Generic Construction of Probability Spaces for Paths of Stochastic Processes in Isabelle/HOL

Fabian Immler

October 13, 2012

Abstract

Stochastic processes are used in probability theory to describe the evolution of random systems over time. The principal mathematical problem is the construction of a probability space for the paths of stochastic processes. The Daniell-Kolmogorov theorem solves this problem: it shows how a family of finite-dimensional distributions defines the distribution of the stochastic process. The construction is generic, i.e., it works for discrete time as well as for continuous time.

Starting from the existing formalizations of measure theory and product probability spaces in Isabelle/HOL, we provide a formal proof of the Daniell-Kolmogorov theorem in Isabelle/HOL. This requires us to formalize concepts from topology, namely polish spaces and regularity of measures on polish spaces.

These results can serve as a foundation to formalize for example discrete-time or continuous-time Markov chains, Markov decision processes, or physical phenomena like Brownian motion.

This work is described in the Master's thesis of Immler [1]

Contents

theory *Auxiliarities*
imports *Probability*
begin

1 Auxiliarities

1.1 Functions: Injective and Inverse

lemma *inj-on-vimage-image-eq*:
  assumes *inj-on* $f\ X\ A \subseteq X$ **shows** $f\ -`\ f\ `\ A \cap X = A$
⟨*proof*⟩

lemma *inv-into-inv-into-superset-eq*:
  assumes *inj-on* $f\ B$
  assumes *bij-betw* $f\ A\ A'\ a \in A\ A \subseteq B$
  shows *inv-into* A' (*inv-into* $B\ f$) $a = f\ a$
⟨*proof*⟩

lemma *f-inv-into-onto*:
  fixes $f{::}'a \Rightarrow 'b$ **and** $A{::}'a$ *set* **and** $B{::}'b$ *set*
  assumes *inj-on* $f\ A\ B \subseteq f\ `\ A$

shows f ‘ *inv-into* A f ‘ B = B
⟨*proof*⟩

lemma *inj-on-image-subset-iff*: *inj-on* f $(A \cup B)$ ==> $(f`A$ <= $f`B)$ = $(A$<=$B)$
⟨*proof*⟩

lemma *inv-into-eq*:
assumes *inj-on* f A *inj-on* g A
assumes $x \in g$ ‘ A
assumes $\bigwedge i.\ i \in A \Longrightarrow f\ i = g\ i$
shows *inv-into* A f x = *inv-into* A g x
⟨*proof*⟩

lemma *inv-into-eq'*:
assumes *inj-on* f A *inj-on* f B
assumes $x \in f$ ‘ $(A \cap B)$
shows *inv-into* A f x = *inv-into* B f x
⟨*proof*⟩

1.2 Topology

lemma *borel-def-closed*: *borel* = *sigma UNIV* (*Collect closed*)
⟨*proof*⟩

lemma *compactE'*:
assumes *compact* S $\forall n \geq m.\ f\ n \in S$
obtains l r **where** $l \in S$ *subseq* r $((f \circ r)$ −−−> $l)$ *sequentially*
⟨*proof*⟩

lemma *compact-Union* [*intro*]: *finite* $S \Longrightarrow \forall T{\in}S.$ *compact* $T \Longrightarrow$ *compact* $(\bigcup S)$
⟨*proof*⟩

lemma *closed-UN* [*intro*]: *finite* $A \Longrightarrow \forall x{\in}A.$ *compact* $(B\ x) \Longrightarrow$ *compact* $(\bigcup x{\in}A.$ $B\ x)$
⟨*proof*⟩

1.3 Measures

lemma
UN-finite-countable-eq-Un:
fixes f :: *'a::countable set* ⇒ -
assumes $\bigwedge s.\ P\ s \Longrightarrow$ *finite* s
shows $\bigcup\{f\ s | s.\ P\ s\}$ = $(\bigcup n$::*nat*. *let* s = *set* (*from-nat* n) *in if* $P\ s$ *then* $f\ s$ *else* $\{\})$
⟨*proof*⟩

lemma
countable-finite-comprehension:
fixes f :: *'a::countable set* ⇒ -
assumes $\bigwedge s.\ P\ s \Longrightarrow$ *finite* s

assumes $\bigwedge s.\ P\ s \Longrightarrow f\ s \in$ *sets M*
shows $\bigcup\{f\ s|s.\ P\ s\} \in$ *sets M*
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *union*:
assumes *f*: *positive M f additive M f* **and** $A \in M\ B \in M$
shows $f\ (A \cup B) = f\ A + f\ (B - A)$
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *plus*:
assumes *f*: *positive M f additive M f* **and** $A \in M\ B \in M$
shows $f\ B = f\ (A \cap B) + f\ (B - A)$
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *union-inter-minus-equality*:
assumes *f*: *positive M f additive M f* **and** $A \in M\ B \in M$
shows $f\ (A \cup B) + f\ (A \cap B) + f\ (B - A) = f\ A + f\ B + f\ (B - A)$
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *union-plus-inter-equality*:
assumes *f*: *positive M f additive M f* **and** $A \in M\ B \in M$
shows $f\ (A \cup B) + f\ (A \cap B) = f\ A + f\ B$
⟨*proof*⟩

lemma *emeasure-union-plus-inter-equality*:
assumes $A \in$ *sets M* $B \in$ *sets M*
shows $M\ (A \cup B) + M\ (A \cap B) = M\ A + M\ B$
⟨*proof*⟩

lemma (**in** *finite-measure*) *measure-union*:
assumes $A \in$ *sets M* $B \in$ *sets M*
shows *measure M* $(A \cup B) =$ *measure M A* + *measure M B* − *measure M* $(A$
$\cap\ B)$
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *subtractive*:
assumes *f*: *positive M f additive M f* **and** $A \in M\ B \in M$ **and** $A \subseteq B$
and $f\ A < \infty$
shows $f\ (B - A) = f\ B - f\ A$
⟨*proof*⟩

lemma (**in** *ring-of-sets*) *subadditive*:
assumes *f*: *positive M f additive M f* **and** *A*: *range* $A \subseteq M$ **and** *S*: *finite S*
shows $f\ (\bigcup i{\in}S.\ A\ i) \le (\sum i{\in}S.\ f\ (A\ i))$
⟨*proof*⟩

lemma *finite-Union*:
fixes *A*::′*a*::*countable set*
assumes $\bigwedge i.\ i \in A \Longrightarrow B\ i \in$ *sigma-sets sp C*

shows $\bigcup B$ ‘ $A \in$ *sigma-sets sp C*
⟨*proof*⟩

1.4 Enumeration of Finite Set

definition *enum-finite-max J* = (*SOME n*. ∃ *f*. *J* = *f* ‘ {*i*. *i* < *n*} ∧ *inj-on f* {*i*. *i* < *n*})

definition *enum-finite* **where**
enum-finite J =
(*SOME f*. *J* = *f* ‘ {*i*::*nat*. *i* < *enum-finite-max J*} ∧ *inj-on f* {*i*. *i* < *enum-finite-max J*})

lemma *enum-finite-max*:
assumes *finite J*
shows ∃*f*::*nat*⇒′*a*. *J* = *f* ‘ {*i*. *i* < *enum-finite-max J*} ∧ *inj-on f* {*i*. *i* < *enum-finite-max J*}
⟨*proof*⟩

lemma *enum-finite*:
assumes *finite J*
shows *J* = *enum-finite J* ‘ {*i*::*nat*. *i* < *enum-finite-max J*} ∧
inj-on (*enum-finite J*) {*i*::*nat*. *i* < *enum-finite-max J*}
⟨*proof*⟩

lemma *in-set-enum-exist*:
assumes *finite A*
assumes *y* ∈ *A*
shows ∃ *i*. *y* = *enum-finite A i*
⟨*proof*⟩

1.5 Enumeration of Countable Union of Finite Sets

locale *finite-set-sequence* =
fixes *Js*::*nat* ⇒ ′*a set*
assumes *finite-seq*[*simp*]: *finite* (*Js n*)
begin

definition *set-of-Un* **where** *set-of-Un j* = (*LEAST n*. *j* ∈ *Js n*)

definition *index-in-set* **where** *index-in-set J j* = (*SOME n*. *j* = *enum-finite J n*)

definition *Un-to-nat* **where**
Un-to-nat j = *to-nat* (*set-of-Un j*, *index-in-set* (*Js* (*set-of-Un j*)) *j*)

lemma *inj-on-Un-to-nat*:
shows *inj-on Un-to-nat* ($\bigcup$*n*::*nat*. *Js n*)
⟨*proof*⟩

lemma *inj-Un*[*simp*]:

shows *inj-on* (*Un-to-nat*) (*Js n*)
⟨*proof*⟩

lemma *Un-to-nat-injectiveD*:
assumes *Un-to-nat* x = *Un-to-nat* y
assumes $x \in Js\ i\ y \in Js\ j$
shows $x = y$
⟨*proof*⟩

end

1.6 Sequence of Properties on Subsequences

lemma *subseq-mono*: **assumes** *subseq* $r\ m < n$ **shows** $r\ m < r\ n$
⟨*proof*⟩

locale *subseqs* =
fixes P::*nat*⇒(*nat*⇒*nat*)⇒(*nat*⇒*nat*)⇒*bool*
assumes *ex-subseq*: $\bigwedge n\ s.$ *subseq* $s \Longrightarrow \exists r'.$ *subseq* $r' \wedge P\ n\ s\ r'$
begin

primrec *seqseq* **where**
seqseq 0 = *id*
| *seqseq* (*Suc n*) = *seqseq n o* (*SOME* r'. *subseq* $r' \wedge P\ n$ (*seqseq n*) r')

lemma *seqseq-ex*:
shows *subseq* (*seqseq n*) ∧
(∃ r'. *seqseq* (*Suc n*) = *seqseq n o* r' ∧ *subseq* r' ∧ $P\ n$ (*seqseq n*) r')
⟨*proof*⟩

lemma *subseq-seqseq*:
shows *subseq* (*seqseq n*) ⟨*proof*⟩

definition *reducer* **where** *reducer n* = (*SOME* r'. *subseq* r' ∧ $P\ n$ (*seqseq n*) r')

lemma *subseq-reducer*: *subseq* (*reducer n*) **and** *reducer-reduces*: $P\ n$ (*seqseq n*) (*reducer n*)
⟨*proof*⟩

lemma *seqseq-reducer*[*simp*]:
seqseq (*Suc n*) = *seqseq n o reducer n*
⟨*proof*⟩

declare *seqseq.simps*(*2*)[*simp del*]

definition *diagseq* **where** *diagseq i* = *seqseq i i*

lemma *diagseq-mono*: *diagseq n* < *diagseq* (*Suc n*)
⟨*proof*⟩

lemma *subseq-diagseq*: *subseq diagseq*
⟨*proof*⟩

primrec *fold-reduce* **where**
fold-reduce n 0 = id
| *fold-reduce n (Suc k) = fold-reduce n k o reducer (n + k)*

lemma *subseq-fold-reduce*: *subseq (fold-reduce n k)*
⟨*proof*⟩

lemma *ex-subseq-reduce-index*: *seqseq (n + k) = seqseq n o fold-reduce n k*
⟨*proof*⟩

lemma *seqseq-fold-reduce*: *seqseq n = fold-reduce 0 n*
⟨*proof*⟩

lemma *diagseq-fold-reduce*: *diagseq n = fold-reduce 0 n n*
⟨*proof*⟩

lemma *fold-reduce-add*: *fold-reduce 0 (m + n) = fold-reduce 0 m o fold-reduce m n*
⟨*proof*⟩

lemma *diagseq-add*: *diagseq (k + n) = (seqseq k o (fold-reduce k n)) (k + n)*
⟨*proof*⟩

lemma *diagseq-sub*:
assumes $m \leq n$ **shows** *diagseq n = (seqseq m o (fold-reduce m (n − m))) n*
⟨*proof*⟩

lemma *subseq-diagonal-rest*: *subseq (λx. fold-reduce k x (k + x))*
⟨*proof*⟩

lemma *diagseq-seqseq*: *diagseq o (op + k) = (seqseq k o (λx. fold-reduce k x (k + x)))*
⟨*proof*⟩

lemma *eventually-sequentially-diagseq*:
assumes *⋀n s r. P n s r = (∀ i. Q n ((s o r) i))*
shows *eventually (λi. Q n (diagseq i)) sequentially*
⟨*proof*⟩

lemma *diagseq-holds*:
assumes *seq-property*: *⋀n s r. P n s r = Q n (s o r)*
assumes *subseq-closed*: *⋀n s r. subseq r ⟹ Q n s ⟹ Q n (s o r)*
shows *P n diagseq (op + (Suc n))*
⟨*proof*⟩

end

1.7 Product Sets

lemma *PiE-def'*: Pi_E *I A* = {*f*. (∀ *i* ∈ *I*. *f i* ∈ *A i*) ∧ *f* = *restrict f I*}
⟨*proof*⟩

lemma *prod-emb-def'*: *prod-emb I M J X* = {*a* ∈ Pi_E *I* (λ*i*. *space* (*M i*)). *restrict a J* ∈ *X*}
⟨*proof*⟩

lemma *prod-emb-subsetI*:
assumes *F* ⊆ *G*
shows *prod-emb A M B F* ⊆ *prod-emb A M B G*
⟨*proof*⟩

end

theory *Polish-Space*
imports *Auxiliarities*
begin

2 Topological Formalizations Leading to Polish Spaces

2.1 Characterization of Compact Sets

lemma *pos-approach-nat*:
fixes *e*::*real*
assumes *0* < *e*
obtains *n*::*nat* **where** *1* / (*Suc n*) < *e*
⟨*proof*⟩

TODO: move to Topology-Euclidean-Space

lemma *compact-eq-totally-bounded*:
shows *compact s* ⟷ *complete s* ∧ (∀ *e*>*0*. ∃ *k*. *finite k* ∧ *s* ⊆ (⋃((λ*x*. *ball x e*) ‘ *k*)))
⟨*proof*⟩

2.2 Infimum Distance

definition *infdist x A* = *Inf* {*dist x a*|*a*. *a* ∈ *A*}

lemma *infdist-nonneg*:
assumes *A* ≠ {}
shows *0* ≤ *infdist x A*
⟨*proof*⟩

lemma *infdist-le*:
assumes $a \in A$
assumes $d = dist\ x\ a$
shows $infdist\ x\ A \leq d$
⟨*proof*⟩

lemma *infdist-zero*[*simp*]:
assumes $a \in A$ **shows** $infdist\ a\ A = 0$
⟨*proof*⟩

lemma *infdist-triangle*:
assumes $A \neq \{\}$
shows $infdist\ x\ A \leq infdist\ y\ A + dist\ x\ y$
⟨*proof*⟩

lemma
in-closure-iff-infdist-zero:
assumes $A \neq \{\}$
shows $x \in closure\ A \longleftrightarrow infdist\ x\ A = 0$
⟨*proof*⟩

lemma
in-closed-iff-infdist-zero:
assumes $closed\ A\ A \neq \{\}$
shows $x \in A \longleftrightarrow infdist\ x\ A = 0$
⟨*proof*⟩

lemma *continuous-infdist*:
assumes $A \neq \{\}$
shows $continuous\ (at\ x)\ (\lambda x.\ infdist\ x\ A)$
⟨*proof*⟩

2.3 Topological Basis

context *topological-space*
begin

definition $topological\text{-}basis\ B =$
$((\forall\, b{\in}B.\ open\ b) \wedge (\forall\, x.\ open\ x \longrightarrow (\exists\, B'.\ B' \subseteq B \wedge Union\ B' = x)))$

lemma *topological-basis-iff*:
assumes $\bigwedge B'.\ B' \in B \Longrightarrow open\ B'$
shows $topological\text{-}basis\ B \longleftrightarrow (\forall\, O'.\ open\ O' \longrightarrow (\forall\, x{\in}O'.\ \exists\, B'{\in}B.\ x \in B' \wedge B' \subseteq O'))$
(**is** - $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

lemma *topological-basisI*:
assumes $\bigwedge B'.\ B' \in B \Longrightarrow open\ B'$

assumes $\bigwedge O'\ x.\ open\ O' \Longrightarrow x \in O' \Longrightarrow \exists B'{\in}B.\ x \in B' \wedge B' \subseteq O'$
shows *topological-basis B*
⟨*proof*⟩

lemma *topological-basisE*:
fixes O'
assumes *topological-basis B*
assumes *open* O'
assumes $x \in O'$
obtains B' **where** $B' \in B\ x \in B'\ B' \subseteq O'$
⟨*proof*⟩

end

2.4 Enumerable Basis

class *enumerable-basis* = *topological-space* +
assumes *ex-enum-basis*: $\exists f{::}nat \Rightarrow 'a\ set.\ topological\text{-}basis\ (range\ f)$
begin

definition *enum-basis'*::*nat* ⇒ *'a set*
where *enum-basis'* = *Eps* (*topological-basis o range*)

lemma *enumerable-basis'*: *topological-basis* (*range enum-basis'*)
⟨*proof*⟩

lemmas *enumerable-basisE'* = *topological-basisE*[*OF enumerable-basis'*]

Extend enumeration of basis, such that it is closed under (finite) Union

definition *enum-basis*::*nat* ⇒ *'a set*
where *enum-basis n* = $\bigcup$(*set* (*map enum-basis'* (*from-nat n*)))

lemma
open-enum-basis:
assumes $B \in$ *range enum-basis*
shows *open B*
⟨*proof*⟩

lemma *enumerable-basis*: *topological-basis* (*range enum-basis*)
⟨*proof*⟩

lemmas *enumerable-basisE* = *topological-basisE*[*OF enumerable-basis*]

lemma *open-enumerable-basis-ex*:
assumes *open X*
shows $\exists N.\ X = (\bigcup n{\in}N.\ enum\text{-}basis\ n)$
⟨*proof*⟩

lemma *open-enumerable-basisE*:

assumes *open X*
obtains *N* **where** $X = (\bigcup n{\in}N.\ enum\text{-}basis\ n)$
⟨*proof*⟩

Construction of an Increasing Sequence Approximating Open Sets

lemma *empty-basisI*[*intro*]: $\{\} \in range\ enum\text{-}basis$
⟨*proof*⟩

lemma *union-basisI*[*intro*]:
assumes $A \in range\ enum\text{-}basis$ $B \in range\ enum\text{-}basis$
shows $A \cup B \in range\ enum\text{-}basis$
⟨*proof*⟩

lemma *open-imp-Union-of-incseq*:
assumes *open X*
shows $\exists S.\ incseq\ S \wedge (\bigcup j.\ S\ j) = X \wedge range\ S \subseteq range\ enum\text{-}basis$
⟨*proof*⟩

lemma *open-incseqE*:
assumes *open X*
obtains *S* **where** $incseq\ S\ (\bigcup j.\ S\ j) = X\ range\ S \subseteq range\ enum\text{-}basis$
⟨*proof*⟩

end

lemma *borel-eq-sigma-enum-basis*:
sets borel = *sigma-sets* (*space borel*) (*range enum-basis*)
⟨*proof*⟩

lemma *countable-dense-set*:
shows $\exists x{::}nat \Rightarrow$ -. $\forall (y{::}'a{::}enumerable\text{-}basis\ set).\ open\ y \longrightarrow y \neq \{\} \longrightarrow (\exists n.\ x\ n \in y)$
⟨*proof*⟩

lemma *countable-dense-setE*:
obtains $x :: nat \Rightarrow$ -
where $\bigwedge(y{::}'a{::}enumerable\text{-}basis\ set).\ open\ y \Longrightarrow y \neq \{\} \Longrightarrow \exists n.\ x\ n \in y$
⟨*proof*⟩

2.5 Polish Spaces

Textbooks define Polish spaces as completely metrizable. We assume the topology to be complete for a given metric.

class *polish-space* = *complete-space* + *enumerable-basis*

TODO: Rules in *Topology-Euclidean-Space* should be proved in the *ordered-euclidean-space* locale! Then we can use subclass instead of instance.

instance *ordered-euclidean-space* ⊆ *polish-space*

⟨*proof*⟩

instantiation *nat*::*topological-space*
begin

definition *open-nat*::*nat set* ⇒ *bool*
  where *open-nat s* = *True*

instance ⟨*proof*⟩
end

instantiation *nat*::*metric-space*
begin

definition *dist-nat*::*nat* ⇒ *nat* ⇒ *real*
  where *dist-nat n m* = (*if n* = *m then 0 else 1*)

instance ⟨*proof*⟩
end

instance *nat*::*complete-space*
⟨*proof*⟩

instance *nat*::*polish-space*
⟨*proof*⟩

2.6 Regularity of Measures

lemma *ereal-approx-SUP*:
  fixes x::*ereal*
  assumes *A-notempty*: $A \neq \{\}$
  assumes *f-bound*: $\bigwedge i.\ i \in A \Longrightarrow f\ i \leq x$
  assumes *f-fin*: $\bigwedge i.\ i \in A \Longrightarrow f\ i \neq \infty$
  assumes *f-nonneg*: $\bigwedge i.\ 0 \leq f\ i$
  assumes *approx*: $\bigwedge e.\ (e::real) > 0 \Longrightarrow \exists\, i \in A.\ x \leq f\ i + e$
  shows $x = (SUP\ i : A.\ f\ i)$
⟨*proof*⟩

lemma *ereal-approx-INF*:
  fixes x::*ereal*
  assumes *A-notempty*: $A \neq \{\}$
  assumes *f-bound*: $\bigwedge i.\ i \in A \Longrightarrow x \leq f\ i$
  assumes *f-fin*: $\bigwedge i.\ i \in A \Longrightarrow f\ i \neq \infty$
  assumes *f-nonneg*: $\bigwedge i.\ 0 \leq f\ i$
  assumes *approx*: $\bigwedge e.\ (e::real) > 0 \Longrightarrow \exists\, i \in A.\ f\ i \leq x + e$
  shows $x = (INF\ i : A.\ f\ i)$
⟨*proof*⟩

lemma *INF-approx-ereal*:

fixes *x*::*ereal* **and** *e*::*real*
assumes $e > 0$
assumes *INF*: $x = (INF\ i : A.\ f\ i)$
assumes $|x| \neq \infty$
shows $\exists\, i \in A.\ f\ i < x + e$
⟨*proof*⟩

lemma *SUP-approx-ereal*:
fixes *x*::*ereal* **and** *e*::*real*
assumes $e > 0$
assumes *SUP*: $x = (SUP\ i : A.\ f\ i)$
assumes $|x| \neq \infty$
shows $\exists\, i \in A.\ x \leq f\ i + e$
⟨*proof*⟩

lemma
fixes *M*::′*a*::*polish-space measure*
assumes *sb*: *sets M* = *sets borel*
assumes *emeasure M* (*space M*) $\neq \infty$
assumes $B \in$ *sets borel*
shows *inner-regular*: *emeasure M B* =
(*SUP K* : {*K*. $K \subseteq B \wedge$ *compact K*}. *emeasure M K*) (**is** *?inner B*)
and *outer-regular*: *emeasure M B* =
(*INF U* : {*U*. $B \subseteq U \wedge$ *open U*}. *emeasure M U*) (**is** *?outer B*)
⟨*proof*⟩

end

theory *Fin-Map*
imports *Auxiliarities Polish-Space*
begin

3 Finite Maps

typedef (**open**) (′*i*, ′*a*) *finmap* ((- $\Rightarrow_F$ /-) [*22*, *21*] *21*) =
{(*I*::′*i set*, *f*::′*i* ⇒ ′*a*). *finite I* ∧ *f* ∈ *extensional I*} ⟨*proof*⟩
print-theorems

3.1 Domain and Application

definition *domain* **where** *domain P* = *fst* (*Rep-finmap P*)

lemma *finite-domain*[*simp*, *intro*]: *finite* (*domain P*)
⟨*proof*⟩

definition *proj* (-$_F$ [*1000*] *1000*) **where** *proj P i* = *snd* (*Rep-finmap P*) *i*

declare [[*coercion proj*]]

lemma *extensional-proj*[*simp*, *intro*]: $(P)_F \in$ *extensional* (*domain P*)
⟨*proof*⟩

lemma *proj-undefined*[*simp*, *intro*]: $i \notin$ *domain P* $\Longrightarrow$ *P i* = *undefined*
⟨*proof*⟩

lemma *finmap-eq-iff*: *P* = *Q* $\longleftrightarrow$ (*domain P* = *domain Q* $\wedge$ ($\forall$ *i*$\in$*domain P*. *P i* = *Q i*))
⟨*proof*⟩

3.2 Countable Finite Maps

instance *finmap* :: (*countable*, *countable*) *countable*
⟨*proof*⟩

3.3 Constructor of Finite Maps

definition *finmap-of inds f* = *Abs-finmap* (*inds*, *restrict f inds*)

lemma *proj-finmap-of*[*simp*]:
assumes *finite inds*
shows $(\textit{finmap-of inds } f)_F$ = *restrict f inds*
⟨*proof*⟩

lemma *domain-finmap-of*[*simp*]:
assumes *finite inds*
shows *domain* (*finmap-of inds f*) = *inds*
⟨*proof*⟩

lemma *finmap-of-eq-iff*[*simp*]:
assumes *finite i finite j*
shows *finmap-of i m* = *finmap-of j n* $\longleftrightarrow$ *i* = *j* $\wedge$ *restrict m i* = *restrict n i*
⟨*proof*⟩

lemma
finmap-of-inj-on-extensional-finite:
assumes *finite K*
assumes *S* $\subseteq$ *extensional K*
shows *inj-on* (*finmap-of K*) *S*
⟨*proof*⟩

lemma *finmap-choice*:
assumes $*$: $\bigwedge i.\ i \in I \Longrightarrow \exists x.\ P\ i\ x$ **and** *I*: *finite I*
shows $\exists$ *fm*. *domain fm* = *I* $\wedge$ ($\forall$ *i*$\in$*I*. *P i* (*fm i*))
⟨*proof*⟩

3.4 Product set of Finite Maps

This is *Pi* for Finite Maps, most of this is copied

definition $Pi' :: 'i\ set \Rightarrow ('i \Rightarrow 'a\ set) \Rightarrow ('i \Rightarrow_F 'a)\ set$ **where**
$Pi'\ I\ A = \{\ P.\ domain\ P = I \wedge (\forall i.\ i \in I \longrightarrow (P)_F\ i \in A\ i)\ \}$

syntax
-Pi' :: [*pttrn*, *'a set*, *'b set*] => (*'a* => *'b*) *set* ((*3PI'* -:-./ -) *10*)

syntax (*xsymbols*)
-Pi' :: [*pttrn*, *'a set*, *'b set*] => (*'a* => *'b*) *set* ((*3*Π*'* -∈-./ -) *10*)

syntax (*HTML* **output**)
-Pi' :: [*pttrn*, *'a set*, *'b set*] => (*'a* => *'b*) *set* ((*3*Π*'* -∈-./ -) *10*)

translations
PI' x:*A. B* == *CONST Pi' A* (%*x. B*)

abbreviation
finmapset :: [*'a set*, *'b set*] => ($'a \Rightarrow_F 'b$) *set*
(**infixr** ~> *60*) **where**
A ~> *B* ≡ *Pi' A* (%-. *B*)

notation (*xsymbols*)
finmapset (**infixr** ⇝ *60*)

3.4.1 Basic Properties of *Pi'*

lemma *Pi'-I*[*intro*!]: $domain\ f = A \Longrightarrow (\bigwedge x.\ x \in A \Longrightarrow f\ x \in B\ x) \Longrightarrow f \in Pi'\ A\ B$
⟨*proof*⟩

lemma *Pi'-I'*[*simp*]: $domain\ f = A \Longrightarrow (\bigwedge x.\ x \in A \longrightarrow f\ x \in B\ x) \Longrightarrow f \in Pi'\ A\ B$
⟨*proof*⟩

lemma *finmapsetI*: $domain\ f = A \Longrightarrow (\bigwedge x.\ x \in A \Longrightarrow f\ x \in B) \Longrightarrow f \in A \leadsto B$
⟨*proof*⟩

lemma *Pi'-mem*: $f \in Pi'\ A\ B \Longrightarrow x \in A \Longrightarrow f\ x \in B\ x$
⟨*proof*⟩

lemma *Pi'-iff*: $f \in Pi'\ I\ X \longleftrightarrow domain\ f = I \wedge (\forall i \in I.\ f\ i \in X\ i)$
⟨*proof*⟩

lemma *Pi'E* [*elim*]:
$f \in Pi'\ A\ B \Longrightarrow (f\ x \in B\ x \Longrightarrow domain\ f = A \Longrightarrow Q) \Longrightarrow (x \notin A \Longrightarrow Q) \Longrightarrow Q$
⟨*proof*⟩

lemma *in-Pi′-cong*:
domain f = *domain g* $\Longrightarrow$ ($\bigwedge$ *w*. *w* $\in$ *A* $\Longrightarrow$ *f w* = *g w*) $\Longrightarrow$ *f* $\in$ *Pi′ A B* $\longleftrightarrow$ *g* $\in$ *Pi′ A B*
⟨*proof*⟩

lemma *funcset-mem*: [|*f* $\in$ *A* $\rightsquigarrow$ *B*; *x* $\in$ *A*|] ==> *f x* $\in$ *B*
⟨*proof*⟩

lemma *funcset-image*: *f* $\in$ *A* $\rightsquigarrow$ *B* ==> *f* ‘ *A* $\subseteq$ *B*
⟨*proof*⟩

lemma *Pi′-eq-empty*[*simp*]:
assumes *finite A* **shows** (*Pi′ A B*) = {} $\longleftrightarrow$ ($\exists$ *x*$\in$*A*. *B x* = {})
⟨*proof*⟩

lemma *Pi′-mono*: ($\bigwedge$*x*. *x* $\in$ *A* $\Longrightarrow$ *B x* $\subseteq$ *C x*) $\Longrightarrow$ *Pi′ A B* $\subseteq$ *Pi′ A C*
⟨*proof*⟩

lemma *Pi-Pi′*: *finite A* $\Longrightarrow$ (Pi_E *A B*) = *proj* ‘ *Pi′ A B*
⟨*proof*⟩

3.5 Metric Space of Finite Maps

instantiation *finmap* :: (*type*, *metric-space*) *metric-space*
begin

definition *dist-finmap* **where**
dist P Q = ($\sum$ *i*$\in$*domain P* $\cup$ *domain Q*. *dist* ((P)$_F$ *i*) ((Q)$_F$ *i*)) +
card ((*domain P* − *domain Q*) $\cup$ (*domain Q* − *domain P*))

lemma *dist-finmap-extend*:
assumes *finite X*
shows *dist P Q* = ($\sum$ *i*$\in$*domain P* $\cup$ *domain Q* $\cup$ *X*. *dist* ((P)$_F$ *i*) ((Q)$_F$ *i*)) +
card ((*domain P* − *domain Q*) $\cup$ (*domain Q* − *domain P*))
⟨*proof*⟩

definition *open-finmap* :: (′*a* $\Rightarrow_F$ ′*b*) *set* $\Rightarrow$ *bool* **where**
open-finmap S = ($\forall$ *x*$\in$*S*. $\exists$ *e*>*0*. $\forall$ *y*. *dist y x* < *e* $\longrightarrow$ *y* $\in$ *S*)

lemma *add-eq-zero-iff*[*simp*]:
fixes *a b*::*real*
assumes *a* $\geq$ *0* *b* $\geq$ *0*
shows *a* + *b* = *0* $\longleftrightarrow$ *a* = *0* $\wedge$ *b* = *0*
⟨*proof*⟩

lemma *dist-le-1-imp-domain-eq*:
assumes *dist P Q* < *1*
shows *domain P* = *domain Q*

⟨*proof*⟩

lemma *dist-proj*:
  shows $dist\ ((x)_F\ i)\ ((y)_F\ i) \leq dist\ x\ y$
⟨*proof*⟩

lemma *open-Pi'I*:
  assumes *open-component*: $\bigwedge i.\ i \in I \Longrightarrow open\ (A\ i)$
  shows $open\ (Pi'\ I\ A)$
⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *open-restricted-space*:
  shows $open\ \{m.\ P\ (domain\ m)\}$
⟨*proof*⟩

lemma *closed-restricted-space*:
  shows $closed\ \{m.\ P\ (domain\ m)\}$
⟨*proof*⟩

lemma *continuous-proj*:
  shows $continuous\text{-}on\ s\ (\lambda x.\ (x)_F\ i)$
  ⟨*proof*⟩

3.6 Complete Space of Finite Maps

lemma *tendsto-dist-zero*:
  assumes $(\lambda i.\ dist\ (f\ i)\ g) \ {-}{-}{-}{-}{>}\ 0$
  shows $f\ {-}{-}{-}{-}{>}\ g$
  ⟨*proof*⟩

lemma *tendsto-dist-zero'*:
  assumes $(\lambda i.\ dist\ (f\ i)\ g) \ {-}{-}{-}{-}{>}\ x$
  assumes $0 = x$
  shows $f\ {-}{-}{-}{-}{>}\ g$
  ⟨*proof*⟩

lemma *tendsto-finmap*:
  fixes $f{::}nat \Rightarrow ('i \Rightarrow_F ('a{::}metric\text{-}space))$
  assumes *ind-f*: $\bigwedge n.\ domain\ (f\ n) = domain\ g$
  assumes *proj-g*: $\bigwedge i.\ i \in domain\ g \Longrightarrow (\lambda n.\ (f\ n)\ i)\ {-}{-}{-}{-}{>}\ g\ i$
  shows $f\ {-}{-}{-}{-}{>}\ g$
  ⟨*proof*⟩

instance *finmap* :: (*type*, *complete-space*) *complete-space*

⟨*proof*⟩

3.7 Polish Space of Finite Maps

instantiation *finmap* :: (*countable*, *polish-space*) *polish-space*
begin

definition *enum-basis-finmap* :: *nat* ⇒ (*'a* $\Rightarrow_F$ *'b*) *set* **where**
enum-basis-finmap n =
(*let m* = *from-nat n*::(*'a* $\Rightarrow_F$ *nat*) *in Pi'* (*domain m*) (*enum-basis o* $(m)_F$))

lemma *range-enum-basis-eq*:
range enum-basis-finmap = {*Pi' I S*|*I S*. *finite I* ∧ (∀ *i* ∈ *I*. *S i* ∈ *range enum-basis*)}
⟨*proof*⟩

lemma *in-enum-basis-finmapI*:
assumes *finite I* **assumes** ⋀*i*. *i* ∈ *I* ⟹ *S i* ∈ *range enum-basis*
shows *Pi' I S* ∈ *range enum-basis-finmap*
⟨*proof*⟩

lemma *finmap-topological-basis*:
topological-basis (*range* (*enum-basis-finmap*))
⟨*proof*⟩

lemma *range-enum-basis-finmap-imp-open*:
assumes *x* ∈ *range enum-basis-finmap*
shows *open x*
⟨*proof*⟩

lemma
open-imp-ex-UNION-of-enum:
fixes *X*::(*'a* $\Rightarrow_F$ *'b*) *set*
assumes *open X* **assumes** *X* ≠ {}
shows ∃ *A*::*nat*⇒*'a set*. ∃ *B*::*nat*⇒(*'a* ⇒ *'b set*) . *X* = *UNION UNIV* (λ*i*. *Pi'* (*A i*) (*B i*)) ∧
(∀ *n*. ∀ *i*∈*A n*. (*B n*) *i* ∈ *range enum-basis*) ∧ (∀ *n*. *finite* (*A n*))
⟨*proof*⟩

lemma
open-imp-ex-UNION:
fixes *X*::(*'a* $\Rightarrow_F$ *'b*) *set*
assumes *open X* **assumes** *X* ≠ {}
shows ∃ *A*::*nat*⇒*'a set*. ∃ *B*::*nat*⇒(*'a* ⇒ *'b set*) . *X* = *UNION UNIV* (λ*i*. *Pi'* (*A i*) (*B i*)) ∧
(∀ *n*. ∀ *i*∈*A n*. *open* ((*B n*) *i*)) ∧ (∀ *n*. *finite* (*A n*))
⟨*proof*⟩

lemma

open-basisE:
assumes *open X* **assumes** $X \neq \{\}$
obtains A::*nat*⇒'*a set* **and** B::*nat*⇒('*a* ⇒ '*b set*) **where**
$X = UNION\ UNIV\ (\lambda i.\ Pi'\ (A\ i)\ (B\ i))$ $\bigwedge n\ i.\ i{\in}A\ n \Longrightarrow open\ ((B\ n)\ i)$ $\bigwedge n.$ $finite\ (A\ n)$
⟨*proof*⟩

lemma
open-basis-of-enumE:
assumes *open X* **assumes** $X \neq \{\}$
obtains A::*nat*⇒'*a set* **and** B::*nat*⇒('*a* ⇒ '*b set*) **where**
$X = UNION\ UNIV\ (\lambda i.\ Pi'\ (A\ i)\ (B\ i))$ $\bigwedge n\ i.\ i{\in}A\ n \Longrightarrow (B\ n)\ i \in range$ *enum-basis*
$\bigwedge n.\ finite\ (A\ n)$
⟨*proof*⟩

instance ⟨*proof*⟩

end

3.8 Product Measurable Space of Finite Maps

definition *PiF I M* ≡
sigma
$(\bigcup J \in I.\ (\Pi'\ j{\in}J.\ space\ (M\ j)))$
$\{(\Pi'\ j{\in}J.\ X\ j)\ |X\ J.\ J \in I \wedge X \in (\Pi\ j{\in}J.\ sets\ (M\ j))\}$

abbreviation
$Pi_F\ I\ M \equiv PiF\ I\ M$

syntax
-PiF :: *pttrn* ⇒ '*i set* ⇒ '*a measure* ⇒ ('*i* => '*a*) *measure* ((*3PIF* -:-./ -) *10*)

syntax (*xsymbols*)
-PiF :: *pttrn* ⇒ '*i set* ⇒ '*a measure* ⇒ ('*i* => '*a*) *measure* ((*3*Π_F -∈-./ -) *10*)

syntax (*HTML* **output**)
-PiF :: *pttrn* ⇒ '*i set* ⇒ '*a measure* ⇒ ('*i* => '*a*) *measure* ((*3*Π_F -∈-./ -) *10*)

translations
PIF x:*I*. *M* == *CONST PiF I* (%*x*. *M*)

lemma *PiF-gen-subset*: $\{(\Pi'\ j{\in}J.\ X\ j)\ |X\ J.\ J \in I \wedge X \in (\Pi\ j{\in}J.\ sets\ (M\ j))\}$ ⊆
Pow $(\bigcup J \in I.\ (\Pi'\ j{\in}J.\ space\ (M\ j)))$
⟨*proof*⟩

lemma *space-PiF*: $space\ (PiF\ I\ M) = (\bigcup J \in I.\ (\Pi'\ j{\in}J.\ space\ (M\ j)))$
⟨*proof*⟩

lemma *sets-PiF*:
  $sets\ (PiF\ I\ M) = sigma\text{-}sets\ (\bigcup J \in I.\ (\Pi'\ j{\in}J.\ space\ (M\ j)))$
  $\{(\Pi'\ j{\in}J.\ X\ j)\ |X\ J.\ J \in I \wedge X \in (\Pi\ j{\in}J.\ sets\ (M\ j))\}$
  ⟨*proof*⟩

lemma *sets-PiF-singleton*:
  $sets\ (PiF\ \{I\}\ M) = sigma\text{-}sets\ (\Pi'\ j{\in}I.\ space\ (M\ j))$
  $\{(\Pi'\ j{\in}I.\ X\ j)\ |X.\ X \in (\Pi\ j{\in}I.\ sets\ (M\ j))\}$
  ⟨*proof*⟩

lemma *in-sets-PiFI*:
  assumes $X = (Pi'\ J\ S)\ J \in I\ \bigwedge i.\ i{\in}J \Longrightarrow S\ i \in sets\ (M\ i)$
  shows $X \in sets\ (PiF\ I\ M)$
  ⟨*proof*⟩

lemma *product-in-sets-PiFI*:
  assumes $J \in I\ \bigwedge i.\ i{\in}J \Longrightarrow S\ i \in sets\ (M\ i)$
  shows $(Pi'\ J\ S) \in sets\ (PiF\ I\ M)$
  ⟨*proof*⟩

lemma *singleton-space-subset-in-sets*:
  fixes J
  assumes $J \in I$
  assumes *finite* J
  shows $space\ (PiF\ \{J\}\ M) \in sets\ (PiF\ I\ M)$
  ⟨*proof*⟩

lemma *singleton-subspace-set-in-sets*:
  assumes A: $A \in sets\ (PiF\ \{J\}\ M)$
  assumes *finite* J
  assumes $J \in I$
  shows $A \in sets\ (PiF\ I\ M)$
  ⟨*proof*⟩

lemma
  finite-measurable-singletonI:
  assumes *finite* I
  assumes $\bigwedge J.\ J \in I \Longrightarrow finite\ J$
  assumes MN: $\bigwedge J.\ J \in I \Longrightarrow A \in measurable\ (PiF\ \{J\}\ M)\ N$
  shows $A \in measurable\ (PiF\ I\ M)\ N$
  ⟨*proof*⟩

lemma *space-subset-in-sets*:
  fixes J::$'a$::*countable set set*
  assumes $J \subseteq I$
  assumes $\bigwedge j.\ j \in J \Longrightarrow finite\ j$
  shows $space\ (PiF\ J\ M) \in sets\ (PiF\ I\ M)$
⟨*proof*⟩

lemma *subspace-set-in-sets*:
  fixes J::$'a$::*countable set set*
  assumes A: $A \in$ *sets* (*PiF* J M)
  assumes $J \subseteq I$
  assumes $\bigwedge j.\ j \in J \Longrightarrow$ *finite* j
  shows $A \in$ *sets* (*PiF* I M)
  ⟨*proof*⟩

lemma *finmap-eq-Un*:
  fixes X::($'a$::*countable* $\Rightarrow_F$ $'b$) *set*
  shows $X = (\bigcup n.\ X \cap \{x.$ *domain* $x =$ *set* (*from-nat* n)$\})$
⟨*proof*⟩

lemma
  countable-measurable-PiFI:
  fixes I::$'a$::*countable set set*
  assumes MN: $\bigwedge J.\ J \in I \Longrightarrow$ *finite* $J \Longrightarrow A \in$ *measurable* (*PiF* $\{J\}$ M) N
  shows $A \in$ *measurable* (*PiF* I M) N
  ⟨*proof*⟩

lemma *measurable-PiF*:
  assumes f: $\bigwedge x.\ x \in$ *space* $N \Longrightarrow$ *domain* ($f\,x$) $\in I \wedge (\forall\, i \in$*domain* ($f\,x$). ($f\,x$) i
$\in$ *space* (M i))
  assumes S: $\bigwedge J\ S.\ J \in I \Longrightarrow (\bigwedge i.\ i \in J \Longrightarrow S\ i \in$ *sets* (M i)) $\Longrightarrow$
    f $-$‘ (Pi' J S) $\cap$ *space* $N \in$ *sets* N
  shows $f \in$ *measurable* N (*PiF* I M)
  ⟨*proof*⟩

lemma
  restrict-sets-measurable:
  assumes A: $A \in$ *sets* (*PiF* I M) **and** $J \subseteq I$
  shows $A \cap \{m.$ *domain* $m \in J\} \in$ *sets* (*PiF* J M)
  ⟨*proof*⟩

lemma *measurable-finmap-of*:
  assumes f: $\bigwedge i.\ (\exists\, x \in$ *space* $N.\ i \in J\ x) \Longrightarrow (\lambda x.\ f\ x\ i) \in$ *measurable* N (M i)
  assumes J: $\bigwedge x.\ x \in$ *space* $N \Longrightarrow J\ x \in I$ $\bigwedge x.\ x \in$ *space* $N \Longrightarrow$ *finite* (J x)
  assumes JN: $\bigwedge S.\ \{x.\ J\ x = S\} \cap$ *space* $N \in$ *sets* N
  shows ($\lambda x.$ *finmap-of* (J x) ($f\,x$)) $\in$ *measurable* N (*PiF* I M)
⟨*proof*⟩

lemma *measurable-PiM-finmap-of*:
  assumes *finite* J
  shows *finmap-of* $J \in$ *measurable* (Pi_M J M) (*PiF* $\{J\}$ M)
  ⟨*proof*⟩

lemma *proj-measurable-singleton*:
  assumes $A \in$ *sets* (M i) *finite* I

shows $(\lambda x.\ (x)_F\ i)\ -`\ A \cap \mathit{space}\ (\mathit{PiF}\ \{I\}\ M) \in \mathit{sets}\ (\mathit{PiF}\ \{I\}\ M)$
⟨*proof*⟩

lemma *measurable-proj-singleton*:
fixes *I*
assumes *finite I* $i \in I$
shows $(\lambda x.\ (x)_F\ i) \in \mathit{measurable}\ (\mathit{PiF}\ \{I\}\ M)\ (M\ i)$
⟨*proof*⟩

lemma *measurable-proj-countable*:
fixes $I::'a::\mathit{countable\ set\ set}$
assumes $y \in \mathit{space}\ (M\ i)$
shows $(\lambda x.\ \mathit{if}\ i \in \mathit{domain}\ x\ \mathit{then}\ (x)_F\ i\ \mathit{else}\ y) \in \mathit{measurable}\ (\mathit{PiF}\ I\ M)\ (M\ i)$
⟨*proof*⟩

lemma *measurable-restrict-proj*:
assumes $J \in \mathit{II}$ *finite J*
shows *finmap-of J* $\in \mathit{measurable}\ (\mathit{PiM}\ J\ M)\ (\mathit{PiF}\ \mathit{II}\ M)$
⟨*proof*⟩

lemma
measurable-proj-PiM:
fixes $J\ K\ ::'a::\mathit{countable\ set}$ **and** $I::'a\ \mathit{set\ set}$
assumes *finite J* $J \in I$
assumes $x \in \mathit{space}\ (\mathit{PiM}\ J\ M)$
shows $\mathit{proj} \in$
$\mathit{measurable}\ (\mathit{PiF}\ \{J\}\ M)\ (\mathit{PiM}\ J\ M)$
⟨*proof*⟩

lemma *sets-subspaceI*:
assumes $A \cap \mathit{space}\ M \in \mathit{sets}\ M$
assumes $B \in \mathit{sets}\ M$
shows $A \cap B \in \mathit{sets}\ M$ ⟨*proof*⟩

lemma *space-PiF-singleton-eq-product*:
assumes *finite I*
shows $\mathit{space}\ (\mathit{PiF}\ \{I\}\ M) = (\Pi'\ i{\in}I.\ \mathit{space}\ (M\ i))$
⟨*proof*⟩

adapted from $\mathit{sets}\ (\mathit{Pi}_M\ ?I\ ?M) = \mathit{sigma\text{-}sets}\ (\Pi_E\ i{\in}?I.\ \mathit{space}\ (?M\ i))\ \{\{f \in \Pi_E\ i{\in}?I.\ \mathit{space}\ (?M\ i).\ f\ i \in A\}\ |i\ A.\ i \in\ ?I \wedge A \in \mathit{sets}\ (?M\ i)\}$

lemma *sets-PiF-single*:
assumes *finite I* $I \neq \{\}$
shows $\mathit{sets}\ (\mathit{PiF}\ \{I\}\ M) =$
$\mathit{sigma\text{-}sets}\ (\Pi'\ i{\in}I.\ \mathit{space}\ (M\ i))$
$\{\{f{\in}\Pi'\ i{\in}I.\ \mathit{space}\ (M\ i).\ f\ i \in A\}\ |\ i\ A.\ i \in I \wedge A \in \mathit{sets}\ (M\ i)\}$
(**is** - = *sigma-sets ?Ω ?R*)
⟨*proof*⟩

adapted from $(\bigwedge i.\ i \in ?I \Longrightarrow ?A\ i = ?B\ i) \Longrightarrow Pi_E\ ?I\ ?A = Pi_E\ ?I\ ?B$

lemma *Pi′-cong*:
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i = g\ i$
shows $Pi'\ I\ f = Pi'\ I\ g$
⟨*proof*⟩

adapted from $[\![finite\ ?I; \bigwedge i\ n\ m.\ [\![i \in ?I; n \le m]\!] \Longrightarrow ?A\ n\ i \subseteq ?A\ m\ i]\!] \Longrightarrow (\bigcup_n Pi\ ?I\ (?A\ n)) = (\Pi\ i{\in}?I.\ \bigcup_n ?A\ n\ i)$

lemma *Pi′-UN*:
fixes $A :: nat \Rightarrow 'i \Rightarrow 'a\ set$
assumes *finite I*
assumes *mono*: $\bigwedge i\ n\ m.\ i \in I \Longrightarrow n \le m \Longrightarrow A\ n\ i \subseteq A\ m\ i$
shows $(\bigcup n.\ Pi'\ I\ (A\ n)) = Pi'\ I\ (\lambda i.\ \bigcup n.\ A\ n\ i)$
⟨*proof*⟩

adapted from $[\![finite\ ?I; \bigwedge i.\ i \in ?I \Longrightarrow incseq\ (?S\ i); \bigwedge i.\ i \in ?I \Longrightarrow (\bigcup_j ?S\ i\ j) = space\ (?M\ i); \bigwedge i.\ i \in ?I \Longrightarrow range\ (?S\ i) \subseteq ?E\ i; \bigwedge i.\ i \in ?I \Longrightarrow ?E\ i \subseteq Pow\ (space\ (?M\ i)); \bigwedge i.\ i \in ?I \Longrightarrow sets\ (?M\ i) = sigma\text{-}sets\ (space\ (?M\ i))\ (?E\ i)]\!] \Longrightarrow sets\ (Pi_M\ ?I\ ?M) = sigma\text{-}sets\ (space\ (Pi_M\ ?I\ ?M))\ \{Pi_E\ ?I\ F\ |F.\ \forall i{\in}?I.\ F\ i \in ?E\ i\}$

lemma *sigma-fprod-algebra-sigma-eq*:
fixes $E :: 'i \Rightarrow 'a\ set\ set$
assumes [*simp*]: *finite I* $I \neq \{\}$
assumes *S-mono*: $\bigwedge i.\ i \in I \Longrightarrow incseq\ (S\ i)$
and *S-union*: $\bigwedge i.\ i \in I \Longrightarrow (\bigcup j.\ S\ i\ j) = space\ (M\ i)$
and *S-in-E*: $\bigwedge i.\ i \in I \Longrightarrow range\ (S\ i) \subseteq E\ i$
assumes *E-closed*: $\bigwedge i.\ i \in I \Longrightarrow E\ i \subseteq Pow\ (space\ (M\ i))$
and *E-generates*: $\bigwedge i.\ i \in I \Longrightarrow sets\ (M\ i) = sigma\text{-}sets\ (space\ (M\ i))\ (E\ i)$
defines $P == \{\ Pi'\ I\ F\ |\ F.\ \forall i{\in}I.\ F\ i \in E\ i\ \}$
shows $sets\ (PiF\ \{I\}\ M) = sigma\text{-}sets\ (space\ (PiF\ \{I\}\ M))\ P$
⟨*proof*⟩

lemma *enumerable-sigma-fprod-algebra-sigma-eq*:
assumes $I \neq \{\}$
assumes [*simp*]: *finite I*
shows $sets\ (PiF\ \{I\}\ (\lambda\text{-}.\ borel)) = sigma\text{-}sets\ (space\ (PiF\ \{I\}\ (\lambda\text{-}.\ borel)))$
$\{Pi'\ I\ F\ |F.\ (\forall i{\in}I.\ F\ i \in range\ enum\text{-}basis)\}$
⟨*proof*⟩

adapted from $[\![?I \neq \{\}; finite\ ?I]\!] \Longrightarrow sets\ (Pi_F\ \{?I\}\ (\lambda\text{-}.\ borel)) = sigma\text{-}sets\ (space\ (Pi_F\ \{?I\}\ (\lambda\text{-}.\ borel)))\ \{Pi'\ ?I\ F\ |F.\ \forall i{\in}?I.\ F\ i \in range\ enum\text{-}basis\}$

lemma *enumerable-sigma-prod-algebra-sigma-eq*:
assumes $I \neq \{\}$
assumes [*simp*]: *finite I*
shows $sets\ (PiM\ I\ (\lambda\text{-}.\ borel)) = sigma\text{-}sets\ (space\ (PiM\ I\ (\lambda\text{-}.\ borel)))$
$\{Pi_E\ I\ F\ |F.\ \forall i{\in}I.\ F\ i \in range\ enum\text{-}basis\}$

⟨proof⟩

lemma *product-open-generates-sets-PiF-single*:
  assumes $I \neq \{\}$
  assumes [*simp*]: *finite I*
  shows *sets* (*PiF* {*I*} (λ-. *borel*::′*b*::*enumerable-basis measure*)) =
    sigma-sets (*space* (*PiF* {*I*} (λ-. *borel*))) {*Pi′ I F* |*F*. (∀ *i*∈*I*. *F i* ∈ *Collect open*)}
⟨proof⟩

lemma *product-open-generates-sets-PiM*:
  assumes $I \neq \{\}$
  assumes [*simp*]: *finite I*
  shows *sets* (*PiM I* (λ-. *borel*::′*b*::*enumerable-basis measure*)) =
    sigma-sets (*space* (*PiM I* (λ-. *borel*))) {Pi_E *I F* |*F*. ∀ *i*∈*I*. *F i* ∈ *Collect open*}
⟨proof⟩

lemma *finmap-UNIV*[*simp*]: (⋃ *J*∈*Collect finite*. *J* ⇝ *UNIV*) = *UNIV* ⟨proof⟩

lemma *borel-eq-PiF-borel*:
  shows (*borel* :: (′*i*::*countable* $\Rightarrow_F$ ′*a*::*polish-space*) *measure*) =
  PiF (*Collect finite*) (λ-. *borel* :: ′*a measure*)
⟨proof⟩

3.9 Measure preservation

Measure preservation is not used at the moment.

definition *measure-preserving f A B* ⟷ *f* ∈ *measurable A B* ∧ (∀ *x* ∈ *sets B*. *distr A B f x* = *B x*)

lemma
  assumes *measure-preserving f A B*
  shows *measure-preserving-distr*: ⋀*x*. *x* ∈ *sets B* ⟹ *distr A B f x* = *B x*
  and *measure-preserving-measurable*: *f* ∈ *measurable A B*
  ⟨proof⟩

lemma *measure-preservingI*:
  assumes *f* ∈ *measurable A B* ⋀*x*. *x* ∈ *sets B* ⟹ *distr A B f x* = *B x*
  shows *measure-preserving f A B*
  ⟨proof⟩

lemma *measure-preservingI′*[*intro*]:
  assumes *AB*: *f* ∈ *measurable A B*
  assumes *m*: ⋀*x*. *x* ∈ *sets B* ⟹ *emeasure A* (*f* −‘ *x* ∩ *space A*) = *emeasure B x*
  shows *measure-preserving f A B*
  ⟨proof⟩

lemma

measure-preserving-comp:
assumes *AB*: *measure-preserving f A B*
assumes *BC*: *measure-preserving g B C*
shows *measure-preserving* (*g o f*) *A C*
⟨*proof*⟩

3.10 Isomorphism between Functions and Finite Maps

lemma
measurable-compose:
fixes $f::'a \Rightarrow 'b$
assumes *inj*: $\bigwedge j.\ j \in J \Longrightarrow f'\ (f\ j) = j$
assumes *finite J*
shows ($\lambda m.$ *compose J m f*) $\in$ *measurable* (*PiM* (*f* ‘ *J*) (λ-. *M*)) (*PiM J* (λ-. *M*))
⟨*proof*⟩

lemma
measurable-compose-inv:
fixes $f::'a \Rightarrow 'b$
assumes *inj*: $\bigwedge j.\ j \in J \Longrightarrow f'\ (f\ j) = j$
assumes *finite J*
shows ($\lambda m.$ *compose* (*f* ‘ *J*) *m f′*) $\in$ *measurable* (*PiM J* (λ-. *M*)) (*PiM* (*f* ‘ *J*) (λ-. *M*))
⟨*proof*⟩

locale *function-to-finmap* =
fixes $J::'a$ *set* **and** $f :: 'a \Rightarrow 'b::countable$ **and** f'
assumes [*simp*]: *finite J*
assumes *inv*: $i \in J \Longrightarrow f'\ (f\ i) = i$
begin

to measure finmaps

definition *fm* = (*finmap-of* (*f* ‘ *J*)) *o* ($\lambda g.$ *compose* (*f* ‘ *J*) *g f′*)

lemma *domain-fm*[*simp*]: *domain* (*fm x*) = *f* ‘ *J*
⟨*proof*⟩

lemma *fm-restrict*[*simp*]: *fm* (*restrict y J*) = *fm y*
⟨*proof*⟩

lemma *fm-product*:
assumes $\bigwedge i.$ *space* (*M i*) = *UNIV*
shows *fm* −‘ *Pi′* (*f* ‘ *J*) *S* ∩ *space* (Pi_M *J M*) = (Π_E *j* ∈ *J*. *S* (*f j*))
⟨*proof*⟩

lemma *fm-measurable*:
assumes *f* ‘ *J* ∈ *N*
shows *fm* ∈ *measurable* (Pi_M *J* (λ-. *M*)) (Pi_F *N* (λ-. *M*))

⟨*proof*⟩

lemma *proj-fm*:
assumes $x \in J$
shows *fm* m $(f\ x) = m\ x$
⟨*proof*⟩

lemma *inj-on-compose-f'*: *inj-on* $(\lambda g.\ compose\ (f\ `\ J)\ g\ f')$ $(extensional\ J)$
⟨*proof*⟩

lemma *inj-on-fm*:
assumes $\bigwedge i.\ space\ (M\ i) = UNIV$
shows *inj-on fm* $(space\ (Pi_M\ J\ M))$
⟨*proof*⟩

lemma *fm-vimage-image-eq*:
assumes $\bigwedge i.\ space\ (M\ i) = UNIV$
assumes $X \in sets\ (Pi_M\ J\ M)$
shows $fm\ -`\ fm\ `\ X \cap space\ (Pi_M\ J\ M) = X$
⟨*proof*⟩

to measure functions

definition $mf = (\lambda g.\ compose\ J\ g\ f)\ o\ proj$

lemma
assumes $x \in space\ (Pi_M\ J\ (\lambda\text{-}.\ M))$ *finite* J
shows *proj* (*finmap-of* $J\ x) = x$
⟨*proof*⟩

lemma
assumes $x \in space\ (Pi_F\ \{J\}\ (\lambda\text{-}.\ M))$
shows *finmap-of* J $(proj\ x) = x$
⟨*proof*⟩

lemma *mf-fm*:
assumes $x \in space\ (Pi_M\ J\ (\lambda\text{-}.\ M))$
shows $mf\ (fm\ x) = x$
⟨*proof*⟩

lemma *mf-measurable*:
assumes $space\ M = UNIV$
shows $mf \in measurable\ (PiF\ \{f\ `\ J\}\ (\lambda\text{-}.\ M))\ (PiM\ J\ (\lambda\text{-}.\ M))$
⟨*proof*⟩

lemma *fm-image-measurable*:
assumes $space\ M = UNIV$
assumes $X \in sets\ (Pi_M\ J\ (\lambda\text{-}.\ M))$
shows $fm\ `\ X \in sets\ (PiF\ \{f\ `\ J\}\ (\lambda\text{-}.\ M))$
⟨*proof*⟩

lemma *fm-image-measurable-finite*:
assumes *space M* = *UNIV*
assumes *X* ∈ *sets* (*Pi$_M$* *J* (λ-. *M*::′*c measure*))
shows *fm* ‘ *X* ∈ *sets* (*PiF* (*Collect finite*) (λ-. *M*::′*c measure*))
⟨*proof*⟩

measure on finmaps

definition *mapmeasure M N* = *distr M* (*PiF* (*Collect finite*) *N*) (*fm*)

lemma *sets-mapmeasure*[*simp*]: *sets* (*mapmeasure M N*) = *sets* (*PiF* (*Collect finite*) *N*)
⟨*proof*⟩

lemma *space-mapmeasure*[*simp*]: *space* (*mapmeasure M N*) = *space* (*PiF* (*Collect finite*) *N*)
⟨*proof*⟩

lemma *mapmeasure-PiF*:
assumes *s1*: *space M* = *space* (*Pi$_M$* *J* (λ-. *N*))
assumes *s2*: *sets M* = (*Pi$_M$* *J* (λ-. *N*))
assumes *space N* = *UNIV*
assumes *X* ∈ *sets* (*PiF* (*Collect finite*) (λ-. *N*))
shows *emeasure* (*mapmeasure M* (λ-. *N*)) *X* = *emeasure M* ((*fm* −‘ *X* ∩ *extensional J*))
⟨*proof*⟩

lemma *mapmeasure-PiM*:
fixes *N*::′*c measure*
assumes *s1*: *space M* = *space* (*Pi$_M$* *J* (λ-. *N*))
assumes *s2*: *sets M* = (*Pi$_M$* *J* (λ-. *N*))
assumes *N*: *space N* = *UNIV*
assumes *X*: *X* ∈ *sets M*
shows *emeasure M X* = *emeasure* (*mapmeasure M* (λ-. *N*)) (*fm* ‘ *X*)
⟨*proof*⟩

end

end

theory *Projective-Limit*
imports *Probability Polish-Space Fin-Map*
begin

4 Projective Limit

Formalization of the Daniell-Kolmogorov theorem.

4.1 (Finite) Product of Measures

TODO: unifiy with Pi_M

definition
$PiP\ I\ M\ P = extend\text{-}measure$
$(\Pi_E\ i{\in}I.\ space\ (M\ i))$
$\{x.\ (domain\ x \neq \{\} \vee I = \{\}) \wedge$
$finite\ (domain\ x) \wedge domain\ x \subseteq I \wedge (x)_F \in (\Pi_E\ i{\in}(domain\ x).\ sets\ (M\ i))\}$
$(\lambda x.\ prod\text{-}emb\ I\ M\ (domain\ x)\ (Pi_E\ (domain\ x)\ (x)_F))$
$(\lambda x.\ emeasure\ (P\ (domain\ x))\ (Pi_E\ (domain\ x)\ (x)_F))$

definition *proj-algebra* **where**
$proj\text{-}algebra\ I\ M = (\lambda x.\ prod\text{-}emb\ I\ M\ (domain\ x)\ (Pi_E\ (domain\ x)\ (x)_F))$ ‘
$\{x.\ (domain\ x \neq \{\} \vee I = \{\}) \wedge$
$finite\ (domain\ x) \wedge domain\ x \subseteq I \wedge (x)_F \in (\Pi_E\ i{\in}domain\ x.\ sets\ (M\ i))\}$

lemma *proj-algebra-eq-prod-algebra*:
$proj\text{-}algebra\ I\ M = prod\text{-}algebra\ I\ M$
⟨*proof*⟩

lemma
shows *proj-algebra-eq*:
$proj\text{-}algebra\ I\ M = \{prod\text{-}emb\ I\ M\ J\ (Pi_E\ J\ F)\ |J\ F.$
$(J \neq \{\} \vee I = \{\}) \wedge finite\ J \wedge J \subseteq I \wedge (\forall\, i \in J.\ F\ i \in sets\ (M\ i))\}$
⟨*proof*⟩

lemma *proj-algebra-eq'*:
assumes $I \neq \{\}$
shows $proj\text{-}algebra\ I\ M =$
$\{prod\text{-}emb\ I\ M\ J\ (Pi_E\ J\ F)\ |J\ F.\ J \neq \{\} \wedge finite\ J \wedge J \subseteq I \wedge (\forall\, i \in J.\ F\ i$
$\in sets\ (M\ i))\}$
⟨*proof*⟩

lemma *space-PiP*[*simp*]: $space\ (PiP\ I\ M\ P) = space\ (PiM\ I\ M)$
⟨*proof*⟩

lemma *sets-PiP'*: $sets\ (PiP\ I\ M\ P) = sigma\text{-}sets\ (\Pi_E\ i{\in}I.\ space\ (M\ i))\ (proj\text{-}algebra$
$I\ M)$
⟨*proof*⟩

lemma *sets-PiP*[*simp*]: $sets\ (PiP\ I\ M\ P) = sets\ (PiM\ I\ M)$
⟨*proof*⟩

lemma *measurable-PiP1*[*simp*]: $measurable\ (PiP\ I\ M\ P)\ M' = measurable\ (\Pi_M$

$i{\in}I.\ M\ i)\ M'$
⟨*proof*⟩

lemma *measurable-PiP2*[*simp*]: *measurable* M' (*PiP I M P*) = *measurable* M' (Π_M $i{\in}I.\ M\ i$)
⟨*proof*⟩

4.2 Projective Family

locale *projective-family* =
fixes I::'*i set* **and** P::'*i set* ⇒ ('*i* ⇒ '*a*) *measure* **and** M::('*i* ⇒ '*a measure*)
assumes *projective*: $\bigwedge J\ H.\ J \subseteq H \Longrightarrow H \subseteq I \Longrightarrow$ *finite* $H \Longrightarrow$
($P\ H$) (*prod-emb H M J X*) = ($P\ J$) X
assumes *prob-space*: $\bigwedge J.$ *prob-space* ($P\ J$)
assumes *proj-sets*: $\bigwedge J.$ *sets* ($P\ J$) = *sets* (*PiM J M*)
assumes *proj-space*: $\bigwedge J.$ *space* ($P\ J$) = *space* (*PiM J M*)
assumes *measure-space*: $\bigwedge i.$ *prob-space* ($M\ i$)
— TODO: generalize definitions from *product-prob-space* to *product-measure-space*

begin

lemma *measurable-P1*[*simp*]: *measurable* ($P\ J$) M' = *measurable* (Π_M $i{\in}J.\ M\ i$) M'
⟨*proof*⟩

lemma *measurable-P2*[*simp*]: *measurable* M' ($P\ J$) = *measurable* M' (Π_M $i{\in}J.$ $M\ i$)
⟨*proof*⟩

end

sublocale *projective-family* ⊆ M: *prob-space M i* **for** i ⟨*proof*⟩

sublocale *projective-family* ⊆ *prob-space*: *prob-space P J* **for** J ⟨*proof*⟩

sublocale *projective-family* ⊆ *MP*: *product-prob-space M* ⟨*proof*⟩

context *projective-family* **begin**

lemma *emeasure-PiP*:
assumes *finite J*
assumes $J \subseteq I$
assumes A: $\bigwedge i.\ i{\in}J \Longrightarrow A\ i \in$ *sets* ($M\ i$)
shows *emeasure* (*PiP J M P*) (Pi_E $J\ A$) = *emeasure* ($P\ J$) (Pi_E $J\ A$)
⟨*proof*⟩

lemma *PiP-finite*:
assumes *finite J*
assumes $J \subseteq I$

shows *PiP J M P* = *P J* (**is** *?P* = -)
⟨*proof*⟩

lemma *emeasure-fun-emb*[*simp*]:
assumes *L*: $J \subseteq L$ *finite L* $L \subseteq I$ **and** *X*: $X \in$ *sets* (*PiP J M P*)
shows *emeasure* (*PiP L M P*) (*emb L J X*) = *emeasure* (*PiP J M P*) *X*
⟨*proof*⟩

lemma *distr-restrict*:
assumes $J \subseteq K$ *finite K* $K \subseteq I$
shows (*PiP J M P*) = *distr* (*PiP K M P*) (*PiP J M P*) (λf. *restrict f J*) (**is** *?P* = *?D*)
⟨*proof*⟩

4.3 Content on Generator

definition
$\mu G'$ *A* =
(*THE x*. $\forall$ *J*. $J \neq \{\} \longrightarrow$ *finite J* $\longrightarrow J \subseteq I \longrightarrow$
($\forall X \in$*sets* (*PiP J M P*). *A* = *emb I J X* $\longrightarrow$ *x* = *emeasure* (*PiP J M P*) *X*))

lemma $\mu G'$*-spec*:
assumes *J*: $J \neq \{\}$ *finite J* $J \subseteq I$ *A* = *emb I J X* $X \in$ *sets* (*PiP J M P*)
shows $\mu G'$ *A* = *emeasure* (*PiP J M P*) *X*
⟨*proof*⟩

lemma $\mu G'$*-eq*:
$J \neq \{\} \Longrightarrow$ *finite J* $\Longrightarrow J \subseteq I \Longrightarrow X \in$ *sets* (*PiP J M P*) $\Longrightarrow$
$\mu G'$ (*emb I J X*) = *emeasure* (*PiP J M P*) *X*
⟨*proof*⟩

lemma *generator-Ex'*:
assumes $*$: $A \in$ *generator*
shows $\exists$ *J X*. $J \neq \{\} \wedge$ *finite J* $\wedge J \subseteq I \wedge X \in$ *sets* (Π_M $i \in J$. *M i*) $\wedge$ *A* = *emb I J X* $\wedge$
$\mu G'$ *A* = *emeasure* (*PiP J M P*) *X*
⟨*proof*⟩

lemma *generatorE'*:
assumes *A*: $A \in$ *generator*
obtains *J X* **where** $J \neq \{\}$ *finite J* $J \subseteq I$ $X \in$ *sets* (*PiP J M P*) *emb I J X* = *A*
$\mu G'$ *A* = *emeasure* (*PiP J M P*) *X*
⟨*proof*⟩

lemma *positive-*$\mu G'$:
assumes $I \neq \{\}$
shows *positive generator* $\mu G'$
⟨*proof*⟩

lemma *additive-μG'*:
assumes $I \neq \{\}$
shows *additive generator* $\mu G'$
⟨*proof*⟩

end

4.4 Sequences of Finite Maps in Compact Sets

locale *finmap-seqs-into-compact* =
fixes K::*nat* ⇒ (*nat* $\Rightarrow_F$ *'a*::*metric-space*) *set* **and** f::*nat* ⇒ (*nat* $\Rightarrow_F$ *'a*) **and** M
assumes *compact*: $\bigwedge n.$ *compact* (K n)
assumes *f-in-K*: $\bigwedge n.$ K $n \neq \{\}$
assumes *domain-K*: $\bigwedge n.$ $k \in K$ $n \Longrightarrow$ *domain* k = *domain* (f n)
assumes *proj-in-K*:
$\bigwedge t$ n m. $m \geq n \Longrightarrow t \in$ *domain* (f n) $\Longrightarrow$ (f m)$_F$ $t \in$ ($\lambda k.$ (k)$_F$ t) ‘ K n
begin

lemma *proj-in-K'*: ($\exists n. \forall m \geq n.$ (f m)$_F$ $t \in$ ($\lambda k.$ (k)$_F$ t) ‘ K n)
⟨*proof*⟩

lemma *proj-in-KE*:
obtains n **where** $\bigwedge m.$ $m \geq n \Longrightarrow$ (f m)$_F$ $t \in$ ($\lambda k.$ (k)$_F$ t) ‘ K n
⟨*proof*⟩

lemma *compact-projset*:
shows *compact* (($\lambda k.$ (k)$_F$ i) ‘ K n)
⟨*proof*⟩

end

sublocale *finmap-seqs-into-compact* ⊆ *subseqs* λn s r. ($\exists l.$ ($\lambda i.$ ((f *o* s *o* r) i)$_F$ n) −−−−> l)
⟨*proof*⟩

lemma (**in** *finmap-seqs-into-compact*)
diagonal-tendsto: $\exists l.$ ($\lambda i.$ (f (*diagseq* i))$_F$ n) −−−−> l
⟨*proof*⟩

4.5 The Daniell-Kolmogorov theorem

locale *polish-projective* = *projective-family* I P λ-. *borel*::*'a*::*polish-space measure*
for I::*'i set* **and** P
begin

abbreviation *PiB* ≡ (λJ P. *PiP* J (λ-. *borel*) P)

lemma

emeasure-PiB-emb-not-empty:
assumes $I \neq \{\}$
assumes X: $J \neq \{\}$ $J \subseteq I$ *finite* J $\forall i \in J.$ B $i \in$ *sets borel*
shows *emeasure* (*PiB I P*) (*emb I J* (Pi_E J B)) = *emeasure* (*PiB J P*) (Pi_E J B)
⟨*proof*⟩

end

sublocale *polish-projective* $\subseteq$ P: *prob-space* (*PiB I P*)
⟨*proof*⟩

context *polish-projective* **begin**

lemma *emeasure-PiB-emb*:
assumes X: $J \subseteq I$ *finite* J $\forall i \in J.$ B $i \in$ *sets borel*
shows *emeasure* (*PiB I P*) (*emb I J* (Pi_E J B)) = *emeasure* (P J) (Pi_E J B)
⟨*proof*⟩

lemma *measure-PiB-emb*:
assumes $J \subseteq I$ *finite* J $\forall i \in J.$ B $i \in$ *sets borel*
shows *measure* (*PiB I P*) (*emb I J* (Pi_E J B)) = *measure* (P J) (Pi_E J B)
⟨*proof*⟩

end

end

References

[1] F. Immler. Generic construction of probability spaces for paths of stochastic processes in Isabelle/HOL. Master's thesis, Technische Universität München, October 2012. Submitted.