$ **by**(*auto simp: split-beta*)

moreover have $x \in \text{space } X \implies \kappa\ x\ A = (\sum i. \text{case-prod } kij\ (\text{prod-decode } i)\ x\ A)$ **for** $x\ A$

using *suminf-ennreal-2dimen[of $\lambda i. ki\ i\ x\ A\ \lambda(i,j). kij\ i\ j\ x\ A$]*

by(*auto simp: ki(2) kij(2) split-beta'*)

ultimately show *?thesis*

using *that by fastforce*

qed

lemma *image-s-finite-measure*:

assumes $x \in \text{space } X$

shows *s-finite-measure* $(\kappa\ x)$

proof –

obtain ki **where** $ki: \bigwedge i. \text{subprob-kernel } X\ Y\ (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies \kappa\ x\ A = (\sum i. ki\ i\ x\ A)$

by(*metis s-finite-kernels*)

show *?thesis*

using $ki(1)$ [*simplified subprob-kernel-def*] *measurable-space*[*OF ki(1)[simplified subprob-kernel-def]*] *assms*]

by(*auto intro!: s-finite-measureI[where $Mi = \lambda i. ki\ i\ x$] subprob-space.axioms(1) simp: kernel-sets[OF assms] space-subprob-algebra ki(2)[OF assms]*)

qed

corollary *kernel-measurable-s-finite*[*measurable*]: $\kappa \in X \rightarrow_M s\text{-finite-measure-algebra}$
 Y
by(*auto intro!*: *measurable-s-finite-measure-algebra simp*: *kernel-sets image-s-finite-measure*)

lemma *comp-measurable*:

assumes *f*[*measurable*]: $f \in M \rightarrow_M X$
shows *s-finite-kernel* $M Y (\lambda x. \kappa (f x))$

proof –

obtain *ki* **where** $ki: \bigwedge i. \text{subprob-kernel } X Y (ki\ i) \wedge x A. x \in \text{space } X \implies \kappa x$
 $A = (\sum i. ki\ i\ x\ A)$
by(*metis s-finite-kernels*)
show *?thesis*
using *ki(1) measurable-space[OF f]* **by**(*auto intro!*: *s-finite-kernel-subI*[**where**
 $ki = \lambda i x. ki\ i\ (f\ x)$] *simp*: *subprob-kernel-def' ki(2) kernel-sets*)
qed

lemma *distr-s-finite-kernel*:

assumes *f*[*measurable*]: $f \in Y \rightarrow_M Z$
shows *s-finite-kernel* $X Z (\lambda x. \text{distr } (\kappa x) Z f)$

proof –

obtain *ki* **where** $ki: \bigwedge i. \text{subprob-kernel } X Y (ki\ i) \wedge x A. x \in \text{space } X \implies \kappa x$
 $A = (\sum i. ki\ i\ x\ A)$
by(*metis s-finite-kernels*)
hence $1: x \in \text{space } X \implies \text{space } (ki\ i\ x) = \text{space } Y$ **for** $x\ i$
by(*auto simp*: *subprob-kernel-def' intro!*: *subprob-measurableD(1)[of - X Y]*)
have [*measurable*]: $B \in \text{sets } Z \implies (\lambda x. \text{emeasure } (\text{distr } (\kappa x) Z f) B) \in \text{borel-measurable}$
 X **for** B
by(*rule measurable-cong*[**where** $g = \lambda x. \kappa x (f - ' B \cap \text{space } Y)$, *THEN iffD2*])
(*auto simp*: *emeasure-distr sets-eq-imp-space-eq[OF kernel-sets]*)
show *?thesis*
using *ki(1) measurable-distr[OF f]* **by**(*auto intro!*: *s-finite-kernel-subI*[**where**
 $ki = \lambda i x. \text{distr } (ki\ i\ x) Z f$] *simp*: *subprob-kernel-def' emeasure-distr ki(2) sets-eq-imp-space-eq[OF*
 $\text{kernel-sets}] 1$)
qed

lemma *comp-s-finite-measure*:

assumes *s-finite-measure* μ **and** [*measurable-cong*]: *sets* $\mu = \text{sets } X$
shows *s-finite-measure* $(\mu \gg_k \kappa)$

proof(*cases space X = {}*)

case $1: \text{True}$

show *?thesis*

by(*auto simp*: *sets-eq-imp-space-eq[OF assms(2)] 1 bind-kernel-def intro!*: *fi-*
nite-measure.s-finite-measure-finite-measure finite-measureI)

next

case $0: \text{False}$

then have $1: \text{space } \mu \neq \{\}$

by(*simp add*: *sets-eq-imp-space-eq[OF assms(2)]*)

have $2: \text{sets } (\kappa (\text{SOME } x. x \in \text{space } \mu)) = \text{sets } Y$


```

    by(rule someI2-ex, insert 1 kernel-sets) (auto simp: sets-eq-imp-space-eq[OF
assms(2)])
  have sets-bind[measurable-cong]: sets ( $\mu \gg_k \kappa$ ) = sets Y
    by(simp add: bind-kernel-def 1 sets-eq-imp-space-eq[OF 2] 2)
  obtain  $\mu i$  where mui[measurable-cong]:  $\bigwedge i. \text{sets } (\mu i) = \text{sets } X \wedge i. (\mu i) (\text{space } X) \leq 1 \wedge A. \mu A = (\sum i. \mu i A)$ 
    using s-finite-measure.finite-measures[OF assms(1)] assms(2) sets-eq-imp-space-eq[OF
assms(2)] by metis
  obtain ki where ki: $\bigwedge i. \text{subprob-kernel } X Y (ki i) \wedge x A. x \in \text{space } X \implies \kappa x A = (\sum i. ki i x A)$ 
    by(metis s-finite-kernels)
  define Mi where Mi  $\equiv (\lambda n. (\lambda(i,j). \text{measure-of } (\text{space } Y) (\text{sets } Y) (\lambda A. \int^+ x. (ki i x A) \partial(\mu i j))) (\text{prod-decode } n))$ 
  have emeasure:( $\mu \gg_k \kappa$ ) A = ( $\sum i. (Mi i) A$ ) (is ?lhs = ?rhs) if A  $\in$  sets Y
for A
  proof -
    have ?lhs = ( $\int^+ x. (\kappa x A) \partial\mu$ )
      by(simp add: emeasure-bind-kernel[OF assms(2) that 0])
    also have ... = ( $\int^+ x. (\sum i. (ki i x A)) \partial\mu$ )
      by(auto intro!: nn-integral-cong simp: ki sets-eq-imp-space-eq[OF assms(2)])
    also have ... = ( $\sum i. \int^+ x. (ki i x A) \partial\mu$ )
      by(auto intro!: nn-integral-suminf) (metis ki(1) assms(2) measurable-cong-sets
measure-kernel.emeasure-measurable subprob-kernel-def that)
    also have ... = ?rhs
      unfolding Mi-def
    proof(rule suminf-ennreal-2dimen[symmetric])
      fix m
      interpret kim: subprob-kernel X Y ki m
        by(simp add: ki)
      show ( $\int^+ x. (ki m x) A \partial\mu$ ) = ( $\sum n. \text{emeasure } (\text{case } (m, n) \text{ of } (i, j) \Rightarrow \text{measure-of } (\text{space } Y) (\text{sets } Y) (\lambda A. \int^+ x. \text{emeasure } (ki i x) A \partial(\mu i j)) A$ )
        using kim.emeasure-measurable[OF that] by(simp add: kim.nn-integral-measure[OF
mui(1) that] nn-integral-measure-suminf[OF mui(1)[simplified assms(2)[symmetric]]
mui(3)])
      qed
    finally show ?thesis .
  qed
  have fin:finite-measure (Mi i) for i
  proof(rule prod.exhaust[where y=prod-decode i])
    fix j1 j2
    interpret kij: subprob-kernel X Y ki j1
      by(simp add: ki)
    assume pd:prod-decode i = (j1, j2)
    have Mi i (space (Mi i)) = ( $\int^+ x. (ki j1 x (\text{space } Y)) \partial\mu i j2$ )
      by(auto simp: Mi-def pd kij.nn-integral-measure[OF mui(1) sets.top])
    also have ...  $\leq$  ( $\int^+ x. 1 \partial\mu i j2$ )
      by(intro nn-integral-mono) (metis kij.subprob-space mui(1) sets-eq-imp-space-eq)
    also have ...  $\leq 1$ 
      using mui by (simp add: sets-eq-imp-space-eq[OF mui(1)])
  end

```

finally show *finite-measure* ($Mi\ i$)
by (*metis ennreal-one-less-top finite-measureI infinity-ennreal-def less-le-not-le*)
qed
have \exists : *sets* ($Mi\ i$) = *sets* ($\mu \gg_k \kappa$) **for** i
by(*simp add: Mi-def split-beta sets-bind*)
show *s-finite-measure* ($\mu \gg_k \kappa$)
using *emeasure fin* \exists **by** (*auto intro!: exI[where x=Mi] simp: s-finite-measure-def sets-bind*)
qed

end

lemma *s-finite-kernel-empty-trivial*:

assumes *space* $X = \{\}$
shows *s-finite-kernel* $X\ Y\ k$
using *assms* **by**(*auto simp: s-finite-kernel-def s-finite-kernel-axioms-def intro!: measure-kernel-empty-trivial finite-kernel-empty-trivial*)

lemma *s-finite-kernel-def'*: *s-finite-kernel* $X\ Y\ \kappa \longleftrightarrow ((\forall x. x \in \text{space } X \longrightarrow \text{sets } (\kappa\ x) = \text{sets } Y) \wedge (\exists ki. (\forall i. \text{subprob-kernel } X\ Y\ (ki\ i)) \wedge (\forall x\ A. x \in \text{space } X \longrightarrow A \in \text{sets } Y \longrightarrow \text{emeasure } (\kappa\ x)\ A = (\sum i. ki\ i\ x\ A))))$ (**is** $?l \longleftrightarrow ?r$)

proof

assume $?l$
then interpret *s-finite-kernel* $X\ Y\ \kappa$.
from *s-finite-kernels* **obtain** ki **where** $ki: \bigwedge i. \text{subprob-kernel } X\ Y\ (ki\ i) \bigwedge x\ A. x \in \text{space } X \implies \text{emeasure } (\kappa\ x)\ A = (\sum i. \text{emeasure } (ki\ i\ x)\ A)$
by *metis*
thus $?r$
by(*auto simp: kernel-sets*)
qed(*auto intro!: s-finite-kernel-subI*)

lemma(**in** *finite-kernel*) *s-finite-kernel-finite-kernel*: *s-finite-kernel* $X\ Y\ \kappa$

proof

consider *space* $X = \{\} \mid \text{space } X \neq \{\}$ **by** *auto*
then show $\exists ki. \forall i. \text{finite-kernel } X\ Y\ (ki\ i) \wedge (\forall x \in \text{space } X. \forall A \in \text{sets } Y. (\kappa\ x)\ A = (\sum i. (ki\ i\ x)\ A))$
proof *cases*
case 1
then show $?thesis$
by(*auto simp: finite-kernel-def measure-kernel-def finite-kernel-axioms-def measurable-def intro!: exI[where x=0]*)
next
case 2
then have $y: \text{space } Y \neq \{\}$ **by**(*simp add: Y-not-empty*)
define ki **where** $ki\ i \equiv \text{case } i \text{ of } 0 \Rightarrow \kappa \mid \text{Suc } - \Rightarrow (\lambda-. \text{sigma } (\text{space } Y)\ (\text{sets } Y))$ **for** i
have *finite-kernel* $X\ Y\ (ki\ i)$ **for** i
by (*cases i, auto simp: ki-def finite-kernel-axioms*) (*auto simp: emeasure-sigma finite-kernel-def measure-kernel-def finite-kernel-axioms-def y intro!: finite-measureI*)

exI [where $x=1$])
moreover have $(\kappa x) A = (\sum i. (ki i x) A)$ **for** $x A$
by(*simp add: suminf-offset*[where $i=Suc\ 0$ and $f=\lambda i. ki\ i\ x\ A, simplified$],*simp*
add: ki-def emeasure-sigma)
ultimately show *?thesis* **by** *auto*
qed
qed

lemmas(in *subprob-kernel*) *s-finite-kernel-subprob-kernel = s-finite-kernel-finite-kernel*
lemmas(in *prob-kernel*) *s-finite-kernel-prob-kernel = s-finite-kernel-subprob-kernel*

sublocale *finite-kernel* \subseteq *s-finite-kernel*
by(*rule s-finite-kernel-finite-kernel*)

lemma *s-finite-kernel-cong-sets*:
assumes *sets X = sets X' sets Y = sets Y'*
shows *s-finite-kernel X Y = s-finite-kernel X' Y'*
by *standard (simp add: s-finite-kernel-def measurable-cong-sets[OF assms(1) refl]*
sets-eq-imp-space-eq[OF assms(1)] assms(2) measure-kernel-cong-sets[OF assms]
s-finite-kernel-axioms-def finite-kernel-cong-sets[OF assms])

lemma(in *s-finite-kernel*) *s-finite-kernel-cong*:
assumes $\bigwedge x. x \in \text{space } X \implies \kappa x = g x$
shows *s-finite-kernel X Y g*
using *assms s-finite-kernel-axioms* **by**(*auto simp: s-finite-kernel-def s-finite-kernel-axioms-def*
measure-kernel-def cong: measurable-cong)

lemma(in *s-finite-measure*) *s-finite-kernel-const*:
assumes *space M \neq {}*
shows *s-finite-kernel X M ($\lambda x. M$)*
proof
obtain Mi **where** $Mi: \bigwedge i. \text{sets } (Mi\ i) = \text{sets } M \bigwedge i. (Mi\ i) (\text{space } M) \leq 1 \bigwedge A.$
 $M A = (\sum i. Mi\ i\ A)$
by(*metis finite-measures*)
hence $\bigwedge i. \text{subprob-kernel } X\ M\ (\lambda x. Mi\ i)$
by(*auto simp: subprob-kernel-def' space-subprob-algebra sets-eq-imp-space-eq[OF*
Mi(1)] assms intro!: measurable-const subprob-spaceI)
thus $\exists ki. \forall i. \text{finite-kernel } X\ M\ (ki\ i) \wedge (\forall x \in \text{space } X. \forall A \in \text{sets } M. M A = (\sum i.$
 $(ki\ i\ x) A))$
by(*auto intro!: exI[where $x=\lambda i\ x. Mi\ i$] Mi(3) subprob-kernel.finite-kernel*)
qed (*auto simp: assms*)

lemma *s-finite-kernel-pair-countble1*:
assumes *countable A $\bigwedge i. i \in A \implies s-finite-kernel X Y (\lambda x. k (i,x))$*
shows *s-finite-kernel (count-space A $\otimes_M X$) Y k*
proof –
have $\exists ki. (\forall j. \text{subprob-kernel } X\ Y\ (ki\ j)) \wedge (\forall x B. x \in \text{space } X \implies B \in \text{sets}$
 $Y \implies k (i,x) B = (\sum j. ki\ j\ x\ B))$ **if** $i \in A$ **for** i
using *s-finite-kernel.s-finite-kernels*[*OF assms(2)*][*OF that*] **by** *metis*

then obtain ki **where** $ki: \bigwedge i j. i \in A \implies \text{subprob-kernel } X Y (ki\ i\ j) \bigwedge i x B. i \in A \implies x \in \text{space } X \implies B \in \text{sets } Y \implies k(i, x) B = (\sum j. ki\ i\ j\ x\ B)$
by *metis*
then show *?thesis*
using *assms(2)* **by**(*auto simp: s-finite-kernel-def' measure-kernel-pair-countble1 [OF assms(1)] subprob-kernel-def' space-pair-measure intro!: exI [where x= $\lambda j (i, x). ki\ i\ j\ x$] measurable-pair-measure-countable1 assms(1)*)
qed

lemma *s-finite-kernel-s-finite-kernel:*

assumes $\bigwedge i. s\text{-finite-kernel } X Y (ki\ i) \bigwedge x. x \in \text{space } X \implies \text{sets } (k\ x) = \text{sets } Y \bigwedge x A. x \in \text{space } X \implies A \in \text{sets } Y \implies \text{emeasure } (k\ x) A = (\sum i. (ki\ i) x A)$
shows *s-finite-kernel* $X Y k$

proof –

have $\exists kij. (\forall j. \text{subprob-kernel } X Y (kij\ j)) \wedge (\forall x A. x \in \text{space } X \longrightarrow ki\ i\ x\ A = (\sum j. kij\ j\ x\ A))$ **for** i

using *s-finite-kernel.s-finite-kernels [OF assms(1)] [of i]* **by** *metis*

then obtain kij **where** $kij: \bigwedge i j. \text{subprob-kernel } X Y (kij\ i\ j) \bigwedge i x A. x \in \text{space } X \implies ki\ i\ x\ A = (\sum j. kij\ i\ j\ x\ A)$

by *metis*

define ki' **where** $ki' \equiv (\lambda n. \text{case-prod } kij\ (\text{prod-decode } n))$

have $\text{emeasure-sum } k': \text{emeasure } (k\ x) A = (\sum i. \text{emeasure } (ki'\ i\ x) A)$ **if** $x: x \in \text{space } X$ **and** $A: A \in \text{sets } Y$ **for** $x A$

by(*auto simp: assms(3) [OF that] kij(2) [OF x] ki'-def intro!: suminf-ennreal-2dimen [symmetric]*)

have *subprob-kernel* $X Y (ki'\ i)$ **for** i

using *kij(1)* **by**(*auto simp: ki'-def split-beta'*)

thus *?thesis*

by(*auto simp: s-finite-kernel-def' measure-kernel-def assms(2) s-finite-kernel-axioms-def emeasure-sum k' intro!: exI [where x= ki']*)
qed

lemma *s-finite-kernel-finite-sumI:*

assumes [*measurable-cong*]: $\bigwedge x. x \in \text{space } X \implies \text{sets } (\kappa\ x) = \text{sets } Y$

and $\bigwedge i. i \in I \implies \text{subprob-kernel } X Y (ki\ i) \bigwedge x A. x \in \text{space } X \implies A \in \text{sets } Y \implies \text{emeasure } (\kappa\ x) A = (\sum i \in I. ki\ i\ x\ A)$ *finite* $I\ I \neq \{\}$

shows *s-finite-kernel* $X Y \kappa$

proof –

consider $\text{space } X = \{\}$ | $\text{space } X \neq \{\}$ **by** *auto*

then show *?thesis*

proof *cases*

case 1

then show *?thesis*

by(*rule s-finite-kernel-empty-trivial*)

next

case 2

then have $Y: \text{space } Y \neq \{\}$

using *assms measure-kernel.Y-not-empty* **by** (*fastforce simp: subprob-kernel-def*)

define ki' **where** $ki' \equiv (\lambda i x. \text{if } i < \text{card } I \text{ then } ki\ (from\text{-nat-into } I\ i)\ x \text{ else } null\text{-measure } Y)$

```

have [simp]:subprob-kernel X Y (ki' i) for i
  by(cases i < card I) (simp add: ki'-def from-nat-into assms, auto simp:
ki'-def subprob-kernel-def measure-kernel-def subprob-kernel-axioms-def Y intro!:
subprob-spaceI)
  have [simp]: (∑ i. emeasure (ki' i x) A) = (∑ i∈I. ki i x A) for x A
  using suminf-finite[of {..<card I} λi. (if i < card I then ki (from-nat-into I
i) x else null-measure Y) A]
  by(auto simp: sum.reindex-bij-betw[OF bij-betw-from-nat-into-finite[OF assms(4)],symmetric]
ki'-def)
  have [measurable]:B ∈ sets Y ⇒ (λx. emeasure (κ x) B) ∈ borel-measurable
X for B
  using assms(2) by(auto simp: assms(3) subprob-kernel-def' cong: measur-
able-cong)
  show ?thesis
  by (auto simp: s-finite-kernel-def' intro!: exI[where x=ki'] assms)
qed
qed

```

Each kernel does not need to be bounded by a uniform upper-bound in the definition of *s-finite-kernel*

lemma *s-finite-kernel-finite-bounded-sum*:

```

assumes [measurable-cong]: ∧x. x ∈ space X ⇒ sets (κ x) = sets Y
  and ∧i. measure-kernel X Y (ki i) ∧x A. x ∈ space X ⇒ A ∈ sets Y ⇒
κ x A = (∑ i. ki i x A) ∧i x. x ∈ space X ⇒ ki i x (space Y) < ∞
  shows s-finite-kernel X Y κ
proof(cases space X = {})
  case True
  then show ?thesis
  by(simp add: s-finite-kernel-empty-trivial)
next
  case X:False
  then have Y: space Y ≠ {}
  using assms(2)[of 0] by(simp add: measure-kernel-def)
  show ?thesis
  proof(rule s-finite-kernel-s-finite-kernel[where ki=ki,OF - assms(1) assms(3)])
  fix i
  interpret m: measure-kernel X Y ki i by fact
  define kij where kij ≡ (λ(j :: nat) x. if j < nat [enn2real (ki i x (space Y))])
then scale-measure (1 / ennreal [enn2real (ki i x (space Y))]) (ki i x) else sigma
(space Y) (sets Y))
  have sets-kij: sets (kij j x) = sets Y if x ∈ space X for j x
  by(auto simp: m.kernel-sets[OF that] kij-def)
  have emeasure-kij: ki i x A = (∑ j. kij j x A) if x ∈ space X A ∈ sets Y for x
A
  proof -
  have (∑ j. kij j x A) = (∑ j< nat [enn2real (ki i x (space Y))]. scale-measure
(1 / ennreal [enn2real (ki i x (space Y))]) (ki i x) A)
  by(simp add: suminf-offset[where i=nat [enn2real (ki i x (space Y))]] and
f=λj. kij j x A), simp add: kij-def emeasure-sigma)

```

```

also have ... =  $ki\ i\ x\ A$ 
proof(cases nat  $\lceil enn2real\ (ki\ i\ x\ (space\ Y)) \rceil$ )
  case 0
  then show ?thesis
    by simp (metis assms(4) emeasure-eq-0 enn2real-le ennreal-0 infinity-ennreal-def le-zero-eq linorder-not-le m.kernel-space nle-le sets.sets-into-space sets.top that)
  next
    case (Suc n')
    then have ennreal (real-of-int  $\lceil enn2real\ (emeasure\ (ki\ i\ x)\ (space\ Y)) \rceil$ ) > 0
      using ennreal-less-zero-iff by fastforce
    with assms(4)[OF that(1), of i] have [simp]: of-nat (nat  $\lceil enn2real\ (emeasure\ (ki\ i\ x)\ (space\ Y)) \rceil$ ) / ennreal (real-of-int  $\lceil enn2real\ (emeasure\ (ki\ i\ x)\ (space\ Y)) \rceil$ ) = 1
      by (simp add: ennreal-eq-0-iff ennreal-of-nat-eq-real-of-nat)
    show ?thesis
      by(simp add: mult.assoc[symmetric] ennreal-times-divide)
  qed
finally show ?thesis by simp
qed
have sk: subprob-kernel X Y (kij j) for j
proof –
  {
    fix B
    assume [measurable]: B  $\in$  sets Y
    have emeasure (kij j x) B = (if j < nat  $\lceil enn2real\ (ki\ i\ x\ (space\ Y)) \rceil$  then (ki i x) B / ennreal (real-of-int  $\lceil enn2real\ (ki\ i\ x\ (space\ Y)) \rceil$ ) else 0) if x  $\in$  space X for x
      by(auto simp: kij-def emeasure-sigma divide-ennreal-def mult.commute)
    hence ( $\lambda x$ . emeasure (kij j x) B)  $\in$  borel-measurable X
      by(auto simp: kij-def cong: measurable-cong)
  }
moreover {
  fix x
  assume x: x  $\in$  space X
  have subprob-space (kij j x)
  proof –
    have emeasure (kij j x) (space Y)  $\leq$  1
    proof –
      {
        assume 1: j < nat  $\lceil enn2real\ (emeasure\ (ki\ i\ x)\ (space\ Y)) \rceil$ 
        then have emeasure (ki i x) (space Y) > 0
        by (metis ceiling-zero enn2real-0 nat-zero-as-int not-gr-zero not-less-zero)
        with assms(4)[OF x] have [simp]: emeasure (ki i x) (space Y) / emeasure (ki i x) (space Y) = 1
          by simp
        have [simp]: emeasure (ki i x) (space Y) / ennreal (real-of-int  $\lceil enn2real\ (ki\ i\ x\ (space\ Y)) \rceil$ )  $\leq$  1

```

```

      proof(rule order.trans[where b=emeasure (ki i x) (space Y) / ki i x
(space Y),OF divide-le-posI-ennreal])
        show 0 < ennreal (real-of-int [enn2real (ki i x (space Y))])
          using 1 assms(4)[OF x] enn2real-positive-iff top.not-eq-extremum
    by fastforce
      next
        have 1:ennreal (real-of-int [enn2real (ki i x (space Y))]) ≥ ki i x
(space Y)
          using assms(4)[OF x] enn2real-le by (simp add: linorder-neq-iff)
        have ennreal (real-of-int [enn2real (ki i x (space Y))]) / ki i x (space
Y) ≥ 1
          by(rule order.trans[OF - divide-right-mono-ennreal[OF 1,of ki i x
(space Y)]] simp
            thus emeasure (ki i x) (space Y) ≤ ennreal (real-of-int [enn2real (ki
i x (space Y))]) * (emeasure (ki i x) (space Y) / ki i x (space Y))
              by (simp add: 1)
            qed simp
            have 1 / ennreal (real-of-int [enn2real (emeasure (ki i x) (space Y))])
* emeasure (ki i x) (space Y) ≤ 1
              by (simp add: ennreal-divide-times)
            }
            thus ?thesis
              by(auto simp: kij-def emeasure-sigma)
            qed
            thus ?thesis
              by(auto intro!: subprob-spaceI simp: sets-eq-imp-space-eq[OF sets-kij[OF
x,of j]] Y)
            qed
          }
          ultimately show ?thesis
            by(auto simp: subprob-kernel-def measure-kernel-def sets-kij m.Y-not-empty
subprob-kernel-axioms-def)
          qed
          show s-finite-kernel X Y (ki i)
            by(auto intro!: s-finite-kernel-subI simp: emeasure-kij sk m.kernel-sets)
          qed simp-all
        qed
    qed

```

```

lemma(in measure-kernel) s-finite-kernel-finite-bounded:
  assumes  $\bigwedge x. x \in \text{space } X \implies \kappa x (\text{space } Y) < \infty$ 
  shows s-finite-kernel X Y  $\kappa$ 
proof(cases space X = {})
  case True
  then show ?thesis
    by(simp add: s-finite-kernel-empty-trivial)
  next
  case False
  then have Y:space Y  $\neq \{\}$  by(simp add: Y-not-empty)
  have measure-kernel X Y (case i of 0  $\implies \kappa$  | Suc x  $\implies \lambda x. \text{null-measure } Y$ ) for i

```

by(cases i , auto simp: measure-kernel-axioms) (auto simp: measure-kernel-def Y)
moreover have $\kappa x A = (\sum i. \text{emeasure } ((\text{case } i \text{ of } 0 \Rightarrow \kappa \mid \text{Suc } x \Rightarrow \lambda x. \text{null-measure } Y) x) A)$ **for** $x A$
by(simp add: suminf-offset[**where** $i=\text{Suc } 0$])
moreover have $x \in \text{space } X \implies \text{emeasure } ((\text{case } i \text{ of } 0 \Rightarrow \kappa \mid \text{Suc } x \Rightarrow \lambda x. \text{null-measure } Y) x) (\text{space } Y) < \top$ **for** $x i$
by(cases i) (use assms in auto)
ultimately show ?thesis
by(auto intro!: s-finite-kernel-finite-bounded-sum[**where** $ki=\lambda i. \text{case } i \text{ of } 0 \Rightarrow \kappa \mid \text{Suc } - \Rightarrow (\lambda x. \text{null-measure } Y)$ and $X=X$ and $Y=Y$] simp: kernel-sets)
qed

lemma(in s-finite-kernel) density-s-finite-kernel:
assumes $f[\text{measurable}]$: case-prod $f \in X \otimes_M Y \rightarrow_M \text{borel}$
shows s-finite-kernel $X Y (\lambda x. \text{density } (\kappa x) (f x))$
proof(cases space $X = \{\}$)
case True
then show ?thesis
by(simp add: s-finite-kernel-empty-trivial)
next
case False
note $Y = Y\text{-not-empty}$ [OF this]
obtain ki' **where** ki' : $\bigwedge i. \text{subprob-kernel } X Y (ki' i) \bigwedge x A. x \in \text{space } X \implies \kappa x A = (\sum i. ki' i x A)$
by(metis s-finite-kernels)
hence sets- ki' [measurable-cong]: $\bigwedge x i. x \in \text{space } X \implies \text{sets } (ki' i x) = \text{sets } Y$
by(auto simp: subprob-kernel-def measure-kernel-def)
define ki **where** $ki \equiv (\lambda i x. \text{density } (ki' i x) (f x))$
have sets- ki : $x \in \text{space } X \implies \text{sets } (ki i x) = \text{sets } Y$ **for** $i x$
using $ki'(1)$ **by**(auto simp: subprob-kernel-def measure-kernel-def ki -def)
have $\text{emeasure-k}:\text{density } (\kappa x) (f x) A = (\sum i. ki i x A)$ **if** $x:x \in \text{space } X$ **and** $A[\text{measurable}]:A \in \text{sets } Y$ **for** $x A$
using kernel-sets[OF x] $ki'(1)$ sets- ki' [OF x] **by**(auto simp: emeasure-density nn-integral-measure-suminf[OF - $ki'(2)$, of x] ki -def)
show ?thesis
proof(rule s-finite-kernel-s-finite-kernel[**where** $ki=ki$, OF - - emeasure-k])
fix i
note nn-integral-measurable-subprob-algebra2[OF - $ki'(1)$][of i , simplified subprob-kernel-def[^], measurable]
define kij **where** $kij \equiv (\lambda j x. \text{if } j = 0 \text{ then density } (ki' i x) (\lambda y. \infty * \text{indicator } \{y \in \text{space } Y. f x y = \infty\} y)$
else if $j = (\text{Suc } 0)$ then density $(ki' i x) (\lambda y. f x y * \text{indicator } \{y \in \text{space } Y. f x y < \infty\} y)$
else null-measure Y)
have $\text{emeasure-kij}: ki i x A = (\sum j. kij j x A)$ (**is** ?lhs = ?rhs) **if** $x:x \in \text{space } X$ **and** [measurable]: $A \in \text{sets } Y$ **for** $x A$
proof -
have ?lhs = $(\int^+ y \in A. f x y \partial ki' i x)$

using *sets-ki*[*OF x, of i*] *x* **by**(*auto simp: ki-def emeasure-density*)
also have ... = ($\int^+ y. (\infty * \text{indicator } \{y \in \text{space } Y. f x y = \infty\} y * \text{indicator } A y + f x y * \text{indicator } \{y \in \text{space } Y. f x y < \infty\} y * \text{indicator } A y)$ $\partial ki' i x$)
by(*auto intro!: nn-integral-cong simp: sets-eq-imp-space-eq*[*OF sets-ki'*[*OF x*]] *indicator-def*) (*simp add: top.not-eq-extremum*)
also have ... = *density* (*ki' i x*) ($\lambda y. \infty * \text{indicator } \{y \in \text{space } Y. f x y = \infty\} y$) *A* + *density* (*ki' i x*) ($\lambda y. f x y * \text{indicator } \{y \in \text{space } Y. f x y < \infty\} y$) *A*
using *sets-ki*[*OF x, of i*] *x* **by**(*auto simp: ki-def emeasure-density nn-integral-add*)
also have ... = ?*rhs*
using *suminf-finite*[*of* {..*Suc* (*Suc 0*)} $\lambda j. kij j x A$] **by**(*simp add: kij-def*)
finally show ?*thesis* .
qed
have *sets-kij*[*measurable-cong*]: $x \in \text{space } X \implies \text{sets } (kij j x) = \text{sets } Y$ **for** *j x*
by(*auto simp: kij-def sets-ki'*)
show *s-finite-kernel* *X Y* (*ki i*)
proof(*rule s-finite-kernel-s-finite-kernel*[**where** *ki=kij, OF - - emeasure-kij*])
fix *j*
consider $j = 0 \mid j = \text{Suc } 0 \mid j \neq 0 \mid j \neq \text{Suc } 0$ **by** *auto*
then show *s-finite-kernel* *X Y* (*kij j*)
proof *cases*
case 1
have *emeasure-ki*: *emeasure* (*kij j x*) *A* = ($\sum j. \text{emeasure } (\text{density } (ki' i x) (\text{indicator } \{y \in \text{space } Y. f x y = \top\})) A$) **if** $x: x \in \text{space } X$ **and** [*measurable*]: $A \in \text{sets } Y$ **for** *x A*
using *sets-ki'*[*OF x*] *x* **by**(*auto simp: 1 kij-def emeasure-density nn-integral-suminf*[*symmetric*] *indicator-def intro!: nn-integral-cong*) (*simp add: nn-integral-count-space-nat*[*symmetric*])
have [*simp*]: *subprob-kernel* *X Y* ($\lambda x. \text{density } (ki' i x) (\text{indicator } \{y \in \text{space } Y. f x y = \top\})$)
proof -
have [*simp*]: $x \in \text{space } X \implies \text{set-}nn\text{-integral } (ki' i x) (\text{space } Y) (\text{indicator } \{y \in \text{space } Y. f x y = \top\}) \leq 1$ **for** *x*
by(*rule order.trans*[*OF nn-integral-mono*[**where** $v = \lambda x. 1$]], *insert* *ki'(1)*[*of i*]) (*auto simp: indicator-def subprob-kernel-def subprob-kernel-axioms-def intro!: subprob-space.emeasure-space-le-1*)
show ?*thesis*
by(*auto simp: subprob-kernel-def measure-kernel-def emeasure-density subprob-kernel-axioms-def sets-ki' sets-eq-imp-space-eq*[*OF sets-ki'*] *Y cong: measurable-cong intro!: subprob-spaceI*)
qed
show ?*thesis*
by (*auto simp: s-finite-kernel-def' sets-kij intro!: exI*[**where** $x = \lambda k x. \text{density } (ki' i x) (\text{indicator } \{y \in \text{space } Y. f x y = \top\})$]) *simp: emeasure-ki*)
next
case *j:2*
have *emeasure-ki*: *emeasure* (*kij j x*) *A* = ($\sum k. \text{density } (ki' i x) (\lambda y. f x y * \text{indicator } \{y \in \text{space } Y. \text{of-nat } k \leq f x y \wedge f x y < 1 + \text{of-nat } k\} y) A$) **if** $x: x \in \text{space } X$ **and** [*measurable*]: $A \in \text{sets } Y$ **for** *x A*
proof -
have [*simp*]: $f x y * \text{indicator } \{y \in \text{space } Y. f x y < \top\} y * \text{indicator } A y$

```

= f x y * (∑ k. indicator {y ∈ space Y. of-nat k ≤ f x y ∧ f x y < 1 + of-nat k}
y) * indicator A y if y: y ∈ space Y for y
  proof(cases f x y < ⊤)
    case f: True
      define l where l ≡ floor (enn2real (f x y))
      have nat l ≤ enn2real (f x y) enn2real (f x y) < 1 + nat l
        by (simp-all add: l-def) linarith
      with y have l: of-nat (nat l) ≤ f x y f x y < 1 + of-nat (nat l)
        using Orderings.order-eq-iff enn2real-positive-iff ennreal-enn2real-if
ennreal-of-nat-eq-real-of-nat linorder-not-le of-nat-0-le-iff f by fastforce+
      then have (∑ j. indicator {y ∈ space Y. of-nat j ≤ f x y ∧ f x y < 1
+ of-nat j} y :: ennreal) = (∑ j. if j = nat l then 1 else 0)
        by(auto intro!: suminf-cong simp: indicator-def y) (metis Suc-leI
linorder-neqE-nat linorder-not-less of-nat-Suc of-nat-le-iff order-trans)
      also have ... = 1
        using suminf-finite[where N={nat l} and f=λj. if j = nat l then 1
else (0 :: ennreal)] by simp
      finally show ?thesis
        by(auto, insert f) (auto simp: indicator-def)
      qed(use top.not-eq-extremum in fastforce)
      show ?thesis
        using sets-ki[OF x] sets-ki'[OF x] x by(auto simp: kij-def j emea-
sure-density nn-integral-suminf[symmetric] sets-eq-imp-space-eq[OF sets-ki'[OF x]]
intro!: nn-integral-cong)
      qed
      show ?thesis
      proof(rule s-finite-kernel-finite-bounded-sum[OF sets-kij - emeasure-ki])
        fix k
          show measure-kernel X Y (λx. density (ki' i x) (λy. f x y * indicator {y
∈ space Y. of-nat k ≤ f x y ∧ f x y < 1 + of-nat k} y))
            using sets-ki'[of - i] by(auto simp: measure-kernel-def emeasure-density
Y cong: measurable-cong)
          next
            fix k x
              assume x: x ∈ space X
              have emeasure (density (ki' i x) (λy. f x y * indicator {y ∈ space Y. of-nat
k ≤ f x y ∧ f x y < 1 + of-nat k} y)) (space Y) ≤ 1 + of-nat k
                by(auto simp: emeasure-density x, rule order.trans[OF nn-integral-mono[where
v=λx. 1 + of-nat k]]) (insert subprob-kernel.subprob-space[OF ki'(1)[of i] x], auto
simp: indicator-def subprob-kernel-def subprob-kernel-axioms-def sets-eq-imp-space-eq[OF
sets-ki'[OF x]] intro!: mult-mono[where d=1 :: ennreal, OF order.refl, simplified])
              also have ... < ∞
                by (simp add: of-nat-less-top)
              finally show emeasure (density (ki' i x) (λy. f x y * indicator {y ∈ space
Y. of-nat k ≤ f x y ∧ f x y < 1 + of-nat k} y)) (space Y) < ∞ .
                qed auto
            next
              case 3
                then show ?thesis

```

by(*auto simp: kij-def s-finite-kernel-cong-sets*[of $X X Y, OF - sets-null-measure$ [*symmetric*]])
Y intro!: *s-finite-measure.s-finite-kernel-const finite-measure.s-finite-measure-finite-measure*
finite-measureI)
qed
qed(*auto simp: sets-ki*)
qed(*auto simp: kernel-sets*)
qed

lemma(*in s-finite-kernel*) *nn-integral-measurable-f*:
assumes [*measurable*]: $(\lambda(x,y). f x y) \in \text{borel-measurable } (X \otimes_M Y)$
shows $(\lambda x. \int^+ y. f x y \partial(\kappa x)) \in \text{borel-measurable } X$
proof –
obtain κi **where** $\kappa i: \bigwedge i. \text{subprob-kernel } X Y (\kappa i i) \wedge x A. x \in \text{space } X \implies \kappa x$
 $A = (\sum i. \kappa i i x A)$
by(*metis s-finite-kernels*)
show *?thesis*
proof(*rule measurable-cong*[*THEN iffD2*])
fix x
assume $x \in \text{space } X$
with κi **show** $(\int^+ y. f x y \partial \kappa x) = (\sum i. \int^+ y. f x y \partial \kappa i i x)$
by(*auto intro!*: *nn-integral-measure-suminf*[*symmetric*] *simp: subprob-kernel-def*
kernel-sets measure-kernel-def)
next
show $(\lambda x. \sum i. \int^+ y. f x y \partial \kappa i i x) \in \text{borel-measurable } X$
using $\kappa i(1)$ *nn-integral-measurable-subprob-algebra2*[*OF assms*] **by**(*simp add:*
subprob-kernel-def')
qed
qed

lemma(*in s-finite-kernel*) *nn-integral-measurable-f'*:
assumes $f \in \text{borel-measurable } (X \otimes_M Y)$
shows $(\lambda x. \int^+ y. f(x, y) \partial(\kappa x)) \in \text{borel-measurable } X$
using *nn-integral-measurable-f*[**where** $f = \text{curry } f, \text{simplified}, OF \text{ assms}$] **by** *simp*

lemma(*in s-finite-kernel*) *bind-kernel-s-finite-kernel'*:
assumes *s-finite-kernel* $(X \otimes_M Y) Z$ (*case-prod g*)
shows *s-finite-kernel* $X Z$ $(\lambda x. \kappa x \gg_k g x)$
proof(*cases space X = {}*)
case *True*
then show *?thesis*
by (*simp add: s-finite-kernel-empty-trivial*)
next
case $X:False$
then have $Y:\text{space } Y \neq \{\}$
by(*simp add: Y-not-empty*)
from *s-finite-kernels* **obtain** ki **where** $ki:$
 $\bigwedge i. \text{subprob-kernel } X Y (ki i) \wedge x A. x \in \text{space } X \implies \kappa x A = (\sum i. ki i x A)$
by *metis*
interpret $g:s\text{-finite-kernel } X \otimes_M Y Z$ *case-prod g* **by** *fact*

from $g.s\text{-finite-kernels}[simplified\ space\ pair\ measure]$ **obtain** gi **where** gi :
 $\bigwedge i. subprob\ kernel\ (X \otimes_M Y)\ Z\ (gi\ i) \wedge x\ y\ A. x \in space\ X \implies y \in space\ Y$
 $\implies g\ x\ y\ A = (\sum i. gi\ i\ (x, y)\ A)$
by *auto metis*
define kgi **where** $kgi = (\lambda i\ x. case\ prod\ decode\ i\ of\ (i, j) \Rightarrow (ki\ j\ x \ggg\ curry\ (gi\ i\ x)))$
have $emeasure:emeasure\ (\kappa\ x \ggg_k\ g\ x)\ A = (\sum i. emeasure\ (kgi\ i\ x)\ A)$ (**is** $?lhs = ?rhs$) **if** $x: x \in space\ X$ **and** $A: A \in sets\ Z$ **for** $x\ A$
proof –
interpret $gx: s\text{-finite-kernel}\ Y\ Z\ g\ x$
using $g.comp\ measurable[OF\ measurable\ Pair1\ '[OF\ x]]$ **by** *auto*
have $?lhs = (\int^+ y. g\ x\ y\ A\ \partial\ \kappa\ x)$
using $gx.emeasure\ bind\ kernel[OF\ kernel\ sets[OF\ x]\ A]$
by $(auto\ simp: sets\ eq\ imp\ space\ eq[OF\ kernel\ sets[OF\ x]]\ Y)$
also **have** $... = (\int^+ y. (\sum i. gi\ i\ (x, y)\ A)\ \partial\ \kappa\ x)$
by $(auto\ intro!: nn\ integral\ cong\ simp: sets\ eq\ imp\ space\ eq[OF\ kernel\ sets[OF\ x]]\ gi(2)[OF\ x])$
also **have** $... = (\sum i. \int^+ y. gi\ i\ (x, y)\ A\ \partial\ \kappa\ x)$
using $gi(1)\ x\ A$ **by** $(auto\ intro!: nn\ integral\ suminf\ simp: subprob\ kernel\ def')$
also **have** $... = (\sum i. (\sum j. \int^+ y. gi\ i\ (x, y)\ A\ \partial\ ki\ j\ x))$
by $(rule\ suminf\ cong, rule\ nn\ integral\ measure\ suminf[symmetric], insert\ kernel\ sets[OF\ x]\ ki\ gi(1)\ x\ A)$
 $(auto\ simp: subprob\ kernel\ def\ measure\ kernel\ def\ measurable\ cong\ sets[OF\ sets\ pair\ measure\ cong[OF\ refl\ kernel\ sets[OF\ x]]]\ intro!: measurable\ Pair2[OF\ x])$
also **have** $... = (\sum i. (\sum j. emeasure\ (ki\ j\ x \ggg\ (curry\ (gi\ i\ x))\ A))$
using $sets\ eq\ imp\ space\ eq[of\ ki\ -\ x\ Y]\ ki(1)\ x\ gi(1)\ measurable\ cong\ sets[of\ -\ subprob\ algebra\ Z\ subprob\ algebra\ Z, OF\ sets\ pair\ measure\ cong[of\ X\ X\ Y\ ki\ -\ x]]$
by $(auto\ intro!: suminf\ cong\ emeasure\ bind[OF\ -\ A, symmetric]\ measurable\ Pair2[OF\ x]\ simp: curry\ def\ subprob\ kernel\ def[of\ X]\ subprob\ kernel\ def[of\ X\ \otimes_M\ Y]\ measure\ kernel\ def\ Y)$
also **have** $... = ?rhs$
unfolding $kgi\ def$ **by** $(rule\ suminf\ ennreal\ 2dimen[symmetric])$ $(simp\ add: curry\ def)$
finally **show** $?thesis$.
qed
have $sets: sets\ (\kappa\ x \ggg_k\ g\ x) = sets\ Z$ **if** $x: x \in space\ X$ **for** x
proof –
interpret $gx: s\text{-finite-kernel}\ Y\ Z\ g\ x$
using $g.comp\ measurable[OF\ measurable\ Pair1\ '[OF\ x]]$ **by** *auto*
show $?thesis$
by $(simp\ add: gx.sets\ bind\ kernel[OF\ kernel\ sets[OF\ x]]\ Y)$
qed
have $sk: subprob\ kernel\ X\ Z\ (kgi\ i)$ **for** i
using $ki(1)[of\ snd\ (prod\ decode\ i)]\ gi(1)[of\ fst\ (prod\ decode\ i)]$
by $(auto\ simp: subprob\ kernel\ def'\ kgi\ def\ split\ beta'\ curry\ def)$
show $?thesis$
using sk **by** $(auto\ simp: s\text{-finite-kernel}\ def'\ emeasure\ sets\ subprob\ kernel\ def')$

intro!: $exI[\mathbf{where} \ x=kgi] \text{ measurable-cong}[\mathbf{where} \ g=\lambda x. \sum i. \text{emeasure} \ (kgi \ i \ x) - \mathbf{and} \ f=\lambda x. \text{emeasure} \ (\kappa \ x \ggg_k \ g \ x) \ -, \mathbf{THEN} \ \text{iff}D2])$
qed

corollary(*in* *s-finite-kernel*) *bind-kernel-s-finite-kernel*:
assumes *s-finite-kernel* $Y \ Z \ k'$
shows *s-finite-kernel* $X \ Z \ (\lambda x. \kappa \ x \ggg_k \ k')$
by(*auto intro!*: *bind-kernel-s-finite-kernel'* *s-finite-kernel.comp-measurable*[*OF assms measurable-snd*] *simp: split-beta*)

lemma(*in* *s-finite-kernel*) *nn-integral-bind-kernel*:
assumes $f \in \text{borel-measurable} \ Y \ \text{sets} \ \mu = \text{sets} \ X$
shows $(\int^+ y. f \ y \ \partial(\mu \ggg_k \ \kappa)) = (\int^+ x. (\int^+ y. f \ y \ \partial(\kappa \ x)) \ \partial\mu)$
proof(*cases space* $X = \{\}$)
case *True*
then show *?thesis*
by(*simp add: sets-eq-imp-space-eq*[*OF assms(2)*] *bind-kernel-def nn-integral-empty*)
next
case $X:\text{False}$
then have $\mu:\text{space} \ \mu \neq \{\}$ **by**(*simp add: sets-eq-imp-space-eq*[*OF assms(2)*])
note $1[\text{measurable-cong}] = \text{assms}(2) \ \text{sets-bind-kernel}$ [*OF X assms(2)*]
from *assms(1)* **show** *?thesis*
proof *induction*
case *ih:(cong f g)*
have $(\int^+ y. f \ y \ \partial(\mu \ggg_k \ \kappa)) = (\int^+ y. g \ y \ \partial(\mu \ggg_k \ \kappa)) \ (\int^+ x. \text{integral}^N \ (\kappa \ x) \ f \ \partial\mu) = (\int^+ x. \text{integral}^N \ (\kappa \ x) \ g \ \partial\mu)$
by(*auto intro!*: *nn-integral-cong simp: sets-eq-imp-space-eq*[*OF 1(2)*] *sets-eq-imp-space-eq*[*OF assms(2)*] *sets-eq-imp-space-eq*[*OF kernel-sets*] *ih(3)*)
then show *?case*
by(*simp add: ih*)
next
case (*set A*)
then show *?case*
by(*auto simp: emeasure-bind-kernel*[*OF 1(1) - X*] *sets-eq-imp-space-eq*[*OF 1(1)*] *intro!*: *nn-integral-cong*)
next
case *ih:(mult u c)*
then have $(\int^+ x. \int^+ y. c * u \ y \ \partial\kappa \ x \ \partial\mu) = (\int^+ x. c * \int^+ y. u \ y \ \partial\kappa \ x \ \partial\mu)$
by(*auto intro!*: *nn-integral-cong nn-integral-cmult simp: sets-eq-imp-space-eq*[*OF 1(1)*])
with *ih nn-integral-measurable-f*[*of* $\lambda-. u \ y$] **show** *?case*
by(*auto simp: nn-integral-cmult intro!*: *nn-integral-cong*)
next
case *ih:(add u v)*
then have $(\int^+ x. \int^+ y. v \ y + u \ y \ \partial\kappa \ x \ \partial\mu) = (\int^+ x. (\int^+ y. v \ y \ \partial\kappa \ x) + (\int^+ y. u \ y \ \partial\kappa \ x)) \ \partial\mu$
by(*auto intro!*: *nn-integral-cong simp: nn-integral-add sets-eq-imp-space-eq*[*OF 1(1)*])
with *ih nn-integral-measurable-f*[*of* $\lambda-. u \ y$] *nn-integral-measurable-f*[*of* $\lambda-. v \ y$]

```

v y] show ?case
  by(simp add: nn-integral-add)
next
case ih[measurable]:(seq U)
show ?case (is ?lhs = ?rhs)
proof -
  have ?lhs = (( $\bigsqcup$  i. integralN ( $\mu \gg_k \kappa$ ) (U i)))
  by(rule nn-integral-monotone-convergence-SUP[of U,simplified SUP-apply[of
U UNIV,symmetric]]) (use ih in auto)
  also have ... = ( $\bigsqcup$  i.  $\int^+ x. (\int^+ y. U i y \partial\kappa x) \partial\mu$ )
  by(simp add: ih)
  also have ... = ( $\int^+ x. (\bigsqcup i. (\int^+ y. U i y \partial\kappa x)) \partial\mu$ )
  proof(rule nn-integral-monotone-convergence-SUP[symmetric])
    show incseq ( $\lambda i x. \int^+ y. U i y \partial\kappa x$ )
    by standard+ (auto intro!: le-funI nn-integral-mono simp:le-funD[OF
incseqD[OF ih(3)]])
  qed(use nn-integral-measurable-f[of  $\lambda y. U y$ ] in simp)
  also have ... = ?rhs
  by(rule nn-integral-cong, rule nn-integral-monotone-convergence-SUP[of
U,simplified SUP-apply[of U UNIV,symmetric],OF ih(3),symmetric]) (auto simp:
sets-eq-imp-space-eq[OF 1(1)])
  finally show ?thesis .
qed
qed
qed

```

```

lemma(in s-finite-kernel) bind-kernel-assoc:
  assumes s-finite-kernel Y Z k' sets  $\mu =$  sets X
  shows  $\mu \gg_k (\lambda x. \kappa x \gg_k k') = \mu \gg_k \kappa \gg_k k'$ 
proof(cases space X = {})
case X:False
then have  $\mu$ : space  $\mu \neq \{\}$  and Y:space Y  $\neq \{\}$ 
  by(simp-all add: Y-not-empty sets-eq-imp-space-eq[OF assms(2)])
interpret k':s-finite-kernel Y Z k' by fact
interpret k'': s-finite-kernel X Z  $\lambda x. \kappa x \gg_k k'$ 
  by(rule bind-kernel-s-finite-kernel[OF assms(1)])
show ?thesis
proof(rule measure-eqI)
  fix A
  assume A  $\in$  sets ( $\mu \gg_k (\lambda x. \kappa x \gg_k k')$ )
  then have A[measurable]: A  $\in$  sets Z
  by(simp add: k''.sets-bind-kernel[OF X assms(2)])
  show emeasure ( $\mu \gg_k (\lambda x. \kappa x \gg_k k')$ ) A = emeasure ( $\mu \gg_k \kappa \gg_k k'$ ) A
(is ?lhs = ?rhs)
proof -
  have ?lhs = ( $\int^+ x. emeasure (\kappa x \gg_k k') A \partial\mu$ )
  by(rule k''.emeasure-bind-kernel[OF assms(2) A X])
  also have ... = ( $\int^+ x. \int^+ y. k' y A \partial\kappa x \partial\mu$ )
  using k'.emeasure-bind-kernel[OF kernel-sets A]

```

by(*auto intro!*: *nn-integral-cong simp: sets-eq-imp-space-eq*[*OF assms*(2)]
sets-eq-imp-space-eq[*OF kernel-sets*] *Y*)
also have ... = $(\int^+ y. k' y A \partial(\mu \gg_k \kappa))$
by(*simp add: nn-integral-bind-kernel*[*OF k'.emeasure-measurable*[*OF A*]
assms(2)])
also have ... = *?rhs*
by(*simp add: k'.emeasure-bind-kernel*[*OF sets-bind-kernel*[*OF X assms*(2)]
A] *sets-eq-imp-space-eq*[*OF sets-bind-kernel*[*OF X assms*(2)]] *Y*)
finally show *?thesis* .
qed
qed(*auto simp: k'.sets-bind-kernel*[*OF Y sets-bind-kernel*[*OF X assms*(2)]] *k''.sets-bind-kernel*[*OF X assms*(2)])
qed(*simp add: bind-kernel-def sets-eq-imp-space-eq*[*OF assms*(2)])

lemma *s-finite-kernel-pair-measure*:

assumes *s-finite-kernel X Y k* **and** *s-finite-kernel X Z k'*
shows *s-finite-kernel X (Y \otimes_M Z) ($\lambda x. k x \otimes_M k' x$)*
proof –
interpret *k: s-finite-kernel X Y k* **by** *fact*
interpret *k': s-finite-kernel X Z k'* **by** *fact*
from *k.s-finite-kernels k'.s-finite-kernels* **obtain** *ki ki'*
where *ki: $\bigwedge i. \text{subprob-kernel } X Y (ki\ i) \wedge x A. x \in \text{space } X \implies k\ x\ A = (\sum i. ki\ i\ x\ A)$*
and *ki': $\bigwedge i. \text{subprob-kernel } X Z (ki'\ i) \wedge x A. x \in \text{space } X \implies k'\ x\ A = (\sum i. ki'\ i\ x\ A)$*
by *metis*
then have *1[measurable-cong]: $\bigwedge x\ i. x \in \text{space } X \implies \text{sets } (ki\ i\ x) = \text{sets } Y \wedge x\ i. x \in \text{space } X \implies \text{sets } (ki'\ i\ x) = \text{sets } Z$*
by(*auto simp: subprob-kernel-def measure-kernel-def*)
define *kki* **where** *kki $\equiv (\lambda i\ x. (\lambda(j, i). ki\ i\ x \otimes_M ki'\ j\ x) (\text{prod-decode } i))$*
have *kki1: $\bigwedge i. \text{subprob-kernel } X (Y \otimes_M Z) (kki\ i)$*
using *ki(1) ki'(1)* **by**(*auto simp: subprob-kernel-def' kki-def split-beta intro!*:
measurable-pair-measure)
have *kki2: $(k\ x \otimes_M k'\ x)\ A = (\sum i. (kki\ i\ x)\ A)$ (**is** *?lhs = ?rhs*) **if** *x: x \in space X* **and** *A[measurable]: A \in sets (Y \otimes_M Z)* **for** *x A*
proof –
have *?lhs = $(\int^+ y. \int^+ z. \text{indicator } A\ (y, z) \partial k'\ x\ \partial k\ x)$*
using *x* **by**(*simp add: s-finite-measure.emeasure-pair-measure'*[*OF k'.image-s-finite-measure*])
also have ... = $(\int^+ y. (\sum j. \int^+ z. \text{indicator } A\ (y, z) \partial ki'\ j\ x) \partial k\ x)$
using *ki' x* **by**(*auto intro!*: *nn-integral-cong nn-integral-measure-suminf[symmetric]*
simp: sets-eq-imp-space-eq[*OF k.kernel-sets*[*OF x*]] *subprob-kernel-def measure-kernel-def*
k'.kernel-sets)
also have ... = $(\sum j. \int^+ y. \int^+ z. \text{indicator } A\ (y, z) \partial ki'\ j\ x\ \partial k\ x)$
using *x* **by**(*auto intro!*: *nn-integral-suminf s-finite-measure.borel-measurable-nn-integral-fst'*
s-finite-kernel.image-s-finite-measure[*OF subprob-kernel.s-finite-kernel-subprob-kernel*[*OF*
ki'(1)]])
also have ... = $(\sum j. (\sum i. (\int^+ y. \int^+ z. \text{indicator } A\ (y, z) \partial ki'\ j\ x\ \partial ki\ i\ x)))$
using *x ki* **by**(*auto intro!*: *suminf-cong nn-integral-measure-suminf[symmetric]*
s-finite-measure.borel-measurable-nn-integral-fst' simp: k.kernel-sets[*OF x*] *subprob-kernel-def*)*

```

measure-kernel-def s-finite-kernel.image-s-finite-measure[OF subprob-kernel.s-finite-kernel-subprob-kernel[OF
ki'(1)]]
  also have ... = ( $\sum j. (\sum i. (ki\ i\ x \otimes_M ki'\ j\ x) A)$ )
  using x by(auto simp: s-finite-measure.emeasure-pair-measure'[OF s-finite-kernel.image-s-finite-measure[C
subprob-kernel.s-finite-kernel-subprob-kernel[OF ki'(1)]]])
  also have ... = ?rhs
  unfolding kki-def by(rule suminf-ennreal-2dimen[symmetric]) auto
  finally show ?thesis .
qed
show ?thesis
proof
  fix B
  assume [measurable]:B  $\in$  sets (Y  $\otimes_M$  Z)
  show ( $\lambda x. \text{emeasure } (k\ x \otimes_M k'\ x) B$ )  $\in$  borel-measurable X
  by(rule measurable-cong[where g= $\lambda x. \sum i. (kki\ i\ x) B$ , THEN iffD2], insert
kki1) (auto simp: subprob-kernel-def' kki2)
  qed(auto intro!: exI[where x=kki] simp: subprob-kernel.finite-kernel kki1 kki2
k.kernel-sets k'.kernel-sets space-pair-measure k.Y-not-empty k'.Y-not-empty)
qed

lemma pair-measure-eq-bind-s-finite:
  assumes s-finite-measure  $\mu$  s-finite-measure  $\nu$ 
  shows  $\mu \otimes_M \nu = \mu \ggg_k (\lambda x. \nu \ggg_k (\lambda y. \text{return } (\mu \otimes_M \nu) (x,y)))$ 
proof -
  consider space  $\mu = \{\}$  | space  $\nu = \{\}$  | space  $\mu \neq \{\}$  space  $\nu \neq \{\}$ 
  by auto
  then show ?thesis
  proof cases
    case 1
    then show ?thesis
    by(auto simp: bind-kernel-def space-pair-measure intro!: space-empty)
  next
    case 2
    then have  $\mu \ggg_k (\lambda x. \nu \ggg_k (\lambda y. \text{return } (\mu \otimes_M \nu) (x, y))) = \text{count-space}$ 
    {}
    by(auto simp: bind-kernel-def space-empty)
  with 2 show ?thesis
  by(auto simp: space-pair-measure intro!: space-empty)
  next
    case 3
    show ?thesis
  proof(intro measure-eqI sets-bind-kernel[OF - 3(1),symmetric] sets-bind-kernel[OF
- 3(2)])
    fix A
    assume A[measurable]: A  $\in$  sets ( $\mu \otimes_M \nu$ )
    show  $\text{emeasure } (\mu \otimes_M \nu) A = \text{emeasure } (\mu \ggg_k (\lambda x. \nu \ggg_k (\lambda y. \text{return } (\mu
\otimes_M \nu) (x, y)))) A$  (is ?lhs = ?rhs)
    proof -
      have ?lhs = ( $\int^+ x. \int^+ y. \text{return } (\mu \otimes_M \nu) (x, y) A\ \partial\nu\ \partial\mu$ )

```



```

    by(simp add: s-finite-measure.emeasure-pair-measure'[OF assms(2)])
  also have ... = (∫+ x. (ν ≫k (λy. return (μ ⊗M ν) (x,y))) A ∂μ)
    by(auto intro!: nn-integral-cong measure-kernel.emeasure-bind-kernel[OF -
- A 3(2),symmetric] prob-kernel.axioms(1) simp: prob-kernel-def' simp del: emea-
sure-return)
  also have ... = ?rhs
    by(auto intro!: measure-kernel.emeasure-bind-kernel[OF - - A 3(1),symmetric]
s-finite-kernel.axioms(1) s-finite-kernel.bind-kernel-s-finite-kernel'[where Y=ν] s-finite-measure.s-finite-kernel
assms(2) 3(2)] prob-kernel.s-finite-kernel-prob-kernel[of μ ⊗M ν] simp: prob-kernel-def')
  finally show ?thesis .
qed
qed simp
qed
qed

```

lemma *bind-kernel-rotate-return:*

```

  assumes s-finite-measure μ s-finite-measure ν
  shows μ ≫k (λx. ν ≫k (λy. return (μ ⊗M ν) (x,y))) = ν ≫k (λy. μ ≫k
(λx. return (μ ⊗M ν) (x,y)))
proof -
  consider space μ = {} | space ν = {} | space μ ≠ {} space ν ≠ {}
  by auto
  then show ?thesis
proof cases
  case 1
  then have ν ≫k (λy. μ ≫k (λx. return (μ ⊗M ν) (x,y))) = count-space {}
    by(auto simp: bind-kernel-def space-empty)
  then show ?thesis
    by(auto simp: bind-kernel-def space-pair-measure 1 intro!: space-empty)
next
  case 2
  then have μ ≫k (λx. ν ≫k (λy. return (μ ⊗M ν) (x, y))) = count-space
{}
    by(auto simp: bind-kernel-def space-empty)
  with 2 show ?thesis
    by(auto simp: space-pair-measure bind-kernel-def intro!: space-empty)
next
  case 3
  show ?thesis
    unfolding pair-measure-eq-bind-s-finite[OF assms,symmetric]
proof(intro measure-eqI)
  fix A
  assume A[measurable]:A ∈ sets (μ ⊗M ν)
  show emeasure (μ ⊗M ν) A = emeasure (ν ≫k (λy. μ ≫k (λx. return (μ
⊗M ν) (x, y)))) A (is ?lhs = ?rhs)
proof -
  have ?lhs = (∫+ x. ∫+ y. indicator A (x, y) ∂ν ∂μ)
    by(rule s-finite-measure.emeasure-pair-measure'[OF assms(2) A])
  also have ... = (∫+ y. ∫+ x. return (μ ⊗M ν) (x, y) A ∂μ ∂ν)

```

```

by(simp add: nn-integral-snd'[OF assms] s-finite-measure.nn-integral-fst'[OF
assms(2)])
also have ... = (∫+ y. (μ ≫k (λx. return (μ ⊗M ν) (x, y)))) A ∂ν
by(auto intro!: nn-integral-cong measure-kernel.emeasure-bind-kernel[OF -
- A ∩(1),symmetric] prob-kernel.axioms(1) simp add: prob-kernel-def' simp del:
emeasure-return)
also have ... = ?rhs
by(auto intro!: measure-kernel.emeasure-bind-kernel[OF - - A ∩(2),symmetric]
s-finite-kernel.axioms(1) s-finite-kernel.bind-kernel-s-finite-kernel'[where Y=μ] s-finite-measure.s-finite-kernel
assms(1) ∩(1)] prob-kernel.s-finite-kernel-prob-kernel[of ν ⊗M μ] simp: prob-kernel-def')
finally show ?thesis .
qed
qed(auto intro!: sets-bind-kernel[OF - ∩(2),symmetric] sets-bind-kernel[OF -
∩(1)])
qed
qed

```

lemma bind-kernel-rotate':

assumes s-finite-measure μ s-finite-measure ν s-finite-kernel (μ ⊗_M ν) Z (case-prod f)

shows μ ≫_k (λx. ν ≫_k (λy. f x y)) = ν ≫_k (λy. μ ≫_k (λx. f x y)) (**is** ?lhs = ?rhs)

proof –

interpret sk: s-finite-kernel μ ⊗_M ν Z case-prod f **by** fact

consider space μ = {} | space ν = {} | space μ ≠ {} space ν ≠ {}

by auto

then show ?thesis

proof cases

case 1

then have ?rhs = count-space {}

by(auto simp: bind-kernel-def space-empty)

then show ?thesis

by(auto simp: bind-kernel-def space-pair-measure 1 intro!: space-empty)

next

case 2

then show ?thesis

by(auto simp: space-pair-measure bind-kernel-def intro!: space-empty)

next

case 3

show ?thesis

proof –

have ?lhs = μ ≫_k (λx. ν ≫_k (λy. return (μ ⊗_M ν) (x, y))) ≫_k case-prod f)

by(auto intro!: bind-kernel-cong-All simp: s-finite-kernel.bind-kernel-assoc[OF prob-kernel.s-finite-kernel-prob-kernel assms(3) refl,of ν λy. return (μ ⊗_M ν) (-, y),simplified prob-kernel-def',symmetric] sk.bind-kernel-return space-pair-measure)

also have ... = μ ≫_k (λx. ν ≫_k (λy. return (μ ⊗_M ν) (x,y))) ≫_k (case-prod f)

by(auto simp: s-finite-kernel.bind-kernel-assoc[OF s-finite-kernel.bind-kernel-s-finite-kernel'[OF

s-finite-measure.s-finite-kernel-const[*OF* *assms*(2) $\mathcal{B}(2)$, of μ] *prob-kernel.s-finite-kernel-prob-kernel*, of $\mu \otimes_M \nu \lambda x y. \text{return } (\mu \otimes_M \nu) (x, y)$, *simplified*] *assms*(3) *refl*, *simplified prob-kernel-def'*, *symmetric*])
also have ... = $\nu \gg_k (\lambda y. \mu \gg_k (\lambda x. \text{return } (\mu \otimes_M \nu) (x, y))) \gg_k$
(case-prod f)
by (*simp add: bind-kernel-rotate-return assms*)
also have ... = $\nu \gg_k (\lambda y. \mu \gg_k (\lambda x. \text{return } (\mu \otimes_M \nu) (x, y))) \gg_k$
case-prod f)
by (*auto intro!: s-finite-kernel.bind-kernel-assoc*[*OF* - *assms*(3), *symmetric*]
s-finite-kernel.bind-kernel-s-finite-kernel'[*OF* *s-finite-measure.s-finite-kernel-const*[*OF*
assms(1) $\mathcal{B}(1)$]]) *prob-kernel.s-finite-kernel-prob-kernel*[of $\nu \otimes_M \mu$] *simp: prob-kernel-def'*)
also have ... = ?*rhs*
by (*auto intro!: bind-kernel-cong-All simp: s-finite-kernel.bind-kernel-assoc*[*OF*
prob-kernel.s-finite-kernel-prob-kernel *assms*(3) *refl*, of $\mu \lambda x. \text{return } (\mu \otimes_M \nu) (x,$
 $-)$, *simplified prob-kernel-def'*, *symmetric*] *sk.bind-kernel-return space-pair-measure*)
finally show ?*thesis* .
qed
qed
qed

lemma *bind-kernel-rotate*:

assumes *sets* $\mu = \text{sets } X$ **and** *sets* $\nu = \text{sets } Y$
and *s-finite-measure* μ *s-finite-measure* ν *s-finite-kernel* $(X \otimes_M Y) Z (\lambda(x, y). f x y)$
shows $\mu \gg_k (\lambda x. \nu \gg_k (\lambda y. f x y)) = \nu \gg_k (\lambda y. \mu \gg_k (\lambda x. f x y))$
by (*auto intro!: bind-kernel-rotate' assms simp: s-finite-kernel-cong-sets*[*OF* *sets-pair-measure-cong*[*OF*
assms(1, 2)]]])

lemma(*in s-finite-kernel*) *emeasure-measurable'*:

assumes *A*[*measurable*]: $(\text{SIGMA } x:\text{space } X. A x) \in \text{sets } (X \otimes_M Y)$
shows $(\lambda x. \text{emeasure } (\kappa x) (A x)) \in \text{borel-measurable } X$
proof –
have **: $A x \in \text{sets } Y$ **if** $x \in \text{space } X$ **for** x
proof –
have *Pair* $x - \text{'Sigma (space } X) A = A x$
using *that* **by** *auto*
with *sets-Pair1*[*OF* *A*, of x] **show** $A x \in \text{sets } Y$
by *auto*
qed

have *: $\bigwedge x. \text{fst } x \in \text{space } X \implies \text{snd } x \in A \iff x \in (\text{SIGMA } x:\text{space } X. A x)$

by (*auto simp: fun-eq-iff*)
have $(\lambda(x, y). \text{indicator } (A x) y::\text{ennreal}) \in \text{borel-measurable } (X \otimes_M Y)$
by (*measurable, subst measurable-cong*[*OF* *]) (*auto simp: space-pair-measure*)
then have $(\lambda x. \text{integral}^N (\kappa x) (\text{indicator } (A x))) \in \text{borel-measurable } X$
by (*rule nn-integral-measurable-f*)
moreover have $\text{integral}^N (\kappa x) (\text{indicator } (A x)) = \text{emeasure } (\kappa x) (A x)$ **if** $x \in \text{space } X$ **for** x
using **[*OF that*] *kernel-sets*[*OF that*] **by** (*auto intro!: nn-integral-indicator*)

ultimately show $(\lambda x. \text{emeasure } (\kappa x) (A x)) \in \text{borel-measurable } X$
by(*auto cong: measurable-cong*)
qed

lemma(*in s-finite-kernel*) *measure-measurable'*:
assumes $(\text{SIGMA } x:\text{space } X. A x) \in \text{sets } (X \otimes_M Y)$
shows $(\lambda x. \text{measure } (\kappa x) (A x)) \in \text{borel-measurable } X$
using *emeasure-measurable'[OF assms]* **by**(*simp add: measure-def*)

lemma(*in s-finite-kernel*) *AE-pred*:
assumes $P[\text{measurable}]:\text{Measurable.pred } (X \otimes_M Y) (\text{case-prod } P)$
shows $\text{Measurable.pred } X (\lambda x. \text{AE } y \text{ in } \kappa x. P x y)$

proof –
have $[\text{measurable}]:\text{Measurable.pred } X (\lambda x. \text{emeasure } (\kappa x) \{y \in \text{space } Y. \neg P x y\} = 0)$
proof(*rule pred-eq-const1[where N=borel],rule emeasure-measurable'*)
have $(\text{SIGMA } x:\text{space } X. \{y \in \text{space } Y. \neg P x y\}) = \{xy \in \text{space } (X \otimes_M Y). \neg P (\text{fst } xy) (\text{snd } xy)\}$
by (*auto simp: space-pair-measure*)
also have $\dots \in \text{sets } (X \otimes_M Y)$
by *simp*
finally show $(\text{SIGMA } x:\text{space } X. \{y \in \text{space } Y. \neg P x y\}) \in \text{sets } (X \otimes_M Y)$

qed *simp*
have $\{x \in \text{space } X. \text{almost-everywhere } (\kappa x) (P x)\} = \{x \in \text{space } X. \text{emeasure } (\kappa x) \{y \in \text{space } Y. \neg P x y\} = 0\}$
proof *safe*
fix x
assume $x:x \in \text{space } X$
show $(\text{AE } y \text{ in } \kappa x. P x y) \implies \text{emeasure } (\kappa x) \{y \in \text{space } Y. \neg P x y\} = 0$
using *emeasure-eq-0-AE[of $\lambda y. \neg P x y \kappa x$]*
by(*simp add: sets-eq-imp-space-eq[OF kernel-sets[OF x]]*)
show $\text{emeasure } (\kappa x) \{y \in \text{space } Y. \neg P x y\} = 0 \implies \text{almost-everywhere } (\kappa x) (P x)$
using x **by**(*auto intro!: AE-I[where N={y ∈ space Y. ¬ P x y}] simp: sets-eq-imp-space-eq[OF kernel-sets[OF x]] kernel-sets[OF x]*)
qed
also have $\dots \in \text{sets } X$
by(*simp add: pred-def*)
finally show *?thesis*
by(*simp add: pred-def*)
qed

lemma(*in subprob-kernel*) *integrable-probability-kernel-pred*:
fixes $f :: - \Rightarrow - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
assumes $[\text{measurable}]:(\lambda(x,y). f x y) \in \text{borel-measurable } (X \otimes_M Y)$
shows $\text{Measurable.pred } X (\lambda x. \text{integrable } (\kappa x) (f x))$
proof(*rule measurable-cong[THEN iffD2]*)
show $x \in \text{space } X \implies \text{integrable } (\kappa x) (f x) \longleftrightarrow (\int^+ y. \text{norm } (f x y) \partial(\kappa x)) <$

```

∞ for x
  by(auto simp: integrable-iff-bounded)
next
  have (λ(x,y). ennreal (norm (f x y))) ∈ borel-measurable (X ⊗M Y)
    by measurable
  from nn-integral-measurable-f[OF this]
  show Measurable.pred X (λx. (∫+ y. ennreal (norm (f x y)) ∂κ x) < ∞)
    by simp
qed

corollary integrable-measurable-subprob':
  fixes f :: - ⇒ - ⇒ -::{banach, second-countable-topology}
  assumes [measurable]:(λ(x,y). f x y) ∈ borel-measurable (X ⊗M Y) k ∈ X →M
subprob-algebra Y
  shows Measurable.pred X (λx. integrable (k x) (f x))
  by(auto intro!: subprob-kernel.integrable-probability-kernel-pred[where Y=Y] simp:
subprob-kernel-def')

lemma(in subprob-kernel) integrable-probability-kernel-pred':
  fixes f :: - ⇒ -::{banach, second-countable-topology}
  assumes f ∈ borel-measurable (X ⊗M Y)
  shows Measurable.pred X (λx. integrable (κ x) (curry f x))
  using integrable-probability-kernel-pred[of curry f] assms by auto

lemma(in subprob-kernel) lebesgue-integral-measurable-f-subprob:
  fixes f :: - ⇒ -::{banach, second-countable-topology}
  assumes [measurable]:f ∈ borel-measurable (X ⊗M Y)
  shows (λx. ∫ y. f (x,y) ∂(κ x)) ∈ borel-measurable X
proof -
  from borel-measurable-implies-sequence-metric[OF assms, of 0]
  obtain s where s: ∧i. simple-function (X ⊗M Y) (s i)
    and ∀x∈space (X ⊗M Y). (λi. s i x) → f x
    and ∀i. ∀x∈space (X ⊗M Y). dist (s i x) 0 ≤ 2 * dist (f x) 0
  by auto
  then have *:
    ∧x y. x ∈ space X ⇒ y ∈ space Y ⇒ (λi. s i (x, y)) → f (x,y)
    ∧i x y. x ∈ space X ⇒ y ∈ space Y ⇒ norm (s i (x, y)) ≤ 2 * norm (f (x,
y))
  by (auto simp: space-pair-measure)

  have [measurable]: ∧i. s i ∈ borel-measurable (X ⊗M Y)
    by (rule borel-measurable-simple-function) fact

  have s':∧i. s i ∈ X ⊗M Y →M count-space UNIV
    by (rule measurable-simple-function) fact

  define f' where [abs-def]: f' i x =
    (if integrable (κ x) (curry f x) then Bochner-Integration.simple-bochner-integral
(κ x) (λy. s i (x, y)) else 0) for i x

```

```

have eq: Bochner-Integration.simple-bochner-integral ( $\kappa$  x) ( $\lambda y. s\ i\ (x, y)$ ) =
  ( $\sum z \in s\ i\ \text{'}$  (space X  $\times$  space Y). measure ( $\kappa$  x) { $y \in \text{space } (\kappa\ x). s\ i\ (x, y)$ 
= z}  $\ast_R$  z) if x  $\in$  space X for x i
proof -
  have [measurable-cong]: sets ( $\kappa$  x) = sets Y and [simp]: space ( $\kappa$  x) = space Y
  using that by (simp-all add: kernel-sets kernel-space)
  with that show ?thesis
  using s[THEN simple-functionD(1)]
  unfolding simple-bochner-integral-def
  by (intro sum.mono-neutral-cong-left)
  (auto simp: eq-commute space-pair-measure image-iff cong: conj-cong)
qed

show ?thesis
proof (rule borel-measurable-LIMSEQ-metric)
  fix i
  note [measurable] = integrable-probability-kernel-pred'[OF assms]
  have [measurable]:(SIGMA x:space X. { $y \in \text{space } Y. s\ i\ (x, y) = s\ i\ (a, b)$ })
 $\in$  sets (X  $\otimes_M$  Y) for a b
  proof -
    have (SIGMA x:space X. { $y \in \text{space } Y. s\ i\ (x, y) = s\ i\ (a, b)$ }) = space (X
 $\otimes_M$  Y)  $\cap$  s\ i\ \text{'} {s\ i\ (a, b)}
    by(auto simp: space-pair-measure)
    thus ?thesis
    using s'[of i] by simp
  qed
  show f' i  $\in$  borel-measurable X
  by (auto simp : eq kernel-space f'-def cong: measurable-cong if-cong intro!:
borel-measurable-sum measurable-If borel-measurable-scaleR measure-measurable')
  next
  fix x
  assume x: $x \in \text{space } X$ 
  have ( $\lambda i. \text{Bochner-Integration.simple-bochner-integral } (\kappa\ x) (\lambda y. s\ i\ (x, y))$ )
 $\longrightarrow$  ( $\int y. f\ (x, y) \partial(\kappa\ x)$ ) if int-f:integrable ( $\kappa$  x) (curry f x)
  proof -
    have int-2f: integrable ( $\kappa$  x) ( $\lambda y. 2 \ast \text{norm } (f\ (x, y))$ )
    using int-f by(auto simp: curry-def)
    have ( $\lambda i. \text{integral}^L (\kappa\ x) (\lambda y. s\ i\ (x, y))$ )  $\longrightarrow$  integralL ( $\kappa$  x) (curry f x)
    proof (rule integral-dominated-convergence)
      show curry f x  $\in$  borel-measurable ( $\kappa$  x)
      using int-f by auto
    next
      show  $\wedge i. (\lambda y. s\ i\ (x, y)) \in \text{borel-measurable } (\kappa\ x)$ 
      using x kernel-sets by auto
    next
      show AE xa in  $\kappa$  x. ( $\lambda i. s\ i\ (x, xa)$ )  $\longrightarrow$  curry f x xa
      using x  $\ast(1)$  kernel-space by(auto simp: curry-def)
    next

```

```

    show  $\bigwedge i. AE\ xa\ in\ \kappa\ x.\ norm\ (s\ i\ (x,\ xa)) \leq 2 * norm\ (f\ (x,\ xa))$ 
      using  $x * (2)\ kernel\text{-}space\ by\ auto$ 
  qed fact
  moreover have  $integral^L\ (\kappa\ x)\ (\lambda y.\ s\ i\ (x,\ y)) = Bochner\text{-}Integration.\ simple\text{-}bochner\text{-}integral$ 
     $(\kappa\ x)\ (\lambda y.\ s\ i\ (x,\ y))$  for  $i$ 
  proof -
    have  $Bochner\text{-}Integration.\ simple\text{-}bochner\text{-}integrable\ (\kappa\ x)\ (\lambda y.\ s\ i\ (x,\ y))$ 
    proof (rule  $simple\text{-}bochner\text{-}integrableI\text{-}bounded$ )
      have  $(\lambda y.\ s\ i\ (x,\ y))\ 'space\ Y \subseteq s\ i\ '(space\ X \times space\ Y)$ 
        using  $x\ by\ auto$ 
      then show  $simple\text{-}function\ (\kappa\ x)\ (\lambda y.\ s\ i\ (x,\ y))$ 
        using  $simple\text{-}functionD(1)[OF\ s(1),\ of\ i]\ x\ kernel\text{-}space$ 
      by (intro  $simple\text{-}function\text{-}borel\text{-}measurable$ ) (auto simp:  $space\text{-}pair\text{-}measure$ 
  dest:  $finite\text{-}subset$ )
    next
      have  $(\int^+ y.\ ennreal\ (norm\ (s\ i\ (x,\ y)))\ \partial\kappa\ x) \leq (\int^+ y.\ 2 * norm\ (f$ 
     $(x,\ y)))\ \partial\kappa\ x$ 
        using  $x * (2)\ kernel\text{-}space\ by\ (intro\ nn\text{-}integral\text{-}mono)\ auto$ 
      also have  $\dots < \infty$ 
        using  $int\text{-}2f\ unfolding\ integrable\text{-}iff\text{-}bounded\ by\ simp$ 
      finally show  $(\int^+ y.\ ennreal\ (norm\ (s\ i\ (x,\ y)))\ \partial\kappa\ x) < \infty .$ 
    qed
    then show ?thesis
      by (rule  $simple\text{-}bochner\text{-}integrable\text{-}eq\text{-}integral[symmetric]$ )
  qed
  ultimately show ?thesis
    by (simp add:  $curry\text{-}def$ )
  qed
  thus  $(\lambda i.\ f'\ i\ x) \longrightarrow (\int y.\ f\ (x,\ y)\ \partial(\kappa\ x))$ 
    by (cases  $integrable\ (\kappa\ x)\ (curry\ f\ x)$ ) (simp-all add:  $f'\text{-}def\ not\text{-}integrable\text{-}integral\text{-}eq$ 
   $curry\text{-}def$ )
  qed
  qed

```

lemma(in $s\text{-}finite\text{-}kernel$) $integrable\text{-}measurable\text{-}pred[measurable\ (raw)]$:

```

  fixes  $f :: - \Rightarrow - \Rightarrow - :: \{banach,\ second\text{-}countable\text{-}topology\}$ 
  assumes  $[measurable]:\ case\text{-}prod\ f \in borel\text{-}measurable\ (X \otimes_M Y)$ 
  shows  $Measurable.\ pred\ X\ (\lambda x.\ integrable\ (\kappa\ x)\ (f\ x))$ 
  proof (cases  $space\ X = \{\}$ )
    case True
      from  $space\text{-}empty[OF\ this]$  show ?thesis
        by simp
    next
      case h:False
        obtain  $ki\ where\ ki:\bigwedge i.\ subprob\text{-}kernel\ X\ Y\ (ki\ i) \wedge x\ A.\ x \in space\ X \implies \kappa\ x$ 
           $A = (\sum i.\ ki\ i\ x\ A)$ 
          using  $s\text{-}finite\text{-}kernels\ by\ metis$ 
        have  $[simp]:\ integrable\ (\kappa\ x)\ (f\ x) = ((\sum i.\ \int^+ y.\ ennreal\ (norm\ (f\ x\ y))\ \partial ki\ i\ x)$ 
           $< \infty)$  if  $x \in space\ X$  for  $x$ 

```

using $ki(1)$ *nn-integral-measure-suminf*[of $\lambda i. ki\ i\ x\ \kappa\ x, OF - ki(2)$] *that kernel-sets*

by(*auto simp: integrable-iff-bounded subprob-kernel-def measure-kernel-def*)

note [*measurable*] = *nn-integral-measurable-subprob-algebra2*

show *?thesis*

by(*rule measurable-cong*[**where** $g = \lambda x. (\sum i. \int^+ y. \text{ennreal } (\text{norm } (f\ x\ y))\ \partial(ki\ i\ x)) < \infty, \text{THEN } \text{iffD2}]$) (*insert* $ki(1)$, *auto simp: subprob-kernel-def'*)

qed

lemma(*in s-finite-kernel*) *integral-measurable-f:*

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

assumes [*measurable*]: $f \in \text{borel-measurable } (X \otimes_M Y)$

shows $(\lambda x. \int y. f\ x\ y\ \partial(\kappa\ x)) \in \text{borel-measurable } X$

proof –

obtain ki **where** $ki: \bigwedge i. \text{subprob-kernel } X\ Y\ (ki\ i) \wedge x\ A. x \in \text{space } X \implies \kappa\ x\ A = (\sum i. ki\ i\ x\ A)$

using *s-finite-kernels* **by** *metis*

note [*measurable*] = *integral-measurable-subprob-algebra2*

show *?thesis*

proof(*rule measurable-cong*[**where** $f = (\lambda x. \text{if } \text{integrable } (\kappa\ x)\ (f\ x) \text{ then } (\sum i. \int y. f\ x\ y\ \partial(ki\ i\ x)) \text{ else } 0), \text{THEN } \text{iffD1}]$)

fix x

assume $h: x \in \text{space } X$

{

assume $h': \text{integrable } (\kappa\ x)\ (f\ x)$

have $(\sum i. \int y. f\ x\ y\ \partial(ki\ i\ x)) = (\int y. f\ x\ y\ \partial(\kappa\ x))$

using *lebesgue-integral-measure-suminf*[of $\lambda i. ki\ i\ x\ \kappa\ x, OF - ki(2)\ h'$] $ki(1)$ *kernel-sets*[$OF\ h$] h

by(*auto simp: subprob-kernel-def measure-kernel-def*)

}

thus $(\text{if } \text{integrable } (\kappa\ x)\ (f\ x) \text{ then } (\sum i. \int y. f\ x\ y\ \partial(ki\ i\ x)) \text{ else } 0) = (\int y. f\ x\ y\ \partial(\kappa\ x))$

using *not-integrable-integral-eq* **by** *auto*

qed(*insert* $ki(1)$, *auto simp: subprob-kernel-def'*)

qed

lemma(*in s-finite-kernel*) *integral-measurable-f':*

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

assumes [*measurable*]: $f \in \text{borel-measurable } (X \otimes_M Y)$

shows $(\lambda x. \int y. f\ (x, y)\ \partial(\kappa\ x)) \in \text{borel-measurable } X$

using *integral-measurable-f*[of *curry* f] **by** *simp*

lemma(*in s-finite-kernel*)

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

assumes [*measurable-cong*]: $\mu = \text{sets } X$

and *integrable* $(\mu \gg_{\kappa} f)$

shows *integrable-bind-kernelD1*: *integrable* $\mu\ (\lambda x. \int y. \text{norm } (f\ y)\ \partial\kappa\ x)$ (*is ?g1*)

and *integrable-bind-kernelD1'*: *integrable* μ $(\lambda x. \int y. f y \partial \kappa x)$ (**is** ?g1')
and *integrable-bind-kernelD2*: *AE* x *in* μ . *integrable* $(\kappa x) f$ (**is** ?g2)
and *integrable-bind-kernelD3*: *space* $X \neq \{\}$ $\implies f \in \text{borel-measurable } Y$ (**is**
 $- \implies ?g3$)
proof –
show *1:space* $X \neq \{\} \implies ?g3$
using *assms(2) sets-bind-kernel[OF - assms(1)]* **by** (*simp add: integrable-iff-bounded*
cong: measurable-cong-sets)
have *integrable* μ $(\lambda x. \int y. \text{norm } (f y) \partial \kappa x) \wedge \text{integrable } \mu$ $(\lambda x. \int y. f y \partial \kappa x)$
 $\wedge (AE x \text{ in } \mu. \text{integrable } (\kappa x) f)$
proof (*cases space* $X = \{\}$)
assume *ne: space* $X \neq \{\}$
then have *space* $\mu \neq \{\}$ **by** (*simp add: sets-eq-imp-space-eq[OF assms(1)]*)
note $h = \text{integral-measurable-f[measurable] sets-bind-kernel[OF ne assms(1),measurable-cong]}$
have *g2: ?g2*
unfolding *integrable-iff-bounded AE-conj-iff*
proof *safe*
show *AE* x *in* μ . $f \in \text{borel-measurable } (\kappa x)$
using *assms(2)* **by** (*auto simp: sets-eq-imp-space-eq[OF assms(1)] measurable-*
able-cong-sets[OF kernel-sets])
next
note *nn-integral-measurable-f[measurable]*
have *AE* x *in* μ . $(\int^+ x. \text{ennreal } (\text{norm } (f x)) \partial \kappa x) \neq \infty$
by (*rule nn-integral-PInf-AE,insert assms(2)*) (*auto simp: integrable-iff-bounded*
nn-integral-bind-kernel[OF - assms(1)] intro! :)
thus *AE* x *in* μ . $(\int^+ x. \text{ennreal } (\text{norm } (f x)) \partial \kappa x) < \infty$
by (*simp add: top.not-eq-extremum*)
qed
have [*simp*]: $(\int^+ x. \int^+ x. \text{ennreal } (\text{norm } (f x)) \partial \kappa x \partial \mu) = (\int^+ x. \text{ennreal}$
 $(\int y. \text{norm } (f y) \partial \kappa x) \partial \mu)$
using *g2* **by** (*auto intro!: nn-integral-cong-AE simp: nn-integral-eq-integral*)
have *g1: ?g1*
using *assms(2)* **by** (*auto simp: integrable-iff-bounded measurable-cong-sets[OF*
h(2)] measurable-cong-sets[OF assms(1)] nn-integral-bind-kernel[OF - assms(1)])
have ?g1'
using *assms(2)* **by** (*auto intro!: Bochner-Integration.integrable-bound[OF g1]*)
with *g2 g1* **show** ?thesis
by *auto*
qed (*auto simp: space-empty[of μ] sets-eq-imp-space-eq[OF assms(1)] integrable-iff-bounded*
nn-integral-empty)
thus ?g1 ?g1' ?g2
by *auto*
qed

lemma(**in** *s-finite-kernel*)
fixes $f :: - \Rightarrow - :: \{\text{banach, second-countable-topology}\}$
assumes [*measurable-cong*]: *sets* $\mu = \text{sets } X$
and [*measurable*]: *AE* x *in* μ . *integrable* $(\kappa x) f$ *integrable* μ $(\lambda x. \int y. \text{norm } (f$
 $y) \partial \kappa x) f \in \text{borel-measurable } Y$

shows *integrable-bind-kernel*: $\text{integrable } (\mu \gg_k \kappa) f$
and *integral-bind-kernel*: $(\int y. f y \partial(\mu \gg_k \kappa)) = (\int x. (\int y. f y \partial \kappa x) \partial \mu)$ (**is**
?eq)

proof –

have *integrable* $(\mu \gg_k \kappa) f \wedge (\int y. f y \partial(\mu \gg_k \kappa)) = (\int x. (\int y. f y \partial \kappa x) \partial \mu)$

proof (*cases space X = {}*)

assume *ne*: *space X ≠ {}*

note *sets-bind[measurable-cong]* = *sets-bind-kernel[OF ne assms(1)]*

note *h* = *integral-measurable-f[measurable]*

have *1*: *integrable* $(\mu \gg_k \kappa) f$

unfolding *integrable-iff-bounded*

proof

show $(\int^+ x. \text{ennreal } (\text{norm } (f x)) \partial(\mu \gg_k \kappa)) < \infty$ (**is** *?l < -*)

proof –

have *?l* = $(\int^+ x. \text{ennreal } (\int y. \text{norm } (f y) \partial \kappa x) \partial \mu)$

using *assms(2)* **by** (*auto intro!*: *nn-integral-cong-AE simp: nn-integral-eq-integral*
simp: nn-integral-bind-kernel[OF - assms(1)])

also have $\dots < \infty$

using *assms(3)* **by** (*auto simp: integrable-iff-bounded*)

finally show *?thesis* .

qed

qed *simp*

then have *?eq*

proof *induction*

case *h[measurable]*: (*base A c*)

hence *1*: *integrable* $(\mu \gg_k \kappa)$ (*indicat-real A*)

by *simp*

have *2*: *integrable* μ ($\lambda x. \text{measure } (\kappa x) A$)

by (*rule Bochner-Integration.integrable-cong[where f=λx. Sigma-Algebra.measure*
 $(\kappa x) (A \cap \text{space } (\kappa x)), \text{THEN iffD1, OF refl}]
(insert h integrable-bind-kernelD1[OF assms(1) 1] sets-eq-imp-space-eq[OF
kernel-sets], auto simp: sets-eq-imp-space-eq[OF assms(1)] sets-eq-imp-space-eq[OF
kernel-sets] sets-bind))$

have *AE x in μ. emeasure* $(\kappa x) A \neq \infty$

by (*rule nn-integral-PInf-AE, insert h*) (*auto simp: emeasure-bind-kernel[OF*
assms(1) - ne] sets-bind)

hence *0*: *AE x in μ. emeasure* $(\kappa x) A < \infty$

by (*simp add: top.not-eq-extremum*)

have $(\int x. (\int y. \text{indicat-real } A y *_R c \partial \kappa x) \partial \mu) = (\int x. \text{measure } (\kappa x) A *_R$
 $c \partial \mu)$

using *h integrable-bind-kernelD2[OF assms(1) integrable-real-indicator, of*
 $A]$

by (*auto intro!*: *integral-cong-AE simp: sets-eq-imp-space-eq[OF kernel-sets]*
sets-bind sets-eq-imp-space-eq[OF assms(1)])

also have $\dots = (\int x. \text{measure } (\kappa x) A \partial \mu) *_R c$

using *2* **by** (*auto intro!*: *integral-scaleR-left*)

finally show *?case*

using *h* **by** (*auto simp: measure-bind-kernel[OF assms(1) - ne 0] sets-bind*)

next

```

case ih:(add f g)
show ?case
using ih(1,2) integrable-bind-kernelD2[OF assms(1) ih(1)] integrable-bind-kernelD2[OF
assms(1) ih(2)]
by(auto simp: ih(3,4) Bochner-Integration.integral-add[OF integrable-bind-kernelD1 '[OF
assms(1) ih(1)] integrable-bind-kernelD1 '[OF assms(1) ih(2)],symmetric] intro!:
integral-cong-AE)
next
case ih:(lim f fn)
show ?case (is ?lhs = ?rhs)
proof -
  have conv: AE x in  $\mu$ .  $(\lambda n. \int y. fn\ n\ y\ \partial\kappa\ x) \longrightarrow (\int y. f\ y\ \partial\kappa\ x)$ 
  proof -
    have conv: AE x in  $\mu$ . integrable  $(\kappa\ x)\ f \longrightarrow (\lambda n. \int y. fn\ n\ y\ \partial\kappa\ x) \longrightarrow$ 
     $(\int y. f\ y\ \partial\kappa\ x)$ 
    proof
      fix x
      assume h: x  $\in$  space  $\mu$ 
      then show integrable  $(\kappa\ x)\ f \longrightarrow (\lambda n. \int y. fn\ n\ y\ \partial\kappa\ x) \longrightarrow (\int y. f\ y$ 
       $\partial\kappa\ x)$ 
      using ih by(auto intro!: integral-dominated-convergence[where w= $\lambda x. 2 * norm$ 
       $(f\ x)$ ] simp: sets-eq-imp-space-eq[OF sets-bind] sets-eq-imp-space-eq[OF kernel-sets
      [OF h[simplified sets-eq-imp-space-eq[OF assms(1)]]]]] sets-eq-imp-space-eq[OF
      assms(1)])
      qed
      with conv integrable-bind-kernelD2[OF assms(1) ih(4)]
      show ?thesis by fastforce
    qed
  have ?lhs = lim  $(\lambda n. \int y. fn\ n\ y\ \partial(\mu \gg_{\kappa} \kappa))$ 
  by(rule limI[OF integral-dominated-convergence[where w= $\lambda x. 2 * norm$ 
   $(f\ x)$ ],symmetric]) (use ih in auto)
  also have ... = lim  $(\lambda n. (\int y. (f\ y. fn\ n\ y\ \partial\kappa\ x)\ \partial\ \mu))$ 
  by(simp add: ih)
  also have ... =  $(\int x. \lim (\lambda n. \int y. fn\ n\ y\ \partial\kappa\ x)\ \partial\ \mu)$ 
  proof(rule limI[OF integral-dominated-convergence[where w= $\lambda x. \int y. 2 * norm$ 
   $(f\ y)\ \partial\kappa\ x$ ]])
  fix n
  show AE x in  $\mu$ . norm  $(\int y. fn\ n\ y\ \partial\kappa\ x) \leq (\int y. 2 * norm (f\ y)\ \partial\kappa\ x)$ 
  by(rule AE-mp[OF integrable-bind-kernelD2[OF assms(1) ih(1),of
  n] AE-mp[OF integrable-bind-kernelD2[OF assms(1) ih(4)]]],standard+,rule order.trans
  [OF integral-norm-bound integral-mono[of  $\kappa - \lambda y. norm (fn\ n\ y) - ,OF$ 
   $- - ih(3)$ ][simplified sets-eq-imp-space-eq[OF sets-bind]]]])
  (auto simp: sets-eq-imp-space-eq[OF assms(1)] sets-eq-imp-space-eq[OF
  kernel-sets])
  qed(use ih integrable-bind-kernelD1[OF assms(1) ih(4)] conv limI in
  auto,fastforce)
  also have ... = ?rhs
  using ih conv limI by(auto intro!: integral-cong-AE, blast)
  finally show ?thesis .

```

```

    qed
  qed
  with 1 show ?thesis
  by auto
  qed(auto simp: bind-kernel-def space-empty[of  $\mu$ ] sets-eq-imp-space-eq[OF assms(1)]
integrable-iff-bounded nn-integral-empty Bochner-Integration.integral-empty)
  thus integrable ( $\mu \gg_{\kappa} \nu$ ) f ?eq
  by auto
  qed
end

```

3 Quasi-Borel Spaces

```

theory QuasiBorel
imports HOL-Probability.Probability
begin

```

3.1 Definitions

3.1.1 Quasi-Borel Spaces

```

definition qbs-closed1 :: (real  $\Rightarrow$  'a) set  $\Rightarrow$  bool
  where qbs-closed1 Mx  $\equiv$  ( $\forall a \in Mx. \forall f \in$  (borel :: real measure)  $\rightarrow_M$  (borel ::
real measure).  $a \circ f \in Mx$ )

```

```

definition qbs-closed2 :: ['a set, (real  $\Rightarrow$  'a) set]  $\Rightarrow$  bool
  where qbs-closed2 X Mx  $\equiv$  ( $\forall x \in X. (\lambda r. x) \in Mx$ )

```

```

definition qbs-closed3 :: (real  $\Rightarrow$  'a) set  $\Rightarrow$  bool
  where qbs-closed3 Mx  $\equiv$  ( $\forall P::real \Rightarrow nat. \forall Fi::nat \Rightarrow real \Rightarrow 'a.$ 
 $(P \in$  borel  $\rightarrow_M$  count-space UNIV)  $\longrightarrow$  ( $\forall i. Fi i \in Mx$ )  $\longrightarrow$ 
 $(\lambda r. Fi (P r) r) \in Mx$ )

```

```

lemma separate-measurable:
  fixes P :: real  $\Rightarrow$  nat
  assumes  $\bigwedge i. P - \{i\} \in$  sets borel
  shows  $P \in$  borel  $\rightarrow_M$  count-space UNIV
  by (auto simp add: assms measurable-count-space-eq-countable)

```

```

lemma measurable-separate:
  fixes P :: real  $\Rightarrow$  nat
  assumes  $P \in$  borel  $\rightarrow_M$  count-space UNIV
  shows  $P - \{i\} \in$  sets borel
  by (metis assms borel-singleton measurable-sets-borel sets.empty-sets sets-borel-eq-count-space)

```

```

definition is-quasi-borel X Mx  $\longleftrightarrow$   $Mx \subseteq$  UNIV  $\rightarrow$  X  $\wedge$  qbs-closed1 Mx  $\wedge$  qbs-closed2
X Mx  $\wedge$  qbs-closed3 Mx

```

lemma *is-quasi-borel-intro*[simp]:
assumes $Mx \subseteq UNIV \rightarrow X$
and *qbs-closed1* Mx *qbs-closed2* X Mx *qbs-closed3* Mx
shows *is-quasi-borel* X Mx
using *assms* **by**(*simp add: is-quasi-borel-def*)

typedef *'a quasi-borel* = $\{(X::'a \text{ set}, Mx). \text{is-quasi-borel } X \text{ } Mx\}$
proof
show $(UNIV, UNIV) \in \{(X::'a \text{ set}, Mx). \text{is-quasi-borel } X \text{ } Mx\}$
by (*simp add: is-quasi-borel-def qbs-closed1-def qbs-closed2-def qbs-closed3-def*)
qed

definition *qbs-space* :: *'a quasi-borel* \Rightarrow *'a set* **where**
qbs-space $X \equiv \text{fst } (\text{Rep-quasi-borel } X)$

definition *qbs-Mx* :: *'a quasi-borel* \Rightarrow (*real* \Rightarrow *'a*) *set* **where**
qbs-Mx $X \equiv \text{snd } (\text{Rep-quasi-borel } X)$

declare [[*coercion qbs-space*]]

lemma *qbs-decomp* : $(\text{qbs-space } X, \text{qbs-Mx } X) \in \{(X::'a \text{ set}, Mx). \text{is-quasi-borel } X \text{ } Mx\}$
by (*simp add: qbs-space-def qbs-Mx-def Rep-quasi-borel[simplified]*)

lemma *qbs-Mx-to-X*:
assumes $\alpha \in \text{qbs-Mx } X$
shows $\alpha \in \text{qbs-space } X$
using *qbs-decomp* **assms** **by**(*auto simp: is-quasi-borel-def*)

lemma *qbs-closed1I*:
assumes $\bigwedge \alpha f. \alpha \in Mx \implies f \in \text{borel} \rightarrow_M \text{borel} \implies \alpha \circ f \in Mx$
shows *qbs-closed1* Mx
using *assms* **by**(*simp add: qbs-closed1-def*)

lemma *qbs-closed1-dest*[simp]:
assumes $\alpha \in \text{qbs-Mx } X$
and $f \in \text{borel} \rightarrow_M \text{borel}$
shows $\alpha \circ f \in \text{qbs-Mx } X$
using *assms qbs-decomp* **by** (*auto simp add: is-quasi-borel-def qbs-closed1-def*)

lemma *qbs-closed1-dest'*[simp]:
assumes $\alpha \in \text{qbs-Mx } X$
and $f \in \text{borel} \rightarrow_M \text{borel}$
shows $(\lambda r. \alpha (f r)) \in \text{qbs-Mx } X$
using *qbs-closed1-dest[OF assms]* **by** (*simp add: comp-def*)

lemma *qbs-closed2I*:
assumes $\bigwedge x. x \in X \implies (\lambda r. x) \in Mx$
shows *qbs-closed2* X Mx

using *assms* **by**(*simp add: qbs-closed2-def*)

lemma *qbs-closed2-dest[simp]*:
assumes $x \in \text{qbs-space } X$
shows $(\lambda r. x) \in \text{qbs-Mx } X$
using *assms qbs-decomp[of X]* **by** (*auto simp add: is-quasi-borel-def qbs-closed2-def*)

lemma *qbs-closed3I*:
assumes $\bigwedge(P :: \text{real} \Rightarrow \text{nat}) Fi. P \in \text{borel} \rightarrow_M \text{count-space UNIV} \Longrightarrow (\bigwedge i. Fi \ i \in \text{Mx})$
 $\Longrightarrow (\lambda r. Fi (P r) r) \in \text{Mx}$
shows *qbs-closed3 Mx*
using *assms* **by**(*auto simp: qbs-closed3-def*)

lemma *qbs-closed3I'*:
assumes $\bigwedge(P :: \text{real} \Rightarrow \text{nat}) Fi. (\bigwedge i. P -' \{i\} \in \text{sets borel}) \Longrightarrow (\bigwedge i. Fi \ i \in \text{Mx})$
 $\Longrightarrow (\lambda r. Fi (P r) r) \in \text{Mx}$
shows *qbs-closed3 Mx*
using *assms* **by**(*auto intro!: qbs-closed3I dest: measurable-separate*)

lemma *qbs-closed3-dest[simp]*:
fixes $P :: \text{real} \Rightarrow \text{nat}$ **and** $Fi :: \text{nat} \Rightarrow \text{real} \Rightarrow -$
assumes $P \in \text{borel} \rightarrow_M \text{count-space UNIV}$
and $\bigwedge i. Fi \ i \in \text{qbs-Mx } X$
shows $(\lambda r. Fi (P r) r) \in \text{qbs-Mx } X$
using *assms qbs-decomp[of X]* **by** (*auto simp add: is-quasi-borel-def qbs-closed3-def*)

lemma *qbs-closed3-dest'*:
fixes $P :: \text{real} \Rightarrow \text{nat}$ **and** $Fi :: \text{nat} \Rightarrow \text{real} \Rightarrow -$
assumes $\bigwedge i. P -' \{i\} \in \text{sets borel}$
and $\bigwedge i. Fi \ i \in \text{qbs-Mx } X$
shows $(\lambda r. Fi (P r) r) \in \text{qbs-Mx } X$
using *qbs-closed3-dest[OF separate-measurable[OF assms(1)] assms(2)]* .

lemma *qbs-closed3-dest2*:
assumes *countable I*
and [*measurable*]: $P \in \text{borel} \rightarrow_M \text{count-space } I$
and $\bigwedge i. i \in I \Longrightarrow Fi \ i \in \text{qbs-Mx } X$
shows $(\lambda r. Fi (P r) r) \in \text{qbs-Mx } X$

proof –
have $0: I \neq \{\}$
using *measurable-empty-iff[of count-space I P borel] assms(2)*
by *fastforce*
define P' **where** $P' \equiv \text{to-nat-on } I \circ P$
define Fi' **where** $Fi' \equiv Fi \circ (\text{from-nat-into } I)$
have $1: P' \in \text{borel} \rightarrow_M \text{count-space UNIV}$
by(*simp add: P'-def*)
have $2: \bigwedge i. Fi' \ i \in \text{qbs-Mx } X$

using *assms(3)* *from-nat-into*[*OF 0*] **by**(*simp add: Fi'-def*)
have $(\lambda r. Fi' (P' r) r) \in qbs-Mx X$
using *1 2 measurable-separate* **by** *auto*
thus *?thesis*
using *from-nat-into-to-nat-on*[*OF assms(1)*] *measurable-space*[*OF assms(2)*]
by(*auto simp: Fi'-def P'-def*)
qed

lemma *qbs-closed3-dest2'*:
assumes *countable I*
and [*measurable*]: $P \in borel \rightarrow_M count-space I$
and $\bigwedge i. i \in range P \implies Fi i \in qbs-Mx X$
shows $(\lambda r. Fi (P r) r) \in qbs-Mx X$
proof –
have *0:range P* $\cap I = range P$
using *measurable-space*[*OF assms(2)*] **by** *auto*
have *1:P* $\in borel \rightarrow_M count-space (range P)$
using *restrict-count-space*[*of I range P*] *measurable-restrict-space2*[*OF - assms(2), of range P*]
by(*simp add: 0*)
have *2:countable (range P)*
using *countable-Int2*[*OF assms(1), of range P*]
by(*simp add: 0*)
show *?thesis*
by(*auto intro!: qbs-closed3-dest2*[*OF 2 1 assms(3)*])
qed

lemma *qbs-Mx-indicat*:
assumes $S \in sets borel$ $\alpha \in qbs-Mx X$ $\beta \in qbs-Mx X$
shows $(\lambda r. if r \in S then \alpha r else \beta r) \in qbs-Mx X$
proof –
have $(\lambda r::real. if r \in S then \alpha r else \beta r) = (\lambda r. (\lambda b. if b then \alpha else \beta) (r \in S) r)$
by(*auto simp: indicator-def*)
also have $\dots \in qbs-Mx X$
by(*rule qbs-closed3-dest2*[**where** $I=UNIV$ **and** $Fi=\lambda b. if b then \alpha else \beta$]) (*use assms in auto*)
finally show *?thesis* .
qed

lemma *qbs-space-Mx*: $qbs-space X = \{\alpha x \mid x \alpha. \alpha \in qbs-Mx X\}$
proof *safe*
fix x
assume *1:x* $\in qbs-space X$
show $\exists xa \alpha. x = \alpha xa \wedge \alpha \in qbs-Mx X$
by(*auto intro!: exI*[**where** $x=0$] *exI*[**where** $x=(\lambda r. x)$] *simp: 1*)
qed(*simp add: qbs-Mx-to-X*)

lemma *qbs-space-eq-Mx*:

assumes $qbs-Mx\ X = qbs-Mx\ Y$
shows $qbs-space\ X = qbs-space\ Y$
by(*simp add: qbs-space-Mx assms*)

lemma *qbs-eqI*:

assumes $qbs-Mx\ X = qbs-Mx\ Y$
shows $X = Y$
by (*metis Rep-quasi-borel-inverse prod.exhaust-sel qbs-Mx-def qbs-space-def assms*
qbs-space-eq-Mx[OF assms])

3.1.2 Empty Space

definition *empty-quasi-borel* :: 'a quasi-borel **where**
empty-quasi-borel \equiv *Abs-quasi-borel* ($\{\}, \{\}$)

lemma

shows *eqb-space*[*simp*]: $qbs-space\ empty-quasi-borel = (\{\} :: 'a\ set)$
and *qbs-Mx*[*simp*]: $qbs-Mx\ empty-quasi-borel = (\{\} :: (real \Rightarrow 'a)\ set)$
proof –
have *Rep-quasi-borel* *empty-quasi-borel* = ($\{\} :: 'a\ set, \{\}$)
using *Abs-quasi-borel-inverse* **by**(*auto simp add: Abs-quasi-borel-inverse empty-quasi-borel-def*
qbs-closed1-def qbs-closed2-def qbs-closed3-def is-quasi-borel-def)
thus $qbs-space\ empty-quasi-borel = (\{\} :: 'a\ set)$ $qbs-Mx\ empty-quasi-borel = (\{\} :: (real \Rightarrow 'a)\ set)$
by(*auto simp add: qbs-space-def qbs-Mx-def*)
qed

lemma *qbs-empty-equiv* : $qbs-space\ X = \{\} \longleftrightarrow qbs-Mx\ X = \{\}$

proof *safe*

fix x
assume $qbs-Mx\ X = \{\}$
and $h : x \in qbs-space\ X$
have $(\lambda r. x) \in qbs-Mx\ X$
using h **by** *simp*
thus $x \in \{\}$ **using** $\langle qbs-Mx\ X = \{\} \rangle$ **by** *simp*
qed(*use qbs-Mx-to-X in blast*)

lemma *empty-quasi-borel-iff*:

$qbs-space\ X = \{\} \longleftrightarrow X = empty-quasi-borel$
by(*auto intro!: qbs-eqI simp: qbs-empty-equiv*)

3.1.3 Unit Space

definition *unit-quasi-borel* :: *unit quasi-borel* (1_Q) **where**
unit-quasi-borel \equiv *Abs-quasi-borel* (*UNIV*, *UNIV*)

lemma

shows *unit-qbs-space*[*simp*]: $qbs-space\ unit-quasi-borel = \{\ ()\}$
and *unit-qbs-Mx*[*simp*]: $qbs-Mx\ unit-quasi-borel = \{\lambda r. ()\}$
proof –

have *Rep-quasi-borel unit-quasi-borel* = (*UNIV*, *UNIV*)
using *Abs-quasi-borel-inverse* **by**(*auto simp add: unit-quasi-borel-def qbs-closed1-def qbs-closed2-def qbs-closed3-def is-quasi-borel-def*)
thus *qbs-space unit-quasi-borel* = $\{()\}$ *qbs-Mx unit-quasi-borel* = $\{\lambda r. ()\}$
by(*auto simp add: qbs-space-def qbs-Mx-def UNIV-unit*)
qed

3.1.4 Sub-Spaces

definition *sub-qbs* :: [*'a quasi-borel, 'a set*] \Rightarrow *'a quasi-borel* **where**
sub-qbs X U \equiv *Abs-quasi-borel* (*qbs-space X* \cap *U*, $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$)

lemma

shows *sub-qbs-space*: *qbs-space* (*sub-qbs X U*) = *qbs-space X* \cap *U*
and *sub-qbs-Mx*: *qbs-Mx* (*sub-qbs X U*) = $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$
proof –
have *qbs-closed1* $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$ *qbs-closed2* (*qbs-space X* \cap *U*) $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$
qbs-closed3 $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$
unfolding *qbs-closed1-def qbs-closed2-def qbs-closed3-def* **by** *auto*
hence *Rep-quasi-borel* (*sub-qbs X U*) = (*qbs-space X* \cap *U*, $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$)
by(*auto simp: sub-qbs-def is-quasi-borel-def qbs-Mx-to-X intro!: Abs-quasi-borel-inverse*)
thus *qbs-space* (*sub-qbs X U*) = *qbs-space X* \cap *U* *qbs-Mx* (*sub-qbs X U*) = $\{\alpha. \alpha \in \text{qbs-Mx } X \wedge (\forall r. \alpha r \in U)\}$
by(*simp-all add: qbs-Mx-def qbs-space-def*)
qed

lemma *sub-qbs*:

assumes $U \subseteq \text{qbs-space } X$
shows (*qbs-space* (*sub-qbs X U*), *qbs-Mx* (*sub-qbs X U*)) = (*U*, $\{f \in \text{UNIV} \rightarrow U. f \in \text{qbs-Mx } X\}$)
using *assms* **by** (*auto simp: sub-qbs-space sub-qbs-Mx*)

lemma *sub-qbs-ident*: *sub-qbs X* (*qbs-space X*) = *X*

by(*auto intro!: qbs-eqI simp: sub-qbs-Mx qbs-Mx-to-X*)

lemma *sub-qbs-sub-qbs*: *sub-qbs* (*sub-qbs X A*) *B* = *sub-qbs X* (*A* \cap *B*)

by(*auto intro!: qbs-eqI simp: sub-qbs-Mx sub-qbs-space*)

3.1.5 Image Spaces

definition *map-qbs* :: [*'a* \Rightarrow *'b*] \Rightarrow *'a quasi-borel* \Rightarrow *'b quasi-borel* **where**
map-qbs f X = *Abs-quasi-borel* (*f* ' (*qbs-space X*), $\{f \circ \alpha \mid \alpha. \alpha \in \text{qbs-Mx } X\}$)

lemma

shows *map-qbs-space*: *qbs-space* (*map-qbs f X*) = *f* ' (*qbs-space X*)
and *map-qbs-Mx*: *qbs-Mx* (*map-qbs f X*) = $\{f \circ \alpha \mid \alpha. \alpha \in \text{qbs-Mx } X\}$
proof –

```

have {f ∘ α | α. α ∈ qbs-Mx X} ⊆ UNIV → f ' (qbs-space X)
  using qbs-Mx-to-X by fastforce
moreover have qbs-closed1 {f ∘ α | α. α ∈ qbs-Mx X}
  unfolding qbs-closed1-def using qbs-closed1-dest by (fastforce simp: comp-def)
moreover have qbs-closed2 (f ' (qbs-space X)) {f ∘ α | α. α ∈ qbs-Mx X}
  unfolding qbs-closed2-def by fastforce
moreover have qbs-closed3 {f ∘ α | α. α ∈ qbs-Mx X}
proof (rule qbs-closed3I')
  fix P :: real ⇒ nat and Fi
  assume h: ∧ i::nat. P - ' {i} ∈ sets borel
    ∧ i::nat. Fi i ∈ {f ∘ α | α. α ∈ qbs-Mx X}
  then obtain α i where ha: ∧ i::nat. α i ∈ qbs-Mx X ∧ i. Fi i = f ∘ (α i)
    by auto metis
  hence 1:(λr. α i (P r) r) ∈ qbs-Mx X
    using h(1) qbs-closed3-dest' by blast
  show (λr. Fi (P r) r) ∈ {f ∘ α | α. α ∈ qbs-Mx X}
    by (auto intro!: beXI[where x=(λr. α i (P r) r)] simp add: 1 ha comp-def)
qed
ultimately have Rep-quasi-borel (map-qbs f X) = (f ' (qbs-space X), {f ∘ α | α.
α ∈ qbs-Mx X})
  unfolding map-qbs-def by (auto intro!: Abs-quasi-borel-inverse)
  thus qbs-space (map-qbs f X) = f ' (qbs-space X) qbs-Mx (map-qbs f X) = {f ∘
α | α. α ∈ qbs-Mx X}
  by (simp-all add: qbs-space-def qbs-Mx-def)
qed

```

3.1.6 Binary Product Spaces

definition *pair-qbs* :: [*'a quasi-borel, 'b quasi-borel*] ⇒ (*'a × 'b*) *quasi-borel* (**infixr** \otimes_Q 80) **where**
pair-qbs X Y = *Abs-quasi-borel* (*qbs-space* X × *qbs-space* Y, {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y})

lemma

shows *pair-qbs-space*: *qbs-space* (X \otimes_Q Y) = *qbs-space* X × *qbs-space* Y
and *pair-qbs-Mx*: *qbs-Mx* (X \otimes_Q Y) = {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y}

proof –

have {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y} ⊆ UNIV → *qbs-space* X × *qbs-space* Y

by (*auto simp: mem-Times-iff*[*of - qbs-space* X *qbs-space* Y]; *use qbs-Mx-to-X in fastforce*)

moreover have *qbs-closed1* {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y}

unfolding *qbs-closed1-def* **by** (*metis* (*no-types, lifting*) *comp-assoc mem-Collect-eq qbs-closed1-dest*)

moreover have *qbs-closed2* (*qbs-space* X × *qbs-space* Y) {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y}

unfolding *qbs-closed2-def* **by** *auto*

moreover have *qbs-closed3* {f. *fst* ∘ f ∈ *qbs-Mx* X ∧ *snd* ∘ f ∈ *qbs-Mx* Y}

```

proof(safe intro!: qbs-closed3I)
  fix P :: real  $\Rightarrow$  nat
  fix Fi :: nat  $\Rightarrow$  real  $\Rightarrow$  'a  $\times$  'b
  define Fj :: nat  $\Rightarrow$  real  $\Rightarrow$  'a where Fj  $\equiv$   $\lambda j.$ (fst  $\circ$  Fi j)
  assume  $\forall i.$  Fi i  $\in$  {f. fst  $\circ$  f  $\in$  qbs-Mx X  $\wedge$  snd  $\circ$  f  $\in$  qbs-Mx Y}
  then have  $\bigwedge i.$  Fj i  $\in$  qbs-Mx X by (simp add: Fj-def)
  moreover assume P  $\in$  borel  $\rightarrow_M$  count-space UNIV
  ultimately have ( $\lambda r.$  Fj (P r) r)  $\in$  qbs-Mx X
    by auto
  moreover have fst  $\circ$  ( $\lambda r.$  Fi (P r) r) = ( $\lambda r.$  Fj (P r) r) by (auto simp add:
Fj-def)
  ultimately show fst  $\circ$  ( $\lambda r.$  Fi (P r) r)  $\in$  qbs-Mx X by simp
next
  fix P :: real  $\Rightarrow$  nat
  fix Fi :: nat  $\Rightarrow$  real  $\Rightarrow$  'a  $\times$  'b
  define Fj :: nat  $\Rightarrow$  real  $\Rightarrow$  'b where Fj  $\equiv$   $\lambda j.$ (snd  $\circ$  Fi j)
  assume  $\forall i.$  Fi i  $\in$  {f. fst  $\circ$  f  $\in$  qbs-Mx X  $\wedge$  snd  $\circ$  f  $\in$  qbs-Mx Y}
  then have  $\bigwedge i.$  Fj i  $\in$  qbs-Mx Y by (simp add: Fj-def)
  moreover assume P  $\in$  borel  $\rightarrow_M$  count-space UNIV
  ultimately have ( $\lambda r.$  Fj (P r) r)  $\in$  qbs-Mx Y
    by auto
  moreover have snd  $\circ$  ( $\lambda r.$  Fi (P r) r) = ( $\lambda r.$  Fj (P r) r) by (auto simp add:
Fj-def)
  ultimately show snd  $\circ$  ( $\lambda r.$  Fi (P r) r)  $\in$  qbs-Mx Y by simp
qed
  ultimately have Rep-quasi-borel (X  $\otimes_Q$  Y) = (qbs-space X  $\times$  qbs-space Y,
{f. fst  $\circ$  f  $\in$  qbs-Mx X  $\wedge$  snd  $\circ$  f  $\in$  qbs-Mx Y})
  unfolding pair-qbs-def by(auto intro!: Abs-quasi-borel-inverse is-quasi-borel-intro)
  thus qbs-space (X  $\otimes_Q$  Y) = qbs-space X  $\times$  qbs-space Y qbs-Mx (X  $\otimes_Q$  Y) =
{f. fst  $\circ$  f  $\in$  qbs-Mx X  $\wedge$  snd  $\circ$  f  $\in$  qbs-Mx Y}
  by(simp-all add: qbs-space-def qbs-Mx-def)
qed

lemma pair-qbs-fst:
  assumes qbs-space Y  $\neq$  {}
  shows map-qbs fst (X  $\otimes_Q$  Y) = X
proof(rule qbs-eqI)
  obtain  $\alpha y$  where  $hy:\alpha y \in$  qbs-Mx Y
  using qbs-empty-equiv[of Y] assms by auto
  show qbs-Mx (map-qbs fst (X  $\otimes_Q$  Y)) = qbs-Mx X
  by(auto simp: map-qbs-Mx pair-qbs-Mx hy comp-def intro!: exI[where x= $\lambda r.$ 
(- r,  $\alpha y$  r)])
qed

lemma pair-qbs-snd:
  assumes qbs-space X  $\neq$  {}
  shows map-qbs snd (X  $\otimes_Q$  Y) = Y
proof(rule qbs-eqI)
  obtain  $\alpha x$  where  $hx:\alpha x \in$  qbs-Mx X

```

```

using qbs-empty-equiv[of X] assms by auto
show qbs-Mx (map-qbs snd (X  $\otimes_Q$  Y)) = qbs-Mx Y
by(auto simp: map-qbs-Mx pair-qbs-Mx hx comp-def intro!: exI[where x= $\lambda$ r.
( $\alpha$ x r, - r)])
qed

```

3.1.7 Binary Coproduct Spaces

definition *copair-qbs-Mx* :: [*'a quasi-borel, 'b quasi-borel*] \Rightarrow (*real* \Rightarrow *'a + 'b*) *set*
where

```

copair-qbs-Mx X Y  $\equiv$ 
  {g.  $\exists$  S  $\in$  sets borel.
    (S = {}  $\longrightarrow$  ( $\exists$   $\alpha$ 1  $\in$  qbs-Mx X. g = ( $\lambda$ r. Inl ( $\alpha$ 1 r))))  $\wedge$ 
    (S = UNIV  $\longrightarrow$  ( $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y. g = ( $\lambda$ r. Inr ( $\alpha$ 2 r))))  $\wedge$ 
    ((S  $\neq$  {}  $\wedge$  S  $\neq$  UNIV)  $\longrightarrow$ 
      ( $\exists$   $\alpha$ 1  $\in$  qbs-Mx X.  $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y.
        g = ( $\lambda$ r::real. (if (r  $\in$  S) then Inl ( $\alpha$ 1 r) else Inr ( $\alpha$ 2 r)))))) }

```

definition *copair-qbs* :: [*'a quasi-borel, 'b quasi-borel*] \Rightarrow (*'a + 'b*) *quasi-borel*
(infixr \oplus_Q 65) **where**
copair-qbs X Y \equiv Abs-quasi-borel (qbs-space X $\lt;+\gt$ qbs-space Y, *copair-qbs-Mx* X Y)

The following is an equivalent definition of *copair-qbs-Mx*.

definition *copair-qbs-Mx2* :: [*'a quasi-borel, 'b quasi-borel*] \Rightarrow (*real* \Rightarrow *'a + 'b*)
set where

```

copair-qbs-Mx2 X Y  $\equiv$ 
  {g. (if qbs-space X = {}  $\wedge$  qbs-space Y = {} then False
    else if qbs-space X  $\neq$  {}  $\wedge$  qbs-space Y = {} then
      ( $\exists$   $\alpha$ 1  $\in$  qbs-Mx X. g = ( $\lambda$ r. Inl ( $\alpha$ 1 r)))
    else if qbs-space X = {}  $\wedge$  qbs-space Y  $\neq$  {} then
      ( $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y. g = ( $\lambda$ r. Inr ( $\alpha$ 2 r)))
    else
      ( $\exists$  S  $\in$  sets borel.  $\exists$   $\alpha$ 1  $\in$  qbs-Mx X.  $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y.
        g = ( $\lambda$ r::real. (if (r  $\in$  S) then Inl ( $\alpha$ 1 r) else Inr ( $\alpha$ 2 r)))) ) }

```

lemma *copair-qbs-Mx-equiv* : *copair-qbs-Mx* (X :: *'a quasi-borel*) (Y :: *'b quasi-borel*)
= *copair-qbs-Mx2* X Y

proof *safe*

```

fix g :: real  $\Rightarrow$  'a + 'b
assume g  $\in$  copair-qbs-Mx X Y
then obtain S where hs:S  $\in$  sets borel  $\wedge$ 
  (S = {}  $\longrightarrow$  ( $\exists$   $\alpha$ 1  $\in$  qbs-Mx X. g = ( $\lambda$ r. Inl ( $\alpha$ 1 r))))  $\wedge$ 
  (S = UNIV  $\longrightarrow$  ( $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y. g = ( $\lambda$ r. Inr ( $\alpha$ 2 r))))  $\wedge$ 
  ((S  $\neq$  {}  $\wedge$  S  $\neq$  UNIV)  $\longrightarrow$ 
    ( $\exists$   $\alpha$ 1  $\in$  qbs-Mx X.
       $\exists$   $\alpha$ 2  $\in$  qbs-Mx Y.
        g = ( $\lambda$ r::real. (if (r  $\in$  S) then Inl ( $\alpha$ 1 r) else Inr ( $\alpha$ 2 r))))))

```

```

    by (auto simp add: copair-qbs-Mx-def)
  consider  $S = \{\}$  |  $S = UNIV$  |  $S \neq \{\}$   $\wedge S \neq UNIV$  by auto
  then show  $g \in \text{copair-qbs-Mx2 } X Y$ 
  proof cases
    assume  $S = \{\}$ 
    from hs this have  $\exists \alpha 1 \in \text{qbs-Mx } X. g = (\lambda r. \text{Inl } (\alpha 1 r))$  by simp
    then obtain  $\alpha 1$  where  $h1: \alpha 1 \in \text{qbs-Mx } X \wedge g = (\lambda r. \text{Inl } (\alpha 1 r))$  by auto
    have  $\text{qbs-space } X \neq \{\}$ 
      using qbs-empty-equiv h1
      by auto
    then have  $(\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\}) \vee (\text{qbs-space } X \neq \{\} \wedge$ 
 $\text{qbs-space } Y \neq \{\})$ 
      by simp
    then show  $g \in \text{copair-qbs-Mx2 } X Y$ 
  proof
    assume  $\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\}$ 
    then show  $g \in \text{copair-qbs-Mx2 } X Y$ 
      by (simp add: copair-qbs-Mx2-def  $\langle \exists \alpha 1 \in \text{qbs-Mx } X. g = (\lambda r. \text{Inl } (\alpha 1 r)) \rangle$ )
  next
    assume  $\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\}$ 
    then obtain  $\alpha 2$  where  $\alpha 2 \in \text{qbs-Mx } Y$  using qbs-empty-equiv by force
    define  $S' :: \text{real set}$ 
      where  $S' \equiv UNIV$ 
    define  $g' :: \text{real} \Rightarrow 'a + 'b$ 
      where  $g' \equiv (\lambda r::\text{real}. (\text{if } (r \in S') \text{ then } \text{Inl } (\alpha 1 r) \text{ else } \text{Inr } (\alpha 2 r)))$ 
    from  $\langle \text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\} \rangle$  h1  $\langle \alpha 2 \in \text{qbs-Mx } Y \rangle$ 
    have  $g' \in \text{copair-qbs-Mx2 } X Y$ 
      by (force simp add: S'-def g'-def copair-qbs-Mx2-def)
    moreover have  $g = g'$ 
      using h1 by (simp add: g'-def S'-def)
    ultimately show ?thesis
      by simp
  qed
next
  assume  $S = UNIV$ 
  from hs this have  $\exists \alpha 2 \in \text{qbs-Mx } Y. g = (\lambda r. \text{Inr } (\alpha 2 r))$  by simp
  then obtain  $\alpha 2$  where  $h2: \alpha 2 \in \text{qbs-Mx } Y \wedge g = (\lambda r. \text{Inr } (\alpha 2 r))$  by auto
  have  $\text{qbs-space } Y \neq \{\}$ 
    using qbs-empty-equiv h2
    by auto
  then have  $(\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\}) \vee (\text{qbs-space } X \neq \{\} \wedge$ 
 $\text{qbs-space } Y \neq \{\})$ 
    by simp
  then show  $g \in \text{copair-qbs-Mx2 } X Y$ 
  proof
    assume  $\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\}$ 
    then show ?thesis
      by (simp add: copair-qbs-Mx2-def  $\langle \exists \alpha 2 \in \text{qbs-Mx } Y. g = (\lambda r. \text{Inr } (\alpha 2 r)) \rangle$ )
  next

```

```

assume  $qbs\text{-}space\ X \neq \{\}$   $\wedge$   $qbs\text{-}space\ Y \neq \{\}$ 
then obtain  $\alpha 1$  where  $\alpha 1 \in qbs\text{-}Mx\ X$  using  $qbs\text{-}empty\text{-}equiv$  by force
define  $S' :: real\ set$ 
  where  $S' \equiv \{\}$ 
define  $g' :: real \Rightarrow 'a + 'b$ 
  where  $g' \equiv (\lambda r :: real. (if\ (r \in S')\ then\ Inl\ (\alpha 1\ r)\ else\ Inr\ (\alpha 2\ r)))$ 
from  $\langle qbs\text{-}space\ X \neq \{\} \wedge qbs\text{-}space\ Y \neq \{\} \rangle h2$   $\langle \alpha 1 \in qbs\text{-}Mx\ X \rangle$ 
have  $g' \in copair\text{-}qbs\text{-}Mx2\ X\ Y$ 
  by( $force\ simp\ add: S'\text{-}def\ g'\text{-}def\ copair\text{-}qbs\text{-}Mx2\text{-}def$ )
moreover have  $g = g'$ 
  using  $h2$  by( $simp\ add: g'\text{-}def\ S'\text{-}def$ )
ultimately show  $?thesis$ 
  by  $simp$ 
qed
next
assume  $S \neq \{\} \wedge S \neq UNIV$ 
then have
   $h: \exists \alpha 1 \in qbs\text{-}Mx\ X.$ 
     $\exists \alpha 2 \in qbs\text{-}Mx\ Y.$ 
       $g = (\lambda r :: real. (if\ (r \in S)\ then\ Inl\ (\alpha 1\ r)\ else\ Inr\ (\alpha 2\ r)))$ 
  using  $hs$  by  $simp$ 
then have  $qbs\text{-}space\ X \neq \{\} \wedge qbs\text{-}space\ Y \neq \{\}$ 
  by ( $metis\ empty\text{-}iff\ qbs\text{-}empty\text{-}equiv$ )
thus  $?thesis$ 
  using  $hs\ h$  by( $auto\ simp\ add: copair\text{-}qbs\text{-}Mx2\text{-}def$ )
qed

next
fix  $g :: real \Rightarrow 'a + 'b$ 
assume  $g \in copair\text{-}qbs\text{-}Mx2\ X\ Y$ 
then have
   $h: if\ qbs\text{-}space\ X = \{\} \wedge qbs\text{-}space\ Y = \{\} then\ False$ 
     $else\ if\ qbs\text{-}space\ X \neq \{\} \wedge qbs\text{-}space\ Y = \{\} then$ 
       $(\exists \alpha 1 \in qbs\text{-}Mx\ X. g = (\lambda r. Inl\ (\alpha 1\ r)))$ 
     $else\ if\ qbs\text{-}space\ X = \{\} \wedge qbs\text{-}space\ Y \neq \{\} then$ 
       $(\exists \alpha 2 \in qbs\text{-}Mx\ Y. g = (\lambda r. Inr\ (\alpha 2\ r)))$ 
     $else$ 
       $(\exists S \in sets\ borel. \exists \alpha 1 \in qbs\text{-}Mx\ X. \exists \alpha 2 \in qbs\text{-}Mx\ Y.$ 
         $g = (\lambda r :: real. (if\ (r \in S)\ then\ Inl\ (\alpha 1\ r)\ else\ Inr\ (\alpha 2\ r))))$ 
  by( $simp\ add: copair\text{-}qbs\text{-}Mx2\text{-}def$ )
consider ( $qbs\text{-}space\ X = \{\} \wedge qbs\text{-}space\ Y = \{\}$ ) |
  ( $qbs\text{-}space\ X \neq \{\} \wedge qbs\text{-}space\ Y = \{\}$ ) |
  ( $qbs\text{-}space\ X = \{\} \wedge qbs\text{-}space\ Y \neq \{\}$ ) |
  ( $qbs\text{-}space\ X \neq \{\} \wedge qbs\text{-}space\ Y \neq \{\}$ ) by  $auto$ 
then show  $g \in copair\text{-}qbs\text{-}Mx\ X\ Y$ 
proof  $cases$ 
  assume  $qbs\text{-}space\ X = \{\} \wedge qbs\text{-}space\ Y = \{\}$ 
  then show  $?thesis$ 

```

```

    using ⟨g ∈ copair-qbs-Mx2 X Y⟩ by(simp add: copair-qbs-Mx2-def)
next
  assume qbs-space X ≠ {} ∧ qbs-space Y = {}
  from h this have ∃α1 ∈ qbs-Mx X. g = (λr. Inl (α1 r)) by simp
  thus ?thesis
    by(auto simp add: copair-qbs-Mx-def)
next
  assume qbs-space X = {} ∧ qbs-space Y ≠ {}
  from h this have ∃α2 ∈ qbs-Mx Y. g = (λr. Inr (α2 r)) by simp
  thus ?thesis
    unfolding copair-qbs-Mx-def
    by(force simp add: copair-qbs-Mx-def)
next
  assume qbs-space X ≠ {} ∧ qbs-space Y ≠ {}
  from h this obtain S α1 α2 where Sag:
    S ∈ sets borel α1 ∈ qbs-Mx X α2 ∈ qbs-Mx Y g = (λr. if r ∈ S then Inl (α1
r) else Inr (α2 r))
    by auto
  consider S = {} | S = UNIV | S ≠ {} S ≠ UNIV by auto
  then show g ∈ copair-qbs-Mx X Y
  proof cases
    assume S = {}
    then have [simp]: (λr. if r ∈ S then Inl (α1 r) else Inr (α2 r)) = (λr. Inr
(α2 r))
      by simp
    show ?thesis
      using ⟨α2 ∈ qbs-Mx Y⟩ unfolding copair-qbs-Mx-def
      by(auto intro! : bexI[where x=UNIV] simp: Sag)
  next
    assume S = UNIV
    then have (λr. if r ∈ S then Inl (α1 r) else Inr (α2 r)) = (λr. Inl (α1 r))
      by simp
    then show ?thesis
      using Sag by(auto simp add: copair-qbs-Mx-def)
  next
    assume S ≠ {} S ≠ UNIV
    then show ?thesis
      using Sag by(auto simp add: copair-qbs-Mx-def)
  qed
qed
qed

lemma
  shows copair-qbs-space: qbs-space (X ⊕Q Y) = qbs-space X <+> qbs-space Y
  (is ?goal1)
  and copair-qbs-Mx: qbs-Mx (X ⊕Q Y) = copair-qbs-Mx X Y (is ?goal2)
proof -
  have copair-qbs-Mx X Y ⊆ UNIV → qbs-space X <+> qbs-space Y
  proof

```

```

fix g
assume g ∈ copair-qbs-Mx X Y
then obtain S where hs:S ∈ sets borel ∧
  (S = {} → (∃ α1 ∈ qbs-Mx X. g = (λr. Inl (α1 r)))) ∧
  (S = UNIV → (∃ α2 ∈ qbs-Mx Y. g = (λr. Inr (α2 r)))) ∧
  ((S ≠ {} ∧ S ≠ UNIV) →
    (∃ α1 ∈ qbs-Mx X.
     ∃ α2 ∈ qbs-Mx Y.
     g = (λr::real. (if (r ∈ S) then Inl (α1 r) else Inr (α2 r)))))
by (auto simp add: copair-qbs-Mx-def)
consider S = {} | S = UNIV | S ≠ {} ∧ S ≠ UNIV by auto
then show g ∈ UNIV → qbs-space X <+> qbs-space Y
proof cases
  assume S = {}
  then show ?thesis
    using hs qbs-Mx-to-X by auto
  next
  assume S = UNIV
  then show ?thesis
    using hs qbs-Mx-to-X by auto
  next
  assume S ≠ {} ∧ S ≠ UNIV
  then have ∃ α1 ∈ qbs-Mx X. ∃ α2 ∈ qbs-Mx Y.
    g = (λr::real. (if (r ∈ S) then Inl (α1 r) else Inr (α2 r))) using hs by
simp
  then show ?thesis
    by(auto dest: qbs-Mx-to-X)
  qed
qed
moreover have qbs-closed1 (copair-qbs-Mx X Y)
proof(rule qbs-closed1I)
  fix g and f :: real ⇒ real
  assume g ∈ copair-qbs-Mx X Y and [measurable]: f ∈ borel →M borel
  then have g ∈ copair-qbs-Mx2 X Y using copair-qbs-Mx-equiv by auto
  consider (qbs-space X = {} ∧ qbs-space Y = {}) |
    (qbs-space X ≠ {} ∧ qbs-space Y = {}) |
    (qbs-space X = {} ∧ qbs-space Y ≠ {}) |
    (qbs-space X ≠ {} ∧ qbs-space Y ≠ {}) by auto
  then have g ∘ f ∈ copair-qbs-Mx2 X Y
  proof cases
    assume qbs-space X = {} ∧ qbs-space Y = {}
    then show ?thesis
      using ⟨g ∈ copair-qbs-Mx2 X Y⟩ qbs-empty-equiv by(simp add: copair-qbs-Mx2-def)
    next
    assume qbs-space X ≠ {} ∧ qbs-space Y = {}
    then obtain α1 where h1:α1 ∈ qbs-Mx X ∧ g = (λr. Inl (α1 r))
      using ⟨g ∈ copair-qbs-Mx2 X Y⟩ by(auto simp add: copair-qbs-Mx2-def)
    then have α1 ∘ f ∈ qbs-Mx X
      by auto

```



```

moreover have  $g \circ f = (\lambda r. \text{Inl } ((\alpha 1 \circ f) r))$ 
  using  $h1$  by auto
ultimately show ?thesis
  using  $\langle \text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\} \rangle$  by(force simp add: copair-qbs-Mx2-def)
next
  assume  $(\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\})$ 
  then obtain  $\alpha 2$  where  $h2: \alpha 2 \in \text{qbs-Mx } Y \wedge g = (\lambda r. \text{Inr } (\alpha 2 r))$ 
  using  $\langle g \in \text{copair-qbs-Mx2 } X Y \rangle$  by(auto simp add: copair-qbs-Mx2-def)
  then have  $\alpha 2 \circ f \in \text{qbs-Mx } Y$ 
  by auto
  moreover have  $g \circ f = (\lambda r. \text{Inr } ((\alpha 2 \circ f) r))$ 
  using  $h2$  by auto
  ultimately show ?thesis
  using  $\langle (\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\}) \rangle$  by(force simp add: copair-qbs-Mx2-def)
next
  assume  $\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\}$ 
  then have  $\exists S \in \text{sets borel}. \exists \alpha 1 \in \text{qbs-Mx } X. \exists \alpha 2 \in \text{qbs-Mx } Y.$ 
   $g = (\lambda r::\text{real}. (\text{if } (r \in S) \text{ then } \text{Inl } (\alpha 1 r) \text{ else } \text{Inr } (\alpha 2 r)))$ 
  using  $\langle g \in \text{copair-qbs-Mx2 } X Y \rangle$  by(simp add: copair-qbs-Mx2-def)
  then show ?thesis
proof safe
  fix  $S \alpha 1 \alpha 2$ 
  assume [measurable]:  $S \in \text{sets borel}$  and  $\alpha 1 \in \text{qbs-Mx } X \alpha 2 \in \text{qbs-Mx } Y$ 
   $g = (\lambda r. \text{if } r \in S \text{ then } \text{Inl } (\alpha 1 r) \text{ else } \text{Inr } (\alpha 2 r))$ 
  have  $f -' S \in \text{sets borel}$ 
  using  $\langle S \in \text{sets borel} \rangle \langle f \in \text{borel-measurable borel} \rangle$  measurable-sets-borel
by blast
  moreover have  $\alpha 1 \circ f \in \text{qbs-Mx } X$ 
  using  $\langle \alpha 1 \in \text{qbs-Mx } X \rangle$  by(auto simp add: qbs-closed1-def)
  moreover have  $\alpha 2 \circ f \in \text{qbs-Mx } Y$ 
  using  $\langle \alpha 2 \in \text{qbs-Mx } Y \rangle$  by(auto simp add: qbs-closed1-def)
  moreover have  $(\lambda r. \text{if } r \in S \text{ then } \text{Inl } (\alpha 1 r) \text{ else } \text{Inr } (\alpha 2 r)) \circ f = (\lambda r. \text{if } r \in f -' S \text{ then } \text{Inl } ((\alpha 1 \circ f) r) \text{ else } \text{Inr } ((\alpha 2 \circ f) r))$ 
  by auto
  ultimately show  $(\lambda r. \text{if } r \in S \text{ then } \text{Inl } (\alpha 1 r) \text{ else } \text{Inr } (\alpha 2 r)) \circ f \in \text{copair-qbs-Mx2 } X Y$ 
  using  $\langle \text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\} \rangle$  by(force simp add: copair-qbs-Mx2-def)
  qed
qed
  thus  $g \circ f \in \text{copair-qbs-Mx } X Y$ 
  using copair-qbs-Mx-equiv by auto
qed
moreover have qbs-closed2  $(\text{qbs-space } X <+> \text{qbs-space } Y)$  (copair-qbs-Mx } X Y)
proof(rule qbs-closed2I)
  fix  $y$ 

```

```

assume  $y \in \text{qbs-space } X \lt + \gt \text{qbs-space } Y$ 
then consider  $y \in \text{Inl } \langle \text{qbs-space } X \rangle \mid y \in \text{Inr } \langle \text{qbs-space } Y \rangle$ 
  by auto
thus  $(\lambda r. y) \in \text{copair-qbs-Mx } X \ Y$ 
proof cases
  case 1
    then obtain  $x$  where  $x: y = \text{Inl } x \ x \in \text{qbs-space } X$ 
      by auto
    define  $\alpha 1 :: \text{real} \Rightarrow -$  where  $\alpha 1 \equiv (\lambda r. x)$ 
    have  $\alpha 1 \in \text{qbs-Mx } X$  using  $\langle x \in \text{qbs-space } X \rangle \text{qbs-decomp}$ 
      by (force simp add: qbs-closed2-def  $\alpha 1$ -def)
    moreover have  $(\lambda r. \text{Inl } x) = (\lambda l. \text{Inl } (\alpha 1 \ l))$  by (simp add:  $\alpha 1$ -def)
    moreover have  $\{\} \in \text{sets borel}$  by auto
    ultimately show  $(\lambda r. y) \in \text{copair-qbs-Mx } X \ Y$ 
      by (auto simp add: copair-qbs-Mx-def x)
  next
    case 2
      then obtain  $x$  where  $x: y = \text{Inr } x \ x \in \text{qbs-space } Y$ 
        by auto
      define  $\alpha 2 :: \text{real} \Rightarrow -$  where  $\alpha 2 \equiv (\lambda r. x)$ 
      have  $\alpha 2 \in \text{qbs-Mx } Y$  using  $\langle x \in \text{qbs-space } Y \rangle \text{qbs-decomp}$ 
        by (force simp add: qbs-closed2-def  $\alpha 2$ -def)
      moreover have  $(\lambda r. \text{Inr } x) = (\lambda l. \text{Inr } (\alpha 2 \ l))$  by (simp add:  $\alpha 2$ -def)
      moreover have  $UNIV \in \text{sets borel}$  by auto
      ultimately show  $(\lambda r. y) \in \text{copair-qbs-Mx } X \ Y$ 
        unfolding copair-qbs-Mx-def
        by (auto intro!: beXI[where x=UNIV] simp: x)
    qed
  qed
moreover have qbs-closed3 (copair-qbs-Mx X Y)
proof (safe intro!: qbs-closed3I)
  fix  $P :: \text{real} \Rightarrow \text{nat}$ 
  fix  $Fi :: \text{nat} \Rightarrow \text{real} \Rightarrow - + -$ 
  assume  $P \in \text{borel} \rightarrow_M \text{count-space } UNIV$ 
     $\forall i. Fi \ i \in \text{copair-qbs-Mx } X \ Y$ 
  then have  $\forall i. Fi \ i \in \text{copair-qbs-Mx2 } X \ Y$  using copair-qbs-Mx-equiv by blast
  consider  $(\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y = \{\}) \mid$ 
     $(\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\}) \mid$ 
     $(\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\}) \mid$ 
     $(\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\})$  by auto
  then have  $(\lambda r. Fi \ (P \ r) \ r) \in \text{copair-qbs-Mx2 } X \ Y$ 
proof cases
  assume  $\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y = \{\}$ 
  then show ?thesis
    using  $\langle \forall i. Fi \ i \in \text{copair-qbs-Mx2 } X \ Y \rangle \text{qbs-empty-equiv}$ 
    by (simp add: copair-qbs-Mx2-def)
  next
  assume  $\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\}$ 
  then have  $\forall i. \exists \alpha i. \alpha i \in \text{qbs-Mx } X \wedge Fi \ i = (\lambda r. \text{Inl } (\alpha i \ r))$ 

```

```

using  $\langle \forall i. Fi i \in \text{copair-qbs-Mx2 } X Y \rangle$  by(auto simp add: copair-qbs-Mx2-def)
then have  $\exists \alpha 1. \forall i. \alpha 1 i \in \text{qbs-Mx } X \wedge Fi i = (\lambda r. \text{Inl } (\alpha 1 i r))$ 
  by(rule choice)
then obtain  $\alpha 1 :: \text{nat} \Rightarrow \text{real} \Rightarrow -$ 
  where  $h1: \forall i. \alpha 1 i \in \text{qbs-Mx } X \wedge Fi i = (\lambda r. \text{Inl } (\alpha 1 i r))$  by auto
define  $\beta :: \text{real} \Rightarrow -$  where  $\beta \equiv (\lambda r. \alpha 1 (P r) r)$ 
from  $\langle P \in \text{borel} \rightarrow_M \text{count-space UNIV} \rangle$   $h1$ 
have  $\beta \in \text{qbs-Mx } X$  by (simp add: beta-def)
moreover have  $(\lambda r. Fi (P r) r) = (\lambda r. \text{Inl } (\beta r))$ 
  using  $h1$  by(simp add: beta-def)
ultimately show ?thesis
  using  $\langle \text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y = \{\} \rangle$  by (auto simp add: co-
pair-qbs-Mx2-def)
next
assume  $\text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\}$ 
then have  $\forall i. \exists \alpha i. \alpha i \in \text{qbs-Mx } Y \wedge Fi i = (\lambda r. \text{Inr } (\alpha i r))$ 
using  $\langle \forall i. Fi i \in \text{copair-qbs-Mx2 } X Y \rangle$  by(auto simp add: copair-qbs-Mx2-def)
then have  $\exists \alpha 2. \forall i. \alpha 2 i \in \text{qbs-Mx } Y \wedge Fi i = (\lambda r. \text{Inr } (\alpha 2 i r))$ 
  by(rule choice)
then obtain  $\alpha 2 :: \text{nat} \Rightarrow \text{real} \Rightarrow -$ 
  where  $h2: \forall i. \alpha 2 i \in \text{qbs-Mx } Y \wedge Fi i = (\lambda r. \text{Inr } (\alpha 2 i r))$  by auto
define  $\beta :: \text{real} \Rightarrow -$  where  $\beta \equiv (\lambda r. \alpha 2 (P r) r)$ 
from  $\langle P \in \text{borel} \rightarrow_M \text{count-space UNIV} \rangle$   $h2$ 
have  $\beta \in \text{qbs-Mx } Y$  by(simp add: beta-def)
moreover have  $(\lambda r. Fi (P r) r) = (\lambda r. \text{Inr } (\beta r))$ 
  using  $h2$  by(simp add: beta-def)
ultimately show ?thesis
  using  $\langle \text{qbs-space } X = \{\} \wedge \text{qbs-space } Y \neq \{\} \rangle$  by (auto simp add: co-
pair-qbs-Mx2-def)
next
assume  $\text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\}$ 
then have  $\forall i. \exists Si. Si \in \text{sets borel} \wedge (\exists \alpha 1 i \in \text{qbs-Mx } X. \exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in Si) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 
using  $\langle \forall i. Fi i \in \text{copair-qbs-Mx2 } X Y \rangle$  by (auto simp add: copair-qbs-Mx2-def)
then have  $\exists S. \forall i. S i \in \text{sets borel} \wedge (\exists \alpha 1 i \in \text{qbs-Mx } X. \exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 
  by(rule choice)
then obtain  $S :: \text{nat} \Rightarrow \text{real set}$ 
  where  $hs : \forall i. S i \in \text{sets borel} \wedge (\exists \alpha 1 i \in \text{qbs-Mx } X. \exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 
  by auto
then have  $\forall i. \exists \alpha 1 i. \alpha 1 i \in \text{qbs-Mx } X \wedge (\exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 
  by blast
then have  $\exists \alpha 1. \forall i. \alpha 1 i \in \text{qbs-Mx } X \wedge (\exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 
  by(rule choice)
then obtain  $\alpha 1$  where  $h1: \forall i. \alpha 1 i \in \text{qbs-Mx } X \wedge (\exists \alpha 2 i \in \text{qbs-Mx } Y.$ 
   $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then Inl } (\alpha 1 i r) \text{ else Inr } (\alpha 2 i r))))$ 

```

```

    by auto
  define  $\beta 1 :: \text{real} \Rightarrow -$  where  $\beta 1 \equiv (\lambda r. \alpha 1 (P r) r)$ 
  from  $\langle P \in \text{borel} \rightarrow_M \text{count-space UNIV} \rangle h1$ 
  have  $\beta 1 \in \text{qbs-Mx } X$  by(simp add:  $\beta 1$ -def)
  from  $h1$  have  $\forall i. \exists \alpha 2 i. \alpha 2 i \in \text{qbs-Mx } Y \wedge$ 
     $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then } \text{Inl } (\alpha 1 i r) \text{ else } \text{Inr } (\alpha 2 i r)))$ 
  by auto
  then have  $\exists \alpha 2. \forall i. \alpha 2 i \in \text{qbs-Mx } Y \wedge$ 
     $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then } \text{Inl } (\alpha 1 i r) \text{ else } \text{Inr } (\alpha 2 i r)))$ 
  by(rule choice)
  then obtain  $\alpha 2$ 
  where  $h2: \forall i. \alpha 2 i \in \text{qbs-Mx } Y \wedge$ 
     $Fi i = (\lambda r :: \text{real}. (\text{if } (r \in S i) \text{ then } \text{Inl } (\alpha 1 i r) \text{ else } \text{Inr } (\alpha 2 i r)))$ 
  by auto
  define  $\beta 2 :: \text{real} \Rightarrow -$  where  $\beta 2 \equiv (\lambda r. \alpha 2 (P r) r)$ 
  from  $\langle P \in \text{borel} \rightarrow_M \text{count-space UNIV} \rangle h2$ 
  have  $\beta 2 \in \text{qbs-Mx } Y$  by(simp add:  $\beta 2$ -def)
  define  $A :: \text{nat} \Rightarrow \text{real set}$  where  $A \equiv (\lambda i. S i \cap P - \{i\})$ 
  have [measurable]:  $\bigwedge i. A i \in \text{sets borel}$ 
  using  $A$ -def  $hs$  measurable-separate[ $OF \langle P \in \text{borel} \rightarrow_M \text{count-space UNIV} \rangle$ ]
by blast
  define  $S' :: \text{real set}$  where  $S' \equiv \{r. r \in S (P r)\}$ 
  have  $S' = (\bigcup i :: \text{nat}. A i)$ 
  by(auto simp add:  $S'$ -def  $A$ -def)
  hence  $S' \in \text{sets borel}$  by auto
  from  $h2$  have  $(\lambda r. Fi (P r) r) = (\lambda r. (\text{if } r \in S' \text{ then } \text{Inl } (\beta 1 r)$ 
     $\text{else } \text{Inr } (\beta 2 r)))$ 
  by(auto simp add:  $\beta 1$ -def  $\beta 2$ -def  $S'$ -def)
  thus  $(\lambda r. Fi (P r) r) \in \text{copair-qbs-Mx2 } X Y$ 
  using  $\langle \text{qbs-space } X \neq \{\} \wedge \text{qbs-space } Y \neq \{\} \rangle \langle S' \in \text{sets borel} \rangle \langle \beta 1 \in \text{qbs-Mx } X \rangle$ 
   $\langle \beta 2 \in \text{qbs-Mx } Y \rangle$ 
  by(auto simp add: copair-qbs-Mx2-def)
qed
  thus  $(\lambda r. Fi (P r) r) \in \text{copair-qbs-Mx } X Y$ 
  using copair-qbs-Mx-equiv by auto
qed
  ultimately have  $\text{Rep-quasi-borel } (\text{copair-qbs } X Y) = (\text{qbs-space } X <+> \text{qbs-space } Y, \text{copair-qbs-Mx } X Y)$ 
  unfolding copair-qbs-def by(auto intro!: Abs-quasi-borel-inverse)
  thus ?goal1 ?goal2
  by(simp-all add: qbs-space-def qbs-Mx-def)
qed

```

lemma copair-qbs-MxD:

```

  assumes  $g \in \text{qbs-Mx } (X \oplus_Q Y)$ 
  and  $\bigwedge \alpha. \alpha \in \text{qbs-Mx } X \implies g = (\lambda r. \text{Inl } (\alpha r)) \implies P g$ 
  and  $\bigwedge \beta. \beta \in \text{qbs-Mx } Y \implies g = (\lambda r. \text{Inr } (\beta r)) \implies P g$ 
  and  $\bigwedge S \alpha \beta. (S :: \text{real set}) \in \text{sets borel} \implies S \neq \{\} \implies S \neq \text{UNIV} \implies \alpha \in \text{qbs-Mx } X \implies \beta \in \text{qbs-Mx } Y \implies g = (\lambda r. \text{if } r \in S \text{ then } \text{Inl } (\alpha r) \text{ else } \text{Inr } (\beta r))$ 

```

$r)) \implies P g$
shows $P g$
using *assms* **by** (*fastforce simp: copair-qbs-Mx copair-qbs-Mx-def*)

3.1.8 Product Spaces

definition $PiQ :: 'a \text{ set} \Rightarrow ('a \Rightarrow 'b \text{ quasi-borel}) \Rightarrow ('a \Rightarrow 'b) \text{ quasi-borel}$ **where**
 $PiQ I X \equiv Abs\text{-quasi-borel} (\Pi_E i \in I. \text{qbs-space } (X i), \{\alpha. \forall i. (i \in I \longrightarrow (\lambda r. \alpha r i) \in \text{qbs-Mx } (X i)) \wedge (i \notin I \longrightarrow (\lambda r. \alpha r i) = (\lambda r. \text{undefined}))\})$

syntax

$-PiQ :: p\text{trn} \Rightarrow 'i \text{ set} \Rightarrow 'a \text{ quasi-borel} \Rightarrow ('i \Rightarrow 'a) \text{ quasi-borel}$ ($(\exists \Pi_Q -\in-./ -)$
 $10)$

translations

$\Pi_Q x \in I. X == CONST PiQ I (\lambda x. X)$

lemma

shows $PiQ\text{-space: qbs-space } (PiQ I X) = (\Pi_E i \in I. \text{qbs-space } (X i))$ (**is** *?goal1*)
and $PiQ\text{-Mx: qbs-Mx } (PiQ I X) = \{\alpha. \forall i. (i \in I \longrightarrow (\lambda r. \alpha r i) \in \text{qbs-Mx } (X i)) \wedge (i \notin I \longrightarrow (\lambda r. \alpha r i) = (\lambda r. \text{undefined}))\}$ (**is** $- = ?Mx$)

proof –

have $?Mx \subseteq UNIV \rightarrow (\Pi_E i \in I. \text{qbs-space } (X i))$

using $\text{qbs-Mx-to-X}[of - X -]$ **by** *auto metis*

moreover **have** $\text{qbs-closed1 } ?Mx$

proof (*safe intro!: qbs-closed1I*)

fix αi **and** $f :: \text{real} \Rightarrow \text{real}$

assume $h[\text{measurable}]: \forall i. (i \in I \longrightarrow (\lambda r. \alpha r i) \in \text{qbs-Mx } (X i)) \wedge (i \notin I \longrightarrow (\lambda r. \alpha r i) = (\lambda r. \text{undefined}))$

$f \in \text{borel} \rightarrow_M \text{borel}$

show $(\lambda r. (\alpha \circ f) r i) \in \text{qbs-Mx } (X i)$ **if** $i: i \in I$

proof –

have $(\lambda r. \alpha r i) \circ f \in \text{qbs-Mx } (X i)$

using $h i$ **by** *auto*

thus $(\lambda r. (\alpha \circ f) r i) \in \text{qbs-Mx } (X i)$

by (*simp add: comp-def*)

qed

show $i \notin I \implies (\lambda r. (\alpha \circ f) r i) = (\lambda r. \text{undefined})$

by (*metis comp-apply h(1)*)

qed

moreover **have** $\text{qbs-closed2 } (\Pi_E i \in I. \text{qbs-space } (X i))$ $?Mx$

by (*rule qbs-closed2I*) (*auto simp: PiE-def extensional-def Pi-def*)

moreover **have** $\text{qbs-closed3 } ?Mx$

proof (*rule qbs-closed3I*)

fix $P :: \text{real} \Rightarrow \text{nat}$ **and** Fi

assume $h: P \in \text{borel} \rightarrow_M \text{count-space } UNIV$

$\bigwedge i::\text{nat}. Fi i \in ?Mx$

show $(\lambda r. Fi (P r) r) \in ?Mx$

proof *safe*

fix i

```

assume  $hi:i \in I$ 
then show  $(\lambda r. Fi (P r) r i) \in qbs-Mx (X i)$ 
  using  $h$  qbs-closed3-dest[OF h(1),of  $\lambda j r. Fi j r i$ ]
  by auto
next
show  $\bigwedge i. i \notin I \implies (\lambda r. Fi (P r) r i) = (\lambda r. undefined)$ 
  using  $h$  by auto meson
qed
qed
ultimately have  $Rep-quasi-borel (PiQ I X) = (\Pi_{E i \in I. qbs-space (X i), ?Mx}$ 
  by(auto intro!: Abs-quasi-borel-inverse is-quasi-borel-intro simp: PiQ-def)
thus ?goal1  $qbs-Mx (PiQ I X) = ?Mx$ 
  by(simp-all add: qbs-space-def qbs-Mx-def)
qed

```

lemma *prod-qbs-MxI*:

```

assumes  $\bigwedge i. i \in I \implies (\lambda r. \alpha r i) \in qbs-Mx (X i)$ 
  and  $\bigwedge i. i \notin I \implies (\lambda r. \alpha r i) = (\lambda r. undefined)$ 
shows  $\alpha \in qbs-Mx (PiQ I X)$ 
using assms by(auto simp: PiQ-Mx)

```

lemma *prod-qbs-MxD*:

```

assumes  $\alpha \in qbs-Mx (PiQ I X)$ 
shows  $\bigwedge i. i \in I \implies (\lambda r. \alpha r i) \in qbs-Mx (X i)$ 
  and  $\bigwedge i. i \notin I \implies (\lambda r. \alpha r i) = (\lambda r. undefined)$ 
  and  $\bigwedge i r. i \notin I \implies \alpha r i = undefined$ 
using assms by(auto simp: PiQ-Mx dest: fun-cong[where  $g=(\lambda r. undefined)$ ])

```

lemma *PiQ-eqI*:

```

assumes  $\bigwedge i. i \in I \implies X i = Y i$ 
shows  $PiQ I X = PiQ I Y$ 
by(auto intro!: qbs-eqI simp: PiQ-Mx assms)

```

lemma *PiQ-empty: qbs-space (PiQ {} X) = { $\lambda i. undefined$ }*

```

by(auto simp: PiQ-space)

```

lemma *PiQ-empty-Mx: qbs-Mx (PiQ {} X) = { $\lambda r i. undefined$ }*

```

by(auto simp: PiQ-Mx meson)

```

3.1.9 Coproduct Spaces

definition *coPiQ-Mx* :: [*'a set, 'a \implies 'b quasi-borel*] \implies (*real \implies 'a \times 'b*) **set** **where**
coPiQ-Mx I X \equiv { $\lambda r. (f r, \alpha (f r) r) \mid f \alpha. f \in borel \rightarrow_M count-space I \wedge (\forall i \in range f. \alpha i \in qbs-Mx (X i))$ }

definition *coPiQ-Mx'* :: [*'a set, 'a \implies 'b quasi-borel*] \implies (*real \implies 'a \times 'b*) **set** **where**
coPiQ-Mx' I X \equiv { $\lambda r. (f r, \alpha (f r) r) \mid f \alpha. f \in borel \rightarrow_M count-space I \wedge (\forall i. (i \in range f \vee qbs-space (X i) \neq \{\}) \longrightarrow \alpha i \in qbs-Mx (X i))$ }

lemma *coPiQ-Mx-eq*:
 $coPiQ\text{-}Mx\ I\ X = coPiQ\text{-}Mx'\ I\ X$

proof *safe*
fix α
assume $\alpha \in coPiQ\text{-}Mx\ I\ X$
then obtain $f\ \beta$ **where** *hfb*:
 $f \in borel \rightarrow_M count\text{-}space\ I \wedge i. i \in range\ f \implies \beta\ i \in qbs\text{-}Mx\ (X\ i)\ \alpha = (\lambda r. (f\ r, \beta\ (f\ r)\ r))$

unfolding *coPiQ-Mx-def* **by** *blast*
define β' **where** $\beta' \equiv (\lambda i. \text{if } i \in range\ f \text{ then } \beta\ i$
 $\text{else if } qbs\text{-}space\ (X\ i) \neq \{\}\ \text{then } (SOME\ \gamma. \gamma \in qbs\text{-}Mx$
 $(X\ i))$
 $\text{else } \beta\ i)$

have $1:\alpha = (\lambda r. (f\ r, \beta'\ (f\ r)\ r))$
by (*simp add: hfb(3) β' -def*)

have $2:\wedge i. qbs\text{-}space\ (X\ i) \neq \{\} \implies \beta'\ i \in qbs\text{-}Mx\ (X\ i)$

proof $-$
fix i
assume $hne:qbs\text{-}space\ (X\ i) \neq \{\}$
then obtain x **where** $x \in qbs\text{-}space\ (X\ i)$ **by** *auto*
hence $(\lambda r. x) \in qbs\text{-}Mx\ (X\ i)$ **by** *auto*
thus $\beta'\ i \in qbs\text{-}Mx\ (X\ i)$
by (*cases* $i \in range\ f$) (*auto simp: β' -def hfb(2) hne intro!: someI2[where*
 $a=\lambda r. x]$)

qed
show $\alpha \in coPiQ\text{-}Mx'\ I\ X$
using *hfb(1,2) 1 2 β' -def* **by** (*auto simp: coPiQ-Mx'-def intro!: exI[where $x=f$]*
 $exI[where $x=\beta'$])$

next
fix α
assume $\alpha \in coPiQ\text{-}Mx'\ I\ X$
then obtain $f\ \beta$ **where** *hfb*:
 $f \in borel \rightarrow_M count\text{-}space\ I \wedge i. qbs\text{-}space\ (X\ i) \neq \{\} \implies \beta\ i \in qbs\text{-}Mx\ (X\ i)$
 $\wedge i. i \in range\ f \implies \beta\ i \in qbs\text{-}Mx\ (X\ i)\ \alpha = (\lambda r. (f\ r, \beta\ (f\ r)\ r))$

unfolding *coPiQ-Mx'-def* **by** *blast*
show $\alpha \in coPiQ\text{-}Mx\ I\ X$
by (*auto simp: hfb(4) coPiQ-Mx-def intro!: hfb(1) hfb(3)*)

qed

definition *coPiQ* :: [*'a set, 'a \Rightarrow 'b quasi-borel*] \Rightarrow (*'a \times 'b quasi-borel* **where**
 $coPiQ\ I\ X \equiv Abs\text{-}quasi\text{-}borel\ (SIGMA\ i:I. qbs\text{-}space\ (X\ i), coPiQ\text{-}Mx\ I\ X)$

syntax
 $-coPiQ :: pptrn \Rightarrow 'i\ set \Rightarrow 'a\ quasi\text{-}borel \Rightarrow ('i \times 'a)\ quasi\text{-}borel\ ((\exists \Pi_Q -\in - / -)$
 $10)$

translations
 $\Pi_Q\ x \in I. X \Rightarrow CONST\ coPiQ\ I\ (\lambda x. X)$

lemma

shows $\text{coPiQ-space: qbs-space (coPiQ I X) = (SIGMA } i:I. \text{ qbs-space (X } i))$ (is ?goal1)
and $\text{coPiQ-Mx: qbs-Mx (coPiQ I X) = coPiQ-Mx I X}$ (is ?goal2)
proof –
have $\text{coPiQ-Mx I X} \subseteq \text{UNIV} \rightarrow (\text{SIGMA } i:I. \text{ qbs-space (X } i))$
by(fastforce simp: coPiQ-Mx-def dest: measurable-space qbs-Mx-to-X)
moreover have qbs-closed1 (coPiQ-Mx I X)
proof(rule qbs-closed1I)
fix α **and** $f :: \text{real} \Rightarrow \text{real}$
assume $\alpha \in \text{coPiQ-Mx I X}$
and $1[\text{measurable}]: f \in \text{borel} \rightarrow_M \text{borel}$
then obtain β g **where** ha:
 $\bigwedge i. i \in \text{range } g \implies \beta \ i \in \text{qbs-Mx (X } i)$ $\alpha = (\lambda r. (g \ r, \beta \ (g \ r) \ r))$ **and**
 $[\text{measurable}]: g \in \text{borel} \rightarrow_M \text{count-space } I$
by(fastforce simp: coPiQ-Mx-def)
then have $\bigwedge i. i \in \text{range } g \implies \beta \ i \circ f \in \text{qbs-Mx (X } i)$
by simp
thus $\alpha \circ f \in \text{coPiQ-Mx I X}$
unfolding coPiQ-Mx-def **by** (auto intro!: exI[where $x=g \circ f$] exI[where $x=\lambda i. \beta \ i \circ f$] simp: ha(2))
qed
moreover have qbs-closed2 (SIGMA $i:I. \text{ qbs-space (X } i)$) (coPiQ-Mx I X)
proof(safe intro!: qbs-closed2I)
fix $i \ x$
assume $i \in I$ $x \in \text{qbs-space (X } i)$
then show $(\lambda r. (i, x)) \in \text{coPiQ-Mx I X}$
by(auto simp: coPiQ-Mx-def intro!: exI[where $x=\lambda r. i$])
qed
moreover have qbs-closed3 (coPiQ-Mx I X)
proof(rule qbs-closed3I)
fix $P :: \text{real} \Rightarrow \text{nat}$ **and** Fi
assume $h[\text{measurable}]: P \in \text{borel} \rightarrow_M \text{count-space UNIV}$
 $\bigwedge i :: \text{nat}. Fi \ i \in \text{coPiQ-Mx I X}$
then have $\forall i. \exists fi \ \alpha i. Fi \ i = (\lambda r. (fi \ r, \alpha i \ (fi \ r) \ r)) \wedge fi \ i \in \text{borel} \rightarrow_M \text{count-space}$
 $I \wedge (\forall j. (j \in \text{range } fi \vee \text{qbs-space (X } j) \neq \{\}) \longrightarrow \alpha i \ j \in \text{qbs-Mx (X } j))$
by(auto simp: coPiQ-Mx-eq coPiQ-Mx'-def)
then obtain fi **where**
 $\forall i. \exists \alpha i. Fi \ i = (\lambda r. (fi \ i \ r, \alpha i \ (fi \ i \ r) \ r)) \wedge fi \ i \in \text{borel} \rightarrow_M \text{count-space } I \wedge$
 $(\forall j. (j \in \text{range } (fi \ i) \vee \text{qbs-space (X } j) \neq \{\}) \longrightarrow \alpha i \ j \in \text{qbs-Mx (X } j))$
by(fastforce intro!: choice)
then obtain αi **where**
 $\forall i. Fi \ i = (\lambda r. (fi \ i \ r, \alpha i \ i \ (fi \ i \ r) \ r)) \wedge fi \ i \in \text{borel} \rightarrow_M \text{count-space } I \wedge (\forall j.$
 $(j \in \text{range } (fi \ i) \vee \text{qbs-space (X } j) \neq \{\}) \longrightarrow \alpha i \ i \ j \in \text{qbs-Mx (X } j))$
by(fastforce intro!: choice)
then have $hf[\text{measurable}]:$
 $\bigwedge i. Fi \ i = (\lambda r. (fi \ i \ r, \alpha i \ i \ (fi \ i \ r) \ r)) \wedge i. fi \ i \in \text{borel} \rightarrow_M \text{count-space } I \wedge i$
 $j. j \in \text{range } (fi \ i) \implies \alpha i \ i \ j \in \text{qbs-Mx (X } j) \wedge i \ j. \text{qbs-space (X } j) \neq \{\} \implies \alpha i \ i \ j$
 $\in \text{qbs-Mx (X } j)$
by auto


```

define  $f'$  where  $f' \equiv (\lambda r. fi (P r) r)$ 
define  $\alpha'$  where  $\alpha' \equiv (\lambda i r. \alpha i (P r) i r)$ 
have  $1: (\lambda r. Fi (P r) r) = (\lambda r. (f' r, \alpha' (f' r) r))$ 
  by(simp add:  $\alpha'$ -def  $f'$ -def hf)
have  $f' \in \text{borel} \rightarrow_M \text{count-space } I$ 
  by(simp add:  $f'$ -def)
moreover have  $\bigwedge i. i \in \text{range } f' \implies \alpha' i \in \text{qbs-Mx } (X i)$ 
proof –
  fix  $i$ 
  assume  $hi: i \in \text{range } f'$ 
  then obtain  $r$  where  $hr: i = fi (P r) r$  by(auto simp:  $f'$ -def)
  hence  $i \in \text{range } (fi (P r))$  by simp
  hence  $\alpha i (P r) i \in \text{qbs-Mx } (X i)$  by(simp add: hf)
  hence  $\text{qbs-space } (X i) \neq \{\}$ 
    by(auto simp: qbs-empty-equiv)
  hence  $\bigwedge j. \alpha i j i \in \text{qbs-Mx } (X i)$ 
    by(simp add: hf(4))
  then show  $\alpha' i \in \text{qbs-Mx } (X i)$ 
    by(auto simp:  $\alpha'$ -def h(1) intro!: qbs-closed3-dest[of P  $\lambda j. \alpha i j i$ ])
qed
ultimately show  $(\lambda r. Fi (P r) r) \in \text{coPiQ-Mx } I X$ 
  by(auto simp: 1 coPiQ-Mx-def intro!: exI[where  $x=f'$ ])
qed
ultimately have  $\text{Rep-quasi-borel } (\text{coPiQ } I X) = (\text{SIGMA } i:I. \text{qbs-space } (X i),$ 
 $\text{coPiQ-Mx } I X)$ 
  unfolding coPiQ-def by(fastforce intro!: Abs-quasi-borel-inverse)
thus  $?goal1 ?goal2$ 
  by(simp-all add: qbs-space-def qbs-Mx-def)
qed

```

lemma *coPiQ-MxI*:

```

assumes  $f \in \text{borel} \rightarrow_M \text{count-space } I$ 
  and  $\bigwedge i. i \in \text{range } f \implies \alpha i \in \text{qbs-Mx } (X i)$ 
  shows  $(\lambda r. (f r, \alpha (f r) r)) \in \text{qbs-Mx } (\text{coPiQ } I X)$ 
  using assms unfolding coPiQ-Mx-def coPiQ-Mx by blast

```

lemma *coPiQ-eqI*:

```

assumes  $\bigwedge i. i \in I \implies X i = Y i$ 
shows  $\text{coPiQ } I X = \text{coPiQ } I Y$ 
using assms by(auto intro!: qbs-eqI simp: coPiQ-Mx coPiQ-Mx-def) (metis UNIV-I
measurable-space space-borel space-count-space)+

```

3.1.10 List Spaces

We define the quasi-Borel spaces on list using the following isomorphism.

$$\text{List}(X) \cong \prod_{n \in \mathbb{N}} \prod_{0 \leq i < n} X$$

definition *list-nil* :: $\text{nat} \times (\text{nat} \Rightarrow 'a)$ **where**
list-nil $\equiv (0, \lambda n. \text{undefined})$
definition *list-cons* :: $['a, \text{nat} \times (\text{nat} \Rightarrow 'a)] \Rightarrow \text{nat} \times (\text{nat} \Rightarrow 'a)$ **where**
list-cons $x\ l \equiv (\text{Suc} (\text{fst } l), (\lambda n. \text{if } n = 0 \text{ then } x \text{ else } (\text{snd } l) (n - 1)))$

fun *from-list* :: $'a \text{ list} \Rightarrow \text{nat} \times (\text{nat} \Rightarrow 'a)$ **where**
from-list [] = *list-nil* |
from-list (a#l) = *list-cons* a (*from-list* l)

fun *to-list'* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow 'a) \Rightarrow 'a \text{ list}$ **where**
to-list' 0 = [] |
to-list' (Suc n) f = f 0 # *to-list'* n ($\lambda n. f (\text{Suc } n)$)

definition *to-list* :: $\text{nat} \times (\text{nat} \Rightarrow 'a) \Rightarrow 'a \text{ list}$ **where**
to-list $\equiv \text{case-prod } \text{to-list}'$

lemma *inj-on-to-list*: *inj-on* (*to-list* :: $\text{nat} \times (\text{nat} \Rightarrow 'a) \Rightarrow 'a \text{ list}$) (*SIGMA*
n:*UNIV*. *PiE* {..*n*} *A*)

proof(*safe intro!*: *inj-onI*)
fix *n m* **and** *x y* :: $\text{nat} \Rightarrow 'a$
assume *h*:*to-list* (*n*, *x*) = *to-list* (*m*, *y*)
have 1: $\bigwedge a. \text{length } (\text{to-list } (n, a)) = n$ **for** *n*
by(*induction n*) (*auto simp: to-list-def*)
show *n*:*n* = *m*
using 1 *arg-cong*[*OF h*,*of length*] **by** *metis*
show $x \in \text{PiE } \{..*n\} A \Longrightarrow y \in \text{PiE } \{..*m\} A \Longrightarrow x = y**$
using *h* **unfolding** *n*
proof(*induction m* *arbitrary: x y A*)
case *ih*:(*Suc m*)
then have *to-list* (*m*, ($\lambda n. x (\text{Suc } n)$)) = *to-list* (*m*, ($\lambda n. y (\text{Suc } n)$))
by(*auto simp: to-list-def*)
hence 1:($\lambda n. x (\text{Suc } n) = \lambda n. y (\text{Suc } n)$)
using *ih*(2-4) **by**(*intro ih*(1)[*of -* $\lambda n. A (\text{Suc } n)$]) *auto*
show ?*case*
proof
fix *n*
show $x\ n = y\ n$
proof(*cases n*)
assume $n = 0$
then show $x\ n = y\ n$
using *ih*(4) **by**(*auto simp: to-list-def*)
qed(*use fun-cong*[*OF 1*] **in** *auto*)
qed
qed(*auto simp: to-list-def*)
qed

Definition

definition *list-qbs* :: $'a \text{ quasi-borel} \Rightarrow 'a \text{ list quasi-borel}$ **where**
list-qbs *X* $\equiv \text{map-qbs } \text{to-list} (\text{II}_Q\ n \in (\text{UNIV} :: \text{nat set}). \text{II}_Q\ i \in \{..*n\}). X)*$

definition *list-head* :: $\text{nat} \times (\text{nat} \Rightarrow 'a) \Rightarrow 'a$ **where**
list-head $l = \text{snd } l \ 0$

definition *list-tail* :: $\text{nat} \times (\text{nat} \Rightarrow 'a) \Rightarrow \text{nat} \times (\text{nat} \Rightarrow 'a)$ **where**
list-tail $l = (\text{fst } l - 1, \lambda m. (\text{snd } l) (\text{Suc } m))$

lemma *list-simp1*: $\text{list-nil} \neq \text{list-cons } x \ l$
by (*simp add: list-nil-def list-cons-def*)

lemma *list-simp2*:
assumes $\text{list-cons } a \ al = \text{list-cons } b \ bl$
shows $a = b \ al = bl$
proof –
have $a = \text{snd } (\text{list-cons } a \ al) \ 0 \ b = \text{snd } (\text{list-cons } b \ bl) \ 0$
by (*auto simp: list-cons-def*)
thus $a = b$
by(*simp add: assms*)

next
have $\text{fst } al = \text{fst } bl$
using *assms* **by** (*simp add: list-cons-def*)
moreover have $\text{snd } al = \text{snd } bl$
proof
fix n
have $\text{snd } al \ n = \text{snd } (\text{list-cons } a \ al) (\text{Suc } n)$
by (*simp add: list-cons-def*)
also have $\dots = \text{snd } (\text{list-cons } b \ bl) (\text{Suc } n)$
by (*simp add: assms*)
also have $\dots = \text{snd } bl \ n$
by (*simp add: list-cons-def*)
finally show $\text{snd } al \ n = \text{snd } bl \ n$.

qed
ultimately show $al = bl$
by (*simp add: prod.expand*)

qed

lemma
shows *list-simp3*: $\text{list-head } (\text{list-cons } a \ l) = a$
and *list-simp4*: $\text{list-tail } (\text{list-cons } a \ l) = l$
by(*simp-all add: list-head-def list-cons-def list-tail-def*)

lemma *list-decomp1*:
assumes $l \in \text{qbs-space } (\Pi_Q \ n \in (\text{UNIV} :: \text{nat set}). \Pi_Q \ i \in \{..<n\}. X)$
shows $l = \text{list-nil} \vee$
 $(\exists a \ l'. a \in \text{qbs-space } X \wedge l' \in \text{qbs-space } (\Pi_Q \ n \in (\text{UNIV} :: \text{nat set}). \Pi_Q$
 $i \in \{..<n\}. X) \wedge l = \text{list-cons } a \ l')$
proof(*cases l*)
case *hl*:(*Pair n f*)
show *?thesis*
proof(*cases n*)

```

case 0
then show ?thesis
  using assms hl by (simp add: list-nil-def coPiQ-space PiQ-space)
next
case hn:(Suc n')
define f' where f' ≡ λm. f (Suc m)
have l = list-cons (f 0) (n',f')
  unfolding hl hn list-cons-def
proof safe
  fix m
  show f = (λm. if m = 0 then f 0 else snd (n', f') (m - 1))
  proof
    fix m
    show f m = (if m = 0 then f 0 else snd (n', f') (m - 1))
      using assms hl by(cases m; fastforce simp: f'-def)
    qed
  qed simp
  moreover have (n', f') ∈ qbs-space (ΠQ n∈(UNIV :: nat set).ΠQ i∈{..n}).
X)
proof –
  have  $\bigwedge x. x \in \{..n'\} \implies f' x \in qbs\text{-space } X$ 
    using assms hl hn by(fastforce simp: f'-def coPiQ-space PiQ-space)
  moreover {
    fix x
    assume 1:x ∉ {..n'}
    hence f' x = undefined
    using hl assms hn by(auto simp: f'-def coPiQ-space PiQ-space)
  }
  ultimately show ?thesis
    by(auto simp add: coPiQ-space PiQ-space)
  qed
  ultimately show ?thesis
    using hl assms by(auto intro!: exI[where x=f 0] exI[where x=(n',λm. if m = 0 then undefined else f (Suc m))] simp: list-cons-def coPiQ-space PiQ-space)
  qed
qed

lemma list-simp5:
  assumes l ∈ qbs-space (ΠQ n∈(UNIV :: nat set).ΠQ i∈{..n}). X
  and l ≠ list-nil
  shows l = list-cons (list-head l) (list-tail l)
proof –
  obtain a l' where hl:
    a ∈ qbs-space X l' ∈ qbs-space (ΠQ n∈(UNIV :: nat set).ΠQ i∈{..n}). X l = list-cons a l'
  using list-decomp1[OF assms(1)] assms(2) by blast
  hence list-head l = a list-tail l = l'
  by(simp-all add: list-simp3 list-simp4)
  thus ?thesis

```

using $hl(3)$ *list-simp2* by *auto*
qed

lemma *list-simp6*:

$list-nil \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X)$
by (*simp add: list-nil-def coPiQ-space PiQ-space*)

lemma *list-simp7*:

assumes $a \in qbs-space X$
and $l \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X)$
shows $list-cons a l \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X)$
using *assms* **by** (*fastforce simp: PiE-def extensional-def list-cons-def coPiQ-space PiQ-space*)

lemma *list-destruct-rule*:

assumes $l \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X)$
 $P list-nil$
and $\bigwedge a l'. a \in qbs-space X \implies l' \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X) \implies P (list-cons a l')$
shows $P l$
by (*rule disjE[OF list-decomp1[OF assms(1)]]*) (*use assms in auto*)

lemma *list-induct-rule*:

assumes $l \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X)$
 $P list-nil$
and $\bigwedge a l'. a \in qbs-space X \implies l' \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X) \implies P l' \implies P (list-cons a l')$
shows $P l$
proof (*cases l*)
case *hl*: (*Pair n f*)
then show *?thesis*
using *assms(1)*
proof (*induction n arbitrary: f l*)
case *0*
then show *?case*
using *assms(2)* **by** (*simp add: coPiQ-space PiQ-space list-nil-def*)
next
case *ih*: (*Suc n*)
then obtain $a l'$ **where** *hl*:
 $a \in qbs-space X \ l' \in qbs-space (\prod_Q n \in (UNIV :: nat set). \prod_Q i \in \{..<n\}. X) \ l = list-cons a l'$
using *list-decomp1* **by** (*simp add: list-nil-def*) *blast*
have $P l'$
using *ih hl(3)*
by (*auto intro!: ih(1)[OF - hl(2), of snd l'] simp: coPiQ-space PiQ-space list-cons-def*)
from *assms(3)[OF hl(1,2) this]*
show *?case*
by (*simp add: hl(3)*)

qed
qed

lemma *to-list-simp1*: *to-list list-nil* = []
by(*simp add: to-list-def list-nil-def*)

lemma *to-list-simp2*:
assumes $l \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
shows *to-list (list-cons a l) = a # to-list l*
using *assms* **by**(*auto simp: PiE-def to-list-def list-cons-def coPiQ-space PiQ-space*)

lemma *to-list-set*:
assumes $l \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
shows *set (to-list l) \subseteq qbs-space X*
by(*rule list-induct-rule[OF assms]*) (*auto simp: to-list-simp1 to-list-simp2*)

lemma *from-list-length*: *fst (from-list l) = length l*
by(*induction l, simp-all add: list-cons-def list-nil-def*)

lemma *from-list-in-list-of*:
assumes *set l \subseteq qbs-space X*
shows *from-list l \in qbs-space $(\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$*
using *assms* **by**(*induction l*) (*auto simp: PiE-def extensional-def Pi-def coPiQ-space PiQ-space list-nil-def list-cons-def*)

lemma *from-list-in-list-of'*: *from-list l \in qbs-space $(\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. \text{Abs-quasi-borel } (\text{UNIV}, \text{UNIV}))$*

proof –
have *set l \subseteq qbs-space $(\text{Abs-quasi-borel } (\text{UNIV}, \text{UNIV}))$*
by(*simp add: qbs-space-def Abs-quasi-borel-inverse[of (UNIV, UNIV), simplified is-quasi-borel-def qbs-closed1-def qbs-closed2-def qbs-closed3-def, simplified]*)
thus *?thesis*
using *from-list-in-list-of* **by** *blast*
qed

lemma *list-cons-in-list-of*:
assumes *set (a#l) \subseteq qbs-space X*
shows *list-cons a (from-list l) \in qbs-space $(\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$*
using *from-list-in-list-of[OF assms]* **by** *simp*

lemma *from-list-to-list-ident*:
to-list (from-list l) = l
by(*induction l*) (*simp add: to-list-def list-nil-def, simp add: to-list-simp2[OF from-list-in-list-of']*)

lemma *to-list-from-list-ident*:
assumes $l \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
shows *from-list (to-list l) = l*
proof(*rule list-induct-rule[OF assms]*)

fix $a\ l'$
assume $h: l' \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
and $ih: \text{from-list } (\text{to-list } l') = l'$
show $\text{from-list } (\text{to-list } (\text{list-cons } a\ l')) = \text{list-cons } a\ l'$
by $(\text{auto simp add: to-list-simp2}[\text{OF } h] ih[\text{simplified}])$
qed $(\text{simp add: to-list-simp1})$

definition $\text{rec-list}' :: 'b \Rightarrow ('a \Rightarrow (\text{nat} \times (\text{nat} \Rightarrow 'a)) \Rightarrow 'b \Rightarrow 'b) \Rightarrow (\text{nat} \times (\text{nat} \Rightarrow 'a)) \Rightarrow 'b$ **where**
 $\text{rec-list}'\ t0\ fl \equiv (\text{rec-list } t0\ (\lambda x\ l'. f\ x\ (\text{from-list } l'))\ (\text{to-list } l))$

lemma $\text{rec-list}'\text{-simp1}$:
 $\text{rec-list}'\ t\ f\ \text{list-nil} = t$
by $(\text{simp add: rec-list}'\text{-def to-list-simp1})$

lemma $\text{rec-list}'\text{-simp2}$:
assumes $l \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
shows $\text{rec-list}'\ t\ f\ (\text{list-cons } x\ l) = f\ x\ l\ (\text{rec-list}'\ t\ f\ l)$
by $(\text{simp add: rec-list}'\text{-def to-list-simp2}[\text{OF } \text{assms}] \text{to-list-from-list-ident}[\text{OF } \text{assms, simplified}])$

lemma list-qbs-space : $\text{qbs-space } (\text{list-qbs } X) = \text{lists } (\text{qbs-space } X)$
using to-list-set **by** $(\text{auto simp: list-qbs-def map-qbs-space image-def from-list-to-list-ident from-list-in-list-of subset-iff intro!}:\ \text{bexI}[\text{where } x=\text{from-list } -])$

3.1.11 Option Spaces

The option spaces is defined using the following isomorphism.

$$\text{Option}(X) \cong X + 1$$

definition $\text{option-qbs} :: 'a\ \text{quasi-borel} \Rightarrow 'a\ \text{option quasi-borel}$ **where**
 $\text{option-qbs } X = \text{map-qbs } (\lambda x. \text{case } x\ \text{of } \text{Inl } y \Rightarrow \text{Some } y \mid \text{Inr } y \Rightarrow \text{None})\ (X \oplus_Q 1_Q)$

lemma option-qbs-space : $\text{qbs-space } (\text{option-qbs } X) = \{\text{Some } x \mid x. x \in \text{qbs-space } X\} \cup \{\text{None}\}$
by $(\text{auto simp: option-qbs-def map-qbs-space copair-qbs-space})\ (\text{metis } \text{InrI image-eqI insert-iff old.sum.simps}(6), \text{metis } \text{InlI image-iff sum.case}(1))$

3.1.12 Function Spaces

definition $\text{exp-qbs} :: ['a\ \text{quasi-borel}, 'b\ \text{quasi-borel}] \Rightarrow ('a \Rightarrow 'b)\ \text{quasi-borel}$ (**infix** \Rightarrow_Q 61) **where**
 $X \Rightarrow_Q Y \equiv \text{Abs-quasi-borel } (\{f. \forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y\}, \{g. \forall \alpha \in \text{borel-measurable borel}. \forall \beta \in \text{qbs-Mx } X. (\lambda r. g\ (\alpha\ r)\ (\beta\ r)) \in \text{qbs-Mx } Y\})$

lemma
shows exp-qbs-space : $\text{qbs-space } (\text{exp-qbs } X\ Y) = \{f. \forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y\}$

and $exp\text{-}qbs\text{-}Mx: qbs\text{-}Mx (exp\text{-}qbs X Y) = \{g. \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y\}$
proof –
have $\{g:: real \Rightarrow -. \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y\} \subseteq UNIV \rightarrow \{f. \forall \alpha \in qbs\text{-}Mx X. f \circ \alpha \in qbs\text{-}Mx Y\}$
proof safe
fix $g :: real \Rightarrow -$ **and** $r :: real$ **and** α
assume $h: \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y$ $\alpha \in qbs\text{-}Mx X$
have $[simp]: g r \circ \alpha = (\lambda l. g r (\alpha l))$ **by** $(auto simp: comp\text{-}def)$
thus $g r \circ \alpha \in qbs\text{-}Mx Y$
using h **by** $auto$
qed
moreover have $qbs\text{-}closed3 \{g. \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y\}$
by $(rule qbs\text{-}closed3I, auto) (rule qbs\text{-}closed3\text{-}dest, auto)$
ultimately have $Rep\text{-}quasi\text{-}borel (exp\text{-}qbs X Y) = (\{f. \forall \alpha \in qbs\text{-}Mx X. f \circ \alpha \in qbs\text{-}Mx Y\}, \{g. \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y\})$
unfolding $exp\text{-}qbs\text{-}def$ **by** $(auto intro!: Abs\text{-}quasi\text{-}borel\text{-}inverse\ is\ quasi\text{-}borel\text{-}intro qbs\text{-}closed1I qbs\text{-}closed2I simp: comp\text{-}def)$
thus $qbs\text{-}space (exp\text{-}qbs X Y) = \{f. \forall \alpha \in qbs\text{-}Mx X. f \circ \alpha \in qbs\text{-}Mx Y\}$
 $qbs\text{-}Mx (exp\text{-}qbs X Y) = \{g. \forall \alpha \in borel\text{-}measurable borel. \forall \beta \in qbs\text{-}Mx X. (\lambda r. g (\alpha r) (\beta r)) \in qbs\text{-}Mx Y\}$
by $(simp\text{-}all add: qbs\text{-}space\text{-}def qbs\text{-}Mx\text{-}def)$
qed

3.1.13 Ordering on Quasi-Borel Spaces

inductive-set $generating\text{-}Mx :: 'a\ set \Rightarrow (real \Rightarrow 'a)\ set \Rightarrow (real \Rightarrow 'a)\ set$
for $X :: 'a\ set$ **and** $Mx :: (real \Rightarrow 'a)\ set$
where
 $Basic: \alpha \in Mx \Longrightarrow \alpha \in generating\text{-}Mx X Mx$
 $| Const: x \in X \Longrightarrow (\lambda r. x) \in generating\text{-}Mx X Mx$
 $| Comp : f \in (borel :: real\ measure) \rightarrow_M (borel :: real\ measure) \Longrightarrow \alpha \in generating\text{-}Mx X Mx \Longrightarrow \alpha \circ f \in generating\text{-}Mx X Mx$
 $| Part : (\bigwedge i. Fi\ i \in generating\text{-}Mx X Mx) \Longrightarrow P \in borel \rightarrow_M count\text{-}space (UNIV :: nat\ set) \Longrightarrow (\lambda r. Fi (P r) r) \in generating\text{-}Mx X Mx$

lemma $generating\text{-}Mx\text{-}to\text{-}space:$

assumes $Mx \subseteq UNIV \rightarrow X$
shows $generating\text{-}Mx X Mx \subseteq UNIV \rightarrow X$

proof

fix α
assume $\alpha \in generating\text{-}Mx X Mx$
then show $\alpha \in UNIV \rightarrow X$
by $(induct\ rule: generating\text{-}Mx.induct) (use\ assms\ in\ auto)$

qed


```

lemma generating-Mx-closed1:
  qbs-closed1 (generating-Mx X Mx)
  by (simp add: generating-Mx.Comp qbs-closed1I)

lemma generating-Mx-closed2:
  qbs-closed2 X (generating-Mx X Mx)
  by (simp add: generating-Mx.Const qbs-closed2I)

lemma generating-Mx-closed3:
  qbs-closed3 (generating-Mx X Mx)
  by(simp add: qbs-closed3I generating-Mx.Part)

lemma generating-Mx-Mx:
  generating-Mx (qbs-space X) (qbs-Mx X) = qbs-Mx X
proof safe
  fix  $\alpha$ 
  assume  $\alpha \in \text{generating-Mx } (qbs\text{-space } X) (qbs\text{-Mx } X)$ 
  then show  $\alpha \in qbs\text{-Mx } X$ 
  by(rule generating-Mx.induct) (auto intro!: qbs-closed1-dest[simplified comp-def]
simp: qbs-closed3-dest')
  next
  fix  $\alpha$ 
  assume  $\alpha \in qbs\text{-Mx } X$ 
  then show  $\alpha \in \text{generating-Mx } (qbs\text{-space } X) (qbs\text{-Mx } X) ..$ 
qed

instantiation quasi-borel :: (type) order-bot
begin

inductive less-eq-quasi-borel :: 'a quasi-borel  $\Rightarrow$  'a quasi-borel  $\Rightarrow$  bool where
  qbs-space X  $\subset$  qbs-space Y  $\Longrightarrow$  less-eq-quasi-borel X Y
| qbs-space X = qbs-space Y  $\Longrightarrow$  qbs-Mx Y  $\subseteq$  qbs-Mx X  $\Longrightarrow$  less-eq-quasi-borel X
  Y

lemma le-quasi-borel-iff:
  X  $\leq$  Y  $\longleftrightarrow$  (if qbs-space X = qbs-space Y then qbs-Mx Y  $\subseteq$  qbs-Mx X else
qbs-space X  $\subset$  qbs-space Y)
  by(auto elim: less-eq-quasi-borel.cases intro: less-eq-quasi-borel.intros)

definition less-quasi-borel :: 'a quasi-borel  $\Rightarrow$  'a quasi-borel  $\Rightarrow$  bool where
  less-quasi-borel X Y  $\longleftrightarrow$  (X  $\leq$  Y  $\wedge$   $\neg$  Y  $\leq$  X)

definition bot-quasi-borel :: 'a quasi-borel where
  bot-quasi-borel = empty-quasi-borel

instance
proof
  show bot  $\leq$  a for a :: 'a quasi-borel
  using qbs-empty-equiv

```

by(auto simp add: le-quasi-borel-iff bot-quasi-borel-def)
qed (auto simp: le-quasi-borel-iff less-quasi-borel-def split: if-split-asm intro: qbs-eqI)
end

definition *inf-quasi-borel* :: [*'a quasi-borel, 'a quasi-borel*] \Rightarrow *'a quasi-borel* **where**
inf-quasi-borel *X X'* = *Abs-quasi-borel* (*qbs-space* *X* \cap *qbs-space* *X'*, *qbs-Mx* *X* \cap
qbs-Mx *X'*)

lemma *inf-quasi-borel-correct*: *Rep-quasi-borel* (*inf-quasi-borel* *X X'*) = (*qbs-space*
X \cap *qbs-space* *X'*, *qbs-Mx* *X* \cap *qbs-Mx* *X'*)

by(auto intro!: *Abs-quasi-borel-inverse* simp: *inf-quasi-borel-def is-quasi-borel-def*
qbs-closed1-def qbs-closed2-def qbs-closed3-def dest: qbs-Mx-to-X)

lemma *inf-qbs-space[simp]*: *qbs-space* (*inf-quasi-borel* *X X'*) = *qbs-space* *X* \cap *qbs-space*
X'

by (simp add: *qbs-space-def inf-quasi-borel-correct*)

lemma *inf-qbs-Mx[simp]*: *qbs-Mx* (*inf-quasi-borel* *X X'*) = *qbs-Mx* *X* \cap *qbs-Mx* *X'*
by (simp add: *qbs-Mx-def inf-quasi-borel-correct*)

definition *max-quasi-borel* :: *'a set* \Rightarrow *'a quasi-borel* **where**
max-quasi-borel *X* = *Abs-quasi-borel* (*X*, *UNIV* \rightarrow *X*)

lemma *max-quasi-borel-correct*: *Rep-quasi-borel* (*max-quasi-borel* *X*) = (*X*, *UNIV*
 \rightarrow *X*)

by(fastforce intro!: *Abs-quasi-borel-inverse*

simp: max-quasi-borel-def qbs-closed1-def qbs-closed2-def qbs-closed3-def is-quasi-borel-def)

lemma *max-qbs-space[simp]*: *qbs-space* (*max-quasi-borel* *X*) = *X*

by (simp add: *qbs-space-def max-quasi-borel-correct*)

lemma *max-qbs-Mx[simp]*: *qbs-Mx* (*max-quasi-borel* *X*) = *UNIV* \rightarrow *X*

by (simp add: *qbs-Mx-def max-quasi-borel-correct*)

instantiation *quasi-borel* :: (*type*) *semilattice-sup*
begin

definition *sup-quasi-borel* :: [*'a quasi-borel* \Rightarrow *'a quasi-borel* \Rightarrow *'a quasi-borel*] **where**
sup-quasi-borel *X Y* \equiv (if *qbs-space* *X* = *qbs-space* *Y* then *inf-quasi-borel* *X Y*
else if *qbs-space* *X* \subset *qbs-space* *Y* then *Y*
else if *qbs-space* *Y* \subset *qbs-space* *X* then *X*
else *max-quasi-borel* (*qbs-space* *X* \cup *qbs-space* *Y*))

instance

proof

fix *X Y* :: *'a quasi-borel*

let ?*X* = *qbs-space* *X*

let ?*Y* = *qbs-space* *Y*

```

consider ?X = ?Y | ?X ⊂ ?Y | ?Y ⊂ ?X | ?X ⊂ ?X ∪ ?Y ∧ ?Y ⊂ ?X ∪ ?Y
  by auto
then show X ≤ X ⊔ Y
proof(cases)
  case 1
    show ?thesis
    unfolding sup-quasi-borel-def
    by(rule less-eq-quasi-borel.intros(2),simp-all add: 1)
  next
    case 2
      then show ?thesis
      unfolding sup-quasi-borel-def
      by (simp add: less-eq-quasi-borel.intros(1))
    next
      case 3
        then show ?thesis
        unfolding sup-quasi-borel-def
        by auto
      next
        case 4
          then show ?thesis
          unfolding sup-quasi-borel-def
          by(auto simp: less-eq-quasi-borel.intros(1))
    qed
  next
    fix X Y :: 'a quasi-borel
    let ?X = qbs-space X
    let ?Y = qbs-space Y
    consider ?X = ?Y | ?X ⊂ ?Y | ?Y ⊂ ?X | ?X ⊂ ?X ∪ ?Y ∧ ?Y ⊂ ?X ∪ ?Y
      by auto
    then show Y ≤ X ⊔ Y
    proof(cases)
      case 1
        show ?thesis
        unfolding sup-quasi-borel-def
        by(rule less-eq-quasi-borel.intros(2)) (simp-all add: 1)
      next
        case 2
          then show ?thesis
          unfolding sup-quasi-borel-def
          by auto
        next
          case 3
            then show ?thesis
            unfolding sup-quasi-borel-def
            by (auto simp add: less-eq-quasi-borel.intros(1))
          next
            case 4
              then show ?thesis

```

```

    unfolding sup-quasi-borel-def
    by(auto simp: less-eq-quasi-borel.intros(1))
qed
next
fix X Y Z :: 'a quasi-borel
assume h: X ≤ Z Y ≤ Z
let ?X = qbs-space X
let ?Y = qbs-space Y
let ?Z = qbs-space Z
consider ?X = ?Y | ?X ⊂ ?Y | ?Y ⊂ ?X | ?X ⊂ ?X ∪ ?Y ∧ ?Y ⊂ ?X ∪ ?Y
  by auto
then show sup X Y ≤ Z
proof cases
  case 1
  show ?thesis
    unfolding sup-quasi-borel-def
    apply(simp add: 1, rule less-eq-quasi-borel.cases[OF h(1)])
    apply(rule less-eq-quasi-borel.intros(1))
    apply auto[1]
    apply simp
    apply(rule less-eq-quasi-borel.intros(2))
    apply(simp add: 1)
    apply(rule less-eq-quasi-borel.cases[OF h(2)])
    using 1
    apply fastforce
    apply simp
    by (metis 1 h(2) inf-qbs-Mx le-inf-iff le-quasi-borel-iff)

next
  case 2
  then show ?thesis
    unfolding sup-quasi-borel-def
    using h(2) by auto
next
  case 3
  then show ?thesis
    unfolding sup-quasi-borel-def
    using h(1) by auto
next
  case 4
  then have [simp]: ?X ≠ ?Y ~ (?X ⊂ ?Y) ~ (?Y ⊂ ?X)
    by auto
  have [simp]: ?X ⊆ ?Z ?Y ⊆ ?Z
    by (metis h(1) dual-order.order-iff-strict less-eq-quasi-borel.cases)
    (metis h(2) dual-order.order-iff-strict less-eq-quasi-borel.cases)
  then consider ?X ∪ ?Y = ?Z | ?X ∪ ?Y ⊂ ?Z
    by blast
  then show ?thesis
    unfolding sup-quasi-borel-def

```

```

    apply cases
    apply simp
    apply (rule less-eq-quasi-borel.intros(2))
    apply simp
    using qbs-Mx-to-X apply auto[1]
    by (simp add: less-eq-quasi-borel.intros(1))
  qed
qed

end

end

```

3.2 Morphisms of Quasi-Borel Spaces

theory *QBS-Morphism*

imports

QuasiBorel

begin

abbreviation *qbs-morphism* :: [*'a quasi-borel, 'b quasi-borel*] \Rightarrow (*'a \Rightarrow 'b*) *set*
(infixr \rightarrow_Q 60) **where**

$X \rightarrow_Q Y \equiv \text{qbs-space } (X \Rightarrow_Q Y)$

lemma *qbs-morphismI*: $(\bigwedge \alpha. \alpha \in \text{qbs-Mx } X \Longrightarrow f \circ \alpha \in \text{qbs-Mx } Y) \Longrightarrow f \in X \rightarrow_Q Y$

by (*auto simp: exp-qbs-space*)

lemma *qbs-morphism-def*: $X \rightarrow_Q Y = \{f \in \text{qbs-space } X \rightarrow \text{qbs-space } Y. \forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y\}$

unfolding *exp-qbs-space*

proof *safe*

fix *f x*

assume $h: x \in \text{qbs-space } X \ \forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y$

then have $(\lambda r. x) \in \text{qbs-Mx } X$

by *simp*

hence $f \circ (\lambda r. x) \in \text{qbs-Mx } Y$

using *h* **by** *blast*

with *qbs-Mx-to-X* **show** $f x \in \text{qbs-space } Y$

by *auto*

qed *auto*

lemma *qbs-morphism-Mx*:

assumes $f \in X \rightarrow_Q Y \ \alpha \in \text{qbs-Mx } X$

shows $f \circ \alpha \in \text{qbs-Mx } Y$

using *assms* **by** (*auto simp: qbs-morphism-def*)

```

lemma qbs-morphism-space:
  assumes  $f \in X \rightarrow_Q Y$   $x \in \text{qbs-space } X$ 
  shows  $f x \in \text{qbs-space } Y$ 
  using assms by(auto simp: qbs-morphism-def)

lemma qbs-morphism-ident[simp]:
   $id \in X \rightarrow_Q X$ 
  by(auto intro: qbs-morphismI)

lemma qbs-morphism-ident'[simp]:
   $(\lambda x. x) \in X \rightarrow_Q X$ 
  using qbs-morphism-ident by(simp add: id-def)

lemma qbs-morphism-comp:
  assumes  $f \in X \rightarrow_Q Y$   $g \in Y \rightarrow_Q Z$ 
  shows  $g \circ f \in X \rightarrow_Q Z$ 
  using assms by (simp add: comp-assoc Pi-def qbs-morphism-def)

lemma qbs-morphism-compose-rev:
  assumes  $f \in Y \rightarrow_Q Z$  and  $g \in X \rightarrow_Q Y$ 
  shows  $(\lambda x. f (g x)) \in X \rightarrow_Q Z$ 
  using qbs-morphism-comp[OF assms(2,1)] by(simp add: comp-def)

lemma qbs-morphism-compose:
  assumes  $g \in X \rightarrow_Q Y$  and  $f \in Y \rightarrow_Q Z$ 
  shows  $(\lambda x. f (g x)) \in X \rightarrow_Q Z$ 
  using qbs-morphism-compose-rev[OF assms(2,1)] .

lemma qbs-morphism-cong':
  assumes  $\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$ 
  and  $f \in X \rightarrow_Q Y$ 
  shows  $g \in X \rightarrow_Q Y$ 
proof(rule qbs-morphismI)
  fix  $\alpha$ 
  assume  $1:\alpha \in \text{qbs-Mx } X$ 
  have  $g \circ \alpha = f \circ \alpha$ 
  proof
    fix  $x$ 
    have  $\alpha x \in \text{qbs-space } X$ 
    using 1 qbs-decomp[of X] qbs-Mx-to-X by auto
    thus  $(g \circ \alpha) x = (f \circ \alpha) x$ 
    using assms(1) by simp
  qed
  thus  $g \circ \alpha \in \text{qbs-Mx } Y$ 
  using 1 assms(2) by(simp add: qbs-morphism-def)
qed

lemma qbs-morphism-cong:
  assumes  $\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$ 

```

shows $f \in X \rightarrow_Q Y \iff g \in X \rightarrow_Q Y$
using *assms* **by** (*auto simp: qbs-morphism-cong'[of - f g] qbs-morphism-cong'[of - g f]*)

lemma *qbs-morphism-const*:
assumes $y \in \text{qbs-space } Y$
shows $(\lambda x. y) \in X \rightarrow_Q Y$
using *assms* **by** (*auto intro: qbs-morphismI*)

lemma *qbs-morphism-from-empty*: $\text{qbs-space } X = \{\} \implies f \in X \rightarrow_Q Y$
by (*auto intro!: qbs-morphismI simp: qbs-empty-equiv*)

lemma *unit-quasi-borel-terminal*: $\exists! f. f \in X \rightarrow_Q \text{unit-quasi-borel}$
by (*fastforce simp: qbs-morphism-def*)

definition *to-unit-quasi-borel* :: $'a \Rightarrow \text{unit } (!_Q)$ **where**
to-unit-quasi-borel $\equiv (\lambda r. ())$

lemma *to-unit-quasi-borel-morphism*:
 $!_Q \in X \rightarrow_Q \text{unit-quasi-borel}$
by (*auto simp add: to-unit-quasi-borel-def qbs-morphism-def*)

lemma *qbs-morphism-subD*:
assumes $f \in X \rightarrow_Q \text{sub-qbs } Y A$
shows $f \in X \rightarrow_Q Y$
using *qbs-morphism-Mx[OF assms]* **by** (*auto intro!: qbs-morphismI simp: sub-qbs-Mx*)

lemma *qbs-morphism-subI1*:
assumes $f \in X \rightarrow_Q Y \wedge x. x \in \text{qbs-space } X \implies f x \in A$
shows $f \in X \rightarrow_Q \text{sub-qbs } Y A$
using *qbs-morphism-space[OF assms(1)] qbs-morphism-Mx[OF assms(1)] assms(2)*
qbs-Mx-to-X[of - X]
by (*auto intro!: qbs-morphismI simp: sub-qbs-Mx*)

lemma *qbs-morphism-subI2*:
assumes $f \in X \rightarrow_Q Y$
shows $f \in \text{sub-qbs } X A \rightarrow_Q Y$
using *qbs-morphism-Mx[OF assms]* **by** (*auto intro!: qbs-morphismI simp: sub-qbs-Mx*)

corollary *qbs-morphism-subsubI*:
assumes $f \in X \rightarrow_Q Y \wedge x. x \in A \implies f x \in B$
shows $f \in \text{sub-qbs } X A \rightarrow_Q \text{sub-qbs } Y B$
by (*rule qbs-morphism-subI1*) (*auto intro!: qbs-morphism-subI2 assms simp: sub-qbs-space*)

lemma *map-qbs-morphism-f*: $f \in X \rightarrow_Q \text{map-qbs } f X$
by (*auto intro!: qbs-morphismI simp: map-qbs-Mx*)

lemma *map-qbs-morphism-inverse-f*:
assumes $\wedge x. x \in \text{qbs-space } X \implies g (f x) = x$

shows $g \in \text{map-qbs } f X \rightarrow_Q X$
proof –
{
 fix α
 assume $h:\alpha \in \text{qbs-Mx } X$
 from $\text{qbs-Mx-to-X}[OF \text{ this}] \text{ assms}$ **have** $g \circ (f \circ \alpha) = \alpha$
 by *auto*
 with h **have** $g \circ (f \circ \alpha) \in \text{qbs-Mx } X$ **by** *simp*
}
thus *?thesis*
 by(*auto intro!*: *qbs-morphismI simp: map-qbs-Mx*)
qed

lemma *pair-qbs-morphismI*:
assumes $\bigwedge \alpha \beta. \alpha \in \text{qbs-Mx } X \implies \beta \in \text{qbs-Mx } Y$
 $\implies (\lambda r. f (\alpha r, \beta r)) \in \text{qbs-Mx } Z$
shows $f \in (X \otimes_Q Y) \rightarrow_Q Z$
using *assms* **by**(*fastforce intro!*: *qbs-morphismI simp: pair-qbs-Mx comp-def*)

lemma *pair-qbs-MxD*:
assumes $\gamma \in \text{qbs-Mx } (X \otimes_Q Y)$
obtains $\alpha \beta$ **where** $\alpha \in \text{qbs-Mx } X \beta \in \text{qbs-Mx } Y \gamma = (\lambda x. (\alpha x, \beta x))$
using *assms* **by**(*auto simp: pair-qbs-Mx*)

lemma *pair-qbs-MxI*:
assumes $(\lambda x. \text{fst } (\gamma x)) \in \text{qbs-Mx } X$ **and** $(\lambda x. \text{snd } (\gamma x)) \in \text{qbs-Mx } Y$
shows $\gamma \in \text{qbs-Mx } (X \otimes_Q Y)$
using *assms* **by**(*auto simp: pair-qbs-Mx comp-def*)

lemma
shows *fst-qbs-morphism*: $\text{fst} \in X \otimes_Q Y \rightarrow_Q X$
 and *snd-qbs-morphism*: $\text{snd} \in X \otimes_Q Y \rightarrow_Q Y$
by(*auto intro!*: *pair-qbs-morphismI simp: comp-def*)

lemma *qbs-morphism-pair-iff*:
 $f \in X \rightarrow_Q Y \otimes_Q Z \iff \text{fst} \circ f \in X \rightarrow_Q Y \wedge \text{snd} \circ f \in X \rightarrow_Q Z$
by(*auto intro!*: *qbs-morphism-comp fst-qbs-morphism snd-qbs-morphism*)
 (*auto dest: qbs-morphism-Mx intro!*: *qbs-morphismI simp: pair-qbs-Mx comp-assoc[symmetric]*)

lemma *qbs-morphism-Pair*:
assumes $f \in Z \rightarrow_Q X$
 and $g \in Z \rightarrow_Q Y$
 shows $(\lambda z. (f z, g z)) \in Z \rightarrow_Q X \otimes_Q Y$
unfolding *qbs-morphism-pair-iff*
using *assms* **by** (*auto simp: comp-def*)

lemma *qbs-morphism-curry*: $\text{curry} \in \text{exp-qbs } (X \otimes_Q Y) Z \rightarrow_Q \text{exp-qbs } X (\text{exp-qbs } Y Z)$
by(*auto intro!*: *qbs-morphismI simp: pair-qbs-Mx exp-qbs-Mx comp-def*)

corollary *curry-preserves-morphisms*:

assumes $(\lambda xy. f (fst xy) (snd xy)) \in X \otimes_Q Y \rightarrow_Q Z$

shows $f \in X \rightarrow_Q Y \Rightarrow_Q Z$

using *qbs-morphism-space*[*OF qbs-morphism-curry assms*] **by** (*auto simp: curry-def*)

lemma *qbs-morphism-eval*:

$(\lambda fx. (fst fx) (snd fx)) \in (X \Rightarrow_Q Y) \otimes_Q X \rightarrow_Q Y$

by(*auto intro!: qbs-morphismI simp: pair-qbs-Mx exp-qbs-Mx comp-def*)

corollary *qbs-morphism-app*:

assumes $f \in X \rightarrow_Q (Y \Rightarrow_Q Z)$ $g \in X \rightarrow_Q Y$

shows $(\lambda x. (f x) (g x)) \in X \rightarrow_Q Z$

by(*rule qbs-morphism-cong'[where f=($\lambda fx. (fst fx) (snd fx)$) \circ ($\lambda x. (f x, g x)$), *OF* - *qbs-morphism-comp*[*OF qbs-morphism-Pair*[*OF assms*] *qbs-morphism-eval*]]) *auto**

ML-file \langle *qbs.ML* \rangle

attribute-setup *qbs* = \langle

Attrib.add-del Qbs.qbs-add Qbs.qbs-del

declaration of qbs rule

method-setup *qbs* = \langle *Scan.lift (Scan.succeed (METHOD o Qbs.qbs-tac))* \rangle

simproc-setup *qbs* ($x \in$ *qbs-space* *X*) = \langle *K Qbs.simproc* \rangle

declare

fst-qbs-morphism[*qbs*]

snd-qbs-morphism[*qbs*]

qbs-morphism-const[*qbs*]

qbs-morphism-ident[*qbs*]

qbs-morphism-ident'[*qbs*]

qbs-morphism-curry[*qbs*]

lemma [*qbs*]:

shows *qbs-morphism-Pair1*: $Pair \in X \rightarrow_Q Y \Rightarrow_Q (X \otimes_Q Y)$

by(*auto intro!: qbs-morphismI simp: exp-qbs-Mx pair-qbs-Mx comp-def*)

lemma *qbs-morphism-case-prod*[*qbs*]: $case-prod \in exp-qbs X (exp-qbs Y Z) \rightarrow_Q exp-qbs (X \otimes_Q Y) Z$

by(*fastforce intro!: qbs-morphismI simp: exp-qbs-Mx pair-qbs-Mx comp-def split-beta'*)

lemma *uncurry-preserves-morphisms*:

assumes [*qbs*]: $(\lambda x y. f (x,y)) \in X \rightarrow_Q Y \Rightarrow_Q Z$

shows $f \in X \otimes_Q Y \rightarrow_Q Z$

by(*rule qbs-morphism-cong'[where f=case-prod ($\lambda x y. f (x,y)$)], *simp*) *qbs**

lemma *qbs-morphism-comp'*[*qbs*]: $comp \in Y \Rightarrow_Q Z \rightarrow_Q (X \Rightarrow_Q Y) \Rightarrow_Q X \Rightarrow_Q Z$

by(*auto intro!: qbs-morphismI simp: exp-qbs-Mx*)

lemma *arg-swap-morphism*:

assumes $f \in X \rightarrow_Q \text{exp-qbs } Y \ Z$
shows $(\lambda y x. f x y) \in Y \rightarrow_Q \text{exp-qbs } X \ Z$
using *assms by simp*

lemma *exp-qbs-comp-morphism*:

assumes $f \in W \rightarrow_Q \text{exp-qbs } X \ Y$
and $g \in W \rightarrow_Q \text{exp-qbs } Y \ Z$
shows $(\lambda w. g w \circ f w) \in W \rightarrow_Q \text{exp-qbs } X \ Z$
using *assms by qbs*

lemma *arg-swap-morphism-map-qbs1*:

assumes $g \in \text{exp-qbs } W \ (\text{exp-qbs } X \ Y) \rightarrow_Q \ Z$
shows $(\lambda k. g (k \circ f)) \in \text{exp-qbs } (\text{map-qbs } f \ W) \ (\text{exp-qbs } X \ Y) \rightarrow_Q \ Z$
using *assms map-qbs-morphism-f by qbs*

lemma *qbs-morphism-map-prod[qbs]*: $\text{map-prod} \in X \Rightarrow_Q Y \rightarrow_Q (W \Rightarrow_Q Z) \Rightarrow_Q (X \otimes_Q W) \Rightarrow_Q (Y \otimes_Q Z)$

by(*auto intro!*: *qbs-morphismI simp: exp-qbs-Mx pair-qbs-Mx map-prod-def comp-def case-prod-beta'*)

lemma *qbs-morphism-pair-swap*:

assumes $f \in X \otimes_Q Y \rightarrow_Q Z$
shows $(\lambda(x,y). f (y,x)) \in Y \otimes_Q X \rightarrow_Q Z$
using *assms by simp*

lemma

shows *qbs-morphism-pair-assoc1*: $(\lambda((x,y),z). (x,(y,z))) \in (X \otimes_Q Y) \otimes_Q Z \rightarrow_Q X \otimes_Q (Y \otimes_Q Z)$
and *qbs-morphism-pair-assoc2*: $(\lambda(x,(y,z)). ((x,y),z)) \in X \otimes_Q (Y \otimes_Q Z) \rightarrow_Q (X \otimes_Q Y) \otimes_Q Z$
by *simp-all*

lemma *Inl-qbs-morphism[qbs]*: $\text{Inl} \in X \rightarrow_Q X \oplus_Q Y$

by(*auto intro!*: *qbs-morphismI beXI[where x={}] simp: copair-qbs-Mx copair-qbs-Mx-def comp-def*)

lemma *Inr-qbs-morphism[qbs]*: $\text{Inr} \in Y \rightarrow_Q X \oplus_Q Y$

by(*auto intro!*: *qbs-morphismI beXI[where x=UNIV] simp: copair-qbs-Mx copair-qbs-Mx-def comp-def*)

lemma *case-sum-qbs-morphism[qbs]*: $\text{case-sum} \in X \Rightarrow_Q Z \rightarrow_Q (Y \Rightarrow_Q Z) \Rightarrow_Q (X \oplus_Q Y \Rightarrow_Q Z)$

by(*auto intro!*: *qbs-morphismI qbs-Mx-indicat simp: copair-qbs-Mx copair-qbs-Mx-def exp-qbs-Mx case-sum-if*)

lemma *map-sum-qbs-morphism[qbs]*: $\text{map-sum} \in X \Rightarrow_Q Y \rightarrow_Q (X' \Rightarrow_Q Y') \Rightarrow_Q (X \oplus_Q X' \Rightarrow_Q Y \oplus_Q Y')$

```

proof(rule qbs-morphismI)
  fix  $\alpha$ 
  assume  $\alpha \in \text{qbs-Mx } (X \Rightarrow_Q Y)$ 
  then have  $ha[\text{measurable}]: \forall (k :: \text{real} \Rightarrow \text{real}) \in \text{borel-measurable borel}. \forall a \in \text{qbs-Mx } X. (\lambda r. \alpha (k r) (a r)) \in \text{qbs-Mx } Y$ 
  by (auto simp: exp-qbs-Mx)
  show  $\text{map-sum} \circ \alpha \in \text{qbs-Mx } ((X' \Rightarrow_Q Y') \Rightarrow_Q X \oplus_Q X' \Rightarrow_Q Y \oplus_Q Y')$ 
  unfolding exp-qbs-Mx
  proof safe
    fix  $\beta$   $b$  and  $f g :: \text{real} \Rightarrow \text{real}$ 
    assume  $h[\text{measurable}]: \forall (k :: \text{real} \Rightarrow \text{real}) \in \text{borel-measurable borel}. \forall b \in \text{qbs-Mx } X'. (\lambda r. \beta (k r) (b r)) \in \text{qbs-Mx } Y'$ 
     $f \in \text{borel-measurable borel } g \in \text{borel-measurable borel}$ 
    and  $b: b \in \text{qbs-Mx } (X \oplus_Q X')$ 
    show  $(\lambda r. (\text{map-sum} \circ \alpha) (f (g r)) (\beta (g r)) (b r)) \in \text{qbs-Mx } (Y \oplus_Q Y')$ 
    proof(rule copair-qbs-MxD[OF b])
      fix  $a$ 
      assume  $a \in \text{qbs-Mx } X$   $b = (\lambda r. \text{Inl } (a r))$ 
      with  $ha$  show  $(\lambda r. (\text{map-sum} \circ \alpha) (f (g r)) (\beta (g r)) (b r)) \in \text{qbs-Mx } (Y \oplus_Q Y')$ 
      by(auto simp: copair-qbs-Mx copair-qbs-Mx-def intro!: beXI[where  $x=\{\}$ ])
    next
      fix  $a$ 
      assume  $a \in \text{qbs-Mx } X'$   $b = (\lambda r. \text{Inr } (a r))$ 
      with  $h(1)$  show  $(\lambda r. (\text{map-sum} \circ \alpha) (f (g r)) (\beta (g r)) (b r)) \in \text{qbs-Mx } (Y \oplus_Q Y')$ 
      by(auto simp: copair-qbs-Mx copair-qbs-Mx-def intro!: beXI[where  $x=UNIV$ ])
    next
      fix  $S$   $a'$ 
      assume  $S \in \text{sets borel } S \neq \{\} S \neq UNIV a \in \text{qbs-Mx } X a' \in \text{qbs-Mx } X' b = (\lambda r. \text{if } r \in S \text{ then } \text{Inl } (a r) \text{ else } \text{Inr } (a' r))$ 
      with  $h$   $ha$  show  $(\lambda r. (\text{map-sum} \circ \alpha) (f (g r)) (\beta (g r)) (b r)) \in \text{qbs-Mx } (Y \oplus_Q Y')$ 
      by simp (fastforce simp: copair-qbs-Mx copair-qbs-Mx-def intro!: beXI[where  $x=S$ ])
    qed
  qed
qed

```

lemma qbs-morphism-component-singleton[qbs]:

```

assumes  $i \in I$ 
shows  $(\lambda x. x i) \in (\Pi_Q i \in I. (M i)) \rightarrow_Q M i$ 
by(auto intro!: qbs-morphismI simp: comp-def assms PiQ-Mx)

```

lemma qbs-morphism-component-singleton':

```

assumes  $f \in Y \rightarrow_Q (\Pi_Q i \in I. X i) g \in Z \rightarrow_Q Y i \in I$ 
shows  $(\lambda x. f (g x) i) \in Z \rightarrow_Q X i$ 
by(auto intro!: qbs-morphism-compose[OF assms(2)] qbs-morphism-compose[OF assms(1)] qbs-morphism-component-singleton assms)

```

lemma *product-qbs-canonical1*:
assumes $\bigwedge i. i \in I \implies f i \in Y \rightarrow_Q X i$
and $\bigwedge i. i \notin I \implies f i = (\lambda y. \text{undefined})$
shows $(\lambda y i. f i y) \in Y \rightarrow_Q (\prod_Q i \in I. X i)$
using *assms qbs-morphism-Mx[OF assms(1)]* **by** (*auto intro!*: *qbs-morphismI simp*:
PiQ-Mx comp-def)

lemma *product-qbs-canonical2*:
assumes $\bigwedge i. i \in I \implies f i \in Y \rightarrow_Q X i$
 $\bigwedge i. i \notin I \implies f i = (\lambda y. \text{undefined})$
 $g \in Y \rightarrow_Q (\prod_Q i \in I. X i)$
 $\bigwedge i. i \in I \implies f i = (\lambda x. x i) \circ g$
and $y \in \text{qbs-space } Y$
shows $g y = (\lambda i. f i y)$
proof (*intro ext*)
fix i
show $g y i = f i y$
proof (*cases i \in I*)
case *True*
then show *?thesis*
using *assms(4)[of i]* **by** *simp*
next
case *False*
with *qbs-morphism-space[OF assms(3)] assms(2,3,5)* **show** *?thesis*
by (*auto simp: PiQ-Mx PiQ-space*)
qed
qed

lemma *merge-qbs-morphism*:
 $\text{merge } I J \in (\prod_Q i \in I. (M i)) \otimes_Q (\prod_Q j \in J. (M j)) \rightarrow_Q (\prod_Q i \in I \cup J. (M i))$
proof (*rule qbs-morphismI*)
fix α
assume $h: \alpha \in \text{qbs-Mx } ((\prod_Q i \in I. (M i)) \otimes_Q (\prod_Q j \in J. (M j)))$
show $\text{merge } I J \circ \alpha \in \text{qbs-Mx } (\prod_Q i \in I \cup J. (M i))$
proof –
{
fix i
assume $i \in I \cup J$
then consider $i \in I \mid i \in I \wedge i \in J \mid i \notin I \wedge i \in J$
by *auto*
hence $(\lambda r. (\text{merge } I J \circ \alpha) r i) \in \text{qbs-Mx } (M i)$
by *cases (insert h, auto simp: merge-def split-beta' pair-qbs-Mx PiQ-Mx)*
}
thus *?thesis*
by (*auto simp: PiQ-Mx (auto simp: merge-def split-beta')*)
qed
qed

lemma *ini-morphism*[qbs]:
assumes $j \in I$
shows $(\lambda x. (j, x)) \in X j \rightarrow_Q (\coprod_Q i \in I. X i)$
by(*fastforce intro!*: *qbs-morphismI exI*[**where** $x = \lambda r. j$] *simp*: *coPiQ-Mx-def comp-def assms coPiQ-Mx*)

lemma *coPiQ-canonical1*:
assumes *countable I*
and $\bigwedge i. i \in I \implies f i \in X i \rightarrow_Q Y$
shows $(\lambda(i, x). f i x) \in (\coprod_Q i \in I. X i) \rightarrow_Q Y$
proof(*rule qbs-morphismI*)
fix α
assume $\alpha \in \text{qbs-Mx } (\text{coPiQ } I X)$
then obtain βg **where** *ha*:
 $\bigwedge i. i \in \text{range } g \implies \beta i \in \text{qbs-Mx } (X i) \alpha = (\lambda r. (g r, \beta (g r) r))$ **and**
hg[*measurable*]: $g \in \text{borel } \rightarrow_M \text{count-space } I$
by(*fastforce simp*: *coPiQ-Mx-def coPiQ-Mx*)
define f' **where** $f' \equiv (\lambda i r. f i (\beta i r))$
have $\text{range } g \subseteq I$
using *measurable-space*[*OF hg*] **by** *auto*
hence $1: (\bigwedge i. i \in \text{range } g \implies f' i \in \text{qbs-Mx } Y)$
using *qbs-morphism-Mx*[*OF assms(2) ha(1), simplified comp-def*]
by(*auto simp*: *f'-def*)
have $(\lambda(i, x). f i x) \circ \alpha = (\lambda r. f' (g r) r)$
by(*auto simp*: *ha(2) f'-def*)
also have $\dots \in \text{qbs-Mx } Y$
by(*auto intro!*: *qbs-closed3-dest2'*[*OF assms(1) hg, of f', OF 1*])
finally show $(\lambda(i, x). f i x) \circ \alpha \in \text{qbs-Mx } Y$.
qed

lemma *coPiQ-canonical1'*:
assumes *countable I*
and $\bigwedge i. i \in I \implies (\lambda x. f (i, x)) \in X i \rightarrow_Q Y$
shows $f \in (\coprod_Q i \in I. X i) \rightarrow_Q Y$
using *coPiQ-canonical1*[**where** $f = \text{curry } f$] *assms* **by**(*auto simp*: *curry-def*)

lemma *None-qbs*[qbs]: $\text{None} \in \text{qbs-space } (\text{option-qbs } X)$
by(*simp add*: *option-qbs-space*)

lemma *Some-qbs*[qbs]: $\text{Some} \in X \rightarrow_Q \text{option-qbs } X$
proof –
have $1: \text{Some} = (\lambda x. \text{case } x \text{ of } \text{Inl } y \Rightarrow \text{Some } y \mid \text{Inr } y \Rightarrow \text{None}) \circ \text{Inl}$
by *standard auto*
show *?thesis*
unfolding *option-qbs-def*
by(*rule qbs-morphism-cong'*[*OF - qbs-morphism-comp*[*OF Inl-qbs-morphism map-qbs-morphism-f*]]) (*simp add*: 1)
qed

```

lemma case-option-qbs-morphism[qbs]: case-option ∈ qbs-space ( $Y \Rightarrow_Q (X \Rightarrow_Q Y) \Rightarrow_Q \text{option-qbs } X \Rightarrow_Q Y$ )
proof(rule curry-preserves-morphisms[OF arg-swap-morphism])
  have ( $\lambda x y. \text{case } x \text{ of } \text{None} \Rightarrow \text{fst } y \mid \text{Some } z \Rightarrow \text{snd } y z$ ) = ( $\lambda x y. \text{case } x \text{ of } \text{Inr } - \Rightarrow \text{fst } y \mid \text{Inl } z \Rightarrow \text{snd } y z$ ) ∘ ( $\lambda z. \text{case } z \text{ of } \text{Some } x \Rightarrow \text{Inl } x \mid \text{None} \Rightarrow \text{Inr } ()$ )
    by standard+ (simp add: option.case-eq-if)
  also have ... ∈ option-qbs  $X \rightarrow_Q Y \otimes_Q (X \Rightarrow_Q Y) \Rightarrow_Q Y$ 
    unfolding option-qbs-def by(rule qbs-morphism-comp[OF map-qbs-morphism-inverse-f])
(auto simp: copair-qbs-space)
  finally show ( $\lambda x y. \text{case } x \text{ of } \text{None} \Rightarrow \text{fst } y \mid \text{Some } x \Rightarrow \text{snd } y x$ ) ∈ option-qbs
 $X \rightarrow_Q Y \otimes_Q (X \Rightarrow_Q Y) \Rightarrow_Q Y$  .
qed

```

```

lemma rec-option-qbs-morphism[qbs]: rec-option ∈ qbs-space ( $Y \Rightarrow_Q (X \Rightarrow_Q Y) \Rightarrow_Q \text{option-qbs } X \Rightarrow_Q Y$ )
proof –
  have [simp]: rec-option = case-option
    by standard+ (metis option.case-eq-if option.exhaust-sel option.simps(6) option.simps(7))
  show ?thesis by simp
qed

```

```

lemma bind-option-qbs-morphism[qbs]: ( $\gg=$ ) ∈ qbs-space (option-qbs  $X \Rightarrow_Q (X \Rightarrow_Q \text{option-qbs } Y) \Rightarrow_Q \text{option-qbs } Y$ )
by(simp add: Option.bind-def)

```

```

lemma Let-qbs-morphism[qbs]: Let ∈  $X \Rightarrow_Q (X \Rightarrow_Q Y) \Rightarrow_Q Y$ 
proof –
  have [simp]: Let = ( $\lambda x f. f x$ ) by standard+ auto
  show ?thesis by simp
qed

```

end

3.3 Relation to Measurable Spaces

```

theory Measure-QuasiBorel-Adjunction
  imports QuasiBorel QBS-Morphism Lemmas-S-Finite-Measure-Monad
begin

```

We construct the adjunction between **Meas** and **QBS**, where **Meas** is the category of measurable spaces and measurable functions, and **QBS** is the category of quasi-Borel spaces and morphisms.

3.3.1 The Functor R

```

definition measure-to-qbs :: 'a measure ⇒ 'a quasi-borel where
measure-to-qbs  $X \equiv \text{Abs-quasi-borel } (\text{space } X, \text{borel } \rightarrow_M X)$ 

```

declare [[*coercion measure-to-qbs*]]

lemma

shows *qbs-space-R*: $qbs\text{-space } (measure\text{-to}\text{-}qbs\ X) = space\ X$ (**is** ?goal1)

and *qbs-Mx-R*: $qbs\text{-Mx } (measure\text{-to}\text{-}qbs\ X) = borel\ \rightarrow_M\ X$ (**is** ?goal2)

proof –

have *Rep-quasi-borel* $(measure\text{-to}\text{-}qbs\ X) = (space\ X, borel\ \rightarrow_M\ X)$

by(*auto intro!*: *Abs-quasi-borel-inverse is-quasi-borel-intro qbs-closed1I qbs-closed2I*)

simp: *measure-to-qbs-def dest:measurable-space*) (*rule qbs-closed3I, auto*)

thus ?goal1 ?goal2

by (*simp-all add: qbs-space-def qbs-Mx-def*)

qed

The following lemma says that *measure-to-qbs* is a functor from **Meas** to **QBS**.

lemma *r-preserves-morphisms*:

$X \rightarrow_M Y \subseteq (measure\text{-to}\text{-}qbs\ X) \rightarrow_Q (measure\text{-to}\text{-}qbs\ Y)$

by(*auto intro!*: *qbs-morphismI simp: qbs-Mx-R*)

lemma *measurable-imp-qbs-morphism*: $f \in M \rightarrow_M N \implies f \in M \rightarrow_Q N$

using *r-preserves-morphisms by blast*

3.3.2 The Functor L

definition *sigma-Mx* :: 'a quasi-borel \Rightarrow 'a set set **where**

sigma-Mx $X \equiv \{U \cap qbs\text{-space } X \mid U. \forall \alpha \in qbs\text{-Mx } X. \alpha - ' U \in sets\ borel\}$

definition *qbs-to-measure* :: 'a quasi-borel \Rightarrow 'a measure **where**

qbs-to-measure $X \equiv Abs\text{-measure } (qbs\text{-space } X, sigma\text{-Mx } X, \lambda A. (if\ A = \{\} \text{ then } 0 \text{ else if } A \in -\ sigma\text{-Mx } X \text{ then } 0 \text{ else } \infty))$

lemma *measure-space-L*: $measure\text{-space } (qbs\text{-space } X) (sigma\text{-Mx } X) (\lambda A. (if\ A = \{\} \text{ then } 0 \text{ else if } A \in -\ sigma\text{-Mx } X \text{ then } 0 \text{ else } \infty))$

unfolding *measure-space-def*

proof *safe*

show *sigma-algebra* $(qbs\text{-space } X) (sigma\text{-Mx } X)$

proof(*rule sigma-algebra.intro*)

show *algebra* $(qbs\text{-space } X) (sigma\text{-Mx } X)$

proof

have $\forall U \in sigma\text{-Mx } X. U \subseteq qbs\text{-space } X$

using *sigma-Mx-def subset-iff* **by** *fastforce*

thus $sigma\text{-Mx } X \subseteq Pow\ (qbs\text{-space } X)$ **by** *auto*

next

show $\{\} \in sigma\text{-Mx } X$

unfolding *sigma-Mx-def* **by** *auto*

next

fix A

fix B

```

assume  $A \in \text{sigma-Mx } X$ 
       $B \in \text{sigma-Mx } X$ 
then have  $\exists Ua. A = Ua \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel})$ 
  by (simp add: sigma-Mx-def)
then obtain  $Ua$  where  $pa:A = Ua \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel})$  by auto
have  $\exists Ub. B = Ub \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ub \in \text{sets borel})$ 
  using  $\langle B \in \text{sigma-Mx } X \rangle$  sigma-Mx-def by auto
then obtain  $Ub$  where  $pb:B = Ub \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ub \in \text{sets borel})$  by auto
from  $pa$   $pb$  have [simp]: $\forall \alpha \in \text{qbs-Mx } X. \alpha - ' (Ua \cap Ub) \in \text{sets borel}$ 
  by auto
from this  $pa$   $pb$  sigma-Mx-def have [simp]: $(Ua \cap Ub) \cap \text{qbs-space } X \in \text{sigma-Mx } X$  by blast
from  $pa$   $pb$  have [simp]: $A \cap B = (Ua \cap Ub) \cap \text{qbs-space } X$  by auto
thus  $A \cap B \in \text{sigma-Mx } X$  by simp
next
fix  $A$ 
fix  $B$ 
assume  $A \in \text{sigma-Mx } X$ 
       $B \in \text{sigma-Mx } X$ 
then have  $\exists Ua. A = Ua \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel})$ 
  by (simp add: sigma-Mx-def)
then obtain  $Ua$  where  $pa:A = Ua \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel})$  by auto
have  $\exists Ub. B = Ub \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ub \in \text{sets borel})$ 
  using  $\langle B \in \text{sigma-Mx } X \rangle$  sigma-Mx-def by auto
then obtain  $Ub$  where  $pb:B = Ub \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ub \in \text{sets borel})$  by auto
from  $pa$   $pb$  have [simp]: $A - B = (Ua \cap - Ub) \cap \text{qbs-space } X$  by auto
from  $pa$   $pb$  have  $\forall \alpha \in \text{qbs-Mx } X. \alpha - ' (Ua \cap - Ub) \in \text{sets borel}$ 
  by (metis Diff-Compl double-compl sets.Diff vimage-Compl vimage-Int)
hence  $1:A - B \in \text{sigma-Mx } X$ 
  using sigma-Mx-def  $\langle A - B = Ua \cap - Ub \cap \text{qbs-space } X \rangle$  by blast
show  $\exists C \subseteq \text{sigma-Mx } X. \text{finite } C \wedge \text{disjoint } C \wedge A - B = \bigcup C$ 
proof
  show  $\{A - B\} \subseteq \text{sigma-Mx } X \wedge \text{finite } \{A - B\} \wedge \text{disjoint } \{A - B\} \wedge A - B = \bigcup \{A - B\}$ 
  using  $1$  by auto
qed
next
fix  $A$ 
fix  $B$ 
assume  $A \in \text{sigma-Mx } X$ 
       $B \in \text{sigma-Mx } X$ 
then have  $\exists Ua. A = Ua \cap \text{qbs-space } X \wedge (\forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel})$ 

```



```

    by (simp add: sigma-Mx-def)
    then obtain  $Ua$  where  $pa:A = Ua \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' Ua \in sets borel)$  by auto
    have  $\exists Ub. B = Ub \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' Ub \in sets borel)$ 
      using  $\langle B \in sigma-Mx X \rangle sigma-Mx-def$  by auto
    then obtain  $Ub$  where  $pb:B = Ub \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' Ub \in sets borel)$  by auto
    from  $pa pb$  have  $A \cup B = (Ua \cup Ub) \cap qbs-space X$  by auto
    from  $pa pb$  have  $\forall \alpha \in qbs-Mx X. \alpha - '(Ua \cup Ub) \in sets borel$  by auto
    then show  $A \cup B \in sigma-Mx X$ 
      unfolding sigma-Mx-def
      using  $\langle A \cup B = (Ua \cup Ub) \cap qbs-space X \rangle$  by blast
  next
  have  $\forall \alpha \in qbs-Mx X. \alpha - '(UNIV) \in sets borel$ 
    by simp
  thus  $qbs-space X \in sigma-Mx X$ 
    unfolding sigma-Mx-def
    by blast
qed
next
show sigma-algebra-axioms (sigma-Mx X)
  unfolding sigma-algebra-axioms-def
proof safe
  fix  $A :: nat \Rightarrow -$ 
  assume  $1: range A \subseteq sigma-Mx X$ 
  then have  $2: \forall i. \exists Ui. A i = Ui \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' Ui \in sets borel)$ 
    unfolding sigma-Mx-def by auto
  then have  $\exists U :: nat \Rightarrow -. \forall i. A i = U i \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' (U i) \in sets borel)$ 
    by (rule choice)
  from this obtain  $U$  where  $pu: \forall i. A i = U i \cap qbs-space X \wedge (\forall \alpha \in qbs-Mx X. \alpha - ' (U i) \in sets borel)$ 
    by auto
  hence  $\forall \alpha \in qbs-Mx X. \alpha - ' (\bigcup (range U)) \in sets borel$ 
    by (simp add: countable-Un-Int(1) vimage-UN)
  from  $pu$  have  $\bigcup (range A) = (\bigcup i::nat. (U i \cap qbs-space X))$  by blast
  hence  $\bigcup (range A) = \bigcup (range U) \cap qbs-space X$  by auto
  thus  $\bigcup (range A) \in sigma-Mx X$ 
    using sigma-Mx-def  $\langle \forall \alpha \in qbs-Mx X. \alpha - ' \bigcup (range U) \in sets borel \rangle$  by
blast
qed
qed
next
show countably-additive (sigma-Mx X) ( $\lambda A. if A = \{\} then 0 else if A \in - sigma-Mx X then 0 else \infty$ )
proof (rule countably-additiveI)
  fix  $A :: nat \Rightarrow -$ 
  assume  $h: range A \subseteq sigma-Mx X$ 

```

```

       $\bigcup (\text{range } A) \in \text{sigma-Mx } X$ 
consider  $\bigcup (\text{range } A) = \{\} \mid \bigcup (\text{range } A) \neq \{\}$ 
  by auto
then show  $(\sum i. \text{if } A \ i = \{\} \text{ then } 0 \text{ else if } A \ i \in - \text{sigma-Mx } X \text{ then } 0 \text{ else } \infty) =$ 
   $(\text{if } \bigcup (\text{range } A) = \{\} \text{ then } 0 \text{ else if } \bigcup (\text{range } A) \in - \text{sigma-Mx } X$ 
   $\text{ then } 0 \text{ else } (\infty :: \text{ennreal}))$ 
proof cases
  case 1
    then have  $\bigwedge i. A \ i = \{\}$ 
    by simp
    thus ?thesis
    by(simp add: 1)
  next
  case 2
    then obtain  $j \text{ where } hj:A \ j \neq \{\}$ 
    by auto
    have  $(\sum i. \text{if } A \ i = \{\} \text{ then } 0 \text{ else if } A \ i \in - \text{sigma-Mx } X \text{ then } 0 \text{ else } \infty) =$ 
     $(\infty :: \text{ennreal})$ 
    proof -
      have  $hsum:\bigwedge N f. \text{sum } f \ \{..<N\} \leq (\sum n. (f \ n :: \text{ennreal}))$ 
      by (simp add: sum-le-suminf)
      have  $hsum':\bigwedge P f. (\exists j. j \in P \wedge f \ j = (\infty :: \text{ennreal})) \implies \text{finite } P \implies \text{sum}$ 
       $f \ P = \infty$ 
      by auto
      have  $h1:(\sum i<j+1. \text{if } A \ i = \{\} \text{ then } 0 \text{ else if } A \ i \in - \text{sigma-Mx } X \text{ then } 0$ 
       $\text{ else } \infty) = (\infty :: \text{ennreal})$ 
      proof(rule hsum')
        show  $\exists ja. ja \in \{..<j + 1\} \wedge (\text{if } A \ ja = \{\} \text{ then } 0 \text{ else if } A \ ja \in -$ 
         $\text{sigma-Mx } X \text{ then } 0 \text{ else } \infty) = (\infty :: \text{ennreal})$ 
        proof(rule exI[where x=j],rule conjI)
          have  $A \ j \in \text{sigma-Mx } X$ 
          using  $h(1)$  by auto
          then show  $(\text{if } A \ j = \{\} \text{ then } 0 \text{ else if } A \ j \in - \text{sigma-Mx } X \text{ then } 0 \text{ else } \infty) = (\infty :: \text{ennreal})$ 
          using  $hj$  by simp
        qed simp
      qed simp
      have  $(\sum i<j+1. \text{if } A \ i = \{\} \text{ then } 0 \text{ else if } A \ i \in - \text{sigma-Mx } X \text{ then } 0$ 
       $\text{ else } \infty) \leq (\sum i. \text{if } A \ i = \{\} \text{ then } 0 \text{ else if } A \ i \in - \text{sigma-Mx } X \text{ then } 0 \text{ else } (\infty ::$ 
       $\text{ennreal}))$ 
      by(rule hsum)
      thus ?thesis
      by(simp only: h1) (simp add: top.extremum-unique)
    qed
    moreover have  $(\text{if } \bigcup (\text{range } A) = \{\} \text{ then } 0 \text{ else if } \bigcup (\text{range } A) \in -$ 
     $\text{sigma-Mx } X \text{ then } 0 \text{ else } \infty) = (\infty :: \text{ennreal})$ 
    using 2  $h(2)$  by simp
    ultimately show ?thesis

```

by simp
 qed
 qed
 qed(simp add: positive-def)

lemma

shows space-L: space (qbs-to-measure X) = qbs-space X (is ?goal1)
 and sets-L: sets (qbs-to-measure X) = sigma-Mx X (is ?goal2)
 and emeasure-L: emeasure (qbs-to-measure X) = ($\lambda A. \text{if } A = \{\} \vee A \notin \text{sigma-Mx } X \text{ then } 0 \text{ else } \infty$) (is ?goal3)
 proof –
 have Rep-measure (qbs-to-measure X) = (qbs-space X, sigma-Mx X, $\lambda A. (\text{if } A = \{\} \text{ then } 0 \text{ else if } A \in - \text{sigma-Mx } X \text{ then } 0 \text{ else } \infty)$)
 unfolding qbs-to-measure-def by(auto intro!: Abs-measure-inverse simp: measure-space-L)
 thus ?goal1 ?goal2 ?goal3
 by(auto simp: sets-def space-def emeasure-def)
 qed

lemma qbs-Mx-sigma-Mx-contr:

assumes qbs-space X = qbs-space Y
 and qbs-Mx X \subseteq qbs-Mx Y
 shows sigma-Mx Y \subseteq sigma-Mx X
 using assms by(auto simp: sigma-Mx-def)

The following lemma says that *qbs-to-measure* is a functor from **QBS** to **Meas**.

lemma l-preserves-morphisms:

$X \rightarrow_Q Y \subseteq (qbs-to-measure X) \rightarrow_M (qbs-to-measure Y)$
 proof safe
 fix f
 assume h: f $\in X \rightarrow_Q Y$
 show f $\in (qbs-to-measure X) \rightarrow_M (qbs-to-measure Y)$
 proof(rule measurableI)
 fix A
 assume A $\in \text{sets } (qbs-to-measure Y)$
 then obtain Ua where pa: A = Ua \cap qbs-space Y $\wedge (\forall \alpha \in \text{qbs-Mx } Y. \alpha -' Ua \in \text{sets borel})$
 by (auto simp: sigma-Mx-def sets-L)
 have $\forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y$
 $\forall \alpha \in \text{qbs-Mx } X. \alpha -' (f -' (qbs-space Y)) = UNIV$
 using qbs-morphism-space[OF h] qbs-morphism-Mx[OF h] by (auto simp: qbs-Mx-to-X)
 hence $\forall \alpha \in \text{qbs-Mx } X. \alpha -' (f -' A) = \alpha -' (f -' Ua)$
 by (simp add: pa)
 from pa this qbs-morphism-def have $\forall \alpha \in \text{qbs-Mx } X. \alpha -' (f -' A) \in \text{sets borel}$
 by (simp add: vimage-comp $\langle \forall \alpha \in \text{qbs-Mx } X. f \circ \alpha \in \text{qbs-Mx } Y \rangle$)
 thus $f -' A \cap \text{space } (qbs-to-measure X) \in \text{sets } (qbs-to-measure X)$
 using sigma-Mx-def by(auto simp: sets-L space-L)

qed (*insert qbs-morphism-space*[*OF h*], *auto simp: space-L*)
qed

lemma *qbs-morphism-imp-measurable*: $f \in X \rightarrow_Q Y \implies f \in \text{qbs-to-measure } X \rightarrow_M \text{qbs-to-measure } Y$
using *l-preserves-morphisms by blast*

abbreviation *qbs-borel* (*borel_Q*) **where** *borel_Q* \equiv *measure-to-qbs borel*
abbreviation *qbs-count-space* (*count'-space_Q*) **where** *qbs-count-space I* \equiv *measure-to-qbs (count-space I)*

lemma
shows *qbs-space-qbs-borel*[*simp*]: *qbs-space borel_Q* = *UNIV*
and *qbs-space-count-space*[*simp*]: *qbs-space (qbs-count-space I)* = *I*
and *qbs-Mx-qbs-borel*: *qbs-Mx borel_Q* = *borel-measurable borel*
and *qbs-Mx-count-space*: *qbs-Mx (qbs-count-space I)* = *borel* \rightarrow_M *count-space I*
by(*simp-all add: qbs-space-R qbs-Mx-R*)

lemma
shows *qbs-space-qbs-borel'*[*qbs*]: $r \in \text{qbs-space borel}_Q$
and *qbs-space-count-space-UNIV'*[*qbs*]: $x \in \text{qbs-space (qbs-count-space (UNIV :: (- :: countable) set))}$
by *simp-all*

lemma *qbs-Mx-is-morphisms*: $\text{qbs-Mx } X = \text{borel}_Q \rightarrow_Q X$

proof *safe*

fix $\alpha :: \text{real} \Rightarrow -$
assume $\alpha \in \text{borel}_Q \rightarrow_Q X$
have $id \in \text{qbs-Mx borel}_Q$ **by** (*simp add: qbs-Mx-R*)
then have $\alpha \circ id \in \text{qbs-Mx } X$
using *qbs-morphism-Mx*[*OF* $\langle \alpha \in \text{borel}_Q \rightarrow_Q X \rangle$]
by *blast*
then show $\alpha \in \text{qbs-Mx } X$ **by** *simp*

qed(*auto intro!: qbs-morphismI simp: qbs-Mx-qbs-borel*)

lemma *exp-qbs-Mx'*: $\text{qbs-Mx (exp-qbs } X \ Y) = \{g. \text{case-prod } g \in \text{borel}_Q \otimes_Q X \rightarrow_Q Y\}$

by(*auto simp: qbs-Mx-qbs-borel comp-def qbs-Mx-is-morphisms split-beta' intro!: curry-preserves-morphisms*)

lemma *arg-swap-morphism'*:

assumes $(\lambda g. f (\lambda w x. g x w)) \in \text{exp-qbs } X (\text{exp-qbs } W \ Y) \rightarrow_Q Z$
shows $f \in \text{exp-qbs } W (\text{exp-qbs } X \ Y) \rightarrow_Q Z$

proof(*rule qbs-morphismI*)

fix α
assume $\alpha \in \text{qbs-Mx (exp-qbs } W (\text{exp-qbs } X \ Y))$
then have $(\lambda((r,w),x). \alpha r w x) \in (\text{borel}_Q \otimes_Q W) \otimes_Q X \rightarrow_Q Y$
by(*auto simp: qbs-Mx-is-morphisms dest: uncurry-preserves-morphisms*)

hence $(\lambda(r,w,x). \alpha r w x) \in \text{borel}_Q \otimes_Q W \otimes_Q X \rightarrow_Q Y$
by *(auto intro! : qbs-morphism-cong'*[**where** $f = (\lambda((r,w),x). \alpha r w x) \circ (\lambda(x, y, z). ((x, y), z))$ **and** $g = \lambda(r,w,x). \alpha r w x$]*OF qbs-morphism-pair-assoc2*)]
hence $(\lambda(r,x,w). \alpha r w x) \in \text{borel}_Q \otimes_Q X \otimes_Q W \rightarrow_Q Y$
by *(auto intro! : qbs-morphism-cong'*[**where** $f = (\lambda(r,w,x). \alpha r w x) \circ \text{map-prod id } (\lambda(x,y). (y,x))$ **and** $g = (\lambda(r,x,w). \alpha r w x)$] *qbs-morphism-comp qbs-morphism-map-prod qbs-morphism-pair-swap*)]
hence $(\lambda((r,x),w). \alpha r w x) \in (\text{borel}_Q \otimes_Q X) \otimes_Q W \rightarrow_Q Y$
by *(auto intro! : qbs-morphism-cong'*[**where** $f = (\lambda(r,x,w). \alpha r w x) \circ (\lambda((x, y), z). (x, y, z))$ **and** $g = \lambda((r,x),w). \alpha r w x$]*OF qbs-morphism-pair-assoc1*)]
hence $(\lambda r x w. \alpha r w x) \in \text{qbs-Mx } (\text{exp-qbs } X \ (\text{exp-qbs } W \ Y))$
by *(auto simp: qbs-Mx-is-morphisms split-beta')*
from *qbs-morphism-Mx*[*OF assms this*] **show** $f \circ \alpha \in \text{qbs-Mx } Z$
by *(auto simp: comp-def)*
qed

lemma *qbs-Mx-subset-of-measurable*: $\text{qbs-Mx } X \subseteq \text{borel} \rightarrow_M \text{qbs-to-measure } X$

proof

fix α

assume $\alpha \in \text{qbs-Mx } X$

show $\alpha \in \text{borel} \rightarrow_M \text{qbs-to-measure } X$

proof *(rule measurableI)*

fix x

show $\alpha x \in \text{space } (\text{qbs-to-measure } X)$

using *qbs-Mx-to-X* $\langle \alpha \in \text{qbs-Mx } X \rangle$ **by** *(simp add: space-L)*

next

fix A

assume $A \in \text{sets } (\text{qbs-to-measure } X)$

then have $\alpha - \langle (\text{qbs-space } X) = \text{UNIV} \rangle$

using $\langle \alpha \in \text{qbs-Mx } X \rangle$ *qbs-Mx-to-X* **by** *(auto simp: sets-L)*

then show $\alpha - \langle A \cap \text{space borel} \in \text{sets borel} \rangle$

using $\langle \alpha \in \text{qbs-Mx } X \rangle$ $\langle A \in \text{sets } (\text{qbs-to-measure } X) \rangle$

by *(auto simp add: sigma-Mx-def sets-L)*

qed

qed

lemma *L-max-of-measurables*:

assumes $\text{space } M = \text{qbs-space } X$

and $\text{qbs-Mx } X \subseteq \text{borel} \rightarrow_M M$

shows $\text{sets } M \subseteq \text{sets } (\text{qbs-to-measure } X)$

proof

fix U

assume $U \in \text{sets } M$

from *sets.sets-into-space*[*OF this*] *in-mono*[*OF assms(2)*] *measurable-sets-borel*[*OF - this*]

show $U \in \text{sets } (\text{qbs-to-measure } X)$

using *assms(1)*

by *(auto intro! : exI*[**where** $x = U$] *simp: sigma-Mx-def sets-L)*

qed

```

lemma qbs-Mx-are-measurable[simp, measurable]:
  assumes  $\alpha \in \text{qbs-Mx } X$ 
  shows  $\alpha \in \text{borel} \rightarrow_M \text{qbs-to-measure } X$ 
  using assms qbs-Mx-subset-of-measurable by auto

lemma measure-to-qbs-cong-sets:
  assumes sets  $M = \text{sets } N$ 
  shows measure-to-qbs  $M = \text{measure-to-qbs } N$ 
  by(rule qbs-eqI) (simp add: qbs-Mx-R measurable-cong-sets[OF - assms])

lemma lr-sets[simp]:
  sets  $X \subseteq \text{sets} (\text{qbs-to-measure} (\text{measure-to-qbs } X))$ 
  unfolding sets-L
proof safe
  fix  $U$ 
  assume  $U \in \text{sets } X$ 
  then have  $U \cap \text{space } X = U$  by simp
  moreover have  $\forall \alpha \in \text{borel} \rightarrow_M X. \alpha - ' U \in \text{sets borel}$ 
    using  $\langle U \in \text{sets } X \rangle$  by(auto simp add: measurable-def)
  ultimately show  $U \in \text{sigma-Mx} (\text{measure-to-qbs } X)$ 
    by(auto simp add: sigma-Mx-def qbs-Mx-R qbs-space-R)
qed

lemma(in standard-borel) lr-sets-ident[simp, measurable-cong]:
  sets  $(\text{qbs-to-measure} (\text{measure-to-qbs } M)) = \text{sets } M$ 
  unfolding sets-L
proof safe
  fix  $V$ 
  assume  $V \in \text{sigma-Mx} (\text{measure-to-qbs } M)$ 
  then obtain  $U$  where  $H2: V = U \cap \text{space } M \wedge \alpha :: \text{real} \Rightarrow -. \alpha \in \text{borel} \rightarrow_M M$ 
   $\Rightarrow \alpha - ' U \in \text{sets borel}$ 
    by(auto simp: sigma-Mx-def qbs-Mx-R qbs-space-R)
  consider  $\text{space } M = \{\}$  |  $\text{space } M \neq \{\}$  by auto
  then show  $V \in \text{sets } M$ 
  proof cases
    case 1
      then show ?thesis
        by(simp add: H2)
    next
      case 2
        have from-real  $- ' V = \text{from-real} - ' (U \cap \text{space } M)$  using  $H2$  by auto
        also have  $\dots = \text{from-real} - ' U$  using from-real-measurable'[OF 2] by(auto simp add: measurable-def)
        finally have to-real  $- ' \text{from-real} - ' U \cap \text{space } M \in \text{sets } M$ 
          by (meson 2 H2(2) from-real-measurable' measurable-sets to-real-measurable)
        moreover have to-real  $- ' \text{from-real} - ' U \cap \text{space } M = U \cap \text{space } M$ 
          by auto
        ultimately show ?thesis using  $H2$  by simp
  qed

```

qed
qed(*insert lr-sets, auto simp: sets-L*)

corollary *sets-lr-polish-borel*[*simp, measurable-cong*]: *sets (qbs-to-measure qbs-borel)*
= *sets (borel :: (- :: polish-space) measure)*
by(*auto intro!: standard-borel.lr-sets-ident standard-borel-ne.standard-borel*)

corollary *sets-lr-count-space*[*simp, measurable-cong*]: *sets (qbs-to-measure (qbs-count-space (UNIV :: (- :: countable) set))) = sets (count-space UNIV)*
by(*rule standard-borel.lr-sets-ident*) (*auto intro!: standard-borel-ne.standard-borel*)

lemma *map-qbs-embed-measure1*:
assumes *inj-on f (space M)*
shows *map-qbs f (measure-to-qbs M) = measure-to-qbs (embed-measure M f)*
proof(*safe intro!: qbs-eqI*)
fix α
show $\alpha \in \text{qbs-Mx } (\text{map-qbs } f \text{ (measure-to-qbs } M)) \implies \alpha \in \text{qbs-Mx } (\text{measure-to-qbs } (\text{embed-measure } M f))$
using *measurable-embed-measure2*'[*OF assms*] **by**(*auto intro!: qbs-eqI simp: map-qbs-Mx qbs-Mx-R*)
next
fix α
assume $\alpha \in \text{qbs-Mx } (\text{measure-to-qbs } (\text{embed-measure } M f))$
hence $h[\text{measurable}]: \alpha \in \text{borel} \rightarrow_M \text{embed-measure } M f$
by(*simp add: qbs-Mx-R*)
have [*measurable*]:*the-inv-into (space M) f* $\in \text{embed-measure } M f \rightarrow_M M$
by (*simp add: assms in-sets-embed-measure measurable-def sets.sets-into-space space-embed-measure the-inv-into-into the-inv-into-vimage*)
show $\alpha \in \text{qbs-Mx } (\text{map-qbs } f \text{ (measure-to-qbs } M))$
using *measurable-space*[*OF h*] *f-the-inv-into-f*[*OF assms*] **by**(*auto intro!: exI[where x=the-inv-into (space M) f o alpha] simp: map-qbs-Mx qbs-Mx-R space-embed-measure*)
qed

lemma *map-qbs-embed-measure2*:
assumes *inj-on f (qbs-space X)*
shows *sets (qbs-to-measure (map-qbs f X)) = sets (embed-measure (qbs-to-measure X) f)*
proof *safe*
fix A
assume $A \in \text{sets } (\text{qbs-to-measure } (\text{map-qbs } f X))$
then obtain U **where** $A = U \cap f^{-1} \text{qbs-space } X \wedge \alpha. \alpha \in \text{qbs-Mx } X \implies (f \circ \alpha)^{-1} U \in \text{sets borel}$
by(*auto simp: sets-L sigma-Mx-def map-qbs-space map-qbs-Mx*)
thus $A \in \text{sets } (\text{embed-measure } (\text{qbs-to-measure } X) f)$
by(*auto simp: sets-L sets-embed-measure'[of - qbs-to-measure X, simplified space-L, OF assms] sigma-Mx-def qbs-Mx-to-X vimage-def intro!: exI[where x=f^{-1} U \cap qbs-space X]*)
next
fix A

```

assume A:A ∈ sets (embed-measure (qbs-to-measure X) f)
then obtain B U where A = f ′ B B = U ∩ qbs-space X ∧ α. α ∈ qbs-Mx X
⇒ α - ′ U ∈ sets borel
by(auto simp: sets-L sigma-Mx-def sets-embed-measure ′[of - qbs-to-measure
X,simplified space-L,OF assms])
thus A ∈ sets (qbs-to-measure (map-qbs f X))
using measurable-sets-borel[OF measurable-comp[OF qbs-Mx-are-measurable
measurable-embed-measure2 ′[of - qbs-to-measure X,simplified space-L,OF assms]]]
A
by(auto simp: sets-L sigma-Mx-def map-qbs-space map-qbs-Mx intro!: exI[where
x=f ′ (U ∩ qbs-space X)])
qed

```

```

lemma(in standard-borel) map-qbs-embed-measure2 ′:
assumes inj-on f (space M)
shows sets (qbs-to-measure (map-qbs f (measure-to-qbs M))) = sets (embed-measure
M f)
by(auto intro!: standard-borel.lr-sets-ident[OF standard-borel-embed-measure] assms
simp: map-qbs-embed-measure1[OF assms])

```

3.3.3 The Adjunction

```

lemma lr-adjunction-correspondence :
X →Q (measure-to-qbs Y) = (qbs-to-measure X) →M Y
proof safe

```

```

fix f
assume f ∈ X →Q (measure-to-qbs Y)
show f ∈ qbs-to-measure X →M Y
proof(rule measurableI)
fix x
assume x ∈ space (qbs-to-measure X)
thus f x ∈ space Y
using qbs-morphism-space[OF ‹f ∈ X →Q (measure-to-qbs Y)›]
by (auto simp: qbs-space-R space-L)
next
fix A
assume A ∈ sets Y
have ∀ α ∈ qbs-Mx X. f ∘ α ∈ qbs-Mx (measure-to-qbs Y)
using qbs-morphism-Mx[OF ‹f ∈ X →Q (measure-to-qbs Y)›] by auto
hence ∀ α ∈ qbs-Mx X. f ∘ α ∈ borel →M Y by (simp add: qbs-Mx-R)
hence ∀ α ∈ qbs-Mx X. α - ′ (f - ′ A) ∈ sets borel
using ‹A ∈ sets Y› measurable-sets-borel vimage-comp by metis
thus f - ′ A ∩ space (qbs-to-measure X) ∈ sets (qbs-to-measure X)
using sigma-Mx-def by (auto simp: space-L sets-L)
qed

```

```

next

```



```

fix f
assume f ∈ qbs-to-measure X →M Y
show f ∈ X →Q measure-to-qbs Y
proof(rule qbs-morphismI)
  fix α
  assume α ∈ qbs-Mx X
  have f ∘ α ∈ borel →M Y
  proof(rule measurableI)
    fix x :: real
    from ⟨α ∈ qbs-Mx X⟩ qbs-Mx-to-X have α x ∈ qbs-space X by auto
    hence α x ∈ space (qbs-to-measure X) by (simp add: space-L)
    thus (f ∘ α) x ∈ space Y
    using ⟨f ∈ qbs-to-measure X →M Y⟩
    by (metis comp-def measurable-space)
  next
  fix A
  assume A ∈ sets Y
  from ⟨f ∈ qbs-to-measure X →M Y⟩ measurable-sets this measurable-def
  have f -‘ A ∩ space (qbs-to-measure X) ∈ sets (qbs-to-measure X)
    by blast
  hence f -‘ A ∩ qbs-space X ∈ sigma-Mx X by (simp add: sets-L space-L)
  then have ∃ V. f -‘ A ∩ qbs-space X = V ∩ qbs-space X ∧ (∀ β ∈ qbs-Mx
X. β -‘ V ∈ sets borel)
    by (simp add: sigma-Mx-def)
  then obtain V where h: f -‘ A ∩ qbs-space X = V ∩ qbs-space X ∧ (∀ β ∈
qbs-Mx X. β -‘ V ∈ sets borel) by auto
  have 1: α -‘ (f -‘ A) = α -‘ (f -‘ A ∩ qbs-space X)
    using ⟨α ∈ qbs-Mx X⟩ qbs-Mx-to-X by blast
  have 2: α -‘ (V ∩ qbs-space X) = α -‘ V
    using ⟨α ∈ qbs-Mx X⟩ qbs-Mx-to-X by blast
  from 1 2 h have (f ∘ α) -‘ A = α -‘ V by (simp add: vimage-comp)
  from this h ⟨α ∈ qbs-Mx X⟩ show (f ∘ α) -‘ A ∩ space borel ∈ sets borel by
simp
  qed
  thus f ∘ α ∈ qbs-Mx (measure-to-qbs Y)
    by(simp add: qbs-Mx-R)
  qed
qed

lemma(in standard-borel) standard-borel-r-full-faithful:
  M →M Y = measure-to-qbs M →Q measure-to-qbs Y
proof
  have measure-to-qbs M →Q measure-to-qbs Y ⊆ qbs-to-measure (measure-to-qbs
M) →M qbs-to-measure (measure-to-qbs Y)
    by (simp add: l-preserves-morphisms)
  also have ... = M →M qbs-to-measure (measure-to-qbs Y)
    using measurable-cong-sets by auto
  also have ... ⊆ M →M Y
    by(rule measurable-mono[OF lr-sets]) (simp-all add: qbs-space-R space-L)

```

finally show $\text{measure-to-qbs } M \rightarrow_Q \text{ measure-to-qbs } Y \subseteq M \rightarrow_M Y$.
qed(*rule r-preserves-morphisms*)

lemma *qbs-morphism-dest*:
assumes $f \in X \rightarrow_Q \text{ measure-to-qbs } Y$
shows $f \in \text{qbs-to-measure } X \rightarrow_M Y$
using *assms lr-adjunction-correspondence* **by auto**

lemma(*in standard-borel*) *qbs-morphism-dest*:
assumes $k \in \text{measure-to-qbs } M \rightarrow_Q \text{ measure-to-qbs } Y$
shows $k \in M \rightarrow_M Y$
using *standard-borel-r-full-faithful assms* **by auto**

lemma *qbs-morphism-measurable-intro*:
assumes $f \in \text{qbs-to-measure } X \rightarrow_M Y$
shows $f \in X \rightarrow_Q \text{ measure-to-qbs } Y$
using *assms lr-adjunction-correspondence* **by auto**

lemma(*in standard-borel*) *qbs-morphism-measurable-intro*:
assumes $k \in M \rightarrow_M Y$
shows $k \in \text{measure-to-qbs } M \rightarrow_Q \text{ measure-to-qbs } Y$
using *standard-borel-r-full-faithful assms* **by auto**

lemma *r-preserves-product* :
 $\text{measure-to-qbs } (X \otimes_M Y) = \text{measure-to-qbs } X \otimes_Q \text{ measure-to-qbs } Y$
by(*auto intro!*: *qbs-eqI simp: measurable-pair-iff pair-qbs-Mx qbs-Mx-R*)

lemma *l-product-sets*:
 $\text{sets } (\text{qbs-to-measure } X \otimes_M \text{ qbs-to-measure } Y) \subseteq \text{sets } (\text{qbs-to-measure } (X \otimes_Q Y))$

proof(*rule sets-pair-in-sets*)

fix $A B$

assume $h:A \in \text{sets } (\text{qbs-to-measure } X) B \in \text{sets } (\text{qbs-to-measure } Y)$

then obtain $Ua Ub$ **where** hu :

$A = Ua \cap \text{qbs-space } X \forall \alpha \in \text{qbs-Mx } X. \alpha - ' Ua \in \text{sets borel}$

$B = Ub \cap \text{qbs-space } Y \forall \alpha \in \text{qbs-Mx } Y. \alpha - ' Ub \in \text{sets borel}$

by(*auto simp add: sigma-Mx-def sets-L*)

show $A \times B \in \text{sets } (\text{qbs-to-measure } (X \otimes_Q Y))$

proof –

have $A \times B = Ua \times Ub \cap \text{qbs-space } (X \otimes_Q Y) \wedge (\forall \alpha \in \text{qbs-Mx } (X \otimes_Q Y).$

$\alpha - ' (Ua \times Ub) \in \text{sets borel}$)

using hu **by**(*auto simp add: vimage-Times pair-qbs-space pair-qbs-Mx*)

thus *?thesis*

by(*auto simp add: sigma-Mx-def sets-L intro!: exI[where x=Ua × Ub]*)

qed

qed

corollary *qbs-borel-prod*: $\text{qbs-borel } \otimes_Q \text{ qbs-borel} = (\text{qbs-borel} :: ('a::\text{second-countable-topology} \times 'b::\text{second-countable-topology}) \text{ quasi-borel})$

```

by(simp add: r-preserves-product[symmetric] borel-prod)

corollary qbs-count-space-prod: qbs-count-space (UNIV :: ('a :: countable) set)
⊗Q qbs-count-space (UNIV :: ('b :: countable) set) = qbs-count-space UNIV
  by(auto simp: r-preserves-product[symmetric] count-space-prod)

lemma r-preserves-product': measure-to-qbs (ΠM i∈I. M i) = (ΠQ i∈I. mea-
sure-to-qbs (M i))
proof(rule qbs-eqI)
  show qbs-Mx (measure-to-qbs (PiM I M)) = qbs-Mx (ΠQ i∈I. measure-to-qbs
(M i))
  proof safe
    fix f :: real ⇒ -
    assume f ∈ qbs-Mx (measure-to-qbs (PiM I M))
    with measurable-space[of f borel PiM I M] show f ∈ qbs-Mx (ΠQ i∈I. mea-
sure-to-qbs (M i))
    by(auto simp: qbs-Mx-R PiQ-Mx space-PiM intro!:ext[of λr. f r -])
  next
    fix f :: real ⇒ -
    assume f ∈ qbs-Mx (ΠQ i∈I. measure-to-qbs (M i))
    then have ∧i. i ∈ I ⇒ (λr. f r i) ∈ borel →M M i ∧i. i ∉ I ⇒ (λr. f r i)
= (λr. undefined)
    by (auto simp: qbs-Mx-R PiQ-Mx)
    with measurable-space[OF this(1)] fun-cong[OF this(2)] show f ∈ qbs-Mx
(measure-to-qbs (PiM I M))
    by(auto intro!: measurable-PiM-single' simp: qbs-Mx-R)
  qed
qed

lemma PiQ-qbs-borel:
(ΠQ i::('a:: countable)∈UNIV. (qbs-borel :: ('b::second-countable-topology quasi-borel)))
= qbs-borel
  by(simp add: r-preserves-product'[symmetric] measure-to-qbs-cong-sets[OF sets-PiM-equal-borel])

lemma qbs-morphism-from-countable:
fixes X :: 'a quasi-borel
assumes countable (qbs-space X)
      qbs-Mx X ⊆ borel →M count-space (qbs-space X)
      and ∧i. i ∈ qbs-space X ⇒ f i ∈ qbs-space Y
      shows f ∈ X →Q Y
proof(rule qbs-morphismI)
  fix α
  assume α ∈ qbs-Mx X
  then have [measurable]: α ∈ borel →M count-space (qbs-space X)
    using assms(2) ..
  define k :: 'a ⇒ real ⇒ -
    where k ≡ (λi -. f i)
  have f ∘ α = (λr. k (α r) r)
    by(auto simp add: k-def)

```

also have $\dots \in \text{qbs-Mx } Y$
 by(rule qbs-closed3-dest2[OF assms(1)]) (use assms(3) k-def in simp-all)
 finally show $f \circ \alpha \in \text{qbs-Mx } Y$.
 qed

corollary *qbs-morphism-count-space'*:
 assumes $\bigwedge i. i \in I \implies f i \in \text{qbs-space } Y$ countable I
 shows $f \in \text{qbs-count-space } I \rightarrow_Q Y$
 using assms by(auto intro!: qbs-morphism-from-countable simp: qbs-Mx-R)

corollary *qbs-morphism-count-space*:
 assumes $\bigwedge i. f i \in \text{qbs-space } Y$
 shows $f \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q Y$
 using assms by(auto intro!: qbs-morphism-from-countable simp: qbs-Mx-R)

lemma [*qbs*]:
 shows *not-qbs-pred*: $\text{Not} \in \text{qbs-count-space } UNIV \rightarrow_Q \text{qbs-count-space } UNIV$
 and *or-qbs-pred*: $(\vee) \in \text{qbs-count-space } UNIV \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *and-qbs-pred*: $(\wedge) \in \text{qbs-count-space } UNIV \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *implies-qbs-pred*: $(\longrightarrow) \in \text{qbs-count-space } UNIV \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *iff-qbs-pred*: $(\longleftrightarrow) \in \text{qbs-count-space } UNIV \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 by(auto intro!: qbs-morphism-count-space)

lemma [*qbs*]:
 shows *less-count-qbs-pred*: $(<) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *le-count-qbs-pred*: $(\leq) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *eq-count-qbs-pred*: $(=) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *plus-count-qbs-morphism*: $(+) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *minus-count-qbs-morphism*: $(-) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *mult-count-qbs-morphism*: $(*) \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{exp-qbs } (\text{qbs-count-space } UNIV)$
 and *Suc-qbs-morphism*: $\text{Suc} \in \text{qbs-count-space } UNIV \rightarrow_Q \text{qbs-count-space } UNIV$
 by(auto intro!: qbs-morphism-count-space)

lemma *qbs-morphism-product-iff*:
 $f \in X \rightarrow_Q (\prod_Q i :: (- :: \text{countable}) \in UNIV. Y) \iff f \in X \rightarrow_Q \text{qbs-count-space } UNIV \Rightarrow_Q Y$

proof
 assume $h: f \in X \rightarrow_Q (\prod_Q i \in UNIV. Y)$
 show $f \in X \rightarrow_Q \text{qbs-count-space } UNIV \Rightarrow_Q Y$

by(*rule arg-swap-morphism, rule qbs-morphism-count-space*) (*simp add: qbs-morphism-component-singleton h qbs-morphism-ident*)

next

assume $f \in X \rightarrow_Q \text{qbs-count-space } UNIV \Rightarrow_Q Y$
from *qbs-morphism-space*[*OF arg-swap-morphism*[*OF this*]]
show $f \in X \rightarrow_Q (\prod_Q i \in UNIV. Y)$
by(*auto intro!: product-qbs-canonical1* [**where** $f = (\lambda i x. f x i)$])

qed

lemma *qbs-morphism-pair-countable1*:

assumes *countable* (*qbs-space* X)
 $\text{qbs-Mx } X \subseteq \text{borel} \rightarrow_M \text{count-space } (\text{qbs-space } X)$
and $\bigwedge i. i \in \text{qbs-space } X \implies f i \in Y \rightarrow_Q Z$
shows $(\lambda(x,y). f x y) \in X \otimes_Q Y \rightarrow_Q Z$
by(*auto intro!: uncurry-preserves-morphisms qbs-morphism-from-countable*[*OF assms(1,2)*] *assms(3)*)

lemma *qbs-morphism-pair-countable2*:

assumes *countable* (*qbs-space* Y)
 $\text{qbs-Mx } Y \subseteq \text{borel} \rightarrow_M \text{count-space } (\text{qbs-space } Y)$
and $\bigwedge i. i \in \text{qbs-space } Y \implies (\lambda x. f x i) \in X \rightarrow_Q Z$
shows $(\lambda(x,y). f x y) \in X \otimes_Q Y \rightarrow_Q Z$
by(*auto intro!: qbs-morphism-pair-swap*[*of case-prod* ($\lambda x y. f y x$),*simplified*] *qbs-morphism-pair-countable1 assms*)

corollary *qbs-morphism-pair-count-space1*:

assumes $\bigwedge i. f i \in Y \rightarrow_Q Z$
shows $(\lambda(x,y). f x y) \in \text{qbs-count-space } (UNIV :: ('a :: \text{countable}) \text{ set}) \otimes_Q Y \rightarrow_Q Z$
by(*auto intro!: qbs-morphism-pair-countable1 simp: qbs-Mx-R assms*)

corollary *qbs-morphism-pair-count-space2*:

assumes $\bigwedge i. (\lambda x. f x i) \in X \rightarrow_Q Z$
shows $(\lambda(x,y). f x y) \in X \otimes_Q \text{qbs-count-space } (UNIV :: ('a :: \text{countable}) \text{ set}) \rightarrow_Q Z$
by(*auto intro!: qbs-morphism-pair-countable2 simp: qbs-Mx-R assms*)

lemma *qbs-morphism-compose-countable'*:

assumes [*qbs*]: $\bigwedge i. i \in I \implies (\lambda x. f i x) \in X \rightarrow_Q Y$ $g \in X \rightarrow_Q \text{qbs-count-space } I$ *countable* I
shows $(\lambda x. f (g x) x) \in X \rightarrow_Q Y$

proof –

have [*qbs*]: $f \in \text{qbs-count-space } I \rightarrow_Q X \Rightarrow_Q Y$
by(*auto intro!: qbs-morphism-count-space' simp: assms(3)*)
show *?thesis*
by *simp*

qed

lemma *qbs-morphism-compose-countable*:

assumes [simp]: $\bigwedge i: 'i::\text{countable}. (\lambda x. f i x) \in X \rightarrow_Q Y \ g \in X \rightarrow_Q (qbs\text{-count-space UNIV})$
shows $(\lambda x. f (g x) x) \in X \rightarrow_Q Y$
by(rule qbs-morphism-compose-countable'[of UNIV f]) simp-all

lemma qbs-morphism-op:

assumes case-prod $f \in X \otimes_M Y \rightarrow_M Z$
shows $f \in \text{measure-to-qbs } X \rightarrow_Q \text{measure-to-qbs } Y \Rightarrow_Q \text{measure-to-qbs } Z$
using r-preserves-morphisms assms
by(fastforce simp: r-preserves-product[symmetric] intro!: curry-preserves-morphisms)

lemma [qbs]:

shows plus-qbs-morphism: $(+) \in (qbs\text{-borel} :: (-::\{\text{second-countable-topology, topological-monoid-add}\}) \text{quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and plus-ereal-qbs-morphism: $(+) \in (qbs\text{-borel} :: \text{ereal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and diff-qbs-morphism: $(-) \in (qbs\text{-borel} :: (-::\{\text{second-countable-topology, real-normed-vector}\}) \text{quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and diff-ennreal-qbs-morphism: $(-) \in (qbs\text{-borel} :: \text{ennreal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and diff-ereal-qbs-morphism: $(-) \in (qbs\text{-borel} :: \text{ereal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and times-qbs-morphism: $(*) \in (qbs\text{-borel} :: (-::\{\text{second-countable-topology, real-normed-algebra}\}) \text{quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and times-ennreal-qbs-morphism: $(*) \in (qbs\text{-borel} :: \text{ennreal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and times-ereal-qbs-morphism: $(*) \in (qbs\text{-borel} :: \text{ereal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and divide-qbs-morphism: $(/) \in (qbs\text{-borel} :: (-::\{\text{second-countable-topology, real-normed-div-algebra}\}) \text{quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and divide-ennreal-qbs-morphism: $(/) \in (qbs\text{-borel} :: \text{ennreal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and divide-ereal-qbs-morphism: $(/) \in (qbs\text{-borel} :: \text{ereal quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and log-qbs-morphism: $\log \in qbs\text{-borel} \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and root-qbs-morphism: $\text{root} \in qbs\text{-count-space UNIV} \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and scaleR-qbs-morphism: $(*_R) \in qbs\text{-borel} \rightarrow_Q (qbs\text{-borel} :: (-::\{\text{second-countable-topology, real-normed-vector}\}) \text{quasi-borel}) \Rightarrow_Q qbs\text{-borel}$
and qbs-morphism-inner: $(\cdot) \in qbs\text{-borel} \rightarrow_Q (qbs\text{-borel} :: (-::\{\text{second-countable-topology, real-inner}\}) \text{quasi-borel}) \Rightarrow_Q qbs\text{-borel}$
and dist-qbs-morphism: $\text{dist} \in (qbs\text{-borel} :: (-::\{\text{second-countable-topology, metric-space}\}) \text{quasi-borel}) \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and powr-qbs-morphism: $(\text{powr}) \in qbs\text{-borel} \rightarrow_Q qbs\text{-borel} \Rightarrow_Q (qbs\text{-borel} :: \text{real quasi-borel})$
and max-qbs-morphism: $(\text{max} :: (-::\{\text{second-countable-topology, linorder-topology}\}) \Rightarrow - \Rightarrow -) \in qbs\text{-borel} \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and min-qbs-morphism: $(\text{min} :: (-::\{\text{second-countable-topology, linorder-topology}\}) \Rightarrow - \Rightarrow -) \in qbs\text{-borel} \rightarrow_Q qbs\text{-borel} \Rightarrow_Q qbs\text{-borel}$
and sup-qbs-morphism: $(\text{sup} :: (-::\{\text{lattice, second-countable-topology, linorder-topology}\})$

$\Rightarrow - \Rightarrow -) \in \text{qbs-borel} \rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-borel}$
and *inf-qbs-morphism*: (*inf* :: (- :: {*lattice*, *second-countable-topology*, *linorder-topology*})
 $\Rightarrow - \Rightarrow -) \in \text{qbs-borel} \rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-borel}$
and *less-qbs-pred*: (<) $\in (\text{qbs-borel} :: - :: \{\text{second-countable-topology}, \text{linorder-topology}\}$
quasi-borel) $\rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-count-space UNIV}$
and *eq-qbs-pred*: (=) $\in (\text{qbs-borel} :: - :: \{\text{second-countable-topology}, \text{linorder-topology}\}$
quasi-borel) $\rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-count-space UNIV}$
and *le-qbs-pred*: (\leq) $\in (\text{qbs-borel} :: - :: \{\text{second-countable-topology}, \text{linorder-topology}\}$
quasi-borel) $\rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-count-space UNIV}$
by(*auto intro!*: *qbs-morphism-op*)

lemma [*qbs*]:

shows *abs-real-qbs-morphism*: *abs* $\in (\text{qbs-borel} :: \text{real quasi-borel}) \rightarrow_Q \text{qbs-borel}$
and *abs-ereal-qbs-morphism*: *abs* $\in (\text{qbs-borel} :: \text{ereal quasi-borel}) \rightarrow_Q \text{qbs-borel}$
and *real-floor-qbs-morphism*: (*floor* :: *real* \Rightarrow *int*) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-count-space UNIV}$
and *real-ceiling-qbs-morphism*: (*ceiling* :: *real* \Rightarrow *int*) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-count-space UNIV}$
and *exp-qbs-morphism*: (*exp*::'a::{*real-normed-field*,*banach*} \Rightarrow 'a) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *ln-qbs-morphism*: *ln* $\in (\text{qbs-borel} :: \text{real quasi-borel}) \rightarrow_Q \text{qbs-borel}$
and *sqrt-qbs-morphism*: *sqrt* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *of-real-qbs-morphism*: (*of-real* :: - \Rightarrow (- :: *real-normed-algebra*)) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *sin-qbs-morphism*: (*sin* :: - \Rightarrow (- :: {*real-normed-field*,*banach*})) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *cos-qbs-morphism*: (*cos* :: - \Rightarrow (- :: {*real-normed-field*,*banach*})) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *arctan-qbs-morphism*: *arctan* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *Re-qbs-morphism*: *Re* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *Im-qbs-morphism*: *Im* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *sgn-qbs-morphism*: (*sgn*::- :: *real-normed-vector* \Rightarrow -) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *norm-qbs-morphism*: *norm* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *invers-qbs-morphism*: (*inverse* :: - \Rightarrow (- :: *real-normed-div-algebra*)) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *invers-ennreal-qbs-morphism*: (*inverse* :: - \Rightarrow *ennreal*) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *invers-ereal-qbs-morphism*: (*inverse* :: - \Rightarrow *ereal*) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *uminus-qbs-morphism*: (*uminus* :: - \Rightarrow (- :: {*second-countable-topology*, *real-normed-vector*})) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *ereal-qbs-morphism*: *ereal* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *real-of-ereal-qbs-morphism*: *real-of-ereal* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *enn2ereal-qbs-morphism*: *enn2ereal* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *e2ennreal-qbs-morphism*: *e2ennreal* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *ennreal-qbs-morphism*: *ennreal* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *qbs-morphism-nth*: ($\lambda x :: \text{real}^n. x \ \$ \ i$) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *qbs-morphism-product-candidate*: $\bigwedge i. (\lambda x. x \ i) \in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
and *uminus-ereal-qbs-morphism*: (*uminus* :: - \Rightarrow *ereal*) $\in \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$
by(*auto intro!*: *set-mp[OF r-preserves-morphisms]*)

lemma *qbs-morphism-sum*:

fixes $f :: 'c \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, topological-comm-monoid-add}\}$
assumes $\bigwedge i. i \in S \implies f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \sum_{i \in S}. f i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by** (*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-suminf-order*:

fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{complete-linorder, second-countable-topology, linorder-topology, topological-comm-monoid-add}\}$
assumes $\bigwedge i. f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \sum i. f i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by** (*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-prod*:

fixes $f :: 'c \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, real-normed-field}\}$
assumes $\bigwedge i. i \in S \implies f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \prod_{i \in S}. f i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by** (*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-Min*:

finite $I \implies (\bigwedge i. i \in I \implies f i \in X \rightarrow_Q \text{qbs-borel}) \implies (\lambda x. \text{Min} ((\lambda i. f i x) 'I) :: 'b :: \{\text{second-countable-topology, linorder-topology}\}) \in X \rightarrow_Q \text{qbs-borel}$
by (*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-Max*:

finite $I \implies (\bigwedge i. i \in I \implies f i \in X \rightarrow_Q \text{qbs-borel}) \implies (\lambda x. \text{Max} ((\lambda i. f i x) 'I) :: 'b :: \{\text{second-countable-topology, linorder-topology}\}) \in X \rightarrow_Q \text{qbs-borel}$
by (*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-Max2*:

fixes $f :: - \Rightarrow - \Rightarrow 'a :: \{\text{second-countable-topology, dense-linorder, linorder-topology}\}$
shows *finite* $I \implies (\bigwedge i. f i \in X \rightarrow_Q \text{qbs-borel}) \implies (\lambda x. \text{Max}\{f i x \mid i. i \in I\}) \in X \rightarrow_Q \text{qbs-borel}$
by (*simp add: lr-adjunction-correspondence*)

lemma [*qbs*]:

shows *qbs-morphism-liminf*: $\text{liminf} \in (\text{qbs-count-space UNIV} \Rightarrow_Q \text{qbs-borel}) \Rightarrow_Q (\text{qbs-borel} :: 'a :: \{\text{complete-linorder, second-countable-topology, linorder-topology}\}) \text{quasi-borel}$

and *qbs-morphism-limsup*: $\text{limsup} \in (\text{qbs-count-space UNIV} \Rightarrow_Q \text{qbs-borel}) \Rightarrow_Q (\text{qbs-borel} :: 'a :: \{\text{complete-linorder, second-countable-topology, linorder-topology}\}) \text{quasi-borel}$

and *qbs-morphism-lim*: $\text{lim} \in (\text{qbs-count-space UNIV} \Rightarrow_Q \text{qbs-borel}) \Rightarrow_Q (\text{qbs-borel} :: 'a :: \{\text{complete-linorder, second-countable-topology, linorder-topology}\}) \text{quasi-borel}$

proof (*safe intro!*: *qbs-morphismI*)

fix $f :: \text{real} \Rightarrow \text{nat} \Rightarrow 'a$

assume $f \in \text{qbs-Mx} (\text{count-space}_Q \text{ UNIV} \Rightarrow_Q \text{borel}_Q)$

then have [*measurable*]: $\bigwedge i. (\lambda r. f r i) \in \text{borel-measurable borel}$

by(*auto simp: qbs-Mx-is-morphisms*) (*metis PiQ-qbs-borel measurable-product-then-coordinatewise qbs-Mx-is-morphisms qbs-Mx-qbs-borel qbs-morphism-product-iff*)
show $\liminf \circ f \in \text{qbs-Mx borel}_Q \limsup \circ f \in \text{qbs-Mx borel}_Q \lim \circ f \in \text{qbs-Mx borel}_Q$
qed

lemma qbs-morphism-SUP:
fixes $F :: - \Rightarrow - \Rightarrow :: \{\text{complete-linorder, linorder-topology, second-countable-topology}\}$
assumes $\text{countable } I \wedge i. i \in I \implies F i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \bigsqcup_{i \in I}. F i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms by(simp add: lr-adjunction-correspondence)*

lemma qbs-morphism-INF:
fixes $F :: - \Rightarrow - \Rightarrow :: \{\text{complete-linorder, linorder-topology, second-countable-topology}\}$
assumes $\text{countable } I \wedge i. i \in I \implies F i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \bigsqcap_{i \in I}. F i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms by(simp add: lr-adjunction-correspondence)*

lemma qbs-morphism-cSUP:
fixes $F :: - \Rightarrow - \Rightarrow 'a :: \{\text{conditionally-complete-linorder, linorder-topology, second-countable-topology}\}$
assumes $\text{countable } I \wedge i. i \in I \implies F i \in X \rightarrow_Q \text{qbs-borel} \wedge x. x \in \text{qbs-space } X \implies \text{bdd-above } ((\lambda i. F i x) ' I)$
shows $(\lambda x. \bigsqcup_{i \in I}. F i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms by(simp add: lr-adjunction-correspondence space-L)*

lemma qbs-morphism-cINF:
fixes $F :: - \Rightarrow - \Rightarrow 'a :: \{\text{conditionally-complete-linorder, linorder-topology, second-countable-topology}\}$
assumes $\text{countable } I \wedge i. i \in I \implies F i \in X \rightarrow_Q \text{qbs-borel} \wedge x. x \in \text{qbs-space } X \implies \text{bdd-below } ((\lambda i. F i x) ' I)$
shows $(\lambda x. \bigsqcap_{i \in I}. F i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms by(simp add: lr-adjunction-correspondence space-L)*

lemma qbs-morphism-lim-metric:
fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{banach, second-countable-topology}\}$
assumes $\wedge i. f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \lim (\lambda i. f i x)) \in X \rightarrow_Q \text{qbs-borel}$
using *assms by(simp add: lr-adjunction-correspondence)*

lemma qbs-morphism-LIMSEQ-metric:
fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \text{metric-space}$
assumes $\wedge i. f i \in X \rightarrow_Q \text{qbs-borel} \wedge x. x \in \text{qbs-space } X \implies (\lambda i. f i x) \longrightarrow g x$
shows $g \in X \rightarrow_Q \text{qbs-borel}$
using *borel-measurable-LIMSEQ-metric[where M=qbs-to-measure X] assms by(auto simp add: lr-adjunction-correspondence space-L)*

lemma *power-qbs-morphism*[qbs]:
 (power :: (- :: {power, real-normed-algebra}) \Rightarrow nat \Rightarrow -) \in qbs-borel \rightarrow_Q qbs-count-space
 UNIV \Rightarrow_Q qbs-borel
 by(rule arg-swap-morphism) (auto intro!: qbs-morphism-count-space set-mp[OF
 r-preserves-morphisms])

lemma *power-ennreal-qbs-morphism*[qbs]:
 (power :: ennreal \Rightarrow nat \Rightarrow -) \in qbs-borel \rightarrow_Q qbs-count-space UNIV \Rightarrow_Q qbs-borel
 by(rule arg-swap-morphism) (auto intro!: qbs-morphism-count-space set-mp[OF
 r-preserves-morphisms])

lemma *qbs-morphism-compw*: ($\widetilde{}$) \in (X \Rightarrow_Q X) \rightarrow_Q qbs-count-space UNIV \Rightarrow_Q
 (X \Rightarrow_Q X)

proof(rule arg-swap-morphism, rule qbs-morphism-count-space)

fix n

show ($\lambda y. y \widetilde{} n$) \in X \Rightarrow_Q X \rightarrow_Q X \Rightarrow_Q X

by(induction n) simp-all

qed

lemma *qbs-morphism-compose-n*[qbs]:

assumes [qbs]: $f \in X \rightarrow_Q X$

shows ($\lambda n. f \widetilde{} n$) \in qbs-count-space UNIV \rightarrow_Q X \Rightarrow_Q X

proof(intro qbs-morphism-count-space)

fix n

show $f \widetilde{} n \in X \rightarrow_Q X$

by (induction n) simp-all

qed

lemma *qbs-morphism-compose-n'*:

assumes $f \in X \rightarrow_Q X$

shows $f \widetilde{} n \in X \rightarrow_Q X$

using qbs-morphism-space[OF qbs-morphism-compose-n[OF assms]] by(simp add:
 exp-qbs-space qbs-space-R)

lemma *qbs-morphism-uminus-eq-ereal*[simp]:

($\lambda x. - f x :: ereal$) \in X \rightarrow_Q qbs-borel \longleftrightarrow $f \in X \rightarrow_Q$ qbs-borel (is ?l = ?r)

by(simp add: lr-adjunction-correspondence)

lemma *qbs-morphism-ereal-iff*:

shows ($\lambda x. ereal (f x)$) \in X \rightarrow_Q qbs-borel \longleftrightarrow $f \in X \rightarrow_Q$ qbs-borel

by(simp add: borel-measurable-ereal-iff lr-adjunction-correspondence)

lemma *qbs-morphism-ereal-sum*:

fixes $f :: 'c \Rightarrow 'a \Rightarrow ereal$

assumes $\bigwedge i. i \in S \implies f i \in X \rightarrow_Q$ qbs-borel

shows ($\lambda x. \sum i \in S. f i x$) \in X \rightarrow_Q qbs-borel

using assms by(simp add: lr-adjunction-correspondence)

lemma *qbs-morphism-ereal-prod*:

fixes $f :: 'c \Rightarrow 'a \Rightarrow \text{ereal}$
assumes $\bigwedge i. i \in S \Longrightarrow f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \prod_{i \in S}. f i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by**(*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-extreal-suminf*:
fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow \text{ereal}$
assumes $\bigwedge i. f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. (\sum i. f i x)) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by**(*simp add: lr-adjunction-correspondence*)

lemma *qbs-morphism-ennreal-iff*:
assumes $\bigwedge x. x \in \text{qbs-space } X \Longrightarrow 0 \leq f x$
shows $(\lambda x. \text{ennreal } (f x)) \in X \rightarrow_Q \text{qbs-borel} \longleftrightarrow f \in X \rightarrow_Q \text{qbs-borel}$
using *borel-measurable-ennreal-iff*[**where** $M = \text{qbs-to-measure } X$] *assms*
by(*simp add: space-L lr-adjunction-correspondence*)

lemma *qbs-morphism-prod-ennreal*:
fixes $f :: 'c \Rightarrow 'a \Rightarrow \text{ennreal}$
assumes $\bigwedge i. i \in S \Longrightarrow f i \in X \rightarrow_Q \text{qbs-borel}$
shows $(\lambda x. \prod_{i \in S}. f i x) \in X \rightarrow_Q \text{qbs-borel}$
using *assms* **by**(*simp add: space-L lr-adjunction-correspondence*)

lemma *count-space-qbs-morphism*:
 $f \in \text{qbs-count-space } (UNIV :: 'a \text{ set}) \rightarrow_Q \text{qbs-borel}$
by(*auto intro!: set-mp[OF r-preserves-morphisms]*)

declare *count-space-qbs-morphism*[**where** $'a = - :: \text{countable, qbs}$]

lemma *count-space-count-space-qbs-morphism*:
 $f \in \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set}) \rightarrow_Q \text{qbs-count-space } (UNIV :: (- :: \text{countable}) \text{ set})$
by(*auto intro!: set-mp[OF r-preserves-morphisms]*)

lemma *qbs-morphism-case-nat'*:
assumes [*qbs*]: $i = 0 \Longrightarrow f \in X \rightarrow_Q Y$
 $\bigwedge j. i = \text{Suc } j \Longrightarrow (\lambda x. g x j) \in X \rightarrow_Q Y$
shows $(\lambda x. \text{case-nat } (f x) (g x) i) \in X \rightarrow_Q Y$
by (*cases i*) *simp-all*

lemma *qbs-morphism-case-nat*[*qbs*]:
 $\text{case-nat} \in X \rightarrow_Q (\text{qbs-count-space } UNIV \Rightarrow_Q X) \Rightarrow_Q \text{qbs-count-space } UNIV$
 $\Rightarrow_Q X$
by(*rule curry-preserves-morphisms, rule arg-swap-morphism*) (*auto intro!: qbs-morphism-count-space qbs-morphism-case-nat'*)

lemma *qbs-morphism-case-nat''*:
assumes $f \in X \rightarrow_Q Y$ $g \in X \rightarrow_Q (\prod_Q i \in UNIV. Y)$

shows $(\lambda x. \text{case-nat } (f x) (g x)) \in X \rightarrow_Q (\Pi_Q i \in \text{UNIV}. Y)$
using *assms* **by** (*simp* *add*: *qbs-morphism-product-iff*)

lemma *qbs-morphism-rec-nat*[*qbs*]: $\text{rec-nat} \in X \rightarrow_Q (\text{count-space UNIV} \Rightarrow_Q X \Rightarrow_Q X) \Rightarrow_Q \text{count-space UNIV} \Rightarrow_Q X$
proof (*rule* *curry-preserves-morphisms*, *rule* *arg-swap-morphism*, *rule* *qbs-morphism-count-space*)
fix *n*
show $(\lambda y. \text{rec-nat } (fst y) (snd y) n) \in X \otimes_Q (\text{qbs-count-space UNIV} \Rightarrow_Q X \Rightarrow_Q X) \rightarrow_Q X$
by (*induction* *n*) *simp-all*
qed

lemma *qbs-morphism-Max-nat*:
fixes $P :: \text{nat} \Rightarrow 'a \Rightarrow \text{bool}$
assumes $\bigwedge i. P i \in X \rightarrow_Q \text{qbs-count-space UNIV}$
shows $(\lambda x. \text{Max } \{i. P i x\}) \in X \rightarrow_Q \text{qbs-count-space UNIV}$
using *assms* **by** (*simp* *add*: *lr-adjunction-correspondence*)

lemma *qbs-morphism-Min-nat*:
fixes $P :: \text{nat} \Rightarrow 'a \Rightarrow \text{bool}$
assumes $\bigwedge i. P i \in X \rightarrow_Q \text{qbs-count-space UNIV}$
shows $(\lambda x. \text{Min } \{i. P i x\}) \in X \rightarrow_Q \text{qbs-count-space UNIV}$
using *assms* **by** (*simp* *add*: *lr-adjunction-correspondence*)

lemma *qbs-morphism-sum-nat*:
fixes $f :: 'c \Rightarrow 'a \Rightarrow \text{nat}$
assumes $\bigwedge i. i \in S \implies f i \in X \rightarrow_Q \text{qbs-count-space UNIV}$
shows $(\lambda x. \sum i \in S. f i x) \in X \rightarrow_Q \text{qbs-count-space UNIV}$
using *assms* **by** (*simp* *add*: *lr-adjunction-correspondence*)

lemma *qbs-morphism-case-enat'*:
assumes $f[\text{qbs}]: f \in X \rightarrow_Q \text{qbs-count-space UNIV}$ **and** $[\text{qbs}]: \bigwedge i. g i \in X \rightarrow_Q Y$
 $h \in X \rightarrow_Q Y$
shows $(\lambda x. \text{case } f x \text{ of } \text{enat } i \Rightarrow g i x \mid \infty \Rightarrow h x) \in X \rightarrow_Q Y$
proof (*rule* *qbs-morphism-compose-countable*[*OF* - *f*])
fix *i*
show $(\lambda x. \text{case } i \text{ of } \text{enat } i \Rightarrow g i x \mid \infty \Rightarrow h x) \in X \rightarrow_Q Y$
by (*cases* *i*) *simp-all*
qed

lemma *qbs-morphism-case-enat*[*qbs*]: $\text{case-enat} \in \text{qbs-space } ((\text{qbs-count-space UNIV} \Rightarrow_Q X) \Rightarrow_Q X \Rightarrow_Q \text{qbs-count-space UNIV} \Rightarrow_Q X)$
proof –
note *qbs-morphism-case-enat'*[*qbs*]
show *?thesis*
by (*auto* *intro!*: *curry-preserves-morphisms*, *rule* *qbs-morphismI*) (*simp* *add*: *qbs-Mx-is-morphisms* *comp-def*, *qbs*, *simp-all*)
qed

lemma *qbs-morphism-restrict*[qbs]:
assumes $X: \bigwedge i. i \in I \implies f i \in X \rightarrow_Q (Y i)$
shows $(\lambda x. \lambda i \in I. f i x) \in X \rightarrow_Q (\prod_Q i \in I. Y i)$
using *assms* **by**(*auto intro!*: *product-qbs-canonical1*)

lemma *If-qbs-morphism*[qbs]: $If \in \text{qbs-count-space } UNIV \rightarrow_Q X \Rightarrow_Q X \Rightarrow_Q X$
proof(*rule qbs-morphismI*)
show $\alpha \in \text{qbs-Mx } (\text{count-space}_Q UNIV) \implies If \circ \alpha \in \text{qbs-Mx } (X \Rightarrow_Q X \Rightarrow_Q X)$ **for** α
by(*auto intro!*: *qbs-Mx-indicat*[**where** $S = \{r. \alpha (- (- r))\}$,*simplified*]) *simp*:
qbs-Mx-count-space exp-qbs-Mx)
qed

lemma *normal-density-qbs*[qbs]: $\text{normal-density} \in \text{qbs-borel} \rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-borel}$
 $\Rightarrow_Q \text{qbs-borel}$
proof –
have [*simp*]: $\text{normal-density} = (\lambda \mu \sigma x. 1 / \text{sqrt } (2 * \text{pi} * \sigma^2) * \text{exp } (-(x - \mu)^2 / (2 * \sigma^2)))$
by *standard+* (*auto simp: normal-density-def*)
show *?thesis*
by *simp*
qed

lemma *erlang-density-qbs*[qbs]: $\text{erlang-density} \in \text{qbs-count-space } UNIV \rightarrow_Q \text{qbs-borel}$
 $\Rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{qbs-borel}$
proof –
have [*simp*]: $\text{erlang-density} = (\lambda k l x. (\text{if } x < 0 \text{ then } 0 \text{ else } (l \wedge (\text{Suc } k) * x \wedge k * \text{exp } (- l * x)) / \text{fact } k))$
by *standard+* (*auto simp: erlang-density-def*)
show *?thesis*
by *simp*
qed

lemma *list-nil-qbs*[qbs]: $[] \in \text{qbs-space } (\text{list-qbs } X)$
by(*simp add: list-qbs-space*)

lemma *list-cons-qbs-morphism*: $\text{list-cons} \in X \rightarrow_Q (\prod_Q n \in (UNIV :: \text{nat set}). \prod_Q i \in \{..<n\}. X) \Rightarrow_Q (\prod_Q n \in (UNIV :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
proof(*intro curry-preserves-morphisms pair-qbs-morphismI*)
fix $\alpha \beta$
assume $h: \alpha \in \text{qbs-Mx } X$
 $\beta \in \text{qbs-Mx } (\prod_Q n \in (UNIV :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$
then obtain γf **where** *hf*:
 $\beta = (\lambda r. (f r, \gamma (f r) r)) f \in \text{borel} \rightarrow_M \text{count-space } UNIV \bigwedge i. i \in \text{range } f \implies$
 $\gamma i \in \text{qbs-Mx } (\prod_Q j \in \{..<i\}. X)$
by(*auto simp: coPiQ-Mx-def coPiQ-Mx*)
define $f' \beta'$
where $f' \equiv (\lambda r. \text{Suc } (f r)) \beta' \equiv (\lambda i r n. \text{if } n = 0 \text{ then } \alpha r \text{ else } \gamma (i - 1) r (n$

```

- 1))
then have ( $\lambda r. \text{list-cons } (\text{fst } (\alpha r, \beta r)) (\text{snd } (\alpha r, \beta r))$ ) = ( $\lambda r. (f' r, \beta' (f' r))$ )
proof(rule coPiQ-MxI)
  show  $f' \in \text{borel} \rightarrow_M \text{count-space UNIV}$ 
    using hf by(simp add: f'- $\beta'$ -def(1))
next
  fix  $j$ 
  assume  $hj:j \in \text{range } f'$ 
  then have  $hj':j - 1 \in \text{range } f'$ 
    by(auto simp: f'- $\beta'$ -def(1))
  show  $\beta' j \in \text{qbs-Mx } (\prod_Q i \in \{..<j\}. X)$ 
  proof(rule prod-qbs-MxI)
    fix  $i$ 
    assume  $hi:i \in \{..<j\}$ 
    then consider  $i = 0 \mid 0 < i < j$ 
      by auto
    then show ( $\lambda r. \beta' j r i$ )  $\in \text{qbs-Mx } X$ 
    proof cases
      case 1
      then show ?thesis by(simp add: h(1) f'- $\beta'$ -def(2))
    next
      case 2
      then have  $i - 1 \in \{..<j - 1\}$  by simp
      from prod-qbs-MxD(1)[OF hf(3)[OF hj'] this] 2
      show ?thesis
        by(simp add: f'- $\beta'$ -def(2))
    qed
  next
  fix  $i$ 
  assume  $hi:i \notin \{..<j\}$ 
  then have  $i \neq 0 \mid i - \text{Suc } 0 \notin \{..<j - \text{Suc } 0\}$ 
    using f'- $\beta'$ -def(1) hj by fastforce+
  with prod-qbs-MxD(2)[OF hf(3)[OF hj']]
  show ( $\lambda r. \beta' j r i$ ) = ( $\lambda r. \text{undefined}$ )
    by(simp add: f'- $\beta'$ -def(2))
  qed
qed
finally show ( $\lambda r. \text{list-cons } (\text{fst } (\alpha r, \beta r)) (\text{snd } (\alpha r, \beta r))$ )  $\in \text{qbs-Mx } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..<n\}. X)$  .
qed

```

```

corollary cons-qbs-morphism[qbs]:  $\text{Cons} \in X \rightarrow_Q (\text{list-qbs } X) \Rightarrow_Q \text{list-qbs } X$ 
proof(rule arg-swap-morphism)
  show ( $\lambda x y. y \# x$ )  $\in \text{list-qbs } X \rightarrow_Q X \Rightarrow_Q \text{list-qbs } X$ 
  proof(rule qbs-morphism-cong')
    show ( $\lambda l x. x \# \text{to-list } l$ )  $\circ \text{from-list} \in \text{list-qbs } X \rightarrow_Q X \Rightarrow_Q \text{list-qbs } X$ 

```

```

proof(rule qbs-morphism-comp)
  show ( $\lambda l x. x \# \text{to-list } l$ )  $\in$  ( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )  $\rightarrow_Q$ 
 $X \Rightarrow_Q \text{list-qbs } X$ 
  proof(rule curry-preserves-morphisms)
  show ( $\lambda x. \text{snd } lx \# \text{to-list } (\text{fst } lx)$ )  $\in$  ( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )
 $\otimes_Q X \rightarrow_Q \text{list-qbs } X$ 
  proof(rule qbs-morphism-cong^)
  show  $\text{to-list} \circ (\lambda(l, x). \text{from-list } (x \# \text{to-list } l)) \in$  ( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )
 $\otimes_Q X \rightarrow_Q \text{list-qbs } X$ 
  proof(rule qbs-morphism-comp)
  show ( $\lambda(l, x). \text{from-list } (x \# \text{to-list } l)$ )  $\in$  ( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )
 $\otimes_Q X \rightarrow_Q$  ( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )
  by(rule qbs-morphism-cong^[OF - uncurry-preserves-morphisms[of  $\lambda(l, x). \text{list-cons } x l, \text{simplified}, \text{OF arg-swap-morphism}[\text{OF list-cons-qbs-morphism}]]$ ] (auto simp: pair-qbs-space to-list-from-list-ident))
  qed(simp add: list-qbs-def map-qbs-morphism-f)
  qed(auto simp: pair-qbs-space to-list-from-list-ident to-list-simp2)
qed
qed(auto simp: list-qbs-def to-list-from-list-ident intro!: map-qbs-morphism-inverse-f)
qed(simp add: from-list-to-list-ident)
qed

```

lemma *rec-list-morphism'*:

```

rec-list'  $\in$  qbs-space ( $Y \Rightarrow_Q (X \Rightarrow_Q (\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X) \Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q$ 
( $\Pi_Q n \in (UNIV :: \text{nat set}). \Pi_Q i \in \{..<n\}. X$ )  $\Rightarrow_Q Y$ )
proof(intro curry-preserves-morphisms[OF arg-swap-morphism] coPiQ-canonical1')
  fix  $n$ 
  show ( $\lambda x y. \text{rec-list}' (\text{fst } y) (\text{snd } y) (n, x)$ )  $\in$  ( $\Pi_Q i \in \{..<n\}. X$ )  $\rightarrow_Q \text{exp-qbs } (Y \otimes_Q \text{exp-qbs } X$ 
( $\text{exp-qbs } (\Pi_Q n \in UNIV. \Pi_Q i \in \{..<n\}. X) (\text{exp-qbs } Y Y))) Y$ 
  proof(induction  $n$ )
  case 0
  show ?case
  proof(rule curry-preserves-morphisms[OF qbs-morphismI])
  fix  $\alpha$ 
  assume  $h: \alpha \in \text{qbs-Mx } ((\Pi_Q i \in \{..<0::\text{nat}\}. X) \otimes_Q Y \otimes_Q \text{exp-qbs } X (\text{exp-qbs } (\Pi_Q n \in UNIV. \Pi_Q i \in \{..<n::\text{nat}\}. X) (\text{exp-qbs } Y Y)))$ 
  have  $\bigwedge r. \text{fst } (\alpha r) = (\lambda n. \text{undefined})$ 
  proof –
  fix  $r$ 
  have  $\bigwedge i. (\lambda r. \text{fst } (\alpha r) i) = (\lambda r. \text{undefined})$ 
  using  $h$  by(auto simp: exp-qbs-Mx PiQ-Mx pair-qbs-Mx comp-def split-beta')
  thus  $\text{fst } (\alpha r) = (\lambda n. \text{undefined})$ 
  by(fastforce dest: fun-cong)
  qed
  hence ( $\lambda xy. \text{rec-list}' (\text{fst } (\text{snd } xy)) (\text{snd } (\text{snd } xy)) (0, \text{fst } xy)) \circ \alpha = (\lambda x. \text{fst } (\text{snd } (\alpha x)))$ 
  by(auto simp: rec-list'-simp1[simplified list-nil-def] comp-def split-beta')
  also have  $\dots \in \text{qbs-Mx } Y$ 
  using  $h$  by(auto simp: pair-qbs-Mx comp-def)

```

finally show $(\lambda xy. \text{rec-list}' (fst (snd xy)) (snd (snd xy)) (0, fst xy)) \circ \alpha \in$
qbs-Mx Y .
qed
next
case *ih*:(*Suc n*)
show *?case*
proof(*rule qbs-morphismI*)
fix α
assume $h:\alpha \in \text{qbs-Mx } (\prod_Q i \in \{..< \text{Suc } n\}. X)$
define α' **where** $\alpha' \equiv (\lambda r. \text{snd } (\text{list-tail } (\text{Suc } n, \alpha r)))$
define a **where** $a \equiv (\lambda r. \alpha r 0)$
then have $ha:a \in \text{qbs-Mx } X$
using h **by**(*auto simp: PiQ-Mx*)
have $1:\alpha' \in \text{qbs-Mx } (\prod_Q i \in \{..< n\}. X)$
using h **by**(*fastforce simp: PiQ-Mx list-tail-def \alpha'-def*)
hence $2: \bigwedge r. (n, \alpha' r) \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X)$
using *qbs-Mx-to-X*[*of \alpha'*] **by** (*fastforce simp: PiQ-space coPiQ-space*)
have $3: \bigwedge r. (\text{Suc } n, \alpha r) \in \text{qbs-space } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X)$
using *qbs-Mx-to-X*[*of \alpha*] **by** (*fastforce simp: PiQ-space coPiQ-space*)
have $4: \bigwedge r. (n, \alpha' r) = \text{list-tail } (\text{Suc } n, \alpha r)$
by(*simp add: list-tail-def \alpha'-def*)
have $5: \bigwedge r. (\text{Suc } n, \alpha r) = \text{list-cons } (a r) (n, \alpha' r)$
unfolding *a-def* **by**(*simp add: list-simp5[OF 3,simplified 4[symmetric],simplified list-head-def list-cons-def list-nil-def] list-cons-def*) *auto*
have $6: (\lambda r. (n, \alpha' r)) \in \text{qbs-Mx } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X)$
using 1 **by**(*auto intro!: coPiQ-MxI simp: PiQ-space coPiQ-space*)

have $(\lambda x y. \text{rec-list}' (fst y) (snd y) (\text{Suc } n, x)) \circ \alpha = (\lambda r y. \text{rec-list}' (fst y) (snd y) (\text{Suc } n, \alpha r))$
by *auto*
also have $\dots = (\lambda r y. \text{snd } y (a r) (n, \alpha' r) (\text{rec-list}' (fst y) (snd y) (n, \alpha' r)))$
by(*simp only: 5 rec-list'-simp2[OF 2]*)
also have $\dots \in \text{qbs-Mx } (\text{exp-qbs } (Y \otimes_Q \text{exp-qbs } X (\text{exp-qbs } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) (\text{exp-qbs } Y Y)))) Y$
proof –
have $(\lambda(r,y). \text{snd } y (a r) (n, \alpha' r) (\text{rec-list}' (fst y) (snd y) (n, \alpha' r))) =$
 $(\lambda(y,x1,x2,x3). y x1 x2 x3) \circ (\lambda(r,y). (\text{snd } y, a r, (n, \alpha' r), \text{rec-list}' (fst y) (snd y) (n, \alpha' r)))$
by *auto*
also have $\dots \in \text{qbs-borel } \otimes_Q (Y \otimes_Q \text{exp-qbs } X (\text{exp-qbs } (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) (\text{exp-qbs } Y Y)))) \rightarrow_Q Y$
proof(*rule qbs-morphism-comp*[*where Y=exp-qbs X (exp-qbs (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) (\text{exp-qbs } Y Y)) \otimes_Q X \otimes_Q (\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) \otimes_Q Y*])
show $(\lambda(r, y). (\text{snd } y, a r, (n, \alpha' r), \text{rec-list}' (fst y) (snd y) (n, \alpha' r))) \in \text{qbs-borel } \otimes_Q Y \otimes_Q \text{exp-qbs } X (\text{exp-qbs } ((\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) (\text{exp-qbs } Y Y)) \rightarrow_Q \text{exp-qbs } X (\text{exp-qbs } ((\prod_Q n \in (\text{UNIV} :: \text{nat set}). \prod_Q i \in \{..< n\}. X) (\text{exp-qbs } Y Y))))$

$set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) \otimes_Q X \otimes_Q (\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X) \otimes_Q Y$
unfolding *split-beta'*
proof(*safe intro!*: *qbs-morphism-Pair*)
show $(\lambda x. a (fst x)) \in qbs-borel \otimes_Q Y \otimes_Q exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) \rightarrow_Q X$
using *ha qbs-Mx-is-morphisms[of X] ha by auto*
next
show $(\lambda x. (n, \alpha' (fst x))) \in qbs-borel \otimes_Q Y \otimes_Q exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) \rightarrow_Q (\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)$
using *6 by(simp add: qbs-Mx-is-morphisms) (use fst-qbs-morphism qbs-morphism-compose in blast)*
next
show $(\lambda x. rec-list' (fst (snd x)) (snd (snd x)) (n, \alpha' (fst x))) \in qbs-borel \otimes_Q Y \otimes_Q exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) \rightarrow_Q Y$
using *qbs-morphism-Mx[OF ih 1, simplified comp-def] uncurry-preserves-morphisms[of $(\lambda(x,y). rec-list' (fst y) (snd y) (n, \alpha' x)) qbs-borel Y \otimes_Q exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) Y] qbs-Mx-is-morphisms[of exp-qbs (Y \otimes_Q exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y))) Y]$*
by(*fastforce simp: split-beta'*)
qed *qbs*
next
show $(\lambda(y, x1, x2, x3). y x1 x2 x3) \in exp-qbs X (exp-qbs ((\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)) (exp-qbs Y Y)) \otimes_Q X \otimes_Q (\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X) \otimes_Q Y \rightarrow_Q Y$
by *simp*
qed
finally show *?thesis*
by(*simp add: exp-qbs-Mx'*)
qed
finally show $(\lambda x y. rec-list' (fst y) (snd y) (Suc n, x)) \circ \alpha \in qbs-Mx (exp-qbs (Y \otimes_Q exp-qbs X (exp-qbs (\Pi_Q n \in UNIV. \Pi_Q i \in \{.. < n\}. X) (exp-qbs Y Y))) Y)$
by *blast*
qed
qed
qed *simp*

lemma *rec-list-morphism[qbs]*: $rec-list \in qbs-space (Y \Rightarrow_Q (X \Rightarrow_Q list-qbs X \Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q list-qbs X \Rightarrow_Q Y)$

proof(*rule curry-preserves-morphisms[OF arg-swap-morphism]*)

show $(\lambda l yf. rec-list (fst yf) (snd yf) l) \in list-qbs X \rightarrow_Q Y \otimes_Q (X \Rightarrow_Q list-qbs X \Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q Y$

proof(*rule qbs-morphism-cong'*[**where** $f = (\lambda l' (y, f). rec-list y f (to-list l')) \circ from-list, OF - qbs-morphism-comp$ [**where** $Y = (\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)$]])

show $(\lambda l' (y, f). rec-list y f (to-list l')) \in (\Pi_Q n \in (UNIV :: nat set). \Pi_Q i \in \{.. < n\}. X)$

```

X)  $\rightarrow_Q Y \otimes_Q (X \Rightarrow_Q \text{list-qbs } X \Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q Y$ 
  apply(rule arg-swap-morphism,simp only: split-beta' list-qbs-def)
  apply(rule uncurry-preserves-morphisms)
  apply(rule arg-swap-morphism)
  apply(rule arg-swap-morphism')
  apply(rule qbs-morphism-cong'[OF - arg-swap-morphism-map-qbs1 [OF arg-swap-morphism'[OF
arg-swap-morphism[OF rec-list-morphism']]])
  apply(auto simp: rec-list'-def from-list-to-list-ident)
  done
qed(auto simp: from-list-to-list-ident list-qbs-def to-list-from-list-ident intro!: map-qbs-morphism-inverse-f)
qed

```

hide-const (open) *list-nil list-cons list-head list-tail from-list rec-list' to-list'*

hide-fact (open) *list-simp1 list-simp2 list-simp3 list-simp4 list-simp5 list-simp6
list-simp7 from-list-in-list-of' list-cons-qbs-morphism rec-list'-simp1
to-list-from-list-ident from-list-in-list-of to-list-set to-list-simp1 to-list-simp2
list-head-def list-tail-def from-list-length
list-cons-in-list-of rec-list-morphism' rec-list'-simp2 list-decomp1 list-destruct-rule
list-induct-rule from-list-to-list-ident*

corollary *case-list-morphism[qbs]: case-list \in qbs-space ((Y :: 'b quasi-borel) \Rightarrow_Q
(X :: 'a quasi-borel) $\Rightarrow_Q \text{list-qbs } X \Rightarrow_Q Y) \Rightarrow_Q \text{list-qbs } X \Rightarrow_Q Y$*

proof –

```

  have [simp]: case-list = ( $\lambda y (f :: 'a \Rightarrow 'a \text{ list} \Rightarrow 'b) l. \text{rec-list } y (\lambda x l' y. f x l') l$ )
  proof standard+
    fix y :: 'b and f :: 'a  $\Rightarrow$  'a list  $\Rightarrow$  'b and l :: 'a list
    show (case l of []  $\Rightarrow$  y | x # xa  $\Rightarrow$  f x xa) = rec-list y ( $\lambda x l' y. f x l')$  l
    by (cases l) auto
  qed

```

qed

show ?thesis

by simp

qed

lemma *fold-qbs-morphism[qbs]: fold \in qbs-space ((X $\Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q \text{list-qbs}$
 $X \Rightarrow_Q Y \Rightarrow_Q Y)$*

proof –

```

  have [simp]: fold = ( $\lambda f l. \text{rec-list id } (\lambda x xs l. l \circ f x) l$ )
  apply standard+
  subgoal for f l x
    by (induction l arbitrary: x) simp-all
  done

```

done

show ?thesis

by simp

qed

lemma [qbs]:

shows *foldr-qbs-morphism: foldr \in qbs-space ((X $\Rightarrow_Q Y \Rightarrow_Q Y) \Rightarrow_Q \text{list-qbs } X$
 $\Rightarrow_Q Y \Rightarrow_Q Y)$*

and foldl-qbs-morphism: $foldl \in qbs\text{-space } ((X \Rightarrow_Q Y \Rightarrow_Q X) \Rightarrow_Q X \Rightarrow_Q list\text{-qbs } Y \Rightarrow_Q X)$
and zip-qbs-morphism: $zip \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q list\text{-qbs } Y \Rightarrow_Q list\text{-qbs } (pair\text{-qbs } X Y))$
and append-qbs-morphism: $append \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
and concat-qbs-morphism: $concat \in qbs\text{-space } (list\text{-qbs } (list\text{-qbs } X) \Rightarrow_Q list\text{-qbs } X)$
and drop-qbs-morphism: $drop \in qbs\text{-space } (qbs\text{-count-space } UNIV \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
and take-qbs-morphism: $take \in qbs\text{-space } (qbs\text{-count-space } UNIV \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
and rev-qbs-morphism: $rev \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
by(*auto simp: foldr-def foldl-def zip-def append-def concat-def drop-def take-def rev-def*)

lemma [*qbs*]:

fixes $X :: 'a \text{ quasi-borel}$ **and** $Y :: 'b \text{ quasi-borel}$
shows *map-qbs-morphism:* $map \in qbs\text{-space } ((X \Rightarrow_Q Y) \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } Y)$ (**is** *?map*)
and *filter-qbs-morphism:* $filter \in qbs\text{-space } ((X \Rightarrow_Q count\text{-space}_Q UNIV) \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$ (**is** *?filter*)
and *length-qbs-morphism:* $length \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q qbs\text{-count-space } UNIV)$ (**is** *?length*)
and *tl-qbs-morphism:* $tl \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$ (**is** *?tl*)
and *list-all-qbs-morphism:* $list\text{-all} \in qbs\text{-space } ((X \Rightarrow_Q qbs\text{-count-space } UNIV) \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q qbs\text{-count-space } UNIV)$ (**is** *?list-all*)
and *bind-list-qbs-morphism:* $(\gg=) \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q (X \Rightarrow_Q list\text{-qbs } Y) \Rightarrow_Q list\text{-qbs } Y)$ (**is** *?bind*)

proof –

have [*simp*]: $map = (\lambda f. rec\text{-list } [] (\lambda x \text{ xs } l. f x \# l))$

apply *standard+*

subgoal for $f \ l$

by(*induction l simp-all*)

done

have [*simp*]: $filter = (\lambda P. rec\text{-list } [] (\lambda x \text{ xs } l. if P x then x \# l else l))$

apply *standard+*

subgoal for $f \ l$

by(*induction l simp-all*)

done

have [*simp*]: $length = (\lambda l. foldr (\lambda _ n. Suc n) l 0)$

apply *standard*

subgoal for l

by (*induction l simp-all*)

done

have [*simp*]: $tl = (\lambda l. case \ l \ of \ [] \Rightarrow \ [] \ | \ _ \ \# \ xs \Rightarrow \ xs)$

by *standard (simp add: tl-def)*

have [*simp*]: $list\text{-all} = (\lambda P \ xs. foldr (\lambda x \ b. b \wedge P x) xs \ True)$

apply (*standard,standard*)

subgoal for P xs
 by(*induction xs arbitrary: P*) *auto*
done
 have [*simp*]: $List.bind = (\lambda xs f. concat (map f xs))$
 by *standard+* (*simp add: List.bind-def*)
 show ?*map* ?*filter* ?*length* ?*tl* ?*list-all* ?*bind*
 by *simp-all*
qed

lemma *list-eq-qbs-morphism*[*qbs*]:
 assumes [*qbs*]: $(=) \in qbs\text{-space } (X \Rightarrow_Q X \Rightarrow_Q count\text{-space } UNIV)$
 shows $(=) \in qbs\text{-space } (list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q count\text{-space } UNIV)$
proof –
 have [*simp*]: $(=) = (\lambda xs ys. length\ xs = length\ ys \wedge list\text{-all } (case\text{-prod } (=)) (zip\ xs\ ys))$
 using *Ball-set list-eq-iff-zip-eq* by *fastforce*
 show ?*thesis*
 by *simp*
qed

lemma *insort-key-qbs-morphism*[*qbs*]:
 shows $insort\text{-key} \in qbs\text{-space } ((X \Rightarrow_Q (borel_Q :: 'b :: \{second\text{-countable-topology}, linorder\text{-topology}\} quasi\text{-borel})) \Rightarrow_Q X \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$ (**is** ?*g1*)
 and $insort\text{-key} \in qbs\text{-space } ((X \Rightarrow_Q count\text{-space}_Q (UNIV :: (- :: countable) set)) \Rightarrow_Q X \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$ (**is** ?*g2*)
proof –
 have [*simp*]: $insort\text{-key} = (\lambda f x. rec\text{-list } [x] (\lambda y ys l. if\ f\ x \leq f\ y\ then\ x\#\#y\#ys\ else\ y\#\#l))$
 apply *standard+*
 subgoal for $f\ x\ l$
 by(*induction l*) *simp-all*
done
 show ?*g1* ?*g2*
 by *simp-all*
qed

lemma *sort-key-qbs-morphism*[*qbs*]:
 shows $sort\text{-key} \in qbs\text{-space } ((X \Rightarrow_Q (borel_Q :: 'b :: \{second\text{-countable-topology}, linorder\text{-topology}\} quasi\text{-borel})) \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
 and $sort\text{-key} \in qbs\text{-space } ((X \Rightarrow_Q count\text{-space}_Q (UNIV :: (- :: countable) set)) \Rightarrow_Q list\text{-qbs } X \Rightarrow_Q list\text{-qbs } X)$
 unfolding *sort-key-def* by *simp-all*

lemma *sort-qbs-morphism*[*qbs*]:
 shows $sort \in list\text{-qbs } (borel_Q :: 'b :: \{second\text{-countable-topology}, linorder\text{-topology}\} quasi\text{-borel}) \rightarrow_Q list\text{-qbs } borel_Q$
 and $sort \in list\text{-qbs } (count\text{-space}_Q (UNIV :: (- :: countable) set)) \rightarrow_Q list\text{-qbs } (count\text{-space}_Q UNIV)$
 by *simp-all*

3.3.4 Morphism Pred

abbreviation $qbs\text{-}pred\ X\ P \equiv P \in X \rightarrow_Q qbs\text{-}count\text{-}space\ (UNIV\ ::\ bool\ set)$

lemma $qbs\text{-}pred\text{-}iff\text{-}measurable\text{-}pred$:
 $qbs\text{-}pred\ X\ P = Measurable.pred\ (qbs\text{-}to\text{-}measure\ X)\ P$
by ($simp\ add$: $lr\text{-}adjunction\text{-}correspondence$)

lemma(**in** $standard\text{-}borel$) $qbs\text{-}pred\text{-}iff\text{-}measurable\text{-}pred$:
 $qbs\text{-}pred\ (measure\text{-}to\text{-}qbs\ M)\ P = Measurable.pred\ M\ P$
by ($simp\ add$: $qbs\text{-}pred\text{-}iff\text{-}measurable\text{-}pred\ measurable\text{-}cong\text{-}sets[OF\ lr\text{-}sets\text{-}ident\ refl]$)

lemma $qbs\text{-}pred\text{-}iff\text{-}sets$:
 $\{x \in space\ (qbs\text{-}to\text{-}measure\ X).\ P\ x\} \in sets\ (qbs\text{-}to\text{-}measure\ X) \longleftrightarrow qbs\text{-}pred\ X\ P$
by ($simp\ add$: $Measurable.pred\text{-}def\ lr\text{-}adjunction\text{-}correspondence\ space\text{-}L$)

lemma
assumes $[qbs]: P \in X \rightarrow_Q Y \Rightarrow_Q qbs\text{-}count\text{-}space\ UNIV\ f \in X \rightarrow_Q Y$
shows $indicator\text{-}qbs\text{-}morphism''$: $(\lambda x. indicator\ \{y. P\ x\ y\}\ (f\ x)) \in X \rightarrow_Q qbs\text{-}borel$ (**is** $?g1$)
and $indicator\text{-}qbs\text{-}morphism''$: $(\lambda x. indicator\ \{y \in qbs\text{-}space\ Y.\ P\ x\ y\}\ (f\ x)) \in X \rightarrow_Q qbs\text{-}borel$ (**is** $?g2$)
proof –
have $[simp]: \{x \in qbs\text{-}space\ X.\ P\ x\ (f\ x)\} = \{x \in qbs\text{-}space\ X.\ f\ x \in qbs\text{-}space\ Y \wedge P\ x\ (f\ x)\}$
using $qbs\text{-}morphism\text{-}space[OF\ assms(2)]$ **by** $blast$
show $?g1\ ?g2$
using $qbs\text{-}morphism\text{-}app[OF\ assms,\ simplified\ qbs\text{-}pred\text{-}iff\text{-}sets[symmetric]]\ qbs\text{-}morphism\text{-}space[OF\ assms(2)]$
by ($auto\ intro!$: $borel\text{-}measurable\text{-}indicator'\ simp$: $lr\text{-}adjunction\text{-}correspondence\ space\text{-}L$)
qed

lemma
assumes $[qbs]: P \in X \rightarrow_Q Y \Rightarrow_Q qbs\text{-}count\text{-}space\ UNIV$
shows $indicator\text{-}qbs\text{-}morphism[qbs]$: $(\lambda x. indicator\ \{y \in qbs\text{-}space\ Y.\ P\ x\ y\}) \in X \rightarrow_Q Y \Rightarrow_Q qbs\text{-}borel$ (**is** $?g1$)
and $indicator\text{-}qbs\text{-}morphism'$: $(\lambda x. indicator\ \{y. P\ x\ y\}) \in X \rightarrow_Q Y \Rightarrow_Q qbs\text{-}borel$ (**is** $?g2$)
proof –
note $indicator\text{-}qbs\text{-}morphism''[qbs]\ indicator\text{-}qbs\text{-}morphism''[qbs]$
show $?g1\ ?g2$
by ($auto\ intro!$: $curry\text{-}preserves\text{-}morphisms[OF\ pair\text{-}qbs\text{-}morphismI]\ simp$: $qbs\text{-}Mx\text{-}is\text{-}morphisms$)
qed

lemma $indicator\text{-}qbs[qbs]$:
assumes $qbs\text{-}pred\ X\ P$
shows $indicator\ \{x. P\ x\} \in X \rightarrow_Q qbs\text{-}borel$
using $assms$ **by** ($auto\ simp$: $lr\text{-}adjunction\text{-}correspondence$)

lemma *All-qbs-pred*[qbs]: *qbs-pred* (*count-space*_Q (*UNIV* :: ('a :: countable) set) \Rightarrow_Q *count-space*_Q *UNIV*) *All*
proof(*rule qbs-morphismI*)
 fix *a* :: *real* \Rightarrow 'a \Rightarrow *bool*
 assume *a* \in *qbs-Mx* (*count-space*_Q *UNIV* \Rightarrow_Q *count-space*_Q *UNIV*)
 hence [*measurable*]: $\bigwedge f g. f \in \text{borel-measurable borel} \implies g \in \text{borel} \rightarrow_M \text{count-space UNIV} \implies (\lambda x::\text{real}. a (f x) (g x)) \in \text{borel} \rightarrow_M \text{count-space UNIV}$
 by(*auto simp add: exp-qbs-Mx qbs-Mx-R*)
 show *All* \circ *a* \in *qbs-Mx* (*count-space*_Q *UNIV*)
 by(*simp add: comp-def qbs-Mx-R*)
qed

lemma *Ex-qbs-pred*[qbs]: *qbs-pred* (*count-space*_Q (*UNIV* :: ('a :: countable) set) \Rightarrow_Q *count-space*_Q *UNIV*) *Ex*
proof(*rule qbs-morphismI*)
 fix *a* :: *real* \Rightarrow 'a \Rightarrow *bool*
 assume *a* \in *qbs-Mx* (*count-space*_Q *UNIV* \Rightarrow_Q *count-space*_Q *UNIV*)
 hence [*measurable*]: $\bigwedge f g. f \in \text{borel-measurable borel} \implies g \in \text{borel} \rightarrow_M \text{count-space UNIV} \implies (\lambda x::\text{real}. a (f x) (g x)) \in \text{borel} \rightarrow_M \text{count-space UNIV}$
 by(*auto simp add: exp-qbs-Mx qbs-Mx-R*)
 show *Ex* \circ *a* \in *qbs-Mx* (*count-space*_Q *UNIV*)
 by(*simp add: comp-def qbs-Mx-R*)
qed

lemma *Ball-qbs-pred-countable*:
assumes $\bigwedge i::'a :: \text{countable}. i \in I \implies \text{qbs-pred } X (P i)$
shows *qbs-pred* *X* ($\lambda x. \forall x \in I. P i x$)
using *assms* **by**(*simp add: qbs-pred-iff-measurable-pred*)

lemma *Ball-qbs-pred*:
assumes *finite* *I* $\bigwedge i. i \in I \implies \text{qbs-pred } X (P i)$
shows *qbs-pred* *X* ($\lambda x. \forall x \in I. P i x$)
using *assms* **by**(*simp add: qbs-pred-iff-measurable-pred*)

lemma *Bex-qbs-pred-countable*:
assumes $\bigwedge i::'a :: \text{countable}. i \in I \implies \text{qbs-pred } X (P i)$
shows *qbs-pred* *X* ($\lambda x. \exists x \in I. P i x$)
using *assms* **by**(*simp add: qbs-pred-iff-measurable-pred*)

lemma *Bex-qbs-pred*:
assumes *finite* *I* $\bigwedge i. i \in I \implies \text{qbs-pred } X (P i)$
shows *qbs-pred* *X* ($\lambda x. \exists x \in I. P i x$)
using *assms* **by**(*simp add: qbs-pred-iff-measurable-pred*)

lemma *qbs-morphism-If-sub-qbs*:
assumes [*qbs*]: *qbs-pred* *X* *P*
 and [*qbs*]: *f* \in *sub-qbs* *X* $\{x \in \text{qbs-space } X. P x\} \rightarrow_Q Y$ *g* \in *sub-qbs* *X* $\{x \in \text{qbs-space } X. \neg P x\} \rightarrow_Q Y$

```

  shows  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \in X \rightarrow_Q \ Y$ 
proof(rule qbs-morphismI)
  fix  $\alpha$ 
  assume  $h:\alpha \in \text{qbs-Mx } X$ 
  interpret standard-borel-ne borel :: real measure by simp
  have [measurable]: Measurable.pred borel  $(\lambda x. P \ (\alpha \ x))$ 
  using  $h$  by(simp add: qbs-pred-iff-measurable-pred[symmetric] qbs-Mx-is-morphisms)
  consider qbs-space  $X = \{\}$ 
    |  $\{x \in \text{qbs-space } X. \neg P \ x\} = \text{qbs-space } X$ 
    |  $\{x \in \text{qbs-space } X. P \ x\} = \text{qbs-space } X$ 
    |  $\{x \in \text{qbs-space } X. P \ x\} \neq \{\}$   $\{x \in \text{qbs-space } X. \neg P \ x\} \neq \{\}$  by blast
  then show  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \circ \alpha \in \text{qbs-Mx } Y$  (is ?f  $\in$  -)
proof cases
  case 1
  with  $h$  show ?thesis
    by(simp add: qbs-empty-equiv)
next
  case 2
  have [simp]:  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \circ \alpha = g \circ \alpha$ 
    by standard (use qbs-Mx-to-X[OF  $h$ ] 2 in auto)
  show ?thesis
    using 2 qbs-morphism-Mx[OF assms(3)]  $h$  by(simp add: sub-qbs-ident)
next
  case 3
  have [simp]:  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \circ \alpha = f \circ \alpha$ 
    by standard (use qbs-Mx-to-X[OF  $h$ ] 3 in auto)
  show ?thesis
    using 3 qbs-morphism-Mx[OF assms(2)]  $h$  by(simp add: sub-qbs-ident)
next
  case 4
  then obtain  $x0 \ x1$  where
     $x0:x0 \in \text{qbs-space } X \ P \ x0$  and  $x1:x1 \in \text{qbs-space } X \ \neg P \ x1$ 
    by blast
  define  $a0$  where  $a0 = (\lambda r. \text{if } P \ (\alpha \ r) \ \text{then } \alpha \ r \ \text{else } x0)$ 
  define  $a1$  where  $a1 = (\lambda r. \text{if } \neg P \ (\alpha \ r) \ \text{then } \alpha \ r \ \text{else } x1)$ 
  have  $a0 \in \text{qbs-Mx} \ (\text{sub-qbs } X \ \{x \in \text{qbs-space } X. P \ x\})$   $a1 \in \text{qbs-Mx} \ (\text{sub-qbs } X \ \{x \in \text{qbs-space } X. \neg P \ x\})$ 
    using  $x0 \ x1$  qbs-Mx-to-X[OF  $h$ ]  $h$ 
    by(auto simp: sub-qbs-Mx a0-def a1-def intro!: qbs-closed3-dest2'[of UNIV  $\lambda r. P \ (\alpha \ r) \ \lambda b \ r. \text{if } b \ \text{then } \alpha \ r \ \text{else } x0$ ] (simp-all add: qbs-Mx-is-morphisms)
    from qbs-morphism-Mx[OF assms(2) this(1)] qbs-morphism-Mx[OF assms(3) this(2)]
  have  $h0:(\lambda r. f \ (a0 \ r)) \in \text{qbs-Mx } Y$   $(\lambda r. g \ (a1 \ r)) \in \text{qbs-Mx } Y$ 
    by (simp-all add: comp-def)
  have [simp]:  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \circ \alpha = (\lambda r. \text{if } P \ (\alpha \ r) \ \text{then } f \ (a0 \ r) \ \text{else } g \ (a1 \ r))$ 
    by standard (auto simp: comp-def a0-def a1-def)
  show  $(\lambda x. \text{if } P \ x \ \text{then } f \ x \ \text{else } g \ x) \circ \alpha \in \text{qbs-Mx } Y$ 
    using  $h \ h0$  by(simp add: qbs-Mx-is-morphisms)

```

qed
qed

3.3.5 The Adjunction w.r.t. Ordering

lemma *l-mono: mono qbs-to-measure*

proof

fix $X Y :: 'a \text{ quasi-borel}$

show $X \leq Y \implies \text{qbs-to-measure } X \leq \text{qbs-to-measure } Y$

proof(*induction rule: less-eq-quasi-borel.induct*)

case (1 $X Y$)

then show ?case

by(*simp add: less-eq-measure.intros(1) space-L*)

next

case (2 $X Y$)

then have $\text{sigma-Mx } X \subseteq \text{sigma-Mx } Y$

by(*auto simp add: sigma-Mx-def*)

then consider $\text{sigma-Mx } X \subset \text{sigma-Mx } Y \mid \text{sigma-Mx } X = \text{sigma-Mx } Y$

by *auto*

then show ?case

apply(*cases*)

apply(*rule less-eq-measure.intros(2)*)

apply(*simp-all add: 2 space-L sets-L*)

by(*rule less-eq-measure.intros(3),simp-all add: 2 sets-L space-L emeasure-L*)

qed

qed

lemma *r-mono: mono measure-to-qbs*

proof

fix $M N :: 'a \text{ measure}$

show $M \leq N \implies \text{measure-to-qbs } M \leq \text{measure-to-qbs } N$

proof(*induction rule: less-eq-measure.inducts*)

case (1 $M N$)

then show ?case

by(*simp add: less-eq-quasi-borel.intros(1) qbs-space-R*)

next

case (2 $M N$)

then have $(\text{borel} :: \text{real measure}) \rightarrow_M N \subseteq \text{borel} \rightarrow_M M$

by(*simp add: measurable-mono*)

then consider $(\text{borel} :: \text{real measure}) \rightarrow_M N \subset \text{borel} \rightarrow_M M \mid (\text{borel} :: \text{real measure}) \rightarrow_M N = \text{borel} \rightarrow_M M$

by *auto*

then show ?case

by *cases (rule less-eq-quasi-borel.intros(2),simp-all add: 2 qbs-space-R qbs-Mx-R)+*

next

case (3 $M N$)

then show ?case

apply –

by(*rule less-eq-quasi-borel.intros(2)*) (*simp-all add: measurable-mono qbs-space-R*)


```

qbs-Mx-R)
  qed
qed

lemma rl-order-adjunction:
   $X \leq \text{qbs-to-measure } Y \iff \text{measure-to-qbs } X \leq Y$ 
proof
  assume 1:  $X \leq \text{qbs-to-measure } Y$ 
  then show  $\text{measure-to-qbs } X \leq Y$ 
  proof(induction rule: less-eq-measure.cases)
    case (1 M N)
    then have [simp]:  $\text{qbs-space } Y = \text{space } N$ 
      by(simp add: 1(2)[symmetric] space-L)
    show ?case
      by(rule less-eq-quasi-borel.intros(1),simp add: 1 qbs-space-R)
  next
    case (2 M N)
    then have [simp]:  $\text{qbs-space } Y = \text{space } N$ 
      by(simp add: 2(2)[symmetric] space-L)
    show ?case
  proof(rule less-eq-quasi-borel.intros(2))
    show  $\text{qbs-Mx } Y \subseteq \text{qbs-Mx } (\text{measure-to-qbs } X)$ 
      unfolding qbs-Mx-R
    proof
      fix  $\alpha$ 
      assume  $h: \alpha \in \text{qbs-Mx } Y$ 
      show  $\alpha \in \text{borel} \rightarrow_M X$ 
      proof(rule measurableI)
        show  $\bigwedge x. \alpha \ x \in \text{space } X$ 
          using qbs-Mx-to-X[OF h] by (auto simp add: 2)
      next
        fix  $A$ 
        assume  $A \in \text{sets } X$ 
        then have  $A \in \text{sets } (\text{qbs-to-measure } Y)$ 
          using 2 by auto
        then obtain  $U$  where
           $hu: A = U \cap \text{space } N$  ( $\forall \alpha \in \text{qbs-Mx } Y. \alpha -' U \in \text{sets borel}$ )
          by(auto simp add: sigma-Mx-def sets-L)
        have  $\alpha -' A = \alpha -' U$ 
          using qbs-Mx-to-X[OF h] by(auto simp add: hu)
        thus  $\alpha -' A \cap \text{space borel} \in \text{sets borel}$ 
          using h hu(2) by simp
      qed
    qed
  qed(auto simp: 2 qbs-space-R)
next
  case (3 M N)
  then have [simp]:  $\text{qbs-space } Y = \text{space } N$ 
    by(simp add: 3(2)[symmetric] space-L)

```

```

show ?case
proof(rule less-eq-quasi-borel.intros(2))
  show qbs-Mx Y  $\subseteq$  qbs-Mx (measure-to-qbs X)
    unfolding qbs-Mx-R
  proof
    fix  $\alpha$ 
    assume  $h:\alpha \in$  qbs-Mx Y
    show  $\alpha \in$  borel  $\rightarrow_M$  X
    proof(rule measurableI)
      show  $\bigwedge x. \alpha x \in$  space X
        using qbs-Mx-to-X[OF h] by(auto simp: 3)
    next
      fix A
      assume  $A \in$  sets X
      then have  $A \in$  sets (qbs-to-measure Y)
        using 3 by auto
      then obtain U where
         $hu:A = U \cap$  space N ( $\forall \alpha \in$  qbs-Mx Y.  $\alpha -' U \in$  sets borel)
        by(auto simp add: sigma-Mx-def sets-L)
      have  $\alpha -' A = \alpha -' U$ 
        using qbs-Mx-to-X[OF h] by(auto simp add: hu)
      thus  $\alpha -' A \cap$  space borel  $\in$  sets borel
        using h hu(2) by simp
      qed
    qed
  qed(auto simp: 3 qbs-space-R)
qed
next
assume measure-to-qbs X  $\leq$  Y
then show X  $\leq$  qbs-to-measure Y
proof(induction rule: less-eq-quasi-borel.cases)
  case (1 A B)
    have [simp]: space X = qbs-space A
      by(simp add: 1(1)[symmetric] qbs-space-R)
    show ?case
      by(rule less-eq-measure.intros(1)) (simp add: 1 space-L)
  next
    case (2 A B)
      then have  $hmy:qbs-Mx Y \subseteq$  borel  $\rightarrow_M$  X
        using qbs-Mx-R by blast
      have [simp]: space X = qbs-space A
        by(simp add: 2(1)[symmetric] qbs-space-R)
      have sets X  $\subseteq$  sigma-Mx Y
      proof
        fix U
        assume  $hu:U \in$  sets X
        show U  $\in$  sigma-Mx Y
          unfolding sigma-Mx-def
        proof(safe intro!: exI[where  $x=U$ ])

```

```

show  $\bigwedge x. x \in U \implies x \in \text{qbs-space } Y$ 
  using sets.sets-into-space[OF hu]
  by(auto simp add: 2)
next
fix  $\alpha$ 
assume  $\alpha \in \text{qbs-Mx } Y$ 
then have  $\alpha \in \text{borel} \rightarrow_M X$ 
  using hmy by(auto)
thus  $\alpha - ' U \in \text{sets borel}$ 
  using hu by(simp add: measurable-sets-borel)
qed
qed
then consider sets  $X = \text{sigma-Mx } Y \mid \text{sets } X \subset \text{sigma-Mx } Y$ 
  by auto
then show ?case
proof cases
  case 1
  show ?thesis
  proof(rule less-eq-measure.intros(3))
  show  $\text{emeasure } X \leq \text{emeasure } (\text{qbs-to-measure } Y)$ 
    unfolding emeasure-L
  proof(rule le-funI)
  fix  $U$ 
  consider  $U = \{\} \mid U \notin \text{sigma-Mx } Y \mid U \neq \{\} \wedge U \in \text{sigma-Mx } Y$ 
    by auto
  then show  $\text{emeasure } X U \leq (\text{if } U = \{\} \vee U \notin \text{sigma-Mx } Y \text{ then } 0 \text{ else}$ 
 $\infty)$ 
    proof cases
      case 1
      then show ?thesis by simp
    next
      case h:2
      then have  $U \notin \text{sigma-Mx } A$ 
        using qbs-Mx-sigma-Mx-contr[OF 2(3)[symmetric] 2(4)] 2(2) by auto
      hence  $U \notin \text{sets } X$ 
        using lr-sets 2(1) sets-L by blast
      thus ?thesis
        by(simp add: h emeasure-notin-sets)
    next
      case 3
      then show ?thesis
        by simp
    qed
  qed
qed(simp-all add: 1 2 space-L sets-L)
next
case h2:2
show ?thesis
  by(rule less-eq-measure.intros(2)) (simp add: space-L 2, simp add: h2 sets-L)

```

```

    qed
  qed
qed
end

```

4 The S-Finite Measure Monad

```

theory Monad-QuasiBorel
  imports
    Measure-QuasiBorel-Adjunction
    Kernels

```

```
begin
```

4.1 The S-Finite Measure Monad

4.1.1 Space of S-Finite Measures

```

locale in-Mx =
  fixes X :: 'a quasi-borel
  and  $\alpha$  :: real  $\Rightarrow$  'a
  assumes in-Mx[simp]:  $\alpha \in$  qbs-Mx X
begin

```

```

lemma  $\alpha$ -measurable[measurable]:  $\alpha \in$  borel  $\rightarrow_M$  qbs-to-measure X
  using in-Mx qbs-Mx-subset-of-measurable by blast

```

```

lemma  $\alpha$ -qbs-morphism[qbs]:  $\alpha \in$  qbs-borel  $\rightarrow_Q$  X
  using in-Mx by(simp only: qbs-Mx-is-morphisms)

```

```

lemma X-not-empty: qbs-space X  $\neq$  {}
  using in-Mx by(auto simp: qbs-empty-equiv simp del: in-Mx)

```

```

lemma inverse-UNIV[simp]:  $\alpha - ' (qbs-space X) = UNIV$ 
  by fastforce

```

```
end
```

```

locale qbs-s-finite = in-Mx X  $\alpha$  + s-finite-measure  $\mu$ 
  for X :: 'a quasi-borel and  $\alpha$  and  $\mu$  :: real measure +
  assumes mu-sets[measurable-cong]: sets  $\mu =$  sets borel
begin

```

```

lemma mu-not-empty: space  $\mu \neq$  {}
  by(simp add: sets-eq-imp-space-eq[OF mu-sets])

```

```
end
```

```

lemma qbs-s-finite-All:
  assumes  $\alpha \in \text{qbs-Mx } X \text{ s-finite-kernel } M \text{ borel } k \ x \in \text{space } M$ 
  shows qbs-s-finite  $X \ \alpha \ (k \ x)$ 
proof –
  interpret s-finite-kernel  $M \ \text{borel } k$  by fact
  show ?thesis
  using assms(1,3) image-s-finite-measure[OF assms(3)] by(auto simp: qbs-s-finite-def
in-Mx-def qbs-s-finite-axioms-def kernel-sets)
qed

locale qbs-prob = in-Mx  $X \ \alpha$  + real-distribution  $\mu$ 
  for  $X :: 'a \text{ quasi-borel}$  and  $\alpha \ \mu$ 
begin

lemma qbs-s-finite: qbs-s-finite  $X \ \alpha \ \mu$ 
  by(auto simp: qbs-s-finite-def qbs-s-finite-axioms-def in-Mx-def s-finite-measure-prob)

sublocale qbs-s-finite by(rule qbs-s-finite)

end

lemma(in qbs-s-finite) qbs-probI: prob-space  $\mu \implies \text{qbs-prob } X \ \alpha \ \mu$ 
  by(auto simp: qbs-prob-def in-Mx-def real-distribution-def real-distribution-axioms-def
mu-sets)

locale pair-qbs-s-finites = pq1: qbs-s-finite  $X \ \alpha \ \mu$  + pq2: qbs-s-finite  $Y \ \beta \ \nu$ 
  for  $X :: 'a \text{ quasi-borel}$  and  $\alpha \ \mu$  and  $Y :: 'b \text{ quasi-borel}$  and  $\beta \ \nu$ 
begin

lemma ab-measurable[measurable]: map-prod  $\alpha \ \beta \in \text{borel } \otimes_M \text{ borel} \rightarrow_M \text{qbs-to-measure}$ 
 $(X \ \otimes_Q \ Y)$ 
proof –
  have map-prod  $\alpha \ \beta \in \text{qbs-to-measure} (\text{measure-to-qbs} (\text{borel } \otimes_M \text{ borel})) \rightarrow_M$ 
 $\text{qbs-to-measure} (X \ \otimes_Q \ Y)$ 
  by(auto intro!: set-mp[OF l-preserves-morphisms] simp: r-preserves-product)
  moreover have sets  $(\text{qbs-to-measure} (\text{measure-to-qbs} (\text{borel } \otimes_M \text{ borel}))) = \text{sets}$ 
 $((\text{borel } \otimes_M \text{ borel}) :: (\text{real} \times \text{real}) \text{ measure})$ 
  by(auto intro!: standard-borel.lr-sets-ident pair-standard-borel-ne standard-borel-ne.standard-borel)
  ultimately show ?thesis by simp
qed

end

locale pair-qbs-probs = pq1: qbs-prob  $X \ \alpha \ \mu$  + pq2: qbs-prob  $Y \ \beta \ \nu$ 
  for  $X :: 'a \text{ quasi-borel}$  and  $\alpha \ \mu$  and  $Y :: 'b \text{ quasi-borel}$  and  $\beta \ \nu$ 
begin
sublocale pair-qbs-s-finites
  by standard
end

```

locale *pair-qbs-s-finite* = *pq1*: *qbs-s-finite* $X \alpha \mu$ + *pq2*: *qbs-s-finite* $X \beta \nu$
for $X :: 'a \text{ quasi-borel}$ **and** $\alpha \mu$ **and** $\beta \nu$
begin
sublocale *pair-qbs-s-finites* $X \alpha \mu X \beta \nu$
by *standard*
end

locale *pair-qbs-prob* = *pq1*: *qbs-prob* $X \alpha \mu$ + *pq2*: *qbs-prob* $X \beta \nu$
for $X :: 'a \text{ quasi-borel}$ **and** $\alpha \mu$ **and** $\beta \nu$
begin

sublocale *pair-qbs-s-finite* $X \alpha \mu \beta \nu$
by *standard*

sublocale *pair-qbs-probs* $X \alpha \mu X \beta \mu$
by *standard*

end

type-synonym $'a \text{ qbs-s-finite-t} = 'a \text{ quasi-borel} * (\text{real} \Rightarrow 'a) * \text{real measure}$

definition *qbs-s-finite-eq* :: $['a \text{ qbs-s-finite-t}, 'a \text{ qbs-s-finite-t}] \Rightarrow \text{bool}$ **where**

qbs-s-finite-eq $p1 p2 \equiv$
 $(\text{let } (X, \alpha, \mu) = p1;$
 $(Y, \beta, \nu) = p2 \text{ in}$
 $\text{qbs-s-finite } X \alpha \mu \wedge \text{qbs-s-finite } Y \beta \nu \wedge X = Y \wedge$
 $\text{distr } \mu (\text{qbs-to-measure } X) \alpha = \text{distr } \nu (\text{qbs-to-measure } Y) \beta)$

definition *qbs-s-finite-eq'* :: $['a \text{ qbs-s-finite-t}, 'a \text{ qbs-s-finite-t}] \Rightarrow \text{bool}$ **where**

qbs-s-finite-eq' $p1 p2 \equiv$
 $(\text{let } (X, \alpha, \mu) = p1;$
 $(Y, \beta, \nu) = p2 \text{ in}$
 $\text{qbs-s-finite } X \alpha \mu \wedge \text{qbs-s-finite } Y \beta \nu \wedge X = Y \wedge$
 $(\forall f \in X \rightarrow_Q (\text{qbs-borel} :: \text{ennreal quasi-borel}). (\int^+ x. f (\alpha x) \partial \mu) = (\int^+ x. f$
 $(\beta x) \partial \nu)))$

lemma(**in** *qbs-s-finite*)

shows *qbs-s-finite-eq-refl*[*simp*]: *qbs-s-finite-eq* $(X, \alpha, \mu) (X, \alpha, \mu)$
and *qbs-s-finite-eq'-refl*[*simp*]: *qbs-s-finite-eq'* $(X, \alpha, \mu) (X, \alpha, \mu)$
by(*simp-all add: qbs-s-finite-eq-def qbs-s-finite-eq'-def qbs-s-finite-axioms*)

lemma(**in** *pair-qbs-s-finite*)

shows *qbs-s-finite-eq-intro*: $\text{distr } \mu (\text{qbs-to-measure } X) \alpha = \text{distr } \nu (\text{qbs-to-measure } X) \beta \implies \text{qbs-s-finite-eq } (X, \alpha, \mu) (X, \beta, \nu)$
and *qbs-s-finite-eq'-intro*: $(\bigwedge f. f \in X \rightarrow_Q \text{qbs-borel} \implies (\int^+ x. f (\alpha x) \partial \mu) = (\int^+ x. f (\beta x) \partial \nu)) \implies \text{qbs-s-finite-eq}' (X, \alpha, \mu) (X, \beta, \nu)$
by(*simp-all add: qbs-s-finite-eq-def qbs-s-finite-eq'-def pq1.qbs-s-finite-axioms pq2.qbs-s-finite-axioms*)

lemma *qbs-s-finite-eq-dest*:

assumes *qbs-s-finite-eq* (X, α, μ) (Y, β, ν)
shows *qbs-s-finite* $X \alpha \mu$ *qbs-s-finite* $Y \beta \nu$ $Y = X \text{ distr } \mu$ (*qbs-to-measure* X)
 $\alpha = \text{distr } \nu$ (*qbs-to-measure* X) β
using *assms* **by**(*auto simp: qbs-s-finite-eq-def*)

lemma *qbs-s-finite-eq'-dest*:

assumes *qbs-s-finite-eq'* (X, α, μ) (Y, β, ν)
shows *qbs-s-finite* $X \alpha \mu$ *qbs-s-finite* $Y \beta \nu$ $Y = X \wedge f. f \in X \rightarrow_Q \text{qbs-borel} \implies$
 $(\int^{+x}. f (\alpha x) \partial \mu) = (\int^{+x}. f (\beta x) \partial \nu)$
using *assms* **by**(*auto simp: qbs-s-finite-eq'-def*)

lemma(**in** *qbs-prob*) *qbs-s-finite-eq-qbs-prob-cong*:

assumes *qbs-s-finite-eq* (X, α, μ) (Y, β, ν)
shows *qbs-prob* $Y \beta \nu$

proof –

interpret *qs: pair-qbs-s-finites* $X \alpha \mu$ $Y \beta \nu$

using *assms*(1) **by**(*auto simp: qbs-s-finite-eq-def pair-qbs-s-finites-def*)

show *?thesis*

by(*auto intro!: qs.pq2.qbs-probI prob-space-distrD[of \beta - qbs-to-measure Y]*)

(*auto simp: qbs-s-finite-eq-dest*(3)[*OF assms*] *qbs-s-finite-eq-dest*(4)[*OF assms, symmetric*]
intro!: prob-space-distr)

qed

lemma

shows *qbs-s-finite-eq-symp: symp qbs-s-finite-eq*

and *qbs-s-finite-eq-transp: transp qbs-s-finite-eq*

by(*simp-all add: qbs-s-finite-eq-def transp-def symp-def*)

quotient-type $'a \text{ qbs-measure} = 'a \text{ qbs-s-finite-t} / \text{partial: qbs-s-finite-eq}$

morphisms *rep-qbs-measure qbs-measure*

proof(*rule part-equivI*)

let $?U = \text{UNIV} :: 'a \text{ set}$

let $?Uf = \text{UNIV} :: (\text{real} \Rightarrow 'a) \text{ set}$

let $?f = (\lambda-. \text{undefined}) :: \text{real} \Rightarrow 'a$

have *qbs-s-finite* (*Abs-quasi-borel* ($?U, ?Uf$)) $?f$ (*return borel 0*)

unfolding *qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def*

proof *safe*

have *Rep-quasi-borel* (*Abs-quasi-borel* ($?U, ?Uf$)) = ($?U, ?Uf$)

using *Abs-quasi-borel-inverse* **by** (*auto simp add: qbs-closed1-def qbs-closed2-def qbs-closed3-def is-quasi-borel-def*)

thus $(\lambda-. \text{undefined}) \in \text{qbs-Mx}$ (*Abs-quasi-borel* ($?U, ?Uf$))

by(*simp add: qbs-Mx-def*)

next

show *s-finite-measure* (*return borel 0*)

by(*auto intro!: sigma-finite-measure.s-finite-measure prob-space-imp-sigma-finite prob-space-return*)

qed *simp-all*

thus $\exists x :: 'a \text{ qbs-s-finite-t. qbs-s-finite-eq } x x$

by(*auto simp: qbs-s-finite-eq-def intro!: exI[where x=(Abs-quasi-borel (?U, ?Uf),*

?f, return borel 0))

qed(simp-all add: qbs-s-finite-eq-symp qbs-s-finite-eq-transp)

interpretation qbs-measure : quot-type qbs-s-finite-eq Abs-qbs-measure Rep-qbs-measure

using Abs-qbs-measure-inverse Rep-qbs-measure

by(simp add: quot-type-def equivp-implies-part-equivp qbs-measure-equivp Rep-qbs-measure-inverse Rep-qbs-measure-inject) blast

syntax

-qbs-measure :: 'a quasi-borel \Rightarrow (real \Rightarrow 'a) \Rightarrow real measure \Rightarrow 'a qbs-measure
($\llbracket \cdot, / \cdot, / \cdot \rrbracket_{sfin}$)

translations

$\llbracket X, \alpha, \mu \rrbracket_{sfin} \equiv \text{CONST } \text{qbs-measure } (X, \alpha, \mu)$

lemma rep-qbs-s-finite-measure': $\exists X \alpha \mu. p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \wedge \text{qbs-s-finite } X \alpha \mu$
by(rule qbs-measure.abs-induct,auto simp add: qbs-s-finite-eq-def)

lemma rep-qbs-s-finite-measure:

obtains $X \alpha \mu$ **where** $p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu$

using that rep-qbs-s-finite-measure' **by** blast

definition qbs-null-measure :: 'a quasi-borel \Rightarrow 'a qbs-measure **where**

qbs-null-measure $X \equiv \llbracket X, \text{SOME } a. a \in \text{qbs-Mx } X, \text{null-measure borel} \rrbracket_{sfin}$

lemma qbs-null-measure-s-finite: qbs-space $X \neq \{\}$ \implies qbs-s-finite X (SOME $a. a \in \text{qbs-Mx } X$) (null-measure borel)

by(auto simp: qbs-empty-equiv qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def some-in-eq intro!: finite-measure.s-finite-measure-finite-measure finite-measureI)

lemma(in qbs-s-finite) in-Rep-qbs-measure':

assumes qbs-s-finite-eq $(X, \alpha, \mu) (X', \alpha', \mu')$

shows $(X', \alpha', \mu') \in \text{Rep-qbs-measure } \llbracket X, \alpha, \mu \rrbracket_{sfin}$

by (metis assms mem-Collect-eq qbs-s-finite-eq-refl qbs-measure-def qbs-measure.abs-def qbs-measure.abs-inverse)

lemmas(in qbs-s-finite) in-Rep-qbs-measure = in-Rep-qbs-measure'[OF qbs-s-finite-eq-refl]

lemma(in qbs-s-finite) if-in-Rep-qbs-measure:

assumes $(X', \alpha', \mu') \in \text{Rep-qbs-measure } \llbracket X, \alpha, \mu \rrbracket_{sfin}$

shows $X' = X$

qbs-s-finite $X' \alpha' \mu'$

qbs-s-finite-eq $(X, \alpha, \mu) (X', \alpha', \mu')$

proof –

show $h: X' = X$

using assms qbs-measure.Rep-qbs-measure[of $\llbracket X, \alpha, \mu \rrbracket_{sfin}$]

by auto (metis mem-Collect-eq qbs-s-finite-eq-dest(3) qbs-s-finite-eq-refl qbs-measure-def qbs-measure.abs-def qbs-measure.abs-inverse)

next

show qbs-s-finite $X' \alpha' \mu'$


```

    using assms qbs-measure.Rep-qbs-measure[of  $\llbracket X, \alpha, \mu \rrbracket_{sfin}$ ]
    by (auto simp: qbs-s-finite-eq-dest(2))
next
show qbs-s-finite-eq  $(X, \alpha, \mu)$   $(X', \alpha', \mu')$ 
  using assms qbs-measure.Rep-qbs-measure[of  $\llbracket X, \alpha, \mu \rrbracket_{sfin}$ ]
  by auto (metis mem-Collect-eq qbs-s-finite-eq-dest(3) qbs-s-finite-eq-refl qbs-measure-def
qbs-measure.abs-def qbs-measure.abs-inverse)
qed

```

```

lemma qbs-s-finite-eq-1-imp-2:
  assumes qbs-s-finite-eq  $(X, \alpha, \mu)$   $(Y, \beta, \nu)$   $f \in X \rightarrow_Q (qbs-borel :: (- :: \{banach\})
quasi-borel)$ 
  shows  $(\int x. f (\alpha x) \partial\mu) = (\int x. f (\beta x) \partial\nu)$  (is ?lhs = ?rhs)
proof -
interpret pq : pair-qbs-s-finite  $X \alpha \mu \beta \nu$ 
  using assms by (auto intro!: pair-qbs-s-finite.intro simp: qbs-s-finite-eq-def)
have [measurable]:  $f \in qbs-to-measure X \rightarrow_M borel$ 
  using assms by (simp add: lr-adjunction-correspondence)
have ?lhs =  $(\int x. f x \partial(\text{distr } \mu (qbs-to-measure X) \alpha))$ 
  by (simp add: integral-distr)
also have ... =  $(\int x. f x \partial(\text{distr } \nu (qbs-to-measure X) \beta))$ 
  by (simp add: qbs-s-finite-eq-dest(4)[OF assms(1)])
also have ... = ?rhs
  by (simp add: integral-distr)
finally show ?thesis .
qed

```

```

lemma qbs-s-finite-eq-equiv: qbs-s-finite-eq = qbs-s-finite-eq'
proof (rule ext[OF ext])
show  $\bigwedge a b :: 'a \text{ qbs-s-finite-t. } qbs-s-finite-eq a b = qbs-s-finite-eq' a b$ 
proof safe
fix  $X Y :: 'a \text{ quasi-borel and } \alpha \beta \mu \nu$ 
{
  assume  $h: qbs-s-finite-eq (X, \alpha, \mu) (Y, \beta, \nu)$ 
  then interpret pq : pair-qbs-s-finite  $X \alpha \mu \beta \nu$ 
  by (auto intro!: pair-qbs-s-finite.intro simp: qbs-s-finite-eq-def)
  show qbs-s-finite-eq'  $(X, \alpha, \mu) (Y, \beta, \nu)$ 
  unfolding qbs-s-finite-eq-dest(3)[OF h]
proof (rule pq.qbs-s-finite-eq'-intro)
fix  $f :: 'a \Rightarrow \text{ennreal}$ 
  assume  $f: f \in X \rightarrow_Q qbs-borel$ 
  show  $(\int^+ x. f (\alpha x) \partial\mu) = (\int^+ x. f (\beta x) \partial\nu)$  (is ?lhs = ?rhs)
  proof -
  have ?lhs =  $(\int^+ x. f x \partial(\text{distr } \mu (qbs-to-measure X) \alpha))$ 
  by (rule nn-integral-distr[symmetric]) (use f lr-adjunction-correspondence
in auto)
  also have ... =  $(\int^+ x. f x \partial(\text{distr } \nu (qbs-to-measure X) \beta))$ 
  by (simp add: qbs-s-finite-eq-dest(4)[OF h])
  also have ... = ?rhs

```

```

    by(rule nn-integral-distr) (use f lr-adjunction-correspondence in auto)
    finally show ?thesis .
  qed
}
}
assume h:qbs-s-finite-eq' (X,α,μ) (Y,β,ν)
then interpret pq : pair-qbs-s-finite X α μ β ν
  by(auto intro!: pair-qbs-s-finite.intro simp: qbs-s-finite-eq'-def)
show qbs-s-finite-eq (X,α,μ) (Y,β,ν)
  unfolding qbs-s-finite-eq'-dest(β)[OF h]
proof(rule pq.qbs-s-finite-eq-intro[OF measure-eqI])
  fix U
  assume hu[measurable]:U ∈ sets (distr μ (qbs-to-measure X) α)
  show emeasure (distr μ (qbs-to-measure X) α) U = emeasure (distr ν
(qbs-to-measure X) β) U
    (is ?lhs = ?rhs)
  proof -
    have ?lhs = (∫+ x. indicator U x ∂ (distr μ (qbs-to-measure X) α))
      using hu by simp
    also have ... = (∫+ x. indicator U (α x) ∂μ)
      by(rule nn-integral-distr) (use hu in auto)
    also have ... = (∫+ x. indicator U (β x) ∂ν)
      by(auto intro!: qbs-s-finite-eq'-dest(α)[OF h] simp: lr-adjunction-correspondence)
    also have ... = (∫+ x. indicator U x ∂ (distr ν (qbs-to-measure X) β))
      by(rule nn-integral-distr[symmetric]) (use hu in auto)
    also have ... = ?rhs
      using hu by simp
    finally show ?thesis .
  qed
qed simp
}
qed
qed

```

lemma *qbs-s-finite-measure-eq*: $qbs-s-finite-eq (X, \alpha, \mu) (Y, \beta, \nu) \implies \llbracket X, \alpha, \mu \rrbracket_{sfin} = \llbracket Y, \beta, \nu \rrbracket_{sfin}$
 using *Quotient3-rel*[OF *Quotient3-qbs-measure*] **by** *blast*

lemma(in *pair-qbs-s-finite*) *qbs-s-finite-measure-eq*:
 $distr \mu (qbs-to-measure X) \alpha = distr \nu (qbs-to-measure X) \beta \implies \llbracket X, \alpha, \mu \rrbracket_{sfin} = \llbracket X, \beta, \nu \rrbracket_{sfin}$
 by(auto intro!: *qbs-s-finite-measure-eq qbs-s-finite-eq-intro*)

lemma(in *pair-qbs-s-finite*) *qbs-s-finite-measure-eq'*:
 $(\bigwedge f. f \in X \rightarrow_Q qbs-borel \implies (\int^+ x. f (\alpha x) \partial \mu) = (\int^+ x. f (\beta x) \partial \nu)) \implies \llbracket X, \alpha, \mu \rrbracket_{sfin} = \llbracket X, \beta, \nu \rrbracket_{sfin}$
 using *qbs-s-finite-eq'-intro[simplified qbs-s-finite-eq-equiv[symmetric]]* by(auto intro!: *qbs-s-finite-measure-eq simp: qbs-s-finite-eq-def*)

lemma(in *pair-qbs-s-finite*) *qbs-s-finite-measure-eq-inverse*:
assumes $\llbracket X, \alpha, \mu \rrbracket_{sfin} = \llbracket X, \beta, \nu \rrbracket_{sfin}$
shows *qbs-s-finite-eq* (X, α, μ) (X, β, ν) *qbs-s-finite-eq'* (X, α, μ) (X, β, ν)
using *Quotient3-rel[OF Quotient3-qbs-measure, of (X,α,μ) (X,β,ν),simplified]*
by(*simp-all add: assms qbs-s-finite-eq-equiv*)

lift-definition *qbs-space-of* :: 'a *qbs-measure* \Rightarrow 'a *quasi-borel*
is fst by(*auto simp: qbs-s-finite-eq-def*)

lemma(in *qbs-s-finite*) *qbs-space-of*[*simp*]:
qbs-space-of $\llbracket X, \alpha, \mu \rrbracket_{sfin} = X$ **by**(*simp add: qbs-space-of.abs-eq*)

lemma *rep-qbs-space-of*:
assumes *qbs-space-of* $s = X$
shows $\exists \alpha \mu. s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \wedge$ *qbs-s-finite* $X \alpha \mu$
proof –
obtain $X' \alpha \mu$ **where** *hs*:
 $s = \llbracket X', \alpha, \mu \rrbracket_{sfin}$ *qbs-s-finite* $X' \alpha \mu$
using *rep-qbs-s-finite-measure'[of s]* **by** *auto*
then interpret *qs:qbs-s-finite* $X' \alpha \mu$
by *simp*
show *?thesis*
using *assms hs(2)* **by**(*auto simp add: hs(1)*)
qed

corollary *qbs-s-space-of-not-empty*: *qbs-space* (*qbs-space-of* X) $\neq \{\}$
by *transfer (auto simp: qbs-s-finite-eq-def qbs-s-finite-def in-Mx-def qbs-empty-equiv)*

4.1.2 The S-Finite Measure Monad

definition *monadM-qbs* :: 'a *quasi-borel* \Rightarrow 'a *qbs-measure quasi-borel* **where**
monadM-qbs $X \equiv$ *Abs-quasi-borel* ($\{s. \text{qbs-space-of } s = X\}, \{\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$)

lemma
shows *monadM-qbs-space*: *qbs-space* (*monadM-qbs* X) = $\{s. \text{qbs-space-of } s = X\}$
and *monadM-qbs-Mx*: *qbs-Mx* (*monadM-qbs* X) = $\{\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$
proof –
have $\{\lambda r::\text{real}. \llbracket X, \alpha, k r \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\} \subseteq UNIV \rightarrow \{s. \text{qbs-space-of } s = X\}$
proof *safe*
fix $x \alpha$ **and** $k :: \text{real} \Rightarrow \text{real measure}$
assume $h:\alpha \in \text{qbs-Mx } X \text{ s-finite-kernel borel borel } k$
interpret $k:s\text{-finite-kernel borel borel } k$ **by** *fact*
interpret *qbs-s-finite* $X \alpha k x$
using *k.image-s-finite-measure h(1)* **by**(*auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def k.kernel-sets*)

show *qbs-space-of* $\llbracket X, \alpha, k \rrbracket_{sfin} = X$
by *simp*
qed
moreover have *qbs-closed1* $\{\lambda r. \llbracket X, \alpha, k \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$
proof(*safe intro!*: *qbs-closed1I*)
fix α **and** $f :: \text{real} \Rightarrow \text{real}$ **and** $k :: \text{real} \Rightarrow \text{real measure}$
assume $h: f \in \text{borel-measurable borel } \alpha \in \text{qbs-Mx } X \text{ s-finite-kernel borel borel } k$
then show $\exists \alpha' ka. (\lambda r. \llbracket X, \alpha, k \rrbracket_{sfin}) \circ f = (\lambda r. \llbracket X, \alpha', ka \rrbracket_{sfin}) \wedge \alpha' \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } ka$
by(*auto intro!*: $\text{exI}[\text{where } x=\alpha] \text{ exI}[\text{where } x=\lambda x. k (f x)] \text{ simp: s-finite-kernel.comp-measurable}[OF h(3,1)]$)
qed
moreover have *qbs-closed2* $\{s. \text{qbs-space-of } s = X\} \{\lambda r. \llbracket X, \alpha, k \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$
proof(*safe intro!*: *qbs-closed2I*)
fix s
assume $h: X = \text{qbs-space-of } s$
from *rep-qbs-space-of*[*OF this[symmetric]*] **obtain** $\alpha \mu$ **where** $s: s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu$
by *auto*
then interpret *qbs-s-finite* $X \alpha \mu$ **by** *simp*
show $\exists \alpha k. (\lambda r. s) = (\lambda r. \llbracket \text{qbs-space-of } s, \alpha, k \rrbracket_{sfin}) \wedge \alpha \in \text{qbs-Mx } (\text{qbs-space-of } s) \wedge \text{s-finite-kernel borel borel } k$
by(*auto intro!*: $\text{exI}[\text{where } x=\alpha] \text{ exI}[\text{where } x=\lambda r. \mu] \text{ s-finite-kernel-const simp: } s(1) \text{ s-finite-kernel-cong-sets}[OF - \text{mu-sets}[symmetric]] \text{ sets-eq-imp-space-eq}[OF \text{mu-sets}]$)
qed
moreover have *qbs-closed3* $\{\lambda r. \llbracket X, \alpha, k \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$
proof(*safe intro!*: *qbs-closed3I*)
fix $P :: \text{real} \Rightarrow \text{nat}$ **and** $Fi :: \text{nat} \Rightarrow -$
assume $P[\text{measurable}]: P \in \text{borel} \rightarrow_M \text{count-space UNIV}$
and $\forall i. Fi i \in \{\lambda r::\text{real}. \llbracket X, \alpha, k \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$
then obtain $\alpha i ki$ **where** $Fi: \bigwedge i. Fi i = (\lambda r. \llbracket X, \alpha i i, ki i \rrbracket_{sfin}) \bigwedge i. \alpha i i \in \text{qbs-Mx } X \bigwedge i. \text{s-finite-kernel borel borel } (ki i)$
by *auto metis*
interpret *nat-real: standard-borel-ne count-space* $(UNIV :: \text{nat set}) \otimes_M (\text{borel} :: \text{real measure})$
by(*auto intro!*: *pair-standard-borel-ne*)
note [*simp*] = *nat-real.from-real-to-real[simplified space-pair-measure, simplified]*
define α **where** $\alpha \equiv (\lambda r. \text{case-prod } \alpha i (\text{nat-real.from-real } r))$
define k **where** $k \equiv (\lambda r. \text{distr } (\text{distr } (ki (P r) r) (\text{count-space UNIV} \otimes_M \text{borel})) (\lambda r'. (P r, r')))$ *borel nat-real.to-real*
have $\alpha: \alpha \in \text{qbs-Mx } X$
unfolding α -*def* *qbs-Mx-is-morphisms*
proof(*rule qbs-morphism-compose*[**where** $g = \text{nat-real.from-real}$ **and** $Y = \text{qbs-count-space UNIV} \otimes_Q \text{qbs-borel}$])
show *nat-real.from-real* $\in \text{qbs-borel} \rightarrow_Q \text{qbs-count-space UNIV} \otimes_Q \text{qbs-borel}$

```

    by(simp add: r-preserves-product[symmetric] standard-borel.standard-borel-r-full-faithful[of
borel :: real measure,simplified,symmetric] standard-borel-ne.standard-borel)
  next
    show case-prod  $\alpha i \in \text{qbs-count-space UNIV} \otimes_Q \text{qbs-borel} \rightarrow_Q X$ 
    using Fi(2) by(auto intro!: qbs-morphism-pair-count-space1 simp: qbs-Mx-is-morphisms)
  qed
  have sets-ki[measurable-cong]: sets (ki i r) = sets borel sets (k r) = sets borel
for i r
  using Fi(3) by(auto simp: s-finite-kernel-def measure-kernel-def k-def)
  interpret k:s-finite-kernel borel borel k
  proof -
    have 1:k = ( $\lambda(i,r). \text{distr (ki i r) borel } (\lambda r'. \text{nat-real.to-real } (i, r'))$ )  $\circ (\lambda r.$ 
(P r, r))
    by standard (auto simp: k-def distr-distr comp-def)
    have s-finite-kernel borel borel ...
    unfolding comp-def
    by(rule s-finite-kernel.comp-measurable[where X=count-space UNIV  $\otimes_M$ 
borel],rule s-finite-kernel-pair-countble1, auto intro!: s-finite-kernel.distr-s-finite-kernel[OF
Fi(3)])
    thus s-finite-kernel borel borel k by(simp add: 1)
  qed
  have ( $\lambda r. \text{Fi (P r) r}$ ) = ( $\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin}$ )
  unfolding Fi(1)
  proof
    fix r
    interpret pq:pair-qbs-s-finite X  $\alpha i$  (P r) ki (P r) r  $\alpha k r$ 
    by(auto simp: pair-qbs-s-finite-def qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def
k.image-s-finite-measure s-finite-kernel.image-s-finite-measure[OF Fi(3)] sets-ki  $\alpha$ 
Fi(2))
    show  $\llbracket X, \alpha i$  (P r), ki (P r) r  $\rrbracket_{sfin} = \llbracket X, \alpha, k r \rrbracket_{sfin}$ 
    by(rule pq.qbs-s-finite-measure-eq, simp add: k-def distr-distr comp-def,simp
add:  $\alpha$ -def)
  qed
  thus  $\exists \alpha k. (\lambda r. \text{Fi (P r) r}) = (\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin}) \wedge \alpha \in \text{qbs-Mx } X \wedge$ 
s-finite-kernel borel borel k
  by(auto intro!: exI[where x= $\alpha$ ] exI[where x=k] simp:  $\alpha k$ .s-finite-kernel-axioms)
  qed
  ultimately have Rep-quasi-borel (monadM-qbs X) = ( $\{s. \text{qbs-space-of } s = X\}$ ,
 $\{\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$ )
  by(auto intro!: Abs-quasi-borel-inverse simp: monadM-qbs-def is-quasi-borel-def)
  thus qbs-space (monadM-qbs X) =  $\{s. \text{qbs-space-of } s = X\}$  qbs-Mx (monadM-qbs
X) =  $\{\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin} \mid \alpha k. \alpha \in \text{qbs-Mx } X \wedge \text{s-finite-kernel borel borel } k\}$ 
  by(simp-all add: qbs-space-def qbs-Mx-def)
  qed

lemma monadM-qbs-empty-iff: qbs-space X =  $\{\}$   $\longleftrightarrow$  qbs-space (monadM-qbs X)
=  $\{\}$ 
  by(auto simp: monadM-qbs-space qbs-s-space-of-not-empty) (meson in-Mx.intro
qbs-closed2-dest qbs-s-finite.qbs-space-of qbs-s-finite-def rep-qbs-s-finite-measure')

```

lemma(in *qbs-s-finite*) *in-space-monadM*[*qbs*]: $\llbracket X, \alpha, \mu \rrbracket_{sfin} \in \text{qbs-space } (\text{monadM-qbs } X)$

by(*simp add: monadM-qbs-space*)

lemma *rep-qbs-space-monadM*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$

obtains $\alpha \mu$ **where** $s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu$

using *rep-qbs-space-of-assms that by(auto simp: monadM-qbs-space)*

lemma *rep-qbs-space-monadM-sigma-finite*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$

obtains $\alpha \mu$ **where** $s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu \text{ sigma-finite-measure } \mu$

proof –

obtain $\alpha \mu$ **where** $s:s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu$

by(*metis rep-qbs-space-monadM-assms*)

hence *standard-borel-ne μ s-finite-measure μ*

by(*auto intro!: standard-borel-ne-sets[of borel μ] simp: qbs-s-finite-def qbs-s-finite-axioms-def*)

from *exists-push-forward[OF this] obtain $\mu' f$ where f :*

$f \in (\text{borel} :: \text{real measure}) \rightarrow_M \mu \text{ sets } \mu' = \text{sets borel sigma-finite-measure } \mu'$

distr $\mu' \mu f = \mu$

by *metis*

hence [*measurable*]: $f \in \text{borel-measurable borel}$

using $s(2)$ **by**(*auto simp: qbs-s-finite-def qbs-s-finite-axioms-def cong: measurable-cong-sets*)

interpret *pair-qbs-s-finite* $X \alpha \mu \alpha \circ f \mu'$

proof –

have *qbs-s-finite* $X (\alpha \circ f) \mu'$

using $s(2)$ **by**(*auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def[of μ'] f(2,3) sigma-finite-measure.s-finite-measure*)

thus *pair-qbs-s-finite* $X \alpha \mu (\alpha \circ f) \mu'$

by(*auto simp: pair-qbs-s-finite-def s(2)*)

qed

have $\llbracket X, \alpha, \mu \rrbracket_{sfin} = \llbracket X, \alpha \circ f, \mu' \rrbracket_{sfin}$

proof –

have [*simp*]: $\text{distr } \mu (\text{qbs-to-measure } X) \alpha = \text{distr } (\text{distr } \mu' \mu f) (\text{qbs-to-measure } X) \alpha$

by(*simp add: f(4)*)

show *?thesis*

by(*auto intro!: qbs-s-finite-measure-eq simp: distr-distr*)

qed

with $s(1)$ *pq2.qbs-s-finite-axioms f(3) that*

show *?thesis by metis*

qed

lemma *qbs-space-of-in*: $s \in \text{qbs-space } (\text{monadM-qbs } X) \implies \text{qbs-space-of } s = X$

by(*simp add: monadM-qbs-space*)

lemma *in-qbs-space-of*: $s \in \text{qbs-space } (\text{monadM-qbs } (\text{qbs-space-of } s))$

by(simp add: monadM-qbs-space)

4.1.3 l

lift-definition qbs-l :: 'a qbs-measure \Rightarrow 'a measure
is $\lambda p. \text{distr } (\text{snd } (\text{snd } p)) \text{ (qbs-to-measure (fst } p)) \text{ (fst } (\text{snd } p))$
by(auto simp: qbs-s-finite-eq-def)

lemma(in qbs-s-finite) qbs-l: qbs-l $\llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} = \text{distr } \mu \text{ (qbs-to-measure } X) \alpha$
by(simp add: qbs-l.abs-eq)

interpretation qbs-l-s-finite: s-finite-measure qbs-l (s :: 'a qbs-measure)

proof(transfer)

show $\bigwedge s:: 'a \text{ qbs-s-finite-t. qbs-s-finite-eq } s \ s \Longrightarrow \text{s-finite-measure } (\text{distr } (\text{snd } (\text{snd } s)) \text{ (qbs-to-measure (fst } s)) \text{ (fst } (\text{snd } s))))$

proof safe

fix X :: 'a quasi-borel

fix $\alpha \ \mu$

assume qbs-s-finite-eq (X, α, μ) (X, α, μ)

then interpret qbs-s-finite X $\alpha \ \mu$

by(simp add: qbs-s-finite-eq-def)

show s-finite-measure (distr (snd (snd (X, α, μ))) (qbs-to-measure (fst (X, α, μ)))) (fst (snd (X, α, μ))))

by(auto intro!: s-finite-measure.s-finite-measure-distr simp: s-finite-measure-axioms)

qed

qed

lemma space-qbs-l: qbs-space (qbs-space-of s) = space (qbs-l s)
by(transfer, auto simp: space-L)

lemma space-qbs-l-ne: space (qbs-l s) $\neq \{\}$
by transfer (auto simp: qbs-s-finite-eq-def qbs-s-finite-def in-Mx-def space-L qbs-empty-equiv)

lemma qbs-l-sets: sets (qbs-to-measure (qbs-space-of s)) = sets (qbs-l s)
by(transfer, simp)

lemma qbs-null-measure-in-Mx: qbs-space X $\neq \{\}$ \Longrightarrow qbs-null-measure X \in qbs-space (monadM-qbs X)

by(simp add: qbs-s-finite.in-space-monadM[OF qbs-null-measure-s-finite] qbs-null-measure-def)

lemma qbs-null-measure-null-measure: qbs-space X $\neq \{\}$ \Longrightarrow qbs-l (qbs-null-measure X) = null-measure (qbs-to-measure X)

by(auto simp: qbs-null-measure-def qbs-s-finite.qbs-l[OF qbs-null-measure-s-finite] null-measure-distr)

lemma space-qbs-l-in:

assumes s \in qbs-space (monadM-qbs X)

shows space (qbs-l s) = qbs-space X

by (metis assms qbs-s-finite.qbs-space-of rep-qbs-space-monadM space-qbs-l)

lemma *sets-qbs-l*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
shows $\text{sets } (\text{qbs-l } s) = \text{sets } (\text{qbs-to-measure } X)$
using *assms qbs-l-sets qbs-space-of-in* **by** *blast*

lemma *measurable-qbs-l*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
shows $\text{qbs-l } s \rightarrow_M M = X \rightarrow_Q \text{measure-to-qbs } M$
by (*auto simp: measurable-cong-sets[OF qbs-l-sets[of s,simplified qbs-space-of-in[OF assms(1)],symmetric] refl] lr-adjunction-correspondence*)

lemma *measurable-qbs-l'*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
shows $\text{qbs-l } s \rightarrow_M M = \text{qbs-to-measure } X \rightarrow_M M$
by (*simp add: measurable-qbs-l[OF assms] lr-adjunction-correspondence*)

lemma *rep-qbs-Mx-monadM*:

assumes $\gamma \in \text{qbs-Mx } (\text{monadM-qbs } X)$
obtains α **where** $\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{s\text{fin}}) \alpha \in \text{qbs-Mx } X$ *s-finite-kernel borel borel* $k \wedge r. \text{qbs-s-finite } X \alpha (k \ r)$

proof –

have $\bigwedge \alpha \ r \ k. \alpha \in \text{qbs-Mx } X \implies \text{s-finite-kernel borel borel } k \implies \text{qbs-s-finite } X \alpha (k \ r)$

by (*auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def s-finite-kernel.image-s-finite-measure*)
(*auto simp: s-finite-kernel-def measure-kernel-def*)

thus *?thesis*

using *that assms* **by** (*fastforce simp: monadM-qbs-Mx*)

qed

lemma *qbs-l-measurable[measurable]:qbs-l* $\in \text{qbs-to-measure } (\text{monadM-qbs } X) \rightarrow_M$ *s-finite-measure-algebra* $(\text{qbs-to-measure } X)$

proof (*rule qbs-morphism-dest[OF qbs-morphismI]*)

fix γ

assume $\gamma \in \text{qbs-Mx } (\text{monadM-qbs } X)$

from *rep-qbs-Mx-monadM* [*OF this*] **obtain** α **where** h :

$\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{s\text{fin}}) \alpha \in \text{qbs-Mx } X$ *s-finite-kernel borel borel* $k \wedge r. \text{qbs-s-finite } X \alpha (k \ r)$

by *metis*

show $\text{qbs-l} \circ \gamma \in \text{qbs-Mx } (\text{measure-to-qbs } (\text{s-finite-measure-algebra } (\text{qbs-to-measure } X)))$

by (*auto simp add: qbs-Mx-R comp-def h(1) qbs-s-finite.qbs-l[OF h(4)] h(2,3) intro!: s-finite-kernel.kernel-measurable-s-finite s-finite-kernel.distr-s-finite-kernel[where Y=borel]*)

qed

lemma *qbs-l-measure-kernel: measure-kernel* $(\text{qbs-to-measure } (\text{monadM-qbs } X))$ $(\text{qbs-to-measure } X)$ *qbs-l*

proof (*cases qbs-space X = {}*)


```

case True
with monadM-qbs-empty-iff[of X,simplified this] show ?thesis
  by(auto intro!: measure-kernel-empty-trivial simp: space-L)
next
case 1:False
show ?thesis
proof
  show  $\bigwedge x. x \in \text{space } (\text{qbs-to-measure } (\text{monadM-qbs } X)) \implies \text{sets } (\text{qbs-l } x) =$ 
  sets (qbs-to-measure X)
  using qbs-l-sets by(auto simp: space-L monadM-qbs-space)
next
  show space (qbs-to-measure X)  $\neq$  {}
  by(simp add: space-L 1)
qed (rule measurable-emeasure-kernel-s-finite-measure-algebra[OF qbs-l-measurable])
qed

```

```

lemma qbs-l-inj: inj-on qbs-l (qbs-space (monadM-qbs X))
by standard (auto simp: monadM-qbs-space, transfer, auto simp: qbs-s-finite-eq-def)

```

```

lemma qbs-l-morphism:

```

```

  assumes [measurable]:A  $\in$  sets (qbs-to-measure X)
  shows ( $\lambda s. \text{qbs-l } s \ A$ )  $\in$  monadM-qbs X  $\rightarrow_Q$  qbs-borel
proof(rule qbs-morphismI)
  fix  $\gamma$ 
  assume h: $\gamma \in$  qbs-Mx (monadM-qbs X)
  hence [qbs]:  $\gamma \in$  qbs-borel  $\rightarrow_Q$  monadM-qbs X
  by(simp-all add: qbs-Mx-is-morphisms)
  from rep-qbs-Mx-monadM[OF h(1)] obtain  $\alpha$   $k$  where hk:
   $\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{sfin}) \ \alpha \in$  qbs-Mx X s-finite-kernel borel borel k  $\bigwedge$  r. qbs-s-finite
  X  $\alpha$  (k r)
  by metis
  then interpret a: in-Mx X  $\alpha$  by(simp add: in-Mx-def)
  have k[measurable-cong]:sets (k r) = sets borel for  $r$ 
  using hk(3) by(auto simp: s-finite-kernel-def measure-kernel-def)
  show ( $\lambda s. \text{emeasure } (\text{qbs-l } s) \ A$ )  $\circ \gamma \in$  qbs-Mx qbs-borel
  by(auto simp: hk(1) qbs-s-finite.qbs-l[OF hk(4)] comp-def qbs-Mx-qbs-borel
  emeasure-distr sets-eq-imp-space-eq[OF k] intro!: s-finite-kernel.emeasure-measurable'[OF
  hk(3)] measurable-sets-borel[OF - assms])
qed

```

```

lemma qbs-l-finite-pred: qbs-pred (monadM-qbs X) ( $\lambda s. \text{finite-measure } (\text{qbs-l } s)$ )

```

```

proof –
  have qbs-space X  $\in$  sets (qbs-to-measure X)
  by (metis sets.top space-L)
  note qbs-l-morphism[OF this,qbs]
  have [simp]:finite-measure (qbs-l s)  $\iff$  qbs-l s X  $\neq$   $\infty$  if  $s \in$  monadM-qbs X
for  $s$ 
  by(auto intro!: finite-measureI dest: finite-measure.emeasure-finite simp: space-qbs-l-in[OF
  that])

```

show *?thesis*
by(*simp cong: qbs-morphism-cong*)
qed

lemma *qbs-l-subprob-pred: qbs-pred (monadM-qbs X) (λs. subprob-space (qbs-l s))*

proof –

have *qbs-space X ∈ sets (qbs-to-measure X)*

by (*metis sets.top space-L*)

note *qbs-l-morphism[OF this,qbs]*

have [*simp*]:*subprob-space (qbs-l s) ↔ qbs-l s X ≤ 1 if s ∈ monadM-qbs X for*
s

by(*auto intro!: subprob-spaceI dest: subprob-space.subprob-emeasure-le-1 simp:*
space-qbs-l-ne) (*simp add: space-qbs-l-in[OF that]*)

show *?thesis*

by(*simp cong: qbs-morphism-cong*)

qed

lemma *qbs-l-prob-pred: qbs-pred (monadM-qbs X) (λs. prob-space (qbs-l s))*

proof –

have *qbs-space X ∈ sets (qbs-to-measure X)*

by (*metis sets.top space-L*)

note *qbs-l-morphism[OF this,qbs]*

have [*simp*]:*prob-space (qbs-l s) ↔ qbs-l s X = 1 if s ∈ monadM-qbs X for s*

by(*auto intro!: prob-spaceI simp: space-qbs-l-ne*) (*auto simp add: space-qbs-l-in[OF*
that] dest: prob-space.emeasure-space-1)

show *?thesis*

by(*simp cong: qbs-morphism-cong*)

qed

4.1.4 Return

definition *return-qbs :: 'a quasi-borel ⇒ 'a ⇒ 'a qbs-measure where*

return-qbs X x ≡ [[X, λr. x, SOME μ. real-distribution μ]]_{sfin}

lemma(*in real-distribution*)

assumes *x ∈ qbs-space X*

shows *return-qbs: return-qbs X x = [[X, λr. x, M]]_{sfin}*

and *return-qbs-prob: qbs-prob X (λr. x) M*

and *return-qbs-s-finite: qbs-s-finite X (λr. x) M*

proof –

interpret *qs1: qbs-prob X λr. x M*

by(*auto simp: qbs-prob-def in-Mx-def real-distribution-axioms intro!: qbs-closed2-dest*
assms)

show *return-qbs X x = [[X, λr. x, M]]_{sfin}*

unfolding *return-qbs-def*

proof(*rule someI2*)

show *real-distribution (return borel 0) by (auto simp: real-distribution-def*
real-distribution-axioms-def,rule prob-space-return) simp

next

```

fix N
assume real-distribution N
then interpret qs2: qbs-s-finite X λr. x N
  by(auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def real-distribution-def
real-distribution-axioms-def intro!: qbs-closed2-dest assms sigma-finite-measure.s-finite-measure
prob-space-imp-sigma-finite)
  interpret pair-qbs-s-finite X λr. x N λr. x M
    by standard
  show  $\llbracket X, \lambda r. x, N \rrbracket_{sfin} = \llbracket X, \lambda r. x, M \rrbracket_{sfin}$ 
    by(auto intro!: qbs-s-finite-measure-eq measure-eqI simp: emeasure-distr) (metis
⟨real-distribution N⟩ emeasure-space-1 prob-space.emeasure-space-1 qs2.mu-sets real-distribution.axioms(1)
sets-eq-imp-space-eq space-borel space-eq-univ)
  qed
  show qbs-prob X (λr. x) M qbs-s-finite X (λr. x) M
    by(simp-all add: qs1.qbs-prob-axioms qs1.qbs-s-finite-axioms)
qed

```

```

lemma return-qbs-comp:
  assumes α ∈ qbs-Mx X
  shows (return-qbs X ∘ α) = (λr.  $\llbracket X, \alpha, \text{return borel } r \rrbracket_{sfin}$ )
proof
  fix r
  interpret pqp: pair-qbs-prob X λk. α r return borel 0 α return borel r
    by(simp add: assms qbs-Mx-to-X[OF assms] pair-qbs-prob-def qbs-prob-def
in-Mx-def real-distribution-def real-distribution-axioms-def prob-space-return)
  show (return-qbs X ∘ α) r =  $\llbracket X, \alpha, \text{return borel } r \rrbracket_{sfin}$ 
    by(auto simp: pqp.pq1.return-qbs[OF qbs-Mx-to-X[OF assms]] distr-return in-
tro!: pqp.qbs-s-finite-measure-eq)
qed

```

```

corollary return-qbs-morphism[qbs]: return-qbs X ∈ X →Q monadM-qbs X
proof(rule qbs-morphismI)
  interpret rr : real-distribution return borel 0
    by(simp add: real-distribution-def real-distribution-axioms-def prob-space-return)
  fix α
  assume h:α ∈ qbs-Mx X
  then have 1:return-qbs X ∘ α = (λr.  $\llbracket X, \alpha, \text{return borel } r \rrbracket_{sfin}$ )
    by(rule return-qbs-comp)
  show return-qbs X ∘ α ∈ qbs-Mx (monadM-qbs X)
    by(auto simp: 1 monadM-qbs-Mx h prob-kernel-def' intro!: exI[where x=α]
exI[where x=return borel] prob-kernel.s-finite-kernel-prob-kernel)
qed

```

4.1.5 Bind

```

definition bind-qbs :: ['a qbs-measure, 'a ⇒ 'b qbs-measure] ⇒ 'b qbs-measure
where
  bind-qbs s f ≡ (let (X, α, μ) = rep-qbs-measure s;
                    Y = qbs-space-of (f (α undefined)));

```

$$(\beta, k) = (\text{SOME } (\beta, k). f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \wedge \beta \in \text{qbs-Mx } Y \wedge \text{s-finite-kernel borel borel } k) \text{ in } \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin}$$

adhoc-overloading *Monad-Syntax.bind bind-qbs*

lemma(in *qbs-s-finite*)

assumes $s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$

$f \in X \rightarrow_Q \text{monadM-qbs } Y$

$\beta \in \text{qbs-Mx } Y$

s-finite-kernel borel borel k

and $(f \circ \alpha) = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin})$

shows *bind-qbs-s-finite:qbs-s-finite* $Y \beta (\mu \ggg_k k)$

and *bind-qbs*: $s \ggg f = \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin}$

proof –

interpret k : *s-finite-kernel borel borel* k **by** *fact*

interpret *s-fin*: *qbs-s-finite* $Y \beta \mu \ggg_k k$

by(*auto simp: qbs-s-finite-def in-Mx-def assms(3) mu-sets qbs-s-finite-axioms-def k.sets-bind-kernel[OF - mu-sets] intro!:k.comp-s-finite-measure s-finite-measure-axioms*)

show $s \ggg f = \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin}$

proof –

{

fix $X' \alpha' \mu'$

assume $(X', \alpha', \mu') \in \text{Rep-qbs-measure } \llbracket X, \alpha, \mu \rrbracket_{sfin}$

then have h : $X' = X$ *qbs-s-finite* $X' \alpha' \mu'$ *qbs-s-finite-eq* $(X, \alpha, \mu) (X', \alpha', \mu')$

by(*simp-all add: if-in-Rep-qbs-measure*)

then interpret *s-fin-pq1*: *pair-qbs-s-finite* $X \alpha \mu \alpha' \mu'$

by(*auto simp: pair-qbs-s-finite-def qbs-s-finite-axioms*)

have [*simp*]: *qbs-space-of* $(f (\alpha' r)) = Y$ **for** r

using *qbs-Mx-to-X[OF qbs-morphism-Mx[OF assms(2) s-fin-pq1.pq2.in-Mx]*, of

$r]$

by(*auto simp: monadM-qbs-space*)

have (let $Y = \text{qbs-space-of } (f (\alpha' \text{undefined}))$ in case *SOME* (β, k) . $(\lambda r. f (\alpha' r)) = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \wedge \beta \in \text{qbs-Mx } Y \wedge \text{s-finite-kernel borel borel } k$ of

$(\beta, k) \Rightarrow \llbracket Y, \beta, \mu' \ggg_k k \rrbracket_{sfin} = \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin}$

proof –

have (case *SOME* (β, k) . $(\lambda r. f (\alpha' r)) = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \wedge \beta \in \text{qbs-Mx } Y \wedge \text{s-finite-kernel borel borel } k$ of $(\beta, k) \Rightarrow \llbracket Y, \beta, \mu' \ggg_k k \rrbracket_{sfin} = \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin}$

proof(*rule someI2-ex*)

show $\exists a$. case a of $(\beta, k) \Rightarrow (\lambda r. f (\alpha' r)) = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \wedge \beta \in \text{qbs-Mx } Y \wedge \text{s-finite-kernel borel borel } k$

using *qbs-morphism-Mx[OF assms(2) s-fin-pq1.pq2.in-Mx]*

by(*auto simp: comp-def monadM-qbs-Mx*)

next

show $\bigwedge x$. case x of $(\beta, k) \Rightarrow (\lambda r. f (\alpha' r)) = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \wedge \beta \in \text{qbs-Mx } Y \wedge \text{s-finite-kernel borel borel } k \Longrightarrow (\text{case } x \text{ of } (\beta, k) \Rightarrow \llbracket Y, \beta, \mu' \ggg_k k \rrbracket_{sfin} = \llbracket Y, \beta, \mu \ggg_k k \rrbracket_{sfin})$

proof *safe*

```

    fix  $\beta'$   $k'$ 
      assume  $h':(\lambda r. f (\alpha' r)) = (\lambda r. \llbracket Y, \beta', k' r \rrbracket_{sfin})$   $\beta' \in qbs-Mx Y$ 
      s-finite-kernel borel borel  $k'$ 
      interpret  $k'$ : s-finite-kernel borel borel  $k'$  by fact
      have qbs-s-finite  $Y \beta' (\mu' \gg_k k')$ 
      by(auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def  $h'(2)$ )
       $k'.sets-bind-kernel[OF - s-fin-pq1.pq2.mu-sets]$  s-fin-pq1.pq2.mu-sets intro! $k'.comp-s-finite-measure$ 
      s-fin-pq1.pq2.s-finite-measure-axioms)
      then interpret s-fin-pq2: pair-qbs-s-finite  $Y \beta' \mu' \gg_k k' \beta \mu \gg_k k$ 
      by(auto simp: pair-qbs-s-finite-def s-fin.qbs-s-finite-axioms)
      show  $\llbracket Y, \beta', \mu' \gg_k k \rrbracket_{sfin} = \llbracket Y, \beta, \mu \gg_k k \rrbracket_{sfin}$ 
      proof(rule s-fin-pq2.qbs-s-finite-measure-eq)
        show  $distr (\mu' \gg_k k') (qbs-to-measure Y) \beta' = distr (\mu \gg_k k)$ 
        ( $qbs-to-measure Y) \beta$  (is ?lhs = ?rhs)
      proof -
        have ?lhs =  $\mu' \gg_k (\lambda r. distr (k' r) (qbs-to-measure Y) \beta')$ 
        by(simp add:  $k'.distr-bind-kernel[OF - s-fin-pq1.pq2.mu-sets]$ )
        also have ... =  $\mu' \gg_k (\lambda r. qbs-l \llbracket Y, \beta', k' r \rrbracket_{sfin})$ 
        by(rule bind-kernel-cong-All, rule qbs-s-finite.qbs-l[symmetric, OF
        qbs-s-finite-All[where  $k=k'$  and  $M=borel$ ]]) (auto simp:  $k'.s-finite-kernel-axioms$ )
        also have ... =  $\mu' \gg_k (\lambda r. qbs-l (f (\alpha' r)))$ 
        by(auto simp: fun-cong[OF  $h'(1)$ ])
        also have ... =  $distr \mu' (qbs-to-measure X) \alpha' \gg_k (\lambda x. qbs-l (f x))$ 
        by(simp add: measure-kernel.bind-kernel-distr[OF measure-kernel.measure-kernel-comp[OF
        qbs-l-measure-kernel set-mp[OF l-preserves-morphisms assms(2)]]] sets-eq-imp-space-eq[OF
        s-fin-pq1.pq2.mu-sets])
        also have ... =  $distr \mu (qbs-to-measure X) \alpha \gg_k (\lambda x. qbs-l (f x))$ 
        by(simp add: qbs-s-finite-eq-dest(4)[OF  $h(3)$ ])
        also have ... =  $\mu \gg_k (\lambda r. qbs-l (f (\alpha r)))$ 
        by(simp add: measure-kernel.bind-kernel-distr[OF measure-kernel.measure-kernel-comp[OF
        qbs-l-measure-kernel set-mp[OF l-preserves-morphisms assms(2)]]] sets-eq-imp-space-eq[OF
        mu-sets])
        also have ... =  $\mu \gg_k (\lambda r. qbs-l \llbracket Y, \beta, k r \rrbracket_{sfin})$ 
        by(simp add: fun-cong[OF assms(5), simplified comp-def])
        also have ... =  $\mu \gg_k (\lambda r. distr (k r) (qbs-to-measure Y) \beta)$ 
        by(rule bind-kernel-cong-All, rule qbs-s-finite.qbs-l[OF qbs-s-finite-All[where
         $k=k$  and  $M=borel$ ]]) (auto simp:  $k.s-finite-kernel-axioms$ )
        also have ... = ?rhs
        by(simp add:  $k.distr-bind-kernel[OF - mu-sets]$ )
      finally show ?thesis .
    qed
  qed
  qed
  qed
  thus ?thesis by simp
  qed
}
show ?thesis
unfolding bind-qbs-def rep-qbs-measure-def qbs-measure.rep-def assms(1)

```

by(rule someI2, rule in-Rep-qbs-measure, auto) fact
 qed
 show qbs-s-finite Y β ($\mu \gg_k k$)
 by(rule s-fin.qbs-s-finite-axioms)
 qed

lemma bind-qbs-morphism':

assumes $f \in X \rightarrow_Q \text{monadM-qbs } Y$
 shows $(\lambda x. x \gg f) \in \text{monadM-qbs } X \rightarrow_Q \text{monadM-qbs } Y$
proof(rule qbs-morphismI)
 fix γ
 assume $\gamma \in \text{qbs-Mx } (\text{monadM-qbs } X)$
 from rep-qbs-Mx-monadM[OF this] **obtain** α k **where** h :
 $\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{sfin}) \alpha \in \text{qbs-Mx } X \text{ s-finite-kernel borel borel } k \wedge r. \text{qbs-s-finite}$
 $X \ \alpha \ (k \ r)$
 by metis
 from rep-qbs-Mx-monadM[OF qbs-morphism-Mx[OF assms this(2)]] **obtain** α'
 k' **where** h' :
 $f \circ \alpha = (\lambda r. \llbracket Y, \alpha', k' \ r \rrbracket_{sfin}) \alpha' \in \text{qbs-Mx } Y \text{ s-finite-kernel borel borel } k' \wedge r.$
 $\text{qbs-s-finite } Y \ \alpha' \ (k' \ r)$
 by metis
 have [simp]: $(\lambda x. x \gg f) \circ \gamma = (\lambda r. \llbracket Y, \alpha', k \ r \gg_k k \rrbracket_{sfin})$
 by standard (simp add: h(1) qbs-s-finite.bind-qbs[OF h(4) - assms h'(2,3,1)])
 show $(\lambda x. x \gg f) \circ \gamma \in \text{qbs-Mx } (\text{monadM-qbs } Y)$
 using h'(2) **by**(auto simp: s-finite-kernel.bind-kernel-s-finite-kernel[OF h(3)
 $h'(3)] \text{monadM-qbs-Mx intro!} \text{: exI[where } x=\alpha \text{']})$
 qed

lemma bind-qbs-return':

assumes $x \in \text{qbs-space } (\text{monadM-qbs } X)$
 shows $x \gg \text{return-qbs } X = x$
proof –
 obtain α μ **where** h : $\text{qbs-s-finite } X \ \alpha \ \mu \ x = \llbracket X, \alpha, \mu \rrbracket_{sfin}$
 using rep-qbs-space-monadM[OF assms] **by** blast
then interpret qs : $\text{qbs-s-finite } X \ \alpha \ \mu$ **by** simp
interpret prob-kernel borel borel return borel
 by(simp add: prob-kernel-def')
 show ?thesis
 by(simp add: qs.bind-qbs[OF h(2) return-qbs-morphism - - return-qbs-comp]
 $s\text{-finite-kernel-axioms bind-kernel-return''[OF qs.mu-sets] h(2)[symmetric]})$
 qed

lemma bind-qbs-return:

assumes $f \in X \rightarrow_Q \text{monadM-qbs } Y$
 and $x \in \text{qbs-space } X$
 shows $\text{return-qbs } X \ x \gg f = f \ x$
proof –
 from rep-qbs-space-monadM[OF qbs-morphism-space[OF assms]] **obtain** α μ

where h :

```

  f x =  $\llbracket Y, \alpha, \mu \rrbracket_{sfin}$  qbs-s-finite  $Y \alpha \mu$  by auto
  then interpret  $qs:qbs-s-finite Y \alpha \mu$  by simp
  interpret  $sk: s-finite-kernel$  borel borel  $\lambda r. \mu$ 
  by(auto intro!: s-finite-measure.s-finite-kernel-const simp: s-finite-kernel-cong-sets[OF
  refl qs.mu-sets[symmetric]] qs.s-finite-measure-axioms qs.mu-not-empty)
  interpret  $rd: real-distribution$  return borel 0
  by(simp add: real-distribution-def prob-space-return real-distribution-axioms-def)
  interpret  $qbs-prob X \lambda r. x$  return borel 0
  by(rule rd.return-qbs-prob[OF assms(2)])
  show ?thesis
  using bind-qbs[OF rd.return-qbs[OF assms(2)]] assms(1) qs.in-Mx sk.s-finite-kernel-axioms]
  by(simp add: h(1) sk.bind-kernel-return)
qed

```

lemma *bind-qbs-assoc*:

```

assumes  $s \in qbs-space$  (monadM-qbs  $X$ )
   $f \in X \rightarrow_Q monadM-qbs Y$ 
  and  $g \in Y \rightarrow_Q monadM-qbs Z$ 
  shows  $s \ggg (\lambda x. f x \ggg g) = (s \ggg f) \ggg g$  (is ?lhs = ?rhs)
proof -
  obtain  $\alpha \mu$  where  $h:s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$  qbs-s-finite  $X \alpha \mu$ 
  using rep-qbs-space-monadM[OF assms(1)] by blast
  then interpret  $qs: qbs-s-finite X \alpha \mu$  by simp
  from rep-qbs-Mx-monadM[OF qbs-morphism-Mx[OF assms(2) qs.in-Mx]] obtain
   $\beta k$  where  $h'$ :
     $f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \beta \in qbs-Mx Y$  s-finite-kernel borel borel  $k \wedge r.$ 
    qbs-s-finite  $Y \beta (k r)$ 
  by metis
  from rep-qbs-Mx-monadM[OF qbs-morphism-Mx[OF assms(3) h'(2)]] obtain  $\gamma$ 
   $k'$  where  $h''$ :
     $g \circ \beta = (\lambda r. \llbracket Z, \gamma, k' r \rrbracket_{sfin}) \gamma \in qbs-Mx Z$  s-finite-kernel borel borel  $k' \wedge r.$ 
    qbs-s-finite  $Z \gamma (k' r)$ 
  by metis
  have  $1:(\lambda x. f x \ggg g) \circ \alpha = (\lambda r. \llbracket Z, \gamma, k r \ggg_k k' \rrbracket_{sfin})$ 
  by standard (simp add: qbs-s-finite.bind-qbs[OF h'(4) fun-cong[OF h'(1),simplified]
  assms(3) h''(2,3,1)])

```

```

  have ?lhs =  $\llbracket Z, \gamma, \mu \ggg_k (\lambda r. k r \ggg_k k') \rrbracket_{sfin}$ 
  by(rule qs.bind-qbs[OF h(1) qbs-morphism-compose[OF assms(2) bind-qbs-morphism'[OF
  assms(3)]] h''(2) s-finite-kernel.bind-kernel-s-finite-kernel[OF h'(3) h''(3)] 1)
  also have ... =  $\llbracket Z, \gamma, \mu \ggg_k k \ggg_k k' \rrbracket_{sfin}$ 
  by(simp add: s-finite-kernel.bind-kernel-assoc[OF h'(3) h''(3) qs.mu-sets])
  also have ... = ?rhs
  by(simp add: qbs-s-finite.bind-qbs[OF qs.bind-qbs-s-finite[OF h(1) assms(2)
  h'(2,3,1)]] qs.bind-qbs[OF h(1) assms(2) h'(2,3,1)] assms(3) h''(2,3,1))
  finally show ?thesis .
qed

```

lemma *bind-qbs-cong*:

assumes $[qbs]:s \in \text{qbs-space } (\text{monadM-qbs } X)$

$\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$

and $[qbs]:f \in X \rightarrow_Q \text{monadM-qbs } Y$

shows $s \ggg f = s \ggg g$

proof –

from *rep-qbs-space-monadM*[*OF assms*(1)] **obtain** $\alpha \mu$ **where** *h*:

$s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \alpha \mu$ **by** *auto*

interpret *qbs-s-finite* $X \alpha \mu$ **by** *fact*

from *rep-qbs-Mx-monadM*[*OF qbs-morphism-Mx*[*OF assms*(3) *in-Mx*]] **obtain** βk **where** *h'*:

$f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \beta \in \text{qbs-Mx } Y \text{ s-finite-kernel borel borel } k$ **by** *metis*

have $g: g \in X \rightarrow_Q \text{monadM-qbs } Y$ $g \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin})$

using *qbs-Mx-to-X*[*OF in-Mx*] *assms*(2) *fun-cong*[*OF h'*(1)]

by(*auto simp: assms*(2)[*symmetric*] *cong: qbs-morphism-cong*)

show *?thesis*

by(*simp add: bind-qbs*[*OF h*(1) *assms*(3) *h'*(2,3,1)] *bind-qbs*[*OF h*(1) *g*(1) *h'*(2,3) *g*(2)])

qed

4.1.6 The Functorial Action

definition *distr-qbs* :: $['a \text{ quasi-borel}, 'b \text{ quasi-borel}, 'a \Rightarrow 'b, 'a \text{ qbs-measure}] \Rightarrow 'b$
qbs-measure **where**

$\text{distr-qbs} - Y f s x \equiv s x \ggg \text{return-qbs } Y \circ f$

lemma *distr-qbs-morphism'*:

assumes $f \in X \rightarrow_Q Y$

shows *distr-qbs* $X Y f \in \text{monadM-qbs } X \rightarrow_Q \text{monadM-qbs } Y$

unfolding *distr-qbs-def*

by(*rule bind-qbs-morphism'*[*OF qbs-morphism-comp*[*OF assms return-qbs-morphism*]])

lemma(*in qbs-s-finite*)

assumes $s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$

and $f \in X \rightarrow_Q Y$

shows *distr-qbs-s-finite:qbs-s-finite* $Y (f \circ \alpha) \mu$

and *distr-qbs: distr-qbs* $X Y f s = \llbracket Y, f \circ \alpha, \mu \rrbracket_{sfin}$

by(*auto intro!*: *bind-qbs*[*OF assms*(1) *qbs-morphism-comp*[*OF assms*(2) *return-qbs-morphism*], *of f \circ \alpha return borel ,simplified bind-kernel-return''*[*OF mu-sets*]] *bind-qbs-s-finite*[*OF assms*(1) *qbs-morphism-comp*[*OF assms*(2) *return-qbs-morphism*], *of f \circ \alpha return borel ,simplified bind-kernel-return''*[*OF mu-sets*]])

simp: distr-qbs-def return-qbs-comp[*OF qbs-morphism-Mx*[*OF assms*(2) *in-Mx*], *simplified comp-assoc*[*symmetric*]] *qbs-morphism-Mx*[*OF assms*(2) *in-Mx*] *prob-kernel.s-finite-kernel-prob-kernel*[*of borel borel return borel,simplified prob-kernel-def'*])

lemma(*in qbs-prob*)

assumes $s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$

and $f \in X \rightarrow_Q Y$

shows *distr-qbs-prob:qbs-prob* $Y (f \circ \alpha) \mu$

by(*auto simp: distr-qbs-def prob-space-axioms intro!: qbs-s-finite.qbs-probI*[*OF distr-qbs-s-finite*[*OF assms*]])

We show that M is a functor i.e. M preserve identity and composition.

lemma *distr-qbs-id*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
shows $\text{distr-qbs } X X \text{ id } s = s$
using *bind-qbs-return*'[*OF assms*] **by**(*simp add: distr-qbs-def*)

lemma *distr-qbs-comp*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
 $f \in X \rightarrow_Q Y$
and $g \in Y \rightarrow_Q Z$
shows $((\text{distr-qbs } Y Z g) \circ (\text{distr-qbs } X Y f)) s = \text{distr-qbs } X Z (g \circ f) s$

proof –

from *rep-qbs-space-monadM*[*OF assms(1)*] **obtain** $\alpha \mu$ **where** h :
 $\text{qbs-s-finite } X \alpha \mu s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$ **by** *metis*
have $\text{qbs-s-finite } Y (f \circ \alpha) \mu \text{ distr-qbs } X Y f s = \llbracket Y, f \circ \alpha, \mu \rrbracket_{sfin}$
by(*simp-all add: qbs-s-finite.distr-qbs-s-finite*[*OF h assms(2)*] *qbs-s-finite.distr-qbs*[*OF h assms(2)*])
from $\text{qbs-s-finite.distr-qbs}$ [*OF this assms(3)*] $\text{qbs-s-finite.distr-qbs}$ [*OF h qbs-morphism-comp*[*OF assms(2,3)*]]
show *?thesis*
by(*simp add: comp-assoc*)
qed

4.1.7 Join

definition *join-qbs* :: ' a *qbs-measure* *qbs-measure* \Rightarrow ' a *qbs-measure* **where**
 $\text{join-qbs} \equiv (\lambda \text{sst. } \text{sst} \ggg \text{id})$

lemma *join-qbs-morphism*[*qbs*]: $\text{join-qbs} \in \text{monadM-qbs } (\text{monadM-qbs } X) \rightarrow_Q \text{monadM-qbs } X$

by(*simp add: join-qbs-def bind-qbs-morphism*'[*OF qbs-morphism-ident*])

lemma

assumes $\text{qbs-s-finite } (\text{monadM-qbs } X) \beta \mu$
 $\text{ssx} = \llbracket \text{monadM-qbs } X, \beta, \mu \rrbracket_{sfin}$
 $\alpha \in \text{qbs-Mx } X$
 $\text{s-finite-kernel borel borel } k$
and $\beta = (\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin})$
shows $\text{qbs-s-finite-join-qbs-s-finite: } \text{qbs-s-finite } X \alpha (\mu \ggg_k k)$
and $\text{qbs-s-finite-join-qbs: } \text{join-qbs } \text{ssx} = \llbracket X, \alpha, \mu \ggg_k k \rrbracket_{sfin}$
using $\text{qbs-s-finite.bind-qbs}$ [*OF assms(1,2)*] $\text{qbs-morphism-ident assms(3,4)}$] $\text{qbs-s-finite.bind-qbs-s-finite}$ [*OF assms(1,2)*] $\text{qbs-morphism-ident assms(3,4)}$]
by(*auto simp: assms(5) join-qbs-def*)

4.1.8 Strength

definition *strength-qbs* :: [*a quasi-borel, 'b quasi-borel, 'a × 'b qbs-measure*] ⇒ (*a × 'b*) *qbs-measure* **where**
strength-qbs $W X = (\lambda(w, sx). \text{let } (-, \alpha, \mu) = \text{rep-qbs-measure } sx$
in $\llbracket W \otimes_Q X, \lambda r. (w, \alpha r), \mu \rrbracket_{sfin}$)

lemma(*in qbs-s-finite*)

assumes $w \in \text{qbs-space } W$

and $sx = \llbracket X, \alpha, \mu \rrbracket_{sfin}$

shows *strength-qbs-s-finite*: $\text{qbs-s-finite } (W \otimes_Q X) (\lambda r. (w, \alpha r)) \mu$

and *strength-qbs*: $\text{strength-qbs } W X (w, sx) = \llbracket W \otimes_Q X, \lambda r. (w, \alpha r), \mu \rrbracket_{sfin}$

proof –

interpret *qs*: $\text{qbs-s-finite } W \otimes_Q X \lambda r. (w, \alpha r) \mu$

by(*auto simp: qbs-s-finite-def s-finite-measure-axioms qbs-s-finite-axioms-def mu-sets in-Mx-def assms(1) intro!: pair-qbs-MxI*)

show *qbs-s-finite* $(W \otimes_Q X) (\lambda r. (w, \alpha r)) \mu$ **by** (*rule qs.qbs-s-finite-axioms*)

show *strength-qbs* $W X (w, sx) = \llbracket W \otimes_Q X, \lambda r. (w, \alpha r), \mu \rrbracket_{sfin}$

proof –

{

fix $X' \alpha' \mu'$

assume $(X', \alpha', \mu') \in \text{Rep-qbs-measure } \llbracket X, \alpha, \mu \rrbracket_{sfin}$

then have h : $X' = X \text{ qbs-s-finite } X' \alpha' \mu' \text{ qbs-s-finite-eq } (X, \alpha, \mu) (X', \alpha', \mu')$

by(*simp-all add: if-in-Rep-qbs-measure*)

then interpret *qs'*: $\text{qbs-s-finite } W \otimes_Q X \lambda r. (w, \alpha' r) \mu'$

by(*auto simp: qbs-s-finite-def in-Mx-def assms(1) intro!: pair-qbs-MxI*)

interpret *pq*: $\text{pair-qbs-s-finite } W \otimes_Q X \lambda r. (w, \alpha r) \mu \lambda r. (w, \alpha' r) \mu'$

by(*auto simp: qs.qbs-s-finite-axioms qs'.qbs-s-finite-axioms pair-qbs-s-finite-def*)

have $\llbracket W \otimes_Q X, \lambda r. (w, \alpha' r), \mu' \rrbracket_{sfin} = \llbracket W \otimes_Q X, \lambda r. (w, \alpha r), \mu \rrbracket_{sfin}$

proof(*rule pq.qbs-s-finite-measure-eq[symmetric]*)

fix $f :: - \Rightarrow \text{ennreal}$

assume $f \in W \otimes_Q X \rightarrow_Q \text{qbs-borel}$

then have f : $\text{curry } f w \in X \rightarrow_Q \text{qbs-borel}$

by (*metis assms(1) qbs-morphism-curry qbs-morphism-space*)

show $(\int^+ x. f (w, \alpha x) \partial\mu) = (\int^+ x. f (w, \alpha' x) \partial\mu')$ (**is** *?lhs = ?rhs*)

proof –

have *?lhs* = $(\int^+ x. \text{curry } f w (\alpha x) \partial\mu)$ **by** *simp*

also have ... = $(\int^+ x. \text{curry } f w (\alpha' x) \partial\mu')$

using $h(3)$ f **by**(*auto simp: qbs-s-finite-eq-equiv qbs-s-finite-eq'-def h(1)*)

also have ... = *?rhs* **by** *simp*

finally show *?thesis* .

qed

qed

}

show *?thesis*

by(*simp add: strength-qbs-def rep-qbs-measure-def qbs-measure.rep-def assms(2)*)

(*rule someI2, rule in-Rep-qbs-measure, auto, fact*)

qed

qed

lemma(in *qbs-prob*)
assumes $w \in \text{qbs-space } W$
and $sx = \llbracket X, \alpha, \mu \rrbracket_{sfin}$
shows *strength-qbs-prob*: $\text{qbs-prob } (W \otimes_Q X) (\lambda r. (w, \alpha r)) \mu$
by(*auto intro!*: *qbs-s-finite.qbs-probI*[*OF strength-qbs-s-finite*[*OF assms*]] *prob-space-axioms*)

lemma *strength-qbs-natural*:
assumes $f \in X \rightarrow_Q X'$
 $g \in Y \rightarrow_Q Y'$
 $x \in \text{qbs-space } X$
and $sy \in \text{qbs-space } (\text{monadM-qbs } Y)$
shows $(\text{distr-qbs } (X \otimes_Q Y) (X' \otimes_Q Y') (\text{map-prod } f g) \circ \text{strength-qbs } X Y)$
 $(x, sy) = (\text{strength-qbs } X' Y' \circ \text{map-prod } f (\text{distr-qbs } Y Y' g)) (x, sy)$
(is ?lhs = ?rhs)

proof –
from *rep-qbs-space-monadM*[*OF assms*(4)] **obtain** $\alpha \mu$
where $h: sy = \llbracket Y, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } Y \alpha \mu$ **by** *metis*
have $?lhs = (\text{distr-qbs } (X \otimes_Q Y) (X' \otimes_Q Y') (\text{map-prod } f g)) (\llbracket X \otimes_Q Y,$
 $\lambda r. (x, \alpha r), \mu \rrbracket_{sfin})$
by(*simp add*: *qbs-s-finite.strength-qbs*[*OF h*(2) *assms*(3) *h*(1)])
also have $\dots = \llbracket X' \otimes_Q Y', \text{map-prod } f g \circ (\lambda r. (x, \alpha r)), \mu \rrbracket_{sfin}$
using *assms* **by**(*simp add*: *qbs-s-finite.distr-qbs*[*OF qbs-s-finite.strength-qbs-s-finite*[*OF*
 h (2) *assms*(3) *h*(1)] *refl*])
also have $\dots = \llbracket X' \otimes_Q Y', \lambda r. (f x, (g \circ \alpha) r), \mu \rrbracket_{sfin}$ **by** (*simp add*: *comp-def*)
also have $\dots = ?rhs$
by(*simp add*: *qbs-s-finite.strength-qbs*[*OF qbs-s-finite.distr-qbs-s-finite*[*OF h*(2, 1)
assms(2)] *qbs-morphism-space*[*OF assms*(1, 3)] *qbs-s-finite.distr-qbs*[*OF h*(2, 1) *assms*(2)]])
finally show *?thesis* .
qed

context
begin

interpretation $rr : \text{standard-borel-ne borel } \otimes_M \text{ borel} :: (\text{real } \times \text{ real}) \text{ measure}$
by(*auto intro!*: *pair-standard-borel-ne*)

declare $rr.\text{from-real-to-real}$ [*simplified space-pair-measure, simplified, simp*]

lemma *rr-from-real-to-real-id*[*simp*]: $rr.\text{from-real} \circ rr.\text{to-real} = id$
by(*auto simp*: *comp-def*)

lemma
assumes $\alpha \in \text{qbs-Mx } X$
 $\beta \in \text{qbs-Mx } (\text{monadM-qbs } Y)$
 $\gamma \in \text{qbs-Mx } Y$
 $s\text{-finite-kernel borel borel } k$
and $\beta = (\lambda r. \llbracket Y, \gamma, k r \rrbracket_{sfin})$
shows *strength-qbs-ab-r-s-finite*: $\text{qbs-s-finite } (X \otimes_Q Y) (\text{map-prod } \alpha \gamma \circ$
 $rr.\text{from-real}) (\text{distr } (\text{return borel } r \otimes_M k r) \text{ borel } rr.\text{to-real})$

and *strength-qbs-ab-r*: *strength-qbs* $X Y$ $(\alpha r, \beta r) = \llbracket X \otimes_Q Y, \text{map-prod } \alpha \gamma \circ \text{rr.from-real}, \text{distr (return borel } r \otimes_M k r) \text{ borel rr.to-real} \rrbracket_{\text{sf in}}$ (**is** *?goal2*)

proof –

interpret *k*: *s-finite-kernel* *borel* *borel* *k* **by** *fact*

note $1[\text{measurable-cong}] = \text{sets-return}[\text{of borel } r] k.\text{kernel-sets}[\text{of } r, \text{simplified}]$

show *qbs-s-finite* $(X \otimes_Q Y)$ $(\text{map-prod } \alpha \gamma \circ \text{rr.from-real})$ $(\text{distr (return borel } r \otimes_M k r) \text{ borel rr.to-real})$

using *assms(1,3)* **by** $(\text{auto simp: } qbs\text{-s-finite-def in-Mx-def } qbs\text{-s-finite-axioms-def } qbs\text{-Mx-is-morphisms } r\text{-preserves-product}[\text{symmetric}] \text{ standard-borel-ne. standard-borel intro!: } s\text{-finite-measure.s-finite-measure-distr}[OF \text{ pair-measure-s-finite-measure}[OF \text{ prob-space.s-finite-measure-prob}[OF \text{ prob-space-return}[of } r \text{ borel}]]] k.\text{image-s-finite-measure}[of } r])]$ *qbs-morphism-comp* **where** $Y = qbs\text{-borel} \otimes_Q qbs\text{-borel}$ *qbs-morphism-space* $[OF \text{ qbs-morphism-space}[OF \text{ qbs-morphism-map-prod}]]$ *standard-borel.qbs-morphism-measurable-intro* $[of \text{ borel} :: \text{real measure}]$)

then interpret *qs*: *qbs-s-finite* $X \otimes_Q Y$ *map-prod* $\alpha \gamma \circ \text{rr.from-real}$ *distr* $(\text{return borel } r \otimes_M k r) \text{ borel rr.to-real}$.

interpret *qs2*: *qbs-s-finite* Y γ *k* r

by $(\text{auto simp: } qbs\text{-s-finite-def } k.\text{image-s-finite-measure in-Mx-def } assms \text{ qbs-s-finite-axioms-def } k.\text{kernel-sets})$

interpret *pq*: *pair-qbs-s-finite* $X \otimes_Q Y$ $\lambda l. (\alpha r, \gamma l) k r$ *map-prod* $\alpha \gamma \circ \text{rr.from-real}$ *distr* $(\text{return borel } r \otimes_M k r) \text{ borel rr.to-real}$

by $(\text{auto simp: } \text{pair-qbs-s-finite-def } qs.\text{qbs-s-finite-axioms } qs2.\text{strength-qbs-s-finite}[OF \text{ qbs-Mx-to-X}[OF \text{ assms}(1), \text{of } r] \text{ fun-cong}[OF \text{ assms}(5)]]])$

have $[\text{measurable}]: \text{map-prod } \alpha \gamma \in \text{borel} \otimes_M \text{borel} \rightarrow_M \text{qbs-to-measure } (X \otimes_Q Y)$

proof –

have $\text{map-prod } \alpha \gamma \in \text{qbs-borel} \otimes_Q \text{qbs-borel} \rightarrow_Q X \otimes_Q Y$

using *assms* **by** $(\text{auto intro!: } qbs\text{-morphism-map-prod simp: } qbs\text{-Mx-is-morphisms})$

also have $\dots \subseteq \text{qbs-to-measure } (\text{qbs-borel} \otimes_Q \text{qbs-borel}) \rightarrow_M \text{qbs-to-measure } (X \otimes_Q Y)$

by $(\text{rule } l\text{-preserves-morphisms})$

also have $\dots = \text{borel} \otimes_M \text{borel} \rightarrow_M \text{qbs-to-measure } (X \otimes_Q Y)$

using *rr.lr-sets-ident* *l-preserves-morphisms* **by** $(\text{auto simp add: } r\text{-preserves-product}[\text{symmetric}])$

finally show *?thesis* .

qed

show *?goal2*

unfolding *qs2.strength-qbs* $[OF \text{ qbs-Mx-to-X}[OF \text{ assms}(1), \text{of } r] \text{ fun-cong}[OF \text{ assms}(5)]]$

proof $(\text{rule } pq.\text{qbs-s-finite-measure-eq})$

show $\text{distr } (k r) (\text{qbs-to-measure } (X \otimes_Q Y)) (\lambda l. (\alpha r, \gamma l)) = \text{distr } (\text{distr } (\text{return borel } r \otimes_M k r) \text{ borel rr.to-real}) (\text{qbs-to-measure } (X \otimes_Q Y)) (\text{map-prod } \alpha \gamma \circ \text{rr.from-real})$

(is *?lhs = ?rhs*)

proof –

have *?lhs* $= \text{distr } (k r) (\text{qbs-to-measure } (X \otimes_Q Y)) (\text{map-prod } \alpha \gamma \circ \text{Pair } r)$

by $(\text{simp add: } \text{comp-def})$

also have $\dots = \text{distr } (\text{distr } (k r) (\text{borel} \otimes_M \text{borel}) (\text{Pair } r)) (\text{qbs-to-measure } (X \otimes_Q Y)) (\text{map-prod } \alpha \gamma)$

```

    by(auto intro!: distr-distr[symmetric])
  also have ... = distr (return borel r  $\otimes_M$  k r) (qbs-to-measure (X  $\otimes_Q$  Y))
  (map-prod  $\alpha$   $\gamma$ )
  proof -
    have return borel r  $\otimes_M$  k r = distr (k r) (borel  $\otimes_M$  borel) ( $\lambda l. (r, l)$ )
    by(auto intro!: measure-eqI simp: sets-pair-measure-cong[OF refl 1(2)])
  qs2.emeasure-pair-measure-alt' emeasure-distr nn-integral-return[OF - qs2.measurable-emeasure-Pair']
    thus ?thesis by simp
  qed
  also have ... = ?rhs
  by(simp add: distr-distr comp-def)
  finally show ?thesis .
  qed
  qed
  qed

```

```

lemma strength-qbs-morphism[qbs]: strength-qbs X Y  $\in$  X  $\otimes_Q$  monadM-qbs Y
 $\rightarrow_Q$  monadM-qbs (X  $\otimes_Q$  Y)
proof(rule pair-qbs-morphismI)
  fix  $\alpha$   $\beta$ 
  assume h: $\alpha \in$  qbs-Mx X
     $\beta \in$  qbs-Mx (monadM-qbs Y)
  from rep-qbs-Mx-monadM[OF this(2)] obtain  $\gamma$  k where hb:
     $\beta = (\lambda r. \llbracket Y, \gamma, k r \rrbracket_{sfin}) \gamma \in$  qbs-Mx Y s-finite-kernel borel borel k
  by metis
  have s-finite-kernel borel borel ( $\lambda r. distr (return borel r \otimes_M k r) borel rr.to-real$ )
  by(auto intro!: s-finite-kernel.distr-s-finite-kernel[where Y=borel  $\otimes_M$  borel]
  s-finite-kernel-pair-measure[OF prob-kernel.s-finite-kernel-prob-kernel] simp:hb prob-kernel-def)
  thus ( $\lambda r. strength-qbs X Y (\alpha r, \beta r) \in$  qbs-Mx (monadM-qbs (X  $\otimes_Q$  Y)))
  using strength-qbs-ab-r[OF h hb(2,3,1)] strength-qbs-ab-r-s-finite[OF h hb(2,3,1)]
  by(auto simp: monadM-qbs-Mx qbs-s-finite-def in-Mx-def intro!: exI[where
  x=map-prod  $\alpha$   $\gamma \circ rr.from-real$ ] exI[where x= $\lambda r. distr (return borel r \otimes_M k r)$ 
  borel rr.to-real])
  qed

```

```

lemma bind-qbs-morphism[qbs]: ( $\gg=$ )  $\in$  monadM-qbs X  $\rightarrow_Q$  (X  $\Rightarrow_Q$  monadM-qbs
Y)  $\Rightarrow_Q$  monadM-qbs Y
proof -
  {
    fix f s
    assume h: $f \in$  X  $\rightarrow_Q$  monadM-qbs Y  $s \in$  qbs-space (monadM-qbs X)
    from rep-qbs-space-monadM[OF this(2)] obtain  $\alpha$   $\mu$  where h':
       $s = \llbracket X, \alpha, \mu \rrbracket_{sfin}$  qbs-s-finite X  $\alpha$   $\mu$  by metis
    then interpret qbs-s-finite X  $\alpha$   $\mu$  by simp
    from rep-qbs-Mx-monadM[OF qbs-morphism-Mx[OF h(1) in-Mx]] obtain  $\beta$  k
      where hb: $f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \beta \in$  qbs-Mx Y s-finite-kernel borel
      k by metis
    have join-qbs (distr-qbs ((X  $\Rightarrow_Q$  monadM-qbs Y)  $\otimes_Q$  X) (monadM-qbs Y)
      ( $\lambda fx. fst fx (snd fx)$ ) (strength-qbs (X  $\Rightarrow_Q$  monadM-qbs Y) X (f, s))) = s  $\gg=$  f

```

```

using qbs-s-finite-join-qbs[OF qbs-s-finite.distr-qbs-s-finite[OF strength-qbs-s-finite[of
f X  $\Rightarrow_Q$  monadM-qbs Y, OF h(1) h'(1)] strength-qbs[of f X  $\Rightarrow_Q$  monadM-qbs Y, OF
h(1) h'(1)] qbs-morphism-eval] qbs-s-finite.distr-qbs[OF strength-qbs-s-finite[of f X
 $\Rightarrow_Q$  monadM-qbs Y, OF h(1) h'(1)] strength-qbs[of f X  $\Rightarrow_Q$  monadM-qbs Y, OF
h(1) h'(1)] qbs-morphism-eval] hb(2,3)] hb(1)
by(simp add: bind-qbs[OF h'(1) h(1) hb(2,3,1)] comp-def)
}
thus ?thesis
by(auto intro!: arg-swap-morphism[OF curry-preserves-morphisms[OF qbs-morphism-cong'[of
- join-qbs  $\circ$  (distr-qbs (exp-qbs X (monadM-qbs Y)  $\otimes_Q$  X) (monadM-qbs Y) ( $\lambda fx.$ 
(fst fx) (snd fx)))]  $\circ$  (strength-qbs (exp-qbs X (monadM-qbs Y)) X)]]) qbs-morphism-comp
distr-qbs-morphism' strength-qbs-morphism join-qbs-morphism qbs-morphism-eval
simp: pair-qbs-space)
qed

```

lemma strength-qbs-law1:

```

assumes x  $\in$  qbs-space (unit-quasi-borel  $\otimes_Q$  monadM-qbs X)
shows snd x = (distr-qbs (unit-quasi-borel  $\otimes_Q$  X) X snd  $\circ$  strength-qbs unit-quasi-borel
X) x

```

proof –

obtain α μ **where** h:

```

qbs-s-finite X  $\alpha$   $\mu$  (snd x) =  $\llbracket X, \alpha, \mu \rrbracket_{sf}$ in

```

```

using rep-qbs-space-monadM[of snd x X] assms by (auto simp: pair-qbs-space)

```

metis

```

have [simp]: ( $\lambda ()$ , snd x) = x

```

```

using SigmaE assms by (auto simp: pair-qbs-space)

```

show ?thesis

```

using qbs-s-finite.distr-qbs[OF qbs-s-finite.strength-qbs-s-finite[OF h(1) - h(2), of
fst x unit-quasi-borel] qbs-s-finite.strength-qbs[OF h(1) - h(2)] snd-qbs-morphism]

```

```

by(auto simp: comp-def, simp add: h(2))

```

qed

lemma strength-qbs-law2:

```

assumes x  $\in$  qbs-space ((X  $\otimes_Q$  Y)  $\otimes_Q$  monadM-qbs Z)

```

```

shows (strength-qbs X (Y  $\otimes_Q$  Z)  $\circ$  (map-prod id (strength-qbs Y Z))  $\circ$  ( $\lambda((x,y),z).$ 
(x,(y,z)))) x =

```

```

(distr-qbs ((X  $\otimes_Q$  Y)  $\otimes_Q$  Z) (X  $\otimes_Q$  (Y  $\otimes_Q$  Z)) ( $\lambda((x,y),z).$  (x,(y,z)))

```

```

 $\circ$  strength-qbs (X  $\otimes_Q$  Y) Z) x

```

```

(is ?lhs = ?rhs)

```

proof –

obtain α μ **where** h:

```

qbs-s-finite Z  $\alpha$   $\mu$  snd x =  $\llbracket Z, \alpha, \mu \rrbracket_{sf}$ in

```

```

using rep-qbs-space-monadM[of snd x Z] assms by (auto simp: pair-qbs-space)

```

metis

```

then have ?lhs =  $\llbracket X \otimes_Q Y \otimes_Q Z, \lambda r. (fst (fst x), snd (fst x), \alpha r), \mu \rrbracket_{sf}$ in

```

```

using assms qbs-s-finite.strength-qbs-s-finite[OF h(1) - h(2), of snd (fst x) Y]

```

```

by(auto intro!: qbs-s-finite.strength-qbs simp: pair-qbs-space)

```

also have ... = ?rhs

```

using qbs-s-finite.distr-qbs[OF qbs-s-finite.strength-qbs-s-finite[OF h(1) - h(2), of

```

$\text{fst } x \ X \otimes_Q Y \text{] } \text{qbs-s-finite.strength-qbs[OF } h(1) - h(2), \text{of } \text{fst } x \ X \otimes_Q Y \text{] } \text{qbs-morphism-pair-assoc1} \text{]}$
 assms
by($\text{auto simp: comp-def pair-qbs-space}$)
finally show $?thesis$.
qed

lemma strength-qbs-law3 :
assumes $x \in \text{qbs-space } (X \otimes_Q Y)$
shows $\text{return-qbs } (X \otimes_Q Y) \ x = (\text{strength-qbs } X \ Y \circ (\text{map-prod id } (\text{return-qbs } Y))) \ x$
proof –
interpret $qp: \text{qbs-prob } Y \ \lambda r. \ \text{snd } x \ \text{return borel } 0$
using $\text{assms by (auto simp: prob-space-return pair-qbs-space qbs-prob-def in-Mx-def real-distribution-def real-distribution-axioms-def)}$
show $?thesis$
using $qp.\text{strength-qbs[OF - qp.return-qbs[of snd } x \ Y], \text{of } \text{fst } x \ X] \ qp.\text{return-qbs[OF assms] assms}$
by($\text{auto simp: pair-qbs-space}$)
qed

lemma strength-qbs-law4 :
assumes $x \in \text{qbs-space } (X \otimes_Q \text{monadM-qbs } (\text{monadM-qbs } Y))$
shows $(\text{strength-qbs } X \ Y \circ \text{map-prod id } \text{join-qbs}) \ x = (\text{join-qbs} \circ \text{distr-qbs } (X \otimes_Q \text{monadM-qbs } Y) \ (\text{monadM-qbs } (X \otimes_Q Y))) \ (\text{strength-qbs } X \ Y) \circ \text{strength-qbs } X \ (\text{monadM-qbs } Y)) \ x$
 $(\text{is } ?lhs = ?rhs)$
proof –
from $\text{assms rep-qbs-space-monadM[of snd } x \ \text{monadM-qbs } Y]$ **obtain** $\beta \ \mu$
where $h: \text{qbs-s-finite } (\text{monadM-qbs } Y) \ \beta \ \mu \ \text{snd } x = \llbracket \text{monadM-qbs } Y, \ \beta, \ \mu \rrbracket_{\text{sfine}}$
by ($\text{auto simp: pair-qbs-spacemetis}$)
with $\text{rep-qbs-Mx-monadM[of } \beta \ Y]$ **obtain** $\gamma \ k$
where $h': \gamma \in \text{qbs-Mx } Y \ \text{s-finite-kernel borel borel } k \ \beta = (\lambda r. \llbracket Y, \ \gamma, \ k \ r \rrbracket_{\text{sfine}})$
and $h'': \bigwedge r. \ \text{qbs-s-finite } Y \ \gamma \ (k \ r)$
by($\text{auto simp: qbs-s-finite-def in-Mx-defmetis}$)
have $?lhs = \llbracket X \otimes_Q Y, \ \lambda r. \ (\text{fst } x, \ \gamma \ r), \ \mu \ggg_k k \rrbracket_{\text{sfine}}$
using $\text{qbs-s-finite.strength-qbs[OF qbs-s-finite-join-qbs-s-finite[OF } h \ h'] - \text{qbs-s-finite-join-qbs[OF } h \ h'], \text{of } \text{fst } x \ X] \ \text{assms}$
by($\text{auto simp: pair-qbs-space}$)
also have $\dots = ?rhs$
using $\text{qbs-s-finite-join-qbs[OF qbs-s-finite.distr-qbs-s-finite[OF qbs-s-finite.strength-qbs-s-finite[OF } h(1) - h(2), \text{of } \text{fst } x \ X] \ \text{qbs-s-finite.strength-qbs[OF } h(1) - h(2), \text{of } \text{fst } x] \ \text{strength-qbs-morphism} \text{]}$
 $\text{qbs-s-finite.distr-qbs[OF qbs-s-finite.strength-qbs-s-finite[OF } h(1) - h(2), \text{of } \text{fst } x \ X]$
 $\text{qbs-s-finite.strength-qbs[OF } h(1) - h(2), \text{of } \text{fst } x] \ \text{strength-qbs-morphism} \text{] pair-qbs-MxI}$
 $h'(2), \text{of } \lambda r. \ (\text{fst } x, \ \gamma \ r), \ \text{simplified comp-def qbs-s-finite.strength-qbs[OF } h'' - \text{fun-cong[OF } h'(3), \text{of } \text{fst } x \ X]] \ \text{assms } h'(1)$
by($\text{auto simp: pair-qbs-space qbs-s-finite-def in-Mx-def}$)
finally show $?thesis$.
qed

lemma *distr-qbs-morphism[qbs]*: $\text{distr-qbs } X \ Y \in (X \Rightarrow_Q Y) \rightarrow_Q (\text{monadM-qbs } X \Rightarrow_Q \text{monadM-qbs } Y)$

proof –

have [*simp*]: $\text{distr-qbs } X \ Y = (\lambda f \text{ sx. } \text{sx} \gg \text{return-qbs } Y \circ f)$

by *standard+* (*auto simp: distr-qbs-def*)

show *?thesis*

by *simp*

qed

lemma

assumes $\alpha \in \text{qbs-Mx } X \ \beta \in \text{qbs-Mx } Y$

shows $\text{return-qbs-pair-Mx: return-qbs } (X \otimes_Q Y) (\alpha \ r, \beta \ k) = \llbracket X \otimes_Q Y, \text{map-prod } \alpha \ \beta \circ \text{rr.from-real} \rrbracket_{\text{sfin}}$
 $\text{distr (return borel } r \otimes_M \text{return borel } k) \text{ borel rr.to-real}]_{\text{sfin}}$

and $\text{return-qbs-pair-Mx-prob: qbs-prob } (X \otimes_Q Y) (\text{map-prod } \alpha \ \beta \circ \text{rr.from-real})$
 $(\text{distr (return borel } r \otimes_M \text{return borel } k) \text{ borel rr.to-real})$

proof –

note [*measurable-cong*] = *sets-return[of borel]*

interpret *qp: qbs-prob* $X \otimes_Q Y$ *map-prod* $\alpha \ \beta \circ \text{rr.from-real}$ *distr (return borel*
 $r \otimes_M \text{return borel } k) \text{ borel rr.to-real}$

using *qbs-closed1-dest[OF assms(1)] qbs-closed1-dest[OF assms(2)]*

by (*auto intro!: prob-space.prob-space-distr prob-space-pair simp: comp-def prob-space-return*
pair-qbs-Mx qbs-prob-def in-Mx-def real-distribution-def real-distribution-axioms-def)

show $\text{qbs-prob } (X \otimes_Q Y) (\text{map-prod } \alpha \ \beta \circ \text{rr.from-real}) (\text{distr (return borel } r$
 $\otimes_M \text{return borel } k) \text{ borel rr.to-real})$

by *standard*

show $\text{return-qbs } (X \otimes_Q Y) (\alpha \ r, \beta \ k) = \llbracket X \otimes_Q Y, \text{map-prod } \alpha \ \beta \circ \text{rr.from-real},$
 $\text{distr (return borel } r \otimes_M \text{return borel } k) \text{ borel rr.to-real} \rrbracket_{\text{sfin}}$ (**is** *?lhs = ?rhs*)

proof –

have *?lhs* = $(\text{strength-qbs } X \ Y \circ \text{map-prod id (return-qbs } Y)) (\alpha \ r, \beta \ k)$

by (*rule strength-qbs-law3[of (α r, β k) X Y], insert assms*) (*auto simp:*
qbs-Mx-to-X pair-qbs-space)

also have $\dots = \text{strength-qbs } X \ Y (\alpha \ r, \llbracket Y, \beta, \text{return borel } k \rrbracket_{\text{sfin}})$

using *fun-cong[OF return-qbs-comp[OF assms(2)]] by simp*

also have $\dots = \text{?rhs}$

by (*rule strength-qbs-ab-r[OF assms(1) - assms(2)]*) (*auto intro!: qbs-closed2-dest*
qbs-s-finite.in-space-monadM s-finite-measure.s-finite-kernel-const[of return borel k, simplified
s-finite-kernel-cong-sets[OF - sets-return]] prob-space.s-finite-measure-prob simp: qbs-s-finite-def
in-Mx-def qbs-s-finite-axioms-def assms(2) prob-space-return)

finally show *?thesis* .

qed

qed

lemma *bind-bind-return-distr:*

assumes *s-finite-measure* μ

and *s-finite-measure* ν

and [*measurable-cong*]: *sets* $\mu = \text{sets borel sets } \nu = \text{sets borel}$

shows $\mu \gg_k (\lambda r. \nu \gg_k (\lambda l. \text{distr (return borel } r \otimes_M \text{return borel } l) \text{ borel}$
 $\text{rr.to-real}))$

 = $\text{distr } (\mu \otimes_M \nu) \text{ borel rr.to-real}$


```

    (is ?lhs = ?rhs)
  proof -
    interpret rd1: s-finite-measure  $\mu$  by fact
    interpret rd2: s-finite-measure  $\nu$  by fact
    have ne: space  $\mu \neq \{\}$  space  $\nu \neq \{\}$ 
      by(auto simp: sets-eq-imp-space-eq assms(3,4))

    have ?lhs =  $\mu \ggg_k (\lambda r. \nu \ggg_k (\lambda l. \text{distr } (\text{return } (\text{borel} \otimes_M \text{borel})) (r,l)) \text{ borel } rr.\text{to-real})$ 
      by(simp add: pair-measure-return)
    also have ... =  $\mu \ggg_k (\lambda r. \nu \ggg_k (\lambda l. \text{distr } (\text{return } (\mu \otimes_M \nu)) (r, l)) \text{ borel } rr.\text{to-real})$ 
      by(auto intro!: return-sets-cong sets-pair-measure-cong simp: assms(3,4))
    thus ?thesis by simp
  qed
  also have ... =  $\mu \ggg_k (\lambda r. \text{distr } (\nu \ggg_k (\lambda l. (\text{return } (\mu \otimes_M \nu)) (r, l)))) \text{ borel } rr.\text{to-real}$ 
    by(auto intro!: bind-kernel-cong-All measure-kernel.distr-bind-kernel[of  $\nu \mu \otimes_M \nu, \text{symmetric}$ ] simp: ne measure-kernel-def space-pair-measure)
  also have ... =  $\text{distr } (\mu \ggg_k (\lambda r. \nu \ggg_k (\lambda l. \text{return } (\mu \otimes_M \nu)) (r, l)))) \text{ borel } rr.\text{to-real}$ 
    by(auto intro!: measure-kernel.distr-bind-kernel[of  $\mu \mu \otimes_M \nu, \text{symmetric}$ ] s-finite-kernel.axioms(1) s-finite-kernel.bind-kernel-s-finite-kernel'[where  $Y=\nu$ ] s-finite-measure.s-finite-kernel.assms(2)] prob-kernel.s-finite-kernel-prob-kernel[of  $\mu \otimes_M \nu$ ] simp: ne prob-kernel-def')
  also have ... = ?rhs
    by(simp add: pair-measure-eq-bind-s-finite[OF assms(1,2),symmetric])
  finally show ?thesis .
qed

end

context
begin
interpretation rr : standard-borel-ne borel  $\otimes_M$  borel :: (real  $\times$  real) measure
  by(auto intro!: pair-standard-borel-ne)

lemma from-real-rr-qbs-morphism[qbs]: rr.from-real  $\in$  qbs-borel  $\rightarrow_Q$  qbs-borel  $\otimes_Q$  qbs-borel
  by (metis borel-prod qbs-Mx-R qbs-Mx-is-morphisms qbs-borel-prod rr.from-real-measurable)

end

context pair-qbs-s-finites
begin

interpretation rr : standard-borel-ne borel  $\otimes_M$  borel :: (real  $\times$  real) measure
  by(auto intro!: pair-standard-borel-ne)

```

sublocale $qbs\text{-}s\text{-}finite\ X \otimes_Q Y\ map\text{-}prod\ \alpha\ \beta \circ rr.\text{from}\text{-}real\ distr\ (\mu \otimes_M \nu)$
borel rr.to-real
by(*auto simp: qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def qbs-Mx-is-morphisms*
pq1.s-finite-measure-axioms pq2.s-finite-measure-axioms intro!: s-finite-measure.s-finite-measure-distr[OF
pair-measure-s-finite-measure])

lemma *qbs-bind-bind-return-qp:*

$\llbracket Y, \beta, \nu \rrbracket_{sfin} \gg (\lambda y. \llbracket X, \alpha, \mu \rrbracket_{sfin} \gg (\lambda x. return\text{-}qbs\ (X \otimes_Q Y)\ (x, y))) = \llbracket X$
 $\otimes_Q Y, map\text{-}prod\ \alpha\ \beta \circ rr.\text{from}\text{-}real, distr\ (\mu \otimes_M \nu)\ borel\ rr.\text{to}\text{-}real \rrbracket_{sfin}$ (**is** *?lhs*
 $= ?rhs$)

proof –

have *?lhs* = $\llbracket X \otimes_Q Y, map\text{-}prod\ \alpha\ \beta \circ rr.\text{from}\text{-}real, \nu \gg_k (\lambda l. \mu \gg_k (\lambda r.$
 $distr\ (return\ borel\ r \otimes_M return\ borel\ l)\ borel\ rr.\text{to}\text{-}real)) \rrbracket_{sfin}$

by(*auto intro!: pq2.bind-qbs[OF refl] s-finite-kernel.bind-kernel-s-finite-kernel'[where*
 $Y=\mu]$ *s-finite-measure.s-finite-kernel-const s-finite-kernel.distr-s-finite-kernel[where*
 $Y=borel \otimes_M borel]$ *prob-kernel.s-finite-kernel-prob-kernel[of borel $\otimes_M \mu]$ simp:*
sets-eq-imp-space-eq[OF pq1.mu-sets] pq1.s-finite-measure-axioms split-beta' pair-measure-return[of
 $- snd -]$ *prob-kernel-def')*

(*auto intro!: pq1.bind-qbs prob-kernel.s-finite-kernel-prob-kernel simp: comp-def*
return-qbs-pair-Mx qbs-Mx-is-morphisms prob-kernel-def')

also have ... = *?rhs*

proof –

have $\nu \gg_k (\lambda l. \mu \gg_k (\lambda r. distr\ (return\ borel\ r \otimes_M return\ borel\ l)\ borel$
 $rr.\text{to}\text{-}real)) = distr\ (\mu \otimes_M \nu)\ borel\ rr.\text{to}\text{-}real$

by(*auto simp: bind-bind-return-distr[OF pq1.s-finite-measure-axioms pq2.s-finite-measure-axioms*
pq1.mu-sets pq2.mu-sets, symmetric] pq1.s-finite-measure-axioms pq2.s-finite-measure-axioms
prob-kernel-def' intro!: bind-kernel-rotate[where $Z=borel$] prob-kernel.s-finite-kernel-prob-kernel)

thus *?thesis* **by** *simp*

qed

finally show *?thesis* .

qed

lemma *qbs-bind-bind-return-pq:*

$\llbracket X, \alpha, \mu \rrbracket_{sfin} \gg (\lambda x. \llbracket Y, \beta, \nu \rrbracket_{sfin} \gg (\lambda y. return\text{-}qbs\ (X \otimes_Q Y)\ (x, y))) = \llbracket X$
 $\otimes_Q Y, map\text{-}prod\ \alpha\ \beta \circ rr.\text{from}\text{-}real, distr\ (\mu \otimes_M \nu)\ borel\ rr.\text{to}\text{-}real \rrbracket_{sfin}$ (**is** *?lhs*
 $= ?rhs$)

proof –

have *?lhs* = $\llbracket X \otimes_Q Y, map\text{-}prod\ \alpha\ \beta \circ rr.\text{from}\text{-}real, \mu \gg_k (\lambda r. \nu \gg_k (\lambda l.$
 $distr\ (return\ borel\ r \otimes_M return\ borel\ l)\ borel\ rr.\text{to}\text{-}real)) \rrbracket_{sfin}$

by(*auto intro!: pq1.bind-qbs[OF refl] s-finite-kernel.bind-kernel-s-finite-kernel'[where*
 $Y=\nu]$ *s-finite-measure.s-finite-kernel-const s-finite-kernel.distr-s-finite-kernel[where*
 $Y=borel \otimes_M borel]$ *prob-kernel.s-finite-kernel-prob-kernel[of borel $\otimes_M \nu]$ simp:*
sets-eq-imp-space-eq[OF pq2.mu-sets] pq2.s-finite-measure-axioms split-beta' pair-measure-return[of
 $- fst -]$ *prob-kernel-def')*

(*auto intro!: pq2.bind-qbs prob-kernel.s-finite-kernel-prob-kernel simp: comp-def*
return-qbs-pair-Mx qbs-Mx-is-morphisms prob-kernel-def')

also have ... = *?rhs*

by(*simp add: bind-bind-return-distr[OF pq1.s-finite-measure-axioms pq2.s-finite-measure-axioms*

pq1.mu-sets pq2.mu-sets)
finally show *?thesis* .
qed

end

lemma *bind-qbs-return-rotate*:

assumes $p \in \text{qbs-space } (\text{monadM-qbs } X)$
and $q \in \text{qbs-space } (\text{monadM-qbs } Y)$
shows $q \gg (\lambda y. p \gg (\lambda x. \text{return-qbs } (X \otimes_Q Y) (x,y))) = p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y)))$
proof –
from *rep-qbs-space-monadM[OF assms(1)] rep-qbs-space-monadM[OF assms(2)]*
obtain $\alpha \mu \beta \nu$ **where** $h: p = \llbracket X, \alpha, \mu \rrbracket_{\text{sf in}} q = \llbracket Y, \beta, \nu \rrbracket_{\text{sf in}} \text{qbs-s-finite } X \alpha \mu \text{qbs-s-finite } Y \beta \nu$
by *metis*
then interpret *pair-qbs-s-finites* $X \alpha \mu Y \beta \nu$
by(*simp add: pair-qbs-s-finites-def*)
show *?thesis*
by(*simp add: h(1,2) qbs-bind-bind-return-pq qbs-bind-bind-return-qp*)
qed

lemma *qbs-bind-bind-return1*:

assumes $[qbs]: f \in X \otimes_Q Y \rightarrow_Q \text{monadM-qbs } Z$
 $p \in \text{qbs-space } (\text{monadM-qbs } X)$
 $q \in \text{qbs-space } (\text{monadM-qbs } Y)$
shows $q \gg (\lambda y. p \gg (\lambda x. f (x,y))) = (q \gg (\lambda y. p \gg (\lambda x. \text{return-qbs } (X \otimes_Q Y) (x,y)))) \gg f$
(is ?lhs = ?rhs)
proof –
have $?lhs = q \gg (\lambda y. p \gg (\lambda x. \text{return-qbs } (X \otimes_Q Y) (x,y) \gg f))$
by(*auto intro!: bind-qbs-cong[OF assms(3),where Y=Z] bind-qbs-cong[OF assms(2),where Y=Z] simp: bind-qbs-return[OF assms(1),simplified pair-qbs-space]*)
also have $\dots = q \gg (\lambda y. (p \gg (\lambda x. \text{return-qbs } (X \otimes_Q Y) (x,y))) \gg f)$
by(*auto intro!: bind-qbs-cong[OF assms(3),where Y=Z] bind-qbs-assoc[OF assms(2) - assms(1)] simp:)*)
also have $\dots = ?rhs$
by(*simp add: bind-qbs-assoc[OF assms(3) - assms(1)]*)
finally show *?thesis* .
qed

lemma *qbs-bind-bind-return2*:

assumes $[qbs]: f \in X \otimes_Q Y \rightarrow_Q \text{monadM-qbs } Z$
 $p \in \text{qbs-space } (\text{monadM-qbs } X)$ $q \in \text{qbs-space } (\text{monadM-qbs } Y)$
shows $p \gg (\lambda x. q \gg (\lambda y. f (x,y))) = (p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y)))) \gg f$
(is ?lhs = ?rhs)
proof –
have $?lhs = p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y) \gg f))$

by(*auto intro!*: *bind-qbs-cong*[*OF assms*(2),**where** $Y=Z$] *bind-qbs-cong*[*OF assms*(3),**where** $Y=Z$] *simp*: *bind-qbs-return*[*OF assms*(1),*simplified pair-qbs-space*])
also have ... = $p \gg (\lambda x. (q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y))) \gg f)$
by(*auto intro!*: *bind-qbs-cong*[*OF assms*(2),**where** $Y=Z$] *bind-qbs-assoc*[*OF assms*(3) - *assms*(1)])
also have ... = *?rhs*
by(*simp add*: *bind-qbs-assoc*[*OF assms*(2) - *assms*(1)])
finally show *?thesis* .
qed

corollary *bind-qbs-rotate*:

assumes $f \in X \otimes_Q Y \rightarrow_Q \text{monadM-qbs } Z$
 $p \in \text{qbs-space } (\text{monadM-qbs } X)$
and $q \in \text{qbs-space } (\text{monadM-qbs } Y)$
shows $q \gg (\lambda y. p \gg (\lambda x. f (x,y))) = p \gg (\lambda x. q \gg (\lambda y. f (x,y)))$
by(*simp add*: *qbs-bind-bind-return1*[*OF assms*] *qbs-bind-bind-return2*[*OF assms*] *bind-qbs-return-rotate assms*)

context *pair-qbs-s-finites*

begin

interpretation *rr* : *standard-borel-ne borel* \otimes_M *borel* :: (*real* \times *real*) *measure*

by(*auto intro!*: *pair-standard-borel-ne*)

lemma

assumes [*qbs*]: $f \in X \otimes_Q Y \rightarrow_Q Z$
shows *qbs-bind-bind-return*: $\llbracket X, \alpha, \mu \rrbracket_{sfin} \gg (\lambda x. \llbracket Y, \beta, \nu \rrbracket_{sfin} \gg (\lambda y. \text{return-qbs } Z (f (x,y)))) = \llbracket Z, f \circ (\text{map-prod } \alpha \beta \circ \text{rr.from-real}), \text{distr } (\mu \otimes_M \nu) \text{ borel } \text{rr.to-real} \rrbracket_{sfin}$ (**is** *?lhs* = *?rhs*)
and *qbs-bind-bind-return-s-finite*: *qbs-s-finite* $Z (f \circ (\text{map-prod } \alpha \beta \circ \text{rr.from-real}))$ (*distr* $(\mu \otimes_M \nu)$ *borel rr.to-real*)
proof –
show *qbs-s-finite* $Z (f \circ (\text{map-prod } \alpha \beta \circ \text{rr.from-real}))$ (*distr* $(\mu \otimes_M \nu)$ *borel rr.to-real*)
using *qbs-s-finite-axioms* **by**(*auto simp*: *qbs-s-finite-def in-Mx-def qbs-Mx-is-morphisms*)
have *?lhs* = $\llbracket X, \alpha, \mu \rrbracket_{sfin} \gg (\lambda x. \llbracket Y, \beta, \nu \rrbracket_{sfin} \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y))) \gg \text{return-qbs } Z \circ f$
by(*auto simp*: *comp-def intro!*: *qbs-bind-bind-return2*[*of return-qbs Z* $\circ f$ - - Z , *simplified comp-def*])
also have ... = $\llbracket X \otimes_Q Y, \text{map-prod } \alpha \beta \circ \text{rr.from-real}, \text{distr } (\mu \otimes_M \nu) \text{ borel } \text{rr.to-real} \rrbracket_{sfin} \gg \text{return-qbs } Z \circ f$
by(*simp add*: *qbs-bind-bind-return-pq*)
also have ... = *?rhs*
by(*rule distr-qbs*[*OF refl assms, simplified distr-qbs-def*])
finally show *?lhs* = *?rhs* .
qed

end

4.1.9 The Probability Monad

definition $monadP\text{-}qbs\ X \equiv sub\text{-}qbs\ (monadM\text{-}qbs\ X)\ \{s.\ prob\text{-}space\ (qbs\text{-}l\ s)\}$

lemma

shows $qbs\text{-}space\text{-}monadPM: s \in qbs\text{-}space\ (monadP\text{-}qbs\ X) \implies s \in qbs\text{-}space\ (monadM\text{-}qbs\ X)$

and $qbs\text{-}Mx\text{-}monadPM: f \in qbs\text{-}Mx\ (monadP\text{-}qbs\ X) \implies f \in qbs\text{-}Mx\ (monadM\text{-}qbs\ X)$

by($simp\text{-}all\ add: monadP\text{-}qbs\text{-}def\ sub\text{-}qbs\text{-}space\ sub\text{-}qbs\text{-}Mx$)

lemma $monadP\text{-}qbs\text{-}space: qbs\text{-}space\ (monadP\text{-}qbs\ X) = \{s.\ qbs\text{-}space\text{-}of\ s = X \wedge prob\text{-}space\ (qbs\text{-}l\ s)\}$

by($auto\ simp: monadP\text{-}qbs\text{-}def\ sub\text{-}qbs\text{-}space\ monadM\text{-}qbs\text{-}space$)

lemma $rep\text{-}qbs\text{-}space\text{-}monadP:$

assumes $s \in qbs\text{-}space\ (monadP\text{-}qbs\ X)$

obtains $\alpha\ \mu$ **where** $s = \llbracket X, \alpha, \mu \rrbracket_{sf\ in}\ qbs\text{-}prob\ X\ \alpha\ \mu$

proof –

obtain $\alpha\ \mu$ **where** $h: s = \llbracket X, \alpha, \mu \rrbracket_{sf\ in}\ qbs\text{-}s\text{-}finite\ X\ \alpha\ \mu$

using $assms\ rep\text{-}qbs\text{-}space\text{-}monadM[of\ s\ X]$ **by**($auto\ simp: monadP\text{-}qbs\text{-}def\ sub\text{-}qbs\text{-}space$)

interpret $qbs\text{-}s\text{-}finite\ X\ \alpha\ \mu$ **by** *fact*

have $prob\text{-}space\ \mu$

by($rule\ prob\text{-}space\text{-}distrD[of\ \alpha - qbs\text{-}to\text{-}measure\ X]$) (*insert\ assms, auto\ simp: qbs\text{-}l[symmetric]\ h(1)[symmetric]\ monadP\text{-}qbs\text{-}space*)

thus *?thesis*

by ($simp\ add: h(1)\ in\text{-}Mx\text{-}axioms\ mu\text{-}sets\ qbs\text{-}prob.\ intro\ real\text{-}distribution\text{-}axioms\text{-}def\ real\text{-}distribution\text{-}def\ that$)

qed

lemma $qbs\text{-}l\text{-}prob\text{-}space:$

$s \in qbs\text{-}space\ (monadP\text{-}qbs\ X) \implies prob\text{-}space\ (qbs\text{-}l\ s)$

by($auto\ simp: monadP\text{-}qbs\text{-}space$)

lemma $monadP\text{-}qbs\text{-}empty\text{-}iff:$

$(qbs\text{-}space\ X = \{\}) = (qbs\text{-}space\ (monadP\text{-}qbs\ X) = \{\})$

proof

show $qbs\text{-}space\ X = \{\} \implies qbs\text{-}space\ (monadP\text{-}qbs\ X) = \{\}$

using $qbs\text{-}s\text{-}space\text{-}of\text{-}not\text{-}empty$ **by**($auto\ simp\ add: monadP\text{-}qbs\text{-}space$)

next

assume $qbs\text{-}space\ (monadP\text{-}qbs\ X) = \{\}$

then **have** $h: \bigwedge s.\ qbs\text{-}space\text{-}of\ s = X \implies \neg\ prob\text{-}space\ (qbs\text{-}l\ s)$

by($simp\ add: monadP\text{-}qbs\text{-}space$)

show $qbs\text{-}space\ X = \{\}$

proof($rule\ ccontr$)

assume $qbs\text{-}space\ X \neq \{\}$

then **obtain** a **where** $a: a \in qbs\text{-}Mx\ X$ **by** ($auto\ simp: qbs\text{-}empty\text{-}equiv$)

then **interpret** $qbs\text{-}prob\ X$ *a return borel 0*

by($auto\ simp: qbs\text{-}prob\text{-}def\ in\text{-}Mx\text{-}def\ real\text{-}distribution\text{-}axioms\text{-}def\ real\text{-}distribution\text{-}def$)

prob-space-return)
have *qbs-space-of* $\llbracket X, a, \text{return borel } 0 \rrbracket_{sfin} = X \text{ prob-space } (qbs-l \llbracket X, a, \text{return borel } 0 \rrbracket_{sfin})$
by (*auto simp: qbs-l intro!: prob-space-distr*)
with *h show False by simp*
qed
qed

lemma *in-space-monadP-qbs-pred: qbs-pred (monadM-qbs X) ($\lambda s. s \in \text{monadP-qbs } X$)*
by (*rule qbs-morphism-cong'[where f= $\lambda s. \text{prob-space } (qbs-l s)$], auto simp: qbs-l-prob-pred*)
(auto simp: monadP-qbs-def sub-qbs-space)

lemma (*in qbs-prob*) *in-space-monadP[qbs]: $\llbracket X, \alpha, \mu \rrbracket_{sfin} \in \text{qbs-space } (monadP-qbs X)$*
by (*auto simp: monadP-qbs-space qbs-l prob-space-distr*)

lemma *qbs-morphism-monadPD: $f \in X \rightarrow_Q \text{monadP-qbs } Y \implies f \in X \rightarrow_Q \text{monadM-qbs } Y$*
unfolding *monadP-qbs-def by (rule qbs-morphism-subD)*

lemma *qbs-morphism-monadPD': $f \in \text{monadM-qbs } X \rightarrow_Q Y \implies f \in \text{monadP-qbs } X \rightarrow_Q Y$*
unfolding *monadP-qbs-def by (rule qbs-morphism-subI2)*

lemma *qbs-morphism-monadPI:*
assumes $\bigwedge x. x \in \text{qbs-space } X \implies \text{prob-space } (qbs-l (f x)) f \in X \rightarrow_Q \text{monadM-qbs } Y$
shows $f \in X \rightarrow_Q \text{monadP-qbs } Y$
using *assms by (auto simp: monadP-qbs-def intro!: qbs-morphism-subI1)*

lemma *qbs-morphism-monadPI':*
assumes $\bigwedge x. x \in \text{qbs-space } X \implies f x \in \text{qbs-space } (monadP-qbs Y) f \in X \rightarrow_Q \text{monadM-qbs } Y$
shows $f \in X \rightarrow_Q \text{monadP-qbs } Y$
using *assms by (auto intro!: qbs-morphism-monadPI simp: monadP-qbs-space)*

lemma *qbs-morphism-monadPI'':*
assumes $f \in \text{monadM-qbs } X \rightarrow_Q \text{monadM-qbs } Y \bigwedge s. s \in \text{qbs-space } (monadP-qbs X) \implies f s \in \text{qbs-space } (monadP-qbs Y)$
shows $f \in \text{monadP-qbs } X \rightarrow_Q \text{monadP-qbs } Y$
proof –
have $1: \bigwedge X. \text{monadP-qbs } X = \text{sub-qbs } (monadM-qbs X) \{s. \text{qbs-space-of } s = X \wedge \text{prob-space } (qbs-l s)\}$ (**is** $\bigwedge X. ?l X = ?r X$)
proof –
fix *X*
have $?l X = \text{sub-qbs } (sub-qbs (monadM-qbs X) (\text{qbs-space } (monadM-qbs X))) \{s. \text{prob-space } (qbs-l s)\}$
by (*simp add: sub-qbs-ident monadP-qbs-def*)

also have ... = ?r X
by(*auto simp: sub-qbs-sub-qbs monadM-qbs-space Collect-conj-eq*)
finally show ?l X = ?r X .
qed
show ?thesis
unfolding 1 **using** *assms(2)* **by**(*auto intro!: qbs-morphism-subsubI[OF assms(1), of {s. qbs-space-of s = X ∧ prob-space (qbs-l s)} {s. qbs-space-of s = Y ∧ prob-space (qbs-l s)}] simp: 1 sub-qbs-space monadM-qbs-space*)
qed

lemma *monadP-qbs-Mx: qbs-Mx (monadP-qbs X) = {λr. [X, α, k r]_{sfin} | α k. α ∈ qbs-Mx X ∧ k ∈ borel →_M prob-algebra borel}*
proof *safe*
fix γ
assume *h: γ ∈ qbs-Mx (monadP-qbs X)*
then obtain α k **where** *h1:*
γ = (λr. [X, α, k r]_{sfin}) α ∈ qbs-Mx X *s-finite-kernel borel borel k ∧ r. qbs-s-finite X α (k r)*
using *rep-qbs-Mx-monadM[of γ X]* **by**(*simp add: monadP-qbs-def sub-qbs-Mx*)
metis
interpret *s-finite-kernel borel borel k by fact*
have γ ∈ UNIV → {s. qbs-space-of s = X ∧ prob-space (qbs-l s)}
using *h qbs-Mx-to-X[OF h]* **by**(*auto simp: monadP-qbs-def sub-qbs-Mx monadM-qbs-space sub-qbs-space*)
hence ∧r. prob-space (k r)
using *h1(2)* **by**(*auto simp add: h1(1) Pi-iff qbs-s-finite.qbs-l[OF h1(4)] intro!: prob-space-distrD[of α - qbs-to-measure X]*)
hence prob-kernel borel borel k
by(*auto simp: prob-kernel-def prob-kernel-axioms-def measure-kernel-axioms*)
with *h1(1,2)* **show** ∃α k. γ = (λr. [X, α, k r]_{sfin}) ∧ α ∈ qbs-Mx X ∧ k ∈ borel →_M prob-algebra borel
by(*auto intro!: exI[where x=α] exI[where x=k] simp: prob-kernel-def'*)
next
fix α **and** k :: real ⇒ real measure
assume *h: α ∈ qbs-Mx X k ∈ borel →_M prob-algebra borel*
then interpret *pk: prob-kernel borel borel k*
by(*simp add: prob-kernel-def'[symmetric]*)
have *qp: qbs-prob X α (k r) for r*
using *h* **by**(*auto simp: qbs-prob-def in-Mx-def pk.kernel-sets pk.prob-spaces real-distribution-axioms-def real-distribution-def*)
show (λr. [X, α, k r]_{sfin}) ∈ qbs-Mx (monadP-qbs X)
using *h(1) qp* **by**(*auto simp: monadP-qbs-def sub-qbs-Mx monadM-qbs-space qbs-s-finite.qbs-l[OF qbs-prob.qbs-s-finite[OF qp]] qbs-s-finite.qbs-space-of[OF qbs-prob.qbs-s-finite[OF qp]] monadM-qbs-Mx qbs-prob-def real-distribution-def intro!: exI[where x=α] exI[where x=k] h pk.s-finite-kernel-axioms prob-space.prob-space-distr*)
qed

lemma *rep-qbs-Mx-monadP:*
assumes γ ∈ qbs-Mx (monadP-qbs X)

obtains α k **where** $\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{sf\ i\ n}) \alpha \in \text{qbs-Mx } X \ k \in \text{borel} \rightarrow_M \text{prob-algebra borel} \wedge r. \text{qbs-prob } X \ \alpha \ (k \ r)$
proof –
have $\wedge \alpha \ r \ k. \alpha \in \text{qbs-Mx } X \implies k \in \text{borel} \rightarrow_M \text{prob-algebra borel} \implies \text{qbs-prob } X \ \alpha \ (k \ r)$
by(*auto simp: qbs-prob-def in-Mx-def real-distribution-def real-distribution-axioms-def prob-kernel-def [symmetric] prob-kernel-def prob-kernel-axioms-def measure-kernel-def*)
thus *?thesis*
using *assms that* **by**(*fastforce simp: monadP-qbs-Mx*)
qed

lemma *qbs-l-monadP-le1*: $s \in \text{qbs-space } (\text{monadP-qbs } X) \implies \text{qbs-l } s \ A \leq 1$
by(*auto simp: monadP-qbs-space intro!: prob-space.emmeasure-le-1*)

lemma *qbs-l-inj-P*: *inj-on qbs-l* ($\text{qbs-space } (\text{monadP-qbs } X)$)
by(*auto intro!: inj-on-subset [OF qbs-l-inj] simp: monadP-qbs-def sub-qbs-space*)

lemma *qbs-l-measurable-prob*[*measurable*]: $\text{qbs-l} \in \text{qbs-to-measure } (\text{monadP-qbs } X) \rightarrow_M \text{prob-algebra } (\text{qbs-to-measure } X)$

proof(*rule qbs-morphism-dest [OF qbs-morphismI]*)

fix γ
assume $\gamma \in \text{qbs-Mx } (\text{monadP-qbs } X)$
from *rep-qbs-Mx-monadP [OF this]* **obtain** α k **where** $h[\text{measurable}]$:
 $\gamma = (\lambda r. \llbracket X, \alpha, k \ r \rrbracket_{sf\ i\ n}) \alpha \in \text{qbs-Mx } X \ k \in \text{borel} \rightarrow_M \text{prob-algebra borel} \wedge r. \text{qbs-prob } X \ \alpha \ (k \ r)$
by *metis*
show $\text{qbs-l} \circ \gamma \in \text{qbs-Mx } (\text{measure-to-qbs } (\text{prob-algebra } (\text{qbs-to-measure } X)))$
by(*auto simp: qbs-Mx-R comp-def h(1) qbs-s-finite.qbs-l [OF qbs-prob.qbs-s-finite [OF h(4)]]*)
qed

lemma *return-qbs-morphismP*: $\text{return-qbs } X \in X \rightarrow_Q \text{monadP-qbs } X$

proof(*rule qbs-morphismI*)

interpret *rr* : *real-distribution return borel 0*
by(*simp add: real-distribution-def real-distribution-axioms-def prob-space-return*)
fix α
assume $h: \alpha \in \text{qbs-Mx } X$
then have $1: \text{return-qbs } X \circ \alpha = (\lambda r. \llbracket X, \alpha, \text{return borel } r \rrbracket_{sf\ i\ n})$
by(*rule return-qbs-comp*)
show $\text{return-qbs } X \circ \alpha \in \text{qbs-Mx } (\text{monadP-qbs } X)$
by(*auto simp: 1 monadP-qbs-Mx h intro!: exI [where x= α] exI [where x=*return borel*]*)
qed

lemma(*in qbs-prob*)

assumes $s = \llbracket X, \alpha, \mu \rrbracket_{sf\ i\ n}$
 $f \in X \rightarrow_Q \text{monadP-qbs } Y$
 $\beta \in \text{qbs-Mx } Y$
and $g[\text{measurable}]: g \in \text{borel} \rightarrow_M \text{prob-algebra borel}$

and $(f \circ \alpha) = (\lambda r. \llbracket Y, \beta, g r \rrbracket_{sfin})$
shows $bind\text{-}qbs\text{-}prob:qbs\text{-}prob\ Y\ \beta\ (\mu \ggg g)$
and $bind\text{-}qbs': s \ggg f = \llbracket Y, \beta, \mu \ggg g \rrbracket_{sfin}$
proof –
interpret $prob\text{-}kernel\ borel\ borel\ g$
using $assms(4)$ **by** $(simp\ add: prob\text{-}kernel\text{-}def')$
have $prob\text{-}space\ (\mu \ggg g)$
by $(auto\ intro!: prob\text{-}space\text{-}bind'[OF\ -\ g]\ simp: space\text{-}prob\text{-}algebra\ prob\text{-}space\text{-}axioms)$
thus $qbs\text{-}prob\ Y\ \beta\ (\mu \ggg g)\ s \ggg f = \llbracket Y, \beta, \mu \ggg g \rrbracket_{sfin}$
using $qbs\text{-}s\text{-}finite.qbs\text{-}probI[OF\ bind\text{-}qbs\text{-}s\text{-}finite[OF\ assms(1)\ qbs\text{-}morphism\text{-}monadPD[OF\ assms(2)]\ assms(3)\ s\text{-}finite\text{-}kernel\text{-}axioms\ assms(5)]]$
by $(simp\text{-}all\ add: bind\text{-}qbs[OF\ assms(1)\ qbs\text{-}morphism\text{-}monadPD[OF\ assms(2)]\ assms(3)\ s\text{-}finite\text{-}kernel\text{-}axioms\ assms(5)]\ bind\text{-}kernel\text{-}bind[of\ g\ \mu\ borel])$
qed

lemma $bind\text{-}qbs\text{-}morphism'P$:

assumes $f \in X \rightarrow_Q\ monadP\text{-}qbs\ Y$
shows $(\lambda x. x \ggg f) \in monadP\text{-}qbs\ X \rightarrow_Q\ monadP\text{-}qbs\ Y$
proof $(safe\ intro!: qbs\text{-}morphism\text{-}monadPI')$
fix x
assume $x \in qbs\text{-}space\ (monadP\text{-}qbs\ X)$
from $rep\text{-}qbs\text{-}space\text{-}monadP[OF\ this]$ **obtain** $\alpha\ \mu$ **where** $h:x = \llbracket X, \alpha, \mu \rrbracket_{sfin}$
 $qbs\text{-}prob\ X\ \alpha\ \mu$
by $metis$
then interpret $qbs\text{-}prob\ X\ \alpha\ \mu$ **by** $simp$
from $rep\text{-}qbs\text{-}Mx\text{-}monadP[OF\ qbs\text{-}morphism\text{-}Mx[OF\ assms\ in\ Mx]]$ **obtain** $\beta\ g$
where $h'[measurable]$:
 $f \circ \alpha = (\lambda r. \llbracket Y, \beta, g r \rrbracket_{sfin})\ \beta \in qbs\text{-}Mx\ Y\ g \in borel \rightarrow_M\ prob\text{-}algebra\ borel$ **by**
 $metis$
show $x \ggg f \in qbs\text{-}space\ (monadP\text{-}qbs\ Y)$
using $sets\text{-}bind[of\ \mu\ g]\ measurable\text{-}space[OF\ h'(3),simplified\ space\text{-}prob\text{-}algebra]$
by $(auto\ simp: qbs\text{-}prob.bind\text{-}qbs'[OF\ h(2,1)\ assms\ h'(2,3,1)]\ qbs\text{-}prob\text{-}def\ in\ Mx\text{-}def\ h'(2)\ real\text{-}distribution\text{-}def\ real\text{-}distribution\text{-}axioms\text{-}def\ intro!: qbs\text{-}prob.in\text{-}space\text{-}monadP\ prob\text{-}space\text{-}bind[where\ S=borel]\ measurable\text{-}prob\text{-}algebraD)$
qed $(auto\ intro!: qbs\text{-}morphism\text{-}monadPD'\ bind\text{-}qbs\text{-}morphism'[OF\ qbs\text{-}morphism\text{-}monadPD[OF\ assms]])$

lemma $distr\text{-}qbs\text{-}morphism'P$:

assumes $f \in X \rightarrow_Q\ Y$
shows $distr\text{-}qbs\ X\ Y\ f \in monadP\text{-}qbs\ X \rightarrow_Q\ monadP\text{-}qbs\ Y$
unfolding $distr\text{-}qbs\text{-}def$
by $(rule\ bind\text{-}qbs\text{-}morphism'P[OF\ qbs\text{-}morphism\text{-}comp[OF\ assms\ return\text{-}qbs\text{-}morphismP]])$

lemma $join\text{-}qbs\text{-}morphismP$: $join\text{-}qbs \in monadP\text{-}qbs\ (monadP\text{-}qbs\ X) \rightarrow_Q\ monadP\text{-}qbs\ X$

by $(simp\ add: join\text{-}qbs\text{-}def\ bind\text{-}qbs\text{-}morphism'P[OF\ qbs\text{-}morphism\text{-}ident])$

lemma

assumes $qbs\text{-}prob\ (monadP\text{-}qbs\ X)\ \beta\ \mu$

```

     $ssx = \llbracket \text{monadP-qbs } X, \beta, \mu \rrbracket_{sfin}$ 
     $\alpha \in \text{qbs-Mx } X$ 
     $g \in \text{borel} \rightarrow_M \text{prob-algebra borel}$ 
    and  $\beta = (\lambda r. \llbracket X, \alpha, g r \rrbracket_{sfin})$ 
    shows qbs-prob-join-qbs-s-finite:  $\text{qbs-prob } X \alpha (\mu \ggg g)$ 
    and qbs-prob-join-qbs:  $\text{join-qbs } ssx = \llbracket X, \alpha, \mu \ggg g \rrbracket_{sfin}$ 
    using qbs-prob.bind-qbs'[OF assms(1,2) qbs-morphism-ident assms(3,4)] qbs-prob.bind-qbs-prob[OF
assms(1,2) qbs-morphism-ident assms(3,4)]
    by(auto simp: assms(5) join-qbs-def)

```

context
begin

interpretation $rr : \text{standard-borel-ne borel} \otimes_M \text{borel} :: (\text{real} \times \text{real}) \text{ measure}$
by(*auto intro!: pair-standard-borel-ne*)

lemma *strength-qbs-ab-r-prob*:

```

    assumes  $\alpha \in \text{qbs-Mx } X$ 
            $\beta \in \text{qbs-Mx } (\text{monadP-qbs } Y)$ 
            $\gamma \in \text{qbs-Mx } Y$ 
    and [measurable]:  $g \in \text{borel} \rightarrow_M \text{prob-algebra borel}$ 
    and  $\beta = (\lambda r. \llbracket Y, \gamma, g r \rrbracket_{sfin})$ 
    shows  $\text{qbs-prob } (X \otimes_Q Y) (\text{map-prod } \alpha \gamma \circ rr.\text{from-real}) (\text{distr } (\text{return borel } r \otimes_M g r) \text{ borel } rr.\text{to-real})$ 
    using measurable-space[OF assms(4), of r] sets-return[of borel r]
    by(auto intro!: qbs-s-finite.qbs-probI strength-qbs-ab-r-s-finite[OF assms(1) qbs-Mx-monadPM[OF
assms(2)] assms(3) prob-kernel.s-finite-kernel-prob-kernel assms(5), simplified prob-kernel-def', OF
assms(4)] prob-space.prob-space-distr prob-space-pair prob-space-return simp: space-prob-algebra
simp del: sets-return)

```

lemma *strength-qbs-morphismP*: $\text{strength-qbs } X Y \in X \otimes_Q \text{monadP-qbs } Y \rightarrow_Q \text{monadP-qbs } (X \otimes_Q Y)$

proof(*rule pair-qbs-morphismI*)

```

    fix  $\alpha \beta$ 
    assume  $h: \alpha \in \text{qbs-Mx } X$ 
            $\beta \in \text{qbs-Mx } (\text{monadP-qbs } Y)$ 
    from rep-qbs-Mx-monadP[OF this(2)] obtain  $\gamma g$  where hb[measurable]:
            $\beta = (\lambda r. \llbracket Y, \gamma, g r \rrbracket_{sfin}) \gamma \in \text{qbs-Mx } Y g \in \text{borel} \rightarrow_M \text{prob-algebra borel}$ 
    by metis
    show  $(\lambda r. \text{strength-qbs } X Y (\alpha r, \beta r)) \in \text{qbs-Mx } (\text{monadP-qbs } (X \otimes_Q Y))$ 
    using strength-qbs-ab-r-prob[OF h hb(2,3,1)] strength-qbs-ab-r[OF h(1) qbs-Mx-monadPM[OF
h(2)] hb(2) prob-kernel.s-finite-kernel-prob-kernel hb(1), simplified prob-kernel-def', OF
hb(3)]
    by(auto simp: monadP-qbs-Mx qbs-prob-def in-Mx-def intro!: exI[where  $x = \text{map-prod } \alpha \gamma \circ rr.\text{from-real}$ ] exI[where  $x = \lambda r. \text{distr } (\text{return borel } r \otimes_M g r) \text{ borel } rr.\text{to-real}$ ])
    qed

```

end

lemma *bind-qbs-morphismP*: $(\gg=) \in \text{monadP-qbs } X \rightarrow_Q (X \Rightarrow_Q \text{monadP-qbs } Y) \Rightarrow_Q \text{monadP-qbs } Y$

proof –

```

{
  fix f s
  assume h: f ∈ X →Q monadP-qbs Y s ∈ qbs-space (monadP-qbs X)
  from rep-qbs-space-monadP[OF this(2)] obtain α μ where h':
    s =  $\llbracket X, \alpha, \mu \rrbracket_{sfin}$  qbs-prob X α μ by metis
  then interpret qbs-prob X α μ by simp
  from rep-qbs-Mx-monadP[OF qbs-morphism-Mx[OF h(1) in-Mx]] obtain β g
    where hb[measurable]: f ∘ α = (λr.  $\llbracket Y, \beta, g \rrbracket_{sfin}$ ) β ∈ qbs-Mx Y g ∈ borel
  →M prob-algebra borel by metis
  have join-qbs (distr-qbs ((X ⇒Q monadP-qbs Y) ⊗Q X) (monadP-qbs Y)
    (λfx. fst fx (snd fx)) (strength-qbs (X ⇒Q monadP-qbs Y) X (f, s))) = s ≫= f
    using qbs-prob-join-qbs[OF qbs-prob.distr-qbs-prob[OF strength-qbs-prob[of f
      X ⇒Q monadP-qbs Y, OF h(1) h'(1)] strength-qbs[of f X ⇒Q monadP-qbs Y, OF
      h(1) h'(1)] qbs-morphism-eval] qbs-s-finite.distr-qbs[OF strength-qbs-s-finite[of f X
      ⇒Q monadP-qbs Y, OF h(1) h'(1)] strength-qbs[of f X ⇒Q monadP-qbs Y, OF h(1)
      h'(1)] qbs-morphism-eval] hb(2,3)] hb(1)
    by(simp add: bind-qbs[OF h'(1) qbs-morphism-monadPD[OF h(1)] hb(2)
      prob-kernel.s-finite-kernel-prob-kernel hb(1), simplified prob-kernel-def', OF hb(3)]
      comp-def bind-kernel-bind[of g μ borel, OF measurable-prob-algebraD])
  }
  thus ?thesis
  by(auto intro!: arg-swap-morphism[OF curry-preserves-morphisms [OF qbs-morphism-cong'[of
    - join-qbs ∘ (distr-qbs (exp-qbs X (monadP-qbs Y) ⊗Q X) (monadP-qbs Y) (λfx.
      (fst fx) (snd fx))) ∘ (strength-qbs (exp-qbs X (monadP-qbs Y)) X)]]] qbs-morphism-comp
    distr-qbs-morphismP' strength-qbs-morphismP join-qbs-morphismP qbs-morphism-eval
    simp: pair-qbs-space)
  qed

```

corollary *strength-qbs-law1P*:

```

  assumes x ∈ qbs-space (unit-quasi-borel ⊗Q monadP-qbs X)
  shows snd x = (distr-qbs (unit-quasi-borel ⊗Q X) X snd ∘ strength-qbs unit-quasi-borel
    X) x
  by(rule strength-qbs-law1, insert assms) (auto simp: pair-qbs-space qbs-space-monadPM)

```

corollary *strength-qbs-law2P*:

```

  assumes x ∈ qbs-space ((X ⊗Q Y) ⊗Q monadP-qbs Z)
  shows (strength-qbs X (Y ⊗Q Z) ∘ (map-prod id (strength-qbs Y Z)) ∘ (λ((x,y),z).
    (x,(y,z)))) x =
    (distr-qbs ((X ⊗Q Y) ⊗Q Z) (X ⊗Q (Y ⊗Q Z)) (λ((x,y),z). (x,(y,z))))
  ∘ strength-qbs (X ⊗Q Y) Z) x
  by(rule strength-qbs-law2, insert assms) (auto simp: pair-qbs-space qbs-space-monadPM)

```

lemma *strength-qbs-law4P*:

```

  assumes x ∈ qbs-space (X ⊗Q monadP-qbs (monadP-qbs Y))
  shows (strength-qbs X Y ∘ map-prod id join-qbs) x = (join-qbs ∘ distr-qbs (X
    ⊗Q monadP-qbs Y) (monadP-qbs (X ⊗Q Y)) (strength-qbs X Y) ∘ strength-qbs

```

X (*monadP-qbs* Y) x
 (is ?lhs = ?rhs)
proof –
 from *assms rep-qbs-space-monadP*[*of snd x monadP-qbs Y*] **obtain** $\beta \mu$
 where h :*qbs-prob* (*monadP-qbs Y*) $\beta \mu$ *snd x* = \llbracket *monadP-qbs Y, β, μ* \rrbracket_{sfin}
 by (*auto simp: pair-qbs-space*) *metis*
 then **interpret** qp : *qbs-prob monadP-qbs Y $\beta \mu$* **by** *simp*
 from *rep-qbs-Mx-monadP*[*OF qp.in-Mx*] **obtain** γg
 where h' : $\gamma \in$ *qbs-Mx Y g* \in *borel* \rightarrow_M *prob-algebra borel β* = (λr . $\llbracket Y, \gamma, g$
 $r \rrbracket_{sfin}$)
 and h'' : $\bigwedge r$. *qbs-prob Y γ (g r)*
 by *metis*
 have ?lhs = $\llbracket X \otimes_Q Y, \lambda r$. (*fst x, γ r*), $\mu \ggg g \rrbracket_{sfin}$
 using *qbs-s-finite.strength-qbs*[*OF qbs-prob.qbs-s-finite*[*OF qbs-prob-join-qbs-s-finite*[*OF*
 $h h'$]] - *qbs-prob-join-qbs*[*OF h h'*],*of fst x X*] *assms*
 by (*auto simp: pair-qbs-space*)
 also have ... = ?rhs
 using *qbs-prob-join-qbs*[*OF qbs-prob.distr-qbs-prob*[*OF qp.strength-qbs-prob*[*OF* -
 $h(2)$,*of fst x X*] *qp.strength-qbs*[*OF - h(2)*] *strength-qbs-morphismP*] *qbs-s-finite.distr-qbs*[*OF*
 $qp.strength-qbs-s-finite$ [*OF - h(2)*,*of fst x X*] *qp.strength-qbs*[*OF - h(2)*] *strength-qbs-morphismP*]
pair-qbs-MxI h'(2),*of λr . (fst x, γ r)*,*simplified comp-def qbs-s-finite.strength-qbs*[*OF*
 $qbs-prob.qbs-s-finite$ [*OF h'*] - *fun-cong*[*OF h'(3)*]]] *assms*
 by(*auto simp: pair-qbs-space h'*)
 finally **show** ?thesis .
qed

lemma *distr-qbs-morphismP*: *distr-qbs X Y* \in $X \Rightarrow_Q Y \rightarrow_Q$ *monadP-qbs X* \Rightarrow_Q *monadP-qbs Y*

proof –
 note [*qbs*] = *bind-qbs-morphismP return-qbs-morphismP*
 have [*simp*]: *distr-qbs X Y* = (λf sx . $sx \ggg$ *return-qbs Y* $\circ f$)
 by *standard+* (*auto simp: distr-qbs-def*)
 show ?thesis
 by *simp*
qed

lemma *bind-qbs-return-rotateP*:

assumes $p \in$ *qbs-space* (*monadP-qbs X*)
 and $q \in$ *qbs-space* (*monadP-qbs Y*)
 shows $q \ggg (\lambda y$. $p \ggg (\lambda x$. *return-qbs* ($X \otimes_Q Y$) (x, y))) = $p \ggg (\lambda x$. $q \ggg$
 $(\lambda y$. *return-qbs* ($X \otimes_Q Y$) (x, y)))
 by(*auto intro!: bind-qbs-return-rotate qbs-space-monadPM assms*)

lemma *qbs-bind-bind-return1P*:

assumes $f \in$ $X \otimes_Q Y \rightarrow_Q$ *monadP-qbs Z*
 $p \in$ *qbs-space* (*monadP-qbs X*)
 $q \in$ *qbs-space* (*monadP-qbs Y*)
 shows $q \ggg (\lambda y$. $p \ggg (\lambda x$. f (x, y))) = ($q \ggg (\lambda y$. $p \ggg (\lambda x$. *return-qbs* (X
 $\otimes_Q Y$) (x, y)))) $\ggg f$

by(*auto intro!*: *qbs-bind-bind-return1 assms qbs-space-monadPM qbs-morphism-monadPD*)

corollary *qbs-bind-bind-return1P'*:

assumes [*qbs*]: $f \in \text{qbs-space } (X \Rightarrow_Q Y \Rightarrow_Q \text{monadP-qbs } Z)$

$p \in \text{qbs-space } (\text{monadP-qbs } X)$

$q \in \text{qbs-space } (\text{monadP-qbs } Y)$

shows $q \gg (\lambda y. p \gg (\lambda x. f x y)) = (q \gg (\lambda y. p \gg (\lambda x. \text{return-qbs } (X \otimes_Q Y) (x,y)))) \gg (\text{case-prod } f)$

by(*auto intro!*: *qbs-bind-bind-return1P[where f=case-prod f and Z=Z,simplified]*)

lemma *qbs-bind-bind-return2P*:

assumes $f \in X \otimes_Q Y \rightarrow_Q \text{monadP-qbs } Z$

$p \in \text{qbs-space } (\text{monadP-qbs } X)$ $q \in \text{qbs-space } (\text{monadP-qbs } Y)$

shows $p \gg (\lambda x. q \gg (\lambda y. f (x,y))) = (p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y)))) \gg f$

by(*auto intro!*: *qbs-bind-bind-return2 assms qbs-space-monadPM qbs-morphism-monadPD*)

corollary *qbs-bind-bind-return2P'*:

assumes [*qbs*]: $f \in \text{qbs-space } (X \Rightarrow_Q Y \Rightarrow_Q \text{monadP-qbs } Z)$

$p \in \text{qbs-space } (\text{monadP-qbs } X)$

$q \in \text{qbs-space } (\text{monadP-qbs } Y)$

shows $p \gg (\lambda x. q \gg (\lambda y. f x y)) = (p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x,y)))) \gg (\text{case-prod } f)$

by(*auto intro!*: *qbs-bind-bind-return2P[where f=case-prod f and Z=Z,simplified]*)

corollary *bind-qbs-rotateP*:

assumes $f \in X \otimes_Q Y \rightarrow_Q \text{monadP-qbs } Z$

$p \in \text{qbs-space } (\text{monadP-qbs } X)$

and $q \in \text{qbs-space } (\text{monadP-qbs } Y)$

shows $q \gg (\lambda y. p \gg (\lambda x. f (x,y))) = p \gg (\lambda x. q \gg (\lambda y. f (x,y)))$

by(*auto intro!*: *bind-qbs-rotate assms qbs-space-monadPM qbs-morphism-monadPD*)

context *pair-qbs-probs*

begin

interpretation *rr* : *standard-borel-ne borel* \otimes_M *borel* :: (*real* \times *real*) *measure*

by(*auto intro!*: *pair-standard-borel-ne*)

sublocale *qbs-prob* $X \otimes_Q Y$ *map-prod* $\alpha \beta \circ \text{rr.from-real distr } (\mu \otimes_M \nu)$ *borel rr.to-real*

by(*auto simp: qbs-prob-def in-Mx-def real-distribution-def qbs-Mx-is-morphisms real-distribution-axioms-def pq1.prob-space-axioms pq2.prob-space-axioms intro!: prob-space.prob-space-distr prob-space-pair*)

lemma *qbs-bind-bind-return-prob*:

assumes [*qbs*]: $f \in X \otimes_Q Y \rightarrow_Q Z$

shows *qbs-prob* $Z (f \circ (\text{map-prod } \alpha \beta \circ \text{rr.from-real})) (\text{distr } (\mu \otimes_M \nu)$ *borel rr.to-real*)

using *qbs-prob-axioms by*(*auto simp: qbs-prob-def in-Mx-def qbs-Mx-is-morphisms*)

end

4.1.10 Almost Everywhere

lift-definition *qbs-almost-everywhere* :: [*'a qbs-measure, 'a \Rightarrow bool*] \Rightarrow *bool*
is $\lambda(X, \alpha, \mu). \text{almost-everywhere } (\text{distr } \mu \text{ (qbs-to-measure } X) \alpha)$
by(*auto simp: qbs-s-finite-eq-def*) *metis*

syntax

-qbs-almost-everywhere :: *pttrn \Rightarrow 'a \Rightarrow bool \Rightarrow bool* (*AE_Q - in - . - [0,0,10] 10*)

translations

AE_Q x in p. P \equiv CONST qbs-almost-everywhere p ($\lambda x. P$)

lemma *AEq-qbs-l*: (*AE_Q x in p. P x*) = (*AE x in qbs-l p. P x*)
by *transfer (simp add: case-prod-beta')*

lemma(**in** *qbs-s-finite*) *AEq-def*:

(*AE_Q x in $\llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} . P x$*) = (*AE x in (distr μ (qbs-to-measure X) α). P x*)
by(*simp add: qbs-almost-everywhere.abs-eq*)

lemma(**in** *qbs-s-finite*) *AEq-AE*: (*AE_Q x in $\llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} . P x$*) \Longrightarrow (*AE x in μ . P (αx)*)
by(*auto simp: AEq-def intro!: AE-distrD[of α]*)

lemma(**in** *qbs-s-finite*) *AEq-AE-iff*:

assumes [*qbs*]:*qbs-pred X P*

shows (*AE_Q x in $\llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} . P x$*) \longleftrightarrow (*AE x in μ . P (αx)*)

by(*auto simp: AEq-AE AEq-def qbs-pred-iff-sets intro!: AE-distr-iff[THEN iffD2]*)

lemma *AEq-qbs-pred[qbs]*: *qbs-almost-everywhere \in monadM-qbs X \rightarrow_Q (X \Rightarrow_Q qbs-count-space UNIV) \Rightarrow_Q qbs-count-space UNIV*

proof(*rule curry-preserves-morphisms[OF pair-qbs-morphismI]*)

fix $\gamma \beta$

assume *h*: $\gamma \in$ *qbs-Mx (monadM-qbs X)* $\beta \in$ *qbs-Mx (X \Rightarrow_Q qbs-count-space UNIV :: bool set)*)

from *rep-qbs-Mx-monadM[OF h(1)]* **obtain** αk **where** *hk*:

$\gamma = (\lambda r. \llbracket X, \alpha, k r \rrbracket_{s\text{fin}})$ $\alpha \in$ *qbs-Mx X s-finite-kernel borel borel k \wedge r. qbs-s-finite X α (k r)*

by *metis*

interpret *s*:*standard-borel-ne borel :: real measure* **by** *simp*

interpret *s2*: *standard-borel-ne borel \otimes_M borel :: (real \times real) measure* **by**(*simp add: borel-prod*)

have [*measurable*]:*Measurable.pred (borel \otimes_M borel) ($\lambda(x, y). \beta x (\alpha y)$)*

using *h(2) hk(2)* **by**(*simp add: s2.qbs-pred-iff-measurable-pred[symmetric] r-preserves-product qbs-Mx-is-morphisms*)

show ($\lambda r. \text{qbs-almost-everywhere } (\text{fst } (\gamma r, \beta r)) (\text{snd } (\gamma r, \beta r))) \in$ *qbs-Mx (qbs-count-space UNIV)*

using $h(2)$ $hk(2)$ **by** (*simp add: hk(1) qbs-Mx-is-morphisms qbs-s-finite.AEq-AE-iff*[*OF hk(4)*])
(auto simp add: s.qbs-pred-iff-measurable-pred intro!: s-finite-kernel.AE-pred[*OF hk(3)*])
qed

lemma *AEq-I2*[*simp*]:
assumes $p \in \text{qbs-space } (\text{monadM-qbs } X) \wedge x. x \in \text{qbs-space } X \implies P x$
shows $AE_Q x \text{ in } p. P x$
by (*auto simp: space-qbs-l-in*[*OF assms(1)*]) *assms(2)* *AEq-qbs-l*

lemma *AEq-mp*[*elim!*]:
assumes $AE_Q x \text{ in } s. P x$ $AE_Q x \text{ in } s. P x \longrightarrow Q x$
shows $AE_Q x \text{ in } s. Q x$
using *assms* **by** (*auto simp: AEq-qbs-l*)

lemma
shows *AEq-iffI*: $AE_Q x \text{ in } s. P x \implies AE_Q x \text{ in } s. P x \longleftrightarrow Q x \implies AE_Q x \text{ in } s. Q x$
and *AEq-disjI1*: $AE_Q x \text{ in } s. P x \implies AE_Q x \text{ in } s. P x \vee Q x$
and *AEq-disjI2*: $AE_Q x \text{ in } s. Q x \implies AE_Q x \text{ in } s. P x \vee Q x$
and *AEq-conjI*: $AE_Q x \text{ in } s. P x \implies AE_Q x \text{ in } s. Q x \implies AE_Q x \text{ in } s. P x \wedge Q x$
and *AEq-conj-iff*[*simp*]: $(AE_Q x \text{ in } s. P x \wedge Q x) \longleftrightarrow (AE_Q x \text{ in } s. P x) \wedge (AE_Q x \text{ in } s. Q x)$
by (*auto simp: AEq-qbs-l*)

lemma *AEq-symmetric*:
assumes $AE_Q x \text{ in } s. P x = Q x$
shows $AE_Q x \text{ in } s. Q x = P x$
using *assms* **by** (*auto simp: AEq-qbs-l*)

lemma *AEq-impI*: $(P \implies AE_Q x \text{ in } M. Q x) \implies AE_Q x \text{ in } M. P \longrightarrow Q x$
by (*auto simp: AEq-qbs-l AE-impI*)

lemma *AEq-Ball-mp*:
 $s \in \text{qbs-space } (\text{monadM-qbs } X) \implies (\wedge x. x \in \text{qbs-space } X \implies P x) \implies AE_Q x \text{ in } s. P x \longrightarrow Q x \implies AE_Q x \text{ in } s. Q x$
by *auto*

lemma *AEq-cong*:
 $s \in \text{qbs-space } (\text{monadM-qbs } X) \implies (\wedge x. x \in \text{qbs-space } X \implies P x \longleftrightarrow Q x) \implies (AE_Q x \text{ in } s. P x) \longleftrightarrow (AE_Q x \text{ in } s. Q x)$
by *auto*

lemma *AEq-cong-simp*: $s \in \text{qbs-space } (\text{monadM-qbs } X) \implies (\wedge x. x \in \text{qbs-space } X \implies P x = Q x) \implies (AE_Q x \text{ in } s. P x) \longleftrightarrow (AE_Q x \text{ in } s. Q x)$
by (*auto simp: simp-implies-def*)

lemma *AEq-all-countable*: $(AE_Q x \text{ in } s. \forall i. P i x) \longleftrightarrow (\forall i::'i::\text{countable}. AE_Q x \text{ in } s. P i x)$

by(*simp add: AEq-qbs-l AE-all-countable*)

lemma *AEq-ball-countable*: $\text{countable } X \implies (AE_Q x \text{ in } s. \forall y \in X. P x y) \longleftrightarrow (\forall y \in X. AE_Q x \text{ in } s. P x y)$

by(*simp add: AEq-qbs-l AE-ball-countable*)

lemma *AEq-ball-countable'*: $(\bigwedge N. N \in I \implies AE_Q x \text{ in } s. P N x) \implies \text{countable } I \implies AE_Q x \text{ in } s. \forall N \in I. P N x$

unfolding *AEq-ball-countable* **by** *simp*

lemma *AEq-pairwise*: $\text{countable } F \implies \text{pairwise } (\lambda A B. AE_Q x \text{ in } s. R x A B) F \longleftrightarrow (AE_Q x \text{ in } s. \text{pairwise } (R x) F)$

unfolding *pairwise-alt* **by** (*simp add: AEq-ball-countable*)

lemma *AEq-finite-all*: $\text{finite } S \implies (AE_Q x \text{ in } s. \forall i \in S. P i x) \longleftrightarrow (\forall i \in S. AE_Q x \text{ in } s. P i x)$

by(*simp add: AEq-qbs-l AE-finite-all*)

lemma *AE-finite-allI*: $\text{finite } S \implies (\bigwedge s. s \in S \implies AE_Q x \text{ in } M. Q s x) \implies AE_Q x \text{ in } M. \forall s \in S. Q s x$

by(*simp add: AEq-qbs-l AE-finite-all*)

4.1.11 Integral

lift-definition *qbs-nn-integral* :: $['a \text{ qbs-measure}, 'a \Rightarrow \text{ennreal}] \Rightarrow \text{ennreal}$

is $\lambda(X, \alpha, \mu) f. (\int^+ x. f x \partial \text{distr } \mu (\text{qbs-to-measure } X) \alpha)$

by(*auto simp: qbs-s-finite-eq-def*)

lift-definition *qbs-integral* :: $['a \text{ qbs-measure}, 'a \Rightarrow ('b :: \{\text{banach}, \text{second-countable-topology}\})] \Rightarrow 'b$

is $\lambda p f. \text{if } f \in (\text{fst } p) \rightarrow_Q \text{qbs-borel} \text{ then } (\int x. f (\text{fst } (\text{snd } p) x) \partial (\text{snd } (\text{snd } p))) \text{ else } 0$

using *qbs-s-finite-eq-dest(3) qbs-s-finite-eq-1-imp-2* **by** *fastforce*

syntax

-qbs-nn-integral :: $\text{pttrn} \Rightarrow \text{ennreal} \Rightarrow 'a \text{ qbs-measure} \Rightarrow \text{ennreal} (\int^+_Q ((2 \text{ -./ -}) / \partial \text{-}) [60,61] 110)$

translations

$\int^+_Q x. f \partial p \equiv \text{CONST } \text{qbs-nn-integral } p (\lambda x. f)$

syntax

-qbs-integral :: $\text{pttrn} \Rightarrow \text{-} \Rightarrow 'a \text{ qbs-measure} \Rightarrow \text{-} (\int_Q ((2 \text{ -./ -}) / \partial \text{-}) [60,61] 110)$

translations

$\int_Q x. f \partial p \equiv \text{CONST } \text{qbs-integral } p (\lambda x. f)$

lemma(in *qbs-s-finite*)
shows *qbs-nn-integral-def*: $f \in X \rightarrow_Q \text{qbs-borel} \implies (\int^+_Q x. f x \partial \llbracket X, \alpha, \mu \rrbracket_{sfin})$
 $= (\int^+ x. f (\alpha x) \partial \mu)$
and *qbs-nn-integral-def2*: $(\int^+_Q x. f x \partial \llbracket X, \alpha, \mu \rrbracket_{sfin}) = (\int^+ x. f x \partial (\text{distr } \mu$
 $(\text{qbs-to-measure } X) \alpha))$
by(*simp-all add: qbs-nn-integral.abs-eq nn-integral-distr lr-adjunction-correspondence*)

lemma(in *qbs-s-finite*) *qbs-integral-def*:
 $f \in X \rightarrow_Q \text{qbs-borel} \implies (\int_Q x. f x \partial \llbracket X, \alpha, \mu \rrbracket_{sfin}) = (\int x. f (\alpha x) \partial \mu)$
by(*simp add: qbs-integral.abs-eq*)

lemma(in *qbs-s-finite*) *qbs-integral-def2*: $(\int_Q x. f x \partial \llbracket X, \alpha, \mu \rrbracket_{sfin}) = (\int x. f x$
 $\partial (\text{distr } \mu (\text{qbs-to-measure } X) \alpha))$
proof –
consider $f \in X \rightarrow_Q \text{qbs-borel} \mid f \notin X \rightarrow_Q \text{qbs-borel}$ **by** *auto*
thus *?thesis*
proof *cases*
case *h:2*
then **have** $\neg \text{integrable } (\text{qbs-l } \llbracket X, \alpha, \mu \rrbracket_{sfin}) f$
by (*metis borel-measurable-integrable measurable-distr-eq1 qbs-l qbs-morphism-measurable-intro*)
thus *?thesis*
using *h* **by**(*simp add: qbs-l qbs-integral.abs-eq lr-adjunction-correspondence*
not-integrable-integral-eq)
qed(*simp add: qbs-integral.abs-eq lr-adjunction-correspondence integral-distr*)
qed

lemma *qbs-measure-eq1*:
assumes [*qbs*]: $p \in \text{qbs-space } (\text{monadM-qbs } X) \ q \in \text{qbs-space } (\text{monadM-qbs } X)$
and $\bigwedge f. f \in X \rightarrow_Q \text{qbs-borel} \implies (\int^+_Q x. f x \partial p) = (\int^+_Q x. f x \partial q)$
shows $p = q$
proof –
obtain $\alpha \ \mu \ \beta \ \nu$ **where** $h: p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \ q = \llbracket X, \beta, \nu \rrbracket_{sfin}$ *qbs-s-finite* $X \ \alpha$
 $\mu \ \text{qbs-s-finite } X \ \beta \ \nu$
by (*metis rep-qbs-space-monadM assms(1,2)*)
then **interpret** $pq: \text{pair-qbs-s-finite } X \ \alpha \ \mu \ \beta \ \nu$
by(*auto simp: pair-qbs-s-finite-def*)
show *?thesis*
using *assms(3)* **by**(*auto simp: h(1,2) pq.pq1.qbs-nn-integral-def pq.pq2.qbs-nn-integral-def*
intro!: pq.qbs-s-finite-measure-eq')
qed

lemma *qbs-nn-integral-def2-l*: $\text{qbs-nn-integral } s \ f = \text{integral}^N (\text{qbs-l } s) \ f$
by *transfer auto*

lemma *qbs-integral-def2-l*: $\text{qbs-integral } s \ f = \text{integral}^L (\text{qbs-l } s) \ f$
by (*metis in-qbs-space-of qbs-s-finite.qbs-integral-def2 qbs-s-finite.qbs-l rep-qbs-space-monadM*)

lift-definition *qbs-integrable* :: '*a* *qbs-measure* \implies ('*a* \implies '*b*::{*second-countable-topology, banach*})
 $\implies \text{bool}$

```

is  $\lambda p f. f \in \text{fst } p \rightarrow_Q \text{qbs-borel} \wedge \text{integrable } (\text{snd } (\text{snd } p)) (f \circ (\text{fst } (\text{snd } p)))$ 
proof safe
  have  $0: f \in Y \rightarrow_Q \text{qbs-borel integrable } \nu (\lambda x. f (\beta x))$  if qbs-s-finite-eq  $(X, \alpha, \mu)$ 
 $(Y, \beta, \nu) f \in X \rightarrow_Q \text{qbs-borel integrable } \mu (\lambda x. f (\alpha x))$  for  $X :: 'a \text{ quasi-borel}$  and
 $Y \alpha \beta \mu \nu$  and  $f :: - \Rightarrow 'b$ 
  proof -
    interpret pair-qbs-s-finite  $X \alpha \mu \beta \nu$ 
    using qbs-s-finite-eq-dest[OF that(1)] by (auto simp: pair-qbs-s-finite-def)
    show  $f \in Y \rightarrow_Q \text{qbs-borel integrable } \nu (\lambda x. f (\beta x))$ 
    using that qbs-s-finite-eq-dest( $\beta$ )[OF that(1)] by (simp-all add: integrable-distr-eq[symmetric, of
 $\alpha \mu \text{qbs-to-measure } X f]$  integrable-distr-eq[symmetric, of  $\beta \nu \text{qbs-to-measure } X f]$ 
lr-adjunction-correspondence qbs-s-finite-eq-dest( $\beta$ )[OF that(1)])
  qed
  {
    fix  $X Y :: 'a \text{ quasi-borel}$ 
    fix  $\alpha \beta \mu \nu$  and  $f :: - \Rightarrow 'b$ 
    assume  $1: \text{qbs-s-finite-eq } (X, \alpha, \mu) (Y, \beta, \nu)$ 
    then have  $2: \text{qbs-s-finite-eq } (Y, \beta, \nu) (X, \alpha, \mu)$  by (auto simp: qbs-s-finite-eq-def)
    have  $f \in X \rightarrow_Q \text{qbs-borel} \wedge \text{integrable } \mu (f \circ \alpha) \iff f \in Y \rightarrow_Q \text{qbs-borel} \wedge$ 
integrable  $\nu (f \circ \beta)$ 
    unfolding comp-def using  $0[\text{OF } 1, \text{of } f]$   $0[\text{OF } 2, \text{of } f]$  by blast
  }
  thus  $\bigwedge \text{prod1 prod2} :: 'a \text{ qbs-s-finite-t. qbs-s-finite-eq prod1 prod2} \implies (\lambda f. - \Rightarrow$ 
 $'b. f \in \text{fst prod1} \rightarrow_Q \text{borel}_Q \wedge \text{integrable } (\text{snd } (\text{snd prod1})) (f \circ \text{fst } (\text{snd prod1}))) =$ 
 $(\lambda f. f \in \text{fst prod2} \rightarrow_Q \text{borel}_Q \wedge \text{integrable } (\text{snd } (\text{snd prod2})) (f \circ \text{fst } (\text{snd prod2})))$ 
by fastforce
qed

lemma(in qbs-s-finite) qbs-integrable-def:
 $\text{qbs-integrable } \llbracket X, \alpha, \mu \rrbracket_{\text{sf in}} f \iff f \in X \rightarrow_Q \text{qbs-borel} \wedge \text{integrable } \mu (\lambda x. f (\alpha$ 
 $x))$ 
by (simp add: qbs-integrable.abs-eq comp-def)

lemma qbs-integrable-morphism-dest:
assumes  $s \in \text{qbs-space } (\text{monadM-qbs } X)$ 
and qbs-integrable  $s f$ 
shows  $f \in X \rightarrow_Q \text{qbs-borel}$ 
by (metis assms qbs-s-finite.qbs-integrable-def rep-qbs-space-monadM)

lemma qbs-integrable-morphismP:
assumes  $s \in \text{qbs-space } (\text{monadP-qbs } X)$ 
and qbs-integrable  $s f$ 
shows  $f \in X \rightarrow_Q \text{qbs-borel}$ 
by (auto intro!: qbs-integrable-morphism-dest assms qbs-space-monadPM)

lemma(in qbs-s-finite) qbs-integrable-measurable[simp]:
assumes  $\text{qbs-integrable } \llbracket X, \alpha, \mu \rrbracket_{\text{sf in}} f$ 
shows  $f \in \text{qbs-to-measure } X \rightarrow_M \text{borel}$ 
by (auto intro!: qbs-integrable-morphism-dest assms simp: lr-adjunction-correspondence[symmetric])

```

lemma *qbs-integrable-iff-integrable*:
 (*qbs-integrable* (*s*::'a *qbs-measure*) (*f*::'a \Rightarrow 'b::{*second-countable-topology*,*banach*}))
 = (*integrable* (*qbs-l s*) *f*)
proof *transfer*
fix *f*::'a \Rightarrow 'b::{*second-countable-topology*,*banach*}
show *qbs-s-finite-eq s s* \Longrightarrow (*f* \in *fst s* \rightarrow_Q *borel_Q* \wedge *integrable* (*snd* (*snd s*)) (*f*
 \circ *fst* (*snd s*))) = *integrable* (*distr* (*snd* (*snd s*)) (*qbs-to-measure* (*fst s*)) (*fst* (*snd*
s))) *f for s*
proof(*rule prod-cases3*[*of s*])
fix *X*::'a *quasi-borel*
fix α μ
assume *qbs-s-finite-eq s s* **and** *s*: *s* = (*X*, α , μ)
then interpret *qbs-s-finite X* α μ **by**(*simp add: qbs-s-finite-eq-def*)
show *f* \in *fst s* \rightarrow_Q *qbs-borel* \wedge *integrable* (*snd* (*snd s*)) ($\lambda x.$ (*f* \circ *fst* (*snd s*))
x) \longleftrightarrow *integrable* (*distr* (*snd* (*snd s*)) (*qbs-to-measure* (*fst s*)) (*fst* (*snd s*))) *f*
using *integrable-distr-eq*[*of* α μ *qbs-to-measure X f*,*simplified*]
by(*auto simp add: lr-adjunction-correspondence s*)
qed
qed

corollary(**in** *qbs-s-finite*) *qbs-integrable-distr*: *qbs-integrable* $\llbracket X, \alpha, \mu \rrbracket_{sfin}$ *f* = *in-*
tegrable (*distr* μ (*qbs-to-measure X*) α) *f*
by(*simp add: qbs-integrable-iff-integrable qbs-l*)

lemma *qbs-integrable-morphism*[*qbs*]: *qbs-integrable* \in *monadM-qbs X* \rightarrow_Q (*X* \Rightarrow_Q
(*qbs-borel* :: ('a :: {*banach*, *second-countable-topology*}) *quasi-borel*)) \Rightarrow_Q *qbs-count-space*
UNIV

proof(*rule curry-preserves-morphisms*[*OF pair-qbs-morphismI*])
fix γ β
assume *h*: $\gamma \in$ *qbs-Mx* (*monadM-qbs X*) $\beta \in$ *qbs-Mx* (*X* \Rightarrow_Q (*qbs-borel* :: 'a
quasi-borel))
from *rep-qbs-Mx-monadM*[*OF this*(1)] **obtain** α *k*
where *hk*: $\gamma = (\lambda r.$ $\llbracket X, \alpha, k \rrbracket_{sfin}$) $\alpha \in$ *qbs-Mx X s-finite-kernel borel borel k*
 $\wedge r.$ *qbs-s-finite X* α (*k r*)
by *metis*
then interpret *ina*: *in-Mx X* α **by** (*simp add: in-Mx-def*)
interpret *standard-borel-ne borel* :: *real measure* **by** *simp*
have [*measurable*]: β *r* \in *qbs-to-measure X* \rightarrow_M *borel for r*
using *h*(2) **by**(*simp add: qbs-Mx-is-morphisms lr-adjunction-correspondence[symmetric]*)
have [*measurable-cong*]: *sets* (*k r*) = *sets borel for r*
using *hk*(4) *qbs-s-finite.mu-sets* **by** *blast*
have 1: *borel-measurable* (*borel* \otimes_M *borel*) = (*qbs-borel* \otimes_Q *qbs-borel* \rightarrow_Q
qbs-borel :: (*real* \times *real* \Rightarrow 'a) *set*)
by (*metis borel-prod pair-standard-borel qbs-borel-prod standard-borel.standard-borel-r-full-faithful*
standard-borel-axioms)
show ($\lambda r.$ *qbs-integrable* (*fst* (γ *r*, β *r*)) (*snd* (γ *r*, β *r*))) \in *qbs-Mx* (*qbs-count-space*
UNIV)
by(*auto simp: fun-cong*[*OF hk*(1)] *qbs-s-finite.qbs-integrable-distr*[*OF hk*(4)] *in-*

tegrable-distr-eq qbs-Mx-is-morphisms qbs-pred-iff-measurable-pred intro!: s-finite-kernel.integrable-measurable-hk(3)] (insert h(2), simp add: 1 qbs-Mx-is-morphisms split-beta')

qed

lemma(in *qbs-s-finite*) *qbs-integrable-iff-integrable*:

assumes $f \in \text{qbs-to-measure } X \rightarrow_M \text{ borel}$

shows $\text{qbs-integrable } \llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} f = \text{integrable } \mu (\lambda x. f (\alpha x))$

by(*auto intro!: integrable-distr-eq[of $\alpha \mu$ qbs-to-measure X f] simp: assms qbs-integrable-distr*)

lemma *qbs-integrable-iff-bounded*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$

shows $\text{qbs-integrable } s f \iff f \in X \rightarrow_Q \text{qbs-borel} \wedge (\int^+_Q x. \text{ennreal } (\text{norm } (f x)) \partial s) < \infty$

(**is** *?lhs = ?rhs*)

proof –

from *rep-qbs-space-monadM[OF assms]* **obtain** $\alpha \mu$ **where** *hs*:

qbs-s-finite $X \alpha \mu s = \llbracket X, \alpha, \mu \rrbracket_{s\text{fin}}$

by *metis*

then interpret *qs:qbs-s-finite* $X \alpha \mu$ **by** *simp*

have *?lhs = integrable (distr μ (qbs-to-measure X) α) f*

by (*simp add: hs(2) qbs-integrable-iff-integrable qs.qbs-l*)

also have $\dots = (f \in \text{borel-measurable } (\text{distr } \mu (\text{qbs-to-measure } X) \alpha) \wedge ((\int^+ x. \text{ennreal } (\text{norm } (f x)) \partial (\text{distr } \mu (\text{qbs-to-measure } X) \alpha)) < \infty))$

by(*rule integrable-iff-bounded*)

also have $\dots = \text{?rhs}$

by(*auto simp add: hs(2) qs.qbs-nn-integral-def2 lr-adjunction-correspondence*)

finally show *?thesis* .

qed

lemma *not-qbs-integrable-qbs-integral*: $\neg \text{qbs-integrable } s f \implies \text{qbs-integral } s f = 0$

by(*simp add: qbs-integral-def2-l qbs-integrable-iff-integrable not-integrable-integral-eq*)

lemma *qbs-integrable-cong-AE*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$

$AE_Q x \text{ in } s. f x = g x$

and $\text{qbs-integrable } s f g \in X \rightarrow_Q \text{qbs-borel}$

shows $\text{qbs-integrable } s g$

using *assms(2,4)* **by**(*auto intro!: qbs-integrable-iff-integrable[THEN iffD2] Bochner-Integration.integrable-cong-g-f, THEN iffD2] qbs-integrable-iff-integrable[THEN iffD1, OF assms(3)] qbs-integrable-morphism-dest[OF assms(1), of f] simp: AEq-qbs-l measurable-qbs-l[OF assms(1)]*)

lemma *qbs-integrable-cong*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$

$\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$

and $\text{qbs-integrable } s f$

shows $\text{qbs-integrable } s g$

by(*auto intro!: qbs-integrable-iff-integrable[THEN iffD2] Bochner-Integration.integrable-cong[OF refl, of - $g f$, THEN iffD2] qbs-integrable-iff-integrable[THEN iffD1, OF assms(3)]*)

simp: *space-qbs-l-in*[*OF assms*(1)] *assms*(2))

lemma *qbs-integrable-zero*[*simp, intro*]: *qbs-integrable* *s* ($\lambda x. 0$)
by(*simp add*: *qbs-integrable-iff-integrable*)

lemma *qbs-integrable-const*:

assumes *s* \in *qbs-space* (*monadP-qbs X*)
shows *qbs-integrable* *s* ($\lambda x. c$)
using *assms* **by**(*auto intro!*: *qbs-integrable-iff-integrable*[*THEN iffD2*] *finite-measure.integrable-const*
simp: *monadP-qbs-space prob-space-def*)

lemma *qbs-integrable-add*[*simp, intro!*]:

assumes *qbs-integrable* *s f*
and *qbs-integrable* *s g*
shows *qbs-integrable* *s* ($\lambda x. f x + g x$)
by(*rule qbs-integrable-iff-integrable*[*THEN iffD2, OF Bochner-Integration.integrable-add*][*OF qbs-integrable-iff-integrable*[*THEN iffD1, OF assms*(1)]] *qbs-integrable-iff-integrable*[*THEN iffD1, OF assms*(2)]])

lemma *qbs-integrable-diff*[*simp, intro!*]:

assumes *qbs-integrable* *s f*
and *qbs-integrable* *s g*
shows *qbs-integrable* *s* ($\lambda x. f x - g x$)
by(*rule qbs-integrable-iff-integrable*[*THEN iffD2, OF Bochner-Integration.integrable-diff*][*OF qbs-integrable-iff-integrable*[*THEN iffD1, OF assms*(1)]] *qbs-integrable-iff-integrable*[*THEN iffD1, OF assms*(2)]])

lemma *qbs-integrable-sum*[*simp, intro!*]: ($\bigwedge i. i \in I \implies \text{qbs-integrable } s (f i)$) \implies
qbs-integrable *s* ($\lambda x. \sum_{i \in I}. f i x$)
by(*simp add*: *qbs-integrable-iff-integrable*)

lemma *qbs-integrable-scaleR-left*[*simp, intro!*]: *qbs-integrable* *s f* \implies *qbs-integrable*
s ($\lambda x. f x *_{\mathbb{R}} (c :: 'a :: \{\text{second-countable-topology, banach}\})$)
by(*simp add*: *qbs-integrable-iff-integrable*)

lemma *qbs-integrable-scaleR-right*[*simp, intro!*]: *qbs-integrable* *s f* \implies *qbs-integrable*
s ($\lambda x. c *_{\mathbb{R}} (f x :: 'a :: \{\text{second-countable-topology, banach}\})$)
by(*simp add*: *qbs-integrable-iff-integrable*)

lemma *qbs-integrable-mult-iff*:

fixes *f* :: '*a* \Rightarrow *real*
shows (*qbs-integrable* *s* ($\lambda x. c * f x$)) = (*c* = 0 \vee *qbs-integrable* *s f*)
using *qbs-integrable-iff-integrable*[*of s* $\lambda x. c * f x$] *integrable-mult-left-iff*[*of - c f*]
qbs-integrable-iff-integrable[*of s f*]
by *simp*

lemma

fixes *c* :: -::*{real-normed-algebra, second-countable-topology}*
assumes *qbs-integrable* *s f*

```

shows qbs-integrable-mult-right: qbs-integrable  $s$  ( $\lambda x. c * f x$ )
  and qbs-integrable-mult-left: qbs-integrable  $s$  ( $\lambda x. f x * c$ )
using assms by(auto simp: qbs-integrable-iff-integrable)

lemma qbs-integrable-divide-zero[simp, intro!]:
  fixes  $c :: \text{-::}\{\text{real-normed-field, field, second-countable-topology}\}$ 
  shows qbs-integrable  $s f \implies$  qbs-integrable  $s$  ( $\lambda x. f x / c$ )
  by(simp add: qbs-integrable-iff-integrable)

lemma qbs-integrable-inner-left[simp, intro!]:
  qbs-integrable  $s f \implies$  qbs-integrable  $s$  ( $\lambda x. f x \cdot c$ )
  by(simp add: qbs-integrable-iff-integrable)

lemma qbs-integrable-inner-right[simp, intro!]:
  qbs-integrable  $s f \implies$  qbs-integrable  $s$  ( $\lambda x. c \cdot f x$ )
  by(simp add: qbs-integrable-iff-integrable)

lemma qbs-integrable-minus[simp, intro!]:
  qbs-integrable  $s f \implies$  qbs-integrable  $s$  ( $\lambda x. - f x$ )
  by(simp add: qbs-integrable-iff-integrable)

lemma [simp, intro!]:
  assumes qbs-integrable  $s f$ 
  shows qbs-integrable-Re: qbs-integrable  $s$  ( $\lambda x. \text{Re } (f x)$ )
    and qbs-integrable-Im: qbs-integrable  $s$  ( $\lambda x. \text{Im } (f x)$ )
    and qbs-integrable-cnj: qbs-integrable  $s$  ( $\lambda x. \text{cnj } (f x)$ )
  using assms by(simp-all add: qbs-integrable-iff-integrable)

lemma qbs-integrable-of-real[simp, intro!]:
  qbs-integrable  $s f \implies$  qbs-integrable  $s$  ( $\lambda x. \text{of-real } (f x)$ )
  by(simp-all add: qbs-integrable-iff-integrable)

lemma [simp, intro!]:
  assumes qbs-integrable  $s f$ 
  shows qbs-integrable-fst: qbs-integrable  $s$  ( $\lambda x. \text{fst } (f x)$ )
    and qbs-integrable-snd: qbs-integrable  $s$  ( $\lambda x. \text{snd } (f x)$ )
  using assms by(simp-all add: qbs-integrable-iff-integrable)

lemma qbs-integrable-norm:
  assumes qbs-integrable  $s f$ 
  shows qbs-integrable  $s$  ( $\lambda x. \text{norm } (f x)$ )
  by(rule qbs-integrable-iff-integrable[THEN iffD2, OF integrable-norm[OF qbs-integrable-iff-integrable[THEN iffD1, OF assms]]])

lemma qbs-integrable-abs:
  fixes  $f :: \text{-} \Rightarrow \text{real}$ 
  assumes qbs-integrable  $s f$ 
  shows qbs-integrable  $s$  ( $\lambda x. |f x|$ )
  by(rule qbs-integrable-iff-integrable[THEN iffD2, OF integrable-abs[OF qbs-integrable-iff-integrable[THEN

```

iffD1, OF assms]]])

lemma *qbs-integrable-sq*:

fixes $c :: \text{-}::\{\text{real-normed-field, second-countable-topology}\}$
assumes *qbs-integrable* s $(\lambda x. c)$ *qbs-integrable* s f
and *qbs-integrable* s $(\lambda x. (f x)^2)$
shows *qbs-integrable* s $(\lambda x. (f x - c)^2)$
by(*simp add: comm-ring-1-class.power2-diff, rule qbs-integrable-diff, rule qbs-integrable-add*)
(*simp-all add: comm-semiring-1-class.semiring-normalization-rules(16)[of 2]*)
assms qbs-integrable-mult-right power2-eq-square[of c])

lemma *qbs-nn-integral-eq-integral-AEq*:

assumes *qbs-integrable* s f AE_Q x *in* s . $0 \leq f x$
shows $(\int^+_Q x. \text{ennreal } (f x) \partial s) = \text{ennreal } (\int_Q x. f x \partial s)$
using *nn-integral-eq-integral*[*OF qbs-integrable-iff-integrable*[*THEN iffD1, OF assms(1)*]]
] *qbs-integrable-morphism-dest*[*OF in-qbs-space-of assms(1)*] *assms(2)*
by(*simp add: qbs-integral-def2-l qbs-nn-integral-def2-l AEq-qbs-l*)

lemma *qbs-nn-integral-eq-integral*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$ *qbs-integrable* s f
and $\bigwedge x. x \in \text{qbs-space } X \implies 0 \leq f x$
shows $(\int^+_Q x. \text{ennreal } (f x) \partial s) = \text{ennreal } (\int_Q x. f x \partial s)$
using *qbs-nn-integral-eq-integral-AEq*[*OF assms(2) AEq-I2*[*OF assms(1,3)*]] **by**
simp

lemma *qbs-nn-integral-cong-AEq*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$ AE_Q x *in* s . $f x = g x$
shows *qbs-nn-integral* s $f = \text{qbs-nn-integral } s$ g
proof –
from *rep-qbs-space-monadM*[*OF assms(1)*] **obtain** $\alpha \mu$
where $hs: s = \llbracket X, \alpha, \mu \rrbracket_{s \text{ fin}}$ *qbs-s-finite* X $\alpha \mu$ **by** *metis*
then interpret $qs: \text{qbs-s-finite } X$ $\alpha \mu$ **by** *simp*
show *?thesis*
using *assms(2)* **by**(*auto simp: qs.qbs-nn-integral-def2 hs(1) qs.AEq-def intro!*:
nn-integral-cong-AE)
qed

lemma *qbs-nn-integral-cong*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$ $\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$
shows *qbs-nn-integral* s $f = \text{qbs-nn-integral } s$ g
using *qbs-nn-integral-cong-AEq*[*OF assms(1) AEq-I2*[*OF assms*]] **by** *simp*

lemma *qbs-nn-integral-const*:

$(\int^+_Q x. c \partial s) = c * \text{qbs-l } s$ (*qbs-space (qbs-space-of s)*)
by(*simp add: qbs-nn-integral-def2-l space-qbs-l*)

lemma *qbs-nn-integral-const-prob*:

assumes $s \in \text{qbs-space } (\text{monadP-qbs } X)$
shows $(\int^+_Q x. c \partial s) = c$

using *assms* **by**(*simp add: qbs-nn-integral-const prob-space.emeasure-space-1 qbs-l-prob-space space-qbs-l*)

lemma *qbs-nn-integral-add*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
and $[qbs]: f \in X \rightarrow_Q \text{qbs-borel } g \in X \rightarrow_Q \text{qbs-borel}$
shows $(\int^+_Q x. f x + g x \partial s) = (\int^+_Q x. f x \partial s) + (\int^+_Q x. g x \partial s)$
by(*auto simp: qbs-nn-integral-def2-l measurable-qbs-l[OF assms(1)] intro!: nn-integral-add measurable-qbs-l*)

lemma *qbs-nn-integral-cmult*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$ **and** $[qbs]: f \in X \rightarrow_Q \text{qbs-borel}$
shows $(\int^+_Q x. c * f x \partial s) = c * (\int^+_Q x. f x \partial s)$
by(*auto simp: qbs-nn-integral-def2-l measurable-qbs-l[OF assms(1)] intro!: nn-integral-cmult*)

lemma *qbs-integral-cong-AEq*:

assumes $[qbs]: s \in \text{qbs-space } (\text{monadM-qbs } X)$ $f \in X \rightarrow_Q \text{qbs-borel } g \in X \rightarrow_Q \text{qbs-borel}$
and $AE_Q x \text{ in } s. f x = g x$
shows $\text{qbs-integral } s f = \text{qbs-integral } s g$
using *assms(4)* **by**(*auto simp: qbs-integral-def2-l AEq-qbs-l measurable-qbs-l[OF assms(1)] intro!: integral-cong-AE*)

lemma *qbs-integral-cong*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X) \wedge x. x \in \text{qbs-space } X \implies f x = g x$
shows $\text{qbs-integral } s f = \text{qbs-integral } s g$
by(*auto simp: qbs-integral-def2-l space-qbs-l-in[OF assms(1)] assms(2) intro!: Bochner-Integration.integral-cong*)

lemma *qbs-integral-nonneg-AEq*:

fixes $f :: - \Rightarrow \text{real}$
shows $AE_Q x \text{ in } s. 0 \leq f x \implies 0 \leq \text{qbs-integral } s f$
by(*auto simp: qbs-integral-def2-l AEq-qbs-l intro!: integral-nonneg-AE*)

lemma *qbs-integral-nonneg*:

fixes $f :: - \Rightarrow \text{real}$
assumes $s \in \text{qbs-space } (\text{monadM-qbs } X) \wedge x. x \in \text{qbs-space } X \implies 0 \leq f x$
shows $0 \leq \text{qbs-integral } s f$
by(*auto simp: qbs-integral-def2-l space-qbs-l-in[OF assms(1)] assms(2) intro!: Bochner-Integration.integral-nonneg*)

lemma *qbs-integral-mono-AEq*:

fixes $f :: - \Rightarrow \text{real}$
assumes $\text{qbs-integrable } s f \text{ qbs-integrable } s g \text{ } AE_Q x \text{ in } s. f x \leq g x$
shows $\text{qbs-integral } s f \leq \text{qbs-integral } s g$
using *assms* **by**(*auto simp: qbs-integral-def2-l AEq-qbs-l qbs-integrable-iff-integrable intro!: integral-mono-AE*)

lemma *qbs-integral-mono*:

fixes $f :: - \Rightarrow \text{real}$
assumes $s \in \text{qbs-space } (\text{monadM-qbs } X)$
and $\text{qbs-integrable } s \text{ } f \text{ } \text{qbs-integrable } s \text{ } g \wedge x. x \in \text{qbs-space } X \implies f \ x \leq g \ x$
shows $\text{qbs-integral } s \ f \leq \text{qbs-integral } s \ g$
by (*auto simp*: $\text{qbs-integral-def2-l space-qbs-l-in}[\text{OF } \text{assms}(1)] \text{qbs-integrable-iff-integrable}[\text{symmetric}]$
assms intro!: integral-mono)

lemma $\text{qbs-integral-const-prob}$:
assumes $s \in \text{qbs-space } (\text{monadP-qbs } X)$
shows $(\int_Q x. c \ \partial s) = c$
using *assms* **by** (*simp add*: $\text{qbs-integral-def2-l monadP-qbs-space prob-space.prob-space}$)

lemma
assumes $\text{qbs-integrable } s \text{ } f \text{ } \text{qbs-integrable } s \text{ } g$
shows $\text{qbs-integral-add}:(\int_Q x. f \ x + g \ x \ \partial s) = (\int_Q x. f \ x \ \partial s) + (\int_Q x. g \ x \ \partial s)$
and $\text{qbs-integral-diff}:(\int_Q x. f \ x - g \ x \ \partial s) = (\int_Q x. f \ x \ \partial s) - (\int_Q x. g \ x \ \partial s)$
using *assms* **by** (*auto simp*: $\text{qbs-integral-def2-l qbs-integrable-iff-integrable}[\text{symmetric}]$
intro!: $\text{Bochner-Integration.integral-add Bochner-Integration.integral-diff}$)

lemma [*simp*]:
fixes $c :: -::\{\text{real-normed-field,second-countable-topology}\}$
shows $\text{qbs-integral-mult-right-zero}:(\int_Q x. c * f \ x \ \partial s) = c * (\int_Q x. f \ x \ \partial s)$
and $\text{qbs-integral-mult-left-zero}:(\int_Q x. f \ x * c \ \partial s) = (\int_Q x. f \ x \ \partial s) * c$
and $\text{qbs-integral-divide-zero}:(\int_Q x. f \ x / c \ \partial s) = (\int_Q x. f \ x \ \partial s) / c$
by (*auto simp*: $\text{qbs-integral-def2-l}$)

lemma $\text{qbs-integral-minus}[\text{simp}]$: $(\int_Q x. - f \ x \ \partial s) = - (\int_Q x. f \ x \ \partial s)$
by (*auto simp*: $\text{qbs-integral-def2-l}$)

lemma [*simp*]:
shows $\text{qbs-integral-scaleR-right}:(\int_Q x. c *_{\text{R}} f \ x \ \partial s) = c *_{\text{R}} (\int_Q x. f \ x \ \partial s)$
and $\text{qbs-integral-scaleR-left}:(\int_Q x. f \ x *_{\text{R}} c \ \partial s) = (\int_Q x. f \ x \ \partial s) *_{\text{R}} c$
by (*auto simp*: $\text{qbs-integral-def2-l}$)

lemma [*simp*]:
shows $\text{qbs-integral-inner-left}:\text{qbs-integrable } s \ f \implies (\int_Q x. f \ x \cdot c \ \partial s) = (\int_Q x. f \ x \ \partial s) \cdot c$
and $\text{qbs-integral-inner-right}:\text{qbs-integrable } s \ f \implies (\int_Q x. c \cdot f \ x \ \partial s) = c \cdot (\int_Q x. f \ x \ \partial s)$
by (*auto simp*: $\text{qbs-integral-def2-l qbs-integrable-iff-integrable}$)

lemma $\text{integral-complex-of-real}[\text{simp}]$: $(\int_Q x. \text{complex-of-real } (f \ x) \ \partial s) = \text{of-real } (\int_Q x. f \ x \ \partial s)$
by (*simp add*: $\text{qbs-integral-def2-l}$)

lemma $\text{integral-cnj}[\text{simp}]$: $(\int_Q x. \text{cnj } (f \ x) \ \partial s) = \text{cnj } (\int_Q x. f \ x \ \partial s)$
by (*simp add*: $\text{qbs-integral-def2-l}$)

lemma [*simp*]:

assumes *qbs-integrable s f*
shows *qbs-integral-Im*: $(\int_Q x. \text{Im } (f x) \partial s) = \text{Im } (\int_Q x. f x \partial s)$
and *qbs-integral-Re*: $(\int_Q x. \text{Re } (f x) \partial s) = \text{Re } (\int_Q x. f x \partial s)$
using *assms* **by**(*auto simp: qbs-integral-def2-l qbs-integrable-iff-integrable*)

lemma *qbs-integral-of-real[simp]:qbs-integrable s f \implies $(\int_Q x. \text{of-real } (f x) \partial s) = \text{of-real } (\int_Q x. f x \partial s)$*
by(*auto simp: qbs-integral-def2-l qbs-integrable-iff-integrable*)

lemma [*simp*]:
assumes *qbs-integrable s f*
shows *qbs-integral-fst*: $(\int_Q x. \text{fst } (f x) \partial s) = \text{fst } (\int_Q x. f x \partial s)$
and *qbs-integral-snd*: $(\int_Q x. \text{snd } (f x) \partial s) = \text{snd } (\int_Q x. f x \partial s)$
using *assms* **by**(*auto simp: qbs-integral-def2-l qbs-integrable-iff-integrable*)

lemma *real-qbs-integral-def*:
assumes *qbs-integrable s f*
shows *qbs-integral s f = enn2real $(\int^+_Q x. \text{ennreal } (f x) \partial s) - \text{enn2real } (\int^+_Q x. \text{ennreal } (-f x) \partial s)$*
using *qbs-integrable-morphism-dest[OF in-qbs-space-of-assms] assms*
by(*auto simp: qbs-integral-def2-l qbs-nn-integral-def2-l qbs-integrable-iff-integrable[symmetric] intro!: real-lebesgue-integral-def*)

lemma *Markov-inequality-qbs-prob*:
qbs-integrable s f \implies $AE_Q x \text{ in } s. 0 \leq f x \implies 0 < c \implies \mathcal{P}(x \text{ in } qbs-l s. c \leq f x) \leq (\int_Q x. f x \partial s) / c$
by(*auto simp: qbs-integral-def2-l AEq-qbs-l qbs-integrable-iff-integrable intro!: integral-Markov-inequality-measure[where A={}]*)

lemma *Chebyshev-inequality-qbs-prob*:
assumes *s \in qbs-space (monadP-qbs X)*
and *f \in X \rightarrow_Q qbs-borel qbs-integrable s $(\lambda x. (f x)^2)$*
and *0 < e*
shows *$\mathcal{P}(x \text{ in } qbs-l s. e \leq |f x - (\int_Q x. f x \partial s)| \leq (\int_Q x. (f x - (\int_Q x. f x \partial s))^2 \partial s) / e^2$*
using *prob-space.Chebyshev-inequality[OF qbs-l-prob-space[OF assms(1)] - - assms(4),of f] assms(2,3)*
by(*simp add: qbs-integral-def2-l qbs-integrable-iff-integrable lr-adjunction-correspondence measurable-qbs-l'[OF qbs-space-monadPM[OF assms(1)]]*)

lemma *qbs-l-return-qbs*:
assumes *x \in qbs-space X*
shows *qbs-l (return-qbs X x) = return (qbs-to-measure X) x*
proof –
interpret *qp: qbs-prob X $\lambda r. x \text{ return borel } 0$*
by(*auto simp: qbs-prob-def prob-space-return assms in-Mx-def real-distribution-def real-distribution-axioms-def*)
show *?thesis*
by(*simp add: qp.return-qbs[OF assms] distr-return qp.qbs-l*)

qed

lemma *qbs-l-bind-qbs*:

assumes [*qbs*]: $s \in \text{qbs-space } (\text{monadM-qbs } X) f \in X \rightarrow_Q \text{monadM-qbs } Y$
shows $\text{qbs-l } (s \ggg f) = \text{qbs-l } s \ggg_k \text{qbs-l } \circ f$ (**is** ?*lhs* = ?*rhs*)

proof –

from *rep-qbs-space-monadM*[*OF assms*(1)] **obtain** $\alpha \mu$

where $hs: s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \alpha \mu$ **by** *metis*

then interpret *qs*: *qbs-s-finite* $X \alpha \mu$ **by** *simp*

from *rep-qbs-Mx-monadM*[*OF qbs-morphism-Mx*[*OF assms*(2) *qs.in-Mx*]] **obtain**
 βk **where**

$hk: f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \beta \in \text{qbs-Mx } Y \text{ s-finite-kernel borel borel } k \wedge r.$
qbs-s-finite $Y \beta (k r)$

by *metis*

then interpret *sk*: *s-finite-kernel borel borel* k **by** *simp*

interpret *im*: *in-Mx* $Y \beta$ **using** $hk(2)$ **by**(*simp add: in-Mx-def*)

have ?*lhs* = *distr* $(\mu \ggg_k k)$ (*qbs-to-measure* Y) β

by(*simp add: qs.bind-qbs*[*OF hs*(1) *assms*(2) $hk(2,3,1)$] *qbs-s-finite.qbs-l*[*OF*
qs.bind-qbs-s-finite[*OF hs*(1) *assms*(2) $hk(2,3,1)$]]])

also have $\dots = \mu \ggg_k (\lambda x. \text{distr } (k x) (\text{qbs-to-measure } Y) \beta)$

by(*auto intro!*: *sk.distr-bind-kernel simp: qs.mu-sets*)

also have $\dots = \mu \ggg_k (\lambda r. \text{qbs-l } ((f \circ \alpha) r))$

by(*simp add: qbs-s-finite.qbs-l*[*OF hk*(4)] $hk(1)$)

also have $\dots = \mu \ggg_k (\lambda r. (\lambda x. \text{qbs-l } (f x)) (\alpha r))$ **by** *simp*

also have $\dots = \text{distr } \mu (\text{qbs-to-measure } X) \alpha \ggg_k (\lambda x. \text{qbs-l } (f x))$

using *l-preserves-morphisms*[*of* $X \text{monadM-qbs } Y$] *assms*(2)

by(*auto intro!*: *measure-kernel.bind-kernel-distr*[*OF measure-kernel.measure-kernel-comp*[*OF*
qbs-l-measure-kernel],*symmetric*] *simp: sets-eq-imp-space-eq*[*OF qs.mu-sets*])

also have $\dots = ?\text{rhs}$

by(*simp add: hs*(1) *qs.qbs-l comp-def*)

finally show ?*thesis* .

qed

lemma *qbs-l-bind-qbsP*:

assumes [*qbs*]: $s \in \text{qbs-space } (\text{monadP-qbs } X) f \in X \rightarrow_Q \text{monadP-qbs } Y$

shows $\text{qbs-l } (s \ggg f) = \text{qbs-l } s \ggg \text{qbs-l } \circ f$

proof –

have $\text{qbs-l } (s \ggg f) = \text{qbs-l } s \ggg_k \text{qbs-l } \circ f$

by(*auto intro!*: *qbs-l-bind-qbs*[**where** $X=X$ **and** $Y=Y$] *qbs-space-monadPM*
qbs-morphism-monadPD)

also have $\dots = \text{qbs-l } s \ggg \text{qbs-l } \circ f$

using *qbs-l-measurable-prob qbs-morphism-imp-measurable*[*OF assms*(2)]

by(*auto intro!*: *bind-kernel-bind*[**where** $N=\text{qbs-to-measure } Y$] *measurable-prob-algebraD*
simp: measurable-qbs-l'[*OF qbs-space-monadPM*[*OF assms*(1)]]])

finally show ?*thesis* .

qed

lemma *qbs-integrable-return*[*simp, intro*]:

assumes $x \in \text{qbs-space } X \ f \in X \rightarrow_Q \text{qbs-borel}$
shows $\text{qbs-integrable } (\text{return-qbs } X \ x) \ f$
using assms **by** $(\text{auto simp: qbs-integrable-iff-integrable qbs-l-return-qbs}[OF \text{assms}(1)])$
 $\text{lr-adjunction-correspondence nn-integral-return space-L intro!: integrableI-bounded}$

lemma $\text{qbs-integrable-bind-return}$:

assumes $[\text{qbs}]: s \in \text{qbs-space } (\text{monadM-qbs } X) \ f \in Y \rightarrow_Q \text{qbs-borel } g \in X \rightarrow_Q Y$
shows $\text{qbs-integrable } (s \gg (\lambda x. \text{return-qbs } Y \ (g \ x))) \ f = \text{qbs-integrable } s \ (f \circ g)$ **(is ?lhs = ?rhs)**

proof –

from $\text{rep-qbs-space-monadM}[OF \text{assms}(1)]$ **obtain** $\alpha \ \mu$
where $hs: s = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \ \alpha \ \mu$ **by** metis
then interpret $qs: \text{qbs-s-finite } X \ \alpha \ \mu$ **by** simp

have $1: \text{return-qbs } Y \circ (g \circ \alpha) = (\lambda r. \llbracket Y, g \circ \alpha, \text{return borel } r \rrbracket_{sfin})$

by $(\text{auto intro!: return-qbs-comp qbs-morphism-Mx}[OF \text{assms}(3)])$

have $hb: \text{qbs-s-finite } Y \ (g \circ \alpha) \ \mu \ s \gg (\lambda x. \text{return-qbs } Y \ (g \ x)) = \llbracket Y, g \circ \alpha, \mu \rrbracket_{sfin}$

using $\text{qbs-s-finite.bind-qbs}[OF \text{hs}(2,1) \ \text{qbs-morphism-comp}[OF \text{assms}(3) \ \text{return-qbs-morphism}] \ \text{qbs-morphism-Mx}[OF \text{assms}(3)] \ \text{prob-kernel.s-finite-kernel-prob-kernel } 1[\text{simplified comp-assoc[symmetric]}]]$

$\text{qbs-s-finite.bind-qbs-s-finite}[OF \text{hs}(2,1) \ \text{qbs-morphism-comp}[OF \text{assms}(3) \ \text{return-qbs-morphism}] \ \text{qbs-morphism-Mx}[OF \text{assms}(3)] \ \text{prob-kernel.s-finite-kernel-prob-kernel } 1[\text{simplified comp-assoc[symmetric]}]]$

by $(\text{auto simp: prob-kernel-def' comp-def bind-kernel-return''}[OF \text{qs.mu-sets}])$

have $?lhs = \text{integrable } \mu \ (f \circ (g \circ \alpha))$

by $(\text{auto simp: hb}(2) \ \text{intro!: qbs-s-finite.qbs-integrable-iff-integrable}[OF \text{hb}(1), \text{simplified comp-def}] \ \text{simp: comp-def lr-adjunction-correspondence}[symmetric])$

also have $\dots = ?rhs$

by $(\text{auto simp: hs}(1) \ \text{comp-def lr-adjunction-correspondence}[symmetric] \ \text{intro!: qs.qbs-integrable-iff-integrable}[symmetric])$

finally show $?thesis$.

qed

lemma $\text{qbs-nn-integral-morphism}[\text{qbs}]: \text{qbs-nn-integral} \in \text{monadM-qbs } X \rightarrow_Q (X \Rightarrow_Q \text{qbs-borel}) \Rightarrow_Q \text{qbs-borel}$

proof $(\text{rule curry-preserves-morphisms}[OF \text{pair-qbs-morphismI}])$

fix $\alpha \ \beta$

assume $h: \alpha \in \text{qbs-Mx } (\text{monadM-qbs } X) \ \beta \in \text{qbs-Mx } (X \Rightarrow_Q (\text{qbs-borel} :: \text{ennreal quasi-borel}))$

from $\text{rep-qbs-Mx-monadM}[OF \text{h}(1)]$ **obtain** $a \ k$

where $ak: \alpha = (\lambda r. \llbracket X, a, k \ r \rrbracket_{sfin}) \ a \in \text{qbs-Mx } X \ \text{s-finite-kernel borel borel } k \wedge r. \text{qbs-s-finite } X \ a \ (k \ r)$

by metis

have $1: \text{borel-measurable } ((\text{borel} :: \text{real measure}) \otimes_M (\text{borel} :: \text{real measure})) = \text{qbs-borel} \otimes_Q \text{qbs-borel} \rightarrow_Q (\text{qbs-borel} :: \text{ennreal quasi-borel})$

by $(\text{metis borel-prod qbs-borel-prod standard-borel.standard-borel-r-full-faithful standard-borel-ne-borel standard-borel-ne-def})$

show $(\lambda r. \text{qbs-nn-integral } (\text{fst } (\alpha \ r, \beta \ r)) \ (\text{snd } (\alpha \ r, \beta \ r))) \in \text{qbs-Mx } \text{qbs-borel}$

unfolding *qbs-Mx-qbs-borel*
by(*rule measurable-cong*[**where** $f = \lambda r. \int^+ x. \beta r (a x) \partial k r$, *THEN iffD1*],
insert h ak(2))
(auto simp: qbs-s-finite.qbs-nn-integral-def[OF ak(4)] qbs-Mx-is-morphisms
ak(1) 1 intro!: s-finite-kernel.nn-integral-measurable-f[OF ak(3)])
qed

lemma *qbs-nn-integral-return*:
assumes $f \in X \rightarrow_Q \text{qbs-borel}$
and $x \in \text{qbs-space } X$
shows *qbs-nn-integral (return-qbs X x) f = f x*
using *assms by (auto intro!: nn-integral-return simp: qbs-nn-integral-def2-l qbs-l-return-qbs*
space-L lr-adjunction-correspondence)

lemma *qbs-nn-integral-bind*:
assumes $[qbs]: s \in \text{qbs-space (monadM-qbs } X)$
 $f \in X \rightarrow_Q \text{monadM-qbs } Y$ $g \in Y \rightarrow_Q \text{qbs-borel}$
shows *qbs-nn-integral (s \ggg f) g = qbs-nn-integral s ($\lambda y. (\text{qbs-nn-integral (f$*
y) g)) (**is** *?lhs = ?rhs*)

proof –
from *rep-qbs-space-monadM*[*OF assms(1)*] **obtain** $\alpha \mu$
where *hs: s = $\llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \alpha \mu$ by metis*
then interpret *qs: qbs-s-finite X $\alpha \mu$ by simp*
from *rep-qbs-Mx-monadM*[*OF qbs-morphism-Mx*[*OF assms(2) qs.in-Mx*]] **obtain**
 βk
where *hk: f $\circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{sfin}) \beta \in \text{qbs-Mx } Y$ s-finite-kernel borel*
borel k $\wedge r. \text{qbs-s-finite } Y \beta (k r)$
by *metis*
note *sf = qs.bind-qbs*[*OF hs(1) assms(2) hk(2,3,1)*] *qs.bind-qbs-s-finite*[*OF*
hs(1) assms(2) hk(2,3,1)]
have *?lhs = $(\int^+ x. g (\beta x) \partial(\mu \ggg_k k))$*
by(*simp add: sf(1) qbs-s-finite.qbs-nn-integral-def*[*OF sf(2)*])
also have $\dots = (\int^+ r. (\int^+ y. g (\beta y) \partial(k r)) \partial \mu)$
using *assms(3) hk(2) by (auto intro!: s-finite-kernel.nn-integral-bind-kernel*[*OF*
*hk(3)] *qs.mu-sets simp: s-finite-kernel-cong-sets*[*OF qs.mu-sets*] *lr-adjunction-correspondence)**
also have $\dots = ?rhs$
using *fun-cong*[*OF hk(1)*] **by**(*auto simp: hs(1) qs.qbs-nn-integral-def qbs-s-finite.qbs-nn-integral-def*[*OF*
hk(4),symmetric] *intro!: nn-integral-cong*)
finally show *?thesis .*
qed

lemma *qbs-nn-integral-bind-return*:
assumes $[qbs]: s \in \text{qbs-space (monadM-qbs } Y)$ $f \in Z \rightarrow_Q \text{qbs-borel}$ $g \in Y \rightarrow_Q Z$
shows *qbs-nn-integral (s $\ggg (\lambda y. \text{return-qbs } Z (g y))$) f = qbs-nn-integral s (f \circ*
g)
by(*auto simp: qbs-nn-integral-bind*[*OF assms(1) - assms(2)*] *qbs-nn-integral-return*
intro!: qbs-nn-integral-cong[*OF assms(1)*])

lemma *qbs-integral-morphism*[*qbs*]:

$qbs\text{-integral} \in \text{monadM-qbs } X \rightarrow_Q (X \Rightarrow_Q qbs\text{-borel}) \Rightarrow_Q (qbs\text{-borel} :: ('b :: \{\text{second-countable-topology, banach}\}) \text{quasi-borel})$
proof(*rule curry-preserves-morphisms*[*OF pair-qbs-morphismI*])
fix α **and** $\gamma :: - \Rightarrow - \Rightarrow 'b$
assume $h: \alpha \in qbs\text{-Mx } (\text{monadM-qbs } X) \gamma \in qbs\text{-Mx } (X \Rightarrow_Q qbs\text{-borel})$
from *rep-qbs-Mx-monadM*[*OF this(1)*] **obtain** βk
where $hk: \alpha = (\lambda r. \llbracket X, \beta, k r \rrbracket_{s\text{fin}}) \beta \in qbs\text{-Mx } X \text{ s-finite-kernel borel borel } k$
 $\wedge r. qbs\text{-s-finite } X \beta (k r)$
by *metis*
have $1: \text{borel-measurable } ((\text{borel} :: \text{real measure}) \otimes_M (\text{borel} :: \text{real measure})) = qbs\text{-borel} \otimes_Q qbs\text{-borel} \rightarrow_Q (qbs\text{-borel} :: (- :: \{\text{second-countable-topology, banach}\}) \text{quasi-borel})$
by (*metis borel-prod qbs-borel-prod standard-borel.standard-borel-r-full-faithful standard-borel-ne-borel standard-borel-ne-def*)
show $(\lambda r. qbs\text{-integral } (fst (\alpha r, \gamma r)) (snd (\alpha r, \gamma r))) \in qbs\text{-Mx borel}_Q$
unfolding *qbs-Mx-R*
by(*rule measurable-cong*[**where** $f = \lambda r. \int x. \gamma r (\beta x) \partial k r$, *THEN iffD1*], *insert h hk(2)*)
(auto simp: qbs-s-finite.qbs-integral-def[*OF hk(4)*] *qbs-Mx-is-morphisms hk(1)*
1 intro!: s-finite-kernel.integral-measurable-f[*OF hk(3)*])
qed

lemma *qbs-integral-return*:
assumes $[qbs]: f \in X \rightarrow_Q qbs\text{-borel } x \in qbs\text{-space } X$
shows $qbs\text{-integral } (\text{return-qbs } X x) f = f x$
by(*auto simp: qbs-integral-def2-l qbs-l-return-qbs lr-adjunction-correspondence*[*symmetric*]
space-L integral-return)

lemma
assumes $[qbs]: s \in qbs\text{-space } (\text{monadM-qbs } X) f \in X \rightarrow_Q \text{monadM-qbs } Y g \in Y \rightarrow_Q qbs\text{-borel}$
and $qbs\text{-integrable } s (\lambda x. \int_Q y. \text{norm } (g y) \partial f x) AE_Q x \text{ in } s. qbs\text{-integrable } (f x) g$
shows *qbs-integrable-bind*: $qbs\text{-integrable } (s \ggg f) g$ (**is** *?goal1*)
and *qbs-integral-bind*: $(\int_Q y. g y \partial (s \ggg f)) = (\int_Q x. \int_Q y. g y \partial f x \partial s)$ (**is** *?lhs = ?rhs*)
proof –
from *rep-qbs-space-monadM*[*OF assms(1)*] **obtain** $\alpha \mu$
where $hs: s = \llbracket X, \alpha, \mu \rrbracket_{s\text{fin}} qbs\text{-s-finite } X \alpha \mu$ **by** *metis*
then interpret $qs: qbs\text{-s-finite } X \alpha \mu$ **by** *simp*
from *rep-qbs-Mx-monadM*[*OF qbs-morphism-Mx*[*OF assms(2)*] *qs.in-Mx*] **obtain** βk
where $hk: f \circ \alpha = (\lambda r. \llbracket Y, \beta, k r \rrbracket_{s\text{fin}}) \beta \in qbs\text{-Mx } Y \text{ s-finite-kernel borel borel } k \wedge r. qbs\text{-s-finite } Y \beta (k r)$
by *metis*
note $sf = qs.\text{bind-qbs}$ [*OF hs(1) assms(2) hk(2,3,1)*]
have $g[\text{measurable}]: \wedge h M. h \in M \rightarrow_M qbs\text{-to-measure } Y \implies (\lambda x. g (h x)) \in M \rightarrow_M \text{borel}$
using *assms(3)* **by**(*auto simp: lr-adjunction-correspondence*)

interpret $qs2$: qbs - s -finite $Y \beta \mu \gg_k k$
by(rule qs .bind- qbs - s -finite[OF $hs(1)$ $assms(2)$ $hk(2,3,1)$])
show $?goal1$
by(auto simp: sf $qs2$. qbs -integrable-def intro!: s -finite-kernel.integrable-bind-kernel[OF $hk(3)$ qs . μ -sets])
(insert qs . AEq - AE [OF $assms(5)$ [simplified $hs(1)$],simplified fun-cong[OF $hk(1)$,simplified] qbs - s -finite. qbs -integrable-def[OF $hk(4)$]] $assms(4)$ [simplified $hs(1)$ qs . qbs -integrable-def fun-cong[OF $hk(1)$,simplified]],auto simp: $hs(1)$ qs . qbs -integrable-def qbs - s -finite. qbs -integral-def[OF $hk(4)$])
have $?lhs = (\int r. g (\beta r) \partial(\mu \gg_k k))$
by(simp add: sf $qs2$. qbs -integral-def)
also have $\dots = (\int r. (\int l. g (\beta l) \partial k r) \partial \mu)$
using qs . AEq - AE [OF $assms(5)$ [simplified $hs(1)$],simplified fun-cong[OF $hk(1)$,simplified] qbs - s -finite. qbs -integrable-def[OF $hk(4)$]] $assms(4)$ [simplified $hs(1)$ qs . qbs -integrable-def fun-cong[OF $hk(1)$,simplified]]
by(auto intro!: s -finite-kernel.integral-bind-kernel[OF $hk(3)$ qs . μ -sets] simp: qbs - s -finite. qbs -integral-def[OF $hk(4)$])
also have $\dots = (\int r. (\int_Q y. g y \partial \llbracket Y, \beta, k r \rrbracket_{sfin}) \partial \mu)$
by(auto intro!: Bochner-Integration.integral-cong simp: qbs - s -finite. qbs -integral-def[OF $hk(4)$])
also have $\dots = ?rhs$
by(auto simp: $hs(1)$ qs . qbs -integral-def fun-cong[OF $hk(1)$,simplified comp-def])
finally show $?lhs = ?rhs$.
qed

lemma qbs -integral-bind-return:

assumes [qbs]: $s \in qbs$ -space (monad M - qbs Y) $f \in Z \rightarrow_Q qbs$ -borel $g \in Y \rightarrow_Q Z$
shows qbs -integral ($s \gg (\lambda y. \text{return-}qbs\ Z (g\ y))$) $f = qbs$ -integral $s (f \circ g)$

proof –

from rep - qbs -space-monad M [OF $assms(1)$] **obtain** $\alpha \mu$
where hs : $s = \llbracket Y, \alpha, \mu \rrbracket_{sfin}$ qbs - s -finite $Y \alpha \mu$ **by** $metis$
then interpret qs : qbs - s -finite $Y \alpha \mu$ **by** $simp$

have hb : qbs - s -finite $Z (g \circ \alpha) \mu s \gg (\lambda y. \text{return-}qbs\ Z (g\ y)) = \llbracket Z, g \circ \alpha, \mu \rrbracket_{sfin}$

using qs .bind- qbs - s -finite[OF $hs(1)$ - qbs -morphism- Mx [OF $assms(3)$ qs .in- Mx] prob-kernel. s -finite-kernel-prob-kernel return- qbs -comp[OF qbs -morphism- Mx [OF $assms(3)$ qs .in- Mx],simplified comp- $assoc$ [symmetric] comp-def[of - g],simplified prob-kernel-def']

by(auto simp: qs .bind- qbs [OF $hs(1)$ - qbs -morphism- Mx [OF $assms(3)$ qs .in- Mx] prob-kernel. s -finite-kernel-prob-kernel return- qbs -comp[OF qbs -morphism- Mx [OF $assms(3)$ qs .in- Mx],simplified comp- $assoc$ [symmetric] comp-def[of - g],simplified prob-kernel-def'] bind-kernel-return''[OF qs . μ -sets])

show $?thesis$

by(simp add: $hb(2)$ qbs - s -finite. qbs -integral-def[OF $hb(1)$] qs . qbs -integral-def[simplified $hs(1)$ [symmetric]])

qed

4.1.12 Binary Product Measures

definition *qbs-pair-measure* :: [*'a qbs-measure, 'b qbs-measure*] ⇒ (*'a × 'b*) *qbs-measure*
 (**infix** \otimes_{Qmes} 80) **where**
qbs-pair-measure-def':*qbs-pair-measure* *p q* ≡ (*p* ≫ (λ*x*. *q* ≫ (λ*y*. *return-qbs*
 (*qbs-space-of p* \otimes_Q *qbs-space-of q*) (*x, y*))))

context *pair-qbs-s-finites*
begin

interpretation *rr* : *standard-borel-ne borel* \otimes_M *borel* :: (*real × real*) *measure*
by(*auto intro!*: *pair-standard-borel-ne*)

lemma

shows *qbs-pair-measure*: $\llbracket X, \alpha, \mu \rrbracket_{sfin} \otimes_{Qmes} \llbracket Y, \beta, \nu \rrbracket_{sfin} = \llbracket X \otimes_Q Y,$
map-prod $\alpha \beta \circ rr.from-real, distr (\mu \otimes_M \nu) borel rr.to-real \rrbracket_{sfin}$
and *qbs-pair-measure-s-finite*: *qbs-s-finite* ($X \otimes_Q Y$) (*map-prod* $\alpha \beta \circ rr.from-real$)
 (*distr* $(\mu \otimes_M \nu) borel rr.to-real$)
by(*simp-all add: qbs-pair-measure-def' pq1.qbs-l pq2.qbs-l qbs-bind-bind-return-pq*
qbs-s-finite-axioms)

lemma *qbs-l-qbs-pair-measure*:

qbs-l ($\llbracket X, \alpha, \mu \rrbracket_{sfin} \otimes_{Qmes} \llbracket Y, \beta, \nu \rrbracket_{sfin}$) = *distr* $(\mu \otimes_M \nu)$ (*qbs-to-measure*
 ($X \otimes_Q Y$)) (*map-prod* $\alpha \beta$)
by(*simp add: qbs-pair-measure qbs-s-finite.qbs-l[OF qbs-pair-measure-s-finite] distr-distr*
comp-assoc)

lemma *qbs-nn-integral-pair-measure*:

assumes [*qbs*]:*f* ∈ $X \otimes_Q Y \rightarrow_Q qbs-borel$
shows $(\int^+_Q z. f z \partial(\llbracket X, \alpha, \mu \rrbracket_{sfin} \otimes_{Qmes} \llbracket Y, \beta, \nu \rrbracket_{sfin})) = (\int^+_Q z. (f \circ$
map-prod $\alpha \beta) z \partial(\mu \otimes_M \nu))$
using *assms by*(*simp add: qbs-nn-integral-def2 qbs-pair-measure distr-distr comp-assoc*
nn-integral-distr lr-adjunction-correspondence)

lemma *qbs-integral-pair-measure*:

assumes [*qbs*]:*f* ∈ $X \otimes_Q Y \rightarrow_Q qbs-borel$
shows $(\int_Q z. f z \partial(\llbracket X, \alpha, \mu \rrbracket_{sfin} \otimes_{Qmes} \llbracket Y, \beta, \nu \rrbracket_{sfin})) = (\int_Q z. (f \circ map-prod$
 $\alpha \beta) z \partial(\mu \otimes_M \nu))$
using *assms by*(*simp add: qbs-integral-def2 qbs-pair-measure distr-distr comp-assoc*
integral-distr lr-adjunction-correspondence)

lemma *qbs-pair-measure-integrable-eq*:

qbs-integrable ($\llbracket X, \alpha, \mu \rrbracket_{sfin} \otimes_{Qmes} \llbracket Y, \beta, \nu \rrbracket_{sfin}$) *f* $\longleftrightarrow f \in X \otimes_Q Y \rightarrow_Q$
qbs-borel \wedge *integrable* $(\mu \otimes_M \nu)$ (*f* \circ (*map-prod* $\alpha \beta$)) (**is** *?h* $\longleftrightarrow ?h1 \wedge ?h2$)

proof *safe*

assume *h*: *?h*

show *?h1*

by(*auto intro!*: *qbs-integrable-morphism-dest[OF - h] simp: qbs-pair-measure-def'*)

have *1*:*integrable* (*distr* $(\mu \otimes_M \nu) borel$ (*to-real-on* (*borel* \otimes_M *borel*))) (*f* \circ


```

(map-prod  $\alpha \beta \circ$  from-real-into (borel  $\otimes_M$  borel)))
  using h[simplified qbs-pair-measure] by(simp add: qbs-integrable-def[of f] comp-def[of
f])
  have integrable ( $\mu \otimes_M \nu$ ) ( $\lambda x. (f \circ (map-prod \alpha \beta \circ$  from-real-into (borel  $\otimes_M$ 
borel))) (to-real-on (borel  $\otimes_M$  borel) x))
    by(intro integrable-distr[OF - 1]) simp
  thus ?h2
    by(simp add: comp-def)
next
  assume h: ?h1 ?h2
  then show ?h
    by(simp add: qbs-pair-measure qbs-integrable-def) (simp add: lr-adjunction-correspondence
integrable-distr-eq[of rr.to-real  $\mu \otimes_M \nu$  borel  $\lambda x. f (map-prod \alpha \beta (rr.from-real
x))$ ] comp-def)
qed

end

```

lemmas(in pair-qbs-probs) qbs-pair-measure-prob = qbs-prob-axioms

context

```

  fixes X Y p q
  assumes p[qbs]:p  $\in$  qbs-space (monadM-qbs X) and q[qbs]:q  $\in$  qbs-space (monadM-qbs
Y)
begin

```

```

lemma qbs-pair-measure-def: p  $\otimes_{Qmes}$  q = p  $\ggg$  ( $\lambda x. q \ggg (\lambda y. return-qbs (X
\otimes_Q Y) (x,y))$ )
  by(simp add: qbs-space-of-in[OF p] qbs-space-of-in[OF q] qbs-pair-measure-def')

```

```

lemma qbs-pair-measure-def2: p  $\otimes_{Qmes}$  q = q  $\ggg$  ( $\lambda y. p \ggg (\lambda x. return-qbs (X
\otimes_Q Y) (x,y))$ )
  by(simp add: bind-qbs-return-rotate qbs-pair-measure-def)

```

lemma

```

  assumes f  $\in$  X  $\otimes_Q$  Y  $\rightarrow_Q$  monadM-qbs Z
  shows qbs-pair-bind-bind-return1': q  $\ggg$  ( $\lambda y. p \ggg (\lambda x. f (x,y))$ ) = p  $\otimes_{Qmes}$  q
 $\ggg$  f
  and qbs-pair-bind-bind-return2': p  $\ggg$  ( $\lambda x. q \ggg (\lambda y. f (x,y))$ ) = p  $\otimes_{Qmes}$  q
 $\ggg$  f
  by(simp-all add: qbs-bind-bind-return1[OF assms] qbs-bind-bind-return2[OF assms]
bind-qbs-return-rotate qbs-pair-measure-def)

```

lemma

```

  assumes [qbs]:f  $\in$  X  $\rightarrow_Q$  exp-qbs Y (monadM-qbs Z)
  shows qbs-pair-bind-bind-return1'': q  $\ggg$  ( $\lambda y. p \ggg (\lambda x. f x y)$ ) = p  $\otimes_{Qmes}$  q
 $\ggg$  ( $\lambda x. f (fst x) (snd x)$ )
  and qbs-pair-bind-bind-return2'': p  $\ggg$  ( $\lambda x. q \ggg (\lambda y. f x y)$ ) = p  $\otimes_{Qmes}$  q
 $\ggg$  ( $\lambda x. f (fst x) (snd x)$ )

```

by(*auto intro!*: *qbs-pair-bind-bind-return1* '[**where** $f = \lambda x. f (fst\ x) (snd\ x)$, *simplified*]
qbs-pair-bind-bind-return2 '[**where** $f = \lambda x. f (fst\ x) (snd\ x)$, *simplified*] *uncurry-preserves-morphisms*)
qbs

lemma *qbs-nn-integral-Fubini-fst*:

assumes [*qbs*]: $f \in X \otimes_Q Y \rightarrow_Q \text{qbs-borel}$

shows $(\int^+_Q x. \int^+_Q y. f(x,y) \partial q \partial p) = (\int^+_Q z. f z \partial(p \otimes_{Q\text{mes}} q))$
(is ?lhs = ?rhs)

proof –

have *?lhs* = $(\int^+_Q x. \int^+_Q y. \text{qbs-nn-integral}(\text{return-qbs}(X \otimes_Q Y)(x, y)) f \partial q \partial p)$

by(*auto intro!*: *qbs-nn-integral-cong p q simp: qbs-nn-integral-return*)

also have ... = *?rhs*

by(*auto intro!*: *qbs-nn-integral-cong[OF p] simp:qbs-nn-integral-bind[OF q - assms] qbs-nn-integral-bind[OF p - assms] qbs-pair-measure-def*)

finally show *?thesis* .

qed

lemma *qbs-nn-integral-Fubini-snd*:

assumes [*qbs*]: $f \in X \otimes_Q Y \rightarrow_Q \text{qbs-borel}$

shows $(\int^+_Q y. \int^+_Q x. f(x,y) \partial p \partial q) = (\int^+_Q z. f z \partial(p \otimes_{Q\text{mes}} q))$ **(is ?lhs = ?rhs)**

proof –

have *?lhs* = $(\int^+_Q y. \int^+_Q x. \text{qbs-nn-integral}(\text{return-qbs}(X \otimes_Q Y)(x, y)) f \partial p \partial q)$

by(*auto intro!*: *qbs-nn-integral-cong p q simp: qbs-nn-integral-return*)

also have ... = *?rhs*

by(*auto intro!*: *qbs-nn-integral-cong[OF q] simp:qbs-nn-integral-bind[OF q - assms] qbs-nn-integral-bind[OF p - assms] qbs-pair-measure-def2*)

finally show *?thesis* .

qed

lemma *qbs-ennintegral-indep-mult*:

assumes [*qbs*]: $f \in X \rightarrow_Q \text{qbs-borel}$ $g \in Y \rightarrow_Q \text{qbs-borel}$

shows $(\int^+_Q z. f(fst\ z) * g(snd\ z) \partial(p \otimes_{Q\text{mes}} q)) = (\int^+_Q x. f\ x \partial p) * (\int^+_Q y. g\ y \partial q)$ **(is ?lhs = ?rhs)**

proof –

have *?lhs* = $(\int^+_Q x. \int^+_Q y. f\ x * g\ y \partial q \partial p)$

using *qbs-nn-integral-Fubini-fst*[**where** $f = \lambda z. f(fst\ z) * g(snd\ z)$] **by** *simp*

also have ... = $(\int^+_Q x. f\ x * \int^+_Q y. g\ y \partial q \partial p)$

by(*simp add: qbs-nn-integral-cmult[OF q]*)

also have ... = *?rhs*

by(*simp add: qbs-nn-integral-cmult[OF p] ab-semigroup-mult-class.mult commute*[**where** $b = \text{qbs-nn-integral}\ q\ g$])

finally show *?thesis* .

qed

end

lemma *qbs-l-qbs-pair-measure*:

assumes *standard-borel M standard-borel N*

defines $X \equiv \text{measure-to-qbs } M$ **and** $Y \equiv \text{measure-to-qbs } N$

assumes $[qbs]: p \in \text{qbs-space } (\text{monadM-qbs } X) \ q \in \text{qbs-space } (\text{monadM-qbs } Y)$

shows $\text{qbs-l } (p \otimes_{Q\text{mes}} q) = \text{qbs-l } p \otimes_M \text{qbs-l } q$

proof –

obtain $\alpha \ \mu \ \beta \ \nu$

where $hp: p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \ \alpha \ \mu$

and $hq: q = \llbracket Y, \beta, \nu \rrbracket_{sfin} \text{qbs-s-finite } Y \ \beta \ \nu$

using *rep-qbs-space-monadM assms(5,6)* **by** *meson*

have $1:\text{sets } (\text{qbs-to-measure } (X \otimes_Q Y)) = \text{sets } (M \otimes_M N)$

by(*auto simp: r-preserves-product[symmetric] X-def Y-def intro!: standard-borel.lr-sets-ident pair-standard-borel assms*)

have $\text{qbs-l } (p \otimes_{Q\text{mes}} q) = \text{qbs-l } p \ggg_k \text{qbs-l } \circ (\lambda x. q \ggg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x, y)))$

by(*auto simp: qbs-pair-measure-def[of p X q Y] intro!: qbs-l-bind-qbs[of - X - X \otimes_Q Y]*)

also have $\dots = \text{qbs-l } p \ggg_k (\lambda x. \text{qbs-l } (q \ggg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x, y))))$

by(*simp add: comp-def*)

also have $\dots = \text{qbs-l } p \ggg_k (\lambda x. \text{qbs-l } q \ggg_k \text{qbs-l } \circ (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x, y)))$

by(*auto intro!: bind-kernel-cong-All qbs-l-bind-qbs[of - Y - X \otimes_Q Y] simp: space-qbs-l-in[OF assms(5)]*)

also have $\dots = \text{qbs-l } p \ggg_k (\lambda x. \text{qbs-l } q \ggg_k (\lambda y. \text{return } (\text{qbs-to-measure } (X \otimes_Q Y)) (x, y)))$

by(*auto simp: comp-def space-qbs-l-in[OF assms(6)] space-qbs-l-in[OF assms(5)] qbs-l-return-qbs intro!: bind-kernel-cong-All*)

also have $\dots = \text{qbs-l } p \ggg_k (\lambda x. \text{qbs-l } q \ggg_k (\lambda y. \text{return } (M \otimes_M N) (x, y)))$

by(*simp add: return-cong[OF 1]*)

also have $\dots = \text{qbs-l } p \ggg_k (\lambda x. \text{qbs-l } q \ggg_k (\lambda y. \text{return } (\text{qbs-l } p \otimes_M \text{qbs-l } q) (x, y)))$

by(*auto cong: return-cong sets-pair-measure-cong simp: sets-qbs-l[OF assms(5)] standard-borel.lr-sets-ident[OF assms(1)] sets-qbs-l[OF assms(6)] standard-borel.lr-sets-ident[OF assms(2)] X-def Y-def*)

also have $\dots = \text{qbs-l } p \otimes_M \text{qbs-l } q$

by(*auto intro!: pair-measure-eq-bind-s-finite[symmetric] qbs-l-s-finite.s-finite-measure-axioms*)

finally show *?thesis* .

qed

lemma *qbs-pair-measure-morphism[qbs]*: $\text{qbs-pair-measure} \in \text{monadM-qbs } X \rightarrow_Q \text{monadM-qbs } Y \Rightarrow_Q \text{monadM-qbs } (X \otimes_Q Y)$

by(*rule curry-preserves-morphisms, rule qbs-morphism-cong'[where f=(\lambda(p,q). (p \ggg (\lambda x. q \ggg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x, y)))))] (auto simp: pair-qbs-space qbs-pair-measure-def)*)

lemma *qbs-pair-measure-morphismP*:

$\text{qbs-pair-measure} \in \text{monadP-qbs } X \rightarrow_Q \text{monadP-qbs } Y \Rightarrow_Q \text{monadP-qbs } (X \otimes_Q Y)$

proof –
note $[qbs] = \text{return-qbs-morphismP bind-qbs-morphismP}$
show *?thesis*
by(*rule curry-preserves-morphisms, rule qbs-morphism-cong'*[**where** $f = (\lambda(p,q). (p \gg (\lambda x. q \gg (\lambda y. \text{return-qbs } (X \otimes_Q Y) (x, y))))$)] (*auto simp: pair-qbs-space qbs-pair-measure-def[OF qbs-space-monadPM qbs-space-monadPM]*)
qed

lemma *qbs-nn-integral-indep1*:
assumes $[qbs]: p \in \text{qbs-space } (\text{monadM-qbs } X) \ q \in \text{qbs-space } (\text{monadP-qbs } X) \ f \in X \rightarrow_Q \text{qbs-borel}$
shows $(\int^+_Q z. f (\text{fst } z) \ \partial(p \otimes_{Q_{\text{mes}}} q)) = (\int^+_Q x. f x \ \partial p)$

proof –
obtain $Y \ \beta \ \nu$ **where** *hq*:
 $q = \llbracket Y, \beta, \nu \rrbracket_{\text{sf in}} \text{qbs-prob } Y \ \beta \ \nu$
using *rep-qbs-space-monadP[OF assms(2)]* **by** *blast*
then interpret *qbs-prob* $Y \ \beta \ \nu$ **by** *simp*
show *?thesis*
by(*simp add: qbs-nn-integral-const-prob[OF in-space-monadP] qbs-nn-integral-Fubini-snd[OF assms(1) in-space-monadM, symmetric] hq(1)*)
qed

lemma *qbs-nn-integral-indep2*:
assumes $[qbs]: q \in \text{qbs-space } (\text{monadM-qbs } Y) \ p \in \text{qbs-space } (\text{monadP-qbs } X) \ f \in Y \rightarrow_Q \text{qbs-borel}$
shows $(\int^+_Q z. f (\text{snd } z) \ \partial(p \otimes_{Q_{\text{mes}}} q)) = (\int^+_Q y. f y \ \partial q)$
proof –
obtain $X \ \alpha \ \mu$ **where** *hp*:
 $p = \llbracket X, \alpha, \mu \rrbracket_{\text{sf in}} \text{qbs-prob } X \ \alpha \ \mu$
using *rep-qbs-space-monadP[OF assms(2)]* **by** *metis*
then interpret *qbs-prob* $X \ \alpha \ \mu$ **by** *simp*
show *?thesis*
by(*simp add: qbs-nn-integral-const-prob[OF in-space-monadP] qbs-nn-integral-Fubini-snd[OF in-space-monadM assms(1), symmetric] hp(1)*)
qed

context
begin

interpretation *rr* : *standard-borel-ne borel* $\otimes_M \text{borel} :: (\text{real} \times \text{real}) \text{ measure}$
by(*auto intro!: pair-standard-borel-ne*)

lemma *qbs-integrable-pair-swap*:
assumes *qbs-integrable* $(p \otimes_{Q_{\text{mes}}} q) \ f$
shows *qbs-integrable* $(q \otimes_{Q_{\text{mes}}} p) \ (\lambda(x,y). f (y,x))$
proof –
obtain $X \ \alpha \ \mu \ Y \ \beta \ \nu$
where *hp*: $p = \llbracket X, \alpha, \mu \rrbracket_{\text{sf in}} \text{qbs-s-finite } X \ \alpha \ \mu$
and *hq*: $q = \llbracket Y, \beta, \nu \rrbracket_{\text{sf in}} \text{qbs-s-finite } Y \ \beta \ \nu$

using *rep-qbs-s-finite-measure* **by** *meson*
interpret *p1: pair-qbs-s-finites X α μ Y β ν*
by(*simp add: pair-qbs-s-finites-def hq hp*)
interpret *p2: pair-qbs-s-finites Y β ν X α μ*
by(*simp add: pair-qbs-s-finites-def hq hp*)
show *?thesis*
using *assms* **by**(*auto simp: hp(1) hq(1) p1.qbs-pair-measure p2.qbs-pair-measure*
p1.qbs-integrable-def p2.qbs-integrable-def)
(*auto simp add: integrable-distr-eq lr-adjunction-correspondence qbs-Mx-is-morphisms*
map-prod-def split-beta' intro!: integrable-product-swap-iff-s-finite[OF p1.pq2.s-finite-measure-axioms
p1.pq1.s-finite-measure-axioms, THEN iffD1])
qed

lemma *qbs-integrable-pair1'*:

assumes [*qbs*]:*p ∈ qbs-space (monadM-qbs X)*
q ∈ qbs-space (monadM-qbs Y)
f ∈ X ⊗_Q Y →_Q qbs-borel
qbs-integrable p (λx. ∫_Q y. norm (f (x,y)) ∂q)
and *AE_Q x in p. qbs-integrable q (λy. f (x,y))*
shows *qbs-integrable (p ⊗_{Qmes} q) f*
proof –
obtain *α μ β ν*
where *hp: p = [X, α, μ]_{sfin} qbs-s-finite X α μ*
and *hq: q = [Y, β, ν]_{sfin} qbs-s-finite Y β ν*
using *rep-qbs-space-monadM assms(1,2)* **by** *meson*
then interpret *pqs: pair-qbs-s-finites X α μ Y β ν*
by(*simp add: pair-qbs-s-finites-def*)
have [*measurable*]: *f ∈ borel-measurable (qbs-to-measure (X ⊗_Q Y))*
by(*simp add: lr-adjunction-correspondence[symmetric]*)
show *?thesis*
using *assms(4) pqs.pq1.AEq-AE[OF assms(5)[simplified hp(1)]]*
by(*auto simp add: pqs.qbs-integrable-def pqs.qbs-pair-measure hp(1) hq(1) inte-*
grable-distr-eq pqs.pq2.qbs-integrable-def pqs.pq1.qbs-integrable-def pqs.pq2.qbs-integral-def
intro!: s-finite-measure.Fubini-integrable' pqs.pq2.s-finite-measure-axioms)
qed

lemma

assumes *qbs-integrable (p ⊗_{Qmes} q) f*
shows *qbs-integrable-pair1D1': qbs-integrable p (λx. ∫_Q y. f (x,y) ∂q)* **(is**
?g1)
and *qbs-integrable-pair1D1-norm': qbs-integrable p (λx. ∫_Q y. norm (f (x,y))*
∂q) **(is** *?g2)*
and *qbs-integrable-pair1D2': AE_Q x in p. qbs-integrable q (λy. f (x,y))* **(is**
?g3)
and *qbs-integrable-pair2D1': qbs-integrable q (λy. ∫_Q x. f (x,y) ∂p)* **(is**
?g4)
and *qbs-integrable-pair2D1-norm': qbs-integrable q (λy. ∫_Q x. norm (f (x,y))*
∂p) **(is** *?g5)*
and *qbs-integrable-pair2D2': AE_Q y in q. qbs-integrable p (λx. f (x,y))* **(is**

?g6)
and *qbs-integral-Fubini-fst'*: $(\int_Q x. \int_Q y. f(x,y) \partial q \partial p) = (\int_Q z. f z \partial(p$
 $\otimes_{Qmes} q))$ (**is** ?g7)
and *qbs-integral-Fubini-snd'*: $(\int_Q y. \int_Q x. f(x,y) \partial p \partial q) = (\int_Q z. f z \partial(p$
 $\otimes_{Qmes} q))$ (**is** ?g8)
proof –
obtain $X \alpha \mu Y \beta \nu$
where $hp: p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{ qbs-s-finite } X \alpha \mu$
and $hq: q = \llbracket Y, \beta, \nu \rrbracket_{sfin} \text{ qbs-s-finite } Y \beta \nu$
by (*meson rep-qbs-space-of*)
then interpret $pqs: \text{pair-qbs-s-finites } X \alpha \mu Y \beta \nu$
by (*simp add: pair-qbs-s-finites-def*)
have $[qbs]: p \in \text{qbs-space (monadM-qbs } X) \text{ } q \in \text{qbs-space (monadM-qbs } Y)$
by (*simp-all add: hp hq*)
note *qbs-pair-measure-morphism*[*qbs*]
have $f[qbs]: f \in X \otimes_Q Y \rightarrow_Q \text{ qbs-borel}$
by (*auto intro! : qbs-integrable-morphism-dest[OF - assms]*)
have $[measurable]: f \in \text{borel-measurable (qbs-to-measure (} X \otimes_Q Y))$
by (*simp add: lr-adjunction-correspondence[symmetric]*)
show ?g1 ?g2 ?g4 ?g5
using *assms*
by (*auto simp: hp(1) hq(1) pqs.qbs-integrable-def pqs.qbs-pair-measure inte-*
grable-distr-eq pqs.pq1.qbs-integrable-def pqs.pq2.qbs-integrable-def pqs.pq2.qbs-integral-def
pqs.pq1.qbs-integral-def intro! : Bochner-Integration.integrable-cong[where g= $\lambda r. \int_Q$
y. f ($\alpha r, y$) $\partial \llbracket Y, \beta, \nu \rrbracket_{sfin}$ and $f = \lambda x. \int y. f (\alpha x, \beta y) \partial \nu$ and $N0 = \mu$, THEN
iffD1] Bochner-Integration.integrable-cong[where g= $\lambda r. \int_Q x. f (x, \beta r) \partial \llbracket X, \alpha,$
 $\mu \rrbracket_{sfin}$ and $f = \lambda y. \int x. f (\alpha x, \beta y) \partial \mu$ and $N0 = \nu$, THEN iffD1])
(auto intro! : pqs.pq2.integrable-fst'[of μ] integrable-snd-s-finite[OF pqs.pq1.s-finite-measure-axioms
pqs.pq2.s-finite-measure-axioms] simp: map-prod-def split-beta')
show ?g3 ?g6
using *assms*
by (*auto simp: hp(1) pqs.pq1.AEq-AE-iff hq(1) pqs.pq2.AEq-AE-iff pqs.qbs-integrable-def*
pqs.qbs-pair-measure integrable-distr-eq)
(auto simp: pqs.pq1.qbs-integrable-def pqs.pq2.qbs-integrable-def map-prod-def
split-beta' intro! : pqs.pq2.AE-integrable-fst'' AE-integrable-snd-s-finite[OF pqs.pq1.s-finite-measure-axioms
pqs.pq2.s-finite-measure-axioms])
show ?g7 ?g8
using *assms*
by (*auto simp: hp(1) hq(1) pqs.qbs-integrable-def pqs.qbs-pair-measure pqs.qbs-integral-def*
pqs.pq1.qbs-integral-def pqs.pq2.qbs-integral-def integral-distr integrable-distr-eq)
(auto simp: map-prod-def split-beta' intro! : pqs.pq2.integral-fst'''[of $\mu \lambda x. f (\alpha$
(fst x), β (snd x)), simplified] integral-snd-s-finite[OF pqs.pq1.s-finite-measure-axioms
pqs.pq2.s-finite-measure-axioms, of $\lambda x y. f (\alpha x, \beta y)$, simplified split-beta'])
qed
end
lemma
assumes $h: \text{qbs-integrable } (p \otimes_{Qmes} q)$ (*case-prod f*)

shows *qbs-integrable-pair1D1*: *qbs-integrable* p ($\lambda x. \int_Q y. f x y \partial q$)
and *qbs-integrable-pair1D1-norm*: *qbs-integrable* p ($\lambda x. \int_Q y. \text{norm} (f x y) \partial q$)
and *qbs-integrable-pair1D2*: $AE_Q x$ in p . *qbs-integrable* q ($\lambda y. f x y$)
and *qbs-integrable-pair2D1*: *qbs-integrable* q ($\lambda y. \int_Q x. f x y \partial p$)
and *qbs-integrable-pair2D1-norm*: *qbs-integrable* q ($\lambda y. \int_Q x. \text{norm} (f x y) \partial p$)
and *qbs-integrable-pair2D2*: $AE_Q y$ in q . *qbs-integrable* p ($\lambda x. f x y$)
and *qbs-integral-Fubini-fst*: $(\int_Q x. \int_Q y. f x y \partial q \partial p) = (\int_Q (x,y). f x y \partial(p$
 $\otimes_{Qmes} q))$ (**is** ?g7)
and *qbs-integral-Fubini-snd*: $(\int_Q y. \int_Q x. f x y \partial p \partial q) = (\int_Q (x,y). f x y \partial(p$
 $\otimes_{Qmes} q))$ (**is** ?g8)
using *qbs-integrable-pair1D1'*[OF h] *qbs-integrable-pair1D1-norm'*[OF h] *qbs-integrable-pair1D2'*[OF
 h] *qbs-integral-Fubini-fst'*[OF h]
qbs-integrable-pair2D1'[OF h] *qbs-integrable-pair2D1-norm'*[OF h] *qbs-integrable-pair2D2'*[OF
 h] *qbs-integral-Fubini-snd'*[OF h]
by *simp-all*

lemma *qbs-integrable-pair2'*:

assumes $p \in \text{qbs-space} (\text{monadM-qbs } X)$
 $q \in \text{qbs-space} (\text{monadM-qbs } Y)$
 $f \in X \otimes_Q Y \rightarrow_Q \text{qbs-borel}$
qbs-integrable q ($\lambda y. \int_Q x. \text{norm} (f (x,y)) \partial p$)
and $AE_Q y$ in q . *qbs-integrable* p ($\lambda x. f (x,y)$)
shows *qbs-integrable* $(p \otimes_{Qmes} q) f$
using *qbs-integrable-pair-swap*[OF *qbs-integrable-pair1'*[OF *assms(2,1)*] *qbs-morphism-pair-swap*[OF
assms(3)],*simplified*], OF *assms(4,5)*]
by *simp*

lemma *qbs-integrable-indep-mult*:

fixes $f :: - \Rightarrow - :: \{\text{real-normed-div-algebra}, \text{second-countable-topology}\}$
assumes *qbs-integrable* p *qbs-integrable* q g
shows *qbs-integrable* $(p \otimes_{Qmes} q) (\lambda x. f (fst x) * g (snd x))$
proof –
obtain $X \alpha \mu Y \beta \nu$
where $hp: p = \llbracket X, \alpha, \mu \rrbracket_{sfin} \text{qbs-s-finite } X \alpha \mu$
and $hq: q = \llbracket Y, \beta, \nu \rrbracket_{sfin} \text{qbs-s-finite } Y \beta \nu$
by (*meson rep-qbs-space-of*)
then interpret $pqs: \text{pair-qbs-s-finites } X \alpha \mu Y \beta \nu$
by(*simp add: pair-qbs-s-finites-def*)
have [*qbs*]: $f \in X \rightarrow_Q \text{qbs-borel}$ $g \in Y \rightarrow_Q \text{qbs-borel}$ $p \in \text{qbs-space} (\text{monadM-qbs } X)$ $q \in \text{qbs-space} (\text{monadM-qbs } Y)$
by(*auto intro!*: *qbs-integrable-morphism-dest* *assms simp:hp hq*)
show ?thesis
by(*auto intro!*: *qbs-integrable-pair1'*[of - $X - Y$] *qbs-integrable-mult-left* *qbs-integrable-norm*
assms(1)] *AEq-I2*[of - X] *simp: norm-mult* *qbs-integrable-mult-right*[OF *assms(2)*])
qed

lemma *qbs-integrable-indep1*:

fixes $f :: - \Rightarrow - :: \{\text{real-normed-div-algebra}, \text{second-countable-topology}\}$
assumes *qbs-integrable* p f $q \in \text{qbs-space} (\text{monadP-qbs } Y)$

shows $qbs\text{-integrable } (p \otimes_{Qmes} q) (\lambda x. f (fst x))$
using $qbs\text{-integrable-indep-mult}[OF\ assms(1)]\ qbs\text{-integrable-const}[OF\ assms(2), of\ 1]$ **by** *simp*

lemma *qbs-integral-indep1*:

fixes $f :: - \Rightarrow - :: \{real\text{-normed-div-algebra}, second\text{-countable-topology}\}$
assumes $qbs\text{-integrable } p\ f\ q \in qbs\text{-space } (monadP\text{-}qbs\ Y)$
shows $(\int_Q z. f (fst z) \partial(p \otimes_{Qmes} q)) = (\int_Q x. f x \partial p)$
using $qbs\text{-integral-Fubini-snd}'[OF\ qbs\text{-integrable-indep1}[OF\ assms]]$
by(*simp add: qbs-integral-const-prob[OF assms(2)]*)

lemma *qbs-integrable-indep2*:

fixes $g :: - \Rightarrow - :: \{real\text{-normed-div-algebra}, second\text{-countable-topology}\}$
assumes $qbs\text{-integrable } q\ g\ p \in qbs\text{-space } (monadP\text{-}qbs\ X)$
shows $qbs\text{-integrable } (p \otimes_{Qmes} q) (\lambda x. g (snd x))$
using $qbs\text{-integrable-pair-swap}[OF\ qbs\text{-integrable-indep1}[OF\ assms]]$
by(*simp add: split-beta'*)

lemma *qbs-integral-indep2*:

fixes $g :: - \Rightarrow - :: \{real\text{-normed-div-algebra}, second\text{-countable-topology}\}$
assumes $qbs\text{-integrable } q\ g\ p \in qbs\text{-space } (monadP\text{-}qbs\ X)$
shows $(\int_Q z. g (snd z) \partial(p \otimes_{Qmes} q)) = (\int_Q y. g y \partial q)$
using $qbs\text{-integral-Fubini-fst}'[OF\ qbs\text{-integrable-indep2}[OF\ assms]]$
by(*simp add: qbs-integral-const-prob[OF assms(2)]*)

lemma *qbs-integral-indep-mult1*:

fixes f **and** $g :: - \Rightarrow - :: \{real\text{-normed-field}, second\text{-countable-topology}\}$
assumes $p \in qbs\text{-space } (monadP\text{-}qbs\ X)$ $q \in qbs\text{-space } (monadP\text{-}qbs\ Y)$
and $qbs\text{-integrable } p\ f\ qbs\text{-integrable } q\ g$
shows $(\int_Q z. f (fst z) * g (snd z) \partial(p \otimes_{Qmes} q)) = (\int_Q x. f x \partial p) * (\int_Q y. g y \partial q)$
using $qbs\text{-integral-Fubini-fst}'[OF\ qbs\text{-integrable-indep-mult}[OF\ assms(3,4)]]$
by *simp*

lemma *qbs-integral-indep-mult2*:

fixes f **and** $g :: - \Rightarrow - :: \{real\text{-normed-field}, second\text{-countable-topology}\}$
assumes $p \in qbs\text{-space } (monadP\text{-}qbs\ X)$ $q \in qbs\text{-space } (monadP\text{-}qbs\ Y)$
and $qbs\text{-integrable } p\ f\ qbs\text{-integrable } q\ g$
shows $(\int_Q z. g (snd z) * f (fst z) \partial(p \otimes_{Qmes} q)) = (\int_Q y. g y \partial q) * (\int_Q x. f x \partial p)$
using $qbs\text{-integral-indep-mult1}[OF\ assms]$ **by**(*simp add: mult.commute*)

4.1.13 The Inverse Function of l

definition *qbs-l-inverse* $:: 'a\ measure \Rightarrow 'a\ qbs\text{-measure}$ **where**

$qbs\text{-l-inverse } M \equiv \llbracket measure\text{-to-}qbs\ M, from\text{-real-into } M, distr\ M\ borel\ (to\text{-real-on } M) \rrbracket_{sfin}$

context *standard-borel-ne*

begin

lemma *qbs-l-inverse-def2*:

assumes [*measurable-cong*]: *sets* $\mu = \text{sets } M$

and *s-finite-measure* μ

shows *qbs-l-inverse* $\mu = \llbracket \text{measure-to-qbs } M, \text{from-real, distr } \mu \text{ borel to-real} \rrbracket_{sfin}$

proof –

interpret *s*: *standard-borel-ne* μ

using *assms standard-borel-ne-axioms standard-borel-ne-sets* **by** *blast*

have [*measurable*]: *s.from-real* $\in \text{borel} \rightarrow_M M$

using *assms(1) measurable-cong-sets s.from-real-measurable* **by** *blast*

show *?thesis*

by(*auto simp: distr-distr qbs-l-inverse-def qbs-s-finite-eq-def qbs-s-finite-def in-Mx-def qbs-Mx-R qbs-s-finite-axioms-def intro!: qbs-s-finite-measure-eq s-finite-measure.s-finite-measure-distr assms cong: measure-to-qbs-cong-sets[OF assms(1)]*) (*auto intro!: distr-cong simp: sets-eq-imp-space-eq[OF assms(1)]*)

qed

lemma

assumes [*measurable-cong*]: *sets* $\mu = \text{sets } M$

shows *qbs-l-inverse-s-finite*: *s-finite-measure* $\mu \implies \text{qbs-s-finite (measure-to-qbs } M) \text{ from-real (distr } \mu \text{ borel to-real)}$

and *qbs-l-inverse-qbs-prob*: *prob-space* $\mu \implies \text{qbs-prob (measure-to-qbs } M) \text{ from-real (distr } \mu \text{ borel to-real)}$

by(*auto simp: qbs-s-finite-def qbs-prob-def in-Mx-def qbs-s-finite-axioms-def real-distribution-def real-distribution-axioms-def qbs-Mx-R intro!: s-finite-measure.s-finite-measure-distr prob-space.prob-space-distr*)

corollary

assumes [*measurable-cong*]: *sets* $\mu = \text{sets } M$

shows *qbs-l-inverse-in-space-monadM*: *s-finite-measure* $\mu \implies \text{qbs-l-inverse } \mu \in \text{qbs-space (monadM-qbs } M)$

and *qbs-l-inverse-in-space-monadP*: *prob-space* $\mu \implies \text{qbs-l-inverse } \mu \in \text{qbs-space (monadP-qbs } M)$

by(*auto simp: qbs-l-inverse-def2[OF assms(1)] qbs-l-inverse-def2[OF assms(1) prob-space.s-finite-measure-prob] assms intro!: qbs-s-finite.in-space-monadM[OF qbs-l-inverse-s-finite] qbs-prob.in-space-monadP[OF qbs-l-inverse-qbs-prob]*)

lemma *qbs-l-qbs-l-inverse*:

assumes [*measurable-cong*]: *sets* $\mu = \text{sets } M$ *s-finite-measure* μ

shows *qbs-l (qbs-l-inverse* $\mu) = \mu$

proof –

interpret *qbs-s-finite measure-to-qbs* M *from-real distr* μ *borel to-real*

by(*auto intro!: qbs-l-inverse-s-finite assms*)

show *?thesis*

using *distr-id'[OF assms(1),simplified sets-eq-imp-space-eq[OF assms(1)]]*

by(*auto simp: qbs-l qbs-l-inverse-def2[OF assms] distr-distr cong: distr-cong*)

qed

corollary *qbs-l-qbs-l-inverse-prob*:

sets $\mu = \text{sets } M \implies \text{prob-space } \mu \implies \text{qbs-l } (\text{qbs-l-inverse } \mu) = \mu$
by(*auto intro!*: *qbs-l-qbs-l-inverse prob-space.s-finite-measure-prob*)

lemma *qbs-l-inverse-qbs-l*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } (\text{measure-to-qbs } M))$

shows $\text{qbs-l-inverse } (\text{qbs-l } s) = s$

proof –

from *rep-qbs-space-monadM[OF assms]* **obtain** $\alpha \mu$ **where** *h*:

$s = \llbracket \text{measure-to-qbs } M, \alpha, \mu \rrbracket_{\text{sfin}} \text{qbs-s-finite } (\text{measure-to-qbs } M) \alpha \mu$

by *metis*

then interpret *qs:qbs-s-finite measure-to-qbs M* $\alpha \mu$ **by** *simp*

have [*simp*]: $\text{distr } \mu (\text{qbs-to-measure } (\text{measure-to-qbs } M)) \alpha = \text{distr } \mu M \alpha$

by(*simp cong: distr-cong*)

interpret *s: standard-borel-ne distr* $\mu M \alpha$

by(*rule standard-borel-ne-sets[of M]*) (*auto simp: standard-borel-ne-axioms*)

have [*measurable*]: $s.\text{from-real} \in \text{borel} \rightarrow_M M \alpha \in \mu \rightarrow_M M$

using *qs.alpha-measurable[simplified measurable-cong-sets[OF refl lr-sets-ident]]*

by(*auto simp: s.from-real-measurable[simplified measurable-cong-sets[OF refl sets-distr]]*)

interpret *pqs:pair-qbs-s-finite measure-to-qbs M s.from-real distr* $\mu \text{ borel } (s.\text{to-real} \circ \alpha) \alpha \mu$

by(*auto simp: pair-qbs-s-finite-def h*) (*auto simp: qbs-s-finite-def in-Mx-def qs.s-finite-measure-axioms qbs-s-finite-axioms-def qbs-Mx-R intro!: s-finite-measure.s-finite-measure-distr*)

show *?thesis*

by(*auto simp add: h(1) qs.qbs-l qbs-l-inverse-def distr-distr cong: measure-to-qbs-cong-sets intro!: pqs.qbs-s-finite-measure-eq*)

(*insert qbs-Mx-to-X[of - measure-to-qbs M], auto simp: comp-def qbs-space-R*)

qed

corollary *qbs-l-inverse-qbs-l-prob*:

assumes $s \in \text{qbs-space } (\text{monadP-qbs } (\text{measure-to-qbs } M))$

shows $\text{qbs-l-inverse } (\text{qbs-l } s) = s$

by(*auto intro!*: *qbs-l-inverse-qbs-l qbs-space-monadPM assms*)

lemma *s-finite-kernel-qbs-morphism*:

assumes *s-finite-kernel* $N M k$

shows $(\lambda x. \text{qbs-l-inverse } (k x)) \in \text{measure-to-qbs } N \rightarrow_Q \text{monadM-qbs } (\text{measure-to-qbs } M)$

proof –

interpret *sfin: s-finite-kernel* $N M k$ **by** *fact*

have $\llbracket \text{measure-to-qbs } M, \text{from-real}, \text{distr } (k x) \text{ borel to-real} \rrbracket_{\text{sfin}} = \text{qbs-l-inverse } (k x)$ **if** $x: x \in \text{space } N$ **for** x

proof –

note *sfin.kernel-sets[OF x,simp, measurable-cong]*

then interpret *skx: standard-borel-ne* $k x$

using *standard-borel-ne-axioms standard-borel-ne-sets* **by** *blast*

interpret *pqs: pair-qbs-s-finite measure-to-qbs M from-real distr* $(k x) \text{ borel to-real } \text{skx.from-real } \text{distr } (k x) \text{ borel } \text{skx.to-real}$

```

using skx.from-real-measurable[simplified measurable-cong-sets[OF refl sfin.kernel-sets[OF
x]]]
by(auto simp: pair-qbs-s-finite-def qbs-s-finite-def in-Mx-def qbs-Mx-R qbs-s-finite-axioms-def
sfin.image-s-finite-measure[OF x] intro!: s-finite-measure.s-finite-measure-distr)
show ?thesis
by(auto simp: qbs-l-inverse-def distr-distr cong: measure-to-qbs-cong-sets in-
tro!: pqs.qbs-s-finite-measure-eq) (auto intro!: distr-cong simp: sfin.kernel-space[OF
x])
qed
moreover have ( $\lambda x. \llbracket \text{measure-to-qbs } M, \text{from-real, distr } (k \ x) \text{ borel to-real} \rrbracket_{sfin}$ )
 $\in \text{measure-to-qbs } N \rightarrow_Q \text{monadM-qbs } (\text{measure-to-qbs } M)$ 
proof(rule qbs-morphismI)
fix  $\alpha :: \text{real} \Rightarrow -$ 
assume  $\alpha \in \text{qbs-Mx } (\text{measure-to-qbs } N)$ 
then have [measurable]:  $\alpha \in \text{borel} \rightarrow_M N$ 
by(simp add: qbs-Mx-R)
show ( $\lambda x. \llbracket \text{measure-to-qbs } M, \text{from-real, distr } (k \ x) \text{ borel to-real} \rrbracket_{sfin}$ )  $\circ \alpha \in$ 
 $\text{qbs-Mx } (\text{monadM-qbs } (\text{measure-to-qbs } M))$ 
by(auto simp: monadM-qbs-Mx qbs-Mx-R intro!: exI[where  $x = \text{from-real}$ ]
exI[where  $x = \lambda x. \text{distr } (k \ (\alpha \ x)) \text{ borel to-real}$ ] s-finite-kernel.comp-measurable[OF
sfin.distr-s-finite-kernel])
qed
ultimately show ?thesis
by(rule qbs-morphism-cong'[of measure-to-qbs N,simplified qbs-space-R])
qed

```

lemma *prob-kernel-qbs-morphism*:

```

assumes [measurable]:  $k \in N \rightarrow_M \text{prob-algebra } M$ 
shows ( $\lambda x. \text{qbs-l-inverse } (k \ x) \in \text{measure-to-qbs } N \rightarrow_Q \text{monadP-qbs } (\text{measure-to-qbs}$ 
 $M)$ )
proof(safe intro!: qbs-morphism-monadPI' s-finite-kernel-qbs-morphism prob-kernel.s-finite-kernel-prob-kernel)
fix  $x$ 
assume  $x \in \text{qbs-space } (\text{measure-to-qbs } N)$ 
then have  $x \in \text{space } N$  by(simp add: qbs-space-R)
from measurable-space[OF assms this]
have [measurable-cong, simp]:  $\text{sets } (k \ x) = \text{sets } M$  and  $p : \text{prob-space } (k \ x)$ 
by(auto simp: space-prob-algebra)
then interpret  $s : \text{standard-borel-ne } k \ x$ 
using standard-borel-ne-axioms standard-borel-ne-sets by blast
show  $\text{qbs-l-inverse } (k \ x) \in \text{qbs-space } (\text{monadP-qbs } (\text{measure-to-qbs } M))$ 
using s.qbs-l-inverse-in-space-monadP[OF refl p] by(simp cong: measure-to-qbs-cong-sets)
qed(simp add: prob-kernel-def)

```

lemma *qbs-l-inverse-return*:

```

assumes  $x \in \text{space } M$ 
shows  $\text{qbs-l-inverse } (\text{return } M \ x) = \text{return-qbs } (\text{measure-to-qbs } M) \ x$ 
proof -
interpret  $s : \text{standard-borel-ne } \text{return } M \ x$ 
by(rule standard-borel-ne-sets[of M]) (auto simp: standard-borel-ne-axioms)

```

show *?thesis*
using *s.qbs-l-inverse-in-space-monadP*[*OF refl prob-space-return*[*OF assms*]]
by(*auto intro!*: *inj-onD*[*OF qbs-l-inj-P*[*of measure-to-qbs M*]] *return-cong qbs-l-inverse-in-space-monadP*
qbs-morphism-space[*OF return-qbs-morphismP*[*of measure-to-qbs M*]] *assms simp*:
s.qbs-l-qbs-l-inverse-prob[*OF refl prob-space-return*[*OF assms*]] *qbs-l-return-qbs*[*of -*
M,simplified qbs-space-R,OF assms] *qbs-space-R cong: measure-to-qbs-cong-sets*)
qed

lemma *qbs-l-inverse-bind-kernel*:
assumes *standard-borel-ne N s-finite-measure M s-finite-kernel M N k*
shows *qbs-l-inverse (M \gg_k k) = qbs-l-inverse M \gg ($\lambda x. qbs-l-inverse (k x)$)*
(is ?lhs = ?rhs)
proof –
interpret *sfin: s-finite-kernel M N k* **by** *fact*
interpret *s: standard-borel-ne N* **by** *fact*
have *sets[simp,measurable-cong]:sets (M \gg_k k) = sets N*
by(*auto intro!*: *sets-bind-kernel*[*OF - space-ne*] *simp: sfin.kernel-sets*)
then interpret *s2: standard-borel-ne M \gg_k k*
using *s.standard-borel-ne-axioms standard-borel-ne-sets* **by** *blast*
have [*measurable*]: *s2.from-real \in borel \rightarrow_M N*
using *measurable-cong-sets s2.from-real-measurable sets* **by** *blast*
have *comp1:($\lambda x. qbs-l-inverse (k x)$) \circ from-real = ($\lambda r. \llbracket$ measure-to-qbs N,
s.from-real, distr (k (from-real r)) borel s.to-real \rrbracket_{sfin})*
proof
fix *r*
have *setskfr[measurable-cong, simp]: sets (k (from-real r)) = sets N*
by(*auto intro!*: *sfin.kernel-sets measurable-space*[*OF from-real-measurable*])
then interpret *s3: standard-borel-ne k (from-real r)*
using *s.standard-borel-ne-axioms standard-borel-ne-sets* **by** *blast*
have [*measurable*]: *s3.from-real \in borel \rightarrow_M N*
using *measurable-cong-sets s3.from-real-measurable setskfr* **by** *blast*
show (*($\lambda x. qbs-l-inverse (k x)$) \circ from-real*) *r = \llbracket measure-to-qbs N, s.from-real,
*distr (k (from-real r)) borel s.to-real \rrbracket_{sfin}**
by(*auto simp: qbs-l-inverse-def qbs-s-finite-eq-def qbs-s-finite-def in-Mx-def*
qbs-s-finite-axioms-def qbs-Mx-R distr-distr measurable-space[*OF from-real-measurable*]
cong: measure-to-qbs-cong-sets intro!: sfin.image-s-finite-measure s-finite-measure.s-finite-measure-distr
qbs-s-finite-measure-eq) (*auto intro!*: *distr-cong simp: sets-eq-imp-space-eq*[*OF set-*
skfr])
qed
have *?lhs = \llbracket measure-to-qbs (M \gg_k k), s2.from-real, distr (M \gg_k k) borel*
s2.to-real \rrbracket_{sfin}
by(*simp add: qbs-l-inverse-def*)
also have *... = \llbracket measure-to-qbs N, s.from-real, distr (M \gg_k k) borel s.to-real \rrbracket_{sfin}*
by(*auto cong: measure-to-qbs-cong-sets intro!: qbs-s-finite-measure-eq distr-cong*
s-finite-measure.s-finite-measure-distr sfin.comp-s-finite-measure assms(2) simp: qbs-s-finite-def
qbs-s-finite-def qbs-s-finite-axioms-def in-Mx-def qbs-Mx-R distr-distr sets-eq-imp-space-eq[*OF*
sets])
also have *... = \llbracket measure-to-qbs N, s.from-real, M \gg_k ($\lambda x. distr (k x) borel$*
s.to-real \rrbracket_{sfin})

```

    by(simp add: sfin.distr-bind-kernel[OF space-ne refl])
    also have ... = [[measure-to-qbs N, s.from-real, distr M borel to-real  $\ggg_k$  ( $\lambda r.$ 
distr (k (from-real r)) borel s.to-real)]sfin]
    proof -
      have M  $\ggg_k$  ( $\lambda x.$  distr (k x) borel s.to-real) = M  $\ggg_k$  ( $\lambda x.$  distr (k (from-real
(to-real x))) borel s.to-real)
      by(auto intro!: bind-kernel-cong-All)
      also have ... = distr M borel to-real  $\ggg_k$  ( $\lambda r.$  distr (k (from-real r)) borel
s.to-real)
      by(auto intro!: measure-kernel.bind-kernel-distr[symmetric,where Y=borel]
space-ne measure-kernel.distr-measure-kernel[where Y=N] sfin.measure-kernel-comp)
      finally show ?thesis by simp
    qed
    also have ... = ?rhs
    by(auto intro!: qbs-s-finite.bind-qbs[OF qbs-l-inverse-s-finite[OF refl assms(2)] -
s.s-finite-kernel-qbs-morphism[OF assms(3)] - - comp1,symmetric] s-finite-kernel.distr-s-finite-kernel[OF
sfin.comp-measurable] simp: qbs-Mx-R) (simp add: qbs-l-inverse-def)
    finally show ?thesis .
  qed

```

lemma *qbs-l-inverse-bind*:

```

    assumes standard-borel-ne N s-finite-measure M k  $\in$  M  $\rightarrow_M$  prob-algebra N
    shows qbs-l-inverse (M  $\ggg$  k) = qbs-l-inverse M  $\ggg$  ( $\lambda x.$  qbs-l-inverse (k x))
    by(auto simp: bind-kernel-bind[OF measurable-prob-algebraD[OF assms(3)],symmetric]
prob-kernel-def' intro!: qbs-l-inverse-bind-kernel assms prob-kernel.s-finite-kernel-prob-kernel)

```

end

4.1.14 PMF and SPMF

definition *qbs-pmf* \equiv ($\lambda p.$ qbs-l-inverse (measure-pmf p))

definition *qbs-mpmf* \equiv ($\lambda p.$ qbs-l-inverse (measure-mpmf p))

declare [[coercion *qbs-pmf*]]

lemma *qbs-pmf-qbsP*:

```

    fixes p :: (- :: countable) pmf
    shows qbs-pmf p  $\in$  qbs-space (monadP-qbs (count-spaceQ UNIV))
    by(auto simp: qbs-pmf-def measure-to-qbs-cong-sets[of count-space UNIV mea-
sure-pmf p,simplified] intro!: standard-borel-ne.qbs-l-inverse-in-space-monadP mea-
sure-pmf.prob-space-axioms)

```

lemma *qbs-pmf-qbs[qbs]*:

```

    fixes p :: (- :: countable) pmf
    shows qbs-pmf p  $\in$  qbs-space (monadM-qbs (count-spaceQ UNIV))
    by (simp add: qbs-pmf-qbsP qbs-space-monadPM)

```

lemma *qbs-mpmf-qbs[qbs]*:

```

    fixes q :: (- :: countable) spmf

```

shows $qbs\text{-}spmf\ q \in qbs\text{-}space\ (monadM\text{-}qbs\ (count\text{-}space_Q\ UNIV))$
by(*auto simp: qbs-spmf-def measure-to-qbs-cong-sets*[of *count-space UNIV measure-spmf q, simplified*] *intro!: standard-borel-ne.qbs-l-inverse-in-space-monadM subprob-space.s-finite-measure-subprob*)

lemma [*simp*]:

fixes $p :: (- :: countable)\ pmf$ **and** $q :: (- :: countable)\ spmf$
shows $qbs\text{-}l\text{-}qbs\text{-}pmf: qbs\text{-}l\ (qbs\text{-}pmf\ p) = measure\text{-}pmf\ p$
and $qbs\text{-}l\text{-}qbs\text{-}spmf: qbs\text{-}l\ (qbs\text{-}spmf\ q) = measure\text{-}spmf\ q$
by(*auto simp: qbs-pmf-def qbs-spmf-def intro!: standard-borel-ne.qbs-l-qbs-l-inverse subprob-space.s-finite-measure-subprob measure-pmf.subprob-space-axioms*)

lemma $qbs\text{-}pmf\text{-}return\text{-}pmf$:

fixes $x :: - :: countable$
shows $qbs\text{-}pmf\ (return\text{-}pmf\ x) = return\text{-}qbs\ (count\text{-}space_Q\ UNIV)\ x$
proof –
note $return\text{-}qbs\text{-}morphismP[qbs]$
show *?thesis*
by(*auto intro!: inj-onD[OF qbs-l-inj-P[where X=count-space_Q UNIV]] return-cong qbs-pmf-qbsP simp: qbs-l-return-qbs return-pmf.rep-eq*)
qed

lemma $qbs\text{-}pmf\text{-}bind\text{-}pmf$:

fixes $p :: ('a :: countable)\ pmf$ **and** $f :: 'a \Rightarrow ('b :: countable)\ pmf$
shows $qbs\text{-}pmf\ (p \gg f) = qbs\text{-}pmf\ p \gg (\lambda x. qbs\text{-}pmf\ (f\ x))$
by(*auto simp: measure-pmf-bind qbs-pmf-def space-prob-algebra measure-pmf.prob-space-axioms intro!: standard-borel-ne.qbs-l-inverse-bind[where N=count-space UNIV] prob-space.s-finite-measure-prob*)

lemma $qbs\text{-}pair\text{-}pmf$:

fixes $p :: ('a :: countable)\ pmf$ **and** $q :: ('b :: countable)\ pmf$
shows $qbs\text{-}pmf\ p \otimes_{Q_{mes}} qbs\text{-}pmf\ q = qbs\text{-}pmf\ (pair\text{-}pmf\ p\ q)$
proof(*rule inj-onD[OF qbs-l-inj-P[of count-space_Q UNIV \otimes_Q count-space_Q UNIV]]*)
show $qbs\text{-}l\ (qbs\text{-}pmf\ p \otimes_{Q_{mes}} qbs\text{-}pmf\ q) = qbs\text{-}l\ (qbs\text{-}pmf\ (pair\text{-}pmf\ p\ q))$
by(*simp add: measure-pair-pmf qbs-l-qbs-pair-measure[OF standard-borel-ne.standard-borel standard-borel-ne.standard-borel, of count-space UNIV count-space UNIV]*)
next
note $[qbs] = qbs\text{-}pmf\text{-}qbsP\ qbs\text{-}pair\text{-}measure\text{-}morphismP$
show $qbs\text{-}pmf\ p \otimes_{Q_{mes}} qbs\text{-}pmf\ q \in qbs\text{-}space\ (monadP\text{-}qbs\ (count\text{-}space_Q\ UNIV\ \otimes_Q\ count\text{-}space_Q\ UNIV))$ $qbs\text{-}pmf\ (pair\text{-}pmf\ p\ q) \in qbs\text{-}space\ (monadP\text{-}qbs\ (count\text{-}space_Q\ UNIV\ \otimes_Q\ count\text{-}space_Q\ UNIV))$
by *auto (simp add: qbs-count-space-prod)*
qed

4.1.15 Density

lift-definition $density\text{-}qbs :: ['a\ qbs\text{-}measure, 'a \Rightarrow ennreal] \Rightarrow 'a\ qbs\text{-}measure$
is $\lambda(X, \alpha, \mu)\ f.$ *if* $f \in X \rightarrow_Q\ qbs\text{-}borel$ *then* $(X, \alpha, density\ \mu\ (f \circ \alpha))$ *else* $(X, SOME\ a.\ a \in qbs\text{-}Mx\ X, null\text{-}measure\ borel)$
proof *safe*

```

fix  $X Y :: 'a$  quasi-borel
fix  $\alpha \beta \mu \nu$  and  $f :: - \Rightarrow$  ennreal
assume  $1:qbs-s-finite-eq (X, \alpha, \mu) (Y, \beta, \nu)$ 
then interpret  $qs: pair-qbs-s-finite X \alpha \mu \beta \nu$ 
  using  $qbs-s-finite-eq-dest[OF 1]$  by(simp add: pair-qbs-s-finite-def)
  have [simp]:(SOME a.  $a \in qbs-Mx X \in qbs-Mx X$  (SOME a.  $a \in qbs-Mx Y \in$ 
 $qbs-Mx X$ 
    using  $qs.pq1.in-Mx$  by(simp-all only: some-in-eq qbs-s-finite-eq-dest[OF 1])
blast+
  {
    assume  $f \in X \rightarrow_Q qbs-borel$ 
    then have  $qbs-s-finite-eq (X, \alpha, density \mu (f \circ \alpha)) (Y, \beta, density \nu (f \circ \beta))$ 
    by(auto simp: qbs-s-finite-eq-def lr-adjunction-correspondence density-distr[symmetric]
comp-def qbs-s-finite-eq-dest[OF 1] qbs-s-finite-def in-Mx-def qbs-s-finite-axioms-def
qs.pq1.mu-sets qs.pq2.mu-sets AE-distr-iff intro!: qs.pq1.s-finite-measure-density qs.pq2.s-finite-measure-densit
  }
  moreover have  $f \in X \rightarrow_Q qbs-borel \implies f \notin Y \rightarrow_Q qbs-borel \implies qbs-s-finite-eq$ 
 $(X, \alpha, density \mu (f \circ \alpha)) (Y, (SOME a. a \in qbs-Mx Y), null-measure borel)$ 
     $f \notin X \rightarrow_Q qbs-borel \implies f \in Y \rightarrow_Q qbs-borel \implies qbs-s-finite-eq (X, (SOME$ 
 $a. a \in qbs-Mx X), null-measure borel) (Y, \beta, density \nu (f \circ \beta))$ 
     $f \notin X \rightarrow_Q qbs-borel \implies f \notin Y \rightarrow_Q qbs-borel \implies qbs-s-finite-eq (X, (SOME$ 
 $a. a \in qbs-Mx X), null-measure borel) (Y, (SOME a. a \in qbs-Mx Y), null-measure$ 
 $borel)$ 
    by(auto simp: qbs-s-finite-eq-dest[OF 1] qbs-s-finite-eq-def qbs-s-finite-def in-Mx-def
qbs-s-finite-axioms-def distr-return null-measure-distr intro!: subprob-space.s-finite-measure-subprob
subprob-spaceI)
  ultimately show  $qbs-s-finite-eq$  (if  $f \in X \rightarrow_Q borel_Q$  then  $(X, \alpha, density \mu (f \circ$ 
 $\alpha))$  else  $(X, SOME aa. aa \in qbs-Mx X, null-measure borel)$ ) (if  $f \in Y \rightarrow_Q borel_Q$ 
then  $(Y, \beta, density \nu (f \circ \beta))$  else  $(Y, SOME a. a \in qbs-Mx Y, null-measure$ 
 $borel)$ )
    by auto
qed

```

lemma(in $qbs-s-finite$)

```

assumes  $f \in X \rightarrow_Q qbs-borel$ 
shows  $density-qbs:density-qbs \llbracket X, \alpha, \mu \rrbracket_{sfin} f = \llbracket X, \alpha, density \mu (f \circ \alpha) \rrbracket_{sfin}$ 
  and  $density-qbs-s-finite: qbs-s-finite X \alpha (density \mu (f \circ \alpha))$ 
using  $assms$  by(auto simp: density-qbs.abs-eq qbs-s-finite-def in-Mx-def lr-adjunction-correspondence
qbs-s-finite-axioms-def mu-sets AE-distr-iff intro!: s-finite-measure-density)

```

lemma $density-qbs-density-qbs-eq$:

```

assumes [qbs]: $s \in qbs-space (monadM-qbs X) f \in X \rightarrow_Q qbs-borel g \in X \rightarrow_Q$ 
 $qbs-borel$ 
shows  $density-qbs (density-qbs s f) g = density-qbs s (\lambda x. f x * g x)$ 
proof –
from  $rep-qbs-space-monadM[OF assms(1)]$  obtain  $\alpha \mu$ 
  where  $hs: s = \llbracket X, \alpha, \mu \rrbracket_{sfin} qbs-s-finite X \alpha \mu$  by  $metis$ 
then interpret  $qbs-s-finite X \alpha \mu$  by  $simp$ 
show  $?thesis$ 

```

using *assms(2,3)* **by**(*simp add: hs(1) density-qbs[OF assms(2)] qbs-s-finite.density-qbs[OF density-qbs-s-finite[OF assms(2)] assms(3)] density-qbs lr-adjunction-correspondence density-density-eq*) (*simp add: comp-def*)
qed

lemma *qbs-l-density-qbs*:

assumes $s \in \text{qbs-space } (\text{monadM-qbs } X) f \in X \rightarrow_Q \text{qbs-borel}$
shows $\text{qbs-l } (\text{density-qbs } s f) = \text{density } (\text{qbs-l } s) f$

proof –

from *rep-qbs-space-monadM[OF assms(1)]*
obtain $\alpha \mu$ **where** $s = \llbracket X, \alpha, \mu \rrbracket_{\text{sf in qbs-s-finite } X} \alpha \mu$
by *metis*
then interpret *qbs-s-finite* $X \alpha \mu$ **by** *simp*
show *?thesis*

using *assms(2)* **by**(*simp add: s(1) qbs-l qbs-s-finite.density-qbs[OF s(2) assms(2)] qbs-s-finite.qbs-l[OF qbs-s-finite.density-qbs-s-finite[OF s(2) assms(2)]] density-distr lr-adjunction-correspondence*) (*simp add: comp-def*)
qed

corollary *qbs-l-density-qbs-indicator*:

assumes $[\text{qbs}]: s \in \text{qbs-space } (\text{monadM-qbs } X) \text{qbs-pred } X P$
shows $\text{qbs-l } (\text{density-qbs } s (\text{indicator } \{x \in \text{qbs-space } X. P x\})) (\text{qbs-space } X) = \text{qbs-l } s \{x \in \text{qbs-space } X. P x\}$

proof –

have $1[\text{measurable}]: \{x \in \text{qbs-space } X. P x\} \in \text{sets } (\text{qbs-to-measure } X)$
by (*metis qbs-pred-iff-sets space-L assms(2)*)
have $2[\text{qbs}]: \text{indicator } \{x \in \text{qbs-space } X. P x\} \in X \rightarrow_Q \text{qbs-borel}$
by(*rule indicator-qbs-morphism''*) *qbs*
show *?thesis*
using *assms(2)* **by**(*auto simp: qbs-l-density-qbs[of - X] emeasure-density[of indicator {x ∈ space (qbs-to-measure X). P x} qbs-l s, OF - sets.top, simplified measurable-qbs-l'[OF assms(1)], OF borel-measurable-indicator[OF predE], simplified space-L space-qbs-l-in[OF assms(1)]] qbs-pred-iff-measurable-pred nn-set-integral-space[of qbs-l s, simplified space-qbs-l-in[OF assms(1)]] nn-integral-indicator[of - qbs-l s, simplified sets-qbs-l[OF assms(1)]]*)

qed

lemma *qbs-nn-integral-density-qbs*:

assumes $[\text{qbs}]: s \in \text{qbs-space } (\text{monadM-qbs } X) f \in X \rightarrow_Q \text{qbs-borel } g \in X \rightarrow_Q \text{qbs-borel}$
shows $(\int^+_Q x. g x \partial(\text{density-qbs } s f)) = (\int^+_Q x. f x * g x \partial s)$
by(*auto simp: qbs-nn-integral-def2-l qbs-l-density-qbs[of - X] measurable-qbs-l'[OF assms(1)] lr-adjunction-correspondence[symmetric] intro!: nn-integral-density*)

lemma *qbs-integral-density-qbs*:

fixes $g :: 'a \Rightarrow 'b :: \{\text{banach, second-countable-topology}\}$ **and** $f :: 'a \Rightarrow \text{real}$
assumes $[\text{qbs}]: s \in \text{qbs-space } (\text{monadM-qbs } X) f \in X \rightarrow_Q \text{qbs-borel } g \in X \rightarrow_Q \text{qbs-borel}$
and $A E_Q x \text{ in } s. f x \geq 0$

shows $(\int_Q x. g x \partial(\text{density-qbs } s f)) = (\int_Q x. f x *_R g x \partial s)$
using *assms(4)* **by** (*auto simp: qbs-integral-def2-l qbs-l-density-qbs[of - X] measurable-qbs-l'[OF assms(1)] lr-adjunction-correspondence[symmetric] AEq-qbs-l intro!: integral-density*)

lemma *density-qbs-morphism[qbs]: density-qbs \in monadM-qbs $X \rightarrow_Q (X \Rightarrow_Q \text{qbs-borel}) \Rightarrow_Q \text{monadM-qbs } X$*

proof (*rule curry-preserves-morphisms[OF pair-qbs-morphismI]*)

fix γ **and** $\beta :: - \Rightarrow - \Rightarrow \text{ennreal}$

assume $h: \gamma \in \text{qbs-Mx } (\text{monadM-qbs } X) \quad \beta \in \text{qbs-Mx } (X \Rightarrow_Q \text{qbs-borel})$

hence $[\text{qbs}]: \gamma \in \text{qbs-borel} \rightarrow_Q \text{monadM-qbs } X \quad \beta \in \text{qbs-borel} \rightarrow_Q X \Rightarrow_Q \text{qbs-borel}$

by (*simp-all add: qbs-Mx-is-morphisms*)

from *rep-qbs-Mx-monadM[OF h(1)]* **obtain** α k **where** $hk:$

$\gamma = (\lambda r. \llbracket X, \alpha, k r \rrbracket_{sfin}) \quad \alpha \in \text{qbs-Mx } X \text{ s-finite-kernel borel borel } k \wedge r. \text{qbs-s-finite } X \alpha (k r)$

by *metis*

then interpret $a: \text{in-Mx } X \alpha$ **by** (*simp add: in-Mx-def*)

have [*measurable*]: $(\lambda(x, y). \beta x (\alpha y)) \in \text{borel-measurable } (\text{borel} \otimes_M \text{borel})$

proof –

have $(\lambda(x, y). \beta x (\alpha y)) \in \text{qbs-borel} \otimes_Q \text{qbs-borel} \rightarrow_Q \text{qbs-borel}$

by *simp*

thus *?thesis*

by (*simp add: lr-adjunction-correspondence qbs-borel-prod borel-prod*)

qed

have [*simp*]: $\text{density-qbs } (\gamma r) (\beta r) = \llbracket X, \alpha, \text{density } (k r) (\beta r \circ \alpha) \rrbracket_{sfin}$ **for** r

using $hk(4)$ **by** (*auto simp add: hk(1) density-qbs.abs-eq[OF qbs-s-finite.qbs-s-finite-eq-refl[OF hk(4)]]*)

show $(\lambda r. \text{density-qbs } (\text{fst } (\gamma r, \beta r)) (\text{snd } (\gamma r, \beta r))) \in \text{qbs-Mx } (\text{monadM-qbs } X)$

by (*auto simp: monadM-qbs-Mx comp-def intro!: exI[where x= α] exI[where x= $\lambda r. \text{density } (k r) (\beta r \circ \alpha)$] s-finite-kernel.density-s-finite-kernel[OF hk(3)]*)

qed

lemma *density-qbs-cong-AE:*

assumes [*qbs*]: $s \in \text{qbs-space } (\text{monadM-qbs } X) \quad f \in X \rightarrow_Q \text{qbs-borel} \quad g \in X \rightarrow_Q \text{qbs-borel}$

and $AE_Q x \text{ in } s. f x = g x$

shows $\text{density-qbs } s f = \text{density-qbs } s g$

proof (*rule inj-onD[OF qbs-l-inj[of X]]*)

show $\text{qbs-l } (\text{density-qbs } s f) = \text{qbs-l } (\text{density-qbs } s g)$

using *assms(4)* **by** (*auto simp: qbs-l-density-qbs[of - X] measurable-qbs-l'[OF assms(1)] AEq-qbs-l lr-adjunction-correspondence[symmetric] intro!: density-cong*)

qed *simp-all*

corollary *density-qbs-cong:*

assumes [*qbs*]: $s \in \text{qbs-space } (\text{monadM-qbs } X) \quad f \in X \rightarrow_Q \text{qbs-borel} \quad g \in X \rightarrow_Q \text{qbs-borel}$

and $\bigwedge x. x \in \text{qbs-space } X \implies f x = g x$

shows $\text{density-qbs } s f = \text{density-qbs } s g$

by(*auto intro!*: *density-qbs-cong-AE*[*of - X*] *AEq-I2*[*of - X*] *assms*(4))

lemma *density-qbs-1*[*simp*]: *density-qbs s* ($\lambda x. 1$) = *s*

proof –

obtain *X* **where** *s*[*qbs*]: *s* \in *qbs-space* (*monadM-qbs X*)

using *in-qbs-space-of* **by** *blast*

show *?thesis*

by(*auto intro!*: *inj-onD*[*OF qbs-l-inj - - s*] *simp*: *qbs-l-density-qbs*[*of - X*] *density-1*)

qed

lemma *pair-density-qbs*:

assumes [*qbs*]: *p* \in *qbs-space* (*monadM-qbs X*) *q* \in *qbs-space* (*monadM-qbs Y*)

and [*qbs*]: *f* \in *X* \rightarrow_Q *qbs-borel* *g* \in *Y* \rightarrow_Q *qbs-borel*

shows *density-qbs p f* \otimes_{Qmes} *density-qbs q g* = *density-qbs* (*p* \otimes_{Qmes} *q*) ($\lambda(x,y). f x * g y$)

proof(*safe intro!*: *qbs-measure-eqI*[*of - X* \otimes_Q *Y*])

fix *h* :: - \Rightarrow *ennreal*

assume *h*[*qbs*]: *h* \in *X* \otimes_Q *Y* \rightarrow_Q *borel_Q*

show ($\int^+_Q z. h z \partial(\text{density-qbs } p f \otimes_{Qmes} \text{density-qbs } q g)$) = ($\int^+_Q z. h z \partial(\text{density-qbs } (p \otimes_{Qmes} q) (\lambda(x,y). f x * g y))$) (**is** *?lhs = ?rhs*)

proof –

have *?lhs* = ($\int^+_Q x. \int^+_Q y. h (x, y) \partial \text{density-qbs } q g \partial \text{density-qbs } p f$)

by(*simp add*: *qbs-nn-integral-Fubini-fst*[*of - X - Y*])

also have ... = ($\int^+_Q x. \int^+_Q y. g y * h (x, y) \partial q \partial \text{density-qbs } p f$)

by(*auto intro!*: *qbs-nn-integral-cong*[*of - X*] *simp*: *qbs-nn-integral-density-qbs*[*of - Y*])

also have ... = *?rhs*

by(*auto simp add*: *qbs-nn-integral-density-qbs*[*of - X*] *qbs-nn-integral-density-qbs*[*of - X* \otimes_Q *Y*] *split-beta'* *qbs-nn-integral-Fubini-fst*[*of - X - Y, symmetric*] *qbs-nn-integral-cmult*[*of - Y*] *mult.assoc intro!*: *qbs-nn-integral-cong*[*of - X*])

finally show *?thesis* .

qed

qed *simp-all*

4.1.16 Normalization

definition *normalize-qbs* :: 'a *qbs-measure* \Rightarrow 'a *qbs-measure* **where**

normalize-qbs s \equiv (*let* *X* = *qbs-space-of s*;

r = *qbs-l s* (*qbs-space X*) *in*

if *r* $\neq 0 \wedge r \neq \infty$ *then* *density-qbs s* ($\lambda x. 1 / r$)

else *qbs-null-measure X*)

lemma

assumes *s* \in *qbs-space* (*monadM-qbs X*)

shows *normalize-qbs*: *qbs-l s* (*qbs-space X*) $\neq 0 \implies$ *qbs-l s* (*qbs-space X*) $\neq \infty \implies$ *normalize-qbs s* = *density-qbs s* ($\lambda x. 1 / \text{emeasure } (qbs-l s) (qbs-space X)$)

and *normalize-qbs0*: *qbs-l s* (*qbs-space X*) = 0 \implies *normalize-qbs s* = *qbs-null-measure X*

and *normalize-qbsinfy*: $qbs-l\ s\ (qbs-space\ X) = \infty \implies normalize-qbs\ s = qbs-null-measure\ X$

by(*auto simp: qbs-space-of-in[OF assms(1)] normalize-qbs-def*)

lemma *normalize-qbs-prob*:

assumes $s \in qbs-space\ (monadM-qbs\ X)$ $qbs-l\ s\ (qbs-space\ X) \neq 0$ $qbs-l\ s\ (qbs-space\ X) \neq \infty$

shows $normalize-qbs\ s \in qbs-space\ (monadP-qbs\ X)$

unfolding *normalize-qbs[OF assms]*

proof –

obtain $\alpha\ \mu$

where $hs: s = \llbracket X, \alpha, \mu \rrbracket_{sfin}\ qbs-s-finite\ X\ \alpha\ \mu$

using *rep-qbs-space-monadM assms(1)* **by** *meson*

interpret $qs: qbs-s-finite\ X\ \alpha\ \mu$ **by** *fact*

have $density-qbs\ s\ (\lambda x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X)) = density-qbs\ \llbracket X, \alpha, \mu \rrbracket_{sfin}\ (\lambda x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X))$

by(*simp add: hs*)

also **have** $\dots \in qbs-space\ (monadP-qbs\ X)$

by(*auto simp add: qs.density-qbs monadP-qbs-space qbs-s-finite.qbs-l[OF qs.density-qbs-s-finite,of $\lambda x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X)$,simplified] qbs-s-finite.qbs-space-of[OF qs.density-qbs-s-finite,of $\lambda x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X)$,simplified] intro!: prob-space.prob-space-distr, auto intro!: prob-spaceI simp: emeasure-density*)

(*insert assms(2,3),auto simp: hs qs.qbs-l emeasure-distr emeasure-distr[of - - qbs-to-measure\ X,OF - sets.top,simplified space-L] divide-eq-1-ennreal ennreal-divide-times*)

finally **show** $density-qbs\ s\ (\lambda x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X)) \in qbs-space\ (monadP-qbs\ X)$.

qed

lemma *normalize-qbs-morphism[qbs]*: $normalize-qbs \in monadM-qbs\ X \rightarrow_Q monadM-qbs\ X$

proof –

have (*if emeasure (qbs-l s) (qbs-space X) $\neq 0 \wedge emeasure (qbs-l s) (qbs-space X) \neq \infty$ then density-qbs s ($\lambda x. 1 / emeasure (qbs-l s) (qbs-space X)$) else qbs-null-measure X*) = *normalize-qbs s* (**is** *?f s = -*) **if** $s:s \in qbs-space\ (monadM-qbs\ X)$ **for** s

by(*simp add: qbs-space-of-in[OF s] normalize-qbs-def*)

moreover **have** ($\lambda s. ?f\ s$) $\in monadM-qbs\ X \rightarrow_Q monadM-qbs\ X$

proof(*cases qbs-space X = {}*)

case *True*

then **show** *?thesis*

by(*simp add: qbs-morphism-from-empty monadM-qbs-empty-iff[of X]*)

next

case $X:False$

have $[qbs]:(\lambda s. emeasure\ (qbs-l\ s)\ (qbs-space\ X)) \in monadM-qbs\ X \rightarrow_Q qbs-borel$

by(*rule qbs-l-morphism[OF sets.top[of qbs-to-measure\ X,simplified space-L]]*)

have $[qbs]: qbs-null-measure\ X \in qbs-space\ (monadM-qbs\ X)$

by(*auto intro!: qbs-null-measure-in-Mx X*)

have $[qbs]: (\lambda s\ x. 1 / emeasure\ (qbs-l\ s)\ (qbs-space\ X)) \in monadM-qbs\ X \rightarrow_Q X \Rightarrow_Q qbs-borel$

by(*rule arg-swap-morphism*) *simp*

```

  show ?thesis
  by qbs
qed
ultimately show ?thesis
  by(simp cong: qbs-morphism-cong)
qed

```

lemma *normalize-qbs-morphismP*:

```

  assumes [qbs]:s ∈ X →Q monadM-qbs Y
  and ∧x. x ∈ qbs-space X ⇒ qbs-l (s x) (qbs-space Y) ≠ 0 ∧x. x ∈ qbs-space
X ⇒ qbs-l (s x) (qbs-space Y) ≠ ∞
  shows (λx. normalize-qbs (s x)) ∈ X →Q monadP-qbs Y
  by(rule qbs-morphism-monadPI'[OF normalize-qbs-prob]) (use assms(2,3) qbs-morphism-space[OF
assms(1)]) in auto

```

lemma *normalize-qbs-monadP-ident*:

```

  assumes s ∈ qbs-space (monadP-qbs X)
  shows normalize-qbs s = s
  using normalize-qbs[OF qbs-space-monadPM[OF assms]] prob-space.emmeasure-space-1[OF
qbs-l-prob-space[OF assms]]
  by(auto simp: space-qbs-l-in[OF qbs-space-monadPM[OF assms]] intro!: inj-onD[OF
qbs-l-inj-P - - assms])

```

corollary *normalize-qbs-idenpotent*: $normalize-qbs (normalize-qbs s) = normalize-qbs s$

proof –

```

  obtain X where s[qbs]: s ∈ qbs-space (monadM-qbs X)
  using in-qbs-space-of by blast
  then have X: qbs-space X ≠ {}
  by (metis qbs-s-space-of-not-empty qbs-space-of-in)
  then obtain x where x:x ∈ qbs-space X by auto
  consider qbs-l s (qbs-space X) = 0 | qbs-l s (qbs-space X) = ⊤ | qbs-l s (qbs-space
X) ≠ 0 qbs-l s (qbs-space X) ≠ ⊤
  by auto
  then show ?thesis
  proof cases
  case 1
  then show ?thesis
  using normalize-qbs0[OF qbs-null-measure-in-Mx[OF X]]
  by(simp add: normalize-qbs0[OF s] qbs-null-measure-null-measure[OF X])
  next
  case 2
  then show ?thesis
  using normalize-qbs0[OF qbs-null-measure-in-Mx[OF X]]
  by(simp add: normalize-qbsinfy[OF s] qbs-null-measure-null-measure[OF X])
  next
  case 3
  have normalize-qbs s ∈ qbs-space (monadP-qbs X)
  by(rule qbs-morphism-space[OF normalize-qbs-morphismP[of λx. s],of X X])

```

$x]$ (*auto simp: 3 x*)
then show *?thesis*
by(*simp add: normalize-qbs-monadP-ident*)
qed
qed

4.1.17 Product Measures

definition *PiQ-measure* :: [*'a set, 'a \Rightarrow 'b qbs-measure*] \Rightarrow (*'a \Rightarrow 'b*) *qbs-measure*
where

PiQ-measure \equiv (λI *si. if* ($\forall i \in I. \exists Mi. \text{standard-borel-ne } Mi \wedge si \ i \in \text{qbs-space}$
(monadM-qbs (measure-to-qbs Mi)))
then if countable I \wedge ($\forall i \in I. \text{prob-space (qbs-l (si i))}$) then
qbs-l-inverse ($\prod_M i \in I. \text{qbs-l (si i)}$)
else if finite I \wedge ($\forall i \in I. \text{sigma-finite-measure (qbs-l (si i))}$)
then qbs-l-inverse ($\prod_M i \in I. \text{qbs-l (si i)}$)
else qbs-null-measure ($\prod_Q i \in I. \text{qbs-space-of (si i)}$)
else qbs-null-measure ($\prod_Q i \in I. \text{qbs-space-of (si i)}$))

syntax

-PiQ-measure :: *pttrn \Rightarrow 'i set \Rightarrow 'a qbs-measure \Rightarrow ('i \Rightarrow 'a) qbs-measure*
(($\exists \Pi_{Qmeas} \text{-}\in\text{-}/ \text{-}$) 10)

translations

$\Pi_{Qmeas} \ x \in I. X == \text{CONST } \text{PiQ-measure } I \ (\lambda x. X)$

context

fixes *I and Mi*

assumes *standard-borel-ne: $\bigwedge i. i \in I \Longrightarrow \text{standard-borel-ne (Mi i)}$*

begin

context

assumes *countableI: countable I*

begin

interpretation *sb: standard-borel-ne $\prod_M i \in I. (\text{borel} :: \text{real measure})$*

by (*simp add: countableI product-standard-borel-ne*)

interpretation *sbM: standard-borel-ne $\prod_M i \in I. Mi \ i$*

by (*simp add: countableI standard-borel-ne product-standard-borel-ne*)

lemma

assumes $\bigwedge i. i \in I \Longrightarrow si \ i \in \text{qbs-space (monadP-qbs (measure-to-qbs (Mi i)))}$

and $\bigwedge i. i \in I \Longrightarrow si \ i = \llbracket \text{measure-to-qbs (Mi i), } \alpha \ i, \mu \ i \rrbracket_{\text{sf in}} \bigwedge i. i \in I \Longrightarrow$
qbs-prob (measure-to-qbs (Mi i)) ($\alpha \ i$) ($\mu \ i$)

shows *PiQ-measure-prob-eq: ($\prod_{Qmeas} i \in I. si \ i$) = $\llbracket \text{measure-to-qbs ($\prod_M i \in I. Mi \ i$), sbM.from-real, distr ($\prod_M i \in I. \text{qbs-l (si i)}$) borel sbM.to-real} \rrbracket_{\text{sf in}}$ (**is** - =
*?rhs)**

and *PiQ-measure-qbs-prob: qbs-prob (measure-to-qbs ($\prod_M i \in I. Mi \ i$)) sbM.from-real*
*(distr ($\prod_M i \in I. \text{qbs-l (si i)}$) borel sbM.to-real) (**is** ?qbsprob)*

proof –
have [*measurable-cong,simp*]: *prob-space* ($\prod_M i \in I. \text{qbs-l } (si \ i)$) *sets* ($\prod_M i \in I. \text{qbs-l } (si \ i) = \text{sets } (\prod_M i \in I. Mi \ i)$)
using *sets-qbs-l*[*OF assms(1)*][*THEN qbs-space-monadPM*] *standard-borel.lr-sets-ident*[*OF standard-borel-ne.standard-borel*][*OF standard-borel-ne*]
by(*auto cong: sets-PiM-cong intro!*: *prob-space-PiM qbs-l-prob-space assms(1)*)
show ?*qbsprob*
by(*auto simp: pair-qbs-s-finite-def intro!*: *qbs-prob.qbs-s-finite sbM.qbs-l-inverse-qbs-prob*)
have ($\prod_{Qmeas} i \in I. si \ i$) = *qbs-l-inverse* ($\prod_M i \in I. \text{qbs-l } (si \ i)$)
using *countableI assms(1)*[*THEN qbs-space-monadPM*] *qbs-l-prob-space*[*OF assms(1)*] *standard-borel-ne* **by**(*auto simp: PiQ-measure-def*)
also have ... = ?*rhs*
by(*auto intro!*: *sbM.qbs-l-inverse-def2 prob-space.s-finite-measure-prob cong: sets-PiM-cong*[*OF refl*])
finally show ($\prod_{Qmeas} i \in I. si \ i$) = ?*rhs* .
qed

lemma *qbs-l-PiQ-measure-prob*:

assumes $\bigwedge i. i \in I \implies si \ i \in \text{qbs-space } (\text{monadP-qbs } (\text{measure-to-qbs } (Mi \ i)))$
shows *qbs-l* ($\prod_{Qmeas} i \in I. si \ i$) = ($\prod_M i \in I. \text{qbs-l } (si \ i)$)

proof –
have *qbs-l* ($\prod_{Qmeas} i \in I. si \ i$) = *qbs-l* (*qbs-l-inverse* ($\prod_M i \in I. \text{qbs-l } (si \ i)$))
using *countableI assms(1)*[*THEN qbs-space-monadPM*] *qbs-l-prob-space*[*OF assms(1)*] *standard-borel-ne* **by**(*auto simp: PiQ-measure-def*)
also have ... = ($\prod_M i \in I. \text{qbs-l } (si \ i)$)
using *sets-qbs-l*[*OF assms(1)*][*THEN qbs-space-monadPM*] *standard-borel.lr-sets-ident*[*OF standard-borel-ne.standard-borel*][*OF standard-borel-ne*]
by(*auto intro!*: *sbM.qbs-l-qbs-l-inverse-prob prob-space-PiM qbs-l-prob-space*[*OF assms(1)*] *cong: sets-PiM-cong*)
finally show ?*thesis* .
qed

end

context

assumes *finI: finite I*

begin

interpretation *sb:standard-borel-ne* $\prod_M i \in I. (\text{borel} :: \text{real measure})$

by (*simp add: finI product-standard-borel-ne countable-finite*)

interpretation *sbM: standard-borel-ne* $\prod_M i \in I. Mi \ i$

by (*simp add: countable-finite finI standard-borel-ne product-standard-borel-ne*)

lemma *qbs-l-PiQ-measure*:

assumes $\bigwedge i. i \in I \implies si \ i \in \text{qbs-space } (\text{monadM-qbs } (\text{measure-to-qbs } (Mi \ i)))$

and $\bigwedge i. i \in I \implies \text{sigma-finite-measure } (\text{qbs-l } (si \ i))$

shows *qbs-l* ($\prod_{Qmeas} i \in I. si \ i$) = ($\prod_M i \in I. \text{qbs-l } (si \ i)$)

proof –

```

have [simp]: s-finite-measure (ΠM i∈I. qbs-l (si i))
proof –
  have (ΠM i∈I. qbs-l (si i)) = (ΠM i∈I. if i ∈ I then qbs-l (si i) else null-measure
(count-space UNIV))
    by(simp cong: PiM-cong)
  also have s-finite-measure ...
    by(auto intro!: sigma-finite-measure.s-finite-measure product-sigma-finite.sigma-finite
finI simp: product-sigma-finite-def assms(2)) (auto intro!: finite-measure.sigma-finite-measure
finite-measureI)
  finally show ?thesis .
qed
have qbs-l (ΠQmeas i∈I. si i) = qbs-l (qbs-l-inverse (ΠM i∈I. qbs-l (si i)))
  using finI assms(1) assms(2) standard-borel-ne by(fastforce simp: PiQ-measure-def)
also have ... = (ΠM i∈I. qbs-l (si i))
  using sets-qbs-l[OF assms(1)] standard-borel.lr-sets-ident[OF standard-borel-ne.standard-borel[OF
standard-borel-ne]]
  by(auto intro!: sbM.qbs-l-qbs-l-inverse prob-space-PiM cong: sets-PiM-cong)
finally show ?thesis .
qed

```

end

end

4.2 Measures

4.2.1 The Lebesgue Measure

definition *lborel-qbs* (*lborel_Q*) **where** *lborel-qbs* ≡ *qbs-l-inverse lborel*

lemma *lborel-qbs-qbs*[*qbs*]: *lborel-qbs* ∈ *qbs-space (monadM-qbs qbs-borel)*

by(auto simp: *lborel-qbs-def measure-to-qbs-cong-sets*[OF *sets-lborel,symmetric*]
intro!: *standard-borel-ne.qbs-l-inverse-in-space-monadM lborel.s-finite-measure-axioms*)

lemma *qbs-l-lborel-qbs*[*simp*]: *qbs-l lborel_Q* = *lborel*

by(auto intro!: *standard-borel-ne.qbs-l-qbs-l-inverse lborel.s-finite-measure-axioms*
simp: lborel-qbs-def)

corollary

shows *qbs-integral-lborel*: (∫_Q x. f x ∂*lborel-qbs*) = (∫ x. f x ∂*lborel*)
and *qbs-nn-integral-lborel*: (∫⁺_Q x. f x ∂*lborel-qbs*) = (∫⁺ x. f x ∂*lborel*)
by(*simp-all add: qbs-integral-def2-l qbs-nn-integral-def2-l*)

lemma(in *standard-borel-ne*) *measure-with-args-morphism*:

assumes *s-finite-kernel X M k*

shows *qbs-l-inverse* ∘ *k* ∈ *measure-to-qbs X* →_Q *monadM-qbs (measure-to-qbs M)*

proof(*safe intro!: qbs-morphismI*)

```

fix  $\alpha :: \text{real} \Rightarrow -$ 
assume  $\alpha \in \text{qbs-Mx}$  (measure-to-qbs X)
then have  $h[\text{measurable}]: \alpha \in \text{borel} \rightarrow_M X$ 
  by (simp add: qbs-Mx-R)
interpret  $s: s\text{-finite-kernel } X \ M \ k$  by fact
have  $1: \bigwedge r. \text{sets } (k (\alpha r)) = \text{sets } M \ \bigwedge r. s\text{-finite-measure } (k (\alpha r))$ 
  using measurable-space[OF h] s.kernel-sets by (auto intro!: s.image-s-finite-measure)
show  $\text{qbs-l-inverse} \circ k \circ \alpha \in \text{qbs-Mx}$  (monadM-qbs (measure-to-qbs M))
  by (auto intro!: exI[where x=from-real] exI[where x=( $\lambda r. \text{distr } (k (\alpha r)) \text{ borel to-real}$ )] s-finite-kernel.comp-measurable[OF s-finite-kernel.distr-s-finite-kernel[OF assms]] simp: monadM-qbs-Mx qbs-Mx-R qbs-l-inverse-def2[OF 1] comp-def)
qed

```

```

lemma (in standard-borel-ne) measure-with-args-morphismP:
  assumes [measurable]:  $\mu \in X \rightarrow_M \text{prob-algebra } M$ 
  shows  $\text{qbs-l-inverse} \circ \mu \in \text{measure-to-qbs } X \rightarrow_Q \text{monadP-qbs}$  (measure-to-qbs M)
  by (rule qbs-morphism-monadPI'[OF - measure-with-args-morphism])
  (insert measurable-space[OF assms], auto simp: qbs-space-R space-prob-algebra prob-kernel-def' intro!: qbs-l-inverse-in-space-monadP prob-kernel.s-finite-kernel-prob-kernel)

```

4.2.2 Counting Measure

```

abbreviation counting-measure-qbs A  $\equiv \text{qbs-l-inverse}$  (count-space A)

```

```

lemma qbs-nn-integral-count-space-nat:
  fixes  $f :: \text{nat} \Rightarrow \text{ennreal}$ 
  shows  $(\int^+_Q i. f \ i \ \partial \text{counting-measure-qbs UNIV}) = (\sum i. f \ i)$ 
  by (simp add: standard-borel-ne.qbs-l-qbs-l-inverse[OF - refl sigma-finite-measure.s-finite-measure[OF sigma-finite-measure-count-space]] qbs-nn-integral-def2-l nn-integral-count-space-nat)

```

4.2.3 Normal Distribution

```

lemma qbs-normal-distribution-qbs:  $(\lambda \mu \ \sigma. \text{density-qbs } \text{lborel}_Q \ (\text{normal-density } \mu \ \sigma)) \in \text{qbs-borel} \Rightarrow_Q \text{qbs-borel} \Rightarrow_Q \text{monadM-qbs } \text{qbs-borel}$ 
  by simp

```

```

lemma qbs-l-qbs-normal-distribution[simp]:  $\text{qbs-l}$  (density-qbs lborelQ (normal-density  $\mu \ \sigma$ )) = density lborel (normal-density  $\mu \ \sigma$ )
  by (auto simp: qbs-l-density-qbs[of - qbs-borel])

```

```

lemma qbs-normal-distribution-P:  $\sigma > 0 \implies \text{density-qbs } \text{lborel}_Q \ (\text{normal-density } \mu \ \sigma) \in \text{qbs-space}$  (monadP-qbs qbs-borel)
  by (auto simp: monadP-qbs-def sub-qbs-space prob-space-normal-density)

```

```

lemma qbs-normal-distribution-integral:
   $(\int_Q x. f \ x \ \partial \ (\text{density-qbs } \text{lborel}_Q \ (\text{normal-density } \mu \ \sigma))) = (\int x. f \ x \ \partial \ (\text{density } \text{lborel} \ (\lambda x. \text{ennreal} \ (\text{normal-density } \mu \ \sigma \ x))))$ 
  by (auto simp: qbs-integral-def2-l)

```


lemma *qbs-normal-distribution-expectation*:

assumes [*measurable*]: $f \in \text{borel-measurable borel}$ **and** [*arith*]: $\sigma > 0$
shows $(\int_Q x. f x \partial (\text{density-qbs lborel}_Q (\text{normal-density } \mu \sigma))) = (\int x. \text{normal-density } \mu \sigma x * f x \partial \text{lborel})$
by (*simp add: qbs-normal-distribution-integral integral-real-density integral-density*)

lemma *qbs-normal-posterior*:

assumes [*arith*]: $\sigma > 0 \ \sigma' > 0$
shows $\text{normalize-qbs} (\text{density-qbs} (\text{density-qbs lborel}_Q (\text{normal-density } \mu \sigma)) (\text{normal-density } \mu' \sigma')) = \text{density-qbs lborel}_Q (\text{normal-density} ((\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2)))$ (**is** ?lhs = ?rhs)

proof –

have 0: $\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2) > 0 \ \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)) > 0$
by (*simp-all add: power2-eq-square sum-squares-gt-zero-iff*)
have 1: $\text{qbs-l} (\text{density-qbs lborel}_Q (\lambda x. \text{ennreal} (1 / \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)) * \text{exp} (- ((\mu - \mu')^2 / (2 * \sigma^2 + 2 * \sigma'^2)))) * \text{normal-density} ((\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2)) x))) \text{UNIV} = \text{ennreal} (\text{exp} (- ((\mu - \mu')^2 / (2 * \sigma^2 + 2 * \sigma'^2))) / \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)))$
using *prob-space.emeasure-space-1[OF prob-space-normal-density[OF 0(1)]]*
by (*auto simp add: qbs-l-density-qbs[of - qbs-borel] emeasure-density ennreal-mult' nn-integral-cmult simp del: times-divide-eq-left (simp add: ennreal-mult'[symmetric])*)
have ?lhs = $\text{normalize-qbs} (\text{density-qbs lborel}_Q (\lambda x. \text{ennreal} (1 / \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)) * \text{exp} (- ((\mu - \mu')^2 / (2 * \sigma^2 + 2 * \sigma'^2)))) * \text{normal-density} ((\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2)) x)))$
by (*simp add: density-qbs-density-qbs-eq[of - qbs-borel] ennreal-mult'[symmetric] normal-density-times del: times-divide-eq-left*)
also have ... = $\text{density-qbs} (\text{density-qbs lborel}_Q (\lambda x. \text{ennreal} (1 / \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)) * \text{exp} (- ((\mu - \mu')^2 / (2 * \sigma^2 + 2 * \sigma'^2)))) * \text{normal-density} ((\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2)) x))) (\lambda x. 1 / \text{emeasure} (\text{qbs-l} (\text{density-qbs lborel}_Q (\lambda x. \text{ennreal} (1 / \text{sqrt} (2 * \text{pi} * (\sigma^2 + \sigma'^2)) * \text{exp} (- ((\mu - \mu')^2 / (2 * \sigma^2 + 2 * \sigma'^2)))) * \text{normal-density} ((\mu * \sigma^2 + \mu' * \sigma^2) / (\sigma^2 + \sigma'^2)) (\sigma * \sigma' / \text{sqrt} (\sigma^2 + \sigma'^2)) x)))) (\text{qbs-space borel}_Q))$
by (*rule normalize-qbs (simp-all add: 1 del: times-divide-eq-left)*)
also have ... = ?rhs
by (*simp add: 1 density-qbs-density-qbs-eq[of - qbs-borel] del: times-divide-eq-left, auto intro!: density-qbs-cong[of - qbs-borel]*)
(insert 0, auto simp: ennreal-1[symmetric] ennreal-mult'[symmetric] divide-ennreal normal-density-def simp del: ennreal-1)
finally show ?thesis .
qed

4.2.4 Uniform Distribution

definition *uniform-qbs* :: 'a qbs-measure \Rightarrow 'a set \Rightarrow 'a qbs-measure **where**
uniform-qbs $\equiv (\lambda s A. \text{qbs-l-inverse} (\text{uniform-measure} (\text{qbs-l } s) A))$

lemma (*in standard-borel-ne*) *qbs-l-uniform-qbs*':

assumes *sets* $\mu = \text{sets } M \text{ s-finite-measure } \mu \ \mu \ A \neq 0$
shows $\text{qbs-l} (\text{uniform-qbs} (\text{qbs-l-inverse } \mu) A) = \text{uniform-measure } \mu \ A$ (**is** ?lhs

```

= ?rhs)
proof -
  have ?lhs = qbs-l (qbs-l-inverse (uniform-measure  $\mu$  A))
  by(simp add: qbs-l-qbs-l-inverse[OF assms(1,2)] uniform-qbs-def)
  also have ... = ?rhs
  proof(rule qbs-l-qbs-l-inverse)
  consider  $\mu$  A =  $\infty$  |  $\mu$  A  $\neq$   $\infty$  by auto
  then show s-finite-measure (uniform-measure  $\mu$  A)
  proof cases
  case 1
  have A[measurable]: A  $\in$  sets  $\mu$ 
  using assms(3) emeasure-notin-sets by blast
  have uniform-measure  $\mu$  A = density  $\mu$  ( $\lambda x$ . 0)
  by(auto simp: uniform-measure-def 1 intro!: density-cong)
  also have ... = null-measure  $\mu$ 
  by(simp add: null-measure-eq-density)
  finally show ?thesis
  by(auto intro!: finite-measure.s-finite-measure-finite-measure finite-measureI)
next
  case 2
  show ?thesis
  by(rule prob-space.s-finite-measure-prob[OF prob-space-uniform-measure[OF
assms(3) 2]])
  qed
  qed(simp add: assms)
  finally show ?thesis .
qed

```

```

corollary(in standard-borel-ne) qbs-l-uniform-qbs:
  assumes s  $\in$  qbs-space (monadM-qbs (measure-to-qbs M)) qbs-l s A  $\neq$  0
  shows qbs-l (uniform-qbs s A) = uniform-measure (qbs-l s) A
  by(simp add: qbs-l-uniform-qbs'[OF sets-qbs-l[OF assms(1),simplified lr-sets-ident]
qbs-l-s-finite.s-finite-measure-axioms assms(2),symmetric] qbs-l-inverse-qbs-l[OF assms(1)])

```

```

lemma interval-uniform-qbs: ( $\lambda a$  b. uniform-qbs lborel $_Q$  {a<..b::real})  $\in$  borel $_Q$ 
 $\Rightarrow_Q$  borel $_Q$   $\Rightarrow_Q$  monadM-qbs borel $_Q$ 

```

```

proof(rule curry-preserves-morphisms)

```

```

  have ( $\lambda xy$ . uniform-qbs lborel $_Q$  {fst xy<..snd xy::real}) = qbs-l-inverse  $\circ$  ( $\lambda xy$ .
uniform-measure lborel {fst xy<..snd xy})

```

```

  by(auto simp: uniform-qbs-def)

```

```

  also have ...  $\in$  measure-to-qbs (borel  $\otimes_M$  borel)  $\rightarrow_Q$  monadM-qbs borel $_Q$ 

```

```

proof(safe intro!: standard-borel-ne.measure-with-args-morphism measure-kernel.s-finite-kernel-finite-bounded)

```

```

  show measure-kernel (borel  $\otimes_M$  borel) borel ( $\lambda xy$ . uniform-measure lborel {fst
xy<..snd xy :: real})

```

```

proof

```

```

  fix B :: real set

```

```

  assume [measurable]:B  $\in$  sets borel

```

```

  have [simp]:emeasure lborel ({fst x<..snd x}  $\cap$  B) / emeasure lborel {fst
x<..snd x} = (if fst x  $\leq$  snd x then emeasure lborel ({fst x<..snd x}  $\cap$  B) /

```

```

ennreal (snd x - fst x) else 0) for x
  by auto
  show (λx. emeasure (uniform-measure lborel {fst x<..M borel)
  by (simp, measurable) auto
qed auto
next
  show (a, b) ∈ space (borel ⊗M borel) ⇒ emeasure (uniform-measure lborel
{fst (a, b)<..Q {fst xy<..Q ⊗Q
borelQ →Q monadM-qbs borelQ
  by (simp add: borel-prod qbs-borel-prod)
qed

```

```

context
  fixes a b :: real
  assumes [arith]: a < b
begin

```

```

lemma qbs-uniform-distribution-expectation:
  assumes f ∈ qbs-borel →Q qbs-borel
  shows (∫+Q x. f x ∂uniform-qbs lborelQ {a<..+x ∈ {a<..

```

4.2.5 Bernoulli Distribution

```

abbreviation qbs-bernoulli :: real ⇒ bool qbs-measure where
qbs-bernoulli ≡ (λx. qbs-pmf (bernoulli-pmf x))

```

```

lemma bernoulli-measurable:
(λx. measure-pmf (bernoulli-pmf x)) ∈ borel →M prob-algebra (count-space UNIV)
proof (rule measurable-prob-algebra-generated [where Ω=UNIV and G=UNIV])
  fix A :: bool set
  have A ⊆ {True, False}
  by auto
  then consider A = {} | A = {True} | A = {False} | A = {False, True}
  by auto
  thus (λa. emeasure (measure-pmf (bernoulli-pmf a)) A) ∈ borel-measurable borel

```

by(*cases, simp-all add: emeasure-measure-pmf-finite bernoulli-pmf.rep-eq UNIV-bool[symmetric]*)
qed (*auto simp add: sets-borel-eq-count-space Int-stable-def measure-pmf.prob-space-axioms*)

lemma *qbs-bernoulli-morphism*: $qbs\text{-bernoulli} \in qbs\text{-borel} \rightarrow_Q \text{monadP}\text{-qbs}$ (*qbs-count-space UNIV*)
using *standard-borel-ne.measure-with-args-morphismP[OF - bernoulli-measurable]*
by (*simp add: qbs-pmf-def comp-def*)

lemma *qbs-bernoulli-expectation*:
assumes [*simp*]: $0 \leq p \leq 1$
shows $(\int_Q x. f x \partial qbs\text{-bernoulli } p) = f \text{ True} * p + f \text{ False} * (1 - p)$
by(*simp add: qbs-integral-def2-l*)

end

5 Examples

5.1 Montecarlo Approximation

theory *Montecarlo*
imports *Monad-QuasiBorel*
begin

declare [[*coercion qbs-l*]]

abbreviation *real-quasi-borel* :: *real quasi-borel* (\mathbb{R}_Q) **where**
real-quasi-borel $\equiv qbs\text{-borel}$

abbreviation *nat-quasi-borel* :: *nat quasi-borel* (\mathbb{N}_Q) **where**
nat-quasi-borel $\equiv qbs\text{-count-space UNIV}$

primrec *montecarlo* :: '*a* *qbs-measure* \Rightarrow ('*a* \Rightarrow *real*) \Rightarrow *nat* \Rightarrow *real qbs-measure*
where

montecarlo - - 0 = *return-qbs* \mathbb{R}_Q 0 |
montecarlo d h (*Suc* n) = *do* { *m* \leftarrow *montecarlo* d h n;
x \leftarrow d;
return-qbs \mathbb{R}_Q ((h *x* + *m* * (*real* n)) / (*real* (*Suc* n)))}

declare
bind-qbs-morphismP[*qbs*]
return-qbs-morphismP[*qbs*]
qbs-pair-measure-morphismP[*qbs*]

lemma *montecarlo-qbs-morphism*[*qbs*]: $montecarlo \in qbs\text{-space}$ ($monadP\text{-qbs } X \Rightarrow_Q$
 $(X \Rightarrow_Q \mathbb{R}_Q) \Rightarrow_Q \mathbb{N}_Q \Rightarrow_Q monadP\text{-qbs } \mathbb{R}_Q$)
by(*simp add: montecarlo-def*)

lemma *qbs-integrable-indep-mult2*[*simp, intro!*]:

fixes $f :: - \Rightarrow \text{real}$
assumes $qbs\text{-integrable } p \ f$
and $qbs\text{-integrable } q \ g$
shows $qbs\text{-integrable } (p \otimes_{Q\text{mes}} q) (\lambda x. g (snd \ x) * f (fst \ x))$
using $qbs\text{-integrable-indep-mult}[OF \text{ assms}]$ **by** $(simp \ add: \ mult.\text{commute})$

lemma *montecarlo-integrable*:

assumes $[qbs]: p \in qbs\text{-space } (monadP\text{-qbs } X) \ h \in X \rightarrow_Q \mathbb{R}_Q \ qbs\text{-integrable } p \ h$
 $qbs\text{-integrable } p (\lambda x. h \ x * h \ x)$

shows $qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x) \ qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x * x)$

proof –

have $qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x) \wedge qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x * x)$

proof(*induction n*)

case 0

then show $?case$

by *simp*

next

case (*Suc n*)

hence $1[intro, simp]: qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x) \ qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x * x)$

by *simp-all*

have $2[intro, simp]: \bigwedge q \ f. \ qbs\text{-integrable } q (\lambda x. f \ x * f \ x) \implies qbs\text{-integrable } q (\lambda x. f \ x * a * (f \ x * b))$ **for** $a \ b :: \text{real}$

by(*auto simp: mult.commute[of - a] mult.assoc intro!: qbs-integrable-scaleR-left[where 'a=real, simplified] qbs-integrable-scaleR-right[where 'a=real, simplified]*) (*auto simp: mult.assoc[of - - b, symmetric] intro!: qbs-integrable-scaleR-left[where 'a=real, simplified]*)

show $?case$

by(*auto simp add: qbs-bind-bind-return2P'[of - $\mathbb{R}_Q \ X \ \mathbb{R}_Q$] split-beta' qbs-pair-measure-def[OF qbs-space-monadPM qbs-space-monadPM, symmetric] qbs-integrable-bind-return[OF qbs-space-monadPM, of - $\mathbb{R}_Q \otimes_Q \ X - \mathbb{R}_Q$] comp-def distrib-right distrib-left intro!: qbs-integrable-indep-mult qbs-integrable-indep1[OF 1(1), of - X] qbs-integrable-indep2[OF assms(3), of - \mathbb{R}_Q] qbs-integrable-indep1[OF 1(2), of - X] qbs-integrable-indep2[OF assms(4), of - \mathbb{R}_Q] qbs-integrable-const[OF assms(1)] qbs-integrable-scaleR-left[where 'a=real, simplified] assms(3,4)]*)

qed

thus $qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x) \ qbs\text{-integrable } (montecarlo \ p \ h \ n) (\lambda x. x * x)$

by *simp-all*

qed

lemma

fixes $n :: \text{nat}$

assumes $[qbs]: p \in qbs\text{-space } (monadP\text{-qbs } X) \ h \in X \rightarrow_Q \mathbb{R}_Q \ qbs\text{-integrable } p \ h$
 $qbs\text{-integrable } p (\lambda x. h \ x * h \ x)$

and $e: e > 0$

and $(\int_Q x. h \ x \ \partial p) = \mu (\int_Q x. (h \ x - \mu)^2 \ \partial p) = \sigma^2$

and $n:n > 0$
shows $\mathcal{P}(y \text{ in montecarlo } p \ h \ n. |y - \mu| \geq e) \leq \sigma^2 / (\text{real } n * e^2)$ (**is** $?P \leq -$)
proof –
note $[intro!] = \text{montecarlo-integrable}[OF \text{ assms}(1-4)] \text{ qbs-integrable-indep-mult}$
 $\text{qbs-integrable-indep1}[OF \text{ montecarlo-integrable}(1)[OF \text{ assms}(1-4)], \text{of} - X] \text{ qbs-integrable-indep2}[OF$
 $\text{ assms}(3), \text{of} - \mathbb{R}_Q] \text{ qbs-integrable-indep1}[OF \text{ montecarlo-integrable}(2)[OF \text{ assms}(1-4)], \text{of}$
 $- X] \text{ qbs-integrable-indep2}[OF \text{ assms}(4), \text{of} - \mathbb{R}_Q] \text{ qbs-integrable-const}[OF \text{ assms}(1)]$
 $\text{qbs-integrable-scaleR-right}[\text{where } 'a=\text{real, simplified}] \text{ qbs-integrable-scaleR-left}[\text{where}$
 $'a=\text{real, simplified}] \text{ assms}(3,4) \text{ qbs-integrable-sq} \text{ qbs-integrable-const}[\text{of montecarlo } p$
 $h - \otimes_{Q_{mes}} p \ \mathbb{R}_Q \otimes_Q X] \text{ qbs-integrable-const}[\text{of montecarlo } p \ h - \mathbb{R}_Q]$
have $\text{integrable}[intro, simp]: \bigwedge q f. \text{qbs-integrable } q (\lambda x. f x * f x) \implies \text{qbs-integrable}$
 $q (\lambda x. f x * a * (f x * b))$ **for** $a \ b :: \text{real}$
by $(\text{auto simp: mult.commute}[\text{of} - a] \text{ mult.assoc}) (\text{auto simp: mult.assoc}[\text{of} - -$
 $b, \text{symmetric}])$
have $\text{exp}:(\int_Q y. y \ \partial(\text{montecarlo } p \ h \ n)) = \mu$ (**is** $?e \ n$) **and** $\text{var}:(\int_Q y. (y - \mu)^2$
 $\partial(\text{montecarlo } p \ h \ n)) = \sigma^2 / n$ (**is** $?v \ n$)
proof –
have $?e \ n \wedge ?v \ n$
using n
proof $(\text{induction } n)$
case 0
then show $?case$
by simp
next
case $ih:(\text{Suc } n)$
consider $n = 0 \mid n > 0$ **by** auto
then show $?case$
proof cases
case 1
then show $?thesis$
by $(\text{auto simp: qbs-integral-indep2}[OF \text{ qbs-integrable-sq}[OF \text{ qbs-integrable-const}[OF$
 $\text{ assms}(1)] \text{ assms}(3)], \text{simplified power2-eq-square}, OF \text{ assms}(4), \text{of} - \text{qbs-borel}] \text{ power2-eq-square}$
 $\text{qbs-bind-bind-return2P}'[\text{of} - \mathbb{R}_Q \ X \ \mathbb{R}_Q] \text{ split-beta}' \text{ qbs-pair-measure-def}[OF \text{ qbs-space-monadPM}$
 $\text{qbs-space-monadPM}, \text{symmetric}] \text{ qbs-integral-bind-return}[OF \text{ qbs-space-monadPM}, \text{of}$
 $- \mathbb{R}_Q \otimes_Q X - \mathbb{R}_Q] \text{ comp-def qbs-integral-indep2}[OF \text{ assms}(3), \text{of} - \mathbb{R}_Q] \text{ qbs-integral-indep2}[OF$
 $\text{ assms}(4), \text{of} - \mathbb{R}_Q] \text{ assms}(6,7)[\text{simplified power2-eq-square}])$
next
case $n[\text{arith}]:2$
with ih **have** $\text{eq}: (\int_Q y. y \ \partial \text{montecarlo } p \ h \ n) = \mu$ $(\int_Q y. (y - \mu)^2$
 $\partial \text{montecarlo } p \ h \ n) = \sigma^2 / \text{real } n$
by simp-all

have $1:?e (\text{Suc } n)$
proof –
have $(\int_Q x. h (\text{snd } x) + \text{fst } x * \text{real } n \ \partial(\text{montecarlo } p \ h \ n \otimes_{Q_{mes}}$
 $p)) = ((\int_Q x. h (\text{snd } x) \ \partial(\text{montecarlo } p \ h \ n \otimes_{Q_{mes}} p)) + (\int_Q x. \text{fst } x * \text{real } n$
 $\partial(\text{montecarlo } p \ h \ n \otimes_{Q_{mes}} p)))$
by $(\text{rule qbs-integral-add}) \text{ auto}$
also have $\dots = \mu + \mu * n$

proof –
have $(\int_Q x. h (snd x) \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p)) = (\int_Q x. h \ x \ \partial p)$
by(*auto intro!*: *qbs-integral-indep2*[*of - - - \mathbb{R}_Q*])
moreover have $(\int_Q x. fst \ x \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p)) = (\int_Q y. y \ \partial \text{montecarlo } p \ h \ n)$
by(*auto intro!*: *qbs-integral-indep1*[*of - - - X*])
ultimately show *?thesis*
by(*simp add*: *eq assms*)
qed
finally have $(\int_Q y. y \ \partial \text{montecarlo } p \ h \ (\text{Suc } n)) = 1 / (\text{Suc } n) * (\mu + \mu * n)$
by(*auto simp*: *qbs-bind-bind-return2P'*[*of - \mathbb{R}_Q X \mathbb{R}_Q*] *split-beta'*
qbs-pair-measure-def[*OF qbs-space-monadPM qbs-space-monadPM,symmetric*] *qbs-integral-bind-return*[*OF qbs-space-monadPM,of - \mathbb{R}_Q $\otimes_Q X - \mathbb{R}_Q$*] *comp-def*)
also have $\dots = 1 / (\text{Suc } n) * (\mu * (1 + \text{real } n))$
by(*simp add*: *distrib-left*)
also have $\dots = \mu$
by *simp*
finally show *?thesis* .
qed
have *2*: *?v* (*Suc n*)
proof –
have $(\int_Q y. (y - \mu)^2 \ \partial \text{montecarlo } p \ h \ (\text{Suc } n)) = (\int_Q x. ((h (snd x) + fst \ x * \text{real } n) / \text{real } (\text{Suc } n) - \mu)^2 \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by(*auto simp*: *qbs-bind-bind-return2P'*[*of - \mathbb{R}_Q X \mathbb{R}_Q*] *split-beta'*
qbs-pair-measure-def[*OF qbs-space-monadPM qbs-space-monadPM,symmetric*] *qbs-integral-bind-return*[*OF qbs-space-monadPM,of - \mathbb{R}_Q $\otimes_Q X - \mathbb{R}_Q$*] *comp-def*)
also have $\dots = (\int_Q x. ((h (snd x) + fst \ x * \text{real } n) / \text{real } (\text{Suc } n) - (\text{Suc } n) * \mu / \text{Suc } n)^2 \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by *simp*
also have $\dots = (\int_Q x. ((h (snd x) + fst \ x * \text{real } n - (\text{Suc } n) * \mu) / \text{Suc } n)^2 \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by(*simp only*: *diff-divide-distrib*[*symmetric*])
also have $\dots = (\int_Q x. ((h (snd x) - \mu + (fst \ x * \text{real } n - \text{real } n * \mu)) / \text{Suc } n)^2 \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by (*simp add*: *add-diff-add distrib-left mult.commute*)
also have $\dots = (\int_Q x. (1 / \text{real } (\text{Suc } n) * (h (snd x) - \mu) + n / \text{real } (\text{Suc } n) * (fst \ x - \mu))^2 \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by(*auto simp*: *add-divide-distrib*[*symmetric*] *pair-qbs-space mult.commute*[*of - real n*]) (*simp add*: *right-diff-distrib*)
also have $\dots = (\int_Q x. (1 / \text{real } (\text{Suc } n) * (h (snd x) - \mu))^2 + (n / \text{real } (\text{Suc } n) * (fst \ x - \mu))^2 + 2 * (1 / \text{real } (\text{Suc } n) * (h (snd x) - \mu)) * (n / \text{real } (\text{Suc } n) * (fst \ x - \mu)) \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by(*simp add*: *power2-sum*)
also have $\dots = (\int_Q x. 1 / (\text{real } (\text{Suc } n))^2 * ((h (snd x) - \mu))^2 + (n / \text{real } (\text{Suc } n))^2 * ((fst \ x - \mu))^2 + 2 * (1 / \text{real } (\text{Suc } n) * (h (snd x) - \mu)) * (n / \text{real } (\text{Suc } n) * (fst \ x - \mu)) \ \partial(\text{montecarlo } p \ h \ n \ \otimes_{Q_{mes}} p))$
by(*simp only*: *power-mult-distrib*) (*simp add*: *power2-eq-square*)
also have $\dots = (\int_Q x. 1 / (\text{real } (\text{Suc } n))^2 * ((h (snd x) - \mu))^2 + (n /$

$\text{real } (\text{Suc } n)^2 * ((\text{fst } x - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p) + (\int_Q x. 2 * (1 / \text{real } (\text{Suc } n) * (\text{h } (\text{snd } x) - \mu)) * (n / \text{real } (\text{Suc } n) * (\text{fst } x - \mu)) \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p))$
by(rule qbs-integral-add, auto) (auto simp: power2-eq-square)
also have ... = $(\int_Q x. 1 / (\text{real } (\text{Suc } n))^2 * ((\text{h } (\text{snd } x) - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p) + (\int_Q x. (n / \text{real } (\text{Suc } n))^2 * ((\text{fst } x - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p)) + (\int_Q x. 2 * (1 / \text{real } (\text{Suc } n) * (\text{h } (\text{snd } x) - \mu)) * (n / \text{real } (\text{Suc } n) * (\text{fst } x - \mu)) \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p))$
proof -
have $(\int_Q x. 1 / (\text{real } (\text{Suc } n))^2 * ((\text{h } (\text{snd } x) - \mu))^2 + (n / \text{real } (\text{Suc } n))^2 * ((\text{fst } x - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p)) = (\int_Q x. 1 / (\text{real } (\text{Suc } n))^2 * ((\text{h } (\text{snd } x) - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p) + (\int_Q x. (n / \text{real } (\text{Suc } n))^2 * ((\text{fst } x - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p))$
by(rule qbs-integral-add, auto) (auto simp: power2-eq-square)
thus ?thesis **by** simp
qed
also have ... = $1 / (\text{real } (\text{Suc } n))^2 * \sigma^2 + (n / \text{real } (\text{Suc } n))^2 * (\sigma^2 / n)$
proof -
have 1: $(\int_Q x. ((\text{h } (\text{snd } x) - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p)) = (\int_Q x. (\text{h } x - \mu)^2 \partial p)$
by(auto intro!: qbs-integral-indep2[of - - \mathbb{R}_Q]) (auto simp: power2-eq-square)
have 2: $(\int_Q x. ((\text{fst } x - \mu))^2 \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p)) = (\int_Q y. (y - \mu)^2 \partial \text{montecarlo } p \text{ h } n)$
by(auto intro!: qbs-integral-indep1[of - - X]) (auto simp: power2-eq-square)
have 3: $(\int_Q x. 2 * (1 / \text{real } (\text{Suc } n) * (\text{h } (\text{snd } x) - \mu)) * (n / \text{real } (\text{Suc } n) * (\text{fst } x - \mu)) \partial(\text{montecarlo } p \text{ h } n \otimes_{Qmes} p)) = 0$ (**is** ?l = -)
proof -
have ?l = $(\int_Q x. 2 * (1 / \text{real } (\text{Suc } n) * (\text{h } x - \mu)) \partial p) * (\int_Q x. (n / \text{real } (\text{Suc } n) * (x - \mu)) \partial \text{montecarlo } p \text{ h } n)$
by(rule qbs-integral-indep-mult2[of - \mathbb{R}_Q - X]) auto
also have ... = 0
by(simp add: qbs-integral-diff[OF montecarlo-integrable(1)[OF assms(1-4)] qbs-integrable-const[of - \mathbb{R}_Q]] eq qbs-integral-const-prob[of - \mathbb{R}_Q])
finally show ?thesis .
qed
show ?thesis
unfolding 3 **by**(simp add: 1 2 eq assms)
qed
also have ... = $1 / (\text{real } (\text{Suc } n))^2 * \sigma^2 + \text{real } n / (\text{real } (\text{Suc } n))^2 * \sigma^2$
by(auto simp: power2-eq-square)
also have ... = $(1 + \text{real } n) * \sigma^2 / (\text{real } (\text{Suc } n))^2$
by (simp add: add-divide-distrib ring-class.ring-distrib(2))
also have ... = $\sigma^2 / \text{real } (\text{Suc } n)$
by(auto simp: power2-eq-square)
finally show ?thesis .
qed
show ?thesis
by(simp only: 1 2)
qed


```

qed
thus ?e n ?v n by simp-all
qed

```

```

have ?P ≤ (∫Q x. (x - μ)2 ∂montecarlo p h n) / e2
  unfolding exp[symmetric] by(rule Chebyshev-inequality-qbs-prob[of montecarlo
p h n qbs-borel λx. x]) (auto simp: power2-eq-square e)
also have ... = σ2 / (real n * e2)
  by(simp add: var)
finally show ?thesis .
qed

```

```
end
```

5.2 Query

```

theory Query
  imports Monad-QuasiBorel
begin

```

```

declare [[coercion qbs-l]]
abbreviation qbs-real :: real quasi-borel      ( $\mathbb{R}_Q$ ) where  $\mathbb{R}_Q \equiv$  qbs-borel
abbreviation qbs-ennreal :: ennreal quasi-borel ( $\mathbb{R}_{Q \geq 0}$ ) where  $\mathbb{R}_{Q \geq 0} \equiv$  qbs-borel
abbreviation qbs-nat :: nat quasi-borel        ( $\mathbb{N}_Q$ ) where  $\mathbb{N}_Q \equiv$  qbs-count-space UNIV
abbreviation qbs-bool :: bool quasi-borel      ( $\mathbb{B}_Q$ ) where  $\mathbb{B}_Q \equiv$  count-spaceQ UNIV

```

```

definition query :: ['a qbs-measure, 'a ⇒ ennreal] ⇒ 'a qbs-measure where
query ≡ (λs f. normalize-qbs (density-qbs s f))

```

```

lemma query-qbs-morphism[qbs]: query ∈ monadM-qbs X →Q (X ⇒Q qbs-borel)
⇒Q monadM-qbs X
  by(simp add: query-def)

```

```

definition condition ≡ (λs P. query s (λx. if P x then 1 else 0))

```

```

lemma condition-qbs-morphism[qbs]: condition ∈ monadM-qbs X ⇒Q (X ⇒Q  $\mathbb{B}_Q$ )
⇒Q monadM-qbs X
  by(simp add: condition-def)

```

```

lemma condition-morphismP:

```

```

  assumes ∧x. x ∈ qbs-space X ⇒  $\mathcal{P}(y \text{ in } qbs-l (s x). P x y) \neq 0$ 
  and [qbs]: s ∈ X →Q monadP-qbs Y P ∈ X →Q Y ⇒Q qbs-count-space UNIV
  shows (λx. condition (s x) (P x)) ∈ X →Q monadP-qbs Y
proof(rule qbs-morphism-cong'[where f=λx. normalize-qbs (density-qbs (s x) (indicator
{y ∈ qbs-space Y. P x y}))])

```

```

fix x
assume x[qbs]:x ∈ qbs-space X
have density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y}) = density-qbs (s x)
(λy. if P x y then 1 else 0)
by(auto intro!: density-qbs-cong[OF qbs-space-monadPM[OF qbs-morphism-space[OF
assms(2) x]]] indicator-qbs-morphism'')
thus normalize-qbs (density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y})) =
condition (s x) (P x)
unfolding condition-def query-def by simp
next
show (λx. normalize-qbs (density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y})))
∈ X →Q monadP-qbs Y
proof(rule normalize-qbs-morphismP[of λx. density-qbs (s x) (indicator {y ∈
qbs-space Y. P x y})])
show (λx. density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y})) ∈ X →Q
monadM-qbs Y
using qbs-morphism-monadPD[OF assms(2)] by simp
next
fix x
assume x:x ∈ qbs-space X
show emeasure (qbs-l (density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y})))
(qbs-space Y) ≠ 0
emeasure (qbs-l (density-qbs (s x) (indicator {y ∈ qbs-space Y. P x y})))
(qbs-space Y) ≠ ∞
using assms(1)[OF x] qbs-l-monadP-le1[OF qbs-morphism-space[OF assms(2)
x]]
by(auto simp add: qbs-l-density-qbs-indicator[OF qbs-space-monadPM[OF
qbs-morphism-space[OF assms(2) x]] qbs-morphism-space[OF assms(3) x]] mea-
sure-def space-qbs-l-in[OF qbs-space-monadPM[OF qbs-morphism-space[OF assms(2)
x]]])
qed
qed

```

lemma query-Bayes:

```

assumes [qbs]: s ∈ qbs-space (monadP-qbs X) qbs-pred X P qbs-pred X Q
shows  $\mathcal{P}(x \text{ in condition } s \ P. \ Q \ x) = \mathcal{P}(x \text{ in } s. \ Q \ x \mid P \ x)$  (is ?lhs = ?pq)
proof –
have X: qbs-space X ≠ {}
using assms(1) by(simp only: monadP-qbs-empty-iff[of X]) blast
note s[qbs] = qbs-space-monadPM[OF assms(1)]
have density-eq: density-qbs s (λx. if P x then 1 else 0) = density-qbs s (indicator
{x ∈ qbs-space X. P x})
by(auto intro!: density-qbs-cong[of - X] indicator-qbs-morphism'')
consider emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space
X) = 0 | emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)
≠ 0 by auto
then show ?thesis
proof cases
case 1

```

```

have 2:normalize-qbs (density-qbs s (λx. if P x then 1 else 0)) = qbs-null-measure
X
  by(rule normalize-qbs0) (auto simp: 1)
  have  $\mathcal{P}(\omega \text{ in } \text{qbs-l } s. P \omega) = \text{measure } (\text{qbs-l } (\text{density-qbs } s (\lambda x. \text{if } P x \text{ then } 1 \text{ else } 0))) (\text{qbs-space } X)$ 
  by(simp add: space-qbs-l-in[OF s] measure-def density-eq qbs-l-density-qbs-indicator[OF
s])
  also have ... = 0
  by(simp add: measure-def 1)
  finally show ?thesis
  by(auto simp: condition-def query-def cond-prob-def 2 1 qbs-null-measure-null-measure[OF
X])
next
case 1[simp]:2
from rep-qbs-space-monadP[OF assms(1)]
obtain  $\alpha \mu$  where  $hs: s = \llbracket X, \alpha, \mu \rrbracket_{\text{sf in qbs-prob } X \alpha \mu}$  by auto
then interpret qp: qbs-prob X  $\alpha \mu$  by simp
have [measurable]:Measurable.pred (qbs-to-measure X) P Measurable.pred (qbs-to-measure
X) Q
  using assms(2,3) by(simp-all add: lr-adjunction-correspondence)
  have 2[simp]: emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0)))
(qbs-space X)  $\neq \top$ 
  by(simp add: hs(1) qp.density-qbs qbs-s-finite.qbs-l[OF qp.density-qbs-s-finite]
emeasure-distr emeasure-distr[where  $N = \text{qbs-to-measure } X, \text{OF - sets.top, simplified space-L}$ ]
emeasure-density, rule order.strict-implies-not-eq[OF order.strict-trans1[OF
qp.nn-integral-le-const[of 1] ennreal-one-less-top]]) auto
  have 3: measure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)
> 0
  using 2 emeasure-eq-ennreal-measure zero-less-measure-iff by fastforce
  have query s (λx. if P x then 1 else 0) = density-qbs (density-qbs s (λx. if P x
then 1 else 0)) (λx. 1 / emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0)))
(qbs-space X))
  unfolding query-def by(rule normalize-qbs) auto
  also have ... = density-qbs s (λx. (if P x then 1 else 0) * (1 / emeasure (qbs-l
(density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)))
  by(simp add: density-qbs-density-qbs-eq[OF qbs-space-monadPM[OF assms(1)]])
  finally have query:query s (λx. if P x then 1 else 0) = ... .
  have ?lhs = measure (density (qbs-l s) (λx. (if P x then 1 else 0) * (1 /
emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)))) {x ∈
space (qbs-l s). Q x}
  by(simp add: condition-def query qbs-l-density-qbs[OF qbs-space-monadPM[OF
assms(1)]])
  also have ... = measure (density  $\mu$  (λx. (if P (α x) then 1 else 0) * (1 /
emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)))) {y. α y
∈ space (qbs-to-measure X) ∧ Q (α y)}
  by(simp add: hs(1) qp.qbs-l density-distr measure-def emeasure-distr)
  also have ... = measure (density  $\mu$  (λx. indicator {r. P (α r)} x * (1 / emeasure
(qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)))) {y. Q (α y)}
  proof –

```

```

    have [simp]:(if P (α r) then 1 else 0) = indicator {r. P (α r)} r for r
      by auto
    thus ?thesis by(simp add: space-L)
  qed
  also have ... = enn2real (1 / emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)) * measure μ {r. P (α r) ∧ Q (α r)}
  proof -
    have n-inf: 1 / emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X) ≠ ∞
    using 1 by(auto simp: ennreal-divide-eq-top-iff)
    show ?thesis
      by(simp add: measure-density-times[OF - - n-inf] Collect-conj-eq)
  qed
  also have ... = (1 / measure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)) * qp.prob {r. P (α r) ∧ Q (α r)}
  proof -
    have 1 / emeasure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X) = ennreal (1 / measure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X))
    by(auto simp add: emeasure-eq-ennreal-measure[OF 2] ennreal-1[symmetric] simp del: ennreal-1 intro!: divide-ennreal) (simp-all add: 3)
    thus ?thesis by simp
  qed also have ... = ?pq
  proof -
    have qp:P(x in s. Q x ∧ P x) = qp.prob {r. P (α r) ∧ Q (α r)}
    by(auto simp: hs(1) qp.qbs-l measure-def emeasure-distr, simp add: space-L)
  meson
    note sets = sets-qbs-l[OF qbs-space-monadPM[OF assms(1)],measurable-cong]
    have [simp]: density (qbs-l s) (λx. if P x then 1 else 0) = density (qbs-l s) (indicator {x∈space (qbs-to-measure X). P x})
    by(auto intro!: density-cong) (auto simp: indicator-def space-L sets-eq-imp-space-eq[OF sets])
    have p: P(x in s. P x) = measure (qbs-l (density-qbs s (λx. if P x then 1 else 0))) (qbs-space X)
    by(auto simp: qbs-l-density-qbs[OF qbs-space-monadPM[OF assms(1),qbs]])
    (auto simp: measure-restricted[of {x ∈ space (qbs-to-measure X). P x} qbs-l s,simplified sets,OF - sets.top,simplified,simplified space-L] space-L sets-eq-imp-space-eq[OF sets])
    thus ?thesis
      by(simp add: qp p cond-prob-def)
  qed
  finally show ?thesis .
  qed
  qed

```

```

lemma qbs-pmf-cond-pmf:
  fixes p :: 'a :: countable pmf
  assumes set-pmf p ∩ {x. P x} ≠ {}
  shows condition (qbs-pmf p) P = qbs-pmf (cond-pmf p {x. P x})
  proof(rule inj-onD[OF qbs-l-inj[of count-space UNIV]])

```

```

note count-space-count-space-qbs-morphism[of P, qbs]
show g1: condition (qbs-pmf p) P ∈ qbs-space (monadM-qbs (count-spaceQ UNIV))
qbs-pmf (cond-pmf p {x. P x}) ∈ qbs-space (monadM-qbs (count-spaceQ UNIV))
  by auto
show qbs-l (condition (qbs-pmf p) P) = qbs-l (qbs-pmf (cond-pmf p {x. P x}))
proof(safe intro!: measure-eqI-countable)
  fix a
  have condition (qbs-pmf p) P = normalize-qbs (density-qbs (qbs-pmf p) (λx. if
P x then 1 else 0))
    by(auto simp: condition-def query-def)
  also have ... = density-qbs (density-qbs (qbs-pmf p) (λx. if P x then 1 else
0)) (λx. 1 / emeasure (qbs-l (density-qbs (qbs-pmf p) (λx. if P x then 1 else 0)))
(qbs-space (count-spaceQ UNIV)))
proof -
  have 1:(∫+ x. ennreal (pmf p x) * (if P x then 1 else 0) ∂count-space UNIV)
= (∫+ x∈{x. P x}. ennreal (pmf p x) ∂count-space UNIV)
    by(auto intro!: nn-integral-cong)
  have ... > 0
    using assms(1) by(force intro!: nn-integral-less[of λx. 0, simplified] simp:
AE-count-space set-pmf-eq' indicator-def)
  hence 2:(∫+ x∈{x. P x}. ennreal (pmf p x) ∂count-space UNIV) ≠ 0
    by auto
  have 3:(∫+ x∈{x. P x}. ennreal (pmf p x) ∂count-space UNIV) ≠ ⊤
proof -
  have (∫+ x∈{x. P x}. ennreal (pmf p x) ∂count-space UNIV) ≤ (∫+ x.
ennreal (pmf p x) ∂count-space UNIV)
    by(auto intro!: nn-integral-mono simp: indicator-def)
  also have ... = 1
    by (simp add: nn-integral-pmf-eq-1)
  finally show ?thesis
    using ennreal-one-neq-top neq-top-trans by fastforce
qed
show ?thesis
  by(rule normalize-qbs) (auto simp: qbs-l-density-qbs[of - count-space UNIV]
emeasure-density nn-integral-measure-pmf 1 2 3)
qed
also have ... = density-qbs (qbs-pmf p) (λx. (if P x then 1 else 0) * (1 / (∫+
x. ennreal (pmf p x) * (if P x then 1 else 0) ∂count-space UNIV)))
  by(simp add: density-qbs-density-qbs-eq[of - count-space UNIV] qbs-l-density-qbs[of
- count-space UNIV] emeasure-density nn-integral-measure-pmf)
  also have ... = density-qbs (qbs-pmf p) (λx. (if P x then 1 else 0) * (1 /
(emeasure (measure-pmf p) (Collect P))))
proof -
  have [simp]: (∫+ x. ennreal (pmf p x) * (if P x then 1 else 0) ∂count-space
UNIV) = emeasure (measure-pmf p) (Collect P) (is ?l = ?r)
proof -
  have ?l = (∫+ x. ennreal (pmf p x) * (if P x then 1 else 0) ∂count-space
{x. P x})
    by(rule nn-integral-count-space-eq) auto

```

also have $\dots = ?r$
by(*auto simp: nn-integral-pmf[symmetric] intro!: nn-integral-cong*)
finally show *?thesis* .
qed
show *?thesis* **by** *simp*
qed
finally show $\text{emeasure } (qbs\text{-}l \text{ (condition } (qbs\text{-}pmf\ p) P)) \{a\} = \text{emeasure } (qbs\text{-}l \text{ (} qbs\text{-}pmf \text{ (cond-pmf } p \text{ \{x. } P \text{ x}\})) \{a\}$
by(*simp add: ennreal-divide-times qbs-l-density-qbs[of - count-space UNIV] emeasure-density cond-pmf.rep-eq[OF assms(1)]*)
qed(*auto simp: sets-qbs-l[OF g1(1)]*)
qed

5.2.1 twoUs

Example from Section 2 in [3].

definition *Uniform* $\equiv (\lambda a\ b::real. \text{uniform-qbs } lborel\text{-qbs } \{a<..**b\})**$

lemma *Uniform-qbs[qbs]*: $\text{Uniform} \in \mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q \Rightarrow_Q \text{monadM-qbs } \mathbb{R}_Q$
unfolding *Uniform-def* **by** (*rule interval-uniform-qbs*)

definition *twoUs* $:: (real \times real) \text{ qbs-measure}$ **where**
twoUs $\equiv \text{do } \{$
 $\quad \text{let } u1 = \text{Uniform } 0\ 1;$
 $\quad \text{let } u2 = \text{Uniform } 0\ 1;$
 $\quad \text{let } y = u1 \otimes_{Q\text{mes}} u2;$
 $\quad \text{condition } y (\lambda(x,y). x < 0.5 \vee y > 0.5)$
 $\quad \}$

lemma *twoUs-qbs*: $\text{twoUs} \in \text{monadM-qbs } (\mathbb{R}_Q \otimes_Q \mathbb{R}_Q)$
by(*simp add: twoUs-def*)

interpretation *rr*: *standard-borel-ne borel* $\otimes_M \text{borel} :: (real \times real) \text{ measure}$
by(*simp add: borel-prod*)

lemma *qbs-l-Uniform[simp]*: $a < b \implies \text{qbs-l } (\text{Uniform } a\ b) = \text{uniform-measure } lborel \{a<..**b\}**$
by(*simp add: standard-borel-ne.qbs-l-uniform-qbs[of borel lborel-qbs] Uniform-def*)

lemma *Uniform-qbsP*:
assumes [*arith*]: $a < b$
shows $\text{Uniform } a\ b \in \text{monadP-qbs } \mathbb{R}_Q$
by(*auto simp: monadP-qbs-def sub-qbs-space intro!: prob-space-uniform-measure*)

interpretation *UniformP-pair*: *pair-prob-space uniform-measure lborel* $\{0<..**1::real\}**$
uniform-measure lborel $\{0<..**1::real\}**$
by(*auto simp: pair-prob-space-def pair-sigma-finite-def intro!: prob-space-imp-sigma-finite prob-space-uniform-measure*)

lemma *qbs-l-Uniform-pair*: $a < b \implies \text{qbs-l } (\text{Uniform } a \ b \otimes_{Qmes} \text{Uniform } a \ b)$
 $= \text{uniform-measure lborel } \{a < .. < b\} \otimes_M \text{uniform-measure lborel } \{a < .. < b\}$
by (*auto intro!*: *qbs-l-qbs-pair-measure*[of borel borel] *standard-borel-ne.standard-borel*
simp: *qbs-l-Uniform*[*symmetric*] *simp del*: *qbs-l-Uniform*)

lemma *Uniform-pair-qbs*[*qbs*]:

assumes $a < b$

shows $\text{Uniform } a \ b \otimes_{Qmes} \text{Uniform } a \ b \in \text{qbs-space } (\text{monadP-qbs } (\mathbb{R}_Q \otimes_Q \mathbb{R}_Q))$

proof –

note [*qbs*] = *qbs-pair-measure-morphismP Uniform-qbsP*[*OF assms*]

show *?thesis*

by *simp*

qed

lemma *twoUs-prob1*: $\mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Qmes} \text{Uniform } 0 \ 1. \text{fst } z < 0.5 \vee \text{snd } z > 0.5) = 3 / 4$

proof –

have [*simp*]: $\{z \in \text{space } (\text{uniform-measure lborel } \{0 < .. < 1 :: \text{real}\} \otimes_M \text{uniform-measure lborel } \{0 < .. < 1 :: \text{real}\}). \text{fst } z * 2 < 1 \vee 1 < \text{snd } z * 2\} = \text{UNIV} \times \{1/2 < ..\} \cup \{.. < 1/2\} \times \text{UNIV}$

by (*auto simp*: *space-pair-measure*)

have $1 : \text{UniformP-pair.prob } (\text{UNIV} \times \{1 / 2 < ..\}) = 1 / 2$

proof –

have [*simp*]: $\{0 < .. < 1\} \cap \{1 / 2 < ..\} = \{1/2 < .. < 1 :: \text{real}\}$ **by** *auto*

thus *?thesis*

by (*auto simp*: *UniformP-pair.M1.measure-times*)

qed

have $2 : \text{UniformP-pair.prob } (\{.. < 1 / 2\} \times \text{UNIV} - \text{UNIV} \times \{1 / 2 < ..\}) = 1 / 4$

proof –

have [*simp*]: $\{.. < 1/2 :: \text{real}\} \times \text{UNIV} - \text{UNIV} \times \{1/2 :: \text{real} < ..\} = \{.. < 1/2\} \times \{.. 1/2\} \cup \{0 < .. < 1\} \cap \{.. < 1/2\} = \{0 < .. < 1/2 :: \text{real}\} \cup \{0 < .. < 1\} \cap \{.. 1/2 :: \text{real}\} = \{0 < .. 1/2\}$

by *auto*

show *?thesis*

by (*auto simp*: *UniformP-pair.M1.measure-times*)

qed

show *?thesis*

by (*auto simp*: *qbs-l-Uniform-pair UniformP-pair.P.finite-measure-Union' 1 2*)

qed

lemma *twoUs-prob2*: $\mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Qmes} \text{Uniform } 0 \ 1. 1/2 < \text{fst } z \wedge (\text{fst } z < 1/2 \vee \text{snd } z > 1/2)) = 1 / 4$

proof –

have [*simp*]: $\{z \in \text{space } (\text{uniform-measure lborel } \{0 < .. < 1 :: \text{real}\} \otimes_M \text{uniform-measure lborel } \{0 < .. < 1 :: \text{real}\}). 1 < \text{fst } z * 2 \wedge (\text{fst } z * 2 < 1 \vee 1 < \text{snd } z * 2)\} = \{1/2 < ..\} \times \{1/2 < ..\}$

by (*auto simp*: *space-pair-measure*)

have [simp]: $\{0 < .. < 1 :: \text{real}\} \cap \{1/2 < ..\} = \{1/2 < .. < 1\}$ **by** *auto*
show ?thesis
by(*auto simp: qbs-l-Uniform-pair UniformP-pair.M1.measure-times*)
qed

lemma *twoUs-qbs-prob*: $\text{twoUs} \in \text{qbs-space} (\text{monadP-qbs } (\mathbb{R}_Q \otimes_Q \mathbb{R}_Q))$

proof –

have $\mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Q_{\text{mes}}} \text{Uniform } 0 \ 1. \text{fst } z < 0.5 \vee \text{snd } z > 0.5) \neq 0$
unfolding *twoUs-prob1* **by** *simp*
note *qbs-morphism-space[OF condition-morphismP[of qbs-borel $\lambda x. \text{Uniform } 0 \ 1 \otimes_{Q_{\text{mes}}} \text{Uniform } 0 \ 1 \ \lambda x \ z. \text{fst } z < 0.5 \vee \text{snd } z > 0.5 \ \mathbb{R}_Q \otimes_Q \mathbb{R}_Q, \text{OF this}], \text{simplified}, \text{qbs}]$*
note *Uniform-pair-qbs[of 0 1, simplified, qbs]*
show ?thesis
by(*simp add: twoUs-def split-beta'*)
qed

lemma $\mathcal{P}((x,y) \text{ in } \text{twoUs}. 1/2 < x) = 1 / 3$

proof –

have $\mathcal{P}((x,y) \text{ in } \text{twoUs}. 1/2 < x) = \mathcal{P}(z \text{ in } \text{twoUs}. 1/2 < \text{fst } z)$
by (*simp add: split-beta'*)
also have $\dots = \mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Q_{\text{mes}}} \text{Uniform } 0 \ 1. 1/2 < \text{fst } z \mid \text{fst } z < 0.5 \vee \text{snd } z > 0.5)$
by(*simp add: twoUs-def split-beta', rule query-Bayes[OF Uniform-pair-qbs[of 0 1, simplified, qbs]]) auto*)
also have $\dots = \mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Q_{\text{mes}}} \text{Uniform } 0 \ 1. 1/2 < \text{fst } z \wedge (\text{fst } z < 1/2 \vee \text{snd } z > 1/2)) / \mathcal{P}(z \text{ in } \text{Uniform } 0 \ 1 \otimes_{Q_{\text{mes}}} \text{Uniform } 0 \ 1. \text{fst } z < 0.5 \vee \text{snd } z > 0.5)$
by(*simp add: cond-prob-def*)
also have $\dots = 1 / 3$
by(*simp only: twoUs-prob2 twoUs-prob1*) *simp*
finally show ?thesis .
qed

5.2.2 Two Dice

Example from Adrian [2, Sect. 2.3].

abbreviation *die* \equiv *qbs-pmf (pmf-of-set {Suc 0..6})*

lemma *die-qbs[qbs]*: $\text{die} \in \text{monadM-qbs } \mathbb{N}_Q$

by *simp*

definition *two-dice* :: *nat qbs-measure* **where**

two-dice \equiv *do* {
 let die1 = die;
 let die2 = die;
 let twodice = die1 $\otimes_{Q_{\text{mes}}}$ *die2;*
 (x,y) ← condition twodice
 ($\lambda(x,y). x = 4 \vee y = 4$);


```

    return-qbs  $\mathbf{N}_Q$  (x + y)
  }

```

lemma *two-dice-qbs*: $\text{two-dice} \in \text{monadM-qbs } \mathbf{N}_Q$
 by(*simp add: two-dice-def*)

lemma *prob-die2*: $\mathcal{P}(x \text{ in qbs-l } (\text{die} \otimes_{Q_{mes}} \text{die}). P x) = \text{real } (\text{card } (\{x. P x\} \cap (\{1..6\} \times \{1..6\}))) / 36$ (is ?P = ?rhs)

proof –

have ?P = *measure-pmf.prob* (*pair-pmf* (*pmf-of-set* {Suc 0..6}) (*pmf-of-set* {Suc 0..6})) {x. P x}

by(*auto simp: qbs-pair-pmf*)

also have ... = *measure-pmf.prob* (*pair-pmf* (*pmf-of-set* {Suc 0..6}) (*pmf-of-set* {Suc 0..6})) ({x. P x} \cap *set-pmf* (*pair-pmf* (*pmf-of-set* {Suc 0..6}) (*pmf-of-set* {Suc 0..6})))

by(*rule measure-Int-set-pmf[symmetric]*)

also have ... = *measure-pmf.prob* (*pair-pmf* (*pmf-of-set* {Suc 0..6}) (*pmf-of-set* {Suc 0..6})) ({x. P x} \cap ({Suc 0..6} \times {Suc 0..6}))

by *simp*

also have ... = ($\sum z \in \{x. P x\} \cap (\{Suc 0..6\} \times \{Suc 0..6\}).$ *pmf* (*pair-pmf* (*pmf-of-set* {Suc 0..6}) (*pmf-of-set* {Suc 0..6}))) z

by(*simp add: measure-measure-pmf-finite*)

also have ... = ($\sum z \in \{x. P x\} \cap (\{Suc 0..6\} \times \{Suc 0..6\}).$ 1 / 36)

by(*rule Finite-Cartesian-Product.sum-cong-aux*) (*auto simp: pmf-pair*)

also have ... = ?rhs

by *auto*

finally show ?thesis .

qed

lemma *dice-prob1*: $\mathcal{P}(z \text{ in qbs-l } (\text{die} \otimes_{Q_{mes}} \text{die}). \text{fst } z = 4 \vee \text{snd } z = 4) = 11 / 36$

proof –

have 1: *Restr* {z. $\text{fst } z = 4 \vee \text{snd } z = 4$ } {Suc 0..6::nat} = {Suc 0..Suc (Suc (Suc (Suc (Suc 0))))} \times {Suc (Suc (Suc (Suc 0)))} \cup {Suc (Suc (Suc (Suc 0)))} \times {Suc 0..(Suc (Suc (Suc 0)))} \cup {Suc (Suc (Suc (Suc 0)))} \times {Suc (Suc (Suc (Suc 0)))}..Suc (Suc (Suc (Suc (Suc (Suc 0))))}

by *fastforce*

have *card* ... = *card* ({Suc 0..Suc (Suc (Suc (Suc (Suc (Suc 0))))}) \times {Suc (Suc (Suc (Suc 0)))} \cup {Suc (Suc (Suc (Suc 0)))} \times {Suc 0..(Suc (Suc (Suc 0)))}) + *card* ({Suc (Suc (Suc (Suc 0)))} \times {Suc (Suc (Suc (Suc (Suc (Suc 0))))}..Suc (Suc (Suc (Suc (Suc 0))))})

by(*rule card-Un-disjnt*) (*auto simp: disjnt-def*)

also have ... = *card* ({Suc 0..Suc (Suc (Suc (Suc (Suc (Suc 0))))}) \times {Suc (Suc (Suc (Suc 0)))}) + *card* ({Suc (Suc (Suc (Suc 0)))} \times {Suc 0..(Suc (Suc (Suc 0)))}) + *card* ({Suc (Suc (Suc (Suc 0)))} \times {Suc (Suc (Suc (Suc (Suc (Suc 0))))}..Suc (Suc (Suc (Suc (Suc 0))))})

proof –

have *card* ({Suc 0..Suc (Suc (Suc (Suc (Suc (Suc 0))))}) \times {Suc (Suc (Suc (Suc 0)))} \cup {Suc (Suc (Suc (Suc 0)))} \times {Suc 0..(Suc (Suc (Suc 0)))}) = *card*

```

({Suc 0..Suc (Suc (Suc (Suc (Suc (Suc 0))))))} × {Suc (Suc (Suc (Suc 0)))} +
card ({Suc (Suc (Suc (Suc 0)))} × {Suc 0..(Suc (Suc (Suc 0)))})
  by(rule card-Un-disjnt) (auto simp: disjnt-def)
  thus ?thesis by simp
qed
also have ... = 11 by auto
finally show ?thesis
  by(auto simp: prob-die2 1)
qed

```

lemma *dice-program-prob*: $\mathcal{P}(x \text{ in two-dice. } P x) = 2 * (\sum_{n \in \{5,6,7,9,10\}} \text{of-bool } (P n) / 11) + \text{of-bool } (P 8) / 11$ (is $?P = ?rp$)

proof –

have $0: (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\}) = \{5,6,7,8,9,10\}$

proof *safe*

show $5 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(1,4)$])

show $6 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(2,4)$])

show $7 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(3,4)$])

show $8 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(4,4)$])

show $9 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(5,4)$])

show $10 \in (\bigcup x \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{(x, y). x = 4 \vee y = 4\}. \{\text{fst } x + \text{snd } x\})$

by(*auto intro!*: *bexI*[**where** $x=(6,4)$])

qed *auto*

have $1: \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{x. \text{fst } x = 4 \vee \text{snd } x = 4\} \neq \{\}$

proof –

have $(1,4) \in \{\text{Suc } 0..6\} \times \{\text{Suc } 0..6\} \cap \{x. \text{fst } x = 4 \vee \text{snd } x = 4\}$

by *auto*

thus ?thesis by *blast*

qed

hence $2: \text{set-pmf } (\text{pair-pmf } (\text{pmf-of-set } \{\text{Suc } 0..6\}) (\text{pmf-of-set } \{\text{Suc } 0..6\})) \cap \{(x, y). x = 4 \vee y = 4\} \neq \{\}$

by(*auto simp*: *split-beta'*)

have *ceq:condition* (*die* $\otimes_{Q_{mes}}$ *die*) $(\lambda(x,y). x = 4 \vee y = 4) = \text{qbs-pmf } (\text{cond-pmf } (\text{pair-pmf } (\text{pmf-of-set } \{\text{Suc } 0..6\}) (\text{pmf-of-set } \{\text{Suc } 0..6\})) \{(x,y). x = 4 \vee y = 4\})$

by(*auto simp*: *split-beta'* *qbs-pair-pmf 1 intro!*: *qbs-pmf-cond-pmf*)

have *two-dice* = *condition* (*die* $\otimes_{Q_{mes}}$ *die*) ($\lambda(x,y). x = 4 \vee y = 4$) \ggg ($\lambda(x,y).$
return-qbs \mathbf{N}_Q ($x + y$))

by(*simp add: two-dice-def*)

also have ... = *qbs-pmf* (*cond-pmf* (*pair-pmf* (*pmf-of-set* {*Suc* 0..6}) (*pmf-of-set*
 {*Suc* 0..6})) ($\lambda(x,y). x = 4 \vee y = 4$)) \ggg ($\lambda z.$ *qbs-pmf* (*return-pmf* (*fst* $z + \text{snd}$
 z)))

by(*simp add: ceq*) (*simp add: qbs-pmf-return-pmf split-beta'*)

also have ... = *qbs-pmf* (*cond-pmf* (*pair-pmf* (*pmf-of-set* {*Suc* 0..6}) (*pmf-of-set*
 {*Suc* 0..6})) ($\lambda(x,y). x = 4 \vee y = 4$)) \ggg ($\lambda z.$ *return-pmf* (*fst* $z + \text{snd}$
 z)))

by(*rule qbs-pmf-bind-pmf[symmetric]*)

finally have *two-dice-eq:two-dice* = *qbs-pmf* (*cond-pmf* (*pair-pmf* (*pmf-of-set*
 {*Suc* 0..6}) (*pmf-of-set* {*Suc* 0..6})) ($\lambda(x,y). x = 4 \vee y = 4$)) \ggg ($\lambda z.$ *return-pmf*
 (*fst* $z + \text{snd}$ z))) .

have *3:measure-pmf.prob* (*pair-pmf* (*pmf-of-set* {*Suc* 0..6}) (*pmf-of-set* {*Suc*
 0..6})) ($\lambda(x, y). x = 4 \vee y = 4$) = 11 / 36

using *dice-prob1* **by**(*auto simp: split-beta' qbs-pair-pmf*)

have *?P* = *measure-pmf.prob* (*cond-pmf* (*pair-pmf* (*pmf-of-set* {*Suc* 0..6})
 (*pmf-of-set* {*Suc* 0..6})) ($\lambda(x, y). x = 4 \vee y = 4$)) \ggg ($\lambda z.$ *return-pmf* (*fst* z
 + *snd* z)) (*is* - = *measure-pmf.prob ?bind* -)

by(*simp add: two-dice-eq*)

also have ... = *measure-pmf.prob ?bind* ($\{x. P\ x\} \cap \text{set-pmf } ?\text{bind}$)

by(*rule measure-Int-set-pmf[symmetric]*)

also have ... = *sum* (*pmf ?bind*) ($\{x. P\ x\} \cap \text{set-pmf } ?\text{bind}$)

by(*rule measure-measure-pmf-finite*) (*auto simp: set-cond-pmf[OF 2]*)

also have ... = *sum* (*pmf ?bind*) ($\{x. P\ x\} \cap \{5, 6, 7, 8, 9, 10\}$)

by(*auto simp: set-cond-pmf[OF 2] 0*)

also have ... = ($\sum n \in \{n. P\ n\} \cap \{5, 6, 7, 8, 9, 10\}.$ *measure-pmf.expectation*
 (*cond-pmf* (*pair-pmf* (*pmf-of-set* {*Suc* 0..6}) (*pmf-of-set* {*Suc* 0..6})) ($\lambda(x, y). x$
 $= 4 \vee y = 4$)) ($\lambda x.$ *indicat-real* { n } (*fst* $x + \text{snd}$ x))) (*is* - = ($\sum - \in -.$ *mea-*
sure-pmf.expectation ?cond -))

by(*simp add: pmf-bind*)

also have ... = ($\sum n \in \{n. P\ n\} \cap \{5, 6, 7, 8, 9, 10\}.$ ($\sum m \in \{(1,4),(2,4),(3,4),(4,4),(5,4),(6,4),(4,1),(4,2),$
indicat-real { n } (*fst* $m + \text{snd}$ m) * *pmf ?cond* m))

proof(*intro Finite-Cartesian-Product.sum-cong-aux integral-measure-pmf-real*)

fix $n\ m$

assume $h:n \in \{n. P\ n\} \cap \{5, 6, 7, 8, 9, 10\}$ $m \in \text{set-pmf } ?\text{cond } \text{indicat-real}$
 $\{n\}$ (*fst* $m + \text{snd}$ m) $\neq 0$

then have $nm:\text{fst } m + \text{snd } m = n$

by(*auto simp: indicator-def*)

have $m:\text{fst } m \neq 0$ $\text{snd } m \neq 0$ $\text{fst } m = 4 \vee \text{snd } m = 4$

using $h(2)$ **by**(*auto simp: set-cond-pmf[OF 2]*)

show $m \in \{(1, 4), (2, 4), (3, 4), (4,4), (5, 4), (6, 4), (4, 1), (4, 2), (4, 3),$
 $(4, 5), (4, 6)\}$

using $h(1)$ $nm\ m$ **by**(*auto,metis prod.collapse*)+

qed *simp*

also have ... = ($\sum n \in \{n. P\ n\} \cap \{5, 6, 7, 8, 9, 10\}.$ ($\sum m \in \{(1,4),(2,4),(3,4),(4,4),(5,4),(6,4),(4,1),(4,2),$
indicat-real { n } (*fst* $m + \text{snd}$ m) * 1 / 11))

```

proof(rule Finite-Cartesian-Product.sum-cong-aux[OF Finite-Cartesian-Product.sum-cong-aux])
  fix n m
  assume h:n ∈ {n. P n} ∩ {5, 6, 7, 8, 9, 10} m ∈ {(1,4),(2,4),(3,4),(4,4),(5,4),(6,4),(4,1),(4,2),(4,3),(4,4),(4,5)}
  have pmf ?cond m = 1 / 11
    using h(2) by(auto simp add: pmf-cond[OF 2] 3 pmf-pair)
  thus indicat-real {n} (fst m + snd m) * pmf ?cond m = indicat-real {n} (fst
m + snd m) * 1 / 11
    by simp
  qed
  also have ... = ?rp
    by fastforce
  finally show ?thesis .
qed

```

corollary

```

 $\mathcal{P}(x \text{ in two-dice. } x = 5) = 2 / 11$ 
 $\mathcal{P}(x \text{ in two-dice. } x = 6) = 2 / 11$ 
 $\mathcal{P}(x \text{ in two-dice. } x = 7) = 2 / 11$ 
 $\mathcal{P}(x \text{ in two-dice. } x = 8) = 1 / 11$ 
 $\mathcal{P}(x \text{ in two-dice. } x = 9) = 2 / 11$ 
 $\mathcal{P}(x \text{ in two-dice. } x = 10) = 2 / 11$ 

```

unfolding dice-program-prob **by** simp-all

5.2.3 Gaussian Mean Learning

Example from Sato et al. Section 8. 2 in [3].

definition Gauss \equiv ($\lambda \mu \sigma$. density-qbs lborel_Q (normal-density $\mu \sigma$))

lemma Gauss-qbs[qbs]: Gauss $\in \mathbf{R}_Q \Rightarrow_Q \mathbf{R}_Q \Rightarrow_Q$ monadM-qbs \mathbf{R}_Q
by(simp add: Gauss-def)

primrec GaussLearn' :: [real, real qbs-measure, real list]
 \Rightarrow real qbs-measure **where**

```

  GaussLearn' - p [] = p
| GaussLearn'  $\sigma$  p (y#ls) = query (GaussLearn'  $\sigma$  p ls)
  (normal-density y  $\sigma$ )

```

lemma GaussLearn'-qbs[qbs]: GaussLearn' $\in \mathbf{R}_Q \Rightarrow_Q$ monadM-qbs $\mathbf{R}_Q \Rightarrow_Q$ list-qbs
 $\mathbf{R}_Q \Rightarrow_Q$ monadM-qbs \mathbf{R}_Q
by(simp add: GaussLearn'-def)

context

```

  fixes  $\sigma$  :: real
  assumes [arith]:  $\sigma > 0$ 
begin

```

abbreviation GaussLearn \equiv GaussLearn' σ

lemma *GaussLearn-qbs*[qbs]: $GaussLearn \in qbs\text{-space } (monadM\text{-qbs } \mathbb{R}_Q \Rightarrow_Q list\text{-qbs } \mathbb{R}_Q \Rightarrow_Q monadM\text{-qbs } \mathbb{R}_Q)$
by *simp*

definition *Total* :: $real\ list \Rightarrow real$ **where** $Total = (\lambda l. foldr (+) l 0)$

lemma *Total-simp*: $Total [] = 0$ $Total (y\#ls) = y + Total\ ls$
by(*simp-all add: Total-def*)

lemma *Total-qbs*[qbs]: $Total \in list\text{-qbs } \mathbb{R}_Q \rightarrow_Q \mathbb{R}_Q$
by(*simp add: Total-def*)

lemma *GaussLearn-Total*:
assumes [*arith*]: $\xi > 0$ $n = length\ L$
shows $GaussLearn\ (Gauss\ \delta\ \xi)\ L = Gauss\ ((Total\ L * \xi^2 + \delta * \sigma^2) / (n * \xi^2 + \sigma^2))\ (sqrt\ ((\xi^2 * \sigma^2) / (n * \xi^2 + \sigma^2)))$
using *assms(2)*
proof(*induction L arbitrary: n*)
case *Nil*
then show *?case*
by(*simp add: Total-def*)
next
case *ih:(Cons a L)*
then obtain n' **where** $n':n = Suc\ n'$ $n' = length\ L$
by *auto*
have $1:\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) > 0$
by(*auto intro!: divide-pos-pos add-nonneg-pos*)
have $\sigma:$ ($sqrt\ (\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2)) * \sigma / sqrt\ (\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) + \sigma^2) = (sqrt\ (\xi^2 * \sigma^2 / (real\ n * \xi^2 + \sigma^2)))$)
proof(*rule power2-eq-imp-eq*)
show $(sqrt\ (\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2)) * \sigma / sqrt\ (\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) + \sigma^2))^2 = (sqrt\ (\xi^2 * \sigma^2 / (real\ n * \xi^2 + \sigma^2)))^2$ (**is** *?lhs = ?rhs*)
proof -
have *?lhs* = $(\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2)) * (\sigma^2 / (\xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) + \sigma^2))$
by (*simp add: power-divide power-mult-distrib*)
also have $\dots = \xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) * (\sigma^2 / ((\xi^2 / (real\ n' * \xi^2 + \sigma^2) + 1) * \sigma^2))$
by (*simp add: distrib-left mult.commute*)
also have $\dots = \xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) * (1 / (\xi^2 / (real\ n' * \xi^2 + \sigma^2) + 1))$
by *simp*
also have $\dots = \xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) * (1 / ((\xi^2 + (real\ n' * \xi^2 + \sigma^2)) / (real\ n' * \xi^2 + \sigma^2)))$
by(*simp only: add-divide-distrib[of ξ^2]*) *auto*
also have $\dots = \xi^2 * \sigma^2 / (real\ n' * \xi^2 + \sigma^2) * ((real\ n' * \xi^2 + \sigma^2) / (\xi^2 + (real\ n' * \xi^2 + \sigma^2)))$
by *simp*

also have ... = $\xi^2 * \sigma^2 / (\xi^2 + (\text{real } n' * \xi^2 + \sigma^2))$
using 1 by force
also have ... = ?rhs
by(simp add: n'(1) distrib-right)
finally show ?thesis .
qed
qed simp-all
have mu: $((\text{Total } L * \xi^2 + \delta * \sigma^2) * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) + a * (\xi^2 * \sigma^2) / (\text{real } n' * \xi^2 + \sigma^2)) / (\xi^2 * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) + \sigma^2) = ((a + \text{Total } L) * \xi^2 + \delta * \sigma^2) / (\text{real } n * \xi^2 + \sigma^2)$ (is ?lhs = ?rhs)
proof -
have ?lhs = $((\text{Total } L * \xi^2 + \delta * \sigma^2) * \sigma^2 + a * (\xi^2 * \sigma^2)) / (\text{real } n' * \xi^2 + \sigma^2) / (\xi^2 * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) + \sigma^2)$
by(simp add: add-divide-distrib)
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) / (\xi^2 * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) + \sigma^2)$
by (simp add: distrib-left mult.commute)
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) * \sigma^2 / (\text{real } n' * \xi^2 + \sigma^2) / ((\xi^2 * \sigma^2 + (\text{real } n' * \xi^2 + \sigma^2) * \sigma^2) / (\text{real } n' * \xi^2 + \sigma^2))$
by (simp add: add-divide-distrib)
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) * \sigma^2 / (\xi^2 * \sigma^2 + (\text{real } n' * \xi^2 + \sigma^2) * \sigma^2)$
using 1 by auto
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) * \sigma^2 / ((\xi^2 + (\text{real } n' * \xi^2 + \sigma^2)) * \sigma^2)$
by(simp only: distrib-right)
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) / (\xi^2 + (\text{real } n' * \xi^2 + \sigma^2))$
by simp
also have ... = $((\text{Total } L * \xi^2 + \delta * \sigma^2) + a * \xi^2) / (\text{real } n * \xi^2 + \sigma^2)$
by(simp add: n'(1) distrib-right)
also have ... = ?rhs
by (simp add: distrib-right)
finally show ?thesis .
qed
show ?case
by(simp add: ih(1)[OF n'(2)]) (simp add: query-def qbs-normal-posterior[OF real-sqrt-gt-zero[OF 1]] Gauss-def Total-simp sigma mu)
qed

lemma GaussLearn-KL-divergence-lem1:

fixes a :: real
assumes [arith]: a > 0 b > 0 c > 0 d > 0
shows $(\lambda n. \ln ((b * (n * d + c)) / (d * (n * b + a)))) \longrightarrow 0$
proof -
have $(\lambda n::nat. \ln ((b * (\text{Suc } n * d + c)) / (d * (\text{Suc } n * b + a)))) = (\lambda n. \ln ((b * (d + c / \text{Suc } n)) / (d * (b + a / \text{Suc } n))))$
proof
fix n
show $\ln (b * (\text{real } (\text{Suc } n) * d + c) / (d * (\text{real } (\text{Suc } n) * b + a))) = \ln (b *$

$(d + c / \text{real } (\text{Suc } n)) / (d * (b + a / \text{real } (\text{Suc } n)))$ (is ln ?l = ln ?r)
proof –
have ?l = $b * (d + c / \text{real } (\text{Suc } n)) / (d * (b + a / \text{real } (\text{Suc } n))) * (\text{Suc } n / \text{Suc } n)$
unfolding times-divide-times-eq distrib-left distrib-right **by** (simp add: mult.assoc mult.commute)
also have ... = ?r **by** simp
finally show ?thesis **by** simp
qed
qed
also have ... $\longrightarrow 0$
apply(rule tendsto-eq-intros(33)[of - 1])
apply(rule Topological-Spaces.tendsto-eq-intros(25)[of - b * d - - b * d, OF LIMSEQ-Suc[OF Topological-Spaces.tendsto-eq-intros(18)[of - b - - d] LIMSEQ-Suc[OF Topological-Spaces.tendsto-eq-intros(18)[of - d - - b]]])
apply(intro Topological-Spaces.tendsto-eq-intros | auto)+
done
finally show ?thesis
by(rule LIMSEQ-imp-Suc)
qed

lemma GaussLearn-KL-divergence-lem1':

fixes b :: real
assumes [arith]: $b > 0 \ d > 0 \ s > 0$
shows $(\lambda n. \ln (\text{sqrt } (b^2 * s^2 / (\text{real } n * b^2 + s^2)) / \text{sqrt } (d^2 * s^2 / (\text{real } n * d^2 + s^2)))) \longrightarrow 0$ (is ?f $\longrightarrow 0$)
proof –
have ?f = $(\lambda n. \ln (\text{sqrt } ((b^2 * (n * d^2 + s^2)) / (d^2 * (n * b^2 + s^2))))$
by(simp add: real-sqrt-divide real-sqrt-mult mult.commute)
also have ... = $(\lambda n. \ln ((b^2 * (n * d^2 + s^2)) / (d^2 * (n * b^2 + s^2)))) / 2$
by (standard, rule ln-sqrt) (auto intro!: divide-pos-pos mult-pos-pos add-nonneg-pos)
also have ... $\longrightarrow 0$
using GaussLearn-KL-divergence-lem1 **by** auto
finally show ?thesis .
qed

lemma GaussLearn-KL-divergence-lem2:

fixes s :: real
assumes [arith]: $s > 0 \ b > 0 \ d > 0$
shows $(\lambda n. ((d * s) / (n * d + s)) / (2 * ((b * s) / (n * b + s)))) \longrightarrow 1 / 2$
proof –
have $(\lambda n::\text{nat}. ((d * s) / (\text{Suc } n * d + s)) / (2 * ((b * s) / (\text{Suc } n * b + s))))$
= $(\lambda n. (d * b + d * s / \text{Suc } n) / (2 * b * d + 2 * b * s / \text{Suc } n))$
proof
fix n
show $d * s / (\text{real } (\text{Suc } n) * d + s) / (2 * (b * s / (\text{real } (\text{Suc } n) * b + s))) =$
 $(d * b + d * s / \text{real } (\text{Suc } n)) / (2 * b * d + 2 * b * s / \text{real } (\text{Suc } n))$ (is ?l = ?r)
proof –

```

have ?l = d * (Suc n * b + s) / ((2 * b) * (Suc n * d + s))
  by(simp add: divide-divide-times-eq)
also have ... = d * (b + s / Suc n) / ((2 * b) * (d + s / Suc n)) * (Suc n /
Suc n)
  proof -
    have 1:(2 * b * d * real (Suc n) + 2 * b * (s / real (Suc n)) * real (Suc
n))= (2 * b) * (Suc n * d + s)
    unfolding distrib-left distrib-right by(simp add: mult.assoc mult.commute)
    show ?thesis
    unfolding times-divide-times-eq distrib-left distrib-right 1
    by (simp add: mult.assoc mult.commute)
  qed
also have ... = ?r
  by(auto simp: distrib-right distrib-left mult.commute)
finally show ?thesis .
qed
qed
also have ...  $\longrightarrow$  1 / 2
  by(rule Topological-Spaces.tendsto-eq-intros(25)[of - d * b - - 2 * b * d, OF
LIMSEQ-Suc LIMSEQ-Suc]) (intro Topological-Spaces.tendsto-eq-intros | auto)+
  finally show ?thesis
  by(rule LIMSEQ-imp-Suc)
qed

```

lemma GaussLearn-KL-divergence-lem2':

```

fixes s :: real
assumes [arith]: s > 0 b > 0 d > 0
shows ( $\lambda n. ((d^2 * s^2) / (n * d^2 + s^2)) / (2 * ((b^2 * s^2) / (n * b^2 +
s^2))) - 1 / 2$ )  $\longrightarrow$  0
using GaussLearn-KL-divergence-lem2[of s^2 b^2 d^2]
by(rule LIM-zero) auto

```

lemma GaussLearn-KL-divergence-lem3:

```

fixes a b c d s K L :: real
assumes [arith]: b > 0 d > 0 s > 0
shows ((K * d + c * s) / (n * d + s) - (L * b + a * s) / (n * b + s))^2 / (2
* ((b * s) / (n * b + s))) = ((((((K - L) * d * b * real n + c * s * b * real n +
K * d * s + c * s * s) - a * s * d * real n - L * b * s - a * s * s))^2 / (d * d *
b * (real n * real n * real n) + s * s * b * real n + 2 * d * s * b * (real n * real
n) + d * d * (real n * real n) * s + s * s * s + 2 * d * s * s * real n))) / (2 * (b
* s)) (is ?lhs = ?rhs)
proof -
  have 0:real n * d + s > 0 real n * b + s > 0
  by(auto intro!: add-nonneg-pos)
  hence 1:real n * d + s  $\neq$  0 real n * b + s  $\neq$  0 by simp-all
  have ?lhs = (((K * d + c * s) * (n * b + s) - (L * b + a * s) * (n * d + s))
/ ((n * d + s) * (n * b + s)))^2 / (2 * (b * s / (n * b + s)))
  unfolding diff-frac-eq[OF 1] by simp
  also have ... = ((((((K * d + c * s) * (n * b + s) - (L * b + a * s) * (n * d +

```


$s)))^2 / ((n * d + s)^2 * (n * b + s))) / (2 * (b * s))$
by(*auto simp: power2-eq-square*)
also have ... = $(((((K * d * (n * b) + c * s * (n * b) + K * d * s + c * s * s) - ((L * b * (n * d) + a * s * (n * d) + L * b * s + a * s * s)))^2 / ((n * d)^2 * (n * b) + s^2 * (n * b) + 2 * (n * d) * s * (n * b) + (n * d)^2 * s + s^2 * s + 2 * (n * d) * s * s))) / (2 * (b * s))$
by(*simp add: power2-sum distrib-left distrib-right is-num-normalize(1)*)
also have ... = $(((((K * d * b * real n + c * s * b * real n + K * d * s + c * s * s) - ((L * b * d * real n + a * s * d * real n + L * b * s + a * s * s)))^2 / (d * d * b * (real n * real n * real n) + s * s * b * real n + 2 * d * s * b * (real n * real n) + d * d * (real n * real n) * s + s * s * s + 2 * d * s * s * real n))) / (2 * (b * s))$
by (*simp add: mult.commute mult.left-commute power2-eq-square*)
also have ... = $(((((K - L) * d * b * real n + c * s * b * real n + K * d * s + c * s * s) - ((a * s * d * real n + L * b * s + a * s * s)))^2 / (d * d * b * (real n * real n * real n) + s * s * b * real n + 2 * d * s * b * (real n * real n) + d * d * (real n * real n) * s + s * s * s + 2 * d * s * s * real n))) / (2 * (b * s))$
proof -
have 1: $K * d * b * real n + c * s * b * real n + K * d * s + c * s * s - (L * b * d * real n + a * s * d * real n + L * b * s + a * s * s) = (K - L) * d * b * real n + c * s * b * real n + K * d * s + c * s * s - (a * s * d * real n + L * b * s + a * s * s)$
by (*simp add: left-diff-distrib*)
show ?thesis
unfolding 1 ..
qed
also have ... = ?rhs
by (*simp add: diff-diff-eq*)
finally show ?thesis .
qed

lemma *GaussLearn-KL-divergence-lem4*:

fixes $a b c d s K L :: real$
assumes [*arith*]: $b > 0 d > 0 s > 0$
shows $(\lambda n. (|c * s * b * real n| + |K * (real n) * d * s| + |c * s * s| + |a * s * d * real n| + |L * (real n) * b * s| + |a * s * s|)^2 / (d * d * b * (real n * real n * real n) + s * s * b * real n + 2 * d * s * b * (real n * real n) + d * d * (real n * real n) * s + s * s * s + 2 * d * s * s * real n) / (2 * (b * s))) \longrightarrow 0$ (**is** $(\lambda n. ?f n) \longrightarrow 0$)
proof -
have $t1: (\lambda n. x / (real n * real n)) \longrightarrow 0$ **for** x
proof -
have $(\lambda n. x / (real n * real n)) = (\lambda n. x / (real n) * (1 / real n))$
by *simp*
also have ... $\longrightarrow 0$
by (*intro Topological-Spaces.tendsto-eq-intros | auto*) +
finally show ?thesis .
qed
have $t4: (\lambda n. x / (real n * real n * real n)) \longrightarrow 0$ **for** x

proof –
have $(\lambda n. x / (\text{real } n * \text{real } n * \text{real } n)) = (\lambda n. x / (\text{real } n) * (1 / \text{real } n) * (1 / \text{real } n))$
by *simp*
also have ... $\longrightarrow 0$
by (*intro Topological-Spaces.tendsto-eq-intros | auto*) +
finally show *?thesis* .
qed
have $t2[\text{tendsto-intros}]: (\lambda n. x / (\text{sqrt } n)) \longrightarrow 0$ **for** x
by(*rule power-tendsto-0-iff[of 2, THEN iffD1], simp-all add: power2-eq-square*)
(*intro Topological-Spaces.tendsto-eq-intros | auto*) +
have $t3: (\lambda n. x / (\text{sqrt } n * \text{real } n)) \longrightarrow 0$ **for** x
proof –
have $(\lambda n. x / (\text{sqrt } n * \text{real } n)) = (\lambda n. x / \text{sqrt } n * (1 / n))$ **by** *simp*
also have ... $\longrightarrow 0$
by (*intro Topological-Spaces.tendsto-eq-intros | auto*) +
finally show *?thesis* .
qed

have $(\lambda n. ?f (\text{Suc } n)) = (\lambda n. (|(c * s * b) / \text{sqrt } (\text{real } (\text{Suc } n))| + |(K * d * s) / \text{sqrt } (\text{real } (\text{Suc } n))| + |(c * s * s) / (\text{sqrt } (\text{Suc } n) * \text{real } (\text{Suc } n))| + |(a * s * d) / \text{sqrt } (\text{real } (\text{Suc } n))| + |(L * b * s) / \text{sqrt } (\text{real } (\text{Suc } n))| + |(a * s * s) / (\text{sqrt } (\text{Suc } n) * \text{real } (\text{Suc } n))|^2 / ((d * d * b + (s * s * b) / (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n))) + (2 * d * s * b) / \text{real } (\text{Suc } n) + (d * d * s) / \text{real } (\text{Suc } n) + (s * s * s) / (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) + (2 * d * s * s) / (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)))) / (2 * (b * s)))$ (**is** - = $(\lambda n. ?g (\text{Suc } n))$)
proof
fix n
show $?f (\text{Suc } n) = ?g (\text{Suc } n)$ (**is** *?lhs = ?rhs*)
proof –
have $?lhs = (|c * s * b * \text{real } (\text{Suc } n)| + |K * d * s * \text{real } (\text{Suc } n)| + |c * s * s| + |a * s * d * \text{real } (\text{Suc } n)| + |L * b * s * \text{real } (\text{Suc } n)| + |a * s * s|^2 / (d * d * b * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) + s * s * b * \text{real } (\text{Suc } n) + 2 * d * s * b * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) + d * d * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) * s + s * s * s + 2 * d * s * s * \text{real } (\text{Suc } n)) / (2 * (b * s)))$
proof –
have $1: K * \text{real } (\text{Suc } n) * d * s = K * d * s * \text{real } (\text{Suc } n) L * \text{real } (\text{Suc } n) * b * s = L * b * s * \text{real } (\text{Suc } n)$
by *auto*
show *?thesis*
unfolding 1 ..
qed
also have ... = $((|c * s * b / \text{sqrt } (\text{real } (\text{Suc } n))| + |K * d * s / \text{sqrt } (\text{real } (\text{Suc } n))| + |(c * s * s) / (\text{sqrt } (\text{Suc } n) * \text{real } (\text{Suc } n))| + |a * s * d / \text{sqrt } (\text{real } (\text{Suc } n))| + |L * b * s / \text{sqrt } (\text{real } (\text{Suc } n))| + |(a * s * s) / (\text{sqrt } (\text{Suc } n) * \text{real } (\text{Suc } n))|^2 / ((d * d * b * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) + s * s * b * \text{real } (\text{Suc } n) + 2 * d * s * b * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) + d * d * (\text{real } (\text{Suc } n) * \text{real } (\text{Suc } n)) * s + s * s * s + 2 * d * s * s * \text{real } (\text{Suc } n)) / (2 * (b * s)))$

by(*simp add: distrib-right left-diff-distrib mult.assoc[symmetric] abs-mult[of - real (Suc n)] del: of-nat-Suc*)
also have ... = $((|c * s * b / \text{sqrt}(\text{real}(Suc\ n))| + |K * d * s / \text{sqrt}(\text{real}(Suc\ n))| + |(c * s * s) / (\text{sqrt}(Suc\ n) * \text{real}(Suc\ n))| + |a * s * d / \text{sqrt}(\text{real}(Suc\ n))| + |L * b * s / \text{sqrt}(\text{real}(Suc\ n))| + |(a * s * s) / (\text{sqrt}(Suc\ n) * \text{real}(Suc\ n))|)^2 * (\text{real}(Suc\ n) * \text{real}(Suc\ n) * \text{real}(Suc\ n))) / (d * d * b * (\text{real}(Suc\ n) * \text{real}(Suc\ n) * \text{real}(Suc\ n)) + s * s * b * \text{real}(Suc\ n) + 2 * d * s * b * (\text{real}(Suc\ n) * \text{real}(Suc\ n)) + d * d * (\text{real}(Suc\ n) * \text{real}(Suc\ n)) * s + s * s * s + 2 * d * s * s * \text{real}(Suc\ n)) / (2 * (b * s))$
by(*simp add: power2-eq-square*)
also have ... = $((|c * s * b / \text{sqrt}(\text{real}(Suc\ n))| + |K * d * s / \text{sqrt}(\text{real}(Suc\ n))| + |(c * s * s) / (\text{sqrt}(Suc\ n) * \text{real}(Suc\ n))| + |a * s * d / \text{sqrt}(\text{real}(Suc\ n))| + |L * b * s / \text{sqrt}(\text{real}(Suc\ n))| + |(a * s * s) / (\text{sqrt}(Suc\ n) * \text{real}(Suc\ n))|)^2 / ((d * d * b * (\text{real}(Suc\ n) * \text{real}(Suc\ n) * \text{real}(Suc\ n)) + s * s * b * \text{real}(Suc\ n) + 2 * d * s * b * (\text{real}(Suc\ n) * \text{real}(Suc\ n)) + d * d * (\text{real}(Suc\ n) * \text{real}(Suc\ n)) * s + s * s * s + 2 * d * s * s * \text{real}(Suc\ n)) / (\text{real}(Suc\ n) * \text{real}(Suc\ n) * \text{real}(Suc\ n)))) / (2 * (b * s))$
by *simp*
also have ... = ?*rhs*
by(*simp add: add-divide-distrib*)
finally show ?*thesis* .
qed
qed
also have ... \longrightarrow 0
apply(*rule LIMSEQ-Suc*)
apply(*rule Topological-Spaces.tendsto-eq-intros(25)[of - 0 - - 2 * (b * s), OF Topological-Spaces.tendsto-eq-intros(25)[of - 0 - - d * d * b]]*)
apply(*intro lim-const-over-n t1 t2 t3 t4 tendsto-diff[of - 0 - - 0, simplified] tendsto-add-zero tendsto-add[of - d * d * b - - 0, simplified] | auto*) +
done
finally show ?*thesis*
by(*rule LIMSEQ-imp-Suc*)
qed

lemma *GaussLearn-KL-divergence-lem5*:

fixes *a b c d K :: real*
assumes [*arith*]: $b > 0\ d > 0\ s > 0\ K > 0\ |f\ l| < K * \text{length}\ l$
shows $|(c * s * b * \text{real}(\text{length}\ l) + f\ l * d * s + c * s * s - a * s * d * \text{real}(\text{length}\ l) - f\ l * b * s - a * s * s)^2 / (d * d * b * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) + s * s * b * \text{real}(\text{length}\ l) + 2 * d * s * b * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) + d * d * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) * s + s * s * s + 2 * d * s * s * \text{real}(\text{length}\ l)) / (2 * (b * s))| \leq |(|c * s * b * \text{real}(\text{length}\ l)| + |K * \text{real}(\text{length}\ l) * d * s| + |c * s * s| + |a * s * d * \text{real}(\text{length}\ l)| + |- K * \text{real}(\text{length}\ l) * b * s| + |a * s * s|)^2 / (d * d * b * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) + s * s * b * \text{real}(\text{length}\ l) + 2 * d * s * b * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) * \text{real}(\text{length}\ l) + d * d * (\text{real}(\text{length}\ l) * \text{real}(\text{length}\ l)) * s + s * s * s + 2 * d * s * s * \text{real}(\text{length}\ l)) / (2 * (b * s))|$ (**is** $|(?l)^2 / ?c1 / ?c2| \leq |(?r)^2 / - / -|$)
proof -
have $?l^2 / ?c1 / ?c2 \leq ?r^2 / ?c1 / ?c2$

proof(*rule divide-right-mono*[*OF divide-right-mono*[*OF abs-le-square-iff*[*THEN iffD1*]]])
show $|?l| \leq |?r|$
proof –
have $|?l| \leq |c * s * b * \text{real } (\text{length } l)| + |f l * d * s| + |c * s * s| + |a * s * d * \text{real } (\text{length } l)| + |f l * b * s| + |a * s * s|$
by *linarith*
also have $\dots \leq |?r|$
by (*auto simp: mult.assoc abs-mult*) (*auto intro!: add-mono*)
finally show *?thesis* .
qed
qed *auto*
thus *?thesis*
by *fastforce*
qed

lemma *GaussLearn-KL-divergence-lem6*:

fixes $a e b c d K :: \text{real}$ **and** $f :: 'a \text{ list} \Rightarrow \text{real}$
assumes [*arith*]: $e > 0 \ b > 0 \ d > 0 \ s > 0$
shows $\exists N. \forall l. \text{length } l \geq N \longrightarrow |f l| < K * \text{length } l \longrightarrow |((f l * d + c * s) / (\text{length } l * d + s) - (f l * b + a * s) / (\text{length } l * b + s))|^2 / (2 * ((b * s) / (\text{length } l * b + s)))| < e$
proof(*cases* $K > 0$)
case K [*arith*]:*True*
from *GaussLearn-KL-divergence-lem4*[*OF assms(2-)*],*of c K a - K*] *assms(1)*
obtain N **where** N :
 $\bigwedge n. n \geq N \implies |(|c * s * b * \text{real } n| + |K * \text{real } n * d * s| + |c * s * s| + |a * s * d * \text{real } n| + |- K * \text{real } n * b * s| + |a * s * s|)^2 / (d * d * b * (\text{real } n * \text{real } n * \text{real } n) + s * s * b * \text{real } n + 2 * d * s * b * (\text{real } n * \text{real } n) + d * d * (\text{real } n * \text{real } n) * s + s * s * s + 2 * d * s * s * \text{real } n) / (2 * (b * s))| < e$
by(*fastforce simp: LIMSEQ-def*)
show *?thesis*
proof(*safe intro!: exI*[**where** $x=N$])
fix $l :: 'a \text{ list}$
assume $l:N \leq \text{length } l \ |f l| < K * \text{real } (\text{length } l)$
show $|((f l * d + c * s) / (\text{real } (\text{length } l) * d + s) - (f l * b + a * s) / (\text{real } (\text{length } l) * b + s))^2 / (2 * (b * s / (\text{real } (\text{length } l) * b + s)))| < e$ (**is** $?l < -$)
proof –
have $?l = |c * s * b * \text{real } (\text{length } l) + f l * d * s + c * s * s - a * s * d * \text{real } (\text{length } l) - f l * b * s - a * s * s|^2 / (d * d * b * (\text{real } (\text{length } l) * \text{real } (\text{length } l) * \text{real } (\text{length } l)) + s * s * b * \text{real } (\text{length } l) + 2 * d * s * b * (\text{real } (\text{length } l) * \text{real } (\text{length } l)) + d * d * (\text{real } (\text{length } l) * \text{real } (\text{length } l)) * s + s * s * s + 2 * d * s * s * \text{real } (\text{length } l)) / (2 * (b * s))|$
unfolding *GaussLearn-KL-divergence-lem3*[*OF assms(2-)*] **by** *simp*
also have $\dots \leq |(|c * s * b * \text{real } (\text{length } l)| + |K * \text{real } (\text{length } l) * d * s| + |c * s * s| + |a * s * d * \text{real } (\text{length } l)| + |- K * \text{real } (\text{length } l) * b * s| + |a * s * s|)^2 / (d * d * b * (\text{real } (\text{length } l) * \text{real } (\text{length } l) * \text{real } (\text{length } l)) + s * s * b * \text{real } (\text{length } l) + 2 * d * s * b * (\text{real } (\text{length } l) * \text{real } (\text{length } l)) + d * d * (\text{real } (\text{length } l) * \text{real } (\text{length } l)) * s + s * s * s + 2 * d * s * s * \text{real } (\text{length } l))$

```

/ (2 * (b * s))|
  by(rule GaussLearn-KL-divergence-lem5) (use l in auto)
  also have ... < e
  by(rule N) fact
  finally show ?thesis .
qed
qed
next
case False
then show ?thesis
  by (metis (no-types, opaque-lifting) abs-ge-zero add-le-cancel-left add-nonneg-nonneg
diff-add-cancel diff-ge-0-iff-ge linorder-not-less of-nat-0-le-iff zero-less-mult-iff)
qed

```

lemma *GaussLearn-KL-divergence*:

```

fixes a b c d e K :: real
assumes [arith]: e > 0 b > 0 d > 0
shows  $\exists N. \forall L. \text{length } L > N \longrightarrow |Total\ L / \text{length } L| < K$ 
 $\longrightarrow KL\text{-divergence (exp 1) (GaussLearn (Gauss a b) L) (GaussLearn$ 
(Gauss c d) L) < e
```

proof –

```

have h:  $\sigma^2 > 0\ b^2 > 0\ d^2 > 0$ 
```

```

by auto
```

```

from GaussLearn-KL-divergence-lem6[of e / 3, OF - h(2,3,1)] obtain N1 where
N1:
```

```

 $\bigwedge l. N1 \leq \text{length } l \implies |Total\ l| < K * \text{real (length } l) \implies |((Total\ l * d^2 + c * \sigma^2) / (\text{real (length } l) * d^2 + \sigma^2) - (Total\ l * b^2 + a * \sigma^2) / (\text{real (length } l) * b^2 + \sigma^2))^2 / (2 * (b^2 * \sigma^2 / (\text{real (length } l) * b^2 + \sigma^2)))| < e / 3$ 
```

```

by fastforce
```

```

from GaussLearn-KL-divergence-lem1'[OF assms(2,3) <σ > 0>]
```

```

have  $\bigwedge e. e > 0 \implies \exists N. \forall n. n \geq N \longrightarrow |\ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } n * b^2 + \sigma^2)) / \text{sqrt } (d^2 * \sigma^2 / (\text{real } n * d^2 + \sigma^2)))| < e$ 
```

```

by(auto simp: LIMSEQ-def)
```

```

from this[of e / 3] obtain N2 where N2:
```

```

 $\bigwedge n. n \geq N2 \implies |\ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } n * b^2 + \sigma^2)) / \text{sqrt } (d^2 * \sigma^2 / (\text{real } n * d^2 + \sigma^2)))| < e / 3$ 
```

```

by auto
```

```

from GaussLearn-KL-divergence-lem2'[OF <σ > 0> assms(2,3)]
```

```

have  $\bigwedge e. e > 0 \implies \exists N. \forall n. n \geq N \longrightarrow |d^2 * \sigma^2 / (\text{real } n * d^2 + \sigma^2) / (2 * (b^2 * \sigma^2 / (\text{real } n * b^2 + \sigma^2))) - 1 / 2| < e$ 
```

```

by(auto simp: LIMSEQ-def)
```

```

from this[of e / 3] obtain N3 where N3:
```

```

 $\bigwedge n. n \geq N3 \implies |d^2 * \sigma^2 / (\text{real } n * d^2 + \sigma^2) / (2 * (b^2 * \sigma^2 / (\text{real } n * b^2 + \sigma^2))) - 1 / 2| < e / 3$ 
```

```

by auto
```

```

define N where  $N = \max (\max N1\ N2) (\max N3\ 1)$ 
```

```

have N:  $N \geq N1\ N \geq N2\ N \geq N3\ N \geq 1$ 
```

```

by(auto simp: N-def)
```

```

show ?thesis
```

```

proof(safe intro!: exI[where  $x=N$ ])
  fix  $L :: \text{real list}$ 
  assume  $l:N < \text{length } L \mid \text{local.Total } L / \text{real } (\text{length } L) < K$ 
  then have  $l': N \leq \text{length } L \mid \text{Total } L < K * \text{real } (\text{length } L)$ 
  using order.strict-trans1[OF N(4) l(1)] by(auto intro!: pos-divide-less-eq[THEN
iffD1])
  show  $KL\text{-divergence } (\text{exp } 1) (\text{GaussLearn } (\text{Gauss } a \ b) \ L) (\text{GaussLearn } (\text{Gauss }
c \ d) \ L) < e$  (is ?lhs < -)
  proof -
    have  $h': \text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) > 0 \ \text{sqrt } (d^2 * \sigma^2 / (\text{real }
(\text{length } L) * d^2 + \sigma^2)) > 0$ 
    by(auto intro!: divide-pos-pos add-nonneg-pos)
    have ?lhs  $\leq \mid ?lhs \mid$ 
    by auto
    also have ... =  $\mid KL\text{-divergence } (\text{exp } 1) (\text{Gauss } ((\text{Total } L * b^2 + a * \sigma^2) /
(\text{real } (\text{length } L) * b^2 + \sigma^2)) (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)))) (\text{Gauss }
((\text{Total } L * d^2 + c * \sigma^2) / (\text{real } (\text{length } L) * d^2 + \sigma^2)) (\text{sqrt } (d^2 * \sigma^2 / (\text{real }
(\text{length } L) * d^2 + \sigma^2)))) \mid$ 
    by(simp add: GaussLearn-Total[OF assms(2) refl] GaussLearn-Total[OF
assms(3) refl])
    also have ... =  $\mid \ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) / \text{sqrt } (d^2
* \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2))) + ((\text{sqrt } (d^2 * \sigma^2 / (\text{real } (\text{length } L) * d^2 +
\sigma^2)))^2 + ((\text{Total } L * d^2 + c * \sigma^2) / (\text{real } (\text{length } L) * d^2 + \sigma^2) - (\text{Total } L * b^2
+ a * \sigma^2) / (\text{real } (\text{length } L) * b^2 + \sigma^2))^2) / (2 * (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L)
* b^2 + \sigma^2)))^2) - 1 / 2 \mid$ 
    by(simp add: KL-normal-density[OF h'] Gauss-def)
    also have ... =  $\mid \ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) / \text{sqrt } (d^2
* \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2))) + (\text{sqrt } (d^2 * \sigma^2 / (\text{real } (\text{length } L) * d^2 +
\sigma^2)))^2 / (2 * (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)))^2) + ((\text{Total } L * d^2 +
c * \sigma^2) / (\text{real } (\text{length } L) * d^2 + \sigma^2) - (\text{Total } L * b^2 + a * \sigma^2) / (\text{real } (\text{length } L)
* b^2 + \sigma^2))^2 / (2 * (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)))^2) - 1 / 2 \mid$ 
    unfolding add-divide-distrib by auto
    also have ... =  $\mid \ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) / \text{sqrt } (d^2
* \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2))) + (d^2 * \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2)) /
(2 * (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2))) + ((\text{Total } L * d^2 + c * \sigma^2) / (\text{real }
(\text{length } L) * d^2 + \sigma^2) - (\text{Total } L * b^2 + a * \sigma^2) / (\text{real } (\text{length } L) * b^2 + \sigma^2))^2 /
(2 * (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2))) - 1 / 2 \mid$ 
    using h' by auto
    also have ...  $\leq \mid \ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) / \text{sqrt } (d^2 *
\sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2))) + ((d^2 * \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2)) / (2
* (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)))) - 1 / 2 + ((\text{Total } L * d^2 + c * \sigma^2) /
(\text{real } (\text{length } L) * d^2 + \sigma^2) - (\text{Total } L * b^2 + a * \sigma^2) / (\text{real } (\text{length } L) * b^2 +
\sigma^2))^2 / (2 * (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2))) \mid$ 
    by auto
    also have ...  $\leq \mid \ln (\text{sqrt } (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)) / \text{sqrt } (d^2 *
\sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2))) + \mid ((d^2 * \sigma^2 / (\text{real } (\text{length } L) * d^2 + \sigma^2)) / (2
* (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2)))) - 1 / 2 + \mid ((\text{Total } L * d^2 + c * \sigma^2) /
(\text{real } (\text{length } L) * d^2 + \sigma^2) - (\text{Total } L * b^2 + a * \sigma^2) / (\text{real } (\text{length } L) * b^2 +
\sigma^2))^2 / (2 * (b^2 * \sigma^2 / (\text{real } (\text{length } L) * b^2 + \sigma^2))) \mid$ 

```

```

    by linarith
  also have ... < e
    using N1[OF order.trans[OF N(1) l'(1) l'(2)]] N2[OF order.trans[OF N(2)
l'(1)]] N3[OF order.trans[OF N(3) l'(1)]] by auto
    finally show ?thesis .
  qed
  qed
  qed
end

```

5.2.4 Continuous Distributions

The following (high-order) program receives a non-negative function f and returns the distribution whose density function is (normalized) f if f is integrable w.r.t. the Lebesgue measure.

definition *dens-to-dist* :: [$'a :: euclidean-space \Rightarrow real$] $\Rightarrow 'a$ qbs-measure **where**
dens-to-dist $\equiv (\lambda f. \text{do } \{$
 query *lborel_Q* *f*
 $\})$

lemma *dens-to-dist-qbs*[*qbs*]: *dens-to-dist* $\in (borel_Q \Rightarrow_Q \mathbb{R}_Q) \rightarrow_Q monadM$ -*qbs*
borel_Q
 by(*simp add: dens-to-dist-def*)

context

```

  fixes f :: 'a :: euclidean-space  $\Rightarrow$  real
  assumes f-qbs[qbs]: f  $\in$  qbs-borel  $\rightarrow_Q \mathbb{R}_Q$ 
    and f-le0: $\bigwedge x. f\ x \geq 0$ 
    and f-int-ne0:qbs-l (density-qbs lborel-qbs f) UNIV  $\neq 0$ 
    and f-integrable: qbs-integrable lborel-qbs f

```

begin

lemma *f-integrable'*[*measurable*]: *integrable* *lborel* *f*
 using *f-integrable* by(*simp add: qbs-integrable-iff-integrable*)

lemma *f-int-neinfy*:

```

  qbs-l (density-qbs lborel-qbs f) UNIV  $\neq \infty$ 
  using f-integrable' f-le0
  by(auto simp: qbs-l-density-qbs[of - qbs-borel] emeasure-density integrable-iff-bounded)

```

lemma *dens-to-dist*: *dens-to-dist* $f = \text{density-qbs } lborel\text{-qbs } (\lambda x. \text{ennreal } (1 / \text{measure } (qbs\text{-l } (density\text{-qbs } lborel\text{-qbs } f)) UNIV * f\ x))$

proof –

```

  have [simp]:ennreal (f x) * (1 / emeasure (qbs-l (density-qbs lborelQ (λx. ennreal
(f x)))) UNIV) = ennreal (f x / measure (qbs-l (density-qbs lborelQ (λx. ennreal
(f x)))) UNIV) for x
  by (metis divide-ennreal emeasure-eq-ennreal-measure ennreal-0 ennreal-times-divide
f-int-ne0 f-int-neinfy f-le0 infinity-ennreal-def mult.comm-neutral zero-less-measure-iff)

```

show *?thesis*
by(*auto simp: dens-to-dist-def query-def normalize-qbs[of - qbs-borel,simplified qbs-space-qbs-borel,OF - f-int-ne0 f-int-neinfty] density-qbs-density-qbs-eq[of - qbs-borel]*)
qed

corollary *qbs-l-dens-to-dist: qbs-l (dens-to-dist f) = density lborel (λx. ennreal (1 / measure (qbs-l (density-qbs lborel-qbs f)) UNIV * f x))*
by(*simp add: dens-to-dist qbs-l-density-qbs[of - qbs-borel]*)

corollary *qbs-integral-dens-to-dist:*
assumes [*qbs*]: $g \in \text{qbs-borel} \rightarrow_{\mathbb{Q}} \mathbb{R}_{\mathbb{Q}}$
shows $(\int_{\mathbb{Q}} x. g x \partial \text{dens-to-dist } f) = (\int_{\mathbb{Q}} x. 1 / \text{measure } (\text{qbs-l } (\text{density-qbs lborel-qbs } f)) \text{ UNIV } * f x * g x \partial \text{lborel}_{\mathbb{Q}})$
using *f-le0 by(simp add: qbs-integral-density-qbs[of - qbs-borel - g ,OF - - - AEq-I2[of - qbs-borel]] dens-to-dist)*

lemma *dens-to-dist-prob[qbs]:dens-to-dist f ∈ qbs-space (monadP-qbs borel_Q)*
using *f-int-neinfty f-int-ne0 by(auto simp: dens-to-dist-def query-def intro!: normalize-qbs-prob)*

end

5.2.5 Normal Distribution

context
fixes $\mu \sigma :: \text{real}$
assumes *sigma-pos[arith]: σ > 0*
begin

We use an unnormalized density function.

definition *normal-f* $\equiv (\lambda x. \text{exp } (-(x - \mu)^2 / (2 * \sigma^2)))$

lemma *nc-normal-f: qbs-l (density-qbs lborel-qbs normal-f) UNIV = ennreal (sqrt (2 * pi * σ²))*

proof –
have *qbs-l (density-qbs lborel-qbs normal-f) UNIV = (∫⁺ x. ennreal (exp (-(x - μ)² / (2 * σ²)))) ∂lborel*
by(*auto simp: qbs-l-density-qbs[of - qbs-borel] normal-f-def emeasure-density*)
also have $\dots = \text{ennreal } (\text{sqrt } (2 * \text{pi} * \sigma^2)) * (\int^+ x. \text{normal-density } \mu \sigma x \partial \text{lborel})$
by(*auto simp: nn-integral-cmult[symmetric] normal-density-def ennreal-mult'[symmetric] intro!: nn-integral-cong*)
also have $\dots = \text{ennreal } (\text{sqrt } (2 * \text{pi} * \sigma^2))$
using *prob-space.emeasure-space-1[OF prob-space-normal-density]*
by(*simp add: emeasure-density*)
finally show *?thesis .*
qed

corollary *measure-qbs-l-dens-to-dist-normal-f: measure (qbs-l (density-qbs lborel-qbs*

normal-f) $UNIV = \text{sqrt } (2 * \pi * \sigma^2)$
by(*simp add: measure-def nc-normal-f*)

lemma *normal-f*:
shows *normal-f* \in *qbs-borel* $\rightarrow_Q \mathbb{R}_Q$
and $\bigwedge x. \text{normal-f } x \geq 0$
and *qbs-l* (*density-qbs lborel-qbs normal-f*) $UNIV \neq 0$
and *qbs-integrable lborel-qbs normal-f*
using *nc-normal-f* **by**(*auto simp: qbs-integrable-iff-integrable integrable-iff-bounded qbs-l-density-qbs[of - qbs-borel] normal-f-def emeasure-density*)

lemma *qbs-l-dens-to-dist-normal-f*: *qbs-l* (*dens-to-dist normal-f*) = *density lborel* (*normal-density* μ σ)
by(*simp add: qbs-l-dens-to-dist[OF normal-f] measure-qbs-l-dens-to-dist-normal-f normal-density-def*) (*simp add: normal-f-def*)

end

5.2.6 Half Normal Distribution

context
fixes $\mu \sigma :: \text{real}$
assumes *sigma-pos[arith]*: $\sigma > 0$
begin

definition *hnormal-f* $\equiv (\lambda x. \text{if } x \leq \mu \text{ then } 0 \text{ else normal-density } \mu \sigma x)$

lemma *nc-hnormal-f*: *qbs-l* (*density-qbs lborel-qbs hnormal-f*) $UNIV = \text{ennreal } (1 / 2)$

proof –

have *qbs-l* (*density-qbs lborel-qbs hnormal-f*) $UNIV = (\int^+ x. \text{ennreal } (\text{if } x \leq \mu \text{ then } 0 \text{ else normal-density } \mu \sigma x) \partial \text{lborel})$

by(*auto simp: qbs-l-density-qbs[of - qbs-borel] hnormal-f-def emeasure-density*)

also have $\dots = (\int^+ x \in \{\mu < ..\}. \text{normal-density } \mu \sigma x \partial \text{lborel})$

by(*auto intro!: nn-integral-cong*)

also have $\dots = 1 / 2 * (\int^+ x. \text{normal-density } \mu \sigma x \partial \text{lborel})$

proof –

have $1 : (\int^+ x. \text{normal-density } \mu \sigma x \partial \text{lborel}) = (\int^+ x \in \{\mu < ..\}. \text{normal-density } \mu \sigma x \partial \text{lborel}) + (\int^+ x \in \{.. \mu\}. \text{normal-density } \mu \sigma x \partial \text{lborel})$

by(*auto simp: nn-integral-add[symmetric] intro!: nn-integral-cong*) (*simp add: indicator-def*)

have $2 : (\int^+ x \in \{\mu < ..\}. \text{normal-density } \mu \sigma x \partial \text{lborel}) = (\int^+ x \in \{.. \mu\}. \text{normal-density } \mu \sigma x \partial \text{lborel})$ (**is** $?l = ?r$)

proof –

have $?l = (\int^+ x. \text{ennreal } (\text{normal-density } \mu \sigma x * \text{indicator } \{\mu < ..\} x) \partial \text{lborel})$

by(*auto intro!: nn-integral-cong simp add: indicator-mult-ennreal mult.commute*)

also have $\dots = \text{ennreal } (\int x. \text{normal-density } \mu \sigma x * \text{indicator } \{\mu < ..\} x \partial \text{lborel})$

by(*auto intro!*: *nn-integral-eq-integral integrable-real-mult-indicator*)
also have ... = *ennreal* ($\int x.$ *normal-density* $\mu \sigma x$ * *indicator* $\{\mu<..\}$ x
∂lebesgue)
by(*simp add: integral-completion*)
also have ... = *ennreal* ($\int x.$ (*if* $x \in \{\mu<..\}$ *then normal-density* $\mu \sigma x$ *else*
0) *∂lebesgue*)
by (*meson indicator-times-eq-if(2)*)
also have ... = *ennreal* ($\int x.$ *normal-density* $\mu \sigma x$ *∂lebesgue-on* $\{\mu<..\}$)
by(*rule ennreal-cong, rule Lebesgue-Measure.integral-restrict-UNIV*) *simp*
also have ... = *ennreal* (*integral* $\{\mu<..\}$ (*normal-density* $\mu \sigma$))
by(*rule ennreal-cong, rule lebesgue-integral-eq-integral*) (*auto simp: in-*
tegrable-restrict-space integrable-completion intro!: integrable-mult-indicator[*where*
'b=real,simplified])
also have ... = *ennreal* (*integral* $\{..<\mu\}$ ($\lambda x.$ *normal-density* $\mu \sigma (- x + 2 * \mu)$))
proof –
have *integral* $\{\mu<..\}$ (*normal-density* $\mu \sigma$) = *integral* $\{..<\mu\}$ ($\lambda x.$ $| - 1 | *_{\mathbb{R}}$
normal-density $\mu \sigma (- x + 2 * \mu)$)
proof(*rule conjunct2*[*OF has-absolute-integral-change-of-variables-1'*][*where*
 $g=\lambda x. - x + 2 * \mu$ **and** $S=\{..<\mu\}$ **and** $g'=\lambda x. - 1$ **and** $f=\text{normal-density } \mu \sigma$
and $b=\text{integral } \{\mu<..\}$ (*normal-density* $\mu \sigma$),*THEN iffD2*],*symmetric*])
fix $x :: \text{real}$
show ($\lambda x. - x + 2 * \mu$) *has-real-derivative* $- 1$ (*at* x *within* $\{..<\mu\}$)
by(*rule derivative-eq-intros(35)*[*of - - 1 - - 0*]) (*auto simp add: De-*
riv.field-differentiable-minus)
next
show *inj-on* ($\lambda x. - x + 2 * \mu$) $\{..<\mu\}$
by(*auto simp: inj-on-def*)
next
have $1: (\lambda x. - x + 2 * \mu) ' \{..<\mu\} = \{\mu<..\}$
by(*auto simp: image-def intro!: bexI*[*where* $x=2 * \mu - \cdot$])
have [*simp*]: *normal-density* $\mu \sigma$ *absolutely-integrable-on* $\{\mu<..\}$
by(*auto simp: absolutely-integrable-measurable comp-def integrable-restrict-space*
integrable-completion intro!: integrable-mult-indicator[*where* *'b=real,simplified*] *mea-*
surable-restrict-space1 measurable-completion)
show *normal-density* $\mu \sigma$ *absolutely-integrable-on* ($\lambda x. - x + 2 * \mu$) ' $\{..<\mu\} \wedge$
integral ($\lambda x. - x + 2 * \mu$) ' $\{..<\mu\}$ (*normal-density* $\mu \sigma$) = *integral*
 $\{\mu<..\}$ (*normal-density* $\mu \sigma$)
unfolding 1 **by** *simp*
qed *auto*
thus ?*thesis* **by** *simp*
qed
also have ... = *ennreal* (*integral* $\{..<\mu\}$ (*normal-density* $\mu \sigma$))
proof –
have ($\lambda x.$ *normal-density* $\mu \sigma (- x + 2 * \mu)$) = *normal-density* $\mu \sigma$
by *standard* (*auto simp: normal-density-def power2-commute*)
thus ?*thesis* **by** *simp*
qed
also have ... = *ennreal* ($\int x.$ *normal-density* $\mu \sigma x$ *∂lebesgue-on* $\{..<\mu\}$)

by(rule *ennreal-cong*, rule *lebesgue-integral-eq-integral[symmetric]*) (auto
simp: integrable-restrict-space integrable-completion intro!: integrable-mult-indicator[where
'b=real,simplified])
also have ... = *ennreal* ($\int x.$ (if $x \in \{..<\mu\}$ then *normal-density* μ σ x else
0) *lebesgue*)
by(rule *ennreal-cong*, rule *Lebesgue-Measure.integral-restrict-UNIV[symmetric]*)
simp
also have ... = *ennreal* ($\int x.$ *normal-density* μ σ x * *indicator* $\{..<\mu\}$ x
lebesgue)
by (*meson indicator-times-eq-if(2)[symmetric]*)
also have ... = *ennreal* ($\int x.$ *normal-density* μ σ x * *indicator* $\{..<\mu\}$ x
lborel)
by(*simp add: integral-completion*)
also have ... = ($\int^+ x.$ *ennreal* (*normal-density* μ σ x * *indicator* $\{..<\mu\}$ x)
lborel)
by(auto *intro!: nn-integral-eq-integral[symmetric] integrable-real-mult-indicator*)
also have ... = ?*r*
using *AE-lborel-singleton* **by**(*fastforce intro!: nn-integral-cong-AE simp:*
indicator-def)
finally show ?*thesis* .
qed
show ?*thesis*
by(*simp add: 1 2*) (*metis (no-types, lifting) ennreal-divide-times mult-2*
mult-2-right mult-divide-eq-ennreal one-add-one top-neq-numeral zero-neq-numeral)
qed
also have ... = *ennreal* ($1 / 2$)
using *prob-space.emmeasure-space-1[OF prob-space-normal-density]*
by(*simp add: emmeasure-density divide-ennreal-def*)
finally show ?*thesis* .
qed

corollary *measure-qbs-l-dens-to-dist-hnormal-f: measure* (*qbs-l* (*density-qbs lborel-qbs*
hnormal-f)) *UNIV* = $1 / 2$
by(*simp add: measure-def nc-hnormal-f del: ennreal-half*)

lemma *hnormal-f:*
shows *hnormal-f* \in *qbs-borel* \rightarrow_Q \mathbb{R}_Q
and $\bigwedge x.$ *hnormal-f* $x \geq 0$
and *qbs-l* (*density-qbs lborel-qbs hnormal-f*) *UNIV* $\neq 0$
and *qbs-integrable lborel-qbs hnormal-f*
using *nc-hnormal-f* **by**(auto *simp: qbs-integrable-iff-integrable integrable-iff-bounded*
qbs-l-density-qbs[of - qbs-borel] hnormal-f-def emmeasure-density simp del: ennreal-half)

lemma *qbs-l* (*dens-to-dist local.hnormal-f*) = *density lborel* ($\lambda x.$ *ennreal* ($2 * (if$ x
 $\leq \mu$ then 0 else *normal-density* μ σ x)))
by(*simp add: qbs-l-dens-to-dist[OF hnormal-f] measure-qbs-l-dens-to-dist-hnormal-f*)
(*simp add: hnormal-f-def*)

end

5.2.7 Erlang Distribution

context

fixes $k :: \text{nat}$ **and** $l :: \text{real}$
assumes $l\text{-pos}[\text{arith}] : l > 0$

begin

definition $\text{erlang-f} \equiv (\lambda x. \text{if } x < 0 \text{ then } 0 \text{ else } x^k * \exp(-l * x))$

lemma $\text{nc-erlang-f} : \text{qbs-l}(\text{density-qbs lborel-qbs erlang-f}) \text{ UNIV} = \text{ennreal}(\text{fact } k / l^{\wedge}(\text{Suc } k))$

proof –

have $\text{qbs-l}(\text{density-qbs lborel-qbs erlang-f}) \text{ UNIV} = (\int^+ x. \text{ennreal}(\text{if } x < 0 \text{ then } 0 \text{ else } x^k * \exp(-l * x)) \partial \text{lborel})$

by($\text{auto simp: qbs-l-density-qbs[of - qbs-borel] erlang-f-def emeasure-density}$)

also have $\dots = \text{ennreal}(\text{fact } k / l^{\wedge}(\text{Suc } k)) * (\int^+ x. \text{erlang-density } k \ l \ x \ \partial \text{lborel})$

by($\text{auto simp: nn-integral-cmult[symmetric] ennreal-mult'[symmetric] erlang-density-def intro!: nn-integral-cong}$)

also have $\dots = \text{ennreal}(\text{fact } k / l^{\wedge}(\text{Suc } k))$

using $\text{prob-space.emeasure-space-1[OF prob-space-erlang-density]}$

by($\text{simp add: emeasure-density}$)

finally show $?thesis$.

qed

corollary $\text{measure-qbs-l-dens-to-dist-erlang-f} : \text{measure}(\text{qbs-l}(\text{density-qbs lborel-qbs erlang-f})) \text{ UNIV} = \text{fact } k / l^{\wedge}(\text{Suc } k)$

by($\text{simp add: measure-def nc-erlang-f}$)

lemma erlang-f :

shows $\text{erlang-f} \in \text{qbs-borel} \rightarrow_Q \mathbb{R}_Q$

and $\bigwedge x. \text{erlang-f } x \geq 0$

and $\text{qbs-l}(\text{density-qbs lborel-qbs erlang-f}) \text{ UNIV} \neq 0$

and $\text{qbs-integrable lborel-qbs erlang-f}$

using nc-erlang-f **by**($\text{auto simp: qbs-integrable-iff-integrable integrable-iff-bounded qbs-l-density-qbs[of - qbs-borel] erlang-f-def emeasure-density}$)

lemma $\text{qbs-l}(\text{dens-to-dist erlang-f}) = \text{density lborel}(\text{erlang-density } k \ l)$

proof –

have $[\text{simp}] : l * l^{\wedge} k * (\text{if } x < 0 \text{ then } 0 \text{ else } x^k * \exp(-l * x)) / \text{fact } k = (\text{if } x < 0 \text{ then } 0 \text{ else } l^{\wedge}(\text{Suc } k) * x^k * \exp(-l * x)) / \text{fact } k$ **for** x

by auto

show $?thesis$

by($\text{simp add: qbs-l-dens-to-dist[OF erlang-f] measure-qbs-l-dens-to-dist-erlang-f erlang-density-def}$) ($\text{simp add: erlang-f-def}$)

qed

end

5.2.8 Uniform Distribution on $(0, 1) \times (0, 1)$.

definition *uniform-f* \equiv *indicat-real* ($\{0 < .. < 1 :: \text{real}\} \times \{0 < .. < 1 :: \text{real}\}$)

lemma

shows *uniform-f-qbs'*[*qbs*]: *uniform-f* \in *qbs-borel* \rightarrow_Q \mathbb{R}_Q
and *uniform-f-qbs*[*qbs*]: *uniform-f* \in $\mathbb{R}_Q \otimes_Q \mathbb{R}_Q \rightarrow_Q \mathbb{R}_Q$

proof –

have *uniform-f* \in $\mathbb{R}_Q \otimes_Q \mathbb{R}_Q \rightarrow_Q \mathbb{R}_Q$

by (*auto simp: uniform-f-def r-preserves-product[symmetric] intro!: rr.qbs-morphism-measurable-intro*)

thus *uniform-f* \in $\mathbb{R}_Q \otimes_Q \mathbb{R}_Q \rightarrow_Q \mathbb{R}_Q$ *uniform-f* \in *qbs-borel* \rightarrow_Q \mathbb{R}_Q

by (*simp-all add: qbs-borel-prod*)

qed

lemma *uniform-f-measurable*[*measurable*]: *uniform-f* \in *borel-measurable borel*

by (*metis borel-prod rr.standard-borel-axioms standard-borel.standard-borel-r-full-faithful uniform-f-qbs'*)

lemma *nc-uniform-f: qbs-l (density-qbs lborel-qbs uniform-f) UNIV = 1*

proof –

have *qbs-l (density-qbs lborel-qbs uniform-f) UNIV* = $(\int^+ z. \text{ennreal } (\text{uniform-f } z) \partial \text{lborel})$

by (*auto simp: qbs-l-density-qbs[of - qbs-borel] emeasure-density*)

also have ... = $(\int^+ z. \text{indicator } \{0 < .. < 1 :: \text{real}\} (\text{fst } z) * \text{indicator } \{0 < .. < 1 :: \text{real}\} (\text{snd } z) \partial (\text{lborel} \otimes_M \text{lborel}))$

by (*auto simp: lborel-prod intro!: nn-integral-cong (auto simp: indicator-def uniform-f-def)*)

also have ... = 1

by (*auto simp: lborel.nn-integral-fst[symmetric] nn-integral-cmult*)

finally show *?thesis* .

qed

corollary *measure-qbs-l-dens-to-dist-uniform-f: measure (qbs-l (density-qbs lborel-qbs uniform-f)) UNIV = 1*

by (*simp add: measure-def nc-uniform-f*)

lemma *uniform-f:*

shows *uniform-f* \in *qbs-borel* \rightarrow_Q \mathbb{R}_Q

and $\bigwedge x. \text{uniform-f } x \geq 0$

and *qbs-l (density-qbs lborel-qbs uniform-f) UNIV* $\neq 0$

and *qbs-integrable lborel-qbs uniform-f*

using *nc-uniform-f* **by** (*auto simp: qbs-integrable-iff-integrable integrable-iff-bounded qbs-l-density-qbs[of - qbs-borel] emeasure-density (auto simp: uniform-f-def)*)

lemma *qbs-l-dens-to-dist-uniform-f: qbs-l (dens-to-dist uniform-f) = density lborel*

$(\lambda x. \text{ennreal } (\text{uniform-f } x))$

by (*simp add: qbs-l-dens-to-dist[OF uniform-f, simplified measure-qbs-l-dens-to-dist-uniform-f]*)

lemma *dens-to-dist uniform-f = Uniform 0 1* $\otimes_{Q_{\text{mes}}}$ *Uniform 0 1*

proof –

```

note qbs-pair-measure-morphismP[qbs] Uniform-qbsP[qbs]
have [simp]:sets (borel :: (real × real) measure) = sets (borel ⊗M borel)
  by(metis borel-prod)
show ?thesis
proof(safe intro!: inj-onD[OF qbs-l-inj[of  $\mathbb{R}_Q \otimes_Q \mathbb{R}_Q$ ]] qbs-space-monadPM
measure-eqI)

  fix A :: (real × real) set
  assume A ∈ sets (qbs-l (dens-to-dist uniform-f))
  then have [measurable]: A ∈ sets (borel ⊗M borel)
    by(auto simp: qbs-l-dens-to-dist-uniform-f)
  show emeasure (qbs-l (dens-to-dist uniform-f)) A = emeasure (qbs-l (Uniform
0 1 ⊗Qmes Uniform 0 1)) A (is ?lhs = ?rhs)
  proof -
    have ?lhs = (∫+ x ∈ A. ennreal (uniform-f x) ∂(lborel ⊗M lborel))
      by(simp add: emeasure-density lborel-prod qbs-l-dens-to-dist-uniform-f)
    also have ... = (∫+ x. indicator A x * indicator {0 <..x) * indicator
{0 <..x) ∂(lborel ⊗M lborel))
      by(auto intro!: nn-integral-cong) (auto simp: indicator-def uniform-f-def)
    also have ... = (∫+ x ∈ {0 <..+ y ∈ {0 <..indicator A (x, y) ∂lborel)
∂lborel)
      by(auto simp add: lborel.nn-integral-fst[symmetric] intro!: nn-integral-cong)
(auto simp: indicator-def)
    also have ... = (∫+ x. (∫+ y. indicator A (x, y) ∂uniform-measure lborel
{0 <..uniform-measure lborel {0 <..by(auto simp: nn-integral-uniform-measure divide-ennreal-def)
    also have ... = ?rhs
      by(auto simp: UniformP-pair.M1.emeasure-pair-measure' qbs-l-Uniform-pair)
    finally show ?thesis .
  qed
next
  show dens-to-dist uniform-f ∈ qbs-space (monadP-qbs ( $\mathbb{R}_Q \otimes_Q \mathbb{R}_Q$ ))
    by(simp add: dens-to-dist-prob[OF uniform-f] qbs-borel-prod)
  qed (auto simp: qbs-l-dens-to-dist-uniform-f qbs-l-Uniform-pair, qbs, simp)
qed

```

5.2.9 If then else

```

definition gt :: (real ⇒ real) ⇒ real ⇒ bool qbs-measure where
gt ≡ (λf r. do {
  x ← dens-to-dist (normal-f 0 1);
  if f x > r
  then return-qbs  $\mathbb{B}_Q$  True
  else return-qbs  $\mathbb{B}_Q$  False
})

```

```

declare normal-f(1)[of 1 0, simplified]

```

```

lemma gt-qbs[qbs]: gt ∈ qbs-space (( $\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$ ) ⇒Q  $\mathbb{R}_Q \Rightarrow_Q$  monadP-qbs  $\mathbb{B}_Q$ )

```

proof –
note $[qbs] = \text{dens-to-dist-prob}[OF \text{ normal-f}[of 1 0, \text{simplified}]] \text{ bind-qbs-morphismP}$
 $\text{return-qbs-morphismP}$
show $?thesis$
by(simp add: gt-def)
qed

lemma

assumes $[qbs]: f \in \mathbb{R}_Q \rightarrow_Q \mathbb{R}_Q$
shows $\mathcal{P}(b \text{ in } gt \ f \ r. \ b = \text{True}) = \mathcal{P}(x \text{ in } \text{std-normal-distribution}. \ f \ x > r)$ (**is**
 $?P1 = ?P2$)

proof –

note $[qbs] = \text{dens-to-dist-prob}[OF \text{ normal-f}[of 1 0, \text{simplified}]] \text{ bind-qbs-morphismP}$
 $\text{return-qbs-morphismP}$

have $1[\text{simp}]: \text{space}(qbs\text{-l}(gt \ f \ r)) = UNIV$

by($\text{simp add: space-qbs-l-in}[OF \text{ qbs-space-monadPM}, of - \mathbb{B}_Q]$)

have $?P1 = (\int b. \text{indicat-real} \{ \text{True} \} \ b \ \partial qbs\text{-l}(gt \ f \ r))$

by $\text{simp}(\text{metis}(\text{full-types}) \text{Collect-cong} \ \text{singleton-conv2})$

also have $\dots = (\int_Q b. \text{indicat-real} \{ \text{True} \} \ b \ \partial(gt \ f \ r))$

by($\text{simp add: qbs-integral-def2-l}$)

also have $\dots = (\int_Q b. \text{indicat-real} \{ \text{True} \} \ b \ \partial(\text{dens-to-dist}(\text{normal-f} \ 0 \ 1) \gg=$
 $(\lambda x. \text{return-qbs} \ \mathbb{B}_Q \ (f \ x > r))))$

proof –

have $[\text{simp}]: gt \ f \ r = \text{dens-to-dist}(\text{normal-f} \ 0 \ 1) \gg= (\lambda x. \text{return-qbs} \ \mathbb{B}_Q \ (f \ x > r))$

by($\text{auto simp: gt-def intro!: bind-qbs-cong}[of - \mathbb{R}_Q - - \mathbb{B}_Q] \text{ qbs-space-monadPM}$
 $\text{qbs-morphism-monadPD}$)

show $?thesis$ **by** simp

qed

also have $\dots = (\int_Q x. (\text{indicat-real} \{ \text{True} \} \circ (\lambda x. f \ x > r)) \ x \ \partial \text{dens-to-dist}(\text{normal-f} \ 0 \ 1))$

by($\text{rule qbs-integral-bind-return}[of - \mathbb{R}_Q]$) ($\text{auto intro!: qbs-space-monadPM}$)

also have $\dots = (\int_Q x. \text{indicat-real} \{ x. f \ x > r \} \ x \ \partial \text{dens-to-dist}(\text{normal-f} \ 0 \ 1))$

by($\text{auto intro!: qbs-integral-cong}[of - \mathbb{R}_Q] \text{ qbs-space-monadPM simp: indica-}$
 tor-def)

also have $\dots = (\int x. \text{indicat-real} \{ x. f \ x > r \} \ x \ \partial \text{dens-to-dist}(\text{normal-f} \ 0 \ 1))$

by($\text{simp add: qbs-integral-def2-l}$)

also have $\dots = ?P2$

by($\text{simp add: qbs-l-densto-dist-normal-f}[of 1 0]$)

finally show $?thesis$.

qed

Examples from Staton [5, Sect. 2.2].

5.2.10 Weekend

Example from Staton [5, Sect. 2.2.1].

This example is formalized in Coq by Affeldt et al. [1].

definition *weekend* :: bool qbs-measure **where**

```
weekend ≡ do {
  let x = qbs-bernoulli (2 / 7);
      f = (λx. let r = if x then 3 else 10 in pmf (poisson-pmf r) 4)
  in query x f
}
```

lemma *weekend-qbs*[qbs]:weekend ∈ qbs-space (monadM-qbs \mathbb{B}_Q)
by(simp add: weekend-def)

lemma *weekend-nc*:

```
defines N ≡ 2 / 7 * pmf (poisson-pmf 3) 4 + 5 / 7 * pmf (poisson-pmf 10)
4
shows qbs-l (density-qbs (bernoulli-pmf (2/7)) (λx. (pmf (poisson-pmf (if x then
3 else 10)) 4))) UNIV = N
proof -
  have [simp]:fact 4 = 4 * fact 3
    by (simp add: fact-numeral)
  show ?thesis
    by (simp add: qbs-l-density-qbs[of -  $\mathbb{B}_Q$ ] emeasure-density ennreal-plus[symmetric]
ennreal-mult'[symmetric] N-def del: ennreal-plus)
qed
```

lemma *qbs-l-weekend*:

```
defines N ≡ 2 / 7 * pmf (poisson-pmf 3) 4 + 5 / 7 * pmf (poisson-pmf 10)
4
shows qbs-l weekend = qbs-l (density-qbs (qbs-bernoulli (2 / 7)) (λx. ennreal
(let r = if x then 3 else 10 in r ^ 4 * exp (- r) / (fact 4 * N)))) (is ?lhs = ?rhs)
proof -
  have [simp]: N > 0
    by(auto simp: N-def intro!: add-pos-pos)
  have ?lhs = qbs-l (density-qbs (density-qbs (qbs-bernoulli (2 / 7)) (λx. ennreal
(let r = if x then 3 else 10 in r ^ 4 * exp (- r) / fact 4))) (λx. 1 / ennreal N))
    using normalize-qbs[of density-qbs (qbs-bernoulli (2/7)) (λx. (pmf (poisson-pmf
(if x then 3 else 10)) 4))]  $\mathbb{B}_Q$ ,simplified] weekend-nc
    by (simp add: weekend-def query-def N-def Let-def)
  also have ... = ?rhs
    by (simp add: density-qbs-density-qbs-eq[of -  $\mathbb{B}_Q$ ] ennreal-mult'[symmetric] en-
nreal-1[symmetric] divide-ennreal del: ennreal-1) (metis (mono-tags, opaque-lifting)
divide-divide-eq-left)
  finally show ?thesis .
qed
```

lemma

```
defines N ≡ 2 / 7 * pmf (poisson-pmf 3) 4 + 5 / 7 * pmf (poisson-pmf 10)
4
shows  $\mathcal{P}(b \text{ in weekend. } b = \text{True}) = 2 / 7 * (3^4 * \exp (- 3)) / \text{fact } 4 * 1 / N$ 
by simp (simp add: qbs-l-weekend measure-def qbs-l-density-qbs[of -  $\mathbb{B}_Q$ ] emea-
sure-density emeasure-measure-pmf-finite ennreal-mult'[symmetric] N-def)
```


5.2.11 Whattime

Example from Staton [5, Sect. 2.2.3]

f is given as a parameter.

definition *whattime* :: (real \Rightarrow real) \Rightarrow real qbs-measure **where**

```
whattime  $\equiv$  ( $\lambda f$ . do {
    let  $T = \text{Uniform } 0 \ 24$  in
    query  $T$  ( $\lambda t$ . let  $r = f \ t$  in
        exponential-density  $r$  (1 / 60))
})
```

lemma *whattime-qbs*[qbs]: *whattime* \in ($\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$) \Rightarrow_Q monadM-qbs \mathbb{R}_Q
 by(*simp add: whattime-def*)

lemma *qbs-l-whattime-sub*:

assumes [qbs]: $f \in \mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$

shows *qbs-l* (density-qbs (Uniform 0 24) (λx . exponential-density ($f \ x$) (1 / 60)))
 $=$ density lborel (λx . indicator {0<..<<24} x / 24 * exponential-density ($f \ x$) (1 / 60))

proof –

have [*measurable*]: $f \in$ borel-measurable borel

by (*simp add: standard-borel.standard-borel-r-full-faithful standard-borel-ne.standard-borel*)

have [*measurable*]: (λx . (exponential-density ($f \ x$) (1 / 60))) \in borel-measurable borel

by(*simp add: exponential-density-def*)

have 1 [*measurable*]: (λx . ennreal (exponential-density ($f \ x$) (1 / 60))) \in borel-measurable (uniform-measure lborel {0<..<<24})

by(*simp add: measurable-cong-sets[OF sets-uniform-measure]*)

show ?thesis

by(*auto simp: qbs-l-density-qbs[of - qbs-borel] emeasure-density emeasure-density[OF 1] nn-integral-uniform-measure nn-integral-divide[symmetric] ennreal-mult' divide-ennreal[symmetric] intro!: measure-eqI nn-integral-cong simp del: times-divide-eq-left*)

(*simp add: ennreal-indicator ennreal-times-divide mult.commute mult.left-commute*)

qed

lemma

assumes [qbs]: $f \in \mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$ **and** [*measurable*]: $U \in$ sets borel

and $\bigwedge r. f \ r \geq 0$

defines $N \equiv$ ($\int t \in \{0 < .. < 24\}. (f \ t * \exp (- 1 / 60 * f \ t)) \ \partial \text{lborel}$)

defines $N' \equiv$ ($\int ^+ t \in \{0 < .. < 24\}. (f \ t * \exp (- 1 / 60 * f \ t)) \ \partial \text{lborel}$)

assumes $N' \neq 0$ **and** $N' \neq \infty$

shows $\mathcal{P}(t \text{ in } \text{whattime } f. t \in U) =$ ($\int t \in \{0 < .. < 24\} \cap U. (f \ t * \exp (- 1 / 60 * f \ t)) \ \partial \text{lborel}$) / N

proof –

have 1: *space* (whattime f) = UNIV

by (*rule space-qbs-l-in[of whattime f \mathbb{R}_Q ,simplified qbs-space-qbs-borel]*) *simp*

have [*measurable*]: $f \in$ borel-measurable borel

by (*simp add: standard-borel.standard-borel-r-full-faithful standard-borel-ne.standard-borel*)

have [measurable]: $(\lambda x. \text{exponential-density } (f \ x) \ (1 / 60)) \in \text{borel-measurable borel}$
by(simp add: measurable-cong-sets[OF sets-uniform-measure] exponential-density-def)
have [measurable]: $(\lambda x. \text{ennreal } (\text{exponential-density } (f \ x) \ (1 / 60))) \in \text{borel-measurable (uniform-measure lborel } \{0 < .. < 24\})$
by(simp add: measurable-cong-sets[OF sets-uniform-measure])
have qbs-ld: qbs-l (density-qbs (Uniform 0 24) $(\lambda x. \text{exponential-density } (f \ x) \ (1 / 60))$) UNIV = $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x) / 24) \ \partial \text{lborel})$
by(auto simp: qbs-l-whattime-sub emeasure-density intro!: nn-integral-cong, auto simp: ennreal-indicator[symmetric] ennreal-mult''[symmetric] exponential-density-def) (simp add: mult.commute)
have int: integrable lborel $(\lambda x. f \ x * \exp (- 1 / 60 * f \ x) * \text{indicat-real } \{0 < .. < 24\} \ x)$
using assms(3,7) **by**(simp add: N'-def integrable-iff-bounded ennreal-mult'' ennreal-indicator top.not-eq-extremum)

have ge: $(\int x \in \{0 < .. < 24\}. (f \ x * \exp (- (f \ x / 60)) / 24) \ \partial \text{lborel}) > 0$
proof -
have $(\int x \in \{0 < .. < 24\}. (f \ x * \exp (- (f \ x / 60))) \ \partial \text{lborel}) > 0$ (is ?l > 0)
proof -
have ennreal ?l = $(\int^+ x. (\text{indicator } \{0 < .. < 24\} \ x * (f \ x * \exp (- (f \ x / 60)))) \ \partial \text{lborel})$
unfolding set-lebesgue-integral-def **by**(simp, rule nn-integral-eq-integral[symmetric]) (insert int assms(3), auto simp: mult.commute)
also have ... = $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x)) \ \partial \text{lborel})$
by (simp add: indicator-mult-ennreal mult.commute)
also have ... > 0
using assms(6) not-gr-zero N'-def **by** blast
finally show ?thesis
using ennreal-less-zero-iff **by** blast
qed
thus ?thesis **by** simp
qed
have ge2: $(\int x \in \{0 < .. < 24\} \cap U. (\text{exponential-density } (f \ x) \ (1 / 60)) \ \partial \text{lborel}) \geq 0$
using assms(3) **by**(auto intro!: integral-nonneg-AE simp: set-lebesgue-integral-def)

have $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x) / 24) \ \partial \text{lborel}) \neq 0$
 $\wedge (\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x) / 24) \ \partial \text{lborel}) \neq \infty$
proof -
have $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x) / 24) \ \partial \text{lborel}) =$
 $(\int^+ x. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x)) * \text{indicator } \{0 < .. < 24\} \ x / 24 \ \partial \text{lborel})$
by(rule nn-integral-cong, insert assms(3)) (auto simp: divide-ennreal[symmetric] ennreal-times-divide mult.commute)
also have ... = $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- 1 / 60 * f \ x)) \ \partial \text{lborel}) / 24$
by(simp add: nn-integral-divide)
finally show ?thesis

using *assms(5,6,7)* **by** (*simp add: ennreal-divide-eq-top-iff*)
qed
hence *normalize-qbs (density-qbs (Uniform 0 24) (λx. (exponential-density (f x) (1 / 60)))) = density-qbs (density-qbs (Uniform 0 24) (λx. ennreal (exponential-density (f x) (1 / 60)))) (λx. 1 / (∫⁺x∈{0<..<<24}. ennreal (f x * exp (- 1 / 60 * f x) / 24) ∂lborel)*)
using *normalize-qbs[of density-qbs (Uniform 0 24) (λx. exponential-density (f x) (1 / 60)) qbs-borel,simplified]* **by**(*simp add: qbs-ld*)
also have ... = *density-qbs (Uniform 0 24) (λx. ennreal (exponential-density (f x) (1 / 60)) / (∫⁺x∈{0<..<<24}. ennreal (f x * exp (- (f x / 60)) / 24) ∂lborel)*)
by(*simp add: density-qbs-density-qbs-eq[of - qbs-borel] ennreal-times-divide*)

finally have $\mathcal{P}(x \text{ in whattime } f. x \in U) = \text{measure } (\text{density } (\text{qbs-l } (\text{Uniform } 0 \ 24)) (\lambda x. \text{ennreal } (\text{exponential-density } (f \ x) \ (1 / 60)) / (\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- (f \ x / 60)) / 24) \ \partial \text{lborel}))) \ U$
unfolding 1 by (*simp add: whattime-def query-def qbs-l-density-qbs[of - qbs-borel]*)
also have ... = *enn2real ((∫⁺x∈{0<..<<24}. (ennreal (exponential-density (f x) (1 / 60)) / (∫⁺x∈{0<..<<24}. ennreal (f x * exp (- (f x / 60)) / 24) ∂lborel) * indicator U x) ∂lborel) / 24)*
by(*simp add: measure-def emeasure-density nn-integral-uniform-measure*)
also have ... = *enn2real ((∫⁺x∈{0<..<<24}. (ennreal (exponential-density (f x) (1 / 60)) * indicator U x) ∂lborel) / (∫⁺x∈{0<..<<24}. ennreal (f x * exp (- (f x / 60)) / 24) ∂lborel) / 24)*
by(*simp add: ennreal-divide-times ennreal-times-divide nn-integral-divide*)
also have ... = *enn2real (ennreal (∫ x∈{0<..<<24} ∩ U. (exponential-density (f x) (1 / 60)) ∂lborel) / ennreal (∫ x∈{0<..<<24}. (f x * exp (- (f x / 60)) / 24) ∂lborel) / ennreal 24)*
proof -
have 1: $(\int^+ x \in \{0 < .. < 24\}. \text{ennreal } (f \ x * \exp (- (f \ x / 60)) / 24) \ \partial \text{lborel}) = \text{ennreal } (\int x \in \{0 < .. < 24\}. (f \ x * \exp (- (f \ x / 60)) / 24) \ \partial \text{lborel})$ **(is ?l = ?r)**
proof -
have ?l = $(\int^+ x. \text{ennreal } (f \ x * \exp (- (f \ x / 60)) / 24 * \text{indicat-real } \{0 < .. < 24\} \ x) \ \partial \text{lborel})$
by (*simp add: nn-integral-set-ennreal*)
also have ... = *ennreal (∫ x. (f x * exp (- (f x / 60)) / 24 * indicat-real {0<..<<24} x) ∂lborel)*
by(*rule nn-integral-eq-integral*) (*use int assms(3) in auto*)
also have ... = ?r
by(*auto simp: set-lebesgue-integral-def intro!: Bochner-Integration.integral-cong ennreal-cong*)
finally show ?thesis .
qed
have 2: $(\int^+ x \in \{0 < .. < 24\}. (\text{ennreal } (\text{exponential-density } (f \ x) \ (1 / 60)) * \text{indicator } U \ x) \ \partial \text{lborel}) = \text{ennreal } (\int x \in \{0 < .. < 24\} \cap U. (\text{exponential-density } (f \ x) \ (1 / 60)) \ \partial \text{lborel})$ **(is ?l = ?r)**
proof -
have ?l = $(\int^+ x. \text{ennreal } (f \ x * \exp (- (f \ x / 60)) * \text{indicat-real } \{0 < .. < 24\} \ x * \text{indicator } U \ x) \ \partial \text{lborel})$
by (*auto intro!: nn-integral-cong simp: exponential-density-def indicator-def*)

also have ... = $\text{ennreal} (\int x. (f x * \exp (- (f x / 60))) * \text{indicat-real} \{0 < .. < 24\})$
 $x * \text{indicator } U x) \partial \text{lborel}$
by(*rule nn-integral-eq-integral*) (*use integrable-real-mult-indicator*[*OF - int*]
assms(*?*) **in auto**)
also have ... = ?*r*
by(*auto simp: set-lebesgue-integral-def indicator-def exponential-density-def*
intro!: Bochner-Integration.integral-cong ennreal-cong)
finally show ?*thesis* .
qed
show ?*thesis*
by(*simp add: 1 2*)
qed
also have ... = $\text{enn2real} (\text{ennreal} ((\int x \in \{0 < .. < 24\} \cap U. (\text{exponential-density} (f x) (1 / 60)) \partial \text{lborel}) / (\int x \in \{0 < .. < 24\}. (f x * \exp (- (f x / 60))) / 24) \partial \text{lborel}) / 24))$
by(*simp only: divide-ennreal*[*OF ge2 ge*] *divide-ennreal*[*OF divide-nonneg-pos*][*OF ge2 ge*]])
also have ... = $(\int x \in \{0 < .. < 24\} \cap U. (\text{exponential-density} (f x) (1 / 60)) \partial \text{lborel}) / (\int x \in \{0 < .. < 24\}. (f x * \exp (- (f x / 60))) / 24) \partial \text{lborel}) / 24$
by(*rule enn2real-ennreal*) (*use ge ge2 in auto*)
also have ... = $(\int x \in \{0 < .. < 24\} \cap U. (f x * \exp (- 1 / 60 * f x)) \partial \text{lborel}) / N$
by(*auto simp: N-def exponential-density-def*)
finally show ?*thesis* .
qed

5.2.12 Distributions on Functions

definition *a-times-x* :: (*real* \Rightarrow *real*) *qbs-measure* **where**

a-times-x \equiv *do* {
a \leftarrow *Uniform* (-*2*) *2*;
return-qbs ($\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$) ($\lambda x. a * x$)
}

lemma *a-times-x-qbs*[*qbs*]: *a-times-x* \in *monadM-qbs* ($\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$)

by(*simp add: a-times-x-def*)

lemma *a-times-x-qbsP*: *a-times-x* \in *monadP-qbs* ($\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$)

proof -

note [*qbs*] = *Uniform-qbsP*[*of -2 2, simplified*] *return-qbs-morphismP* *bind-qbs-morphismP*

show ?*thesis*

by(*simp add: a-times-x-def*)

qed

definition *a-times-x'* :: (*real* \Rightarrow *real*) *qbs-measure* **where**

a-times-x' \equiv *do* {
condition a-times-x ($\lambda f. f 1 \geq 0$)
}

lemma *a-times-x'-qbs*[*qbs*]: *a-times-x'* \in *monadM-qbs* ($\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q$)

by(*simp add: a-times-x'-def*)

lemma *prob-a-times-x*:

assumes [*measurable*]: *Measurable.pred borel P*

shows $\mathcal{P}(f \text{ in } a\text{-times-}x. P (f r)) = \mathcal{P}(a \text{ in } \text{Uniform } (-2) 2. P (a * r))$ (**is** *?lhs = ?rhs*)

proof –

have [*qbs*]: *qbs-pred qbs-borel P*

using *r-preserves-morphisms* **by** *fastforce*

have *?lhs = measure a-times-x ({f. P (f r)} ∩ space a-times-x)*

by (*simp add: Collect-conj-eq inf-sup-aci(1)*)

also have ... = $(\int_Q f. \text{indicat-real } \{f. P (f r)\} f \partial a\text{-times-}x)$

by(*simp add: qbs-integral-def2-l*)

also have ... = *qbs-integral (Uniform (-2) 2) (indicat-real {f. P (f r)} ∘ (*)*)

unfolding *a-times-x-def* **by**(*rule qbs-integral-bind-return[of - qbs-borel]*) *auto*

also have ... = $(\int_Q a. \text{indicat-real } \{a. P (a * r)\} a \partial \text{Uniform } (-2) 2)$

by(*auto simp: comp-def indicator-def*)

also have ... = *?rhs*

by (*simp add: qbs-integral-def2-l*)

finally show *?thesis* .

qed

lemma $\mathcal{P}(f \text{ in } a\text{-times-}x'. f 1 \geq 1) = 1 / 2$ (**is** *?P = -*)

proof –

have *?P = P(f in a-times-x. f 1 ≥ 1 | f 1 ≥ 0)*

by(*simp add: query-Bayes[OF a-times-x-qbsP] a-times-x'-def*)

also have ... = $\mathcal{P}(f \text{ in } a\text{-times-}x. f 1 \geq 1) / \mathcal{P}(f \text{ in } a\text{-times-}x. f 1 \geq 0)$

by(*auto simp add: cond-prob-def*) (*meson dual-order.trans linordered-nonzero-semiring-class.zero-le-one*)

also have ... = $1 / 2$

proof –

have [*simp*]: $\{-2 < .. < 2 :: \text{real}\} \cap \text{Collect } ((\leq) 1) = \{1 .. < 2\} \{-2 < .. < 2 :: \text{real}\} \cap \text{Collect } ((\leq) 0) = \{0 .. < 2\}$

by *auto*

show *?thesis*

by(*auto simp: prob-a-times-x*)

qed

finally show *?thesis* .

qed

Almost everywhere, integrable, and integrations are also interpreted as programs.

lemma $(\lambda g f x. \text{if } (AE_Q y \text{ in } g x. f x y \neq \infty) \text{ then } (\int^+_Q y. f x y \partial(g x)) \text{ else } 0)$

$\in (\mathbb{R}_Q \Rightarrow_Q \text{monadM-qbs } \mathbb{R}_Q) \Rightarrow_Q (\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q \Rightarrow_Q \mathbb{R}_{Q \geq 0}) \Rightarrow_Q \mathbb{R}_Q \Rightarrow_Q$

$\mathbb{R}_{Q \geq 0}$

by *simp*

lemma $(\lambda g f x. \text{if } \text{qbs-integrable } (g x) (f x) \text{ then } \text{Some } (\int_Q y. f x y \partial(g x)) \text{ else } \text{None})$

$\in (\mathbb{R}_Q \Rightarrow_Q \text{monadM-qbs } \mathbb{R}_Q) \Rightarrow_Q (\mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q \Rightarrow_Q \mathbb{R}_Q) \Rightarrow_Q \mathbb{R}_Q \Rightarrow_Q$

option-qbs \mathbb{R}_Q
by *simp*

end

References

- [1] R. Affeldt, C. Cohen, and A. Saito. Semantics of probabilistic programs using s-finite kernels in Coq. In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2023*, page 316, New York, NY, USA, 2023. Association for Computing Machinery.
- [2] A. Sampson. Probabilistic programming. <http://adriansampson.net/doc/ppl.html>. Accessed: January 25. 2023.
- [3] T. Sato, A. Aguirre, G. Barthe, M. Gaboardi, D. Garg, and J. Hsu. Formal verification of higher-order probabilistic programs: reasoning about approximation, convergence, bayesian inference, and optimization. *Proceedings of the ACM on Programming Languages*, 3(POPL):130, Jan 2019.
- [4] S. Staton. Commutative semantics for probabilistic programming. In H. Yang, editor, *Programming Languages and Systems*, pages 855–879, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.
- [5] S. Staton. *Probabilistic Programs as Measures*, page 4374. Cambridge University Press, 2020.
- [6] H. Yang. Semantics of higher-order probabilistic programs with continuous distributions. https://alfa.di.uminho.pt/~nevrenato/probprogschool_slides/Hongseok.pdf. Accessed: February 8. 2023.