

The Kolmogorov-Chentsov Theorem

Christian Pardillo Laursen and Simon Foster
University of York, UK

`{christian.laursen,simon.foster}@york.ac.uk`

October 13, 2025

Abstract

Continuous-time stochastic processes often carry the condition of having almost-surely continuous paths. If some process X satisfies certain bounds on its expectation, then the Kolmogorov-Chentsov theorem lets us construct a modification of X , i.e. a process X' such that $\forall t. X_t = X'_t$ almost surely, that has Hölder continuous paths.

In this work, we mechanise the Kolmogorov-Chentsov theorem. To get there, we develop a theory of stochastic processes, together with Hölder continuity, convergence in measure, and arbitrary intervals of dyadic rationals.

With this, we pave the way towards a construction of Brownian motion. The work is based on the exposition in Achim Klenke's probability theory text [1].

Contents

1	Supporting lemmas	2
2	Intervals of dyadic rationals	6
3	Hölder continuity	22
4	Convergence in measure	30
5	Stochastic processes	40
6	The Kolmogorov-Chentsov theorem	55
6.1	Supporting lemmas	55
6.2	Kolmogorov-Chentsov	56

1 Supporting lemmas

theory *Kolmogorov-Chentsov-Extras*
imports *HOL-Probability.Probability*
begin

lemma *atLeastAtMost-induct*[*consumes 1, case-names base Suc*]:
assumes $x \in \{n..m\}$
and $P\ n$
and $\bigwedge k. \llbracket k \geq n; k < m; P\ k \rrbracket \implies P\ (Suc\ k)$
shows $P\ x$
by (*smt (verit, ccfv-threshold) assms Suc-le-eq atLeastAtMost-iff dec-induct le-trans*)

lemma *eventually-prodI'*:
assumes *eventually* $P\ F$ *eventually* $Q\ G\ \forall x\ y. P\ x \longrightarrow Q\ y \longrightarrow R\ (x,y)$
shows *eventually* $R\ (F \times_F G)$
using *assms eventually-prod-filter* **by** *blast*

Analogous to $\llbracket \text{almost-everywhere } ?M\ ?P; \bigwedge N. \llbracket \bigwedge x. x \in \text{space } ?M - N \implies ?P\ x; N \in \text{null-sets } ?M \rrbracket \implies ?thesis \rrbracket \implies ?thesis$

lemma *AE-I3*:
assumes $\bigwedge x. x \in \text{space } M - N \implies P\ x\ N \in \text{null-sets } M$
shows *AE* x *in* $M. P\ x$
by (*metis (no-types, lifting) assms DiffI eventually-ae-filter mem-Collect-eq subsetI*)

Extends $\llbracket (?f \longrightarrow ?l)\ ?F; (?g \longrightarrow ?m)\ ?F \rrbracket \implies ((\lambda x. \text{dist } (?f\ x)\ (?g\ x)) \longrightarrow \text{dist } ?l\ ?m)\ ?F$

lemma *tendsto-dist-prod*:
fixes $l\ m :: 'a :: \text{metric-space}$
assumes $f: (f \longrightarrow l)\ F$
and $g: (g \longrightarrow m)\ G$
shows $((\lambda x. \text{dist } (f\ (\text{fst } x))\ (g\ (\text{snd } x))) \longrightarrow \text{dist } l\ m)\ (F \times_F G)$
proof (*rule tendstoI*)
fix $e :: \text{real}$ **assume** $0 < e$
then have $e2: 0 < e/2$ **by** *simp*
show $\forall_F\ x\ \text{in } F \times_F G. \text{dist } (\text{dist } (f\ (\text{fst } x))\ (g\ (\text{snd } x)))\ (\text{dist } l\ m) < e$
using *tendstoD [OF f e2] tendstoD [OF g e2]* **apply** (*rule eventually-prodI'*)
apply *simp*
by (*smt (verit) dist-commute dist-norm dist-triangle real-norm-def*)
qed

lemma *borel-measurable-at-within-metric* [*measurable*]:
fixes $f :: 'a :: \text{first-countable-topology} \Rightarrow 'b \Rightarrow 'c :: \text{metric-space}$
assumes [*measurable*]: $\bigwedge t. t \in S \implies f\ t \in \text{borel-measurable } M$
and *convergent*: $\bigwedge \omega. \omega \in \text{space } M \implies \exists l. ((\lambda t. f\ t\ \omega) \longrightarrow l)\ (\text{at } x\ \text{within } S)$
and *nontrivial*: $\text{at } x\ \text{within } S \neq \perp$
shows $(\lambda \omega. \text{Lim } (\text{at } x\ \text{within } S)\ (\lambda t. f\ t\ \omega)) \in \text{borel-measurable } M$
proof –

obtain l **where** $l: \bigwedge \omega. \omega \in \text{space } M \implies ((\lambda t. f \ t \ \omega) \longrightarrow l \ \omega) \text{ (at } x \text{ within } S)$
using *convergent* **by** *metis*
then have *Lim-eq*: $\text{Lim (at } x \text{ within } S) (\lambda t. f \ t \ \omega) = l \ \omega$
if $\omega \in \text{space } M$ **for** ω
using *tendsto-Lim[OF nontrivial]* **that** **by** *blast*
from *nontrivial* **have** $1: x \text{ islimpt } S$
using *trivial-limit-within* **by** *blast*
then obtain $s :: \text{nat} \Rightarrow 'a$ **where** $s: \bigwedge n. s \ n \in S - \{x\} \ s \longrightarrow x$
using *islimpt-sequential* **by** *blast*
then have $\bigwedge n. f \ (s \ n) \in \text{borel-measurable } M$
using s **by** *simp*
moreover have $(\lambda n. (f \ (s \ n) \ \omega)) \longrightarrow l \ \omega$ **if** $\omega \in \text{space } M$ **for** ω
using s [*unfolded tendsto-at-iff-sequentially comp-def, OF that*]
by *blast*
ultimately have $l \in \text{borel-measurable } M$
by (*rule borel-measurable-LIMSEQ-metric*)
then show *?thesis*
using *measurable-cong[OF Lim-eq]* **by** *fast*
qed

lemma *Max-finite-image-ex*:
assumes *finite* $S \ S \neq \{\}$ $P \ (\text{MAX } k \in S. f \ k)$
shows $\exists k \in S. P \ (f \ k)$
using *bexI[of P Max (f ' S) f ' S]* **by** (*simp add: assms*)

lemma *measure-leq-emeasure-ennreal*: $0 \leq x \implies \text{emeasure } M \ A \leq \text{ennreal } x \implies$
 $\text{measure } M \ A \leq x$
apply (*cases* $A \in M$)
apply (*metis Sigma-Algebra.measure-def enn2real-leI*)
apply (*auto simp: measure-notin-sets emeasure-notin-sets*)
done

lemma *Union-add-subset*: $(m :: \text{nat}) \leq n \implies (\bigcup k. A \ (k + n)) \subseteq (\bigcup k. A \ (k + m))$
apply (*subst subset-eq*)
apply *simp*
apply (*metis add.commute le-iff-add trans-le-add1*)
done

lemma *floor-in-Nats [simp]*: $x \geq 0 \implies \lfloor x \rfloor \in \mathbb{N}$
by (*metis nat-0-le of-nat-in-Nats zero-le-floor*)

lemma *triangle-ineq-list*:
fixes $l :: ('a :: \text{metric-space}) \text{ list}$
assumes $l \neq []$
shows $\text{dist (hd } l) \ (\text{last } l) \leq (\sum i=1.. \text{length } l - 1. \text{dist } (l[i]) \ (l[(i-1)]))$
using *assms* **proof** (*induct l rule: rev-nonempty-induct*)
case (*single* x)
then show *?case* **by** *force*

```

next
  case (snoc x xs)
  define S :: 'a list  $\Rightarrow$  real where
    S  $\equiv \lambda l. (\sum i=1..length\ l - 1. dist\ (!i)\ (!i-1))$ 
  have S (xs @ [x]) = dist x (last xs) + S xs
  unfolding S-def apply simp
  apply (subst sum.last-plus)
  apply (simp add: Suc-leI snoc.hyps(1))
  apply (rule arg-cong2[where f=(+)])
  apply (simp add: last-conv-nth nth-append snoc.hyps(1))
  apply (rule sum.cong)
  apply fastforce
  by (smt (verit) Suc-pred atLeastAtMost-iff diff-is-0-eq less-Suc-eq nat-less-le
not-less nth-append)
  moreover have dist (hd xs) (last xs)  $\leq$  S xs
  unfolding S-def using snoc by blast
  ultimately have dist (hd xs) x  $\leq$  S (xs@[x])
  by (smt (verit) dist-triangle2)
  then show ?case
  unfolding S-def using snoc by simp
qed

lemma triangle-ineq-sum:
  fixes f :: nat  $\Rightarrow$  'a :: metric-space
  assumes n  $\leq$  m
  shows dist (f n) (f m)  $\leq (\sum i=Suc\ n..m. dist\ (f\ i)\ (f\ (i-1)))$ 
proof (cases n=m)
  case True
  then show ?thesis by simp
next
  case False
  then have n < m
  using assms by simp
  define l where l  $\equiv map\ (f\ o\ nat)\ [n..m]$ 
  have [simp]: l  $\neq []$ 
  using <n < m> l-def by fastforce
  have [simp]: length l = m - n + 1
  unfolding l-def apply simp
  using assms by linarith
  with l-def have hd l = f n
  by (simp add: assms upto.simps)
  moreover have last l = f m
  unfolding l-def apply (subst last-map)
  using assms apply force
  using <n < m> upto-rec2 by force
  ultimately have dist (f n) (f m) = dist (hd l) (last l)
  by simp
  also have ...  $\leq (\sum i=1..length\ l - 1. dist\ (!i)\ (!i-1))$ 
  by (rule triangle-ineq-list[OF <l  $\neq []$ >])

```

```

also have ... = (∑ i=Suc n..m. dist (f i) (f (i-1)))
  apply (rule sum.reindex-cong[symmetric, where l=(+) n])
  using inj-on-add apply blast
  apply (simp add: assms)
  apply (rule arg-cong2[where f=dist])
  apply simp-all
  unfolding l-def apply (subst nth-map, fastforce)
  apply (subst nth-upto, linarith)
  subgoal by simp (insert nat-int-add, presburger)
  apply (subst nth-map, fastforce)
  apply (subst nth-upto, linarith)
  by (simp add: add-diff-eq nat-int-add nat-diff-distrib')
finally show ?thesis
  by blast
qed

lemma (in product-prob-space) indep-vars-PiM-coordinate:
  assumes I ≠ {}
  shows prob-space.indep-vars (ΠM i∈I. M i) M (λx f. f x) I
proof -
  have distr (PiM I M) (PiM I M) (λx. restrict x I) = distr (PiM I M) (PiM I
M) (λx. x)
  by (rule distr-cong; simp add: space-PiM)
  also have ... = PiM I M
  by simp
  also have ... = PiM I (λi. distr (PiM I M) (M i) (λf. f i))
  using PiM-component by (intro PiM-cong, auto)
  finally have distr (PiM I M) (PiM I M) (λx. restrict x I) = PiM I (λi. distr
(PiM I M) (M i) (λf. f i))
  .
  then show ?thesis
  using assms by (simp add: indep-vars-iff-distr-eq-PiM')
qed

lemma (in prob-space) indep-sets-indep-set:
  assumes indep-sets F I i ∈ I j ∈ I i ≠ j
  shows indep-set (F i) (F j)
  unfolding indep-set-def
proof (rule indep-setsI)
  show (case x of True ⇒ F i | False ⇒ F j) ⊆ events for x
  using assms by (auto split: bool.split simp: indep-sets-def)
  fix A J assume *: J ≠ {} J ⊆ UNIV ∀ja∈J. A ja ∈ (case ja of True ⇒ F i |
False ⇒ F j)
  {
    assume J = UNIV
    then have indep-sets F I {i,j} ⊆ I {i, j} ≠ {} finite {i,j} ∀x ∈ {i,j}. (λx. if
x = i then A True else A False) x ∈ F x
    using * assms apply simp-all
    by (simp add: bool.split-sel)
  }

```

```

    then have prob ( $\bigcap j \in \{i, j\}. \text{if } j = i \text{ then } A \text{ True else } A \text{ False}$ ) = ( $\prod j \in \{i, j\}. \text{prob (if } j = i \text{ then } A \text{ True else } A \text{ False})$ )
    by (rule indep-setsD)
    then have prob ( $A \text{ True} \cap A \text{ False}$ ) = prob ( $A \text{ True}$ ) * prob ( $A \text{ False}$ )
    using assms by (auto simp: ac-simps)
  } note X = this
  consider  $J = \{\text{True}, \text{False}\} \mid J = \{\text{False}\} \mid J = \{\text{True}\}$ 
  using  $*(1,2)$  unfolding UNIV-bool by blast
  then show prob ( $\bigcap (A \text{ ' } J)$ ) = ( $\prod j \in J. \text{prob (A } j)$ )
  using X by (cases; auto)
qed

```

```

lemma (in prob-space) indep-vars-indep-var:
  assumes indep-vars  $M' X I i \in I j \in I i \neq j$ 
  shows indep-var ( $M' i$ ) ( $X i$ ) ( $M' j$ ) ( $X j$ )
  using assms unfolding indep-vars-def indep-var-eq
  by (meson indep-sets-indep-set)

```

end

2 Intervals of dyadic rationals

```

theory Dyadic-Interval
  imports HOL-Analysis.Analysis
begin

```

In this file we describe intervals of dyadic numbers $S..T$ for reals $S T$. We use the floor and ceiling functions to approximate the numbers with increasing accuracy.

```

lemma frac-floor:  $\lfloor x \rfloor = x - \text{frac } x$ 
  by (simp add: frac-def)

```

```

lemma frac-ceil:  $\lceil x \rceil = x + \text{frac } (-x)$ 
  apply (cases  $x = \text{real-of-int } \lfloor x \rfloor$ )
  unfolding ceiling-altdef apply simp
  apply (metis Ints-minus Ints-of-int)
  apply (simp add: frac-neg frac-floor)
  done

```

```

lemma floor-pow2-lim:  $(\lambda n. \lfloor 2^n * T \rfloor / 2^n) \longrightarrow T$ 
proof (intro LIMSEQ-I)
  fix  $r :: \text{real}$  assume  $r > 0$ 
  obtain  $k$  where  $k: 1 / 2^k < r$ 
  by (metis  $\langle r > 0 \rangle$  one-less-numeral-iff power-one-over reals-power-lt-ex semiring-norm(76))
  then have  $\forall n \geq k. \text{norm } (\lfloor 2^n * T \rfloor / 2^n - T) < r$ 
  apply (simp add: frac-floor field-simps)
  by (smt (verit, ccfv-SIG)  $\langle 0 < r \rangle$  frac-lt-1 mult-left-mono power-increasing)

```

then show $\exists no. \forall n \geq no. \text{norm } (\text{real-of-int } \lfloor 2^n * T \rfloor / 2^n - T) < r$
by *blast*
qed

lemma *floor-pow2-leq*: $\lfloor 2^n * T \rfloor / 2^n \leq T$
by (*simp add: frac-floor field-simps*)

lemma *ceil-pow2-lim*: $(\lambda n. \lceil 2^n * T \rceil / 2^n) \longrightarrow T$

proof (*intro LIMSEQ-I*)

fix $r :: \text{real}$ **assume** $r > 0$

obtain k **where** $k: 1 / 2^k < r$

by (*metis* $\langle r > 0 \rangle$ *one-less-numeral-iff power-one-over reals-power-lt-ex semiring-norm*(76))

then have $\forall n \geq k. \text{norm } (\lceil 2^n * T \rceil / 2^n - T) < r$

apply (*simp add: frac-ceil field-simps*)

by (*smt* (*verit*) $\langle 0 < r \rangle$ *frac-lt-1 mult-left-mono power-increasing*)

then show $\exists no. \forall n \geq no. \text{norm } (\lceil 2^n * T \rceil / 2^n - T) < r$

by *blast*

qed

lemma *ceil-pow2-geq*: $\lceil 2^n * T \rceil / 2^n \geq T$

by (*simp add: frac-ceil field-simps*)

dyadic_interval_step n S T is the collection of dyadic numbers in $\{S..T\}$ with denominator 2^n . As $n \rightarrow \infty$ this collection approximates $\{S..T\}$. Compare with *dyadics* $\equiv \bigcup_{k \ m} \{ \text{of-nat } m / (2::?'a)^k \}$

definition *dyadic-interval-step* $:: \text{nat} \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow \text{real set}$

where *dyadic-interval-step* n S $T \equiv (\lambda k. k / (2^n)) \cdot \{ \lceil 2^n * S \rceil .. \lfloor 2^n * T \rfloor \}$

definition *dyadic-interval* $:: \text{real} \Rightarrow \text{real} \Rightarrow \text{real set}$

where *dyadic-interval* S $T \equiv (\bigcup n. \text{dyadic-interval-step } n$ S $T)$

lemma *dyadic-interval-step-empty*[*simp*]: $T < S \implies \text{dyadic-interval-step } n$ S $T = \{\}$

unfolding *dyadic-interval-step-def* **apply** *simp*

by (*smt* (*verit*) *ceil-pow2-geq floor-le-ceiling floor-mono floor-pow2-leq*

linordered-comm-semiring-strict-class.comm-mult-strict-left-mono zero-less-power)

lemma *dyadic-interval-step-singleton*[*simp*]: $X \in \mathbb{Z} \implies \text{dyadic-interval-step } n$ X

$X = \{X\}$

proof –

assume $X \in \mathbb{Z}$

then have $*$: $\lfloor 2^k * X \rfloor = 2^k * X$ **for** $k :: \text{nat}$

by *simp*

then show *?thesis*

unfolding *dyadic-interval-step-def* **apply** (*simp add: ceiling-altdef*)

using $*$ **by** *presburger*

qed

lemma *dyadic-interval-step-zero* [simp]: *dyadic-interval-step* 0 *S T* = *real-of-int* ‘
 $\{ \lceil S \rceil .. \lfloor T \rfloor \}$

unfolding *dyadic-interval-step-def* **by** *simp*

lemma *dyadic-interval-step-mem* [intro]:

assumes $x \geq 0$ $T \geq 0$ $x \leq T$

shows $\lfloor 2^n * x \rfloor / 2^n \in \text{dyadic-interval-step } n \ 0 \ T$

unfolding *dyadic-interval-step-def* **by** (*simp add: assms image-iff floor-mono*)

lemma *dyadic-interval-step-iff*:

$x \in \text{dyadic-interval-step } n \ S \ T \longleftrightarrow$

$(\exists k. k \geq \lceil 2^n * S \rceil \wedge k \leq \lfloor 2^n * T \rfloor \wedge x = k / 2^n)$

unfolding *dyadic-interval-step-def* **by** (*auto simp add: image-iff*)

lemma *dyadic-interval-step-memI* [intro]:

assumes $\exists k::\text{int}. x = k / 2^n$ $x \geq S$ $x \leq T$

shows $x \in \text{dyadic-interval-step } n \ S \ T$

proof –

obtain $k::\text{int}$ **where** $x = k / 2^n$

using *assms(1)* **by** *blast*

then have $k = 2^n * x$

by *simp*

then have $k \geq \lceil 2^n * S \rceil$

by (*simp add: assms(2) ceiling-le*)

moreover from k **have** $k \leq \lfloor 2^n * T \rfloor$

by (*simp add: assms(3) le-floor-iff*)

ultimately show *?thesis*

using *dyadic-interval-step-iff* $\langle x = k / 2^n \rangle$ **by** *blast*

qed

lemma *mem-dyadic-interval*: $x \in \text{dyadic-interval } S \ T \longleftrightarrow (\exists n. x \in \text{dyadic-interval-step } n \ S \ T)$

unfolding *dyadic-interval-def* **by** *blast*

lemma *mem-dyadic-intervalI*: $\exists n. x \in \text{dyadic-interval-step } n \ S \ T \implies x \in \text{dyadic-interval } S \ T$

using *mem-dyadic-interval* **by** *fast*

lemma *dyadic-step-leq*: $x \in \text{dyadic-interval-step } n \ S \ T \implies x \leq T$

unfolding *dyadic-interval-step-def* **apply** *clarsimp*

by (*simp add: divide-le-eq le-floor-iff mult.commute*)

lemma *dyadics-leq*: $x \in \text{dyadic-interval } S \ T \implies x \leq T$

using *dyadic-step-leq mem-dyadic-interval* **by** *blast*

lemma *dyadic-step-geq*: $x \in \text{dyadic-interval-step } n \ S \ T \implies x \geq S$

unfolding *dyadic-interval-step-def* **apply** *clarsimp*

by (*simp add: ceiling-le-iff mult.commute pos-le-divide-eq*)

lemma *dyadics-geq*: $x \in \text{dyadic-interval } S \ T \implies x \geq S$
using *dyadic-step-geq mem-dyadic-interval* **by** *blast*

corollary *dyadic-interval-subset-interval* [*simp*]: $(\text{dyadic-interval } 0 \ T) \subseteq \{0..T\}$
using *dyadics-geq dyadics-leq* **by** *force*

lemma *zero-in-dyadics*: $T \geq 0 \implies 0 \in \text{dyadic-interval-step } n \ 0 \ T$
using *dyadic-interval-step-def* **by** *force*

The following theorem is useful for reasoning with *at_within*

lemma *dyadic-interval-converging-sequence*:

assumes $t \in \{0..T\} \ T \neq 0$

shows $\exists s. \forall n. s \ n \in \text{dyadic-interval } 0 \ T - \{t\} \wedge s \longrightarrow t$

proof –

from *assms* **have** $T > 0$

by *auto*

consider $(\text{eq-0}) \ t = 0 \mid (\text{dyadic}) \ t \in \text{dyadic-interval } 0 \ T - \{0\} \mid (\text{real}) \ t \notin \text{dyadic-interval } 0 \ T$

by *blast*

then show *?thesis*

proof *cases*

case *eq-0*

obtain n **where** $1 \leq 2^n * T$

proof –

assume $*$: $\bigwedge n. 1 \leq 2^n * T \implies \text{thesis}$

obtain n **where** $2^n > 1/T$

using *real-arch-pow* **by** *fastforce*

then have $2^n * T \geq 1$

using $\langle T > 0 \rangle$ **by** (*simp add: field-simps*)

then show *?thesis*

using $*$ **by** *blast*

qed

define $s :: \text{nat} \Rightarrow \text{real}$ **where** $s = (\lambda m. 1/2^{(m+n)})$

have $\forall m. s \ m \in \text{dyadic-interval-step } (m+n) \ 0 \ T - \{0\}$

unfolding *s-def* **apply** (*simp add: dyadic-interval-step-iff*)

using $\langle 1 \leq 2^n * T \rangle$

by (*smt (verit, best) $\langle 0 < T \rangle$ le-add2 mult-right-mono power-increasing-iff*)

then have $\forall m. s \ m \in \text{dyadic-interval } 0 \ T - \{0\}$

using *mem-dyadic-interval* **by** *auto*

moreover {

have $(\lambda m. (1::\text{real})/2^m) \longrightarrow 0$

by (*simp add: divide-real-def LIMSEQ-inverse-realpow-zero*)

then have $s \longrightarrow 0$

unfolding *s-def* **using** *LIMSEQ-ignore-initial-segment* **by** *auto*

}

ultimately show *?thesis*

using *eq-0* **by** *blast*

next

case *dyadic*

```

then have  $t \neq 0$ 
  by blast
from dyadic obtain  $n$  where  $n: t \in \text{dyadic-interval-step } n \ 0 \ T$ 
  by (auto simp: mem-dyadic-interval)
then obtain  $k :: \text{int}$  where  $k: t = k / 2^n \ k \leq \lfloor 2^n * T \rfloor$ 
  using dyadic-interval-step-iff by blast
then have  $k > 0$ 
  using  $\langle t \neq 0 \rangle$  dyadic-interval-step-iff  $n$  by force
define  $s :: \text{nat} \Rightarrow \text{real}$  where  $s \equiv \lambda m. (k * 2^{m+1} - 1) / 2^{m+n+1}$ 
have  $s \ m \in \text{dyadic-interval-step } (m + n + 1) \ 0 \ T$  for  $m$ 
proof -
  have  $k * (2^{m+1}) - 1 \leq \lfloor 2^n * T \rfloor * (2^{m+1}) - 1$ 
    by (smt (verit)  $k(2)$  mult-right-mono zero-le-power)
  also have  $\dots \leq \lfloor 2^n * T \rfloor * \lfloor 2^{m+1} \rfloor$ 
  by (metis add.commute add-le-cancel-left diff-add-cancel diff-self floor-numeral-power
    zero-less-one-class.zero-le-one)
  also have  $\lfloor 2^n * T \rfloor * \lfloor 2^{m+1} \rfloor \leq \lfloor 2^n * T * 2^{m+1} \rfloor$ 
    by (smt (z3)  $\langle 0 < T \rangle$  floor-one floor-power le-mult-floor mult-nonneg-nonneg
of-int-1
    of-int-add one-add-floor one-add-one zero-le-power)
  also have  $\dots = \lfloor (2^{m+n+1}) * T \rfloor$ 
    apply (rule arg-cong[where  $f = \text{floor}$ ])
    by (simp add: power-add)
  finally show ?thesis
    unfolding s-def apply (simp only: dyadic-interval-step-iff)
    apply (rule exI[where  $x = k * (2^{m+1}) - 1$ ])
    by (simp add:  $\langle 0 < k \rangle$ )
qed
then have  $s \ m \in \text{dyadic-interval } 0 \ T$  for  $m$ 
  using mem-dyadic-interval by blast
moreover have  $s \ m \neq t$  for  $m$ 
  unfolding s-def  $k(1)$  by (simp add: power-add field-simps)
moreover have  $s \longrightarrow t$ 
proof
  fix  $e :: \text{real}$  assume  $0 < e$ 
  then obtain  $m$  where  $1 / 2^m < e$ 
    by (metis one-less-numeral-iff power-one-over reals-power-lt-ex semir-
ing-norm(76))
  { fix  $m'$  assume  $m' \geq m$ 
    then have  $1 / 2^{m'} < e$ 
      using  $\langle 1 / 2^m < e \rangle$ 
    by (smt (verit) frac-less2 le-eq-less-or-eq power-strict-increasing zero-less-power)
    then have  $1 / 2^{m'+n+1} < e$ 
    by (smt (verit, ccfv-SIG) divide-less-eq-1-pos half-gt-zero-iff power-less-imp-less-exp
      power-one-over power-strict-decreasing trans-less-add1)
    have  $s \ m' - t = (k * 2^{m'+1} - 1) / 2^{m'+n+1} - k / 2^n$ 
      by (simp add: s-def  $k(1)$ )
    also have  $\dots = ((k * 2^{m'+1} - 1) - (k * 2^{m'+1})) / 2^{m'+n+1}$ 

```

```

n + 1)
  by (simp add: field-simps power-add)
also have ... = -1 / 2(m'+n+1)
  by (simp add: field-simps)
finally have dist (s m') t < e
  unfolding s-def k(1)
  apply (simp add: dist-real-def)
  using <1 / 2(m' + n + 1) < e> by auto
}
then show  $\forall_F x$  in sequentially. dist (s x) t < e
  apply (simp add: eventually-sequentially)
  apply (intro exI[where x=m])
  by simp
qed
ultimately show ?thesis
  by blast
next
case real
then obtain n where dyadic-interval-step n 0 T ≠ {}
  by (metis <0 < T> empty-iff less-eq-real-def zero-in-dyadics)
define s :: nat ⇒ real where s ≡ λm. ⌊2(m+n) * t⌋ / 2(m+n)
have s m ∈ dyadic-interval-step (m+n) 0 T for m
  unfolding s-def
  by (metis assms(1) atLeastAtMost-iff ceiling-zero dyadic-interval-step-iff
floor-mono
mult.commute mult-eq-0-iff mult-right-mono zero-le-floor zero-le-numeral
zero-le-power)
then have s m ∈ dyadic-interval 0 T for m
  using mem-dyadic-interval by blast
moreover have s ⟶ t
  unfolding s-def using LIMSEQ-ignore-initial-segment floor-pow2-lim by
blast
ultimately show ?thesis
  using real by blast
qed
qed

lemma dyadic-interval-dense: closure (dyadic-interval 0 T) = {0..T}
proof (rule subset-antisym)
  have (dyadic-interval 0 T) ⊆ {0..T}
    by (fact dyadic-interval-subset-interval)
  then show closure (dyadic-interval 0 T) ⊆ {0..T}
    by (auto simp: closure-minimal)
  have {0..T} ⊆ closure (dyadic-interval 0 T) if T ≥ 0
    unfolding closure-def
  proof -
    {
      fix x assume x: 0 ≤ x ≤ T x ∉ dyadic-interval 0 T
      then have x > 0

```

```

    unfolding dyadic-interval-def
    using zero-in-dyadics[OF that] order-le-less by blast
  have x islimpt (dyadic-interval 0 T)
  apply (simp add: islimpt-sequential)
  apply (rule exI [where x= $\lambda n. \lfloor 2^n * x \rfloor / 2^n$ ])
  apply safe
  using dyadic-interval-step-mem mem-dyadic-interval x(1,2) apply auto[1]
  apply (smt (verit, ccfv-threshold) dyadic-interval-step-mem mem-dyadic-interval
x)
    using floor-pow2-lim apply blast
  done
}
thus  $\{0..T\} \subseteq \text{dyadic-interval } 0 \ T \cup \{x. x \text{ islimpt dyadic-interval } 0 \ T\}$ 
  by force
qed
then show  $\{0..T\} \subseteq \text{closure } (\text{dyadic-interval } 0 \ T)$ 
  by (cases  $T \geq 0$ ; simp)
qed

corollary dyadic-interval-islimpt:
  assumes  $T > 0 \ t \in \{0..T\}$ 
  shows  $t \text{ islimpt dyadic-interval } 0 \ T$ 
  using assms by (subst limpt-of-closure[symmetric], simp add: dyadic-interval-dense)

corollary at-within-dyadic-interval-nontrivial[simp]:
  assumes  $T > 0 \ t \in \{0..T\}$ 
  shows  $(\text{at } t \text{ within dyadic-interval } 0 \ T) \neq \text{bot}$ 
  using assms dyadic-interval-islimpt trivial-limit-within by blast

lemma dyadic-interval-step-finite[simp]: finite (dyadic-interval-step n S T)
  unfolding dyadic-interval-step-def by simp

lemma dyadic-interval-countable[simp]: countable (dyadic-interval S T)
  by (simp add: dyadic-interval-def dyadic-interval-step-def)

lemma floor-pow2-add-leq:
  fixes  $T :: \text{real}$ 
  shows  $\lfloor 2^n * T \rfloor / 2^n \leq \lfloor 2^{n+k} * T \rfloor / 2^{n+k}$ 
proof (induction k)
  case 0
  then show ?case by simp
next
  case (Suc k)
  let ?f = frac  $(2^{n+k} * T)$ 
  and ?f' = frac  $(2^{n+(Suc k)} * T)$ 
  show ?case
  proof (cases ?f < 1/2)
    case True
    then have ?f + ?f < 1

```

```

    by auto
  then have frac  $((2^{n+k} * T) + (2^{n+k} * T)) = ?f + ?f$ 
    using frac-add by meson
  then have  $?f' = ?f + ?f$ 
    by (simp add: field-simps)
  then have  $\lfloor 2^{n+Suc\ k} * T \rfloor / 2^{n+Suc\ k} = \lfloor 2^{n+k} * T \rfloor / 2^{n+k}$ 
    by (simp add: frac-def)
  then show ?thesis
    using Suc by presburger
next
case False
have  $?f' = \text{frac } (2^{n+k} * T + 2^{n+k} * T)$ 
  by (simp add: field-simps)
then have  $?f' = 2 * ?f - 1$ 
  by (smt (verit, del-insts) frac-add False field-sum-of-halves)
then have  $?f' < ?f$ 
  using frac-lt-1 by auto
then have  $(2^{n+k} * T - ?f) / 2^{n+k} < (2^{n+Suc\ k} * T - ?f) / 2^{n+Suc\ k}$ 
  by (smt (verit, ccfv-threshold) frac-ge-0)
  apply (simp add: field-simps)
  by (smt (verit, ccfv-threshold) frac-ge-0)
then show ?thesis
  by (smt (verit, ccfv-SIG) Suc frac-def)
qed
qed

corollary floor-pow2-mono: mono  $(\lambda n. \lfloor 2^n * (T :: real) \rfloor / 2^n)$ 
  apply (intro monoI)
  subgoal for x y
    using floor-pow2-add-leq[of x T y - x] by force
  done

lemma dyadic-interval-step-Max:  $T \geq 0 \implies \text{Max } (\text{dyadic-interval-step } n\ 0\ T) = \lfloor 2^n * T \rfloor / 2^n$ 
  apply (simp add: dyadic-interval-step-def)
  apply (subst mono-Max-commute[of  $\lambda x. \text{real-of-int } x / 2^n$ , symmetric])
  by (auto simp: mono-def field-simps Max-eq-iff)

lemma dyadic-interval-step-subset:
   $n \leq m \implies \text{dyadic-interval-step } n\ 0\ T \subseteq \text{dyadic-interval-step } m\ 0\ T$ 
proof (rule subsetI)
  fix x assume  $n \leq m$   $x \in \text{dyadic-interval-step } n\ 0\ T$ 
  then obtain k where  $k \geq 0$   $k \leq \lfloor 2^n * T \rfloor$   $x = k / 2^n$ 
  unfolding dyadic-interval-step-def by fastforce
  then have  $k * 2^{m-n} \in \{0 .. \lfloor 2^m * T \rfloor\}$ 
  proof -
    have  $k / 2^n \leq \lfloor 2^m * T \rfloor / 2^m$ 
    by (smt floor-pow2-mono[THEN monoD, OF  $\langle n \leq m \rangle$ ] k(2) divide-right-mono)
  
```

```

of-int-le-iff zero-le-power)
  then have  $k / 2^n * 2^m \leq \lfloor 2^m * T \rfloor$ 
    by (simp add: field-simps)
  moreover have  $k / 2^n * 2^m = k * 2^{(m - n)}$ 
    apply (simp add: field-simps)
    apply (metis  $\langle n \leq m \rangle$  add-diff-inverse-nat not-less power-add)
  done
  ultimately have  $k * 2^{(m - n)} \leq \lfloor 2^m * T \rfloor$ 
    by linarith
  then show  $k * 2^{(m - n)} \in \{0 .. \lfloor 2^m * T \rfloor\}$ 
    using k(1) by simp
qed
then show  $x \in \text{dyadic-interval-step } m \ 0 \ T$ 
  apply (subst dyadic-interval-step-iff)
  apply (rule exI[where  $x = k * 2^{(m - n)}$ ])
  apply simp
  apply (simp add:  $\langle n \leq m \rangle$  k(3) power-diff)
  done
qed

corollary dyadic-interval-step-mono:
  assumes  $x \in \text{dyadic-interval-step } n \ 0 \ T \ n \leq m$ 
  shows  $x \in \text{dyadic-interval-step } m \ 0 \ T$ 
  using assms dyadic-interval-step-subset by blast

lemma dyadic-as-natural:
  assumes  $x \in \text{dyadic-interval-step } n \ 0 \ T$ 
  shows  $\exists! k. x = \text{real } k / 2^n$ 
  using assms
proof (induct n)
  case 0
  then show ?case
    apply simp
    by (metis 0 ceiling-zero div-by-1 dyadic-interval-step-iff mult-not-zero of-nat-eq-iff
of-nat-nat power.simps(1))
next
  case (Suc n)
  then show ?case
    by (auto simp: dyadic-interval-step-iff, metis of-nat-nat)
qed

lemma dyadic-of-natural:
  assumes  $\text{real } k / 2^n \leq T$ 
  shows  $\text{real } k / 2^n \in \text{dyadic-interval-step } n \ 0 \ T$ 
  using assms apply (induct n)
  apply simp
  apply (metis atLeastAtMost-iff imageI le-floor-iff of-int-of-nat-eq of-nat-0-le-iff)
  apply (simp add: dyadic-interval-step-iff)
  by (smt (verit, ccfv-SIG) divide-le-eq le-floor-iff mult.commute of-int-of-nat-eq

```

of-nat-0-le-iff zero-less-power)

lemma *dyadic-interval-minus:*

assumes $x \in \text{dyadic-interval-step } n \ 0 \ T \ y \in \text{dyadic-interval-step } n \ 0 \ T \ x \leq y$

shows $y - x \in \text{dyadic-interval-step } n \ 0 \ T$

proof –

obtain $kx :: \text{nat}$ **where** $x = \text{real } kx / 2^n$

using *dyadic-as-natural* *assms(1)* **by** *blast*

obtain $ky :: \text{nat}$ **where** $y = \text{real } ky / 2^n$

using *dyadic-as-natural* *assms(2)* **by** *blast*

then have $y - x = (ky - kx) / 2^n$

by (*smt* (*verit*, *ccfv-SIG*) $\langle x = \text{real } kx / 2^n \rangle$ *add-diff-inverse-nat* *add-divide-distrib*

assms(3) *divide-strict-right-mono* *of-nat-add* *of-nat-less-iff* *zero-less-power*)

then show *?thesis*

using *dyadic-of-natural*

by (*smt* (*verit*, *best*) *assms(1,2)* *dyadic-step-geq* *dyadic-step-leq*)

qed

lemma *dyadic-times-nat:* $x \in \text{dyadic-interval-step } n \ 0 \ T \implies (x * 2^n) \in \mathbb{N}$

using *dyadic-as-natural* **by** *fastforce*

definition *dyadic-expansion* $x \ n \ b \ k \equiv \text{set } b \subseteq \{0,1\}$

$\wedge \text{length } b = n \wedge x = \text{real-of-int } k + (\sum_{m \in \{1..n\}} \text{real } (b \ ! \ (m-1)) / 2^m)$

lemma *dyadic-expansionI:*

assumes $\text{set } b \subseteq \{0,1\} \ \text{length } b = n \ x = k + (\sum_{m \in \{1..n\}} (b \ ! \ (m-1)) / 2^m)$

m)

shows *dyadic-expansion* $x \ n \ b \ k$

unfolding *dyadic-expansion-def* **using** *assms* **by** *blast*

lemma *dyadic-expansionD:*

assumes *dyadic-expansion* $x \ n \ b \ k$

shows $\text{set } b \subseteq \{0,1\}$

and $\text{length } b = n$

and $x = k + (\sum_{m \in \{1..n\}} (b \ ! \ (m-1)) / 2^m)$

using *assms* **unfolding** *dyadic-expansion-def* **by** *simp-all*

lemma *dyadic-expansion-ex:*

assumes $x \in \text{dyadic-interval-step } n \ 0 \ T$

shows $\exists b \ k. \ \text{dyadic-expansion } x \ n \ b \ k$

using *assms*

proof (*induction* n *arbitrary:* x)

case 0

then show *?case*

unfolding *dyadic-expansion-def* **by** *force*

next

case (*Suc* n)

then obtain k **where** $k: k \in \{0..[2^n (\text{Suc } n) * T]\} \ x = k / 2^n (\text{Suc } n)$

```

    unfolding dyadic-interval-step-def by fastforce
  then have div2:  $k \text{ div } 2 \in \{0..[2^n * T]\}$ 
    using k(1) apply simp
    by (metis divide-le-eq-numeral1(1) floor-divide-of-int-eq floor-mono le-floor-iff
mult.assoc mult.commute of-int-numeral)
  then show ?case
  proof (cases even k)
    case True
    then have  $x = k \text{ div } 2 / 2^n$ 
      by (simp add: k(2) real-of-int-div)
    then have  $x \in \text{dyadic-interval-step } n \ 0 \ T$ 
      using dyadic-interval-step-def div2 by force
    then obtain  $k' \ b$  where  $kb: \text{dyadic-expansion } x \ n \ b \ k'$ 
      using Suc(1) by blast
    show ?thesis
      apply (rule exI[where  $x=b @ [0]$ ])
      apply (rule exI[where  $x=k'$ ])
      unfolding dyadic-expansion-def apply safe
      using kb unfolding dyadic-expansion-def apply simp-all
      apply (auto intro!: sum.cong simp: nth-append)
      done
  next
    case False
    then have  $k = 2 * (k \text{ div } 2) + 1$ 
      by force
    then have  $x = k \text{ div } 2 / 2^n + 1 / 2^n (\text{Suc } n)$ 
      by (simp add: k(2) field-simps)
    then have  $x - 1 / 2^n (\text{Suc } n) \in \text{dyadic-interval-step } n \ 0 \ T$ 
      using div2 by (simp add: dyadic-interval-step-def)
    then obtain  $k' \ b$  where  $kb: \text{dyadic-expansion } (x - 1/2^n (\text{Suc } n)) \ n \ b \ k'$ 
      using Suc(1)[of  $x - 1 / 2^n (\text{Suc } n)$ ] by blast
    have  $x: x = \text{real-of-int } k' + (\sum m = 1..n. b! (m-1) / 2^m) + 1/2^n (\text{Suc } n)$ 
      using dyadic-expansionD(3)[OF kb] by (simp add: field-simps)
    show ?thesis
      apply (rule exI[where  $x=b @ [1]$ ])
      apply (rule exI[where  $x=k'$ ])
      unfolding dyadic-expansion-def apply safe
      using kb x unfolding dyadic-expansion-def apply simp-all
      apply (auto intro!: sum.cong simp: nth-append)
      done
  qed
qed

lemma dyadic-expansion-frac-le-1:
  assumes dyadic-expansion  $x \ n \ b \ k$ 
  shows  $(\sum m \in \{1..n\}. (b! (m-1)) / 2^m) < 1$ 
proof -
  have  $b! (m - 1) \in \{0,1\}$  if  $m \in \{1..n\}$  for  $m$ 
proof -

```



```

from assms have set  $b \subseteq \{0,1\}$  length  $b = n$ 
  unfolding dyadic-expansion-def by blast+
then have  $a < n \implies b \neq \{0,1\}$  for  $a$ 
  using nth-mem by blast
moreover have  $m - 1 < n$ 
  using that by force
ultimately show ?thesis
  by blast
qed
then have  $(\sum_{m \in \{1..n\}} (b \neq \{0,1\} \implies (m-1) / 2^m) \leq (\sum_{m \in \{1..n\}} 1 / 2^m)$ 
  apply (intro sum-mono)
  using assms by fastforce
also have  $\dots = 1 - 1/2^n$ 
  by (induct n, auto)
finally show ?thesis
  by (smt (verit, ccfv-SIG) add-divide-distrib divide-strict-right-mono zero-less-power)
qed

lemma dyadic-expansion-frac-range:
  assumes dyadic-expansion  $x$   $n$   $b$   $k$   $m \in \{1..n\}$ 
  shows  $b \neq \{0,1\} \implies (m-1) \in \{0,1\}$ 
proof -
  have  $m - 1 < \text{length } b$ 
  using dyadic-expansionD(2)[OF assms(1)] assms(2) by fastforce
  then show ?thesis
  using nth-mem dyadic-expansionD(1)[OF assms(1)] by blast
qed

lemma dyadic-expansion-interval:
  assumes dyadic-expansion  $x$   $n$   $b$   $k$   $x \in \{S..T\}$ 
  shows  $x \in \text{dyadic-interval-step } n$   $S$   $T$ 
proof (subst dyadic-interval-step-iff, intro exI, safe)
  define  $k'$  where  $k' \equiv k * 2^n + (\sum_{i = 1..n. b(i-1) * 2^{(n-i)})$ 
  show  $x = k' / 2^n$ 
  apply (simp add: dyadic-expansionD(3)[OF assms(1)] k'-def add-divide-distrib sum-divide-distrib)
  apply (intro sum.cong, simp)
  apply (simp add: field-simps)
  by (metis add-diff-inverse-nat linorder-not-le power-add)
  then have  $k' = \lfloor 2^n * x \rfloor$ 
  by simp
  then show  $k' \leq \lfloor 2^n * T \rfloor$ 
  using assms(2) by (auto intro!: floor-mono mult-left-mono)
  from  $\langle x = k' / 2^n \rangle$  have  $k' = \lceil 2^n * x \rceil$ 
  by force
  then show  $\lceil 2^n * S \rceil \leq k'$ 
  using assms(2) by (auto intro!: ceiling-mono mult-left-mono)
qed

```

lemma *dyadic-expansion-nth-geq*:
assumes *dyadic-expansion* $x\ n\ b\ k\ m \in \{1..n\}\ b \neq (m-1) = 1$
shows $x \geq k + 1/2^m$
proof –
have $(\sum_{i=1..n} f\ i) = f\ m + (\sum_{i \in (\{1..n\} - \{m\})} f\ i)$ **for** $f :: \text{nat} \Rightarrow \text{real}$
by (*meson* *assms*(2) *finite-atLeastAtMost* *sum.remove*)
with *dyadic-expansionD*(3)[*OF* *assms*(1)] *assms*(2,3)
have $x = k + b!(m-1)/2^m + (\sum_{i \in (\{1..n\} - \{m\})} b!(i-1) / 2^i)$
by *simp*
moreover **have** $(\sum_{i \in (\{1..n\} - \{m\})} b!(i-1) / 2^i) \geq 0$
by (*simp* *add: sum-nonneg*)
ultimately show *?thesis*
using *assms*(3) **by** *fastforce*
qed

lemma *dyadic-expansion-frac-geq-0*:
assumes *dyadic-expansion* $x\ n\ b\ k$
shows $(\sum_{m \in \{1..n\}} (b!(m-1)) / 2^m) \geq 0$
proof –
have $b!(m-1) \in \{0,1\}$ **if** $m \in \{1..n\}$ **for** m
using *dyadic-expansion-frac-range*[*OF* *assms*] **that** **by** *blast*
then **have** $(\sum_{m \in \{1..n\}} (b!(m-1)) / 2^m) \geq (\sum_{m \in \{1..n\}} 0)$
by (*intro* *sum-mono*, *fastforce*)
then show *?thesis*
by *auto*
qed

lemma *dyadic-expansion-frac*:
assumes *dyadic-expansion* $x\ n\ b\ k$
shows $\text{frac } x = (\sum_{m \in \{1..n\}} (b!(m-1)) / 2^m)$
apply (*simp* *add: frac-unique-iff*)
apply *safe*
using *dyadic-expansionD*(3)[*OF* *assms*] **apply** *simp*
using *dyadic-expansion-frac-geq-0*[*OF* *assms*] **apply** *simp*
using *dyadic-expansion-frac-le-1*[*OF* *assms*] **apply** *simp*
done

lemma *dyadic-expansion-floor*:
assumes *dyadic-expansion* $x\ n\ b\ k$
shows $k = \lfloor x \rfloor$
proof –
have $x = k + (\sum_{m \in \{1..n\}} (b!(m-1)) / 2^m)$
using *assms* **by** (*rule* *dyadic-expansionD*(3))
then **have** $x = k + \text{frac } x$
using *dyadic-expansion-frac*[*OF* *assms*] **by** *linarith*
then **have** $k = x - \text{frac } x$
by *simp*
then show $k = \lfloor x \rfloor$

by (metis floor-of-int frac-floor)
qed

lemma *sum-interval-pow2-inv*: $(\sum m \in \{Suc\ l..n\}. (1 :: real) / 2^m) = 1 / 2^l - 1 / 2^n$ if $l < n$
 using that **proof** (induct l)
 case 0
 then show ?case
 by (induct n; fastforce)
next
 case (Suc l)
 have $(\sum m \in \{Suc\ l..n\} - \{Suc\ l\}. (1 :: real) / 2^m) = (\sum m = Suc\ l..n. 1 / 2^m) - 1 / 2^{Suc\ l}$
 using Suc by (auto simp add: Suc sum-diff1, linarith)
 moreover have $\{Suc\ l..n\} - \{Suc\ l\} = \{Suc\ (Suc\ l)..n\}$
 by fastforce
 ultimately have $(\sum m = Suc\ (Suc\ l)..n. (1 :: real) / 2^m) = (\sum m = (Suc\ l)..n. 1 / 2^m) - 1 / 2^{Suc\ l}$
 by force
 also have $\dots = 1 / 2^l - 1 / 2^n - 1 / 2^{Suc\ l}$
 using Suc by linarith
 also have $\dots = 1 / 2^{Suc\ l} - 1 / 2^n$
 by (simp add: field-simps)
 finally show ?case
 by blast
qed

lemma *dyadic-expansion-unique*:
 assumes *dyadic-expansion* x n b k
 and *dyadic-expansion* x n c j
 shows $b = c \wedge j = k$
proof (safe, rule ccontr)
 show $j = k$
 using assms *dyadic-expansion-floor* by blast
 assume $b \neq c$
 have eq: $(\sum m \in \{1..n\}. (b ! (m-1)) / 2^m) = (\sum m \in \{1..n\}. (c ! (m-1)) / 2^m)$
proof -
 have $k + (\sum m \in \{1..n\}. (b ! (m-1)) / 2^m) = j + (\sum m \in \{1..n\}. (c ! (m-1)) / 2^m)$
 using assms *dyadic-expansionD*(3) by blast
 then show ?thesis
 using $\langle j = k \rangle$ by linarith
qed
 have ex: $\exists l < n. b ! l \neq c ! l$
 by (metis list-eq-iff-nth-eq assms $\langle b \neq c \rangle$ *dyadic-expansionD*(2))
 define l where $l \equiv \text{LEAST } l. l < n \wedge b ! l \neq c ! l$
 then have l: $l < n \wedge b ! l \neq c ! l$
 unfolding l-def using *LeastI-ex*[OF ex] by blast+

```

have less-l: b ! k = c ! k if <k < l> for k
proof -
  have k < n
    using that l by linarith
  then show b ! k = c ! k
    using that unfolding l-def using not-less-Least by blast
qed
then have l ∈ {0..n-1}
  using l by simp
then have l < n
  apply (simp add: algebra-simps)
  using ex by fastforce
then have b ! l ∈ {0,1} c ! l ∈ {0,1}
  by (metis assms insert-absorb insert-subset dyadic-expansionD(1,2) nth-mem)+
then consider b ! l = 0 ∧ c ! l = 1 | b ! l = 1 ∧ c ! l = 0
  by (smt (verit) LeastI-ex emptyE insertE l-def ex)
then have sum-ge-l-noteq: (∑ m∈{l+1..n}. (b ! (m-1)) / 2 ^ m) ≠ (∑ m∈{l+1..n}.
(c ! (m-1)) / 2 ^ m)
proof cases
  case 1
  have *: ?thesis if l + 1 = n
    using that 1 by auto
  {
    assume <l + 1 < n>
    have (∑ m∈{l+1..n}. (c ! (m-1)) / 2 ^ m) =
      (c ! ((l+1)-1)) / 2 ^ (l+1) + (∑ m∈{Suc (l+1)..n}. (c ! (m-1)) / 2 ^
m)
    by (smt (verit, ccfv-SIG) Suc-eq-plus1 Suc-le-mono Suc-pred' <l ∈ {0..n -
1}>
      atLeastAtMost-iff bot-nat-0.not-eq-extremum ex order-less-trans sum.atLeast-Suc-atMost)
    also have ... ≥ 1 / 2 ^ (l+1)
      apply (simp add: 1)
      apply (rule sum-nonneg)
      using dyadic-expansion-frac-range[OF assms(2)] by simp
    finally have c-ge: (∑ m∈{l+1..n}. (c ! (m-1)) / 2 ^ m) ≥ 1 / 2 ^ (l+1) .
    have (∑ m∈{l+1..n}. (b ! (m-1)) / 2 ^ m) =
      (b ! ((l+1)-1)) / 2 ^ (l+1) + (∑ m∈{Suc (l+1)..n}. (b ! (m-1)) / 2
^ m)
    by (meson <l + 1 < n> nat-less-le sum.atLeast-Suc-atMost)
    also have ... = (∑ m∈{Suc (l+1)..n}. (b ! (m-1)) / 2 ^ m)
      using 1 by auto
    also have ... ≤ (∑ m∈{Suc (l+1)..n}. 1 / 2 ^ m)
      apply (rule sum-mono)
    using dyadic-expansion-frac-range[OF assms(1)] apply (simp add: field-simps)
    by (metis (no-types, lifting) One-nat-def add-leE nle-le plus-1-eq-Suc)
    also have ... < 1 / 2 ^ (l+1)
      using sum-interval-pow2-inv[OF <l + 1 < n>] by fastforce
    finally have (∑ m∈{l+1..n}. (b ! (m-1)) / 2 ^ m) < 1 / 2 ^ (l+1) .
    with c-ge have ?thesis

```

```

    by argo
  }
  then show ?thesis
    using * ⟨l < n⟩ by linarith
next
case 2
have *: ?thesis if l + 1 = n
  using that 2 by auto
{
  assume ⟨l + 1 < n⟩
  have (∑ m ∈ {l+1..n}. (b ! (m-1)) / 2 ^ m) =
    (b ! ((l+1)-1)) / 2 ^ (l+1) + (∑ m ∈ {Suc (l+1)..n}. (b ! (m-1)) / 2 ^
m)
    by (meson ⟨l + 1 < n⟩ nat-less-le sum.atLeast-Suc-atMost)
  also have ... ≥ 1 / 2 ^ (l+1)
    apply (simp add: 2)
    apply (rule sum-nonneg)
    using dyadic-expansion-frac-range[OF assms(1)] by simp
  finally have b-ge: (∑ m ∈ {l+1..n}. (b ! (m-1)) / 2 ^ m) ≥ 1/2 ^ (l+1) .
  have (∑ m ∈ {l+1..n}. (c ! (m-1)) / 2 ^ m) =
    (c ! ((l+1)-1)) / 2 ^ (l+1) + (∑ m ∈ {Suc (l+1)..n}. (c ! (m-1)) / 2
^ m)
    by (meson ⟨l + 1 < n⟩ nat-less-le sum.atLeast-Suc-atMost)
  also have ... = (∑ m ∈ {Suc (l+1)..n}. (c ! (m-1)) / 2 ^ m)
    using 2 by auto
  also have ... ≤ (∑ m ∈ {Suc (l+1)..n}. 1 / 2 ^ m)
    apply (intro sum-mono divide-right-mono)
    using dyadic-expansion-frac-range[OF assms(2)]
    apply (metis (no-types, opaque-lifting) One-nat-def Suc-leI Suc-le-mono
atLeastAtMost-iff
atLeastAtMost-singleton-iff bot-nat-0.extremum bot-nat-0.not-eq-extremum

insert-iff of-nat-eq-1-iff of-nat-le-iff)
  apply simp
  done
  also have ... < 1 / 2 ^ (l+1)
    using sum-interval-pow2-inv[OF ⟨l + 1 < n⟩] by fastforce
  finally have (∑ m ∈ {l+1..n}. (c ! (m-1)) / 2 ^ m) < 1 / 2 ^ (l+1) .
  with b-ge have ?thesis
    by argo
}
then show ?thesis
  using * ⟨l < n⟩ by linarith
qed
moreover have sum-upto-l-eq: (∑ m ∈ {1..l}. (b ! (m-1)) / 2 ^ m) =
  (∑ m ∈ {1..l}. (c ! (m-1)) / 2 ^ m)
  apply (safe intro!: sum.cong)
  apply simp
by (smt (verit, best) Suc-le-eq Suc-pred ⟨l < n⟩ l-def not-less-Least order-less-trans)

```

```

ultimately have  $(\sum_{m \in \{1..n\}} (b ! (m-1)) / 2^m) \neq (\sum_{m \in \{1..n\}} (c ! (m-1)) / 2^m)$ 
proof -
  have  $\{1..n\} = \{1..l\} \cup \{l<..n\}$ 
  using  $\langle l < n \rangle$  by auto
  moreover have  $\{1..l\} \cap \{l<..n\} = \{\}$ 
  using ivl-disj-int-two(8) by blast
  ultimately have split-sum:  $(\sum_{m \in \{1..n\}} (c ! (m-1)) / 2^m) =$ 
 $(\sum_{m=1..l} (c ! (m-1)) / 2^m) + (\sum_{m \in \{l<..n\}} (c ! (m-1)) / 2^m)$ 
  for  $c :: nat$  list
  by (simp add: sum-Un)
  then show ?thesis
  using sum-upto-l-eq sum-ge-l-noteq split-sum[of b] split-sum[of c]
  by (smt (verit, del-insts) Suc-eq-plus1 atLeastSucAtMost-greaterThanAtMost)
qed
then show False
using eq by blast
qed
end

```

3 Hölder continuity

```

theory Holder-Continuous
  imports HOL-Analysis.Analysis
begin

```

Hölder continuity is a weaker version of Lipschitz continuity.

definition *holder-at-within* :: $real \Rightarrow 'a \text{ set} \Rightarrow 'a \Rightarrow ('a :: \text{metric-space} \Rightarrow 'b :: \text{metric-space}) \Rightarrow bool$ **where**

$$\text{holder-at-within } \gamma \ D \ r \ \varphi \equiv \gamma \in \{0 < .. 1\} \wedge$$

$$(\exists \varepsilon > 0. \exists C \geq 0. \forall s \in D. \text{dist } r \ s < \varepsilon \longrightarrow \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s^{\text{powr } \gamma})$$

definition *local-holder-on* :: $real \Rightarrow 'a :: \text{metric-space} \text{ set} \Rightarrow ('a \Rightarrow 'b :: \text{metric-space}) \Rightarrow bool$ **where**

$$\text{local-holder-on } \gamma \ D \ \varphi \equiv \gamma \in \{0 < .. 1\} \wedge$$

$$(\forall t \in D. \exists \varepsilon > 0. \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } s \ t < \varepsilon \wedge \text{dist } r \ t < \varepsilon \longrightarrow \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s^{\text{powr } \gamma}))$$

definition *holder-on* :: $real \Rightarrow 'a :: \text{metric-space} \text{ set} \Rightarrow ('a \Rightarrow 'b :: \text{metric-space}) \Rightarrow bool$ ($--holder'-on \ 1000$) **where**

$$\gamma\text{-holder-on } D \ \varphi \longleftrightarrow \gamma \in \{0 < .. 1\} \wedge (\exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s^{\text{powr } \gamma}))$$

lemma *holder-onI*:

assumes $\gamma \in \{0 < .. 1\} \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s^{\text{powr } \gamma})$

shows γ -holder-on $D \varphi$
unfolding *holder-on-def* **using** *assms* **by** *blast*

We prove various equivalent formulations of local holder continuity, using open and closed balls and inequalities.

lemma *local-holder-on-cball*:

local-holder-on $\gamma D \varphi \longleftrightarrow \gamma \in \{0 < .. 1\} \wedge$
 $(\forall t \in D. \exists \varepsilon > 0. \exists C \geq 0. (\forall r \in \text{cball } t \ \varepsilon \cap D. \forall s \in \text{cball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) (\varphi \ s)$
 $\leq C * \text{dist } r \ s \ \text{powr } \gamma))$
(is $?L \longleftrightarrow ?R$ **)**

proof

assume *: $?L$
{
 fix t **assume** $t \in D$
 then obtain $\varepsilon \ C$ **where** $\varepsilon > 0 \ C \geq 0$
 $\forall r \in \text{ball } t \ \varepsilon \cap D. \forall s \in \text{ball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s \ \text{powr } \gamma$
 using * **unfolding** *local-holder-on-def* **apply** *simp*
 by (*metis Int-iff dist-commute mem-ball*)
 then have **: $\forall r \in \text{cball } t \ (\varepsilon/2) \cap D. \forall s \in \text{cball } t \ (\varepsilon/2) \cap D. \text{dist } (\varphi \ r) (\varphi \ s)$
 $\leq C * \text{dist } r \ s \ \text{powr } \gamma$
 by *auto*
 have $\exists \varepsilon > 0. \exists C \geq 0. \forall r \in \text{cball } t \ \varepsilon \cap D. \forall s \in \text{cball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) (\varphi \ s)$
 $\leq C * \text{dist } r \ s \ \text{powr } \gamma$
 apply (*rule exI[where $x = \varepsilon/2$]*)
 apply (*simp add: $\langle \varepsilon > 0 \rangle$*)
 apply (*rule exI[where $x = C$]*)
 using ** $\langle C \geq 0 \rangle$ **by** *blast*
}
 then show $?R$
 using * *local-holder-on-def* **by** *blast*
next
 assume *: $?R$
 {
 fix t **assume** $t \in D$
 then obtain $\varepsilon \ C$ **where** $eC: \varepsilon > 0 \ C \geq 0$
 $\forall r \in \text{cball } t \ \varepsilon \cap D. \forall s \in \text{cball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s \ \text{powr } \gamma$
 using * **by** *blast*
 then have $\forall r \in D. \forall s \in D. \text{dist } r \ t < \varepsilon \wedge \text{dist } s \ t < \varepsilon \longrightarrow \text{dist } (\varphi \ r) (\varphi \ s)$
 $\leq C * \text{dist } r \ s \ \text{powr } \gamma$
 unfolding *cball-def* **by** (*simp add: dist-commute*)
 then have $\exists \varepsilon > 0. \exists C \geq 0. \forall r \in D. \forall s \in D. \text{dist } r \ t < \varepsilon \wedge \text{dist } s \ t < \varepsilon \longrightarrow$
 $\text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s \ \text{powr } \gamma$
 using eC **by** *blast*
 }
 then show *local-holder-on* $\gamma D \varphi$
 using * **unfolding** *local-holder-on-def* **by** *metis*
qed

corollary *local-holder-on-leq-def*: *local-holder-on* $\gamma D \varphi \longleftrightarrow \gamma \in \{0 < .. 1\} \wedge$

($\forall t \in D. \exists \varepsilon > 0. \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma)$)

unfolding *local-holder-on-cball* **by** (*metis dist-commute Int-iff mem-cball*)

corollary *local-holder-on-ball*: *local-holder-on* $\gamma \ D \ \varphi \longleftrightarrow \gamma \in \{0 < .. 1\} \wedge$
 $(\forall t \in D. \exists \varepsilon > 0. \exists C \geq 0. (\forall r \in \text{ball } t \ \varepsilon \cap D. \forall s \in \text{ball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma))$

unfolding *local-holder-on-def* **by** (*metis dist-commute Int-iff mem-ball*)

lemma *local-holder-on-altdef*:

assumes $D \neq \{\}$

shows *local-holder-on* $\gamma \ D \ \varphi = (\forall t \in D. (\exists \varepsilon > 0. (\gamma\text{-holder-on } ((\text{cball } t \ \varepsilon) \cap D) \ \varphi)))$

unfolding *local-holder-on-cball holder-on-def* **using** *assms* **by** *blast*

lemma *local-holder-on-cong*[*cong*]:

assumes $\gamma = \varepsilon \ C = D \ \bigwedge x. x \in C \implies \varphi \ x = \psi \ x$

shows *local-holder-on* $\gamma \ C \ \varphi \longleftrightarrow \text{local-holder-on } \varepsilon \ D \ \psi$

unfolding *local-holder-on-def* **using** *assms* **by** *presburger*

lemma *local-holder-onI*:

assumes $\gamma \in \{0 < .. 1\} \ (\forall t \in D. \exists \varepsilon > 0. \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } s \ t < \varepsilon \wedge \text{dist } r \ t < \varepsilon \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma))$

shows *local-holder-on* $\gamma \ D \ \varphi$

using *assms* **unfolding** *local-holder-on-def* **by** *blast*

lemma *local-holder-ballI*:

assumes $\gamma \in \{0 < .. 1\}$

and $\bigwedge t. t \in D \implies \exists \varepsilon > 0. \exists C \geq 0. \forall r \in \text{ball } t \ \varepsilon \cap D. \forall s \in \text{ball } t \ \varepsilon \cap D.$

$\text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma$

shows *local-holder-on* $\gamma \ D \ \varphi$

using *assms* **unfolding** *local-holder-on-ball* **by** *blast*

lemma *local-holder-onE*:

assumes *local-holder*: *local-holder-on* $\gamma \ D \ \varphi$

and *gamma*: $\gamma \in \{0 < .. 1\}$

and $t \in D$

obtains $\varepsilon \ C$ **where** $\varepsilon > 0 \ C \geq 0$

$\bigwedge r \ s. r \in \text{ball } t \ \varepsilon \cap D \implies s \in \text{ball } t \ \varepsilon \cap D \implies \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma$

using *assms* **unfolding** *local-holder-on-ball* **by** *auto*

Holder continuity matches up with the existing definitions in *HOL-Analysis.Lipschitz*

lemma *holder-1-eq-lipschitz*: *1-holder-on* $D \ \varphi = (\exists C. \text{lipschitz-on } C \ D \ \varphi)$

unfolding *holder-on-def lipschitz-on-def* **by** (*auto simp: fun-eq-iff dist-commute*)

lemma *local-holder-1-eq-local-lipschitz*:

assumes $T \neq \{\}$

shows *local-holder-on* $1 \ D \ \varphi = \text{local-lipschitz } T \ D \ (\lambda-. \varphi)$


```

proof
  assume *: local-holder-on 1 D  $\varphi$ 
  {
    fix t assume  $t \in D$ 
    then obtain  $\varepsilon$  C where eC:  $\varepsilon > 0$   $C \geq 0$ 
      ( $\forall r \in D. \forall s \in D. \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon \longrightarrow \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s$ )
      using * powr-to-1 unfolding local-holder-on-cball apply simp
      by (metis Int-iff dist-commute mem-cball)
    {
      fix r s assume rs:  $r \in D \ s \in D \ \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon$ 
      then have  $r \in \text{cball } t \ \varepsilon \cap D \ s \in \text{cball } t \ \varepsilon \cap D \ \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s$ 
        unfolding cball-def using rs eC by (auto simp: dist-commute)
    }
    then have  $\forall r \in \text{cball } t \ \varepsilon \cap D. \forall s \in \text{cball } t \ \varepsilon \cap D. \text{dist } (\varphi \ r) (\varphi \ s) \leq C * \text{dist } r \ s$ 
      by (simp add: dist-commute)
    then have C-lipschitz-on ((cball t  $\varepsilon$ )  $\cap D$ )  $\varphi$ 
      using eC lipschitz-on-def by blast
    then have  $\exists \varepsilon > 0. \exists C. C\text{-lipschitz-on } ((\text{cball } t \ \varepsilon) \cap D) \ \varphi$ 
      using eC(1) by blast
  }
  then show local-lipschitz T D ( $\lambda\cdot. \varphi$ )
    unfolding local-lipschitz-def by blast
next
  assume *: local-lipschitz T D ( $\lambda\cdot. \varphi$ )
  {
    fix x assume  $x \in D$ 
    fix t assume  $t \in T$ 
    then obtain u L where uL:  $u > 0 \ \forall t \in \text{cball } t \ u \cap T. L\text{-lipschitz-on } (\text{cball } x \ u \cap D) \ \varphi$ 
      using * x t unfolding local-lipschitz-def by blast
    then have L-lipschitz-on (cball x u  $\cap D$ )  $\varphi$ 
      using t by force
    then have 1-holder-on (cball x u  $\cap D$ )  $\varphi$ 
      using holder-1-eq-lipschitz by blast
    then have  $\exists \varepsilon > 0. (1\text{-holder-on } ((\text{cball } x \ \varepsilon) \cap D) \ \varphi)$ 
      using uL by blast
  }
  then have  $x \in D \implies \exists \varepsilon > 0. (1\text{-holder-on } (\text{cball } x \ \varepsilon \cap D) \ \varphi)$  for x
    using assms by blast
  then show local-holder-on 1 D  $\varphi$ 
    unfolding local-holder-on-cball holder-on-def by (auto simp: dist-commute)
qed

```

```

lemma local-holder-refine:
  assumes g: local-holder-on g D  $\varphi$   $g \leq 1$ 
    and h:  $h \leq g \ h > 0$ 
  shows local-holder-on h D  $\varphi$ 
proof –
  {

```

```

fix  $t$  assume  $t: t \in D$ 
then have  $\exists \varepsilon > 0. \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } g)$ 
using  $g(1)$  unfolding local-holder-on-leq-def by blast
then obtain  $\varepsilon \ C$  where  $eC: \varepsilon > 0 \ C \geq 0$ 
 $(\forall s \in D. \forall r \in D. \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } g)$ 
by blast
let  $?e = \min \varepsilon \ (1/2)$ 
{
fix  $s \ r$  assume  $*: s \in D \ r \in D \ \text{dist } s \ t \leq ?e \ \text{dist } r \ t \leq ?e$ 
then have  $\text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } g$ 
using  $eC$  by simp
moreover have  $\text{dist } r \ s \leq 1$ 
by  $(\text{smt } (\text{verit}) * \text{dist-triangle2 } \text{half-bounded-equal})$ 
ultimately have  $\text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } h$ 
by  $(\text{metis } \text{dual-order.trans } \text{zero-le-dist } \text{powr-mono}' \text{assms}(3) \ eC(2) \text{mult-left-mono})$ 
}
then have  $(\forall s \in D. \forall r \in D. \text{dist } s \ t \leq ?e \wedge \text{dist } r \ t \leq ?e \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } h)$ 
by blast
moreover have  $?e > 0 \ C \geq 0$ 
using  $eC$  by linarith+
ultimately have  $\exists \varepsilon > 0. \exists C \geq 0. (\forall r \in D. \forall s \in D. \text{dist } s \ t \leq \varepsilon \wedge \text{dist } r \ t \leq \varepsilon \longrightarrow \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } h)$ 
by blast
}
then show  $?thesis$ 
unfolding local-holder-on-leq-def using assms by force
qed

lemma holder-uniform-continuous:
assumes  $\gamma\text{-holder-on } X \ \varphi$ 
shows uniformly-continuous-on  $X \ \varphi$ 
unfolding uniformly-continuous-on-def
proof safe
fix  $e::\text{real}$ 
assume  $0 < e$ 
from assms obtain  $C$  where  $C: C \geq 1 \ (\forall r \in X. \forall s \in X. \text{dist } (\varphi \ r) \ (\varphi \ s) \leq C * \text{dist } r \ s \text{ powr } \gamma)$ 
unfolding holder-on-def
by  $(\text{smt } (\text{verit}) \text{dist-eq-0-iff } \text{mult-le-cancel-right1 } \text{powr-0 } \text{powr-gt-zero})$ 
{
fix  $r \ s$  assume  $r \in X \ s \in X$ 
have  $\text{dist-0: } \text{dist } (\varphi \ r) \ (\varphi \ s) = 0 \implies \text{dist } (\varphi \ r) \ (\varphi \ s) < e$ 
using  $\langle 0 < e \rangle$  by linarith
then have holder-neq-0:  $\text{dist } (\varphi \ r) \ (\varphi \ s) < (C + 1) * \text{dist } r \ s \text{ powr } \gamma$  if  $\text{dist } (\varphi \ r) \ (\varphi \ s) > 0$ 

```

```

    using C(2) that
    by (smt (verit, ccfv-SIG) ⟨r ∈ X⟩ ⟨s ∈ X⟩ dist-eq-0-iff mult-le-cancel-right
    powr-gt-zero)
    have gamma:  $\gamma \in \{0 <..1\}$ 
    using assms holder-on-def by blast+
    assume dist r s < (e/C) powr (1 /  $\gamma$ )
    then have C * dist r s powr  $\gamma$  < C * ((e/C) powr (1 /  $\gamma$ )) powr  $\gamma$  if dist (φ
    r) (φ s) > 0
    using holder-neq-0 C(1) powr-less-mono2 gamma by fastforce
    also have ... = e
    using C(1) gamma ⟨0 < e⟩ powr-powr by auto
    finally have dist (φ r) (φ s) < e
    using dist-0 holder-neq-0 C(2) ⟨r ∈ X⟩ ⟨s ∈ X⟩ by fastforce
  }
  then show  $\exists d > 0. \forall x \in X. \forall x' \in X. \text{dist } x' x < d \longrightarrow \text{dist } (\varphi x') (\varphi x) < e$ 
  by (metis C(1) ⟨0 < e⟩ divide-eq-0-iff linorder-not-le order-less-irrefl powr-gt-zero
  zero-less-one)
qed

```

corollary *holder-on-continuous-on*: γ -holder-on X $\varphi \implies$ continuous-on X φ
 using holder-uniform-continuous uniformly-continuous-imp-continuous by blast

lemma *holder-implies-local-holder*: γ -holder-on D $\varphi \implies$ local-holder-on γ D φ
 apply (cases $D = \{\}$)
 apply (simp add: holder-on-def local-holder-on-def)
 apply (simp add: local-holder-on-altdef holder-on-def)
 apply (metis IntD1 inf commute)
 done

lemma *local-holder-imp-continuous*:
 assumes local-holder: local-holder-on γ X φ
 shows continuous-on X φ
 unfolding continuous-on-def
proof safe
 fix x assume x ∈ X
 {
 assume $X \neq \{\}$
 from local-holder obtain ε where $0 < \varepsilon$ and holder: γ -holder-on ((cball x ε)
 $\cap X$) φ
 unfolding local-holder-on-altdef[OF ⟨ $X \neq \{\}$ ⟩] using ⟨x ∈ X⟩ by blast
 have x ∈ ball x ε using ⟨0 < ε ⟩ by simp
 then have (φ $\longrightarrow$ φ x) (at x within cball x ε $\cap X$)
 using holder-on-continuous-on[OF holder] ⟨x ∈ X⟩ unfolding continu-
 ous-on-def by simp
 moreover have $\forall_F xa$ in at x. (xa ∈ cball x ε $\cap X$) = (xa ∈ X)
 using eventually-at-ball[OF ⟨0 < ε ⟩, of x UNIV]
 by eventually-elim auto
 ultimately have (φ $\longrightarrow$ φ x) (at x within X)

```

    by (rule Lim-transform-within-set)
  }
  then show  $(\varphi \longrightarrow \varphi x)$  (at  $x$  within  $X$ )
    by fastforce
qed

```

```

lemma local-holder-compact-imp-holder:
  assumes compact  $I$  local-holder-on  $\gamma$   $I$   $\varphi$ 
  shows  $\gamma$ -holder-on  $I$   $\varphi$ 
proof -
  have *:  $\gamma \in \{0 < .. 1\}$   $(\forall t \in I. \exists \varepsilon. \exists C. \varepsilon > 0 \wedge C \geq 0 \wedge$ 
     $(\forall r \in \text{ball } t \ \varepsilon \cap I. \forall s \in \text{ball } t \ \varepsilon \cap I. \text{dist } (\varphi r) (\varphi s) \leq C * \text{dist } r s \text{ powr } \gamma))$ 
    using assms(2) unfolding local-holder-on-ball by simp-all
  obtain  $\varepsilon$   $C$  where  $eC$ :  $t \in I \implies \varepsilon t > 0 \wedge C t \geq 0 \wedge (\forall r \in \text{ball } t \ (\varepsilon t) \cap I.$ 
 $\forall s \in \text{ball } t \ (\varepsilon t) \cap I. \text{dist } (\varphi r) (\varphi s) \leq C t * \text{dist } r s \text{ powr } \gamma)$  for  $t$ 
    by (metis *(2))
  have  $I \subseteq (\bigcup t \in I. \text{ball } t \ (\varepsilon t))$ 
    apply (simp add: subset-iff)
    using  $eC$  by force
  then obtain  $D$  where  $D$ :  $D \subseteq (\lambda t. \text{ball } t \ (\varepsilon t))$  ‘ $I$  finite  $D$   $I \subseteq \bigcup D$ 
    using compact-eq-Heine-Borel[of  $I$ ] apply (simp add: assms(1))
    by (smt (verit, ccfv-SIG) open-ball imageE mem-Collect-eq subset-iff)
  then obtain  $T$  where  $T$ :  $D = (\lambda t. \text{ball } t \ (\varepsilon t))$  ‘ $T$   $T \subseteq I$  finite  $T$ 
    by (meson finite-subset-image subset-image-iff)

```

ϱ is the Lebesgue number of the cover

```

from  $D$  obtain  $\varrho$  :: real where  $\varrho$ :  $\forall t \in I. \exists U \in D. \text{ball } t \ \varrho \subseteq U \ \varrho > 0$ 
  by (smt (verit, del-Insts) Elementary-Metric-Spaces.open-ball Heine-Borel-lemma
  assms(1) imageE subset-image-iff)
have bounded  $(\varphi \text{ ' } I)$ 
  by (metis compact-continuous-image compact-imp-bounded assms local-holder-imp-continuous)
then obtain  $l$  where  $l$ :  $\forall x \in I. \forall y \in I. \text{dist } (\varphi x) (\varphi y) \leq l$ 
  by (metis bounded-two-points image-eqI)

```

Simply need to construct C_bar such that it is greater than any of these

```

define  $C\_bar$  where  $C\_bar \equiv \max ((\sum t \in T. C t)) (l * \varrho \text{ powr } (- \gamma))$ 
have  $C\_bar\text{-le}$ :  $C\_bar \geq C t$  if  $t \in T$  for  $t$ 
proof -
  have  $ge\text{-}0$ :  $t \in T \implies C t \geq 0$  for  $t$ 
    using  $T(2)$   $eC$  by blast
  then have  $\sum (C \text{ ' } (T - \{t\})) \geq 0$ 
    by (metis (mono-tags, lifting) Diff-subset imageE subset-eq sum-nonneg)
  then have  $(\sum t \in T. C t) \geq C t$ 
    by (metis  $T(3)$   $ge\text{-}0$  sum-nonneg-leq-bound that)
  then have  $\max ((\sum t \in T. C t)) S \geq C t$  for  $S$ 
    by argo
  then show  $C\_bar \geq C t$ 
    unfolding  $C\_bar\text{-def}$  by blast
qed

```

```

{
  fix s r assume sr: s ∈ I r ∈ I
  {
    assume dist s r < ρ
    then obtain t where t: t ∈ T s ∈ ball t (ε t) r ∈ ball t (ε t)
      by (smt (verit) sr D T ρ ball-eq-empty centre-in-ball imageE mem-ball
subset-iff)
    then have dist (φ s) (φ r) ≤ C t * dist s r powr γ
      using eC[of t] T(2) sr by blast
    then have dist (φ s) (φ r) ≤ C-bar * dist s r powr γ
      by (smt (verit, best) t C-bar-le mult-right-mono powr-non-neg)
  } note le-rho = this
  {
    assume dist s r ≥ ρ
    then have dist (φ s) (φ r) ≤ l * (dist s r / ρ) powr γ
    proof -
      have (dist s r / ρ) ≥ 1
        using ⟨dist s r ≥ ρ⟩ ⟨ρ > 0⟩ by auto
      then have (dist s r / ρ) powr γ ≥ 1
        using *(1) ge-one-powr-ge-zero by auto
      then show dist (φ s) (φ r) ≤ l * (dist s r / ρ) powr γ
        using l
      by (metis dist-self dual-order.strict-trans1 leI less-le-not-le mult-less-cancel-left2
sr)
    qed
    also have ... ≤ C-bar * dist s r powr γ
    proof -
      have l * (dist s r / ρ) powr γ = l * ρ powr (- γ) * dist s r powr γ
        using ρ(2) divide-powr-uminus powr-divide by force
      also have ... ≤ C-bar * dist s r powr γ
        unfolding C-bar-def by (simp add: mult-right-mono)
      finally show l * (dist s r / ρ) powr γ ≤ C-bar * dist s r powr γ
        .
    qed
    finally have dist (φ s) (φ r) ≤ C-bar * dist s r powr γ
    .
  }
  then have dist (φ s) (φ r) ≤ C-bar * dist s r powr γ
    using le-rho by argo
}
then have ∀ r ∈ I. ∀ s ∈ I. dist (φ r) (φ s) ≤ C-bar * dist r s powr γ
  by simp
then show ?thesis
  unfolding holder-on-def
  by (metis *(1) C-bar-def dist-self div-by-0 divide-nonneg-pos divide-powr-uminus

      dual-order.trans l max.cobounded2 powr-0 powr-gt-zero)
qed

```

lemma *holder-const*: γ -holder-on C (λ -. c) $\longleftrightarrow \gamma \in \{0 < ..1\}$
unfolding *holder-on-def* **by** *auto*

lemma *local-holder-const*: local-holder-on γ C (λ -. c) $\longleftrightarrow \gamma \in \{0 < ..1\}$
using *holder-const* *holder-implies-local-holder* *local-holder-on-def* **by** *blast*

end

4 Convergence in measure

theory *Measure-Convergence*
imports *HOL-Probability.Probability*
begin

We use measure rather than emeasure because ennreal is not a metric space, which we need to reason about convergence. By intersecting with the set of finite measure A , we don't run into issues where infinity is collapsed to 0.

For finite measures this definition is equal to the definition without set A – see below.

definition *tendsto-measure* :: $'b$ measure $\Rightarrow ('a \Rightarrow 'b \Rightarrow ('c :: \{second-countable-topology, metric-space\}))$
 $\Rightarrow ('b \Rightarrow 'c) \Rightarrow 'a$ filter $\Rightarrow$ bool
where *tendsto-measure* M X l $F \equiv (\forall n. X\ n \in \text{borel-measurable } M) \wedge l \in \text{borel-measurable } M \wedge$
 $(\forall \varepsilon > 0. \forall A \in \text{fmeasurable } M.$
 $((\lambda n. \text{measure } M (\{\omega \in \text{space } M. \text{dist } (X\ n\ \omega) (l\ \omega) > \varepsilon\} \cap A)) \longrightarrow 0) F)$

abbreviation (**in** *prob-space*) *tendsto-prob* (**infixr** $\longrightarrow_P$ 55) **where**
 $(f \longrightarrow_P l) F \equiv \text{tendsto-measure } M\ f\ l\ F$

lemma *tendsto-measure-measurable[measurable-dest]*:
 $\text{tendsto-measure } M\ X\ l\ F \implies X\ n \in \text{borel-measurable } M$
unfolding *tendsto-measure-def* **by** *meson*

lemma *tendsto-measure-measurable-lim[measurable-dest]*:
 $\text{tendsto-measure } M\ X\ l\ F \implies l \in \text{borel-measurable } M$
unfolding *tendsto-measure-def* **by** *meson*

lemma *tendsto-measure-mono*: $F \leq F' \implies \text{tendsto-measure } M\ f\ l\ F' \implies \text{tendsto-measure } M\ f\ l\ F$
unfolding *tendsto-measure-def* **by** (*simp add: tendsto-mono*)

lemma *tendsto-measureI*:
assumes [*measurable*]: $\bigwedge n. X\ n \in \text{borel-measurable } M$ $l \in \text{borel-measurable } M$
and $\bigwedge \varepsilon A. \varepsilon > 0 \implies A \in \text{fmeasurable } M \implies$
 $((\lambda n. \text{measure } M (\{\omega \in \text{space } M. \text{dist } (X\ n\ \omega) (l\ \omega) > \varepsilon\} \cap A)) \longrightarrow 0) F$
shows $\text{tendsto-measure } M\ X\ l\ F$
unfolding *tendsto-measure-def* **using** *assms* **by** *fast*

```

lemma (in finite-measure) finite-tendsto-measureI:
  assumes [measurable]:  $\bigwedge n. f' n \in \text{borel-measurable } M \ f \in \text{borel-measurable } M$ 
  and  $\bigwedge \varepsilon. \varepsilon > 0 \implies ((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) > \varepsilon\}) \longrightarrow 0) \ F$ 
  shows tendsto-measure  $M \ f' f F$ 
proof (intro tendsto-measureI)
  fix  $\varepsilon :: \text{real}$  assume  $\varepsilon > 0$ 
  then have prob-conv:  $((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \varepsilon < \text{dist } (f' n \ \omega) (f \ \omega)\}) \longrightarrow 0) \ F$ 
  using assms by simp
  fix  $A$  assume  $A \in \text{fmeasurable } M$ 
  have  $\bigwedge n. \text{measure } M \ (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' n \ \omega) (f \ \omega)\}) \geq$ 
     $\text{measure } M \ (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' n \ \omega) (f \ \omega)\} \cap A)$ 
  by (rule finite-measure-mono; measurable)
  then show  $((\lambda n. \text{measure } M \ (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' n \ \omega) (f \ \omega)\} \cap A)) \longrightarrow 0) \ F$ 
  by (simp add: tendsto-0-le[OF prob-conv, where K=1])
qed measurable

lemma (in finite-measure) finite-tendsto-measureD:
  assumes [measurable]: tendsto-measure  $M \ f' f F$ 
  shows  $(\forall \varepsilon > 0. ((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) > \varepsilon\}) \longrightarrow 0) \ F)$ 
proof –
  from assms have  $((\lambda n. \text{measure } M \ (\{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) > \varepsilon\} \cap \text{space } M)) \longrightarrow 0) \ F$ 
  if  $\varepsilon > 0$  for  $\varepsilon$ 
  unfolding tendsto-measure-def using that fmeasurable-eq-sets by blast
  then show ?thesis
  by (simp add: sets.Int-space-eq2[symmetric, where M=M])
qed

lemma (in finite-measure) tendsto-measure-leq:
  assumes [measurable]:  $\bigwedge n. f' n \in \text{borel-measurable } M \ f \in \text{borel-measurable } M$ 
  shows tendsto-measure  $M \ f' f F \longleftrightarrow$ 
     $(\forall \varepsilon > 0. ((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) \geq \varepsilon\}) \longrightarrow 0) \ F)$ 
  (is ?L  $\longleftrightarrow$  ?R)
proof (rule iffI, goal-cases)
  case 1
  {
    fix  $\varepsilon :: \text{real}$  assume  $\varepsilon > 0$ 
    then have  $((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) > \varepsilon/2\}) \longrightarrow 0) \ F$ 
    using finite-tendsto-measureD[OF 1] half-gt-zero by blast
    then have  $((\lambda n. \text{measure } M \ \{\omega \in \text{space } M. \text{dist } (f' n \ \omega) (f \ \omega) \geq \varepsilon\}) \longrightarrow 0) \ F$ 
    apply (rule metric-tendsto-imp-tendsto)
    using  $\langle \varepsilon > 0 \rangle$  by (auto intro!: eventuallyI finite-measure-mono)
  }

```

```

    then show ?case
      by simp
next
case 2
{
  fix  $\varepsilon :: \text{real}$  assume  $\varepsilon > 0$ 
  then have *:  $((\lambda n. \mathcal{P}(\omega \text{ in } M. \varepsilon \leq \text{dist } (f' \ n \ \omega) (f \ \omega))) \longrightarrow 0) \ F$ 
    using 2 by blast
  then have  $((\lambda n. \mathcal{P}(\omega \text{ in } M. \varepsilon < \text{dist } (f' \ n \ \omega) (f \ \omega))) \longrightarrow 0) \ F$ 
    apply (rule metric-tendsto-imp-tendsto)
    using  $\langle \varepsilon > 0 \rangle$  by (auto intro!: eventuallyI finite-measure-mono)
}
then show ?case
  by (simp add: finite-tendsto-measureI[OF assms])
qed

```

abbreviation $\text{LIMSEQ-measure } M \ f \ l \equiv \text{tendsto-measure } M \ f \ l \text{ sequentially}$

lemma $\text{LIMSEQ-measure-def}$: $\text{LIMSEQ-measure } M \ f \ l \longleftrightarrow$
 $(\forall n. f \ n \in \text{borel-measurable } M) \wedge (l \in \text{borel-measurable } M) \wedge$
 $(\forall \varepsilon > 0. \forall A \in \text{fmeasurable } M.$
 $(\lambda n. \text{measure } M \ (\{\omega \in \text{space } M. \text{dist } (f \ n \ \omega) (l \ \omega) > \varepsilon\} \cap A)) \longrightarrow 0)$
unfolding $\text{tendsto-measure-def ..}$

lemma LIMSEQ-measureD :
assumes $\text{LIMSEQ-measure } M \ f \ l \ \varepsilon > 0 \ A \in \text{fmeasurable } M$
shows $(\lambda n. \text{measure } M \ (\{\omega \in \text{space } M. \text{dist } (f \ n \ \omega) (l \ \omega) > \varepsilon\} \cap A)) \longrightarrow 0$
using $\text{assms LIMSEQ-measure-def by blast}$

lemma fmeasurable-inter : $\llbracket A \in \text{sets } M; B \in \text{fmeasurable } M \rrbracket \implies A \cap B \in \text{fmeasurable } M$

proof ($\text{intro fmeasurableI, goal-cases measurable finite}$)

case measurable
 then show ?case by simp

next

case finite
 then have $\text{emeasure } M \ (A \cap B) \leq \text{emeasure } M \ B$
 by (simp add: emeasure-mono)
 also have $\text{emeasure } M \ B < \infty$
 using $\text{finite}(2)[\text{THEN fmeasurableD2}]$ by (simp add: top.not-eq-extremum)
 finally show ?case .

qed

lemma $\text{LIMSEQ-measure-emeasure}$:
assumes $\text{LIMSEQ-measure } M \ f \ l \ \varepsilon > 0 \ A \in \text{fmeasurable } M$
and $[\text{measurable}]: \bigwedge i. f \ i \in \text{borel-measurable } M \ l \in \text{borel-measurable } M$
shows $(\lambda n. \text{emeasure } M \ (\{\omega \in \text{space } M. \text{dist } (f \ n \ \omega) (l \ \omega) > \varepsilon\} \cap A)) \longrightarrow 0$
proof –
 have $\text{fmeasurable}: \{\omega \in \text{space } M. \text{dist } (f \ n \ \omega) (l \ \omega) > \varepsilon\} \cap A \in \text{fmeasurable } M$


```

for  $n$ 
  by (rule fmeasurable-inter; simp add: assms(3))
then show ?thesis
  apply (simp add: emeasure-eq-measure2 ennreal-tendsto-0-iff)
  using LIMSEQ-measure-def assms by blast
qed

lemma measure-Lim-within-LIMSEQ:
  fixes  $a :: 'a :: \text{first-countable-topology}$ 
  assumes  $\bigwedge t. X\ t \in \text{borel-measurable}\ M\ L \in \text{borel-measurable}\ M$ 
  assumes  $\bigwedge S. \llbracket (\forall n. S\ n \neq a \wedge S\ n \in T); S \longrightarrow a \rrbracket \implies \text{LIMSEQ-measure}\ M$ 
   $(\lambda n. X\ (S\ n))\ L$ 
  shows tendsto-measure  $M\ X\ L$  (at  $a$  within  $T$ )
  apply (intro tendsto-measureI[OF assms(1,2)])
  unfolding tendsto-measure-def[where  $l=L$ ] tendsto-def apply safe
  apply (rule sequentially-imp-eventually-within)
  using assms unfolding LIMSEQ-measure-def tendsto-def by presburger

definition tendsto-AE ::  $'b\ \text{measure} \Rightarrow ('a \Rightarrow 'b \Rightarrow 'c :: \text{topological-space}) \Rightarrow ('b \Rightarrow 'c) \Rightarrow 'a\ \text{filter} \Rightarrow \text{bool}$  where
   $\text{tendsto-AE}\ M\ f'\ l\ F \longleftrightarrow (AE\ \omega\ \text{in}\ M. ((\lambda n. f'\ n\ \omega) \longrightarrow l\ \omega)\ F)$ 

lemma LIMSEQ-ae-pointwise:  $(\bigwedge x. (\lambda n. f\ n\ x) \longrightarrow l\ x) \implies \text{tendsto-AE}\ M\ f\ l$ 
  sequentially
  unfolding tendsto-AE-def by simp

lemma tendsto-AE-within-LIMSEQ:
  fixes  $a :: 'a :: \text{first-countable-topology}$ 
  assumes  $\bigwedge S. \llbracket (\forall n. S\ n \neq a \wedge S\ n \in T); S \longrightarrow a \rrbracket \implies \text{tendsto-AE}\ M\ (\lambda n. X\ (S\ n))\ L$  sequentially
  shows tendsto-AE  $M\ X\ L$  (at  $a$  within  $T$ )
  oops

lemma LIMSEQ-dominated-convergence:
  fixes  $X :: \text{nat} \Rightarrow \text{real}$ 
  assumes  $X \longrightarrow L\ (\bigwedge n. Y\ n \leq X\ n)\ (\bigwedge n. Y\ n \geq L)$ 
  shows  $Y \longrightarrow L$ 
proof (rule metric-LIMSEQ-I)
  have  $X\ n \geq L$  for  $n$ 
  using assms(2,3)[of  $n$ ] by linarith
  fix  $r :: \text{real}$  assume  $0 < r$ 
  then obtain  $N$  where  $\forall n \geq N. \text{dist}\ (X\ n)\ L < r$ 
  using metric-LIMSEQ-D[OF assms(1)  $\langle 0 < r \rangle$ ] by blast
  then have  $N: \forall n \geq N. X\ n - L < r$ 
  using  $\langle \bigwedge n. L \leq X\ n \rangle$  by (auto simp: dist-real-def)
  have  $\forall n \geq N. Y\ n - L < r$ 
  proof clarify
  fix  $n$  assume  $\langle n \geq N \rangle$ 
  then have  $X\ n - L < r$ 

```

```

    using N by blast
  then show  $Y\ n - L < r$ 
    using assms(2)[of n] by auto
qed
then show  $\exists\ no. \forall\ n \geq no. \text{dist } (Y\ n)\ L < r$ 
  apply (intro exI[where x=N])
  using assms(3) dist-real-def by auto
qed

```

Klenke remark 6.4

lemma *measure-conv-imp-AE-sequentially*:

```

  assumes [measurable]:  $\bigwedge n. f'\ n \in \text{borel-measurable } M$   $f \in \text{borel-measurable } M$ 
    and tendsto-AE  $M\ f'\ f$  sequentially
  shows LIMSEQ-measure  $M\ f'\ f$ 
proof (unfold tendsto-measure-def, safe)
  fix  $\varepsilon :: \text{real}$  assume  $0 < \varepsilon$ 
  fix A assume  $A[\text{measurable}]$ :  $A \in \text{fmeasurable } M$ 

```

From AE convergence we know there's a null set where f' doesn't converge

```

  obtain N where  $N$ :  $N \in \text{null-sets } M$   $\{\omega \in \text{space } M. \neg (\lambda n. f'\ n\ \omega) \longrightarrow f\ \omega\} \subseteq N$ 
    using assms unfolding tendsto-AE-def by (simp add: eventually-ae-filter, blast)
  then have measure-0:  $\text{measure } M\ \{\omega \in \text{space } M. \neg (\lambda n. f'\ n\ \omega) \longrightarrow f\ \omega\} = 0$ 
    by (meson measure-eq-0-null-sets measure-notin-sets null-sets-subset)

```

D is a sequence of sets that converges to N

```

define D where  $D \equiv \lambda n. \{\omega \in \text{space } M. \exists m \geq n. \text{dist } (f'\ m\ \omega)\ (f\ \omega) > \varepsilon\}$ 
have  $\bigwedge n. D\ n \in \text{sets } M$ 
  unfolding D-def by measurable
then have  $[\text{measurable}]$ :  $\bigwedge n. D\ n \cap A \in \text{sets } M$ 
  by simp
have  $(\bigcap n. D\ n) \in \text{sets } M$ 
  unfolding D-def by measurable
then have measurable-D-A:  $(\bigcap n. D\ n \cap A) \in \text{sets } M$ 
  by simp
have  $(\bigcap n. D\ n) \subseteq \{\omega \in \text{space } M. \neg (\lambda n. (f'\ n\ \omega)) \longrightarrow f\ \omega\}$ 
proof (intro subsetI)
  fix x assume  $x \in (\bigcap n. D\ n)$ 
  then have  $x \in \text{space } M$   $\bigwedge n. \exists m \geq n. \varepsilon < \text{dist } (f'\ m\ x)\ (f\ x)$ 
    unfolding D-def by simp-all
  then have  $\neg (\lambda n. f'\ n\ x) \longrightarrow f\ x$ 
    by (simp add: LIMSEQ-def) (meson <0 < ε> not-less-iff-gr-or-eq order-less-le)
  then show  $x \in \{\omega \in \text{space } M. \neg (\lambda n. f'\ n\ \omega) \longrightarrow f\ \omega\}$ 
    using  $\langle x \in \text{space } M \rangle$  by blast
qed
then have measure  $M\ (\bigcap n. D\ n) = 0$ 

```

by (*metis* (*no-types*, *lifting*) $N \langle \bigcap (\text{range } D) \in \text{sets } M \rangle \text{measure-eq-0-null-sets}$
null-sets-subset subset-trans)
then have $\text{measure } M (\bigcap n. D \ n \cap A) = 0$
proof –
have $\text{emeasure } M (\bigcap n. D \ n \cap A) \leq \text{emeasure } M (\bigcap n. D \ n)$
apply (*rule emeasure-mono*)
apply *blast*
unfolding *D-def* **apply** *measurable*
done
then show *?thesis*
by (*smt* (*verit*, *del-insts*) $N \text{Sigma-Algebra.measure-def} \langle \text{measure } M (\bigcap (\text{range } D)) = 0 \rangle$
 $\langle \bigcap (\text{range } D) \in \text{sets } M \rangle \langle \bigcap (\text{range } D) \subseteq \{\omega \in \text{space } M. \neg (\lambda n. f' \ n \ \omega) \longrightarrow f \ \omega \} \rangle$
dual-order.trans enn2real-mono ennreal-zero-less-top measure-nonneg
null-setsD1 null-sets-subset)
qed
moreover have $(\lambda n. \text{measure } M (D \ n \cap A)) \longrightarrow \text{measure } M (\bigcap n. D \ n \cap A)$
apply (*rule Lim-measure-decseq*)
using *A(1)* $\langle \bigwedge n. D \ n \in \text{sets } M \rangle$ **apply** *blast*
subgoal
apply (*intro monotoneI*)
apply *clarsimp*
apply (*simp add: D-def*)
by (*meson order-trans*)
apply (*simp add: A* $\langle \bigwedge n. D \ n \in \text{sets } M \rangle$ *fmeasurableD2 fmeasurable-inter*)
done
ultimately have $\text{measure-D-0}: (\lambda n. \text{measure } M (D \ n \cap A)) \longrightarrow 0$
by *presburger*
have $\bigwedge n. \{\omega \in \text{space } M. \varepsilon < \text{dist } (f' \ n \ \omega) (f \ \omega)\} \cap A \subseteq (D \ n \cap A)$
unfolding *D-def* **by** *blast*
then have $\bigwedge n. \text{emeasure } M (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' \ n \ \omega) (f \ \omega)\} \cap A) \leq \text{emeasure } M (D \ n \cap A)$
by (*rule emeasure-mono*) *measurable*
then have $\bigwedge n. \text{measure } M (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' \ n \ \omega) (f \ \omega)\} \cap A) \leq \text{measure } M (D \ n \cap A)$
unfolding *measure-def* **apply** (*rule enn2real-mono*)
by (*meson A* $\langle \bigwedge n. D \ n \in \text{sets } M \rangle$ *fmeasurableD2 fmeasurable-inter top.not-eq-extremum*)
then show $(\lambda n. \text{measure } M (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f' \ n \ \omega) (f \ \omega)\} \cap A)) \longrightarrow 0$
by (*simp add: LIMSEQ-dominated-convergence[OF measure-D-0]*)
qed *simp-all*

corollary *LIMSEQ-measure-pointwise:*

assumes $\bigwedge x. (\lambda n. f \ n \ x) \longrightarrow f' \ x \ \bigwedge n. f \ n \in \text{borel-measurable } M \ f' \in \text{borel-measurable } M$
shows *LIMSEQ-measure M f f'*
by (*simp add: LIMSEQ-ae-pointwise measure-conv-imp-AE-sequentially assms*)

lemma *Lim-measure-pointwise*:

fixes $a :: 'a :: \text{first-countable-topology}$
assumes $\bigwedge x. ((\lambda n. f\ n\ x) \longrightarrow f'\ x) (\text{at } a \text{ within } T) \bigwedge n. f\ n \in \text{borel-measurable}$
 $M\ f' \in \text{borel-measurable } M$
shows $\text{tendsto-measure } M\ f\ f' (\text{at } a \text{ within } T)$
proof (*intro measure-Lim-within-LIMSEQ*)
fix S **assume** $\forall n. S\ n \neq a \wedge S\ n \in T\ S \longrightarrow a$
then have $(\lambda n. f\ (S\ n)\ x) \longrightarrow f'\ x$ **for** x
using *assms(1)* **by** (*simp add: tendsto-at-iff-sequentially o-def*)
then show $\text{LIMSEQ-measure } M\ (\lambda n. f\ (S\ n))\ f'$
by (*simp add: LIMSEQ-measure-pointwise assms(2,3)*)
qed (*simp-all add: assms*)

corollary *measure-conv-imp-AE-at-within*:

fixes $x :: 'a :: \text{first-countable-topology}$
assumes $[\text{measurable}]: \bigwedge n. f'\ n \in \text{borel-measurable } M\ f \in \text{borel-measurable } M$
and $\text{tendsto-AE } M\ f'\ f (\text{at } x \text{ within } S)$
shows $\text{tendsto-measure } M\ f'\ f (\text{at } x \text{ within } S)$
proof (*rule measure-Lim-within-LIMSEQ[OF assms(1,2)]*)
fix s **assume** $*: \forall n. s\ n \neq x \wedge s\ n \in S\ s \longrightarrow x$
have $\text{AE-seq}: \text{AE } \omega \text{ in } M. \forall X. (\forall i. X\ i \in S - \{x\}) \longrightarrow X \longrightarrow x \longrightarrow ((\lambda n. f'\ n\ \omega) \circ X) \longrightarrow f\ \omega$
using *assms(3)* **by** (*simp add: tendsto-AE-def tendsto-at-iff-sequentially*)
then have $\text{AE } \omega \text{ in } M. (\forall i. s\ i \in S - \{x\}) \longrightarrow s \longrightarrow x \longrightarrow ((\lambda n. f'\ n\ \omega) \circ s) \longrightarrow f\ \omega$
by force
then have $\text{AE } \omega \text{ in } M. ((\lambda n. f'\ n\ \omega) \circ s) \longrightarrow f\ \omega$
using $*$ **by force**
then have $\text{tendsto-AE } M\ (\lambda n. f'\ (s\ n))\ f \text{ sequentially}$
unfolding *tendsto-AE-def comp-def* **by blast**
then show $\text{LIMSEQ-measure } M\ (\lambda n. f'\ (s\ n))\ f$
by (*rule measure-conv-imp-AE-sequentially[OF assms(1,2)]*)
qed

Klenke remark 6.5

lemma (*in sigma-finite-measure*) *LIMSEQ-measure-unique-AE*:

fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, metric-space}\}$
assumes $[\text{measurable}]: \bigwedge n. f\ n \in \text{borel-measurable } M\ l \in \text{borel-measurable } M\ l' \in \text{borel-measurable } M$
and $\text{LIMSEQ-measure } M\ f\ l\ \text{LIMSEQ-measure } M\ f\ l'$
shows $\text{AE } x \text{ in } M. l\ x = l'\ x$

proof –

obtain $A :: \text{nat} \Rightarrow 'a \text{ set}$ **where** $A: \bigwedge i. A\ i \in \text{fmeasurable } M\ (\bigcup i. A\ i) = \text{space } M$

by (*metis fmeasurableI infinity-enreal-def rangeI sigma-finite subset-eq top.not-eq-extremum*)

have $\bigwedge m \in. \{x \in \text{space } M. \text{dist } (l\ x)\ (l'\ x) > \varepsilon\} \cap A\ m \in \text{fmeasurable } M$

by (*intro fmeasurable-inter; simp add: A*)

then have $\text{emeasure-leq}: \text{emeasure } M\ (\{x \in \text{space } M. \text{dist } (l\ x)\ (l'\ x) > \varepsilon\}) \cap A$

$m) \leq$
 $\text{emeasure } M (\{x \in \text{space } M. \text{dist } (l \ x) (f \ n \ x) > \varepsilon/2\} \cap A \ m) +$
 $\text{emeasure } M (\{x \in \text{space } M. \text{dist } (f \ n \ x) (l' \ x) > \varepsilon/2\} \cap A \ m) \text{ if } \varepsilon > 0 \text{ for } n$
 $m \ \varepsilon$
proof –
have $[\text{measurable}]$:
 $\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (l \ x) (f \ n \ x)\} \cap A \ m \in \text{sets } M$
 $\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f \ n \ x) (l' \ x)\} \cap A \ m \in \text{sets } M$
using A **by** $(\text{measurable}; \text{auto})+$
have $\text{dist } (l \ x) (l' \ x) \leq \text{dist } (l \ x) (f \ n \ x) + \text{dist } (f \ n \ x) (l' \ x)$ **for** x
by $(\text{simp add: dist-triangle})$
then have $\{x. \text{dist } (l \ x) (l' \ x) > \varepsilon\} \subseteq \{x. \text{dist } (l \ x) (f \ n \ x) > \varepsilon/2\} \cup \{x. \text{dist}$
 $(f \ n \ x) (l' \ x) > \varepsilon/2\}$
by $(\text{safe, smt (verit, best) field-sum-of-halves})$
then have $\{x \in \text{space } M. \text{dist } (l \ x) (l' \ x) > \varepsilon\} \cap A \ m \subseteq$
 $(\{x \in \text{space } M. \text{dist } (l \ x) (f \ n \ x) > \varepsilon/2\} \cap A \ m) \cup (\{x \in \text{space } M. \text{dist } (f \ n \ x)$
 $(l' \ x) > \varepsilon/2\} \cap A \ m)$
by blast
then have $\text{emeasure } M (\{x \in \text{space } M. \text{dist } (l \ x) (l' \ x) > \varepsilon\} \cap A \ m) \leq$
 $\text{emeasure } M (\{x \in \text{space } M. \text{dist } (l \ x) (f \ n \ x) > \varepsilon/2\} \cap A \ m) \cup \{x \in \text{space } M.$
 $\text{dist } (f \ n \ x) (l' \ x) > \varepsilon/2\} \cap A \ m)$
apply $(\text{rule emeasure-mono})$
using A **by** measurable
also have $\dots \leq \text{emeasure } M (\{x \in \text{space } M. \text{dist } (l \ x) (f \ n \ x) > \varepsilon/2\} \cap A \ m)$
 $+$
 $\text{emeasure } M (\{x \in \text{space } M. \text{dist } (f \ n \ x) (l' \ x) > \varepsilon/2\} \cap A \ m)$
apply $(\text{rule emeasure-subadditive})$
using A **by** measurable
finally show $?thesis$.
qed

moreover have $\text{tendsto-zero}: (\lambda n. \text{emeasure } M (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f$
 $n \ x) (l \ x)\} \cap A \ m)$
 $+ \text{emeasure } M (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f \ n \ x) (l' \ x)\} \cap A \ m)) \longrightarrow 0$
if $\langle \varepsilon > 0 \rangle$ **for** $\varepsilon \ m$
apply $(\text{rule tendsto-add-zero})$
apply $(\text{rule LIMSEQ-measure-emeasure}[OF \text{ assms}(4)])$
prefer 5 **apply** $(\text{rule LIMSEQ-measure-emeasure}[OF \text{ assms}(5)])$
using $\text{that } A$ **apply** simp-all
done
have $\text{dist-}\varepsilon\text{-emeasure}: \text{emeasure } M (\{x \in \text{space } M. \varepsilon < \text{dist } (l \ x) (l' \ x)\} \cap A \ m)$
 $= 0$
if $\langle \varepsilon > 0 \rangle$ **for** $\varepsilon \ m$
proof (rule ccontr)
assume $\text{emeasure } M (\{x \in \text{space } M. \varepsilon < \text{dist } (l \ x) (l' \ x)\} \cap A \ m) \neq 0$
then obtain e **where** $e: e > 0$ $\text{emeasure } M (\{x \in \text{space } M. \varepsilon < \text{dist } (l \ x) (l'$
 $x)\} \cap A \ m) \geq e$
using not-gr-zero **by** blast
have $\exists \text{ no. } \forall n \geq \text{no. } (\text{emeasure } M (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f \ n \ x) (l \ x)\} \cap$

$A\ m)$
 $+ \text{emeasure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ n\ x)\ (l'\ x)\} \cap A\ m)) < e$
proof –
have *measure-tendsto-0*: $(\lambda n. \text{measure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ n\ x)\ (l\ x)\} \cap A\ m))$
 $+ \text{measure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ n\ x)\ (l'\ x)\} \cap A\ m)) \longrightarrow 0$
apply (*rule tendsto-add-zero*)
using *A(1) LIMSEQ-measure-def assms(4,5) half-gt-zero that* **by** *blast+*
have *enn2real* $e > 0$
by (*metis (no-types, lifting) A(1) e(1) e(2) emeasure-neq-0-sets enn2real-eq-0-iff*)

 $\text{enn2real-nonneg } f\text{measurableD2 } f\text{measurable-inter inf-right-idem linorder-not-less}$
 $n\text{less-le top.not-eq-extremum})$
then obtain *no* **where** $\forall n \geq \text{no}. (\text{measure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ n\ x)\ (l\ x)\} \cap A\ m))$
 $+ \text{measure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ n\ x)\ (l'\ x)\} \cap A\ m)) < \text{enn2real } e$
using *LIMSEQ-D[OF measure-tendsto-0 ‹enn2real e > 0›]* **by** (*simp*) *blast*
then show *?thesis*
apply (*safe intro!*: *exI[where x=no]*)
by (*smt (verit, del-insts) A(1) Sigma-Algebra.measure-def add-eq-0-iff-both-eq-0*)

 $\text{emeasure-eq-measure2 emeasure-neq-0-sets enn2real-mono enn2real-plus}$
 enn2real-top
 $\text{ennreal-0 ennreal-zero-less-top } f\text{measurable-inter inf-sup-ord}(2) \text{ le-iff-inf}$
 $\text{linorder-not-less top.not-eq-extremum zero-less-measure-iff})$
qed
then obtain *N* **where** $N: \text{emeasure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ N\ x)\ (l\ x)\} \cap A\ m)$
 $+ \text{emeasure } M\ (\{x \in \text{space } M. \varepsilon / 2 < \text{dist } (f\ N\ x)\ (l'\ x)\} \cap A\ m) < e$
by *auto*
then have $\text{emeasure } M\ (\{x \in \text{space } M. \varepsilon < \text{dist } (l\ x)\ (l'\ x)\} \cap A\ m) < e$
by (*smt (verit, del-insts) emeasure-leq[OF that] Collect-cong dist-commute*
 $e(2) \text{ leD order-less-le-trans})$
then show *False*
using $e(2)$ **by** *auto*
qed
have $\text{emeasure } M\ (\{x \in \text{space } M. 0 < \text{dist } (l\ x)\ (l'\ x)\} \cap A\ m) = 0$ **for** *m*
proof –
have *sets*: $\text{range } (\lambda n. \{x \in \text{space } M. 1/2^n < \text{dist } (l\ x)\ (l'\ x)\} \cap A\ m) \subseteq \text{sets } M$
using *A* **by** *force*
have $(\bigcup n. \{x \in \text{space } M. 1/2^n < \text{dist } (l\ x)\ (l'\ x)\}) = \{x \in \text{space } M. 0 < \text{dist } (l\ x)\ (l'\ x)\}$
apply (*intro subset-antisym subsetI*)
apply *force*
apply *simp*
by (*metis one-less-numeral-iff power-one-over reals-power-lt-ex semiring-norm(76)*
 $\text{zero-less-dist-iff})$

moreover have $\text{emeasure } M (\{x \in \text{space } M. 1/2^n < \text{dist } (l \ x) (l' \ x)\} \cap A \ m) = 0$ **for** n
using $\text{dist-}\varepsilon\text{-emeasure by simp}$
then have $\text{suminf-zero: } (\sum n. \text{emeasure } M (\{x \in \text{space } M. 1/2^n < \text{dist } (l \ x) (l' \ x)\} \cap A \ m)) = 0$
by auto
then have $\text{emeasure } M (\bigcup n. (\{x \in \text{space } M. 1/2^n < \text{dist } (l \ x) (l' \ x)\} \cap A \ m)) \leq 0$
apply ($\text{subst suminf-zero[symmetric]}$)
apply ($\text{rule emeasure-subadditive-countably}$)
by ($\text{simp add: emeasure-subadditive-countably sets}$)
ultimately show $?thesis$
by simp
qed
then have $(\sum m. \text{emeasure } M (\{x \in \text{space } M. 0 < \text{dist } (l \ x) (l' \ x)\} \cap A \ m)) = 0$
by simp
then have $\text{emeasure } M (\bigcup m. \{x \in \text{space } M. 0 < \text{dist } (l \ x) (l' \ x)\} \cap A \ m) = 0$
proof –
let $?S = \lambda m. \{x \in \text{space } M. 0 < \text{dist } (l \ x) (l' \ x)\} \cap A \ m$
have $\text{emeasure } M (\bigcup m. ?S \ m) \leq (\sum m. \text{emeasure } M (?S \ m))$
apply ($\text{rule emeasure-subadditive-countably}$)
using $\langle \bigwedge m \ \varepsilon. \{x \in \text{space } M. \varepsilon < \text{dist } (l \ x) (l' \ x)\} \cap A \ m \in f\text{measurable } M \rangle$
by blast
then show $?thesis$
using $\langle (\sum m. \text{emeasure } M (?S \ m)) = 0 \rangle$ **by force**
qed
then have $\text{emeasure } M \{x \in \text{space } M. 0 < \text{dist } (l \ x) (l' \ x)\} = 0$
using A **by simp**
then show $?thesis$
by ($\text{auto simp add: AE-iff-null}$)
qed

corollary (**in** $\text{sigma-finite-measure}$) $\text{LIMSEQ-ae-unique-AE}$:
fixes $f :: \text{nat} \Rightarrow 'a \Rightarrow 'b :: \{\text{second-countable-topology, metric-space}\}$
assumes $\bigwedge n. f \ n \in \text{borel-measurable } M \ l \in \text{borel-measurable } M \ l' \in \text{borel-measurable } M$
and $\text{tendsto-AE } M \ f \ l \ \text{sequentially tendsto-AE } M \ f \ l' \ \text{sequentially}$
shows $\text{AE } x \text{ in } M. l \ x = l' \ x$
proof –
have $\text{LIMSEQ-measure } M \ f \ l \ \text{LIMSEQ-measure } M \ f \ l'$
using $\text{assms measure-conv-imp-AE-sequentially}$ **by blast**
then show $?thesis$
using $\text{assms}(1-3) \ \text{LIMSEQ-measure-unique-AE}$ **by blast**
qed

lemma (**in** $\text{sigma-finite-measure}$) $\text{tendsto-measure-at-within-eq-AE}$:
fixes $f :: 'b :: \{\text{first-countable-topology}\} \Rightarrow 'a \Rightarrow 'c :: \{\text{second-countable-topology, metric-space}\}$
assumes $f\text{-measurable: } \bigwedge x. x \in S \implies f \ x \in \text{borel-measurable } M$

```

    and l-measurable:  $l \in \text{borel-measurable } M$   $l' \in \text{borel-measurable } M$ 
    and tendsto:  $\text{tendsto-measure } M \text{ f } l \text{ (at } t \text{ within } S) \text{ tendsto-measure } M \text{ f } l' \text{ (at } t$ 
    within  $S$ )
    and not-bot:  $(\text{at } t \text{ within } S) \neq \perp$ 
    shows  $\text{AE } x \text{ in } M. l \ x = l' \ x$ 
  proof -
    from not-bot have  $t \text{ islimpt } S$ 
    using trivial-limit-within by blast
    then obtain  $s :: \text{nat} \Rightarrow 'b$  where  $s: \bigwedge i. s \ i \in S - \{t\} \ s \longrightarrow t$ 
    using islimpt-sequential by meson
    then have fs-measurable:  $\bigwedge n. f \ (s \ n) \in \text{borel-measurable } M$ 
    using f-measurable by blast
    have *:  $\text{LIMSEQ-measure } M \ (\lambda n. f \ (s \ n)) \ l$ 
    if  $l \in \text{borel-measurable } M \text{ tendsto-measure } M \text{ f } l \text{ (at } t \text{ within } S)$  for  $l$ 
    proof (intro tendsto-measureI[OF fs-measurable that(1)], goal-cases)
      case (1  $\varepsilon \ A$ )
      then have  $((\lambda n. \text{measure } M \ (\{\omega \in \text{space } M. \varepsilon < \text{dist } (f \ n \ \omega) \ (l \ \omega)\} \cap A))$ 
 $\longrightarrow 0)(\text{at } t \text{ within } S)$ 
      using that(2) 1 tendsto-measure-def by blast
      then show ?case
      apply (rule filterlim-compose[where  $f=s$ ])
      by (smt (verit, del-insts) DiffD1 DiffD2 eventuallyI filterlim-at insertI1 s)
    qed
    show ?thesis
    apply (rule LIMSEQ-measure-unique-AE[OF fs-measurable l-measurable])
    using * tendsto l-measurable by simp-all
  qed
end

```

5 Stochastic processes

```

theory Stochastic-Processes
  imports Kolmogorov-Chentsov-Extras Dyadic-Interval
begin

```

A stochastic process is an indexed collection of random variables. For compatibility with `product_prob_space` we don't enforce conditions on the index set I in the assumptions.

```

locale stochastic-process = prob-space +
  fixes  $M' :: 'b \text{ measure}$ 
  and  $I :: 't \text{ set}$ 
  and  $X :: 't \Rightarrow 'a \Rightarrow 'b$ 
  assumes random-process[measurable]:  $\bigwedge i. \text{random-variable } M' \ (X \ i)$ 

sublocale stochastic-process  $\subseteq$  product: product-prob-space  $(\lambda t. \text{distr } M \ M' \ (X \ t))$ 
  using prob-space-distr random-process by (blast intro: product-prob-spaceI)

lemma (in prob-space) stochastic-processI:

```


assumes $\bigwedge i. \text{random-variable } M' (X i)$
shows *stochastic-process* $M M' X$
by (*simp add: assms prob-space-axioms stochastic-process-axioms.intro stochastic-process-def*)

typedef $('t, 'a, 'b) \text{stochastic-process} =$
 $\{(M :: 'a \text{ measure}, M' :: 'b \text{ measure}, I :: 't \text{ set}, X :: 't \Rightarrow 'a \Rightarrow 'b).$
 $\text{stochastic-process } M M' X\}$

proof

show (*return (sigma UNIV {{}}, UNIV)*) $x, \text{sigma UNIV UNIV, UNIV, } \lambda - .$
 $c) \in$

$\{(M, M', I, X). \text{stochastic-process } M M' X\}$ **for** $x :: 'a$ **and** $c :: 'b$

by (*simp add: prob-space-return prob-space.stochastic-processI*)

qed

setup-lifting *type-definition-stochastic-process*

lift-definition *proc-source* $:: ('t, 'a, 'b) \text{stochastic-process} \Rightarrow 'a \text{ measure}$
is *fst* .

interpretation *proc-source: prob-space proc-source* X

by (*induction, simp add: proc-source-def Abs-stochastic-process-inverse case-prod-beta' stochastic-process-def*)

lift-definition *proc-target* $:: ('t, 'a, 'b) \text{stochastic-process} \Rightarrow 'b \text{ measure}$
is *fst* $\circ$ *snd* .

lift-definition *proc-index* $:: ('t, 'a, 'b) \text{stochastic-process} \Rightarrow 't \text{ set}$
is *fst* $\circ$ *snd* $\circ$ *snd* .

lift-definition *process* $:: ('t, 'a, 'b) \text{stochastic-process} \Rightarrow 't \Rightarrow 'a \Rightarrow 'b$
is *snd* $\circ$ *snd* $\circ$ *snd* .

declare $[[\text{coercion process}]]$

lemma *stochastic-process-construct* [*simp*]: *stochastic-process* (*proc-source* X) (*proc-target* X) (*process* X)
by (*transfer, force*)

interpretation *stochastic-process proc-source* X *proc-target* X *proc-index* X *process* X
by *simp*

lemma *stochastic-process-measurable* [*measurable*]: *process* $X t \in (\text{proc-source } X)$
 $\rightarrow_M (\text{proc-target } X)$
by (*meson random-process*)

Here we construct a process on a given index set. For this we need to produce measurable functions for indices outside the index set; we use the

constant function, but it needs to point at an element of the target set to be measurable.

context *prob-space*
begin

lift-definition *process-of* :: 'b measure $\Rightarrow$ 't set $\Rightarrow$ ('t $\Rightarrow$ 'a $\Rightarrow$ 'b) $\Rightarrow$ 'b $\Rightarrow$ ('t, 'a, 'b) stochastic-process

is $\lambda M' I X \omega.$ if ($\forall t \in I. X t \in M \rightarrow_M M'$) $\wedge \omega \in \text{space } M'$
 then ($M, M', I, (\lambda t. \text{if } t \in I \text{ then } X t \text{ else } (\lambda -. \omega))$)
 else (return (sigma UNIV { }, UNIV) (SOME x. True), sigma UNIV UNIV, I, $\lambda -. \omega$)
by (simp add: stochastic-processI prob-space-return prob-space.stochastic-processI)

lemma *index-process-of*[simp]: *proc-index* (process-of $M' I X \omega$) = I
by (transfer, auto)

lemma

assumes $\forall t \in I. X t \in M \rightarrow_M M' \omega \in \text{space } M'$

shows

source-process-of[simp]: *proc-source* (process-of $M' I X \omega$) = M **and**

target-process-of[simp]: *proc-target* (process-of $M' I X \omega$) = M' **and**

process-process-of[simp]: *process* (process-of $M' I X \omega$) = ($\lambda t. \text{if } t \in I \text{ then } X t \text{ else } (\lambda -. \omega)$)

using *assms* **by** (transfer, auto)+

lemma *process-of-apply*:

assumes $\forall t \in I. X t \in M \rightarrow_M M' \omega \in \text{space } M' t \in I$

shows *process* (process-of $M' I X \omega$) $t = X t$

using *assms* **by** (meson process-process-of)

end

We define the finite-dimensional distributions of our process.

lift-definition *distributions* :: ('t, 'a, 'b) stochastic-process $\Rightarrow$ 't set $\Rightarrow$ ('t $\Rightarrow$ 'b) measure

is $\lambda(M, M', -, X) T. (\Pi_M t \in T. \text{distr } M M' (X t))$.

lemma *distributions-altdef*: *distributions* $X T = (\Pi_M t \in T. \text{distr } (\text{proc-source } X) (\text{proc-target } X) (X t))$

by (transfer, auto)

lemma *prob-space-distributions*: *prob-space* (distributions $X J$)

unfolding *distributions-altdef*

by (simp add: prob-space-PiM proc-source.prob-space-distr random-process)

lemma *sets-distributions*: *sets* (distributions $X J$) = *sets* (PiM $J (\lambda -. (\text{proc-target } X))$)

by (transfer, auto cong: sets-PiM-cong)

lemma *space-distributions*: *space* (distributions $X J$) = $(\Pi_E i \in J. \text{space } (\text{proc-target } X))$

$X))$
by (*transfer, auto simp add: space-PiM*)

lemma *emeasure-distributions*:
assumes *finite J* $\bigwedge j. j \in J \implies A\ j \in \text{sets } (\text{proc-target } X)$
shows $\text{emeasure } (\text{distributions } X\ J) (Pi_E\ J\ A) = (\prod_{j \in J}. \text{emeasure } (\text{distr } (\text{proc-source } X) (\text{proc-target } X) (X\ j)) (A\ j))$
by (*simp add: assms(1) assms(2) distributions-altdef product.emeasure-PiM*)

interpretation *projective-family* (*proc-index X*) *distributions X* ($\lambda-. \text{proc-target } X$)
proof (*intro projective-family.intro*)
fix *J and H*
let $?I = \text{proc-index } X$
and $?M = \text{proc-source } X$
and $?M' = \text{proc-target } X$
assume $*$: $J \subseteq H$ *finite H* $H \subseteq ?I$
then have $J \subseteq ?I$
by *simp*
show $\text{distributions } X\ J = \text{distr } (\text{distributions } X\ H) (Pi_M\ J\ (\lambda-. ?M')) (\lambda f. \text{restrict } f\ J)$
proof (*rule measure-eqI*)
show $\text{sets } (\text{distributions } X\ J) = \text{sets } (\text{distr } (\text{distributions } X\ H) (Pi_M\ J\ (\lambda-. ?M')) (\lambda f. \text{restrict } f\ J))$
by (*simp add: sets-distributions*)
fix *S* **assume** $S \in \text{sets } (\text{distributions } X\ J)$
then have *in-sets*: $S \in \text{sets } (Pi_M\ J\ (\lambda-. ?M'))$
by (*simp add: sets-distributions*)
have *prod-emb-distr*: $(\text{prod-emb } H\ (\lambda-. ?M')\ J\ S) = (\text{prod-emb } H\ (\lambda t. \text{distr } ?M\ ?M'\ (X\ t))\ J\ S)$
by (*simp add: prod-emb-def*)
have $\text{emeasure } (\text{distr } (\text{distributions } X\ H) (Pi_M\ J\ (\lambda-. ?M')) (\lambda f. \text{restrict } f\ J))\ S =$
 $\text{emeasure } (\text{distributions } X\ H) (\text{prod-emb } H\ (\lambda-. ?M')\ J\ S)$
apply (*rule emeasure-distr-restrict*)
by (*simp-all add: * sets-distributions in-sets*)
also have $\dots = \text{emeasure } (\text{distributions } X\ J)\ S$
unfolding *distributions-altdef*
using $*(1,2)$ *in-sets prod-emb-distr* **by** *force*
finally show $\text{emeasure } (\text{distributions } X\ J)\ S$
 $= \text{emeasure } (\text{distr } (\text{distributions } X\ H) (Pi_M\ J\ (\lambda-. ?M')) (\lambda f. \text{restrict } f\ J))\ S$
by *argo*
qed
qed (*rule prob-space-distributions*)

locale *polish-stochastic* = *stochastic-process M borel :: 'b::polish-space measure I X*
for *M and I and X*

lemma *distributed-cong-random-variable*:

assumes $M = K \ N = L \ AE \ x \text{ in } M. \ X \ x = Y \ x \ X \in M \rightarrow_M \ N \ Y \in K \rightarrow_M \ L$
 $f \in \text{borel-measurable } N$
shows $\text{distributed } M \ N \ X \ f \longleftrightarrow \text{distributed } K \ L \ Y \ f$
using *assms* **by** (*auto simp add: distributed-def distr-cong-AE*)

For all sorted lists of indices, the increments specified by this list are independent

lift-definition *indep-increments* :: $('t :: \text{linorder}, 'a, 'b :: \text{minus}) \text{ stochastic-process} \Rightarrow \text{bool}$ **is**

$\lambda(M, M', I, X).$
 $(\forall l. \text{set } l \subseteq I \wedge \text{sorted } l \wedge \text{length } l \geq 2 \longrightarrow$
 $\text{prob-space.indep-vars } M \ (\lambda-. M') \ (\lambda k \ v. X \ (!k) \ v - X \ (!!(k-1)) \ v) \ \{1..<\text{length } l\})$
 $\left. \right\} .$

lemma *indep-incrementsE*:

assumes *indep-increments* X
and $\text{set } l \subseteq \text{proc-index } X \wedge \text{sorted } l \wedge \text{length } l \geq 2$
shows $\text{prob-space.indep-vars } (\text{proc-source } X) \ (\lambda-. \text{proc-target } X)$
 $(\lambda k \ v. X \ (!k) \ v - X \ (!!(k-1)) \ v) \ \{1..<\text{length } l\}$
using *assms* **by** (*transfer, auto*)

lemma *indep-incrementsI*:

assumes $\bigwedge l. \text{set } l \subseteq \text{proc-index } X \Longrightarrow \text{sorted } l \Longrightarrow \text{length } l \geq 2 \Longrightarrow$
 $\text{prob-space.indep-vars } (\text{proc-source } X) \ (\lambda-. \text{proc-target } X) \ (\lambda k \ v. X \ (!k) \ v - X$
 $(!!(k-1)) \ v) \ \{1..<\text{length } l\}$
shows *indep-increments* X
using *assms* **by** (*transfer, auto*)

lemma *indep-increments-indep-var*:

assumes *indep-increments* $X \ h \in \text{proc-index } X \ j \in \text{proc-index } X \ k \in \text{proc-index } X$
 $h \leq j \ j \leq k$
shows $\text{prob-space.indep-var } (\text{proc-source } X) \ (\text{proc-target } X) \ (\lambda v. X \ j \ v - X \ h \ v)$
 $(\text{proc-target } X) \ (\lambda v. X \ k \ v - X \ j \ v)$

proof –

let $?l = [h, j, k]$
have $\text{set } ?l \subseteq \text{proc-index } X \wedge \text{sorted } ?l \wedge 2 \leq \text{length } ?l$
using *assms* **by** *auto*
then have $\text{prob-space.indep-vars } (\text{proc-source } X) \ (\lambda-. \text{proc-target } X) \ (\lambda k \ v. X$
 $(?l k) \ v - X \ (?l(k-1)) \ v) \ \{1..<\text{length } ?l\}$
by (*rule indep-incrementsE[OF assms(1)]*)
then show *?thesis*
using *proc-source.indep-vars-indep-var* **by** *fastforce*
qed

definition *stationary-increments* $X \longleftrightarrow (\forall t1 \ t2 \ k. t1 > 0 \wedge t2 > 0 \wedge k > 0 \longrightarrow$

$$\begin{aligned} & \text{distr } (\text{proc-source } X) (\text{proc-target } X) (\lambda v. X (t1 + k) v - X t1 v) = \\ & \text{distr } (\text{proc-source } X) (\text{proc-target } X) (\lambda v. X (t2 + k) v - X t2 v) \end{aligned}$$

Processes on the same source measure space, with the same index space, but not necessarily the same target measure since we only care about the measurable target space, not the measure

lift-definition *compatible* :: ('t,'a,'b) stochastic-process $\Rightarrow$ ('t,'a,'b) stochastic-process $\Rightarrow$ bool

is $\lambda(Mx, M'x, Ix, X) (My, M'y, Iy, -). Mx = My \wedge \text{sets } M'x = \text{sets } M'y \wedge Ix = Iy$.

lemma *compatibleI*:

assumes *proc-source* $X = \text{proc-source } Y$ *sets* (*proc-target* X) = *sets* (*proc-target* Y)

proc-index $X = \text{proc-index } Y$

shows *compatible* $X Y$

using *assms* **by** (*transfer*, *auto*)

lemma

assumes *compatible* $X Y$

shows

compatible-source [*dest*]: *proc-source* $X = \text{proc-source } Y$ **and**

compatible-target [*dest*]: *sets* (*proc-target* X) = *sets* (*proc-target* Y) **and**

compatible-index [*dest*]: *proc-index* $X = \text{proc-index } Y$

using *assms* **by** (*transfer*, *auto*)+

lemma *compatible-refl* [*simp*]: *compatible* $X X$

by (*transfer*, *auto*)

lemma *compatible-sym*: *compatible* $X Y \Longrightarrow \text{compatible } Y X$

by (*transfer*, *auto*)

lemma *compatible-trans*:

assumes *compatible* $X Y$ *compatible* $Y Z$

shows *compatible* $X Z$

using *assms* **by** (*transfer*, *auto*)

lemma (**in** *prob-space*) *compatible-process-of*:

assumes *measurable*: $\forall t \in I. X t \in M \rightarrow_M M' \forall t \in I. Y t \in M \rightarrow_M M'$

and $a \in \text{space } M' b \in \text{space } M'$

shows *compatible* (*process-of* $M' I X a$) (*process-of* $M' I Y b$)

using *assms* **by** (*transfer*, *auto*)

definition *modification* :: ('t,'a,'b) stochastic-process $\Rightarrow$ ('t,'a,'b) stochastic-process

$\Rightarrow$ bool **where**

modification $X Y \longleftrightarrow \text{compatible } X Y \wedge (\forall t \in \text{proc-index } X. \text{AE } x \text{ in } \text{proc-source } X. X t x = Y t x)$

lemma *modificationI* [*intro*]:

assumes *compatible* $X\ Y\ \bigwedge t. t \in \text{proc-index } X \implies AE\ x\ \text{in}\ \text{proc-source } X. X\ t\ x = Y\ t\ x$
shows *modification* $X\ Y$
unfolding *modification-def* **using** *assms* **by** *blast*

lemma *modificationD* [*dest*]:
assumes *modification* $X\ Y$
shows *compatible* $X\ Y$
and $\bigwedge t. t \in \text{proc-index } X \implies AE\ x\ \text{in}\ \text{proc-source } X. X\ t\ x = Y\ t\ x$
using *assms* **unfolding** *modification-def* **by** *blast*

lemma *modification-null-set*:
assumes *modification* $X\ Y\ t \in \text{proc-index } X$
obtains N **where** $\{x \in \text{space } (\text{proc-source } X). X\ t\ x \neq Y\ t\ x\} \subseteq N\ N \in \text{null-sets } (\text{proc-source } X)$
proof –
from *assms* **have** $AE\ x\ \text{in}\ \text{proc-source } X. X\ t\ x = Y\ t\ x$
by (*rule* *modificationD*(2))
then **have** $\exists N \in \text{null-sets } (\text{proc-source } X). \{x \in \text{space } (\text{proc-source } X). X\ t\ x \neq Y\ t\ x\} \subseteq N$
by (*simp* *add: eventually-ae-filter*)
then **show** *?thesis*
using *that* **by** *blast*
qed

lemma *modification-refl* [*simp*]: *modification* $X\ X$
by (*simp* *add: modificationI*)

lemma *modification-sym*: *modification* $X\ Y \implies \text{modification } Y\ X$
proof (*rule* *modificationI*)
assume *: *modification* $X\ Y$
then **show** *compat: compatible* $Y\ X$
using *compatible-sym* *modificationD*(1) **by** *blast*
fix t **assume** $t \in \text{proc-index } Y$
then **have** $t \in \text{proc-index } X$
using *compatible-index*[*OF* *compat*] **by** *blast*
have $AE\ x\ \text{in}\ \text{proc-source } Y. X\ t\ x = Y\ t\ x$
using *modificationD*(2)[*OF* * $\langle t \in \text{proc-index } X \rangle$]
compatible-source[*OF* *compat*] **by** *argo*
then **show** $AE\ x\ \text{in}\ \text{proc-source } Y. Y\ t\ x = X\ t\ x$
by *force*
qed

lemma *modification-trans*:
assumes *modification* $X\ Y\ \text{modification } Y\ Z$
shows *modification* $X\ Z$
proof (*intro* *modificationI*)
show *compatible* $X\ Z$
using *compatible-trans* *modificationD*(1) *assms* **by** *blast*

fix t **assume** $t: t \in \text{proc-index } X$
have $XY: AE\ x \text{ in } \text{proc-source } X. \text{ process } X\ t\ x = \text{ process } Y\ t\ x$
by (*fact* $\text{modificationD}(2)[OF\ \text{assms}(1)\ t]$)
have $t \in \text{proc-index } Y\ \text{proc-source } X = \text{proc-source } Y$
using *compatible-index compatible-source assms(1) modificationD(1) t* **by**
blast+
then have $AE\ x \text{ in } \text{proc-source } X. \text{ process } Y\ t\ x = \text{ process } Z\ t\ x$
using $\text{modificationD}(2)[OF\ \text{assms}(2)]$ **by** *presburger*
then show $AE\ x \text{ in } \text{proc-source } X. \text{ process } X\ t\ x = \text{ process } Z\ t\ x$
using XY **by** *fastforce*
qed

lemma *modification-imp-identical-distributions:*
assumes *modification: modification X Y*
and *index: $T \subseteq \text{proc-index } X$*
shows *distributions X T = distributions Y T*
proof –
have $\text{proc-source } X = \text{proc-source } Y$
using *modification* **by** *blast*
moreover have $\text{sets } (\text{proc-target } X) = \text{sets } (\text{proc-target } Y)$
using *modification* **by** *blast*
ultimately have $\text{distr } (\text{proc-source } X) (\text{proc-target } X) (X\ x) =$
 $\text{distr } (\text{proc-source } Y) (\text{proc-target } Y) (Y\ x)$
if $x \in T$ **for** x
apply (*rule distr-cong-AE*)
apply (*metis assms modificationD(2) subset-eq that*)
apply *simp-all*
done
then show *?thesis*
by (*auto simp: distributions-altdef cong: PiM-cong*)
qed

definition *indistinguishable* $:: ('t, 'a, 'b) \text{ stochastic-process} \Rightarrow ('t, 'a, 'b) \text{ stochastic-process}$
 $\Rightarrow \text{bool}$ **where**
 $\text{indistinguishable } X\ Y \longleftrightarrow \text{compatible } X\ Y \wedge$
 $(\exists N \in \text{null-sets } (\text{proc-source } X). \forall t \in \text{proc-index } X. \{x \in \text{space } (\text{proc-source } X). \\ X\ t\ x \neq Y\ t\ x\} \subseteq N)$

lemma *indistinguishableI:*
assumes *compatible X Y*
and $\exists N \in \text{null-sets } (\text{proc-source } X). (\forall t \in \text{proc-index } X. \{x \in \text{space } (\text{proc-source } X). \\ X\ t\ x \neq Y\ t\ x\} \subseteq N)$
shows *indistinguishable X Y*
unfolding *indistinguishable-def* **using** *assms* **by** *blast*

lemma *indistinguishable-null-set:*
assumes *indistinguishable X Y*
obtains N **where**
 $N \in \text{null-sets } (\text{proc-source } X)$

$\bigwedge t. t \in \text{proc-index } X \implies \{x \in \text{space } (\text{proc-source } X). X \ t \ x \neq Y \ t \ x\} \subseteq N$
using *assms* **unfolding** *indistinguishable-def* **by** *force*

lemma *indistinguishableD*:
assumes *indistinguishable* *X Y*
shows *compatible* *X Y*
and $\exists N \in \text{null-sets } (\text{proc-source } X). (\forall t \in \text{proc-index } X. \{x \in \text{space } (\text{proc-source } X). X \ t \ x \neq Y \ t \ x\} \subseteq N)$
using *assms* **unfolding** *indistinguishable-def* **by** *blast+*

lemma *indistinguishable-eq-AE*:
assumes *indistinguishable* *X Y*
shows *AE* *x in proc-source X. $\forall t \in \text{proc-index } X. X \ t \ x = Y \ t \ x$*
using *assms*[*THEN indistinguishableD(2)*] **by** (*auto simp add: eventually-ae-filter*)

lemma *indistinguishable-null-ex*:
assumes *indistinguishable* *X Y*
shows $\exists N \in \text{null-sets } (\text{proc-source } X). \{x \in \text{space } (\text{proc-source } X). \exists t \in \text{proc-index } X. X \ t \ x \neq Y \ t \ x\} \subseteq N$
using *indistinguishableD(2)*[*OF assms*] **by** *blast*

lemma *indistinguishable-reft [simp]*: *indistinguishable* *X X*
by (*auto intro: indistinguishableI*)

lemma *indistinguishable-sym*: *indistinguishable* *X Y* $\implies$ *indistinguishable* *Y X*
unfolding *indistinguishable-def* **apply** (*simp add: compatible-sym*)
by (*smt (verit, ccfv-SIG) Collect-cong compatible-index compatible-source indistinguishable-def*)

lemma *indistinguishable-trans*:
assumes *indistinguishable* *X Y* *indistinguishable* *Y Z*
shows *indistinguishable* *X Z*
proof (*intro indistinguishableI*)
show *compatible* *X Z*
using *assms indistinguishableD(1) compatible-trans* **by** *blast*
have *eq: proc-index X = proc-index Y proc-source X = proc-source Y*
using *compatible-index compatible-source indistinguishableD(1)*[*OF assms(1)*]
by *blast+*
have *AE x in proc-source X. $\forall t \in \text{proc-index } X. X \ t \ x = Y \ t \ x$*
by (*fact indistinguishable-eq-AE*[*OF assms(1)*])
moreover have *AE x in proc-source X. $\forall t \in \text{proc-index } X. Y \ t \ x = Z \ t \ x$*
apply (*subst eq*)
by (*fact indistinguishable-eq-AE*[*OF assms(2)*])
ultimately have *AE x in proc-source X. $\forall t \in \text{proc-index } X. X \ t \ x = Z \ t \ x$*
using *assms* **by** *fastforce*
then obtain *N where* $N \in \text{null-sets } (\text{proc-source } X)$
 $\{x \in \text{space } (\text{proc-source } X). \exists t \in \text{proc-index } X. \text{process } X \ t \ x \neq \text{process } Z \ t \ x\} \subseteq N$
using *eventually-ae-filter* **by** (*smt (verit) Collect-cong eventually-ae-filter*)

then show $\exists N \in \text{null-sets}(\text{proc-source } X). \forall t \in \text{proc-index } X. \{x \in \text{space}(\text{proc-source } X). \text{process } X \ t \ x \neq \text{process } Z \ t \ x\} \subseteq N$
by *blast*
qed

lemma *indistinguishable-modification*: $\text{indistinguishable } X \ Y \implies \text{modification } X \ Y$
apply (*intro modificationI*)
apply (*erule indistinguishableD(1)*)
apply (*drule indistinguishableD(2)*)
using *eventually-ae-filter* **by** *blast*

Klenke 21.5(i)

lemma *modification-countable*:
assumes *modification* $X \ Y$ *countable* (*proc-index* X)
shows *indistinguishable* $X \ Y$
proof (*rule indistinguishableI*)
show *compatible* $X \ Y$
using *assms(1)* *modification-def* **by** *auto*
let $?N = (\lambda t. \{x \in \text{space}(\text{proc-source } X). X \ t \ x \neq Y \ t \ x\})$
from *assms(1)* **have** $\forall t \in \text{proc-index } X. \text{AE } x \text{ in } \text{proc-source } X. X \ t \ x = Y \ t \ x$
unfolding *modification-def* **by** *argo*
then have $\bigwedge t. t \in \text{proc-index } X \implies \exists N \in \text{null-sets}(\text{proc-source } X). ?N \ t \subseteq N$
by (*subst eventually-ae-filter[symmetric]*, *blast*)
then have $\exists N. \forall t \in \text{proc-index } X. N \ t \in \text{null-sets}(\text{proc-source } X) \wedge ?N \ t \subseteq N$
by *meson*
then obtain N **where** $N: \forall t \in \text{proc-index } X. (N \ t) \in \text{null-sets}(\text{proc-source } X)$
 $\wedge ?N \ t \subseteq N \ t$
by *blast*
then have *null*: $(\bigcup t \in \text{proc-index } X. N \ t) \in \text{null-sets}(\text{proc-source } X)$
by (*simp add: null-sets-UN'* *assms(2)*)
moreover have $\forall t \in \text{proc-index } X. ?N \ t \subseteq (\bigcup t \in \text{proc-index } X. N \ t)$
using N **by** *blast*
ultimately show $\exists N \in \text{null-sets}(\text{proc-source } X). (\forall t \in \text{proc-index } X. ?N \ t \subseteq N)$
by *blast*
qed

Klenke 21.5(ii). The textbook statement is more general - we reduce right continuity to regular continuity

lemma *modification-continuous-indistinguishable*:
fixes $X :: (\text{real}, 'a, 'b :: \text{metric-space}) \text{stochastic-process}$
assumes *modification*: *modification* $X \ Y$
and *interval*: $\exists T > 0. \text{proc-index } X = \{0..T\}$
and *rc*: *AE* ω *in* *proc-source* $X. \text{continuous-on}(\text{proc-index } X) (\lambda t. X \ t \ \omega)$
(is *AE* ω *in* *proc-source* $X. ?\text{cont-}X \ \omega)$
AE ω *in* *proc-source* $Y. \text{continuous-on}(\text{proc-index } Y) (\lambda t. Y \ t \ \omega)$
(is *AE* ω *in* *proc-source* $Y. ?\text{cont-}Y \ \omega)$
shows *indistinguishable* $X \ Y$

```

proof (rule indistinguishableI)
  show compatible X Y
    using modification modification-def by blast
  obtain T where T: proc-index X = {0..T} T > 0
    using interval by blast
  define N where N  $\equiv \lambda t. \{x \in \text{space } (\text{proc-source } X). X\ t\ x \neq Y\ t\ x\}$ 
  have 1:  $\forall t \in \text{proc-index } X. \exists S. N\ t \subseteq S \wedge S \in \text{null-sets } (\text{proc-source } X)$ 
    using modificationD(2)[OF modification] by (auto simp add: N-def eventually-ae-filter)

```

S is a null set such that $X_t(x) \neq Y_t(x) \implies x \in S_t$

```

  obtain S where S:  $\forall t \in \text{proc-index } X. N\ t \subseteq S\ t \wedge S\ t \in \text{null-sets } (\text{proc-source } X)$ 
    using bchoice[OF 1] by blast
  have eq:  $\text{proc-source } X = \text{proc-source } Y$   $\text{proc-index } X = \text{proc-index } Y$ 
    using <compatible X Y> compatible-source compatible-index by blast+
  have AE p in proc-source X. ?cont-X p  $\wedge$  ?cont-Y p
    apply (rule AE-conjI)
    using eq rc by argo+

```

R is a set of measure 1 such that if $x \in R$ then the paths at x are continuous for X and Y

```

  then obtain R where R:  $R \subseteq \{p \in \text{space } (\text{proc-source } X). ?\text{cont-X } p \wedge ?\text{cont-Y } p\}$ 
     $R \in \text{sets } (\text{proc-source } X)$   $\text{measure } (\text{proc-source } X)\ R = 1$ 
    using proc-source.AE-E-prob by blast

```

We use an interval of dyadic rationals because we need to produce a countable dense set for $\{0..T\}$, which we have by $\text{closure } (\text{dyadic-interval } 0\ ?T) = \{0..?T\}$.

```

  let ?I = dyadic-interval 0 T
  let ?N' =  $\bigcup n \in ?I. N\ n$ 
  have N-subset:  $\bigwedge t. t \in \text{proc-index } X \implies N\ t \cap R \subseteq ?N'$ 
  proof
    fix t assume t  $\in \text{proc-index } X$ 
    fix p assume *: p  $\in N\ t \cap R$ 
    then obtain  $\varepsilon$  where  $0 < \varepsilon$   $\varepsilon = \text{dist } (X\ t\ p)\ (Y\ t\ p)$ 
      by (simp add: N-def)
    have cont-p: continuous-on {0..T}  $(\lambda t. Y\ t\ p)$  continuous-on {0..T}  $(\lambda t. X\ t\ p)$ 
      using R *(1) T(1)[symmetric] eq(2) by auto
    then have continuous-dist: continuous-on {0..T}  $(\lambda t. \text{dist } (X\ t\ p)\ (Y\ t\ p))$ 
      using continuous-on-dist by fast
    {
      assume  $\forall r \in ?I. X\ r\ p = Y\ r\ p$ 
      then have dist-0:  $\bigwedge r. r \in ?I \implies \text{dist } (X\ r\ p)\ (Y\ r\ p) = 0$ 
        by auto
      have dist  $(X\ t\ p)\ (Y\ t\ p) = 0$ 

```

```

proof –
  have dist-tendsto-0:  $((\lambda t. \text{dist } (X \ t \ p) \ (Y \ t \ p)) \longrightarrow 0)(\text{at } t \text{ within } ?I)$ 
    using dist-0 continuous-dist
    by (smt (verit, best) Lim-transform-within  $\varepsilon$  tendsto-const)
  have XY:  $((\lambda t. X \ t \ p) \longrightarrow X \ t \ p)(\text{at } t \text{ within } ?I) ((\lambda t. Y \ t \ p) \longrightarrow Y \ t \ p)(\text{at } t \text{ within } ?I)$ 
    by (metis cont-p T(1)  $\langle t \in \text{proc-index } X \rangle$  continuous-on-def tendsto-within-subset dyadic-interval-subset-interval) +
  show ?thesis
    apply (rule tendsto-unique[of at t within ?I])
    apply (simp add: trivial-limit-within)
    apply (metis T(1) T(2)  $\langle t \in \text{proc-index } X \rangle$  dyadic-interval-dense islimpt-Icc limpt-of-closure)
    using tendsto-dist[OF XY] dist-tendsto-0
    by simp-all
  qed
  then have False
    using  $\varepsilon$  by force
}
then have  $\exists r \in \text{dyadic-interval } 0 \ T. p \in N \ r$ 
  unfolding N-def using  $\ast R(2)$  sets.sets-into-space by auto
then show  $p \in \bigcup (N \ ^{?I})$ 
  by simp
qed
have null:  $(\text{space } (\text{proc-source } X) - R) \cup (\bigcup r \in ?I. S \ r) \in \text{null-sets } (\text{proc-source } X)$ 
  apply (rule null-sets.Un)
  apply (smt (verit) R(2,3) AE-iff-null-sets proc-source.prob-compl proc-source.prob-eq-0 sets.Diff sets.top)
  by (metis (no-types, lifting) S T(1) dyadic-interval-countable dyadic-interval-subset-interval in-mono null-sets-UN')
  have  $(\bigcup r \in \text{proc-index } X. N \ r) \subseteq (\text{space } (\text{proc-source } X) - R) \cup (\bigcup r \in \text{proc-index } X. N \ r)$ 
    by blast
  also have  $\dots \subseteq (\text{space } (\text{proc-source } X) - R) \cup (\bigcup r \in ?I. N \ r)$ 
    using N-subset N-def by blast
  also have  $\dots \subseteq (\text{space } (\text{proc-source } X) - R) \cup (\bigcup r \in ?I. S \ r)$ 
    by (smt (verit, ccfv-threshold) S T(1) UN-iff Un-iff dyadic-interval-subset-interval in-mono subsetI)
  finally show  $\exists N \in \text{null-sets } (\text{proc-source } X). \forall t \in \text{proc-index } X. \{x \in \text{space } (\text{proc-source } X). \text{process } X \ t \ x \neq \text{process } Y \ t \ x\} \subseteq N$ 
    by (smt (verit) N-def null S SUP-le-iff order-trans)
qed

```

lift-definition *restrict-index* :: $(t, 'a, 'b) \text{ stochastic-process} \Rightarrow 't \text{ set} \Rightarrow (t, 'a, 'b) \text{ stochastic-process}$
is $\lambda(M, M', I, X) \ T. (M, M', T, X) \text{ by fast}$

lemma

shows
 $\text{restrict-index-source}[simp]: \text{proc-source } (\text{restrict-index } X \ T) = \text{proc-source } X$
and
 $\text{restrict-index-target}[simp]: \text{proc-target } (\text{restrict-index } X \ T) = \text{proc-target } X$ **and**
 $\text{restrict-index-index}[simp]: \text{proc-index } (\text{restrict-index } X \ T) = T$ **and**
 $\text{restrict-index-process}[simp]: \text{process } (\text{restrict-index } X \ T) = \text{process } X$
by $(\text{transfer}, \text{force})+$

lemma $\text{restrict-index-override}[simp]: \text{restrict-index } (\text{restrict-index } X \ T) \ S = \text{restrict-index } X \ S$
by $(\text{transfer}, \text{auto})$

lemma $\text{compatible-restrict-index}$:
assumes $\text{compatible } X \ Y$
shows $\text{compatible } (\text{restrict-index } X \ S) \ (\text{restrict-index } Y \ S)$
using assms **unfolding** compatible-def **by** $(\text{transfer}, \text{auto})$

lemma $\text{modification-restrict-index}$:
assumes $\text{modification } X \ Y \ S \subseteq \text{proc-index } X$
shows $\text{modification } (\text{restrict-index } X \ S) \ (\text{restrict-index } Y \ S)$
using assms **unfolding** modification-def
apply $(\text{simp add: compatible-restrict-index})$
apply $(\text{metis restrict-index-source subsetD})$
done

lemma $\text{indistinguishable-restrict-index}$:
assumes $\text{indistinguishable } X \ Y \ S \subseteq \text{proc-index } X$
shows $\text{indistinguishable } (\text{restrict-index } X \ S) \ (\text{restrict-index } Y \ S)$
using assms **unfolding** $\text{indistinguishable-def}$ **by** $(\text{auto simp: compatible-restrict-index})$

lemma AE-eq-minus $[\text{intro}]$:
fixes $a :: 'a \Rightarrow ('b :: \text{real-normed-vector})$
assumes $\text{AE } x \text{ in } M. a \ x = b \ x \ \text{AE } x \text{ in } M. c \ x = d \ x$
shows $\text{AE } x \text{ in } M. a \ x - c \ x = b \ x - d \ x$
using assms **by** fastforce

lemma $\text{modification-indep-increments}$:
fixes $X \ Y :: ('a :: \text{linorder}, 'b, 'c :: \{\text{second-countable-topology}, \text{real-normed-vector}\})$
 $\text{stochastic-process}$
assumes $\text{modification } X \ Y \ \text{sets } (\text{proc-target } Y) = \text{sets borel}$
shows $\text{indep-increments } X \Longrightarrow \text{indep-increments } Y$
proof $(\text{intro indep-incrementsI}, \text{subst proc-source.indep-vars-iff-distr-eq-PiM}, \text{goal-cases})$
case $(1 \ l)$
then show $?case$ **by** simp
next
case $(2 \ l \ i)$
then show $?case$
using assms **apply** measurable
using $\text{modificationD}(1)[\text{OF assms}(1), \text{THEN compatible-source}] \ \text{assms}(2)$

```

    by (metis measurable-cong-sets random-process)+
next
  case (3 l)
  have target-X [measurable]: sets (proc-target X) = sets borel
    using assms by auto
  then have measurable-target:  $f \in M \rightarrow_M \text{proc-target } X = \{f \in \text{borel-measurable } M\}$  for  $f$  and  $M :: 'b \text{ measure}$ 
    using measurable-cong-sets by blast
  have AE  $\omega$  in proc-source X.  $X(l! i) \omega = Y(l! i) \omega$ 
    if  $i \in \{0..<\text{length } l\}$  for  $i$ 
    apply (rule assms(1)[THEN modificationD(2)])
    by (metis 3(2) that assms(1) atLeastLessThan-iff basic-trans-rules(31)
        compatible-index modificationD(1) nth-mem)
  then have AE-eq: AE  $\omega$  in proc-source X.  $X(l! i) \omega - X(l!(i-1)) \omega = Y(l! i)$ 
 $\omega - Y(l!(i-1)) \omega$ 
    if  $i \in \{1..<\text{length } l\}$  for  $i$ 
    using AE-eq-minus that by auto
  have AE-eq': AE  $x$  in proc-source X.  $(\lambda i \in \{1..<\text{length } l\}. X(l! i) x - X(l!(i-1)) x =$ 
 $(\lambda i \in \{1..<\text{length } l\}. Y(l! i) x - Y(l!(i-1)) x)$ 
    proof (rule AE-mp)
      show AE  $\omega$  in proc-source X.  $\forall i \in \{1..<\text{length } l\}. X(l! i) \omega - X(l!(i-1)) \omega = Y(l! i) \omega - Y(l!(i-1)) \omega$ 
        proof -
          {
            fix i assume *:  $i \in \{1..<\text{length } l\}$ 
            obtain N where
               $\{\omega \in \text{space } (\text{proc-source } X). X(l! i) \omega - X(l!(i-1)) \omega \neq Y(l! i) \omega - Y(l!(i-1)) \omega\} \subseteq N$ 
               $N \in \text{proc-source } X \text{ emeasure } (\text{proc-source } X) N = 0$ 
              using AE-eq[OF *, THEN AE-E] .
            then have  $\exists N \in \text{null-sets } (\text{proc-source } X).$ 
               $\{\omega \in \text{space } (\text{proc-source } X). X(l! i) \omega - X(l!(i-1)) \omega \neq Y(l! i) \omega - Y(l!(i-1)) \omega\} \subseteq N$ 
              by blast
            } then obtain N where N:  $N i \in \text{null-sets } (\text{proc-source } X)$ 
               $\{\omega \in \text{space } (\text{proc-source } X). X(l! i) \omega - X(l!(i-1)) \omega \neq Y(l! i) \omega - Y(l!(i-1)) \omega\} \subseteq N i$ 
              if  $i \in \{1..<\text{length } l\}$  for  $i$ 
              by (metis (lifting) ext)
            have  $\{\omega \in \text{space } (\text{proc-source } X). \neg (\forall i \in \{1..<\text{length } l\}. X(l! i) \omega - X(l!(i-1)) \omega = Y(l! i) \omega - Y(l!(i-1)) \omega)\} \subseteq (\bigcup i \in \{1..<\text{length } l\}. N i)$ 
              using N by blast
            moreover have  $(\bigcup i \in \{1..<\text{length } l\}. N i) \in \text{null-sets } (\text{proc-source } X)$ 
              apply (rule null-sets.finite-UN)
              using 3 N by simp-all
            ultimately show ?thesis
              by (blast intro: AE-I)
          }
        qed

```

```

show  $AE\ x$  in proc-source
   $X. (\forall i \in \{1..<length\ l\}. process\ X\ (l!\ i)\ x - process\ X\ (l!\ (i-1))\ x$ 
 $= process\ Y\ (l!\ i)\ x - process\ Y\ (l!\ (i-1))\ x) \longrightarrow$ 
   $(\lambda i \in \{1..<length\ l\}. process\ X\ (l!\ i)\ x - process\ X\ (l!\ (i-1))\ x) =$ 
 $(\lambda i \in \{1..<length\ l\}. process\ Y\ (l!\ i)\ x - process\ Y\ (l!\ (i-1))\ x)$ 
  by (rule AE-I2, auto)
qed
have distr (proc-source Y) ( $Pi_M\ \{1..<length\ l\}\ (\lambda i. proc-target\ Y)$ )
   $(\lambda x. \lambda i \in \{1..<length\ l\}. Y\ (l!\ i)\ x - Y\ (l!\ (i-1))\ x) =$ 
  distr (proc-source X) ( $Pi_M\ \{1..<length\ l\}\ (\lambda i. proc-target\ X)$ )
   $(\lambda x. \lambda i \in \{1..<length\ l\}. X\ (l!\ i)\ x - X\ (l!\ (i-1))\ x)$ 
apply (rule sym)
apply (rule distr-cong-AE)
using assms(1) apply blast
  apply (metis assms(2) sets-PiM-cong target-X)
  apply (fact AE-eq')
  apply simp
  apply (rule measurable-restrict)
  apply (simp add: measurable-target)
subgoal by measurable (meson measurable-target random-process) +
apply (rule measurable-restrict)
by (metis (full-types) assms(2) borel-measurable-diff measurable-cong-sets stochastic-process-measurable)
also have ... =  $Pi_M\ \{1..<length\ l\}\ (\lambda i. distr\ (proc-source\ X)\ (proc-target\ X)$ 
 $(\lambda v. X\ (l!\ i)\ v - X\ (l!\ (i-1))\ v))$ 
  apply (subst proc-source.indep-vars-iff-distr-eq-PiM[symmetric])
subgoal using 3 by simp
  apply simp
  apply (metis (full-types) borel-measurable-diff measurable-cong-sets stochastic-process-measurable target-X)
  apply (rule indep-incrementsE)
  apply (fact 3(1))
using 3(2-) assms(1) by blast
also have ... =  $Pi_M\ \{1..<length\ l\}\ (\lambda i. distr\ (proc-source\ Y)\ (proc-target\ Y)$ 
 $(\lambda v. Y\ (l!\ i)\ v - Y\ (l!\ (i-1))\ v))$ 
  apply (safe intro!: PiM-cong)
  apply (rule distr-cong-AE)
subgoal using assms(1) by blast
subgoal using assms(1) by blast
subgoal using AE-eq by presburger
subgoal by (metis (mono-tags) borel-measurable-diff measurable-target random-process)
by (metis (full-types) assms(2) borel-measurable-diff measurable-cong-sets random-process)
finally show ?case .
qed
end

```

6 The Kolmogorov-Chentsov theorem

theory *Kolmogorov-Chentsov*

imports *Stochastic-Processes Holder-Continuous Dyadic-Interval Measure-Convergence*
begin

6.1 Supporting lemmas

The main contribution of this file is the Kolmogorov-Chentsov theorem: given a stochastic process that satisfies some continuity properties, we can construct a Hölder continuous modification. We first prove some auxiliary lemmas before moving on to the main construction.

Klenke 5.11: Markov inequality. Compare with $\llbracket (\lambda x. ?u\ x * \text{indicator } ?A\ x) \in \text{borel-measurable } ?M; ?A \in \text{sets } ?M \rrbracket \implies \text{emeasure } ?M \{x \in ?A. 1 \leq ?c * ?u\ x\} \leq ?c * \text{set-nn-integral } ?M\ ?A\ ?u$

lemma *nn-integral-Markov-inequality-extended*:

fixes $f :: \text{real} \Rightarrow \text{ennreal}$ **and** $\varepsilon :: \text{real}$ **and** $X :: 'a \Rightarrow \text{real}$

assumes *mono*: $\text{mono-on } (\text{range } X \cup \{0 < ..\})\ f$

and *finite*: $\bigwedge x. f\ x < \infty$

and $e: \varepsilon > 0\ f\ \varepsilon > 0$

and $[\text{measurable}]: X \in \text{borel-measurable } M$

shows $\text{emeasure } M \{p \in \text{space } M. (X\ p) \geq \varepsilon\} \leq (\int^+ x. f\ (X\ x)\ \partial M) / f\ \varepsilon$

proof –

have $f\text{-eq}: f = (\lambda x. \text{ennreal } (\text{enn2real } (f\ x)))$

using *finite* **by** *simp*

have *mono-on* $(\text{range } X)\ (\lambda x. \text{enn2real } (f\ x))$

apply $(\text{intro } \text{mono-onI})$

using $\text{mono}[\text{THEN } \text{mono-onD}]\ \text{finite}$ **by** $(\text{simp add: enn2real-mono})$

then have $f \in \text{borel-measurable } (\text{restrict-space borel } (\text{range } X))$

apply $(\text{subst } f\text{-eq})$

apply $(\text{intro measurable-compose}[\text{where } f = \lambda x. \text{enn2real } (f\ x)\ \text{and } g = \text{ennreal}])$

using *borel-measurable-mono-on-fnc* **apply** *blast*

apply *simp*

done

then have $(\lambda x. f\ (X\ x)) \in \text{borel-measurable } M$

apply $(\text{intro measurable-compose}[\text{where } g = f\ \text{and } f = X\ \text{and } N = \text{restrict-space borel } (\text{range } X)])$

apply $(\text{simp-all add: measurable-restrict-space2})$

done

then have $\{x \in \text{space } M. f\ (X\ x) \geq f\ \varepsilon\} \in \text{sets } M$

by *measurable*

then have $f\ \varepsilon * \text{emeasure } M \{x \in \text{space } M. X\ x \geq \varepsilon\} \leq (\int^+ x \in \{x \in \text{space } M. f\ \varepsilon \leq f\ (X\ x)\}. f\ \varepsilon\ \partial M)$

apply $(\text{simp add: nn-integral-cmult-indicator})$

using $e\ \text{mono-onD}[OF\ \text{mono}]\ \text{zero-le}$ **apply** $(\text{blast intro: mult-left-mono emeasure-mono})$

done

also have $\dots \leq (\int^+ x \in \{x \in \text{space } M. f\ \varepsilon \leq f\ (X\ x)\}. f\ (X\ x)\ \partial M)$

```

    apply (rule nn-integral-mono)
  subgoal for x
    apply (cases f ε ≤ f (X x))
    using ennreal-leI by auto
  done
  also have ... ≤ ∫+ x. f (X x) ∂M
    by (simp add: nn-integral-mono indicator-def)
  finally have emeasure M {p ∈ space M. ε ≤ X p} * f ε / f ε ≤ (∫+ x. f (X x)
    ∂M) / f ε
    by (simp add: divide-right-mono-ennreal field-simps)
  then show ?thesis
    using mult-divide-eq-ennreal finite[of ε] e(2) by simp
qed

```

```

lemma nn-integral-Markov-inequality-extended-rnv:
  fixes f :: real ⇒ real and ε :: real and X :: 'a ⇒ 'b :: real-normed-vector
  assumes [measurable]: X ∈ borel-measurable M
    and mono: mono-on {0..} f
    and e: ε > 0 f ε > 0
  shows emeasure M {p ∈ space M. norm (X p) ≥ ε} ≤ (∫+ x. f (norm (X x))
    ∂M) / f ε
  apply (rule nn-integral-Markov-inequality-extended)
  using mono ennreal-leI unfolding mono-on-def apply force
  apply (simp-all add: e)
  done

```

6.2 Kolmogorov-Chentsov

Klenke theorem 21.6 - Kolmogorov-Chentsov

```

locale Kolmogorov-Chentsov =
  fixes X :: (real, 'a, 'b :: polish-space) stochastic-process
    and a b C γ :: real
  assumes index[simp]: proc-index X = {0..}
    and target-borel[simp]: proc-target X = borel
    and gt-0: a > 0 b > 0 C > 0
    and b-leq-a: b ≤ a
    and gamma: γ ∈ {0 < .. < b/a}
    and expectation: ⋀ s t. [s ≥ 0; t ≥ 0] ⇒
      (∫+ x. dist (X t x) (X s x) powr a ∂proc-source X) ≤ C * dist t s powr
      (1+b)
begin

```

```

lemma gamma-0-1[simp]: γ ∈ {0 < .. 1}
  using gt-0 b-leq-a gamma
  by (metis divide-less-eq-1-pos divide-self greaterThanAtMost-iff
    greaterThanLessThan-iff nless-le order-less-trans)

```

```

lemma gamma-gt-0[simp]: γ > 0
  using gamma greaterThanLessThan-iff by blast

```


lemma *gamma-le-1*[simp]: $\gamma \leq 1$

using *gamma-0-1* **by** *auto*

abbreviation *source* $\equiv$ *proc-source* *X*

lemma *X-borel-measurable*[*measurable*]: $X \ t \in \text{borel-measurable source for } t$

by (*metis random-process target-borel*)

lemma *markov*: $\mathcal{P}(x \text{ in source. } \varepsilon \leq \text{dist } (X \ t \ x) \ (X \ s \ x)) \leq (C * \text{dist } t \ s \ \text{powr } (1 + b)) / \varepsilon \ \text{powr } a$

if $s \geq 0 \ t \geq 0 \ \varepsilon > 0$ **for** $s \ t \ \varepsilon$

proof –

let *?inc* = $\lambda x. \text{dist } (X \ t \ x) \ (X \ s \ x) \ \text{powr } a$

have *emeasure source* $\{x \in \text{space source. } \varepsilon \leq \text{dist } (X \ t \ x) \ (X \ s \ x)\}$

$\leq \text{integral}^N \text{ source } ?inc / \varepsilon \ \text{powr } a$

apply (*rule nn-integral-Markov-inequality-extended*)

using *that(1,2)* **apply** *measurable*

subgoal using *gt-0(1) imageE powr-mono2* **by** (*auto intro: mono-onI*)

using *that* **apply** *simp-all*

done

also have $\dots \leq (C * \text{dist } t \ s \ \text{powr } (1 + b)) / \text{ennreal } (\varepsilon \ \text{powr } a)$

apply (*rule divide-right-mono-ennreal*)

using *expectation[OF that(1,2)] ennreal-leI* **by** *simp*

finally have *emeasure source* $\{x \in \text{space source. } \varepsilon \leq \text{dist } (X \ t \ x) \ (X \ s \ x)\}$

$\leq (C * \text{dist } t \ s \ \text{powr } (1 + b)) / \varepsilon \ \text{powr } a$

using *that(3) divide-ennreal gt-0(3)* **by** *simp*

moreover have $C * \text{dist } t \ s \ \text{powr } (1 + b) / \varepsilon \ \text{powr } a \geq 0$

using *gt-0(3)* **by** *auto*

ultimately show *?thesis*

by (*simp add: proc-source.emeasure-eq-measure*)

qed

lemma *conv-in-prob*:

assumes $t \geq 0$

shows *tendsto-measure (proc-source X) X (X t) (at t within {0..})*

proof –

{

fix $p \ \varepsilon :: \text{real}$ **assume** $0 < p \ 0 < \varepsilon$

let *?q* = $(p * \varepsilon \ \text{powr } a / C) \ \text{powr } (1/(1+b))$

have $0 < ?q$

using $\langle 0 < p \rangle \ \text{gt-0(3)} \ \langle 0 < \varepsilon \rangle$ **by** *simp*

have *p-eq*: $p = (C * ?q \ \text{powr } (1 + b)) / \varepsilon \ \text{powr } a$

using *gt-0* $\langle 0 < ?q \rangle \ \langle 0 < p \rangle$ **by** (*simp add: field-simps powr-powr*)

have $0 < \text{dist } r \ t \wedge \text{dist } r \ t < ?q \longrightarrow \text{dist } \mathcal{P}(x \text{ in source. } \varepsilon \leq \text{dist } (X \ t \ x) \ (X \ r \ x)) \ 0 \leq p$

if $r \in \{0..\}$ **for** r

proof *safe*

assume $0 < \text{dist } r \ t \ \text{dist } r \ t < ?q$

```

    have  $0 \leq r$ 
    using that by auto
    from  $\langle \text{dist } r \ t < ?q \rangle$  have  $C * \text{dist } r \ t \text{ powr } (1 + b) / \varepsilon \text{ powr } a \leq p$ 
    apply (subst p-eq)
    using gt-0(2) gt-0(3) apply (simp add: divide-le-cancel powr-mono2)
    done
    then show  $\text{dist } \mathcal{P}(x \text{ in source. } \varepsilon \leq \text{dist } (\text{process } X \ t \ x) (\text{process } X \ r \ x)) \ 0 \leq$ 
p
    using markov[OF  $\langle 0 \leq r \rangle$  assms  $\langle 0 < \varepsilon \rangle$ ] by (simp add: dist-commute)
qed
    then have  $\exists d > 0. \forall r \in \{0.. \}. \ 0 < \text{dist } r \ t \wedge \text{dist } r \ t < d \longrightarrow$ 
 $\text{dist } \mathcal{P}(x \text{ in source. } \varepsilon \leq \text{dist } (X \ r \ x) (X \ t \ x)) \ 0 \leq p$ 
    apply (intro exI[where  $x = ?q$ ])
    apply (subst(3) dist-commute)
    using  $\langle 0 < p \rangle$  gt-0(3)  $\langle 0 < \varepsilon \rangle$  dist-commute by fastforce
  } then show ?thesis
    by (simp add: finite-measure.tendsto-measure-leq, safe, intro Lim-withinI)
qed

lemma conv-in-prob-finite:
  assumes  $t \geq 0$ 
  shows tendsto-measure (proc-source X) X (X t) (at t within  $\{0..T\}$ )
proof -
  have at t within  $\{0..T\} \leq$  at t within  $\{0.. \}$ 
  by (simp add: at-le)
  then show ?thesis
    apply (rule tendsto-measure-mono)
    using assms by (rule conv-in-prob)
qed

lemma incr:  $\mathcal{P}(x \text{ in source. } 2 \text{ powr } (-\gamma * n) \leq \text{dist } (X ((k - 1) * 2 \text{ powr } - n) x) (X (k * 2 \text{ powr } - n) x))$ 
 $\leq C * 2 \text{ powr } (-n * (1 + b - a * \gamma))$ 
  if  $k \geq 1 \ n \geq 0$  for k n
proof -
  have  $\mathcal{P}(x \text{ in source. } 2 \text{ powr } (-\gamma * n) \leq \text{dist } (X ((k - 1) * 2 \text{ powr } - n) x) (X (k * 2 \text{ powr } - n) x))$ 
 $\leq C * \text{dist } ((k - 1) * 2 \text{ powr } - n) (k * 2 \text{ powr } - n) \text{ powr } (1 + b) / (2 \text{ powr } (-\gamma * n)) \text{ powr } a$ 
  using that by (auto intro: markov)
  also have  $\dots = C * 2 \text{ powr } (-n - b * n) / 2 \text{ powr } (-\gamma * n * a)$ 
  by (auto simp: dist-real-def powr-powr field-simps)
  also have  $\dots = C * 2 \text{ powr } (-n * (1 + b - a * \gamma))$ 
  by (simp add: field-simps powr-add[symmetric])
  finally show ?thesis .
qed

end

```

In order to construct the modification of X , it suffices to construct a modi-

fication of X on $\{0..T\}$ for all finite T , from which we construct the modification on $\{0..\}$ via a countable union.

locale *Kolmogorov-Chentsov-finite* = *Kolmogorov-Chentsov* +
fixes $T :: \text{real}$
assumes *zero-le-T*: $0 < T$
begin

A_n will characterise the set of states with increments that exceed the bounds required for Hölder continuity. As $n \rightarrow \infty$, this approaches the set of states for which X is not Hölder continuous. We define N as this limit, and show that N is a null set. On $\omega \in \Omega - N$, we show that $X(\omega)$ is Hölder continuous (and therefore uniformly continuous) on the dyadic rationals, and construct a modification by taking the continuous extension on the reals.

definition $A \equiv \lambda n. \text{if } 2^{\wedge} n * T < 1 \text{ then space source else}$
 $\{x \in \text{space source}.$
 $\text{Max } \{ \text{dist } (X \text{ (real-of-int } (k - 1) * 2^{\text{powr}} - \text{real } n) x) (X \text{ (real-of-int } k * 2^{\text{powr}} - \text{real } n) x)$
 $\mid k. k \in \{1..[2^{\wedge} n * T]\} \} \geq 2^{\text{powr}} (-\gamma * \text{real } n) \}$

abbreviation $B \equiv \lambda n. (\bigcup m. A (m + n))$

abbreviation $N \equiv \bigcap (\text{range } B)$

lemma *A-geq*: $2^{\wedge} n * T \geq 1 \implies A n = \{x \in \text{space source}.$
 $\text{Max } \{ \text{dist } (X \text{ (real-of-int } (k - 1) * 2^{\text{powr}} - \text{real } n) x) (X \text{ (real-of-int } k * 2^{\text{powr}} - \text{real } n) x)$
 $\mid k. k \in \{1..[2^{\wedge} n * T]\} \} \geq 2^{\text{powr}} (-\gamma * \text{real } n) \}$ **for** n
by (*simp add: A-def*)

lemma *A-measurable[measurable]*: $A n \in \text{sets source}$
unfolding *A-def* **apply** (*cases* $2^{\wedge} n * T < 1$)
apply *simp*
apply (*simp only: if-False*)
apply *measurable*
done

lemma *emeasure-A-leq*:

fixes $n :: \text{nat}$
assumes [*simp*]: $2^{\wedge} n * T \geq 1$
shows *emeasure source* $(A n) \leq C * T * 2^{\text{powr}} (-n * (b - a * \gamma))$

proof –

have *nonempty*: $\{1..[2^{\wedge} n * T]\} \neq \{\}$
using *assms* **by** *fastforce*
have *finite*: *finite* $\{1..[2^{\wedge} n * T]\}$
by *simp*
have *emeasure source* $(A n) \leq \text{emeasure source } (\bigcup k \in \{1..[2^{\wedge} n * T]\}.$
 $\{x \in \text{space source}. \text{dist } (X \text{ (real-of-int } (k - 1) * 2^{\text{powr}} - \text{real } n) x) (X \text{ (real-of-int } k * 2^{\text{powr}} - \text{real } n) x) \geq 2^{\text{powr}} (-\gamma * \text{real } n) \})$

```

    (is emeasure source (A n) ≤ emeasure source ?R)
  proof (rule emeasure-mono, intro subsetI)
    fix x assume *: x ∈ A n
    from * have in-space: x ∈ space source
      using A-measurable sets.sets-into-space by blast
    from * have 2 powr (− γ * real n) ≤ Max {dist (X (real-of-int (k − 1) * 2
    powr − real n) x) (X (real-of-int k * 2 powr − real n) x) | k. k ∈ {1..[2 ^ n * T]}}
      using A-geq assms by blast
    then have ∃ k ∈ {1..[2 ^ n * T]}. 2 powr (− γ * real n) ≤ dist (X (real-of-int
    (k − 1) * 2 powr − real n) x) (X (real-of-int k * 2 powr − real n) x)
      apply (simp only: setcompr-eq-image)
      apply (rule Max-finite-image-ex[where P=λx. 2 powr (− γ * real n) ≤ x,
    OF finite_nonempty])
      apply (metis Collect-mem-eq)
    done
    then show x ∈ ?R
      using in-space by simp
  next
    show ?R ∈ sets source
      by measurable
  qed
  also have ... ≤ (∑ k ∈ {1..[2 ^ n * T]}. emeasure source
    {x ∈ space source. dist (X (real-of-int (k − 1) * 2 powr − real n) x) (X (real-of-int
    k * 2 powr − real n) x) ≥ 2 powr (− γ * real n)})
    apply (rule emeasure-subadditive-finite)
    apply blast
    apply (subst image-subset-iff)
    apply (intro ballI)
    apply measurable
    done
  also have ... ≤ C * 2 powr (− n * (1 + b − a * γ)) * (card {1..[2 ^ n * T]})
  proof −
    {
      fix k assume k ∈ {1..[2 ^ n * T]}
      then have real-of-int k ≥ 1
        by presburger
      then have P(x in source. 2 powr (− γ * real n) ≤ dist (X (real-of-int (k −
      1) * 2 powr − real n) x) (X (real-of-int k * 2 powr − real n) x))
        ≤ C * 2 powr (−(real n) * (1 + b − a * γ))
        using incr gamma by force
    } note X = this
    then have sum (λk. P(x in source. 2 powr (− γ * real n) ≤ dist (X (real-of-int
    (k − 1) * 2 powr − real n) x) (X (real-of-int k * 2 powr − real n) x)))
      {1..[2 ^ n * T]} ≤ of-nat (card {1..[2 ^ n * T]}) * (C * 2
    powr (−(real n) * (1 + b − a * γ)))
      by (fact sum-bounded-above)
    then show ?thesis
      using ennreal-leI by (auto simp: proc-source.emeasure-eq-measure mult.commute)
  qed

```

also have $\dots \leq C * 2^{\text{powr } (-n * (1 + b - a * \gamma))} * \lfloor 2^n * T \rfloor$
using *nonempty zle-iff-zadd* **by** *force*
also have $\dots \leq C * 2^{\text{powr } (-n * (1+b - a * \gamma))} * 2^n * T$
by (*simp add:ennreal-leI gt-0(3)*)
also have $\dots = C * 1/(2^n) * 2^{\text{powr } (-n * (b - a * \gamma))} * 2^n * T$
apply (*intro ennreal-cong*)
apply (*simp add: scale-left-imp-eq field-simps*)
by (*smt (verit) powr-add powr-realpow*)
also have $\dots = C * T * 2^{\text{powr } (-n * (b - a * \gamma))}$
by (*simp add: field-simps*)
finally show *?thesis* .
qed

lemma *measure-A-leq*:
assumes $2^n * T \geq 1$
shows *measure source (A n) $\leq C * T * 2^{\text{powr } (-n * (b - a * \gamma))}$*
apply (*intro measure-leq-emeasure-ennreal*)
subgoal using *gt-0(3) zero-le-T* **by** *auto*
using *emeasure-A-leq* **apply** (*simp add: A-geq assms*)
done

lemma *summable-A*: *summable ($\lambda m.$ measure source (A m))*
proof –
have $b - a * \gamma > 0$
by (*metis diff-gt-0-iff-gt gamma greaterThanLessThan-iff gt-0(1) mult.commute pos-less-divide-eq*)
have $1: 2^{\text{powr } (-\text{real } x * (b - a * \gamma))} = (1 / 2^{\text{powr } (b - a * \gamma)})^x$ **for** x
apply (*cases x = 0*)
by (*simp-all add: field-simps powr-add[symmetric] powr-realpow[symmetric] powr-powr*)
have $2: \text{summable } (\lambda n. 2^{\text{powr } (-n * (b - a * \gamma))})$ **(is summable ?C)**
proof –
have *summable ($\lambda n. (1 / 2^{\text{powr } (b - a * \gamma)})^n$)*
using $\langle b - a * \gamma > 0 \rangle$ **by** *auto*
then show *summable ($\lambda x. 2^{\text{powr } (-\text{real } x * (b - a * \gamma))}$)*
using 1 **by** *simp*
qed
from *zero-le-T* **obtain** N **where** $2^N * T \geq 1$
by (*metis dual-order.order-iff-strict mult.commute one-less-numeral-iff pos-divide-le-eq power-one-over reals-power-lt-ex semiring-norm(76) zero-less-numeral zero-less-power*)
then have $\bigwedge n. n \geq N \implies 2^n * T \geq 1$
by (*smt (verit, best) $\langle 0 < T \rangle$ mult-right-mono power-increasing-iff*)
then have $\bigwedge n. n \geq N \implies \text{norm } (\text{measure source (A n)}) \leq C * T * 2^{\text{powr } (-n * (b - a * \gamma))}$
using *measure-A-leq* **by** *simp*
moreover have *summable ($\lambda n. C * T * 2^{\text{powr } (-n * (b - a * \gamma))}$)*
using 2 *summable-mult* **by** *simp*
ultimately show *?thesis*
using *summable-comparison-test'* **by** *fast*

qed

lemma *lim-B*: $(\lambda n. \text{measure source } (B \ n)) \longrightarrow 0$

proof –

have *measure-B-le*: $\text{measure source } (B \ n) \leq (\sum m. \text{measure source } (A \ (m + n)))$

for *n*

apply (*rule proc-source.finite-measure-subadditive-countably*)

subgoal by *auto*

apply (*subst summable-iff-shift*)

using *summable-A* **by** *blast*

have *lim-A*: $(\lambda n. (\sum m. \text{measure source } (A \ (m + n)))) \longrightarrow 0$

by (*fact suminf-exist-split2[OF summable-A]*)

have *convergent* $(\lambda n. \text{measure source } (B \ n))$

proof (*intro Bseq-monoseq-convergent*)

show *Bseq* $(\lambda n. \text{Sigma-Algebra.measure source } (\bigcup m. A \ (m + n)))$

apply (*rule BseqI'[where K=measure source (bigcup (range A))]*)

apply (*auto intro!: proc-source.finite-measure-mono*)

done

show *monoseq* $(\lambda n. \text{Sigma-Algebra.measure source } (\bigcup m. A \ (m + n)))$

apply (*intro decseq-imp-monoseq[unfolded decseq-def] allI impI proc-source.finite-measure-mono*)

apply (*simp-all add: Union-add-subset*)

done

qed

then obtain *L* **where** *lim-B*: $(\lambda n. \text{measure source } (B \ n)) \longrightarrow L$

unfolding *convergent-def* **by** *auto*

then have $L \geq 0$

by (*simp add: LIMSEQ-le-const*)

moreover have $L \leq 0$

using *measure-B-le* **by** (*simp add: LIMSEQ-le[OF lim-B lim-A]*)

ultimately show *?thesis*

using *lim-B* **by** *simp*

qed

lemma *N-null*: $N \in \text{null-sets source}$

proof –

have $(\lambda n. \text{measure source } (B \ n)) \longrightarrow \text{measure source } N$

apply (*rule proc-source.finite-Lim-measure-decseq*)

using *A-measurable* **apply** *fast*

apply (*intro monotoneI, simp add: Union-add-subset*)

done

then have $\text{measure source } N = 0$

using *lim-B LIMSEQ-unique* **by** *blast*

then show *?thesis*

by (*auto simp add: emeasure-eq-ennreal-measure*)

qed

lemma *notin-N-index*:

assumes $\omega \in \text{space source} - N$

obtains n_0 **where** $\omega \notin (\bigcup n. A \ (n + n_0))$

```

using assms by blast

context
  fixes  $\omega$ 
  assumes  $\omega: \omega \in \text{space source} - N$ 
begin

definition  $n_0 \equiv \text{SOME } m. \omega \notin (\bigcup n. A (n + m)) \wedge m > 0$ 

lemma
  shows  $n\text{-zero}: \omega \notin (\bigcup n. A (n + n_0))$ 
  and  $n\text{-zero-nonzero}: n_0 > 0$ 
proof -
  have  $\exists m. \omega \notin (\bigcup n. A (n + m))$ 
  using  $\omega$  by blast
  then have  $\exists m. \omega \notin (\bigcup n. A (n + m)) \wedge m > 0$ 
  by (metis (no-types, lifting) UNIV-I UN-iff add.comm-neutral not-gr-zero zero-less-Suc)
  then have  $\omega \notin (\bigcup n. A (n + n_0)) \wedge n_0 > 0$ 
  unfolding  $n_0\text{-def}$  by (rule someI-ex)
  then show  $\omega \notin (\bigcup n. A (n + n_0)) \wedge n_0 > 0$ 
  by blast+
qed

lemma  $nzero\text{-ge}: \bigwedge n. n \geq n_0 \implies 2^{\wedge n} * T \geq 1$ 
proof (rule ccontr)
  fix  $n$  assume  $n_0 \leq n \rightarrow 1 \leq 2^{\wedge n} * T$ 
  then have  $A\ n = \text{space source}$ 
  unfolding  $A\text{-def}$  by simp
  then have  $\text{space source} \subseteq (\bigcup m. A (m + n))$ 
  by (smt (verit, del-insts) UNIV-I UN-upper add-0)
  also have  $(\bigcup m. A (m + n)) \subseteq (\bigcup m. A (m + n_0))$ 
  by (simp add: Union-add-subset  $\langle n_0 \leq n \rangle$ )
  finally show False
  using  $\omega$   $n\text{-zero}$  by blast
qed

lemma  $\omega\text{-notin}: \bigwedge n. n \geq n_0 \implies \omega \notin A\ n$ 
by (metis  $n\text{-zero}$  UNIV-I UN-iff add commute le-Suc-ex)

Klenke 21.7

lemma  $X\text{-dyadic-incr}$ :
  assumes  $k \in \{1..[2^{\wedge n} * T]\}$   $n \geq n_0$ 
  shows  $\text{dist } (X ((\text{real-of-int } k - 1) / 2^{\wedge n}) \omega) (X (\text{real-of-int } k / 2^{\wedge n}) \omega) < 2^{\text{powr}} (- \gamma * n)$ 
proof -
  have  $\text{finite } \{1..[2^{\wedge n} * T]\} \{1..[2^{\wedge n} * T]\} \neq \{\}$ 
  using assms  $nzero\text{-ge}$  by blast+
  then have  $\text{fin-nonempty}: \text{finite } \{\text{dist } (X (\text{real-of-int } (k - 1) * 2^{\text{powr}} - \text{real } n) \omega) (X (\text{real-of-int } k * 2^{\text{powr}} - \text{real } n) \omega) \mid k.\}$ 

```

$k \in \{1..[2^{\wedge} n * T]\}$ } $\{dist (X (real-of-int (k - 1) * 2^{powr} - real\ n) \omega) \mid k.$
 $\omega) (X (real-of-int\ k * 2^{powr} - real\ n) \omega) \mid k.$
 $k \in \{1..[2^{\wedge} n * T]\} \neq \{\}$
by *fastforce+*
have $2^{powr} (-\gamma * real\ n)$
 $> Max \{dist (X (real-of-int (k - 1) * 2^{powr} - real\ n) \omega) (X (real-of-int\ k * 2^{powr} - real\ n) \omega) \mid k.$
 $k \in \{1..[2^{\wedge} n * T]\}$
using *nzero-ge*[*OF assms*(2)] *omega-notin*[*OF assms*(2)] ω *A-def* **by** *auto*
then have $2^{powr} (-\gamma * real\ n) > dist (X (real-of-int (k - 1) * 2^{powr} - real\ n) \omega) (X (real-of-int\ k * 2^{powr} - real\ n) \omega)$
using *Max-less-iff*[*OF fin-nonempty*] *assms*(1) **by** *blast*
then show *?thesis*
by (*simp*, *smt* (*verit*, *ccfv-threshold*) *divide-powr-uminus powr-realpow*)
qed

Klenke (21.8)

lemma *dist-dyadic-mn*:

assumes *mn*: $n_0 \leq n \leq m$
and *t-dyadic*: $t \in dyadic-interval-step\ m\ 0\ T$
and *u-dyadic-n*: $u \in dyadic-interval-step\ n\ 0\ T$
and *ut*: $u \leq t \wedge t - u < 2/2^{\wedge} n$
shows $dist (X\ u\ \omega) (X\ t\ \omega) \leq 2^{powr} (-\gamma * n) / (1 - 2^{powr} - \gamma)$
proof –
have *u-dyadic*: $u \in dyadic-interval-step\ m\ 0\ T$
using *mn*(2) *dyadic-interval-step-subset* *u-dyadic-n* **by** *fast*
have $0 < n$
using *mn*(1) *n-zero-nonzero* **by** *linarith*
then have $t - u < 1$
by (*smt* (*verit*) *ut*(2) *One-nat-def* *Suc-le-eq* *divide-le-eq-1-pos* *power-increasing* *power-one-right*)
obtain *b-tu* *k-tu* **where** *tu-exp*: *dyadic-expansion* ($t - u$) *m* *b-tu* *k-tu*
using *dyadic-expansion-ex* *dyadic-interval-minus*[*OF* *u-dyadic* *t-dyadic* $\langle u \leq t \rangle$]
by *blast*
then have $k-tu = 0$
using *dyadic-expansion-floor*[*OF* *tu-exp*] $\langle t - u < 1 \rangle \langle u \leq t \rangle$ **by** *linarith*
have *b-tu-0-1*: $b-tu ! i \in \{0, 1\}$ **if** $i \in \{0..m-1\}$ **for** *i*
using *dyadic-expansionD*(1,2)[*OF* *tu-exp*] *that*
by (*metis* *Suc-pred'* $\langle 0 < n \rangle$ *atLeastAtMost-iff* *le-imp-less-Suc* *le-trans* *less-eq-Suc-le* *mn*(2) *nth-mem* *subsetD*)

And hence $b_i(t - u) = b_i(s - u) = 0$ for $i < n$.

have *b-t-zero*: $b-tu ! i = 0$ **if** $i+1 < n$ **for** *i*
proof (*rule ccontr*)
assume $b-tu ! i \neq 0$
then have $b-tu ! i = 1$
by (*smt* (*verit*) *add-lessD1* *dyadic-expansionD*(1,2) *insertE* *mn*(2) *nth-mem* *order-less-le-trans* *singletonD* *subset-iff* *that* *tu-exp*)


```

then have  $t - u \geq (\text{real-of-int } 0) + 1/2^{\wedge}(i+1)$ 
  apply (intro dyadic-expansion-nth-geq)
  using  $\text{tu-exp } \langle k\text{-tu} = 0 \rangle$  apply blast
  apply (metis One-nat-def Suc-eq-plus1 Suc-le-mono atLeastAtMost-iff le-trans
less-Suc-eq linorder-not-le mn(2) nat-less-le that zero-order(1))
  apply simp
done
moreover have  $1/2^{\wedge}(n-1) \leq 1/(2^{\wedge}(i+1)) :: \text{real}$ 
  apply (intro divide-left-mono)
  apply (metis that Suc-eq-plus1 Suc-leI less-diff-conv power-increasing
power-one-right two-realpow-ge-one)
  by simp-all
ultimately have  $t - u \geq 1 / 2^{\wedge}(n - 1)$ 
  by linarith
then show False
  using  $\langle t - u < 2/2^{\wedge}n \rangle \langle n > 0 \rangle$  by (auto simp: power-diff)
qed
define  $t'$  where  $t' \equiv \lambda l. (u + (\sum i = n..l. b\text{-tu!}(i-1) / 2^{\wedge} i))$ 
have  $t'(n-1) = u$ 
  unfolding  $t'\text{-def}$  using  $\langle n > 0 \rangle$  by simp
have  $t' m = t$ 
proof -
  have  $b\text{-tu-eq-0}: (\sum i = 1..n-1. b\text{-tu!}(i-1) / 2^{\wedge} i) = 0$ 
    by (subst sum-nonneg-eq-0-iff, auto simp add: sum-nonneg-eq-0-iff b-t-zero)
  have  $t - u = (\sum i = 1..m. b\text{-tu!}(i-1) / 2^{\wedge} i)$ 
    using  $\text{tu-exp}[THEN \text{dyadic-expansionD}(3)] \langle k\text{-tu} = 0 \rangle$  by linarith
  also have  $\dots = (\sum i = 1..n-1. b\text{-tu!}(i-1) / 2^{\wedge} i) + (\sum i = n..m. b\text{-tu!}(i-1) / 2^{\wedge} i)$ 
  proof -
    have  $1: \{1..m\} = \{1..n-1\} \cup \{n..m\}$ 
      using  $\langle n > 0 \rangle \text{mn}(2)$  by fastforce
    show ?thesis
      by (subst 1, auto simp: sum.union-disjoint)
  qed
  finally have  $t-u = (\sum i = n..m. b\text{-tu!}(i-1) / 2^{\wedge} i)$ 
    using  $b\text{-tu-eq-0}$  by algebra
  then show ?thesis
    unfolding  $t'\text{-def}$  by argo
qed
have  $t\text{-pos}: t' l \geq 0$  if  $\langle l \in \{n..m\} \rangle$  for  $l$ 
  unfolding  $t'\text{-def}$  apply (rule add-nonneg-nonneg)
  using dyadic-step-geq u-dyadic apply blast
  by (simp add: sum-nonneg)
have  $t'\text{-Suc}: t'(\text{Suc } l) = t' l + b\text{-tu! } l / 2^{\wedge}(\text{Suc } l)$  if  $l \in \{n-1..m-1\}$  for  $l$ 
  unfolding  $t'\text{-def}$  by (simp add: b-t-zero)
have  $\text{le-add-diff}: b \leq c - a \implies a + b \leq c$  for  $a \ b \ c :: \text{real}$ 
  by argo
have  $t'\text{-leq}: t' l \leq t$  if  $\langle l \in \{n..m\} \rangle$  for  $l$ 
  unfolding  $t'\text{-def}$  apply (intro le-add-diff)

```

```

    apply (simp only: tu-exp[THEN dyadic-expansionD(3)] ⟨k-tu = 0⟩ of-int-0
add-0)
    apply (rule sum-mono2)
    using ⟨0 < n⟩ that by auto
    have t'-dyadic: t' l ∈ dyadic-interval-step l 0 T if l ∈ {n..m} for l
    using that
    proof (induct l rule: atLeastAtMost-induct)
    case base
    consider b-tu ! (n - 1) = 0 | b-tu ! (n-1) = 1
    using dyadic-expansion-frac-range[OF tu-exp(1), of n] ⟨0 < n⟩ mn(2) by
auto
    then show ?case
    apply cases
    using ⟨0 < n⟩ t'-def apply simp
    using u-dyadic-n apply blast
    apply (rule dyadic-interval-step-memI)
    apply (simp add: t'-def)
    using u-dyadic-n
    apply (metis add-divide-distrib dyadic-interval-step-iff of-int-1 of-int-add)
    apply (simp add: mn(2) t-pos)
    by (meson t'-leq[OF that] atLeastAtMost-iff dual-order.refl dual-order.trans
dyadic-step-leq mn(2) t'-leq t-dyadic)
    next
    case (Suc l)
    then have t'-dyadic-Suc: t' l ∈ dyadic-interval-step (Suc l) 0 T
    using dyadic-interval-step-mono le-SucI by blast
    from Suc have l ∈ {0..m-1}
    by force
    then consider b-tu!l = 0 | b-tu!l = 1
    using b-tu-0-1 by fastforce
    then obtain k :: int where k: t' l + (b-tu ! l) / 2 ^ Suc l = k / 2 ^ Suc l
    apply cases
    subgoal using dyadic-interval-step-iff t'-dyadic-Suc by auto
    by (metis add.commute add-divide-distrib dyadic-as-natural of-int-of-nat-eq
of-nat-1 of-nat-Suc t'-dyadic-Suc)
    have t' (Suc l) ≤ t
    by (meson Suc atLeastAtMost-iff le-SucI less-eq-Suc-le t'-leq)
    with Suc(1,2) show ?case
    apply (subst t'-Suc)
    apply (metis Suc-leD Suc-pred' ⟨0 < n⟩ atLeastAtMost-iff less-Suc-eq-le
mn(2) order-less-le-trans)
    apply (intro dyadic-interval-step-memI)
    apply (rule exI[where x=k])
    using k apply blast
    using dyadic-step-geq t'-dyadic-Suc apply force
    apply (subst t'-Suc[symmetric])
    apply force
    using dyadic-step-leq order-trans t-dyadic by blast
qed

```

```

have dist (X (t' (n-1)) ω) (X (t' m) ω) ≤ (∑ l=Suc (n-1)..m. dist (X (t' l)
ω) (X (t' (l-1)) ω))
  apply (rule triangle-ineq-sum)
  using diff-le-self dual-order.trans mn(2) by blast
also have ... = (∑ l=n..m. dist (X (t' l) ω) (X (t' (l-1)) ω))
  using Suc-diff-1 ⟨0 < n⟩ by presburger
also have ... ≤ (∑ l=n..m. 2 powr (-γ * l))
proof (rule sum-mono)
  fix l assume *: l ∈ {n..m}
  then have l ∈ {n-1..m}
    by (metis atLeastAtMost-iff less-imp-diff-less linorder-not-less order-le-less)
  from * have [simp]: 0 < l
    using ⟨0 < n⟩ by fastforce
  from ⟨l ∈ {n..m}⟩ have b-tu ! (l-1) ∈ {0, 1}
    apply (intro dyadic-expansion-frac-range)
    apply (rule tu-exp)
    using ⟨n > 0⟩ by simp
  then consider (zero) b-tu ! (l-1) = 0 | (one) b-tu ! (l-1) = 1
    by fast
  then show dist (X (t' l) ω) (X (t' (l-1)) ω) ≤ 2 powr (-γ * l)
  proof cases
    case zero
      have {n..l} = insert l {n..l-1}
        using ⟨0 < n⟩ ⟨l ∈ {n..m}⟩ by auto
      then have sum f {n..l} = sum f {n..l-1} + f l for f :: nat ⇒ real
        by (metis (no-types, opaque-lifting) Groups.add-ac(3) Suc-le-eq Suc-pred'
        ⟨0 < n⟩ atLeastAtMost-iff finite-atLeastAtMost group-cancel.rule0 linorder-not-le
        nle-le sum.insert zero-diff zero-less-diff)
      then have t' l = t' (l-1)
        unfolding t'-def using zero by simp
      then show ?thesis
        by simp
    next
      case one
        then have [simp]: b-tu ! (l - Suc 0) = 1
          by simp
        obtain k where k: k ≥ 0 k ≤ ⌊2l * T⌋ t' l = k/2l
          using t'-dyadic ⟨l ∈ {n..m}⟩ dyadic-interval-step-iff by force
        have t' (l-1) ∈ dyadic-interval-step l 0 T
        proof (cases l = n)
          case True
            then have t' (l-1) = u
              using ⟨t' (n-1) = u⟩ by presburger
            then show t' (l-1) ∈ dyadic-interval-step l 0 T
              using True u-dyadic-n by blast
          next
            case False
              then have l-1 ∈ {n..m}
              by (metis * Suc-eq-plus1 add-leD2 atLeastAtMost-iff diff-le-self dual-order.trans

```

```

    le-antisym not-less-eq-eq ordered-cancel-comm-monoid-diff-class.le-diff-conv2)
  then show  $t' (l-1) \in \text{dyadic-interval-step } l \ 0 \ T$ 
    using  $t'\text{-dyadic dyadic-interval-step-subset diff-le-self}$  by blast
qed
then obtain  $k'$  where  $k': k' \geq 0 \ k' \leq \lfloor 2^l * T \rfloor \ t' (l-1) = k' / 2^l$ 
  using  $\text{dyadic-interval-step-iff}$  by auto
then have  $t'-k: t' (l-1) = (k-1) / 2^l$ 
proof -
  have  $t' l = t' (l-1) + \text{real } (b\text{-tu } ! (l-1)) / 2^l$ 
    using  $t'\text{-Suc[of } l-1]$  apply simp
  using  $* \text{diff-le-mono}$  by presburger
  then have  $k / 2^l = t' (l-1) + 1 / 2^l$ 
    by (simp add:  $k(3)$ )
  then show ?thesis
    by (simp add:  $\text{diff-divide-distrib}$ )
qed
then have  $k \geq 1$ 
  using  $k'(1) \ k'(3)$  by auto
then show ?thesis
  apply (simp only:  $k(3) \ t'-k$ )
  apply (subst  $\text{dist-commute}$ )
  apply (intro  $\text{less-imp-le}$ )
  apply (simp only:  $\text{of-int-diff of-int-1}$ )
  apply (rule  $X\text{-dyadic-incr[of } k \ l]$ )
  using  $k(2)$  apply presburger
  using  $\langle l \in \{n..m\} \rangle \text{mn}(1)$  by auto
qed
qed
also have  $\dots = (\sum_{l=n..m}. (2^{\text{powr } -\gamma})^l)$ 
  apply (intro  $\text{sum.cong; simp add: field-simps}$ )
  by (smt (verit, ccfv-SIG)  $\text{powr-powr[symmetric] mult-minus-left powr-gt-zero}$ 
 $\text{powr-realpow}$ )
also have  $\dots \leq 2^{\text{powr } (-\gamma * n)} / (1 - 2^{\text{powr } -\gamma})$ 
  apply (subst  $\text{sum-gp}$ )
  using  $\langle m \geq n \rangle$  apply (simp add:  $\text{field-simps}$ )
  apply safe
  using  $\text{gamma-gt-0}$  apply force
  apply (rule  $\text{divide-right-mono}$ )
  apply (simp only:  $\text{minus-mult-left}$ )
  apply (subst  $\text{powr-powr[symmetric]}$ )
  apply (subst  $\text{powr-realpow[symmetric]; simp}$ ) +
  by (metis  $\text{diff-ge-0-iff-ge gamma-gt-0 less-eq-real-def}$ 
 $\text{neg-le-0-iff-le one-le-numeral powr-mono powr-nonneg-iff powr-zero-eq-one}$ )
finally show  $\text{dist } (X \ u \ \omega) \ (X \ t \ \omega) \leq 2^{\text{powr } (-\gamma * \text{real } n)} / (1 - 2^{\text{powr } -\gamma})$ 
  using  $\langle t' (n-1) = u \rangle \langle t' m = t \rangle$  by blast
qed

lemma  $\text{dist-dyadic-fixed}$ :
  assumes  $\text{mn}: n_0 \leq n \leq m$ 

```

and *s-dyadic*: $s \in \text{dyadic-interval-step } m \ 0 \ T$
and *t-dyadic*: $t \in \text{dyadic-interval-step } m \ 0 \ T$
and *st*: $s \leq t \ t - s \leq 1/2^n$
shows $\text{dist } (X \ t \ \omega) \ (X \ s \ \omega) \leq 2 * 2^{\text{powr } (-\gamma * n)} / (1 - 2^{\text{powr } -\gamma})$
proof –
have $n > 0$
using *mn(1) n-zero-nonzero* **by** *linarith*
define *u* **where** $u \equiv \lfloor 2^n * s \rfloor / 2^n$
have $u = \text{Max } (\text{dyadic-interval-step } n \ 0 \ s)$
unfolding *u-def* **using** *dyadic-interval-step-Max[symmetric]* *dyadic-step-geq[OF s-dyadic]*
by *blast*
then have *u-dyadic-n*: $u \in \text{dyadic-interval-step } n \ 0 \ T$
using *dyadic-interval-step-mem dyadic-step-geq dyadic-step-leq s-dyadic u-def*
by *force*

Then, $u \leq s < u + 2^{-n}$ and $u \leq t < u + 2^{1-n}$

have $u \leq s$
unfolding *u-def* **using** *floor-pow2-leq* **by** *blast*
have $s < u + 1/2^n$
unfolding *u-def* **apply** (*simp add: field-simps*)
using *floor-le-iff* **apply** *linarith*
done
then have $s - u < 2/2^n$
using $\langle u \leq s \rangle$ **by** *auto*
then have *dist-us*: $\text{dist } (X \ u \ \omega) \ (X \ s \ \omega) \leq 2^{\text{powr } (-\gamma * \text{real } n)} / (1 - 2^{\text{powr } -\gamma})$
by (*rule dist-dyadic-mn[OF mn s-dyadic u-dyadic-n <u ≤ s>]*)
have $u \leq t$
using $\langle u \leq s \rangle \text{st}(1)$ **by** *linarith*
have $t < u + 2/2^n$
using $\langle s < u + 1/2^n \rangle \text{st}(2)$ **by** *force*
then have $t - u < 2/2^n$
by *force*
then have $\text{dist } (X \ u \ \omega) \ (X \ t \ \omega) \leq 2^{\text{powr } (-\gamma * \text{real } n)} / (1 - 2^{\text{powr } -\gamma})$
by (*rule dist-dyadic-mn[OF mn t-dyadic u-dyadic-n <u ≤ t>]*)
then show $\text{dist } (X \ t \ \omega) \ (X \ s \ \omega) \leq 2 * (2^{\text{powr } (-\gamma * n)}) / (1 - 2^{\text{powr } -\gamma})$
using *dist-us* **by** *metric*
qed

definition $C_0 \equiv 2 * 2^{\text{powr } \gamma} / (1 - 2^{\text{powr } -\gamma})$

lemma *C-zero-ge[simp]*: $C_0 > 0$

by (*smt (verit, ccfv-SIG) C0-def divide-pos-pos gamma-gt-0 powr-eq-one-iff powr-less-mono*)

Klenke (21.9)

Let $s, t \in D$ with $|s - t| \leq \frac{1}{2^n}$. By choosing the minimal $n \geq n_0$ such that $|t - s| \geq 2^{-n}$, we obtain by $\llbracket n_0 \leq ?n; ?n \leq ?m; ?s \in \text{dyadic-interval-step}$

$?m \ 0 \ T; ?t \in \text{dyadic-interval-step } ?m \ 0 \ T; ?s \leq ?t; ?t - ?s \leq 1 / 2^{?n} \rceil$
 $\implies \text{dist } (\text{process } X \ ?t \ \omega) (\text{process } X \ ?s \ \omega) \leq 2 * 2^{\text{powr } (-\gamma * \text{real } ?n)} /$
 $(1 - 2^{\text{powr } -\gamma})$:

$$|X_t(\omega) - X_s(\omega)| \leq C_0 |t - s|^\gamma$$

lemma *dist-dyadic*:

assumes $t: t \in \text{dyadic-interval } 0 \ T$
and $s: s \in \text{dyadic-interval } 0 \ T$
and $st\text{-dist}: \text{dist } t \ s \leq 1 / 2^{\wedge n_0}$
shows $\text{dist } (X \ t \ \omega) (X \ s \ \omega) \leq C_0 * (\text{dist } t \ s)^{\text{powr } \gamma}$
proof (*cases* $s = t$)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
define n **where** $n \equiv \text{LEAST } n. \text{dist } t \ s \geq 1 / 2^{\wedge n}$
have $\text{dist } t \ s > 0$
using *False* **by** *simp*
then have $\exists n. \text{dist } t \ s \geq 1 / 2^{\wedge n}$
by (*metis less-eq-real-def one-less-numeral-iff power-one-over reals-power-lt-ex*
semiring-norm(76))
then have $\text{dist } t \ s \geq 1 / 2^{\wedge n}$
unfolding $n\text{-def}$ **by** (*meson LeastI-ex*)
then have $n \geq n_0$
using *order-trans[OF <dist t s ≥ 1/2^{∧n}> st-dist]*
by (*simp add: field-simps*)
have $\text{dist } t \ s \leq 1 / 2^{\wedge (n-1)}$
proof –
have $n-1 < (\text{LEAST } n. \text{dist } t \ s \geq 1 / 2^{\wedge n})$
using $\langle n_0 \leq n \rangle$ $n\text{-zero-nonzero } n\text{-def}$ **by** *fastforce*
then have $\neg (\text{dist } t \ s \geq 1 / 2^{\wedge (n-1)})$
by (*rule not-less-Least*)
then show *?thesis*
by *auto*
qed
obtain m **where** $m: m \geq n \ s \in \text{dyadic-interval-step } m \ 0 \ T \ t \in \text{dyadic-interval-step}$
 $m \ 0 \ T$
by (*metis dyadic-interval-step-mono linorder-not-le mem-dyadic-interval or-*
der.asym s t)
from $\langle n \geq n_0 \rangle$ **consider** (*eq*) $n = n_0 \mid$ (*gt*) $n > n_0$
using *less-eq-real-def* **by** *linarith*
then show *?thesis*
proof *cases*
case *eq*
consider $t \leq s \mid s \leq t$
by *fastforce*
then have $\text{dist } (X \ t \ \omega) (X \ s \ \omega) \leq 2 * 2^{\text{powr } (-\gamma * n)} / (1 - 2^{\text{powr } -\gamma})$
apply *cases*

```

    apply (subst dist-commute)
    apply (rule dist-dyadic-fixed[OF ⟨n ≥ n₀⟩ m(1,3,2)])
    apply simp
    using dist-real-def eq st-dist apply force
    apply (rule dist-dyadic-fixed[OF ⟨n ≥ n₀⟩ m])
    apply simp
    using dist-real-def eq st-dist apply force
  done
  also have ... ≤ C₀ * 2 powr (− γ * n)
    unfolding C₀-def apply (simp add: field-simps)
    apply (intro divide-right-mono mult-left-mono)
    apply (simp add: less-eq-real-def)
    apply simp
    by (smt (verit) gamma-gt-0 powr-le-cancel-iff powr-zero-eq-one)
  also have ... = C₀ * (1/2^n) powr γ
    by (smt (verit, del-Insts) powr-minus-divide powr-powr powr-powr-swap powr-realpow)
  also have ... ≤ C₀ * (dist t s) powr γ
    using ⟨1 / 2^n ≤ dist t s⟩ eq st-dist by auto
  finally show ?thesis .
next
case gt
consider t ≤ s | s ≤ t
  by fastforce
then have dist (X t ω) (X s ω) ≤ 2 * 2 powr (− γ * (n−1)) / (1 − 2 powr
− γ)
  apply cases
  apply (subst dist-commute)
  apply (rule dist-dyadic-fixed[where m=m])
  prefer 7 apply (rule dist-dyadic-fixed[where m=m])
  using gt m apply simp-all
  using ⟨dist t s ≤ 1 / 2^(n−1)⟩ dist-real-def apply force+
  done
  also have ... ≤ C₀ * 2 powr (− γ * n)
    unfolding C₀-def apply simp
    apply (intro divide-right-mono)
    apply (simp add: powr-add[symmetric])
    apply (metis One-nat-def Suc-leI dual-order.refl gt less-imp-Suc-add mi-
nus-diff-eq mult.right-neutral of-nat-1 of-nat-diff right-diff-distrib zero-less-Suc)
    by (metis gamma-gt-0 ge-iff-diff-ge-0 less-eq-real-def neg-le-0-iff-le one-le-numeral
powr-mono powr-zero-eq-one zero-neq-numeral)
  also have ... = C₀ * (1/2^n) powr γ
    by (smt (verit, best) powr-minus-divide powr-powr powr-powr-swap powr-realpow)
  also have C₀ * (1/2^n) powr γ ≤ C₀ * dist t s powr γ
    apply (rule mult-left-mono)
    using ⟨1 / 2^n ≤ dist t s⟩ less-eq-real-def powr-mono2 apply force
    using C-zero-ge by linarith
  finally show ?thesis .
qed
qed

```

definition $K \equiv C_0 * (2^{\wedge \text{nat}} \lceil 2^{\wedge n_0} * T \rceil) \text{ powr } (1 - \gamma)$

lemma $C_0\text{-le-}K$: $C_0 \leq K$
unfolding $K\text{-def}$ **using** $\text{nzero-ge}[of\ n_0]$ $\text{ge-one-powr-ge-zero}$ **by** *force*

lemma $K\text{-pos}$: $0 < K$
using $C_0\text{-le-}K$ $C\text{-zero-ge}$ **by** *linarith*

Klenke (21.10)

lemma $X\text{-dyadic-le-}K'$:

assumes dyadic : $s \in \text{dyadic-interval } 0\ T$ $t \in \text{dyadic-interval } 0\ T$
and st : $s \leq t$
shows $\text{dist } (X\ s\ \omega)\ (X\ t\ \omega) \leq K * \text{dist } s\ t\ \text{powr } \gamma$
proof (*cases* $\text{dist } s\ t \leq 1 / 2^{\wedge n_0}$)
case *True*
then have $C_0 * \text{dist } t\ s\ \text{powr } \gamma \leq K * \text{dist } t\ s\ \text{powr } \gamma$
by (*simp add: C₀-le-K powr-def*)
then show *?thesis*
using $\text{dist-dyadic}[OF\ \text{assms}(1,2)\ \text{True}]$ **by** (*simp add: dist-commute*)
next
case *False*
define $n :: \text{nat}$ **where** $n \equiv \text{nat } \lceil 2^{\wedge n_0} * T \rceil$
have $\text{dist } s\ t / n \leq 1 / 2^{\wedge n_0}$
apply (*simp add: n-def field-simps*)
by (*smt (verit, best) dyadic dist-real-def divide-le-eq-1 dyadics-geq dyadics-leq mem-dyadic-interval mult-mono of-nat-eq-0-iff of-nat-le-0-iff real-nat-ceiling-ge zero-le-power*)
have dist-st : $\text{dist } s\ t / 2^{\wedge n} \leq 1 / 2^{\wedge n_0}$
apply (*rule order-trans[where y=dist s t / n]*)
apply (*rule divide-left-mono; simp?*)
apply (*simp add: n-def zero-le-T*)
apply (*fact <dist s t / n ≤ 1 / 2[∧]n₀>*)
done
define f **where** $f \equiv \lambda i::\text{nat}. (s + (t-s) * i / 2^{\wedge n})$
have $f\text{-inc}$: $f\ k = f\ (k - 1) + (t - s) / 2^{\wedge n}$ **if** $k > 0$ **for** k
proof –
have $f\ (\text{Suc } k) = f\ k + (t - s) / 2^{\wedge n}$ **for** k
by (*simp add: f-def field-simps*)
then show *?thesis*
by (*metis Suc-pred' that*)
qed
have $f\text{-inc-le}$: $\text{dist } (f\ i)\ (f\ (i - 1)) \leq 1 / 2^{\wedge n_0}$ **for** i
proof (*cases i=0*)
case *True*
then show *?thesis* **by** *simp*
next
case *False*


```

then show ?thesis
  using f-inc dist-real-def dist-st st by auto
qed
have f-ge-s:  $\bigwedge i. i \leq 2^n \implies f i \geq s$ 
  unfolding f-def using st by auto
have f-le-t:  $\bigwedge i. i \leq 2^n \implies f i \leq t$ 
  by (smt (verit, del-ists) f-def st divide-le-eq-1 mult-less-cancel-left1 of-nat-1
of-nat-add
  of-nat-le-iff of-nat-power one-add-one times-divide-eq-right zero-less-power)
have f-dyadic:  $f i \in \text{dyadic-interval } 0 \ T \text{ if } i \leq 2^n \text{ for } i$ 
proof (rule mem-dyadic-intervalI)
  have  $f i \leq T$ 
proof -
  have  $f i \leq s + t - s$ 
  using f-le-t[OF that] by simp
  also have  $\dots \leq T$ 
  using dyadic(2) dyadics-leq by simp
  finally show ?thesis .
qed
obtain m where  $s \in \text{dyadic-interval-step } m \ 0 \ T$   $t \in \text{dyadic-interval-step } m \ 0 \ T$ 
  by (metis dyadic dyadic-interval-step-mono mem-dyadic-interval nle-le)
then obtain ks kt where  $ks: s = \text{real } ks / 2^m$  and  $kt: t = \text{real } kt / 2^m$ 
  using dyadic-as-natural by metis
then have  $ks \leq kt$ 
  using st by (simp add: divide-le-cancel)
from ks kt have  $ks = 2^m * s$   $kt = 2^m * t$ 
  by simp-all
from ks kt have  $f i = (ks / 2^m) + (kt / 2^m - ks / 2^m) * i / 2^n$ 
  unfolding f-def by auto
also have  $\dots = (ks * (2^n - i) + kt * i) / 2^{(m+n)}$ 
  using  $\langle ks \leq kt \rangle$  apply (simp add: right-diff-distrib field-simps power-add)
  by (metis distrib-left le-add-diff-inverse mult.commute of-nat-add of-nat-numeral
of-nat-power that)
finally have  $f i \in \text{dyadic-interval-step } (m+n) \ 0 \ T$ 
  apply (intro dyadic-interval-step-memI)
  apply (rule exI[where  $x = \text{int } (ks * (2^n - i) + kt * i)$ ])
  prefer 3 apply (rule  $\langle f i \leq T \rangle$ )
  by simp-all
then show  $\exists n. f i \in \text{dyadic-interval-step } n \ 0 \ T$ 
  by blast
qed
have  $\text{dist } (X \ s \ \omega) \ (X \ t \ \omega) \leq (\sum_{i=1..2^n} \text{dist } (X \ (f \ i) \ \omega) \ (X \ (f \ (i-1)) \ \omega))$ 
proof -
  have  $f \ 0 = s$ 
  unfolding f-def by (simp add: field-simps)
moreover have  $f \ (2^n) = t$ 
  unfolding f-def by (simp add: field-simps)
moreover have  $\text{dist } (X \ (f \ 0) \ \omega) \ (X \ (f \ (2^n)) \ \omega) \leq (\sum_{i=\text{Suc } 0..2^n} \text{dist } (X \ (f \ i) \ \omega) \ (X \ (f \ (i-1)) \ \omega))$ 

```

$(X (f i) \omega) (X (f (i-1)) \omega)$
 by (rule triangle-ineq-sum, simp)
 ultimately show ?thesis
 by simp
 qed
 also have $\dots \leq (\sum_{i=1..2^n::nat.} C_0 * (dist (f i) (f (i-1)))) \text{ powr } \gamma$
 apply (rule sum-mono)
 apply (intro dist-dyadic f-dyadic)
 apply fastforce
 apply fastforce
 using f-inc-le .
 also have $\dots \leq (\sum_{i=1..2^n::nat.} C_0 * (dist t s / 2^n) \text{ powr } \gamma)$
 apply (rule sum-mono)
 using f-inc by (simp add: dist-real-def)
 also have $\dots \leq 2^n * C_0 * (dist t s / 2^n) \text{ powr } \gamma$
 by (subst sum-constant, force)
 also have $\dots \leq K * dist t s \text{ powr } \gamma$
 unfolding K-def n-def apply (simp add: powr-divide field-simps)
 apply (rule mult-left-mono)
 apply (smt (verit, ccfv-threshold) powr-add powr-one zero-le-power)
 by simp
 finally show ?thesis
 by (metis dist-commute)
 qed

lemma *X-dyadic-le-K*:

assumes $s \in \text{dyadic-interval } 0 \ T$
 and $t \in \text{dyadic-interval } 0 \ T$
 shows $dist (X s \omega) (X t \omega) \leq K * dist s t \text{ powr } \gamma$
 by (metis nle-le assms X-dyadic-le-K' dist-commute)

corollary *holder-dyadic*: γ -holder-on (dyadic-interval 0 T) ($\lambda t. X t \omega$)
 apply (intro holder-onI[OF gamma-0-1] exI[where x=K])
 using K-pos X-dyadic-le-K by force

lemma *uniformly-continuous-dyadic*: uniformly-continuous-on (dyadic-interval 0 T) ($\lambda t. X t \omega$)
 using holder-dyadic by (fact holder-uniform-continuous)

lemma *Lim-exists*: $\exists L. ((\lambda s. X s \omega) \longrightarrow L) \text{ (at } t \text{ within (dyadic-interval } 0 \ T))$
 if $t \in \{0..T\}$
 apply (rule uniformly-continuous-on-extension-at-closure[where x = t])
 using that dyadic-interval-dense uniformly-continuous-dyadic apply fast
 using that apply (simp add: dyadic-interval-dense)
 by blast

lemma *Lim-unique*: $\exists ! L. ((\lambda s. X s \omega) \longrightarrow L) \text{ (at } t \text{ within (dyadic-interval } 0 \ T))$
 if $t \in \{0..T\}$

by (*metis that Lim-exists dyadic-interval-islimgt tendsto-Lim trivial-limit-within zero-le-T*)

definition $L \equiv (\lambda t. (Lim \text{ (at } t \text{ within dyadic-interval } 0 \ T) (\lambda s. X \ s \ \omega)))$

lemma $X\text{-tendsto-}L$:

assumes $t \in \{0..T\}$

shows $((\lambda s. X \ s \ \omega) \longrightarrow L \ t) \text{ (at } t \text{ within (dyadic-interval } 0 \ T))$

proof –

have *at t within dyadic-interval 0 T* $\neq \perp$

by (*simp add: trivial-limit-within dyadic-interval-islimgt[OF zero-le-T assms]*)

moreover obtain L' **where** L' : $((\lambda s. X \ s \ \omega) \longrightarrow L') \text{ (at } t \text{ within (dyadic-interval } 0 \ T))$

using *Lim-exists[OF assms]* **by** *blast*

ultimately have $L \ t = L'$

unfolding $L\text{-def}$ **by** (*rule tendsto-Lim*)

then show *?thesis*

using L' **by** *blast*

qed

lemma $L\text{-dist-}K$:

assumes $s: s \in \{0..T\}$

and $t: t \in \{0..T\}$

shows $dist \ (L \ s) \ (L \ t) \leq K * dist \ s \ t \ powr \ \gamma$

proof (*cases s = t*)

case *True*

then show *?thesis* **by** *simp*

next

case *False*

let $?F = \lambda x. \text{at } x \text{ within dyadic-interval } 0 \ T$

have $(?F \ s \times_F \ ?F \ t) \neq \perp$

by (*meson dyadic-interval-islimgt prod-filter-eq-bot s t trivial-limit-within zero-le-T*)

moreover have $((\lambda x. K * dist \ (fst \ x) \ (snd \ x) \ powr \ \gamma) \longrightarrow K * dist \ s \ t \ powr \ \gamma) \text{ (} ?F \ s \times_F \ ?F \ t \text{)}$

apply (*rule tendsto-mult-left*)

apply (*rule tendsto-powr*)

using *tendsto-dist-prod* **apply** *blast*

apply *simp*

using *False* **by** *simp*

moreover have $((\lambda x. dist \ (X \ (fst \ x) \ \omega) \ (X \ (snd \ x) \ \omega)) \longrightarrow dist \ (L \ s) \ (L \ t))$

$(?F \ s \times_F \ ?F \ t)$

using $X\text{-tendsto-}L \ t \ s \text{ tendsto-dist-prod}$ **by** *blast*

moreover have $\forall_F \ x \text{ in } ?F \ s \times_F \ ?F \ t. dist \ (\text{process } X \ (fst \ x) \ \omega) \ (\text{process } X \ (snd \ x) \ \omega)$

$\leq K * dist \ (fst \ x) \ (snd \ x) \ powr \ \gamma$

apply (*rule eventually-prodI'*) **where** $P = \lambda x. x \in \text{dyadic-interval } 0 \ T$

and $Q = \lambda x. x \in \text{dyadic-interval } 0 \ T]$

using *eventually-at-topological* **apply** *blast*

using *eventually-at-topological* **apply** *blast*

using X -dyadic-le- K by simp
 ultimately show ?thesis
 by (rule tendsto-le)
 qed

corollary L -holder: γ -holder-on $\{0..T\}$ L
 using K -pos L -dist- K by (auto intro!: holder-onI[OF gamma-0-1] exI[where $x=K$])

corollary L -local-holder: local-holder-on γ $\{0..T\}$ L
 using holder-implies-local-holder[OF L -holder] by blast

lemma X -dyadic-eq- L :
 assumes $t \in \text{dyadic-interval } 0 \ T$
 shows $X \ t \ \omega = L \ t$
proof –
 have $((\lambda x. X \ x \ \omega) \longrightarrow X \ t \ \omega)$ (at t within dyadic-interval $0 \ T$)
 using continuous-within[symmetric] uniformly-continuous-dyadic uniformly-continuous-imp-continuous
 continuous-on-eq-continuous-within assms by fast
 then show ?thesis
 by (metis L -def assms dyadic-interval-islimpt dyadic-interval-subset-interval
 subsetD
 tendsto-Lim trivial-limit-within zero-le- T)
 qed
 end

definition default :: 'b where default = (SOME x . True)

definition X -tilde :: real $\Rightarrow$ 'a $\Rightarrow$ 'b where
 $X\text{-tilde} \equiv (\lambda t \ \omega. \text{if } \omega \in N \text{ then default else } (Lim \ (at \ t \text{ within dyadic-interval } 0 \ T) \ (\lambda s. X \ s \ \omega)))$

lemma X -tilde-not- N -Lim:
 assumes $\omega \in \text{space source} - N$
 shows $X\text{-tilde} \ t \ \omega = Lim \ (at \ t \text{ within dyadic-interval } 0 \ T) \ (\lambda s. X \ s \ \omega)$
 using assms X -tilde-def by auto

lemma X -tilde-not- N - L :
 assumes $\omega \in \text{space source} - N$
 shows $X\text{-tilde} \ t \ \omega = L \ \omega \ t$
 using assms X -tilde-def L -def[OF assms] by auto

lemma local-holder- X -tilde: local-holder-on γ $\{0..T\}$ $(\lambda t. X\text{-tilde} \ t \ \omega)$
 if $\omega \in \text{space source}$ for ω
proof (cases $\omega \in N$)
 case True
 then show ?thesis
 unfolding X -tilde-def using local-holder-const by fastforce
 next

```

case False
then have 1:  $\omega \in \text{space source} - N$ 
  using that by blast
show ?thesis
  using L-local-holder[OF 1] X-tilde-not-N-L[OF 1]
  by (simp only: False if-False)
qed

corollary X-tilde-eq-L-AE: AE  $\omega$  in source. X-tilde  $t \ \omega = L \ \omega \ t$ 
  apply (rule AE-I[where N=N])
  apply (smt (verit, del-Insts) X-tilde-def Diff-iff L-def mem-Collect-eq subsetI)
  using N-null apply blast+
done

corollary X-tilde-eq-Lim-AE:
  AE  $\omega$  in source. X-tilde  $t \ \omega = \text{Lim} \ (\text{at } t \text{ within dyadic-interval } 0 \ T) \ (\lambda s. X \ s \ \omega)$ 
  apply (rule AE-I[where N=N])
  apply (smt (verit, del-Insts) X-tilde-def Diff-iff L-def mem-Collect-eq subsetI)
  using N-null apply blast+
done

lemma X-tilde-tendsto-AE:  $t \in \{0..T\} \implies \text{tendsto-AE source } X \ (X\text{-tilde } t) \ (\text{at } t \text{ within dyadic-interval } 0 \ T)$ 
  apply (unfold tendsto-AE-def)
  apply (rule AE-I3[where N=N])
  apply (subst X-tilde-not-N-Lim, argo)
  unfolding t2-space-class.Lim-def apply (rule the1I2)
  using Lim-unique apply presburger
  apply blast
  using N-null by blast

end

context Kolmogorov-Chentsov-finite
begin

By (21.5)  $0 \leq ?t \implies \text{tendsto-measure source } (\text{process } X) \ (\text{process } X \ ?t) \ (\text{at } ?t \text{ within } \{0..?T\})$  and (21.11)  $? \omega \in \text{space source} - (\bigcap_n \bigcup_m A \ (m + n)) \implies L \ ? \omega \equiv \lambda t. \text{Lim} \ (\text{at } t \text{ within dyadic-interval } 0 \ T) \ (\lambda s. \text{process } X \ s \ ? \omega), P[X \neq \tilde{X}] = 0$ 

lemma X-tilde-measurable[measurable]:
  assumes  $t \in \{0..T\}$ 
  shows X-tilde  $t \in \text{borel-measurable source}$ 
proof -
  let ?Lim =  $(\lambda \omega. \text{Lim} \ (\text{at } t \text{ within dyadic-interval } 0 \ T) \ (\lambda s. \text{process } X \ s \ \omega))$ 
  have ?Lim  $\in \text{borel-measurable} \ (\text{restrict-space source } (\text{space source} - N))$ 
  unfolding X-tilde-def apply measurable
  using measurable-id measurable-restrict-space1 apply blast
  using assms Lim-exists space-restrict-space apply simp

```

```

using assms dyadic-interval-islimgt trivial-limit-within zero-le-T by blast
then have  $(\lambda\omega. \text{if } \omega \in \text{space source} - N \text{ then } ?Lim \omega \text{ else default}) \in \text{borel-measurable}$ 
source
by (subst measurable-restrict-space-iff[symmetric]; simp)
then show ?thesis
apply (subst measurable-cong[where g=( $\lambda\omega. \text{if } \omega \in \text{space source} - N \text{ then } ?Lim \omega \text{ else default}$ ))])
unfolding X-tilde-def by auto
qed

```

```

lemma X-eq-X-tilde-AE: AE  $\omega$  in source.  $X t \omega = X\text{-tilde } t \omega$  if  $t \in \{0..T\}$  for  $t$ 
apply (rule sigma-finite-measure.tendsto-measure-at-within-eq-AE[where f=process
X and S=dyadic-interval 0 T])
using proc-source.sigma-finite-measure-axioms apply blast
using X-borel-measurable apply blast
apply measurable
using X-tilde-def apply simp
using that apply simp
using tendsto-measure-mono[OF at-le[OF dyadic-interval-subset-interval] conv-in-prob-finite]
that
apply force
using X-borel-measurable X-tilde-measurable X-tilde-tendsto-AE measure-conv-imp-AE-at-within
that apply blast
using dyadic-interval-islimgt that trivial-limit-within zero-le-T by blast

```

```

lemma X-tilde-modification: modification (restrict-index X {0..T})
(prob-space.process-of source (proc-target X) {0..T} X-tilde default)
apply (intro modificationI compatibleI)
apply simp-all
apply (subst restrict-index-source)
apply (auto simp: X-eq-X-tilde-AE)
done
end

```

We have now shown that we can construct a modification of X for any interval $\{0..T\}$. We want to extend this result to construct a modification on the interval $\{0..\}$ - this can be constructed by gluing together all modifications with natural-valued T which results in a countable union of modifications, which itself is a modification.

```

context Kolmogorov-Chentsov
begin

```

```

lemma Kolmogorov-Chentsov-finite:  $T > 0 \implies \text{Kolmogorov-Chentsov-finite } X a$ 
 $b C \gamma T$ 
by (simp add: Kolmogorov-Chentsov-axioms Kolmogorov-Chentsov-finite.intro Kol-
mogorov-Chentsov-finite-axioms-def)

```

```

definition Mod  $\equiv \lambda T. \text{SOME } Y. \text{modification (restrict-index X {0..T}) } Y \wedge$ 
 $(\forall x \in \text{space source. local-holder-on } \gamma \{0..T\} (\lambda t. Y t x))$ 

```

lemma *Mod: modification (restrict-index X {0..T}) (Mod T)*
 $(\forall x \in \text{space source. local-holder-on } \gamma \{0..T\} (\lambda t. (\text{Mod } T) t x))$ **if** $0 < T$ **for** T
proof –
interpret *Kolmogorov-Chentsov-finite X a b C γ T*
using *that by (simp add: Kolmogorov-Chentsov-finite)*
have *modification (restrict-index X {0..T}) (Mod T) $\wedge$*
 $(\forall x \in \text{space source. local-holder-on } \gamma \{0..T\} (\lambda t. (\text{Mod } T) t x))$
unfolding *Mod-def* **apply** *(rule someI-ex)*
apply *(rule exI[where x=prob-space.process-of source (proc-target X) {0..T}*
X-tilde default])
apply *safe*
apply *(fact X-tilde-modification)*
apply *(subst local-holder-on-cong[OF refl refl])*
using *local-holder-X-tilde X-tilde-measurable* **apply** *(auto cong: local-holder-on-cong)*
done
then show *modification (restrict-index X {0..T}) (Mod T)*
 $(\forall x \in \text{space source. local-holder-on } \gamma \{0..T\} (\lambda t. (\text{Mod } T) t x))$
by *blast+*
qed

lemma *compatible-Mod: compatible (restrict-index X {0..T}) (Mod T) if $0 < T$*
for T
using *Mod that modificationD(1) by blast*

lemma *Mod-source[simp]: proc-source (Mod T) = source if $0 < T$ for T*
by *(metis compatible-Mod compatible-source restrict-index-source that)*

lemma *Mod-target: sets (proc-target (Mod T)) = sets (proc-target X) if $0 < T$*
for T
by *(metis compatible-Mod[OF that] compatible-target restrict-index-target)*

lemma *Mod-index[simp]: $0 < T \implies \text{proc-index (Mod T)} = \{0..T\}$*
using *compatible-Mod[THEN compatible-index] by simp*

lemma *indistinguishable-mod:*
indistinguishable (restrict-index (Mod S) {0..min S T}) (restrict-index (Mod T)
 $\{0..min S T\})$
if $S > 0$ $T > 0$ **for** S T
proof –
have **: modification (restrict-index (Mod S) {0..min S T}) (restrict-index (Mod*
 $T) \{0..min S T\})$
proof –
have *modification (restrict-index X {0..min S T}) (restrict-index (Mod S)*
 $\{0..min S T\})$
apply *(cases S $\leq$ T)*
using *Mod(1)[OF that(1)]* **apply** *clarsimp*
apply *(metis modification-restrict-index order-refl restrict-index-index re-*
strict-index-override)

```

    using Mod(1)[OF that(2)] apply clarsimp
    by (metis (mono-tags, opaque-lifting) ‹modification (restrict-index X {0..S})
    (Mod S)› atLeastatMost-subset-iff modification-restrict-index nle-le restrict-index-index
    restrict-index-override)
    moreover have modification (restrict-index X {0..min S T}) (restrict-index
    (Mod T) {0..min S T})
    apply (cases S ≤ T)
    using Mod(1)[OF that(1)] apply clarsimp
    apply (metis Mod(1) atLeastatMost-subset-iff modification-restrict-index
    order.refl restrict-index-index restrict-index-override that(2))
    using Mod(1)[OF that(2)] apply clarsimp
    by (metis (mono-tags, opaque-lifting) atLeastatMost-subset-iff modification-restrict-index
    nle-le restrict-index-index restrict-index-override)
    ultimately show ?thesis
    using modification-sym modification-trans by metis
qed
then show ?thesis
proof (rule modification-continuous-indistinguishable,
    goal-cases - continuous-S continuous-T)
    show ∃ Ta>0. proc-index (restrict-index (Mod S) {0..min S T}) = {0..Ta}
    by (metis that min-def restrict-index-index)
next
case (continuous-S)
have ∀ x ∈ space source. continuous-on {0..S} (λt. (Mod S) t x)
    using Mod[OF that(1)] local-holder-imp-continuous by blast
then have ∀ x ∈ space source. continuous-on {0..min S T} (λt. (Mod S) t x)
    using continuous-on-subset by fastforce
then show ?case
    by (auto simp: that(1))
next
case (continuous-T)
have ∀ x ∈ space source. continuous-on {0..T} (λt. (Mod T) t x)
    using Mod[OF that(2)] local-holder-imp-continuous by fast
then have 2: ∀ x ∈ space source. continuous-on {0..min S T} (λt. (Mod T) t
x)
    using continuous-on-subset by fastforce
then show ?case
    by (auto simp: that(2))
qed
qed

```

definition $N S T \equiv \text{SOME } N. N \in \text{null-sets source} \wedge \{\omega \in \text{space source}. \exists t \in \{0..\min S T\}. (\text{Mod } S) t \omega \neq (\text{Mod } T) t \omega\} \subseteq N$

lemma N :

```

    assumes S > 0 T > 0
    shows N S T ∈ null-sets source ∧ {ω ∈ space source. ∃ t ∈ {0..min S T}. (Mod
    S) t ω ≠ (Mod T) t ω} ⊆ N S T
    proof –

```


have $ex: \forall S > 0. \forall T > 0. \exists N. N \in \text{null-sets source} \wedge \{\omega \in \text{space source}. \exists t \in \{0..\min S T\}.$
 $(\text{Mod } S) t \omega \neq (\text{Mod } T) t \omega\} \subseteq N$
apply (fold Bex-def)
using indistinguishable-mod[THEN indistinguishable-null-ex] **by** simp
show ?thesis
unfolding N-def **apply** (rule someI-ex)
using ex assms **by** blast
qed

definition $N\text{-inf}$ **where** $N\text{-inf} \equiv (\bigcup S \in \mathbb{N} - \{0\}. (\bigcup T \in \mathbb{N} - \{0\}. N S T))$

lemma $N\text{-inf-null}$: $N\text{-inf} \in \text{null-sets source}$
unfolding $N\text{-inf-def}$
apply (intro null-sets-UN')
apply (rule countable-Diff)
apply (simp add: Nats-def)+
using N **by** force

lemma $\text{Mod-eq-}N\text{-inf}$: $(\text{Mod } S) t \omega = (\text{Mod } T) t \omega$
if $\omega \in \text{space source} - N\text{-inf } t \in \{0..\min S T\} S \in \mathbb{N} - \{0\} T \in \mathbb{N} - \{0\}$ **for**
 $\omega t S T$
proof –
have $\omega \in \text{space source} - N S T$
using that(1,3,4) **unfolding** $N\text{-inf-def}$ **by** blast
moreover **have** $S > 0 T > 0$
using that(2,3,4) **by** auto
ultimately **have** $\omega \in \{\omega \in \text{space source}. \forall t \in \{0..\min S T\}. (\text{Mod } S) t \omega =$
 $(\text{Mod } T) t \omega\}$
using N **by** auto
then **show** ?thesis
using that(2) **by** blast
qed

definition $\text{default} :: 'b$ **where** $\text{default} = (\text{SOME } x. \text{True})$

definition $X\text{-mod} \equiv \lambda t \omega. \text{if } \omega \in \text{space source} - N\text{-inf} \text{ then } (\text{Mod } \lfloor t+1 \rfloor) t \omega \text{ else default}$

definition $X\text{-mod-process} \equiv \text{prob-space.process-of source (proc-target } X) \{0..\} X\text{-mod default}$

lemma $\text{Mod-measurable}[\text{measurable}]$: $\forall t \in \{0..\}. X\text{-mod } t \in \text{source} \rightarrow_M \text{proc-target } X$
proof (intro ballI)
fix $t :: \text{real}$ **assume** $t \in \{0..\}$
then **have** $0 < \lfloor t + 1 \rfloor$
by force
then **show** $X\text{-mod } t \in \text{source} \rightarrow_M \text{proc-target } X$

```

    unfolding X-mod-def apply measurable
    apply (subst measurable-cong-sets[where  $M' = \text{proc-source } (\text{Mod } \lfloor t + 1 \rfloor)$ ])
and  $N' = \text{proc-target } (\text{Mod } \lfloor t + 1 \rfloor)$ 
    using Mod-source  $\langle 0 < \lfloor t + 1 \rfloor \rangle$  apply presburger
    using Mod-target  $\langle 0 < \lfloor t + 1 \rfloor \rangle$  apply presburger
    apply (meson random-process)
    apply simp
    apply (metis N-inf-null Int-def null-setsD2 sets.Int-space-eq1 sets.compl-sets)
  done
qed

```

lemma *modification-X-mod-process: modification X X-mod-process*

proof (intro modificationI ballI)

show compatible X X-mod-process

apply (intro compatibleI)

unfolding X-mod-process-def

apply (subst proc-source.source-process-of)

using Mod-measurable proc-source.prob-space-axioms **apply** auto

done

fix t **assume** $t \in \text{proc-index } X$

then have $t \in \{0..\}$

by simp

then have $\text{real-of-int } \lfloor t \rfloor + 1 > 0$

by (simp add: add.commute add-pos-nonneg)

then have $\exists N \in \text{null-sets source}. \forall \omega \in \text{space source} - N.$

$X \ t \ \omega = (\text{prob-space.process-of source } (\text{proc-target } X) \ \{0..\} \ X\text{-mod default}) \ t$

ω

proof –

have 1: $(\text{prob-space.process-of source } (\text{proc-target } X) \ \{0..\} \ X\text{-mod default}) \ t \ \omega$

$= (\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1)) \ t \ \omega$

if $\omega \in \text{space source} - N\text{-inf}$ **for** ω

apply (subst proc-source.process-process-of)

apply measurable

unfolding X-mod-def **using** that $\langle t \in \{0..\} \rangle$ **apply** simp

done

have $\exists N \in \text{null-sets source}. \forall \omega \in \text{space source} - N. X \ t \ \omega = (\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1)) \ t \ \omega$

proof –

obtain N **where** $\{x \in \text{space source}. (\text{restrict-index } X \ \{0..\text{real-of-int } \lfloor t \rfloor + 1\}) \ t \ x \neq (\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1)) \ t \ x\} \subseteq N \wedge$

$N \in \text{null-sets } (\text{proc-source } (\text{restrict-index } X \ \{0..\text{real-of-int } \lfloor t \rfloor + 1\}))$

using Mod(1)[OF $\langle \text{real-of-int } \lfloor t \rfloor + 1 > 0 \rangle$, THEN modification-null-set, of t]

using $\langle t \in \{0..\} \rangle$ **by** fastforce

then show ?thesis

by (smt (verit, ccfv-threshold) DiffE mem-Collect-eq restrict-index-process restrict-index-source subset-eq)

qed

then obtain N **where** $N \in \text{null-sets source } \forall \omega \in \text{space source} - N. X \ t \ \omega =$

```

(Mod (real-of-int  $\lfloor t \rfloor + 1$ ))  $t \omega$ 
  by blast
  then show ?thesis
    apply (intro bestI [where  $x = N \cup N\text{-inf}$ ])
    apply (metis 1 DiffE DiffI UnCI)
    using N-inf-null by blast
  qed
  then show AE  $x$  in source.  $X \ t \ x = X\text{-mod-process } t \ x$ 
    by (smt (verit, del-insts) X-mod-process-def DiffI eventually-ae-filter mem-Collect-eq
subsetI)
  qed

lemma local-holder-X-mod: local-holder-on  $\gamma \ \{0..\}$  ( $\lambda t. \ X\text{-mod } t \ \omega$ ) for  $\omega$ 
proof (cases  $\omega \in \text{space } \text{source} - N\text{-inf}$ )
  case False
  then show ?thesis
    apply (simp only: X-mod-def)
    apply (metis local-holder-const gamma-0-1)
    done
  next
  case True
  then have  $\omega: \omega \in \text{space } \text{source} \ \omega \notin N\text{-inf}$ 
    by simp-all
  then show ?thesis
  proof (intro local-holder-ballI [OF gamma-0-1] ballI)
    fix  $t::\text{real}$  assume  $t \in \{0..\}$ 
    then have real-of-int  $\lfloor t \rfloor + 1 > 0$ 
      using floor-le-iff by force
    have  $t < \text{real-of-int } \lfloor t \rfloor + 1$ 
      by simp
    then have local-holder-on  $\gamma \ \{0..\text{real-of-int } \lfloor t \rfloor + 1\}$  ( $\lambda s. \ \text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1) \ s \ \omega$ )
      using Mod(2)  $\omega(1) \ \langle \text{real-of-int } \lfloor t \rfloor + 1 > 0 \rangle$  by blast
    then obtain  $\varepsilon \ C$  where  $eC: \varepsilon > 0 \ C \geq 0 \ \bigwedge r \ s. r \in \text{ball } t \ \varepsilon \cap \{0..\text{real-of-int } \lfloor t \rfloor + 1\} \implies$ 
       $s \in \text{ball } t \ \varepsilon \cap \{0..\text{real-of-int } \lfloor t \rfloor + 1\} \implies$ 
       $\text{dist } (\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1) \ r \ \omega) \ (\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1) \ s \ \omega) \leq C * \text{dist } r \ s \ \text{powr } \gamma$ 
      by fastforce
    apply –
    apply (erule local-holder-onE)
    apply (rule gamma-0-1)
    using  $\langle t \in \{0..\} \rangle \ \langle t < \text{real-of-int } \lfloor t \rfloor + 1 \rangle$  apply fastforce
    apply blast
    done
  define  $\varepsilon'$  where  $\varepsilon' = \min \varepsilon$  (if frac  $t = 0$  then  $1/2$  else  $1 - \text{frac } t$ )
  have  $e': \varepsilon' \leq \varepsilon \wedge \varepsilon' > 0 \wedge \text{ball } t \ \varepsilon' \cap \{0..\} \subseteq \{0..\text{real-of-int } \lfloor t \rfloor + 1\}$ 
    unfolding  $\varepsilon'\text{-def}$  apply (simp add: eC(1))
    apply safe
    apply (simp-all add: eC(1) dist-real-def frac-lt-1 frac-floor)

```

```

    apply argo+
  done
{
  fix r s
  assume r:  $r \in \text{ball } t \ \varepsilon' \cap \{0..\}$ 
  assume s:  $s \in \text{ball } t \ \varepsilon' \cap \{0..\}$ 

  then have rs-ball:  $r \in \text{ball } t \ \varepsilon \cap \{0..\text{real-of-int } \lfloor t \rfloor + 1\}$ 
     $s \in \text{ball } t \ \varepsilon \cap \{0..\text{real-of-int } \lfloor t \rfloor + 1\}$ 
    using e' r s by auto
  then have  $r \in \{0..\min(\text{real-of-int } \lfloor t \rfloor + 1)(\text{real-of-int } \lfloor r + 1 \rfloor)\}$ 
    by auto
  then have  $(\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1)) \ r \ \omega = X\text{-mod } r \ \omega$ 
    unfolding X-mod-def apply (simp only: True if-True)
    apply (intro Mod-eq-N-inf[OF True])
    apply simp
    using  $\langle t \in \{0..\} \rangle$  by auto
    (metis (no-types, opaque-lifting) floor-in-Nats Nats-1 Nats-add Nats-cases
      of-int-of-nat-eq of-nat-in-Nats, linarith)+
  moreover have  $(\text{Mod } (\text{real-of-int } \lfloor t \rfloor + 1)) \ s \ \omega = X\text{-mod } s \ \omega$ 
    unfolding X-mod-def apply (simp only: True if-True)
    apply (intro Mod-eq-N-inf[OF True])
    using  $\langle s \in \text{ball } t \ \varepsilon \cap \{0..\text{real-of-int } \lfloor t \rfloor + 1\} \rangle$  apply simp
    using  $\langle t \in \{0..\} \rangle$  apply safe
    apply (metis (no-types, opaque-lifting) floor-in-Nats Nats-1 Nats-add
      Nats-cases of-int-of-nat-eq of-nat-in-Nats)
    apply linarith
    apply (smt (verit) Int-iff Nats-1 Nats-add Nats-altdef1 atLeast-iff mem-Collect-eq
      s zero-le-floor)
    apply (metis Int-iff atLeast-iff floor-correct linorder-not-less one-add-floor
      s)
  done
  ultimately have  $\text{dist } (X\text{-mod } r \ \omega) \ (X\text{-mod } s \ \omega) \leq C * \text{dist } r \ s \ \text{powr } \gamma$ 
    using eC(3)[OF rs-ball] by simp
}
then show  $\exists \varepsilon > 0. \exists C \geq 0. \forall r \in \text{ball } t \ \varepsilon \cap \{0..\}. \forall s \in \text{ball } t \ \varepsilon \cap \{0..\}. \text{dist } (X\text{-mod } r \ \omega) \ (X\text{-mod } s \ \omega) \leq C * \text{dist } r \ s \ \text{powr } \gamma$ 
  using e' eC(2) by blast
qed
qed

lemma local-holder-X-mod-process: local-holder-on  $\gamma \ \{0..\}$   $(\lambda t. X\text{-mod-process } t \ \omega)$ 
for  $\omega$ 
  unfolding X-mod-process-def
  by (smt (verit, best) Mod-measurable UNIV-I local-holder-X-mod local-holder-on-cong
    proc-source.process-process-of space-borel target-borel)

```

theorem continuous-modification:

$\exists X'. \text{modification } X \ X' \wedge (\forall \omega. \text{local-holder-on } \gamma \ \{0..\} \ (\lambda t. X' \ t \ \omega))$

```

using local-holder-X-mod-process modification-X-mod-process by blast

end

theorem Kolmogorov-Chentsov:
  fixes  $X :: (\text{real}, 'a, 'b :: \text{polish-space}) \text{ stochastic-process}$ 
  and  $a\ b\ C\ \gamma :: \text{real}$ 
  assumes index[simp]: proc-index  $X = \{0..\}$ 
  and target-borel[simp]: proc-target  $X = \text{borel}$ 
  and gt-0:  $a > 0\ b > 0\ C > 0$ 
  and b-leq-a:  $b \leq a$ 
  and gamma:  $\gamma \in \{0 < .. < b/a\}$ 
  and expectation:  $\bigwedge s\ t. \llbracket s \geq 0; t \geq 0 \rrbracket \implies$ 
    
$$\left(\int^+ x. \text{dist } (X\ t\ x) (X\ s\ x) \text{ powr } a\ \partial \text{proc-source } X\right) \leq C * \text{dist } t\ s\ \text{powr } (1+b)$$

  shows  $\exists X'. \text{modification } X\ X' \wedge (\forall \omega. \text{local-holder-on } \gamma\ \{0..\}\ (\lambda t. X'\ t\ \omega))$ 
  using Kolmogorov-Chentsov.continuous-modification Kolmogorov-Chentsov.intro[OF
assms]
  by blast
end

```

References

- [1] A. Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2020.