Analyzing Discrete-Time Markov Chains with Countable State Space in Isabelle/HOL

Johannes Hölzl

September 23, 2013

Abstract

Discrete-time Markov chains are an important tool in probabilistic analysis of computer systems. For example they are used to describe the behavior of computer programs with probabilistic choice or the time-dependent distribution of input values. Current formalizations of Markov chains are restricted to a finite state space. We extend this to a countable state space, construct the stochastic process of a Markov chain given a matrix of transition probabilities, and prove the equivalence with the axiomatic definition as stochastic process. Based on this we introduce irreducible, recurrent, and aperiodic classes, generating functions and stationary distributions to analyze Markov chains.

This document is generated from the theory files. The main document can be found in [1].

Contents

1 Introduction

The examples from Fig. 1 (a) and (b) in [1] are formalized in Section 6 and Section 7.

1.1 Constant table

In the paper [1] some of the constants where renamed for presentation purposes.

Paper	**Isabelle/HOL theories**
$\mathbb{B}$	*bool*
$\mathbb{N}$	*nat*
$\mathbb{R}$	*real*
$\overline{\mathbb{R}}$	*ereal*
$\omega \circ_n \omega$	*comb-seq* $n\ \omega\ \omega$
$s{\cdot}\omega$	*nat-case* $s\ \omega$
$\chi_A\ x$	*indicator* $A\ x$
$f'\ z \xrightarrow{z \to 1^-} y$	$((\lambda z.\ f'\ z)\ {-}{-}{-}{>}\ y)$ (*at-left 1*)
$X\ n \xrightarrow{n \to \infty} y$	$(\lambda n.\ X\ n)\ {-}{-}{-}{-}{>}\ y$
count	*count-space*
point	*point-measure*
$\Pi_{i \in I} M\ i$	$Pi_M\ I\ (\lambda i.\ M\ i)$
trace	*path*
$\mathcal{P}_x$	*paths* x
$(x, y) \in$ *accessible*	*accessible* $x\ y$
essential	*essential-class*
G_g	*gf-G*
M_g	*gf-U'*
U_g	*gf-U*

1.2 List of Definitions and Theorems

- `locale` Discrete-Time-Markov-Chain – Section 3
- `locale` Discrete-Markov-Kernel – Section 2
- Definition of *trace* – *local.path* in Section 2.3
- Definition 1 (Trace space) $\mathcal{P}_x$ – *paths* in Section 2.4
- Lemma 2 (Splitting rules) in Section 2.5
- Lemma 3 (Chapman-Kolmogorov) in Section 2.6

- Theorem 4 (The trace space $\mathcal{P}_x$ is a Markov chain) in Section 4
- Definition 5 (Markov kernel from the stochastic process) in Section 3.1
- Theorem 6 (The Markov kernel K' has the distribution of X) in Section 3.3
- Definition 7 (Enabled states) in Section 2.1
- Theorem 8 in Section 2.1.1
- Definition 9 (Accessible states) in Section 5.1
- Theorem 10. in Section 5.3.1 (In the paper p is unfolded)
- Definition 11 (Communicating states) in Section 5.2
- Definition 12 (Essential class). in Section 5.7
- Definition 13 (Recurrent states) in Section 5.6
- Definition 14 (Hitting time) in Section 5.8
- Definition 15 (Average hitting time) is does not have it's own constant
- Definition 16 (Positive recurrent states) in Section 5.9
- Definition 17 (Unbound quantities) in Section 5.4 and Section sec:recurrent
- Definition 18 (Time-bounded reachability probabilities) in Section 5.3
- Lemma 19 (Unbound quantities as series of time-bounded probabilities) in Section 5.4.1, 5.4.2, and 5.8.1
- Definition 20 (Generating functions for G, M , U, and F) in Section 5.5 and 5.8
- Lemma 21 (Unbound quantities as limits of generating functions) in Section 5.5 and 5.8.2
- Theorem 22 (Relating recurrent and G) in Section 5.6.1 and 5.5.1
- Corollary 23 (recurrent is invariant on irreducible classes) in Section 5.6.2
- Lemma 24 (Relation between recurrent and H and U) in Section 5.6 and 5.6.3
- Theorem 25 (Recurrent classes are essential) in Section 5.7
- Corollary 26 (Finite essential classes are recurrent) in Section 5.7

- Lemma 27 (Relation between M and U) in Section 5.8.2
- Corollary 28 (Positive recurrent is invariant on irreducible classes) in Section 5.9
- Definition 29 (Stationary distribution) in Section 5.10
- Theorem 31 (Stationary distribution implies positive recurrence) in Section 5.10.1
- Definition 32 (Aperiodic classes) in Section 5.11
- Lemma 33 (Aperiodic class implies non-zero p) in Section 5.11.1
  Here instead of the quantifiers $\exists N.\ \forall n \geq N.\ P\ n$ we use *eventually P sequentially*
- Theorem 34 (Stationary distribution is asymptotic distribution) in Section 5.13

theory *Discrete-Markov-Kernel*
imports *Auxiliary*
begin

2 Discrete Markov Kernel

locale *Discrete-Markov-Kernel* =
fixes S :: *'s set* **and** K :: *'s* ⇒ *'s measure*
assumes *countable-space*[*simp*]: *countable S*
assumes *non-empty-space*: $S \neq \{\}$
assumes *sets-eq-S*[*simp*]: $\bigwedge s.$ *sets* $(K\ s)$ = *Pow S*
assumes *markov-kernel*: $\bigwedge s.$ *prob-space* $(K\ s)$
begin

abbreviation
$D \equiv (\Pi_M\ s{\in}S.\ K\ s)$

abbreviation
D-seq $\equiv (\Pi_M\ n{\in}UNIV :: nat\ set.\ D)$

sublocale K: *prob-space K s* **for** *s* ⟨*proof*⟩

sublocale D: *product-prob-space* $\lambda s.\ K\ s\ S$ ⟨*proof*⟩

sublocale P: *sequence-space D* ⟨*proof*⟩

definition $s0 = (SOME\ s.\ s \in S)$

lemma $s0$: $s0 \in S$
⟨*proof*⟩

lemma *space-eq-S*[*simp*]: $space\ (K\ s) = S$
⟨*proof*⟩

lemma *K-measurable1*: $measurable\ (K\ s)\ M = measurable\ (count\text{-}space\ S)\ M$
⟨*proof*⟩

lemma *K-measurable2*: $measurable\ M\ (K\ s) = measurable\ M\ (count\text{-}space\ S)$
⟨*proof*⟩

lemma *K-measurable1-imp*[*measurable* (*raw*)]:
$f \in measurable\ (count\text{-}space\ S)\ M \Longrightarrow f \in measurable\ (K\ s)\ M$
⟨*proof*⟩

lemma *K-measurable2-imp*[*measurable* (*raw*)]:
$f \in measurable\ M\ (count\text{-}space\ S) \Longrightarrow f \in measurable\ M\ (K\ s)$
⟨*proof*⟩

lemma *AE-all-S*: $(\bigwedge s.\ s \in S \Longrightarrow AE\ x\ in\ M.\ P\ s\ x) \Longrightarrow AE\ x\ in\ M.\ \forall s{\in}S.\ P\ s\ x$
⟨*proof*⟩

lemma *K-eq-density*:
assumes [*simp*]: $s \in S$ **shows** $K\ s = density\ (count\text{-}space\ S)\ (\lambda x.\ emeasure\ (K\ s)\ \{x\})$
⟨*proof*⟩

2.1 Enabled states

definition $E\ s = \{s' \in S.\ emeasure\ (K\ s)\ \{s'\} \neq 0\}$

lemma *E-subset-S*[*simp*]: $E\ s \subseteq S$
⟨*proof*⟩

lemma *E-in-S*[*dest*]: $s' \in E\ s \Longrightarrow s' \in S$
⟨*proof*⟩

lemma *measurable-E*[*measurable* (*raw*)]:
assumes f[*measurable*]: $f \in measurable\ M\ (count\text{-}space\ S)$
and g[*measurable*]: $g \in measurable\ M\ (count\text{-}space\ S)$
shows $(\lambda x.\ f\ x \in E\ (g\ x)) \in measurable\ M\ (count\text{-}space\ UNIV)$
⟨*proof*⟩

lemma *AE-K-iff*:
assumes [*simp*]: $s \in S$

shows $(AE\ t\ in\ K\ s.\ P\ t) \longleftrightarrow (\forall t{\in}E\ s.\ P\ t)$
⟨*proof*⟩

lemma *AE-enabled*:
assumes [*simp*]: $s \in S$ **shows** $AE\ s'\ in\ K\ s.\ s' \in E\ s$
⟨*proof*⟩

2.1.1 Almost every path is everywhere enabled

lemma *AE-all-E*:
assumes $*$: $\bigwedge t.\ t \in E\ s \Longrightarrow AE\ x\ in\ M.\ P\ t\ x$
shows $AE\ x\ in\ M.\ \forall t{\in}E\ s.\ P\ t\ x$
⟨*proof*⟩

2.2 The set of reachable states

inductive-set *reachable* :: $'s\ set \Rightarrow\ 's \Rightarrow\ 's\ set$ **for** Φ :: $'s\ set$ **and** s :: $'s$ **where**
start: $t \in E\ s \Longrightarrow t \in reachable\ \Phi\ s$
| *step*: $t \in reachable\ \Phi\ s \Longrightarrow t' \in E\ t \Longrightarrow t \in \Phi \Longrightarrow t' \in reachable\ \Phi\ s$

lemma *reachable-induct-trans*[*consumes 1*, *case-names start step*]:
assumes *t*: $t \in reachable\ \Phi\ s$
assumes *1*: $\bigwedge t\ s.\ t \in E\ s \Longrightarrow P\ t\ s$
assumes *2*: $\bigwedge t\ t'\ s.\ t' \in reachable\ \Phi\ s \Longrightarrow P\ t'\ s \Longrightarrow t' \in \Phi \Longrightarrow t \in reachable$
$\Phi\ t' \Longrightarrow$
$P\ t\ t' \Longrightarrow P\ t\ s$
shows $P\ t\ s$
⟨*proof*⟩

lemma *reachable-trans*:
assumes $t \in reachable\ \Phi\ s\ t' \in reachable\ \Phi\ t\ t \in \Phi$ **shows** $t' \in reachable\ \Phi\ s$
⟨*proof*⟩

lemma *reachable-step-rev*:
assumes $t \in reachable\ \Phi\ s\ s \in E\ s'\ s \in \Phi$ **shows** $t \in reachable\ \Phi\ s'$
⟨*proof*⟩

lemma *reachable-rev*:
assumes *t*: $t \in reachable\ \Phi\ s$
obtains (*start*) $t \in E\ s$ | (*step*) s' **where** $t \in reachable\ \Phi\ s'\ s' \in \Phi\ s' \in E\ s$
⟨*proof*⟩

lemma *reachable-induct-rev*[*consumes 1*, *case-names start step*]:
assumes *t*: $t \in reachable\ \Phi\ s$
assumes *1*: $\bigwedge s.\ t \in E\ s \Longrightarrow P\ s$
assumes *2*: $\bigwedge t'\ s.\ t' \in E\ s \Longrightarrow t' \in \Phi \Longrightarrow t \in reachable\ \Phi\ t' \Longrightarrow P\ t' \Longrightarrow P\ s$
shows $P\ s$
⟨*proof*⟩

lemma *reachable-in-S*[*dest*]:

assumes $t \in$ *reachable* Φ s **shows** $t \in S$
⟨*proof*⟩

lemma *reachable-closed*:
assumes $s \in R$ $s \in \Phi$ $\forall s{\in}R \cap \Phi.\ E\ s \subseteq R$
shows *reachable* $(\Phi - \Psi)$ $s \subseteq R$
⟨*proof*⟩

lemma *reachable-closed-rev*:
assumes t: $t \in$ *reachable* $(\Phi - \Psi)$ s
and R: $t \in R$ $\{s{\in}\Phi{-}\Psi.\ R \cap E\ s \neq \{\}\} \subseteq R$
and s: $s \in \Phi$ $s \notin \Psi$
shows $s \in R$
⟨*proof*⟩

lemma *reachableE*:
assumes t: $t \in$ *reachable* Φ s
obtains (*path*) ω n **where** $\bigwedge i.\ i \leq n \Longrightarrow \omega\ i \in E$ (*nat-case* s ω i) $\bigwedge i.\ i < n \Longrightarrow \omega\ i \in \Phi$ $\omega\ n = t$
⟨*proof*⟩

lemma *reachableI*:
$(\bigwedge i.\ i \leq n \Longrightarrow \omega\ i \in E$ (*nat-case* s ω i)$) \Longrightarrow (\bigwedge i.\ i < n \Longrightarrow \omega\ i \in \Phi) \Longrightarrow \omega\ n = t \Longrightarrow$
$t \in$ *reachable* Φ s
⟨*proof*⟩

lemma *reachableI2*:
$(\bigwedge i.\ i \leq n \Longrightarrow \omega\ i \in E$ (*nat-case* s ω i)$) \Longrightarrow (\bigwedge i.\ i < n \Longrightarrow \omega\ i \in \Phi) \Longrightarrow \omega\ n \in$ *reachable* Φ s
⟨*proof*⟩

2.3 The path generating function

primrec *path* :: $'s \Rightarrow (nat \Rightarrow {'s} \Rightarrow {'s}) \Rightarrow (nat \Rightarrow {'s})$ **where**
path s x $0 = x\ 0$ (*if* $s \in S$ *then* s *else* $s0$)
| *path* s x (*Suc* n) $= x$ (*Suc* n) (*path* s x n)

abbreviation
S-seq $\equiv (\Pi_M\ n{\in}UNIV{::}nat\ set.\ count\text{-}space\ S)$

lemma *path-in-S*: $x \in$ *space D-seq* $\Longrightarrow$ *path* s x $n \in S$
⟨*proof*⟩

lemma *measurable-path*:
path $s \in$ *measurable D-seq S-seq*
⟨*proof*⟩

lemma *measurable-path'*[*measurable* (*raw*)]:

assumes *f*: $f \in$ *measurable M* (*count-space S*) **and** *g*: $g \in$ *measurable M D-seq*
shows ($\lambda x.$ *path* (*f x*) (*g x*)) $\in$ *measurable M S-seq*
⟨*proof*⟩

lemma *path-nat-case*:
$s \in S \Longrightarrow s' \in S \to S \Longrightarrow$ *path s* (*nat-case s′ ω*) *i* = *nat-case* (*s′ s*) (*path* (*s′ s*) *ω*) *i*
⟨*proof*⟩

lemma *path-comb-seq*:
assumes $s \in S$ **and** *ω*: $\omega \in$ *space D-seq*
shows *path s* (*comb-seq i ω ω′*) = *comb-seq i* (*path s ω*) (*path* (*nat-case s* (*path s ω*) *i*) *ω′*)
⟨*proof*⟩

lemma *measurable-smallest*[*measurable*]: *Measurable.pred S-seq* ($\lambda\omega.$ *smallest ω X i*)
⟨*proof*⟩

2.4 *paths* is a probability space

definition
paths s = *distr D-seq S-seq* (*path s*)

lemma *sets-paths*[*simp*]: *sets* (*paths s*) = *sets S-seq* ⟨*proof*⟩
lemma *space-paths*[*simp*]: *space* (*paths s*) = *space S-seq* ⟨*proof*⟩
lemma *measurable-paths1*[*simp*]:
measurable (*paths s*) *M* = *measurable S-seq M* ⟨*proof*⟩
lemma *measurable-paths2*[*simp*]:
measurable M (*paths s*) = *measurable M S-seq* ⟨*proof*⟩

sublocale *path*: *prob-space paths s* **for** *s*
⟨*proof*⟩

lemma *borel-measurable-positive-integral-paths*[*measurable* (*raw*)]:
assumes *f*: $f \in$ *measurable M* (*count-space S*)
assumes *g*: ($\lambda(x, y).$ *g x y*) $\in$ *borel-measurable* ($M \bigotimes_M$ *S-seq*)
shows ($\lambda x.$ $integral^P$ (*paths* (*f x*)) (*g x*)) $\in$ *borel-measurable M*
⟨*proof*⟩

lemma *borel-measurable-lebesgue-integral-paths*[*measurable* (*raw*)]:
assumes *f*: $f \in$ *measurable M* (*count-space S*)
assumes *g*: ($\lambda(x, y).$ *g x y*) $\in$ *borel-measurable* ($M \bigotimes_M$ *S-seq*)
shows ($\lambda x.$ $integral^L$ (*paths* (*f x*)) (*g x*)) $\in$ *borel-measurable M*
⟨*proof*⟩

2.5 Splitting rules

lemma *positive-integral-split*:
assumes [*simp*]: $s \in S$ **and** *f*[*measurable*]: $f \in$ *borel-measurable S-seq*

shows ($\int^+\omega.\ f\ \omega\ \partial paths\ s$) = ($\int^+\omega.\ (\int^+\omega'.\ f\ (comb\text{-}seq\ i\ \omega\ \omega')\ \partial paths\ (nat\text{-}case\ s\ \omega\ i))\ \partial paths\ s$)
⟨*proof*⟩

lemma *integrable-split-AE*:
assumes [*simp*]: $s \in S$ **and** *f*: *integrable* (*paths s*) *f*
shows *AE* ω *in paths s*. *integrable* (*paths* (*nat-case s* ω *i*)) ($\lambda\omega'$. *f* (*comb-seq i* ω ω'))
⟨*proof*⟩

lemma *integrable-split*:
assumes [*simp*]: $s \in S$ **and** *f*: *integrable* (*paths s*) *f*
shows *integrable* (*paths s*) ($\lambda\omega$. $integral^L$ (*paths* (*nat-case s* ω *i*)) ($\lambda\omega'$. *f* (*comb-seq i* ω ω')))
⟨*proof*⟩

lemma *integral-split*:
assumes [*simp*]: $s \in S$ **and** *f*: *integrable* (*paths s*) *f*
shows ($\int\omega.\ f\ \omega\ \partial(paths\ s)$) = ($\int\omega.\ (\int\omega'.\ f\ (comb\text{-}seq\ i\ \omega\ \omega')\ \partial(paths\ (nat\text{-}case\ s\ \omega\ i)))\ \partial paths\ s$)
⟨*proof*⟩

lemma *emeasure-split*:
assumes [*simp*]: $s \in S$ **and** *A*[*measurable*]: $A \in$ *sets S-seq*
shows *emeasure* (*paths s*) *A* =
($\int^+\omega.\ emeasure\ (paths\ (nat\text{-}case\ s\ \omega\ i))\ (comb\text{-}seq\ i\ \omega\ -`\ A \cap space\ S\text{-}seq)\ \partial paths\ s$)
⟨*proof*⟩

lemma *emeasure-split-Collect*:
assumes $s \in S$ **and** *P*: $\{x \in space\ S\text{-}seq.\ P\ x\} \in sets\ S\text{-}seq$
shows *emeasure* (*paths s*) $\{x \in space\ (paths\ s).\ P\ x\}$ =
($\int^+\omega.\ emeasure\ (paths\ (nat\text{-}case\ s\ \omega\ i))\ \{\omega' \in space\ S\text{-}seq.\ P\ (comb\text{-}seq\ i\ \omega\ \omega')\}\ \partial paths\ s$)
⟨*proof*⟩

lemma *prob-split*:
assumes $s \in S$ **and** *A*: $A \in$ *sets S-seq*
shows *prob s A* =
($\int\omega.\ prob\ (nat\text{-}case\ s\ \omega\ i)\ (comb\text{-}seq\ i\ \omega\ -`\ A \cap space\ S\text{-}seq)\ \partial paths\ s$)
⟨*proof*⟩

lemma *prob-split-Collect*:
assumes $s \in S$ **and** *P*: $\{x \in space\ S\text{-}seq.\ P\ x\} \in sets\ S\text{-}seq$
shows $\mathcal{P}(x\ in\ paths\ s.\ P\ x)$ = ($\int\omega.\ \mathcal{P}(\omega'\ in\ paths\ (nat\text{-}case\ s\ \omega\ i).\ P\ (comb\text{-}seq\ i\ \omega\ \omega'))\ \partial paths\ s$)
⟨*proof*⟩

lemma *AE-split*:

assumes [*simp*]: $s \in S$ **and** P[*measurable*]: $\{x{\in}space\ S\text{-}seq.\ P\ x\} \in sets\ S\text{-}seq$
shows $(AE\ \omega\ in\ paths\ s.\ P\ \omega) \longleftrightarrow$
$(AE\ \omega\ in\ paths\ s.\ AE\ \omega'\ in\ paths\ (nat\text{-}case\ s\ \omega\ i).\ P\ (comb\text{-}seq\ i\ \omega\ \omega'))$
⟨*proof*⟩

2.6 The Generic Chapman-Kolmogorov theorem

lemma *Chapman-Kolmogorov*:
assumes [*simp*]: $s \in S$
assumes [*measurable*]: *Measurable.pred* (*paths s*) *Q1 Measurable.pred* (*paths s*) *P*
assumes *eq*: $\bigwedge\omega\ \omega'.\ \omega \in UNIV \to S \Longrightarrow \omega' \in UNIV \to S \Longrightarrow P\ (comb\text{-}seq\ n\ \omega\ \omega') \longleftrightarrow Q1\ \omega \wedge Q2\ (nat\text{-}case\ (nat\text{-}case\ s\ \omega\ n)\ \omega')$
shows $\mathcal{P}(\omega\ in\ paths\ s.\ P\ \omega) = (\int t.\ \mathcal{P}(\omega\ in\ paths\ s.\ Q1\ \omega \wedge nat\text{-}case\ s\ \omega\ n = t) * \mathcal{P}(\omega\ in\ paths\ t.\ Q2\ (nat\text{-}case\ t\ \omega))\ \partial count\text{-}space\ S)$
⟨*proof*⟩

2.7 Iteration rules and splitting at 1

lemma *distr-K*:
assumes [*simp*]: $s \in S$ **shows** $distr\ (paths\ s)\ (K\ s)\ (\lambda\omega.\ \omega\ 0) = K\ s$
⟨*proof*⟩

lemma *positive-integral-paths-0*:
assumes s[*simp*]: $s \in S$ **shows** $(\int^+\omega.\ f\ (\omega\ 0)\ \partial paths\ s) = (\int^+ s'.\ f\ s'\ \partial K\ s)$
⟨*proof*⟩

lemma *emeasure-paths-0*:
assumes s[*simp*]: $s \in S$
shows $emeasure\ (paths\ s)\ \{\omega{\in}space\ S\text{-}seq.\ P\ (\omega\ 0)\} = emeasure\ (K\ s)\ \{s'{\in}space\ (count\text{-}space\ S).\ P\ s'\}$
⟨*proof*⟩

lemma *measure-paths-0*:
assumes s[*simp*]: $s \in S$
shows $measure\ (paths\ s)\ \{\omega{\in}space\ S\text{-}seq.\ P\ (\omega\ 0)\} = measure\ (K\ s)\ \{s'{\in}space\ (count\text{-}space\ S).\ P\ s'\}$
⟨*proof*⟩

lemma *prob-paths-0*:
assumes s[*simp*]: $s \in S$
shows $\mathcal{P}(\omega\ in\ paths\ s.\ P\ (\omega\ 0)) = \mathcal{P}(t\ in\ K\ s.\ P\ t)$
⟨*proof*⟩

lemma *integrable-paths-0*:
assumes s[*simp*]: $s \in S$
shows $integrable\ (paths\ s)\ (\lambda\omega.\ f\ (\omega\ 0)) = integrable\ (K\ s)\ f$
⟨*proof*⟩

lemma *integral-paths-0*:

assumes $s[simp]$: $s \in S$
shows $integral^L$ $(paths\ s)$ $(\lambda\omega.\ f\ (\omega\ 0)) = integral^L\ (K\ s)\ f$
$\langle proof \rangle$

lemma *AE-paths-0*:
assumes $s[simp]$: $s \in S$
shows $(AE\ \omega\ in\ paths\ s.\ P\ (\omega\ 0)) = (AE\ s'\ in\ K\ s.\ P\ s')$
$\langle proof \rangle$

lemma *positive-integral-iterate*:
assumes $[simp]$: $s \in S$ **and** f: $f \in borel\text{-}measurable\ S\text{-}seq$
shows $(\int^+\omega.\ f\ \omega\ \partial(paths\ s)) = (\int^+ s'.\ (\int^+\omega.\ f\ (nat\text{-}case\ s'\ \omega)\ \partial(paths\ s'))\ \partial K$
$s)$
$\langle proof \rangle$

lemma *integrable-iterate-AE*:
assumes $s[simp]$: $s \in S$ **and** f: $integrable\ (paths\ s)\ f$
shows $AE\ s'\ in\ K\ s.\ integrable\ (paths\ s')\ (\lambda\omega.\ f\ (nat\text{-}case\ s'\ \omega))$
$\langle proof \rangle$

lemma *integrable-iterate*:
assumes $[simp]$: $s \in S$ **and** f: $integrable\ (paths\ s)\ f$
shows $integrable\ (K\ s)\ (\lambda s'.\ integral^L\ (paths\ s')\ (\lambda\omega.\ f\ (nat\text{-}case\ s'\ \omega)))$
$\langle proof \rangle$

lemma *integral-iterate*:
assumes $[simp]$: $s \in S$ **and** f: $integrable\ (paths\ s)\ f$
shows $(\int\omega.\ f\ \omega\ \partial(paths\ s)) = (\int s'.\ (\int\omega.\ f\ (nat\text{-}case\ s'\ \omega)\ \partial(paths\ s'))\ \partial K\ s)$
$\langle proof \rangle$

lemma *emeasure-iterate*:
assumes $[simp]$: $s \in S$ **and** $A[measurable]$: $A \in sets\ S\text{-}seq$
shows $emeasure\ (paths\ s)\ A = (\int^+ s'.\ emeasure\ (paths\ s')\ (nat\text{-}case\ s'\ -`\ A \cap$
$space\ S\text{-}seq)\ \partial K\ s)$
$\langle proof \rangle$

lemma *prob-iterate*:
assumes $s \in S$ **and** A: $A \in sets\ S\text{-}seq$
shows $prob\ s\ A = (\int s'.\ prob\ s'\ (nat\text{-}case\ s'\ -`\ A \cap space\ S\text{-}seq)\ \partial K\ s)$
$\langle proof \rangle$

lemma *prob-iterate-Collect*:
assumes $s \in S$ **and** P: $\{x \in space\ S\text{-}seq.\ P\ x\} \in sets\ S\text{-}seq$
shows $\mathcal{P}(x\ in\ paths\ s.\ P\ x) = (\int s'.\ \mathcal{P}(x\ in\ paths\ s'.\ P\ (nat\text{-}case\ s'\ x))\ \partial K\ s)$
$\langle proof \rangle$

lemma *prob-path*:
$s \in S \Longrightarrow (\bigwedge i.\ i < n \Longrightarrow \omega'\ i \in S) \Longrightarrow$
$\mathcal{P}(\omega\ in\ paths\ s.\ \forall i{<}n.\ \omega\ i = \omega'\ i) = (\prod i{<}n.\ measure\ (K\ (nat\text{-}case\ s\ \omega'\ i))\ \{\omega'$

$i\})$
$\langle proof \rangle$

lemma *emeasure-path*:
$s \in S \Longrightarrow (\bigwedge i.\ i < n \Longrightarrow \omega'\ i \in S) \Longrightarrow$
$emeasure\ (paths\ s)\ \{\omega{\in}space\ (paths\ s).\ \forall i{<}n.\ \omega\ i = \omega'\ i\} = (\prod i{<}n.\ emeasure\ (K\ (nat\text{-}case\ s\ \omega'\ i))\ \{\omega'\ i\})$
$\langle proof \rangle$

lemma *AE-iterate*:
assumes $[simp]$: $s \in S$ **and** $P[measurable]$: $\{x{\in}space\ S\text{-}seq.\ P\ x\} \in sets\ S\text{-}seq$
shows $(AE\ x\ in\ paths\ s.\ P\ x) \longleftrightarrow (AE\ s'\ in\ K\ s.\ AE\ x\ in\ paths\ s'.\ P\ (nat\text{-}case\ s'\ x))$
$\langle proof \rangle$

lemma *AE-all-enabled*:
assumes $s[simp]$: $s \in S$ **shows** $AE\ \omega\ in\ paths\ s.\ \forall i.\ \omega\ i \in E\ (nat\text{-}case\ s\ \omega\ i)$
$\langle proof \rangle$

2.8 Fairness

The fairness proof is similar to Theorem 8.1.5 in Baier 1998 (Habilitation thesis). The differences are

- we only prove it for s-fairness (only one transition)
- our prove works for systems with arbitrary size, i.e. also countable infinite systems

definition $fair\ s\ t\ \omega \longleftrightarrow finite\ \{i.\ \omega\ i = s \wedge \omega\ (Suc\ i) = t\} \longrightarrow finite\ \{i.\ \omega\ i = s\}$

lemma *fairI*:
$(finite\ \{i.\ nat\text{-}case\ s'\ \omega\ i = s \wedge \omega\ i = t\} \Longrightarrow finite\ \{i.\ nat\text{-}case\ s'\ \omega\ i = s\}) \Longrightarrow$
$fair\ s\ t\ (nat\text{-}case\ s'\ \omega)$
$\langle proof \rangle$

lemma *measurable-fair*$[measurable]$: $\{\omega{\in}space\ S\text{-}seq.\ fair\ s\ t\ \omega\} \in sets\ S\text{-}seq$
$\langle proof \rangle$

lemma *positive-integral-prefixes*:
assumes $s[simp]$: $s \in S$
assumes $[measurable]$: $i \in measurable\ S\text{-}seq\ (count\text{-}space\ UNIV)$
and $[measurable]$: $f \in borel\text{-}measurable\ S\text{-}seq$
and f: $AE\ x\ in\ paths\ s.\ 0 \le f\ x$
and $inv\text{-}i$: $\bigwedge \omega\ \omega'.\ 0 < i\ \omega \Longrightarrow (\bigwedge j.\ j < i\ \omega \Longrightarrow \omega\ j = \omega'\ j) \Longrightarrow i\ \omega = i\ \omega'$
and $inv\text{-}f$: $\bigwedge \omega.\ i\ \omega = 0 \Longrightarrow f\ \omega = 0$
shows $(\int^+ \omega.\ f\ \omega\ \partial paths\ s) = (\int^+ \omega.\ indicator\ \{\omega.\ i\ \omega \neq 0\}\ \omega *$
$(\int^+ \omega'.\ f\ (comb\text{-}seq\ (i\ \omega)\ \omega\ \omega')\ \partial paths\ (nat\text{-}case\ s\ \omega\ (i\ \omega)))\ \partial paths\ s)$

$\langle proof \rangle$

lemma *AE-fair*:
  assumes $s[simp]$: $s' \in S$ **and** $s[simp]$: $s \in S$ **and** $t[simp]$: $t \in E\ s$
  shows $AE\ \omega\ in\ paths\ s'.\ fair\ s\ t\ (nat\text{-}case\ s'\ \omega)$
$\langle proof \rangle$

lemma *AE-all-fair*:
  $s' \in S \Longrightarrow AE\ \omega\ in\ paths\ s'.\ \forall s{\in}S.\ \forall t{\in}E\ s.\ fair\ s\ t\ (nat\text{-}case\ s'\ \omega)$
  $\langle proof \rangle$

lemma *fair-eq*: $fair\ sx\ tx\ \omega \longleftrightarrow$
  $(\exists j.\ \forall i{\geq}j.\ \omega\ i \neq sx) \lor (\forall i.\ \exists j{\geq}i.\ \omega\ j = sx \land \omega\ (Suc\ j) = tx)$
  $\langle proof \rangle$

2.9 *until*

definition (**in** $-$) *until* :: $'s\ set \Rightarrow\ 's\ set \Rightarrow (nat \Rightarrow\ 's)\ set$ **where**
  $until\ \Phi\ \Psi = \{\omega.\ \exists n.\ (\forall i{<}n.\ \omega\ i \in \Phi) \land \omega\ n \in \Psi\}$

lemma (**in** $-$) *measurable-until*[*measurable*]: $\{\omega{\in}space\ (\Pi_M\ n{\in}UNIV{::}nat\ set.\ count\text{-}space\ S).\ \omega \in until\ \Phi\ \Psi\} \in sets\ (\Pi_M\ n{\in}UNIV{::}nat\ set.\ count\text{-}space\ S)$
  $\langle proof \rangle$

lemma *untilI*:
  $(\bigwedge i.\ i < n \Longrightarrow \omega\ i \in \Phi) \Longrightarrow \omega\ n \in \Psi \Longrightarrow \omega \in until\ \Phi\ \Psi$
  $\langle proof \rangle$

lemma *untilE*:
  assumes ω: $\omega \in until\ \Phi\ \Psi$
  obtains (*until*) n **where** $\bigwedge i.\ i < n \Longrightarrow \omega\ i \in \Phi - \Psi\ \ \omega\ n \in \Psi$
$\langle proof \rangle$

lemma *until-iff*:
  $\omega \in until\ \Phi\ \Psi \longleftrightarrow (\exists n.\ (\forall i < n.\ \omega\ i \in \Phi - \Psi) \land \omega\ n \in \Psi)$
  $\langle proof \rangle$

lemma *nat-case-until-iff*[*simp*]:
  $nat\text{-}case\ s\ \omega \in until\ \Phi\ \Psi \longleftrightarrow (s \in \Psi \lor (s \in \Phi \land \omega \in until\ \Phi\ \Psi))$
  $\langle proof \rangle$

lemma *comb-seq-until*:
  assumes ω: $\bigwedge j.\ j < i \Longrightarrow \omega\ j \in \Phi - \Psi$
  shows $comb\text{-}seq\ i\ \omega\ \omega' \in until\ \Phi\ \Psi \longleftrightarrow \omega' \in until\ \Phi\ \Psi$
$\langle proof \rangle$

lemma *single-K-measure-le-integral*:
  assumes [*simp*]: $s \in S$ **and** $t \in E\ s$ **and** *nneg*: $AE\ t\ in\ K\ s.\ 0 \leq f\ t$
    and *int*[*measurable*]: *integrable* $(K\ s)\ f$

shows $measure\ (K\ s)\ \{t\} * f\ t \le (\int t.\ f\ t\ \partial K\ s)$
$\langle proof \rangle$

lemma *AE-paths-iff*:
$s \in S \Longrightarrow \{\omega{\in}space\ S\text{-}seq.\ P\ \omega\} \in sets\ S\text{-}seq \Longrightarrow$
$(AE\ \omega\ in\ paths\ s.\ P\ \omega) \longleftrightarrow$
$(\forall\omega.\ (\forall j{<}i.\ \omega\ j \in E\ (nat\text{-}case\ s\ \omega\ j)) \longrightarrow (AE\ \omega'\ in\ paths\ (nat\text{-}case\ s\ \omega\ i).\ P\ (comb\text{-}seq\ i\ \omega\ \omega')))$
(**is** $- \Longrightarrow - \Longrightarrow - \longleftrightarrow (\forall\omega.\ ?S\ i\ s\ P\ \omega)$)
$\langle proof \rangle$

lemma *AE-nuntil-iff-not-reachable*:
assumes $s[simp]$: $s \in S$
shows $(AE\ \omega\ in\ paths\ s.\ nat\text{-}case\ s\ \omega \notin until\ \Phi\ \Psi) \longleftrightarrow$
$s \notin \Psi \wedge (s \in \Phi \longrightarrow reachable\ (\Phi - \Psi)\ s \cap \Psi = \{\})$
$\langle proof \rangle$

lemma *AE-until*:
assumes s: $s \in S$ $s \in \Phi$ **and** Φ: $finite\ (\Phi - \Psi)$ **and** *closed*: $reachable\ (\Phi - \Psi)\ s \subseteq \Phi \cup \Psi$
assumes *enabled*: $\forall t{\in}reachable\ (\Phi - \Psi)\ s \cup \{s\} - \Psi.\ reachable\ (\Phi - \Psi)\ t \cap \Psi \neq \{\}$
shows $AE\ \omega\ in\ paths\ s.\ nat\text{-}case\ s\ \omega \in until\ \Phi\ \Psi$
$\langle proof \rangle$

lemma *AE-until-iff-reachable*:
assumes $[simp]$: $s \in S$ $finite\ (\Phi - \Psi)$
shows $(AE\ \omega\ in\ paths\ s.\ nat\text{-}case\ s\ \omega \in until\ \Phi\ \Psi) \longleftrightarrow$
$(s \in \Phi \wedge reachable\ (\Phi - \Psi)\ s \subseteq \Phi \cup \Psi \wedge$
$(\forall t{\in}reachable\ (\Phi - \Psi)\ s \cup \{s\} - \Psi.\ reachable\ (\Phi - \Psi)\ t \cap \Psi \neq \{\})) \vee s \in \Psi$
$\langle proof \rangle$

2.10 Hitting time (as natural number)

definition $hitting\text{-}time :: 's\ set \Rightarrow (nat \Rightarrow 's) \Rightarrow nat$ **where**
$hitting\text{-}time\ \Phi\ \omega = (LEAST\ i.\ \omega\ i \in \Phi)$

lemma *measurable-hitting-time*[*measurable*]:
$hitting\text{-}time\ \Phi \in measurable\ S\text{-}seq\ (count\text{-}space\ UNIV)$
$\langle proof \rangle$

lemma *hitting-time-eq*:
$\omega\ n \in \Phi \Longrightarrow (\bigwedge i.\ i < n \Longrightarrow \omega\ i \notin \Phi) \Longrightarrow hitting\text{-}time\ \Phi\ \omega = n$
$\langle proof \rangle$

lemma *hitting-time-least*: $i < hitting\text{-}time\ \Phi\ \omega \Longrightarrow \omega\ i \notin \Phi$
$\langle proof \rangle$

lemma
assumes *until*: $\omega \in$ *until* $S\ \Phi$
shows *hitting-time-in*[*intro*]: ω (*hitting-time* $\Phi\ \omega) \in \Phi$
⟨*proof*⟩

lemma *hitting-time-nat-case-Suc*:
assumes $\omega \in$ *until* $S\ \Phi\ s \notin \Phi$
shows *hitting-time* Φ (*nat-case* $s\ \omega$) = *Suc* (*hitting-time* $\Phi\ \omega$)
⟨*proof*⟩

lemma *hitting-time-nat-case-0*:
$s \in \Phi \Longrightarrow$ *hitting-time* Φ (*nat-case* $s\ \omega$) = *0*
⟨*proof*⟩

lemma *positive-integral-hitting-time-finite*:
assumes [*simp*]: $s \in S$ **and** Φ: *finite* $(S - \Phi)$
assumes *until*: *AE* ω *in paths* s. *nat-case* $s\ \omega \in$ *until* $S\ \Phi$
shows ($\int^{+}$ ω. *real* (*hitting-time* Φ (*nat-case* $s\ \omega$)) ∂*paths* s) $\neq \infty$
⟨*proof*⟩

end

lemma *measurable-component-singleton-const*[*measurable-app*]:
assumes f: $f \in$ *measurable* N (Pi_M I (λi. M))
assumes i: $i \in I$
shows (λx. ($f\ x$) i) $\in$ *measurable* $N\ M$
⟨*proof*⟩

lemma *measurable-abs-UNIV*[*measurable*]:
($\bigwedge n$. ($\lambda\omega$. $f\ n\ \omega$) $\in$ *measurable* M ($N\ n$)) $\Longrightarrow$ ($\lambda\omega\ n$. $f\ n\ \omega$) $\in$ *measurable* M (*PiM UNIV N*)
⟨*proof*⟩

lemma *sets-UNIV* [*measurable* (*raw*)]: $A \in$ *sets* (*count-space UNIV*)
⟨*proof*⟩

end

theory *Constructing-Markov-Chain*
imports *Discrete-Markov-Kernel*
begin

3 Markov Chain as Stochastic Process

We can construct for each time-homogeneous discrete-time Markov process a corresponding probability space using *Discrete-Markov-Kernel*. The constructed probability space has the same probabilities.

locale *Discrete-Time-Markov-Chain* = *M*: *prob-space* +
  fixes *S* :: *'s set* **and** *X* :: *nat* ⇒ *'a* ⇒ *'s*
  assumes *S*: *countable S*
  assumes *X*[*measurable*]: ⋀*t*. *X t* ∈ *measurable M* (*count-space S*)
  assumes *MC*: ⋀*n s s'*.
    𝒫(ω *in M*. ∀ *t*≤*n*. *X t* ω = *s t*) ≠ *0* ⟹
    𝒫(ω *in M*. *X* (*Suc n*) ω = *s'* | ∀ *t*≤*n*. *X t* ω = *s t*) =
    𝒫(ω *in M*. *X* (*Suc n*) ω = *s'* | *X n* ω = *s n*)
  assumes *TH*: ⋀*n m s t*.
    𝒫(ω *in M*. *X n* ω = *t*) ≠ *0* ⟹ 𝒫(ω *in M*. *X m* ω = *t*) ≠ *0* ⟹
    𝒫(ω *in M*. *X* (*Suc n*) ω = *s* | *X n* ω = *t*) = 𝒫(ω *in M*. *X* (*Suc m*) ω = *s* | *X m* ω = *t*)
begin

definition *S'* = *Some* ‘ *S* ∪ {*None*}

lemma *None-in-S'*[*simp*]: *None* ∈ *S'*
  ⟨*proof*⟩

lemma *S-not-empty*: *S* ≠ {}
  ⟨*proof*⟩

lemma *S'-not-empty*: *S'* ≠ {}
  ⟨*proof*⟩

lemma *measurable-Some*[*measurable*]: *Some* ∈ *measurable* (*count-space S*) (*count-space S'*)
  ⟨*proof*⟩

lemma *countable-S'*: *countable S'*
  ⟨*proof*⟩

definition *with P f d* = (*if* ∃ *x*. *P x then f* (*SOME x*. *P x*) *else d*)

lemma *withI*[*case-names default exists*]:
  ((⋀*x*. ¬ *P x*) ⟹ *Q d*) ⟹ (⋀*x*. *P x* ⟹ *Q* (*f x*)) ⟹ *Q* (*with P f d*)
  ⟨*proof*⟩

3.1 Construct *K'*

primrec *K'* :: *'s option* ⇒ *'s option measure* **where**
  K' None = *distr M* (*count-space S'*) (λω. *Some* (*X 0* ω))
| *K'* (*Some s*) =
    with (λ*n*. *0* < 𝒫(ω *in M*. *X n* ω = *s*))

($\lambda n.$ *distr* (*uniform-measure* M $\{\omega{\in}$*space* $M.$ X n ω $=$ $s\}$) (*count-space* S') ($\lambda\omega.$ *Some* (X (*Suc* n) ω)))
(*point-measure* S' ($\lambda x.$ *if* x $=$ *Some* (*SOME* $s.$ $s \in S$) *then* 1 *else* 0))

lemma *sets-K*[*simp*]: *sets* (K' s) $=$ *Pow* S'
⟨*proof*⟩

lemma *space-K*[*simp*]: *space* (K' s) $=$ S'
⟨*proof*⟩

3.2 Equations for K'

lemma *emeasure-K-S*[*simp*]:
assumes $0 < \mathcal{P}(\omega$ *in* $M.$ X n $\omega = s)$
shows *emeasure* (K' (*Some* s)) $\{$*Some* $t\}$ $= \mathcal{P}(\omega$ *in* $M.$ X (*Suc* n) $\omega = t \mid X$ n $\omega = s)$
⟨*proof*⟩

lemma *emeasure-K-None*:
emeasure (K' (*Some* s)) $\{$*None*$\}$ $=$ 0
⟨*proof*⟩

lemma *emeasure-K-init*[*simp*]: *emeasure* (K' *None*) $\{t\}$ $= \mathcal{P}(\omega$ *in* $M.$ *Some* (X 0 ω) $= t)$
⟨*proof*⟩

sublocale *K*!: *prob-space* K' s **for** s
⟨*proof*⟩

sublocale *MC*: *Discrete-Markov-Kernel* S' K'
⟨*proof*⟩

declare K'*.simps*[*simp del*]

3.3 Markov Kernel's Path Space equals Stochastic Process

lemma *paths-None-eq*:
paths None $=$ *distr* M *S-seq* ($\lambda\omega$ $n.$ *Some* (X n ω)) (**is** *?L* $=$ *?R*)
⟨*proof*⟩

end

4 Discrete Markov Kernel Constructs a DTMC

context *Discrete-Markov-Kernel*
begin

lemma *split-at*:
assumes [*simp*]: $s \in S$ **and** 2: $\mathcal{P}(\omega'$ *in paths* $s.$ ω' $n = \omega$ $n) \neq 0$

shows $\mathcal{P}(\omega'$ *in paths s.* ω' $(Suc\ n) = s'$ | $\omega'\ n = \omega\ n$ $) = \mathcal{P}(\omega'$ *in paths* $(\omega\ n).$ $\omega'\ 0 = s')$
⟨*proof*⟩

lemma *is-DTMC*:
assumes [*simp*]: $s \in S$
shows *Discrete-Time-Markov-Chain* (*paths s*) *S* $(\lambda n\ \omega.\ \omega\ n)$
⟨*proof*⟩

end

end

theory *Classifying-Markov-Chains*
imports *Discrete-Markov-Kernel Constructing-Markov-Chain*
begin

lemma *int-cases'*: $(\bigwedge n.\ x = int\ n \Longrightarrow P) \Longrightarrow (\bigwedge n.\ x = -\ int\ n \Longrightarrow P) \Longrightarrow P$
⟨*proof*⟩

lemma *nat-abs-int-diff*: $nat\ |int\ a - int\ b| = (if\ a \le b\ then\ b - a\ else\ a - b)$
⟨*proof*⟩

lemma *nat-int-add*: $nat\ (int\ a + int\ b) = a + b$
⟨*proof*⟩

context *Discrete-Markov-Kernel*
begin

5 Analyzing Markov Chains

5.1 Accessible

inductive *accessible* **for** *s* **where**
start: $s \in S \Longrightarrow accessible\ s\ s$
| *step*: $accessible\ s\ t \Longrightarrow t' \in E\ t \Longrightarrow accessible\ s\ t'$

lemma *accessible-in-S1*: **assumes** *accessible s t* **shows** $s \in S$
⟨*proof*⟩

lemma *accessible-in-S2*: **assumes** *accessible s t* **shows** $t \in S$
⟨*proof*⟩

lemma *accessible-trans*:
assumes *accessible s t accessible t t'* **shows** *accessible s t'*
⟨*proof*⟩

lemma *accessible-single*:

$s \in S \Longrightarrow t \in E\ s \Longrightarrow accessible\ s\ t$
⟨*proof*⟩

lemma *accessible-induct-trans*[*consumes 1*, *case-names start step trans*]:
assumes *t*: *accessible s t*
assumes *1*: $\bigwedge s.\ s \in S \Longrightarrow P\ s\ s$
assumes *2*: $\bigwedge s\ t.\ s \in S \Longrightarrow t \in E\ s \Longrightarrow P\ s\ t$
assumes *3*: $\bigwedge t\ t'\ s.\ accessible\ s\ t' \Longrightarrow P\ s\ t' \Longrightarrow accessible\ t'\ t \Longrightarrow P\ t'\ t \Longrightarrow P\ s\ t$
shows $P\ s\ t$
⟨*proof*⟩

lemma *accessible-step-rev*:
assumes $accessible\ s\ t\ s \in E\ s'\ s' \in S$ **shows** $accessible\ s'\ t$
⟨*proof*⟩

lemma *accessible-rev*:
assumes *t*: *accessible s t*
obtains (*start*) $t = s\ \ s \in S$ | (*step*) s' **where** $accessible\ s'\ t\ s' \in E\ s\ s \in S$
⟨*proof*⟩

lemma *accessible-induct-rev*[*consumes 1*, *case-names start step*]:
assumes *t*: *accessible s t*
assumes *1*: $P\ t$
assumes *2*: $\bigwedge t'\ s.\ t' \in E\ s \Longrightarrow accessible\ t'\ t \Longrightarrow P\ t' \Longrightarrow P\ s$
shows $P\ s$
⟨*proof*⟩

lemma *AE-accessible*:
assumes *x*: $x \in S$
shows $AE\ \omega\ in\ paths\ x.\ \forall m.\ accessible\ x\ (nat\text{-}case\ x\ \omega\ m)$
⟨*proof*⟩

5.2 Communicating

definition *communicating* :: $('s \times 's)\ set$ **where**
$communicating = \{(s,\ t).\ accessible\ s\ t \wedge accessible\ t\ s\}$

lemma *irreducibleD*:
$C \in S\ //\ communicating \Longrightarrow a \in C \Longrightarrow b \in C \Longrightarrow (a,\ b) \in communicating$
⟨*proof*⟩

lemma *irreducibleD2*:
$C \in S\ //\ communicating \Longrightarrow a \in C \Longrightarrow (a,\ b) \in communicating \Longrightarrow b \in C$
⟨*proof*⟩

lemma *communicating-in-S*:
assumes $(x,\ y) \in communicating$ **shows** $x \in S\ y \in S$
⟨*proof*⟩

lemma *C-subset-S*: $C \in S$ // *communicating* $\Longrightarrow C \subseteq S$
⟨*proof*⟩

lemma *equiv-communicating*: *equiv S communicating*
⟨*proof*⟩

lemma *communicatingD1*:
$C \in S$ // *communicating* $\Longrightarrow (a,\ b) \in$ *communicating* $\Longrightarrow a \in C \Longrightarrow b \in C$
⟨*proof*⟩

lemma *communicatingD2*:
$C \in S$ // *communicating* $\Longrightarrow (a,\ b) \in$ *communicating* $\Longrightarrow b \in C \Longrightarrow a \in C$
⟨*proof*⟩

lemma *irreducible-MC*: $(\bigwedge x\ y.\ x \in S \Longrightarrow y \in S \Longrightarrow$ *accessible* $x\ y) \Longrightarrow S \in S$ // *communicating*
⟨*proof*⟩

lemma *not-empty-irreducible*:
$C \in S$ // *communicating* $\Longrightarrow C \neq \{\}$
⟨*proof*⟩

5.3 Time-Bounded Quantities

definition $p\ x\ y\ n = \mathcal{P}(\omega$ *in paths* $x.$ *nat-case* $x\ \omega\ n = y)$

lemma *p-nonneg*: $0 \leq p\ x\ y\ n$
⟨*proof*⟩

lemma *p-le-1*: $p\ x\ y\ n \leq 1$
⟨*proof*⟩

lemma *p-x-x-0*[*simp*]: $p\ x\ x\ 0 = 1$
⟨*proof*⟩

lemma *p-0*: $p\ x\ y\ 0 = ($*if* $x = y$ *then* 1 *else* $0)$
⟨*proof*⟩

lemma *p-Suc*:
assumes $x \in S$
shows $p\ x\ y\ (Suc\ n) = (\int x'.\ p\ x'\ y\ n\ \partial K\ x)$
⟨*proof*⟩

lemma *accessibleI-pos*:
$x \in S \Longrightarrow y \in S \Longrightarrow 0 < p\ x\ y\ n \Longrightarrow$ *accessible* $x\ y$
⟨*proof*⟩

lemma *prob-reachable-le*:

assumes [*simp*]: $x \in S$ $y \in S$ $w \in S$ **and** $m \leq n$
shows $p\ x\ y\ m * p\ y\ w\ (n - m) \leq p\ x\ w\ n$
⟨*proof*⟩

lemma *accessibleD-pos*:
assumes *accessible* $x\ y$
shows $\exists\, n.\ 0 < p\ x\ y\ n$
⟨*proof*⟩

5.3.1 Accessibility as probability on traces

lemma *accessible-iff*: *accessible* $x\ y \longleftrightarrow x \in S \wedge y \in S \wedge (\exists\, n.\ 0 < p\ x\ y\ n)$
⟨*proof*⟩

lemma *communicating-iff*: $(x,\ y) \in$ *communicating* $\longleftrightarrow x \in S \wedge y \in S \wedge (\exists\, n.\ 0 < p\ x\ y\ n) \wedge (\exists\, n.\ 0 < p\ y\ x\ n)$
⟨*proof*⟩

definition $u\ x\ y\ n = \mathcal{P}(\omega\ in\ paths\ x.\ (\forall\, i{<}n.\ \omega\ i \neq y) \wedge \omega\ n = y)$

lemma shows *u-nonneg*: $0 \leq u\ x\ y\ n$ **and** *u-le-1*: $u\ x\ y\ n \leq 1$
⟨*proof*⟩

lemma *u-le-p*: $u\ x\ y\ n \leq p\ x\ y\ (Suc\ n)$
⟨*proof*⟩

lemma *p-eq-setsum-p-u*:
assumes x[*simp*]: $x \in S$ **shows** $p\ x\ y\ (Suc\ n) = (\sum i{\leq}n.\ p\ y\ y\ (n - i) * u\ x\ y\ i)$
⟨*proof*⟩

definition $f\ x\ y\ n = \mathcal{P}(\omega\ in\ paths\ x.\ (\forall\, i{<}n.\ nat\text{-}case\ x\ \omega\ i \neq y) \wedge nat\text{-}case\ x\ \omega\ n = y)$

lemma shows *f-nonneg*: $0 \leq f\ x\ y\ n$ **and** *f-le-1*: $f\ x\ y\ n \leq 1$
⟨*proof*⟩

lemma *f-le-p*: $f\ x\ y\ n \leq p\ x\ y\ n$
⟨*proof*⟩

lemma *f-Suc*: $x \neq y \Longrightarrow f\ x\ y\ (Suc\ n) = u\ x\ y\ n$
⟨*proof*⟩

lemma *f-Suc-eq*: $f\ x\ x\ (Suc\ n) = 0$
⟨*proof*⟩

lemma *f-0*: $f\ x\ y\ 0 = (if\ x = y\ then\ 1\ else\ 0)$
⟨*proof*⟩

lemma *atMost-Suc-insert-0*: $\{.. \; Suc \; n\} = insert \; 0 \; (Suc \; ` \; \{.. \; n\})$
⟨*proof*⟩

lemma *p-eq-setsum-p-f*:
assumes $x[simp]$: $x \in S$ **shows** $p \; x \; y \; n = (\sum i{\le}n. \; p \; y \; y \; (n - i) * f \; x \; y \; i)$
⟨*proof*⟩

5.4 Quantities G, U, and F

definition $convergence\text{-}G \; x \; y \; z \longleftrightarrow summable \; (\lambda n. \; p \; x \; y \; n * norm \; z \; \hat{} \; n)$

definition $G \; s \; t = (\int^{+}\omega. \; (\sum i. \; indicator \; \{t\} \; (nat\text{-}case \; s \; \omega \; i)) \; \partial paths \; s)$

definition $U \; s \; t = \mathcal{P}(\omega \; in \; paths \; s. \; \exists \, n. \; \omega \; n = t)$

definition $F \; s \; t = \mathcal{P}(\omega \; in \; paths \; s. \; \exists \, n. \; nat\text{-}case \; s \; \omega \; n = t)$

lemma *G-eq*: $G \; s \; t = (\int^{+}\omega. \; emeasure \; (count\text{-}space \; UNIV) \; \{i. \; nat\text{-}case \; s \; \omega \; i = t\} \; \partial paths \; s)$
⟨*proof*⟩

5.4.1 G equals infinite sum of p

lemma *G-eq-suminf*: $G \; x \; y = (\sum i. \; ereal \; (p \; x \; y \; i))$
⟨*proof*⟩

lemma *G-eq-real-suminf*:
$convergence\text{-}G \; x \; y \; (1{::}real) \Longrightarrow G \; x \; y = ereal \; (\sum i. \; p \; x \; y \; i)$
⟨*proof*⟩

lemma *convergence-norm-G*:
$convergence\text{-}G \; x \; y \; z \Longrightarrow summable \; (\lambda n. \; p \; x \; y \; n * norm \; z \; \hat{} \; n)$
⟨*proof*⟩

lemma *convergence-G*:
$convergence\text{-}G \; x \; y \; (z{::}'a{::}\{banach, \; real\text{-}normed\text{-}div\text{-}algebra\}) \Longrightarrow summable \; (\lambda n. \; p \; x \; y \; n *_R z \; \hat{} \; n)$
⟨*proof*⟩

lemma *convergence-G-less-1*:
fixes z :: - :: $\{banach, \; real\text{-}normed\text{-}field\}$
assumes z: $norm \; z < 1$ **shows** $convergence\text{-}G \; x \; y \; z$
⟨*proof*⟩

lemma *convergence-norm-U*:
fixes z :: - :: *real-normed-div-algebra*
assumes z: $convergence\text{-}G \; x \; y \; z$
shows $summable \; (\lambda n. \; u \; x \; y \; n * norm \; z \; \hat{} \; Suc \; n)$
⟨*proof*⟩

lemma *U-cases*: $U\ s\ s = 1 \lor U\ s\ s < 1$
⟨*proof*⟩

5.4.2 U equals infinite sum of u

lemma *u-sums-U*: *u x y sums U x y*
⟨*proof*⟩

lemma *accessibleI-U*:
assumes $x \in S$ $0 < U\ x\ y$ **shows** *accessible x y*
⟨*proof*⟩

lemma *F-nonneg*: $0 \le F\ x\ y$ ⟨*proof*⟩

lemma *F-le-1*: $F\ x\ y \le 1$ ⟨*proof*⟩

lemma *f-sums-F*: *f x y sums F x y*
⟨*proof*⟩

lemma *U-le-F*: $U\ x\ y \le F\ x\ y$
⟨*proof*⟩

lemma *convergence-norm-F*:
fixes *z* :: - :: *real-normed-div-algebra*
assumes *z*: *convergence-G x y z*
shows *summable* ($\lambda n.\ f\ x\ y\ n * norm\ z\ \hat{}\ n$)
⟨*proof*⟩

lemma *U-bounded*: $0 \le U\ x\ y$ $U\ x\ y \le 1$
⟨*proof*⟩

5.5 Generating functions of G, U, and F

definition $gf\text{-}U\ x\ y\ z = (\sum n.\ u\ x\ y\ n *_R z\ \hat{}\ Suc\ n)$

definition $gf\text{-}G\ x\ y\ z = (\sum n.\ p\ x\ y\ n *_R z\ \hat{}\ n)$

definition $gf\text{-}F\ x\ y\ z = (\sum n.\ f\ x\ y\ n * z\ \hat{}\ n)$

lemma *lim-gf-G*: (($\lambda z.$ *ereal* (*gf-G x y z*)) ---> *G x y*) (*at-left* (*1*::*real*))
⟨*proof*⟩

lemma *gf-U-eq-U*: *gf-U x y 1* = *U x y*
⟨*proof*⟩

lemma *gf-F-eq-F*: *gf-F x y 1* = *F x y*
⟨*proof*⟩

lemma *gf-F-le-1*:
fixes *z* :: *real*

assumes *z*: $0 \le z$ $z \le 1$
shows *gf-F x y z* $\le$ *1*
⟨*proof*⟩

lemma *gf-G-nonneg*:
fixes *z* :: *real*
shows $0 \le z \Longrightarrow z < 1 \Longrightarrow 0 \le$ *gf-G x y z*
⟨*proof*⟩

lemma *gf-F-nonneg*:
fixes *z* :: *real*
shows $0 \le z \Longrightarrow z < 1 \Longrightarrow 0 \le$ *gf-F x y z*
⟨*proof*⟩

lemma *convergence-U*:
fixes *z* :: - :: *banach*
shows *convergence-G x y z* $\Longrightarrow$ *summable* ($\lambda n.$ *u x y n* $*$ *z* ^ *Suc n*)
⟨*proof*⟩

lemma *gf-G-eq-gf-F*:
assumes *x*[*simp*]: $x \in S$
assumes *z*: *norm z* < *1*
shows *gf-G x y z* = *gf-F x y z* $*$ *gf-G y y z*
⟨*proof*⟩

5.5.1 Relate *gf-G* and *gf-U*

lemma *gf-G-eq-gf-U*:
fixes *z* :: *'z* :: {*banach*, *real-normed-field*}
assumes *x*[*simp*]: $x \in S$
assumes *z*: *convergence-G x x z*
shows *gf-G x x z* = *1* / (*1* − *gf-U x x z*) *gf-U x x z* $\neq$ *1*
⟨*proof*⟩

lemma *gf-G-pos*:
fixes *z* :: *real*
assumes *z*: $0 < z$ $z < 1$ **and** $*$: *accessible x y*
shows *0* < *gf-G x y z*
⟨*proof*⟩

lemma *gf-U*: (*gf-U x y* −−−> *U x y*) (*at-left 1*)
⟨*proof*⟩

lemma *gf-F*: (*gf-F x y* −−−> *F x y*) (*at-left 1*)
⟨*proof*⟩

5.6 Recurrent states and *H*

definition *recurrent* :: *'s* ⇒ *bool* **where**
recurrent s ⟷ (*AE* ω *in paths s*. $\exists n.$ ω *n* = *s*)

lemma *recurrent-iff-U-eq-1*: *recurrent s* $\longleftrightarrow$ *U s s* = *1*
⟨*proof*⟩

definition *H s t* = $\mathcal{P}$(ω *in paths s*. *infinite* {*n*. ω *n* = *t*})

lemma *H-eq*:
assumes *s* $\in$ *S*
shows *recurrent s* $\longleftrightarrow$ *H s s* = *1*
$\neg$ *recurrent s* $\longleftrightarrow$ *H s s* = *0*
H s t = *U s t* $*$ *H t t*
⟨*proof*⟩

5.6.1 Relating *recurrent* and *G*

lemma *recurrent-iff-G-infinite*:
assumes *x* $\in$ *S*
shows *recurrent x* $\longleftrightarrow$ *G x x* = ∞
⟨*proof*⟩

5.6.2 *recurrent* is invariant

lemma *recurrent-iffI-communicating*:
assumes (*x*, *y*) $\in$ *communicating*
shows *recurrent x* $\longleftrightarrow$ *recurrent y*
⟨*proof*⟩

5.6.3 *recurrent* and *H* and *U*

lemma *recurrent-accessible*:
assumes *recurrent x accessible x y*
shows *U y x* = *1* *H y x* = *1* *recurrent y* (*x*, *y*) $\in$ *communicating*
⟨*proof*⟩

lemma *recurrent-class*:
assumes *recurrent x*
shows {*y*. *accessible x y*} = *communicating* `` {*x*}
⟨*proof*⟩

lemma *irreducible-recurrent-class*:
assumes *x* $\in$ *S* *recurrent x* **shows** {*y*. *accessible x y*} $\in$ *S* // *communicating*
⟨*proof*⟩

5.7 Essential classes

definition *essential-class* :: *'s set* $\Rightarrow$ *bool* **where**
essential-class S' $\longleftrightarrow$ *S'* $\in$ *S* // *communicating* $\wedge$ $\neg$ ($\exists$ *s*$\in$*S'*. $\exists$ *t*$\in$*S* − *S'*. *accessible s t*)

lemma *essential-classI*:

assumes C: $C \in S$ // *communicating*
assumes *eq*: $\bigwedge x\ y.\ x \in C \Longrightarrow$ *accessible* $x\ y \Longrightarrow y \in C$
shows *essential-class* C
⟨*proof*⟩

lemma *essential-classI2*:
assumes $X \neq \{\}\ X \subseteq S$
assumes *accI*: $\bigwedge x\ y.\ x \in X \Longrightarrow y \in X \Longrightarrow$ *accessible* $x\ y$
assumes *ED*: $\bigwedge x\ y.\ x \in X \Longrightarrow y \in E\ x \Longrightarrow y \in X$
shows *essential-class* X
⟨*proof*⟩

lemma *essential-recurrent-class*:
assumes $x \in S$ *recurrent* x **shows** *essential-class* (*communicating* `` $\{x\}$)
⟨*proof*⟩

lemma *essential-classD1*:
essential-class $C \Longrightarrow x \in C \Longrightarrow x \in S$
⟨*proof*⟩

lemma *essential-classD2*:
essential-class $C \Longrightarrow x \in C \Longrightarrow$ *accessible* $x\ y \Longrightarrow y \in C$
⟨*proof*⟩

lemma *essential-classD3*:
essential-class $C \Longrightarrow x \in C \Longrightarrow y \in C \Longrightarrow (x, y) \in$ *communicating*
⟨*proof*⟩

lemma *finite-essential-class-imp-recurrent*:
assumes C: *essential-class* C *finite* C **and** x: $x \in C$
shows *recurrent* x
⟨*proof*⟩

lemma *essential-class-iff-recurrent*:
finite $C \Longrightarrow C \in S$ // *communicating* $\Longrightarrow$ *essential-class* $C \longleftrightarrow (\forall x \in C.$ *recurrent* $x)$
⟨*proof*⟩

5.8 Hitting times

definition $t\ x\ \omega = ($*if* $\exists i.\ \omega\ i = x$ *then ereal* (*Suc* (*LEAST* $i.\ \omega\ i = x$)) *else* $\infty)$

definition *gf-U'* $x\ y\ z = (\sum n.\ u\ x\ y\ n * Suc\ n * z\ \hat{}\ n)$

lemma *summable-gf-U'*:
assumes z: *norm* $z < 1$
shows *summable* $(\lambda n.\ u\ x\ y\ n * Suc\ n * z\ \hat{}\ n)$
⟨*proof*⟩

lemma *DERIV-gf-U*:
  fixes z :: *real* **assumes** z: $0 < z$ $z < 1$
  shows *DERIV* (*gf-U* x y) z :> *gf-U'* x y z
  ⟨*proof*⟩

lemma *integral-t-eq*:
  recurrent x $\Longrightarrow$ ($\int^+\omega$. t x $\omega\partial$*paths* x) = ($\int^+\omega$. *ereal* (*Suc* (*LEAST* n. ω n = x)) ∂*paths* x)
  ⟨*proof*⟩

lemma *one-le-integral-t*:
  assumes x: *recurrent* x **shows** $1 \leq$ ($\int^+\omega$. t x ω ∂*paths* x)
  ⟨*proof*⟩

5.8.1 Integral equals infinite sum of u

lemma *integral-t-eq-suminf*:
  assumes x: *recurrent* x **and** [*simp*]: $x \in S$
  shows ($\int^+\omega$. t x $\omega\partial$*paths* x) = ($\sum$ i. *ereal* (u x x i $*$ *Suc* i))
⟨*proof*⟩

5.8.2 *gf-U'* approaches the integral of t

lemma *gf-U'-tendsto-integral-t*:
  assumes x: *recurrent* x $x \in S$
  shows ((λz. *ereal* (*gf-U'* x x z)) −−−> ($\int^+\omega$. t x $\omega\partial$*paths* x)) (*at-left* *1*)
  ⟨*proof*⟩

lemma *gf-U'-pos*:
  fixes z :: *real*
  assumes z: $0 < z$ $z < 1$ **and** U x $y \neq 0$
  shows $0 <$ *gf-U'* x y z
  ⟨*proof*⟩

lemma *inverse-gf-U'-tendsto*:
  assumes *recurrent* y $y \in S$
  shows ((λx. − *1* / − *gf-U'* y y x) −−−> *real* (*1* / *integral*^{P} (*paths* y) (t y))) (*at-left* (*1*::*real*))
⟨*proof*⟩

5.9 Positive recurrent

definition *pos-recurrent* x $\longleftrightarrow$ *recurrent* x $\wedge$ ($\int^+\omega$. t x $\omega\partial$*paths* x) $\neq \infty$

lemma *pos-recurrentI-communicating*:
  assumes y: *pos-recurrent* y **and** x: (y, x) $\in$ *communicating*
  shows *pos-recurrent* x
⟨*proof*⟩

lemma *pos-recurrent-iffI-communicating*:

$(y, x) \in communicating \Longrightarrow pos\text{-}recurrent\ y \longleftrightarrow pos\text{-}recurrent\ x$
⟨proof⟩

5.10 Stationary distribution

definition *stationary-distribution* :: *'s measure* ⇒ *bool* **where**
$stationary\text{-}distribution\ N \longleftrightarrow N = point\text{-}measure\ S\ (\lambda x.\ \int^+ s.\ emeasure\ (K\ s)\ \{x\}\ \partial N)$

definition $stat\ C = point\text{-}measure\ S\ (\lambda x.\ inverse\ (\int^+ \omega.\ t\ x\ \omega\ \partial paths\ x) * indicator\ C\ x)$

lemma *sets-stat*[*simp*]: $sets\ (stat\ C) = Pow\ S$
⟨proof⟩

lemma *space-stat*[*simp*]: $space\ (stat\ C) = S$
⟨proof⟩

lemma *stationary-distributionD-eq*: $stationary\text{-}distribution\ N \Longrightarrow$
$N = point\text{-}measure\ S\ (\lambda x.\ \int^+ s.\ emeasure\ (K\ s)\ \{x\}\ \partial N)$
⟨proof⟩

lemma *stationary-distributionD-emeasure*:
assumes *N*: *stationary-distribution N prob-space N* **and** *A*: $A \subseteq S$
shows $emeasure\ N\ A = (\int^+ s.\ emeasure\ (K\ s)\ A\ \partial N)$
⟨proof⟩

lemma *stationary-distributionI*:
assumes *finite-measure N*
assumes *N*[*simp*]: $sets\ N = Pow\ S$
assumes *le*: $\bigwedge y.\ y \in S \Longrightarrow (\int x.\ measure\ (K\ x)\ \{y\}\ \partial N) \leq measure\ N\ \{y\}$
shows *stationary-distribution N*
⟨proof⟩

lemma *stationary-distribution-iterate*:
assumes *N*: *stationary-distribution N* **and** *finite-measure N* **and** [*simp*]: $y \in S$
shows $measure\ N\ \{y\} = (\int x.\ p\ x\ y\ n\ \partial N)$
⟨proof⟩

lemma *stat-subprob*:
assumes *C*: *essential-class C* **and** *pos*: $\forall c \in C.\ pos\text{-}recurrent\ c$
shows $emeasure\ (stat\ C)\ C \leq 1$
⟨proof⟩

lemma *emeasure-stat-not-C*:
assumes $y \notin C$
shows $emeasure\ (stat\ C)\ \{y\} = 0$
⟨proof⟩

5.10.1 *stationary-distribution* implies *pos-recurrent*

lemma *stationary-distributionD*:
assumes C: *essential-class* C
assumes N: *stationary-distribution* N **and** *prob-space* N **and** *null*: *measure* N $(S - C) = 0$
shows $\forall x{\in}C.$ *pos-recurrent* x N $=$ *stat* C
⟨*proof*⟩

lemma *stationary-distribution-imp-int-t*:
assumes C: *essential-class* C *stationary-distribution* N *prob-space* N *measure* N $(S - C) = 0$
assumes x: $x \in C$ **shows** $(\int^{+}\omega.\ t\ x\ \omega\ \partial paths\ x) = 1\ /\ measure\ N\ \{x\}$
⟨*proof*⟩

5.11 Aperiodic classes

definition *period-set* $x = \{i.\ 0 < i \wedge 0 < p\ x\ x\ i\ \}$

definition *period* $C = (SOME\ d.\ \forall x{\in}C.\ d = Gcd\ (period\text{-}set\ x))$

definition *aperiodic* $C \longleftrightarrow C \in S\ //\ communicating \wedge period\ C = 1$

definition *not-ephemeral* $C \longleftrightarrow C \in S\ //\ communicating \wedge \neg\ (\exists x.\ C = \{x\} \wedge p\ x\ x\ 1 = 0)$

end

inductive-set *monoid-closure* :: *nat set* ⇒ *int set* **for** S **where**
base: $s \in S \Longrightarrow int\ s \in monoid\text{-}closure\ S$
| *plus*: $s \in monoid\text{-}closure\ S \Longrightarrow t \in monoid\text{-}closure\ S \Longrightarrow s + t \in monoid\text{-}closure\ S$
| *diff*: $s \in monoid\text{-}closure\ S \Longrightarrow t \in monoid\text{-}closure\ S \Longrightarrow s - t \in monoid\text{-}closure\ S$

lemma *Gcd-nonneg*: $0 \le Gcd\ (S :: int\ set)$
⟨*proof*⟩

lemma *Gcd-eq-Gcd-monoid-closure*:
fixes S :: *nat set*
shows $Gcd\ S = Gcd\ (monoid\text{-}closure\ S)$
⟨*proof*⟩

lemma *monoid-closure-nat-mult*:
$s \in monoid\text{-}closure\ S \Longrightarrow int\ n * s \in monoid\text{-}closure\ S$
⟨*proof*⟩

lemma *monoid-closure-uminus*:
assumes s: $s \in monoid\text{-}closure\ S$ **shows** $-s \in monoid\text{-}closure\ S$
⟨*proof*⟩

lemma *monoid-closure-int-mult*:
$s \in$ *monoid-closure* $S \Longrightarrow i * s \in$ *monoid-closure* S
⟨*proof*⟩

lemma *monoid-closure-setsum*:
assumes X: *finite* X $X \neq \{\}$ $X \subseteq S$
shows $(\sum x{\in}X.\ a\ x * int\ x) \in$ *monoid-closure* S
⟨*proof*⟩

lemma *Gcd-monoid-closure-in-monoid-closure*:
fixes S :: *nat set*
assumes $s \in S$ $s \neq 0$
shows *Gcd* (*monoid-closure* S) $\in$ *monoid-closure* S
⟨*proof*⟩

lemma *Gcd-in-monoid-closure*:
fixes S :: *nat set*
shows $s \in S \Longrightarrow s \neq 0 \Longrightarrow$ *Gcd* $S \in$ *monoid-closure* S
⟨*proof*⟩

lemma *eventually-mult-Gcd*:
fixes S :: *nat set*
assumes S: $\bigwedge s\ t.\ s \in S \Longrightarrow t \in S \Longrightarrow s + t \in S$
assumes s: $s \in S$ $s \neq 0$
shows *eventually* $(\lambda m.\ m *$ *Gcd* $S \in S)$ *sequentially*
⟨*proof*⟩

context *Discrete-Markov-Kernel*
begin

lemma *Gcd-period-set-invariant*:
assumes c: $(x, y) \in$ *communicating*
shows *Gcd* (*period-set* x) = *Gcd* (*period-set* y)
⟨*proof*⟩

lemma *period-eq*:
assumes $C \in S$ // *communicating* $x \in C$
shows *period* C = *Gcd* (*period-set* x)
⟨*proof*⟩

lemma *not-ephemeralD*:
assumes C: *not-ephemeral* C $x \in C$
shows $\exists n{>}0.\ 0 < p\ x\ x\ n$
⟨*proof*⟩

lemma *not-ephemeralD-pos-period*:
assumes C: *not-ephemeral* C
shows $0 <$ *period* C

⟨*proof*⟩

lemma *period-posD*:
assumes *C*: *C* ∈ *S* // *communicating* **and** *0* < *period C x* ∈ *C*
shows ∃ *n*>*0*. *0* < *p x x n*
⟨*proof*⟩

lemma *eventually-periodic*:
assumes *C*: *C* ∈ *S* // *communicating 0* < *period C x* ∈ *C*
shows *eventually* (λ*m*. *0* < *p x x* (*m* ∗ *period C*)) *sequentially*
⟨*proof*⟩

5.11.1 *aperiodic* is equal to non-zero *p*

lemma *aperiodic-eventually-recurrent*:
aperiodic C ⟷ *C* ∈ *S* // *communicating* ∧ (∀ *x*∈*C*. *eventually* (λ*m*. *0* < *p x x m*) *sequentially*)
⟨*proof*⟩

end

5.12 Product Construction for Markov Chains

locale *Pair-Discrete-Markov-Kernel* =
K1!: *Discrete-Markov-Kernel S1 K1* + *K2*!: *Discrete-Markov-Kernel S2 K2* **for** *S1 S2 K1 K2*
begin

definition *Sp* = *S1* × *S2*
definition *Kp* = (λ(*s1*, *s2*). *K1 s1* $\bigotimes_M$ *K2 s2*)

lemma *SpI*[*simp*]: *x1* ∈ *S1* ⟹ *x2* ∈ *S2* ⟹ (*x1*, *x2*) ∈ *Sp*
⟨*proof*⟩

lemma *measurable-K1K2*[*measurable*]: *x1* ∈ *S1* ⟹ *x2* ∈ *S2* ⟹ *f* ∈ *borel-measurable* (*K1 x1* $\bigotimes_M$ *K2 x2*)
⟨*proof*⟩

sublocale *K1K2*!: *pair-prob-space K1 s1 K2 s2* **for** *s1 s2* ⟨*proof*⟩
sublocale *Kp*!: *prob-space Kp s* **for** *s*
⟨*proof*⟩

sublocale *P*!: *Discrete-Markov-Kernel Sp Kp*
⟨*proof*⟩

lemma *sets-Sp*: *sets* (*count-space Sp*) = *sets* (*count-space S1* $\bigotimes_M$ *count-space S2*)
⟨*proof*⟩

lemma *sets-S-seq-Sp*: *sets P.S-seq* = *sets* (Π_M *i*∈*UNIV*. *count-space S1* $\bigotimes_M$ *count-space S2*)
⟨*proof*⟩

lemma *space-S-seq-Sp*: *space P.S-seq* = *space* (Π_M *i*∈*UNIV*. *count-space S1* $\bigotimes_M$ *count-space S2*)
⟨*proof*⟩

lemma *prod-measurable*: (λ(ω*1*, ω*2*) *n*. (ω*1 n*, ω*2 n*)) ∈ *measurable* (*K1.paths x1* $\bigotimes_M$ *K2.paths x2*) *P.S-seq*
⟨*proof*⟩

lemma *P-paths-eq-prod*:
assumes [*simp*]: *x1* ∈ *S1 x2* ∈ *S2*
shows *P.paths* (*x1*, *x2*) = *distr* (*K1.paths x1* $\bigotimes_M$ *K2.paths x2*) *P.S-seq* (λ(ω*1*, ω*2*) *n*. (ω*1 n*, ω*2 n*))
(**is** *?L* = *?R*)
⟨*proof*⟩

lemma *prod-eq-prob-paths*:
assumes *x*[*simp*]: *x1* ∈ *S1 x2* ∈ *S2* **and** [*measurable*]: *Measurable.pred K1.S-seq P1 Measurable.pred K2.S-seq P2*
shows $\mathcal{P}$(ω *in K1.paths x1*. *P1* ω) ∗ $\mathcal{P}$(ω *in K2.paths x2*. *P2* ω) = $\mathcal{P}$(ω *in P.paths* (*x1*, *x2*). *P1* (*fst* ∘ ω) ∧ *P2* (*snd* ∘ ω))
⟨*proof*⟩

lemma *p-eq-p1-p2*:
x1 ∈ *S1* ⟹ *x2* ∈ *S2* ⟹ *y1* ∈ *S1* ⟹ *y2* ∈ *S2* ⟹ *P.p* (*x1*, *x2*) (*y1*, *y2*) *n* = *K1.p x1 y1 n* ∗ *K2.p x2 y2 n*
⟨*proof*⟩

lemma *Sp-iff*: (*a*, *b*) ∈ *Sp* ⟷ *a* ∈ *S1* ∧ *b* ∈ *S2*
⟨*proof*⟩

lemma *P-accessibleD*:
assumes *P.accessible* (*x1*, *x2*) (*y1*, *y2*)**shows** *K1.accessible x1 y1 K2.accessible x2 y2*
⟨*proof*⟩

lemma *aperiodicI-pair*:
assumes *C1*: *K1.aperiodic C1* **and** *C2*: *K2.aperiodic C2*
shows *P.aperiodic* (*C1* × *C2*)
⟨*proof*⟩

lemma *stationary-distributionI-pair*:
assumes *N1*: *K1.stationary-distribution N1 prob-space N1*
assumes *N2*: *K2.stationary-distribution N2 prob-space N2*
shows *P.stationary-distribution* (*N1* $\bigotimes_M$ *N2*)
⟨*proof*⟩

end

5.13 Stationary Distribution is the Limit of the Marginal Distributions

context *Discrete-Markov-Kernel*
begin

lemma *stationary-distribution-imp-limit*:
assumes *aperiodic C essential-class C* **and** *N*: *stationary-distribution N prob-space N measure N* (*S* − *C*) = *0*
assumes [*simp*]: *y* ∈ *C*
shows (λ*n*. ∫ *x*. |*p y x n* − *measure N* {*x*}| ∂*count-space C*) −−−−> *0*
(**is** *?L* −−−−> *0*)
⟨*proof*⟩

lemma *stationary-distribution-imp-p-limit*:
assumes *aperiodic C essential-class C* **and** *N*: *stationary-distribution N prob-space N measure N* (*S* − *C*) = *0*
assumes [*simp*]: *x* ∈ *C y* ∈ *C*
shows *p x y* −−−−> *measure N* {*y*}
⟨*proof*⟩

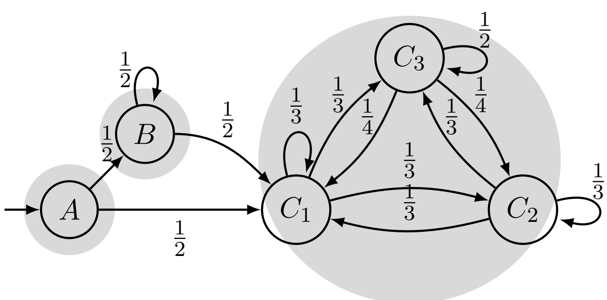
end

end

theory *Example-A*
imports *Classifying-Markov-Chains*
begin

6 Example A

We formalize the following Markov chain:



First we define the state space as its own type:

datatype *state* = *A* | *B* | *C1* | *C2* | *C3*

Now the state space is *UNIV* :: *state set*

lemma *UNIV-state*: *UNIV* = {*A*, *B*, *C1*, *C2*, *C3*}
⟨*proof*⟩

instance *state* :: *finite*
⟨*proof*⟩

The transition function *tau* is easily defined using the case statement, this allows us to give a sparse specification as all *0* cases are collected at the end.

definition *tau* :: *state* ⇒ *state* ⇒ *real* **where**
tau s t = (*case* (*s*, *t*) *of*
(*A*, *B*) ⇒ *1* / *2* | (*A*, *C1*) ⇒ *1* / *2*
| (*B*, *B*) ⇒ *1* / *2* | (*B*, *C1*) ⇒ *1* / *2*
| (*C1*, *C1*) ⇒ *1* / *3* | (*C1*, *C2*) ⇒ *1* / *3* | (*C1*, *C3*) ⇒ *1* / *3*
| (*C2*, *C1*) ⇒ *1* / *3* | (*C2*, *C2*) ⇒ *1* / *3* | (*C2*, *C3*) ⇒ *1* / *3*
| (*C3*, *C1*) ⇒ *1* / *4* | (*C3*, *C2*) ⇒ *1* / *4* | (*C3*, *C3*) ⇒ *1* / *2*
| - ⇒ *0*)

We use the *finite-pmf*-locale which introduces the point measure *tau.M*, and provides us with the necessary simplifier setup.

interpretation *tau*: *finite-pmf UNIV tau s* **for** *s*
⟨*proof*⟩

interpretation *A*: *Discrete-Markov-Kernel UNIV tau.M*
⟨*proof*⟩

6.1 The essential classs {*C1*, *C2*, *C3*}

lemma *A-E-eq*:
A.E x = (*case x of A* ⇒ {*B*, *C1*} | *B* ⇒ {*B*, *C1*} | - ⇒ {*C1*, *C2*, *C3*})
⟨*proof*⟩

lemma *A-essential*: *A.essential-class* {*C1*, *C2*, *C3*}
⟨*proof*⟩

lemma *A-aperiodic*: *A.aperiodic* {*C1*, *C2*, *C3*}
⟨*proof*⟩

6.2 The stationary distribution *n*

Similar to *tau* we introduce *n* using the *finite-pmf*-locale.

definition *n* :: *state* ⇒ *real* **where**
n = (λ*C1* ⇒ *0.3* | *C2* ⇒ *0.3* | *C3* ⇒ *0.4* | - ⇒ *0*)

interpretation *n*: *finite-pmf UNIV n*
⟨*proof*⟩

lemma *stationary-distribution-N*: *A.stationary-distribution n.M*
⟨*proof*⟩

lemma *exclusive-N*: *measure n.M* (*UNIV* − {*C1*, *C2*, *C3*}) = *0*
⟨*proof*⟩

lemma *n-is-limit*:
assumes *x*: *x* ∈ {*C1*, *C2*, *C3*} **and** *y*: *y* ∈ {*C1*, *C2*, *C3*}
shows (*A.p x y*) −−−−> *n y*
⟨*proof*⟩

lemma *C-is-pos-recurrent*: *x* ∈ {*C1*, *C2*, *C3*} ⟹ *A.pos-recurrent x*
⟨*proof*⟩

lemma *C-recurrence-time*:
assumes *x*: *x* ∈ {*C1*, *C2*, *C3*}
shows (∫ $^+$ω. *A.t x* ω ∂*A.paths x*) = *1* / *n x*
⟨*proof*⟩

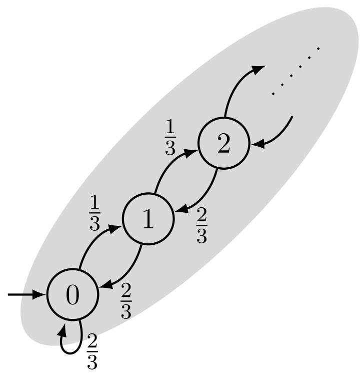
end

theory *Example-B*
imports *Classifying-Markov-Chains*
begin

lemma *one-divide-ereal*: *x* ≠ *0* ⟹ *1* / *ereal x* = *ereal* (*1* / *x*)
⟨*proof*⟩

7 Example B

We now formalize the following Markov chain:



As state space we have the set of natural numbers, the transition function *tau* has three cases:

definition *tau* :: *nat* ⇒ *nat* ⇒ *real* **where**
tau x y = (*if x* + *1* = *y then 1* / *3*
else if x − *1* = *y then 2* / *3*
else 0)

For the special case when $x = 0$ we have $x - 1 = 0$ and hence *tau 0 0* = *2* / *3*.

We pack this transition function into a discrete Markov kernel, by showing that it forms a probability space.

7.1 Discrete Markov Kernel

definition *K x* = *point-measure UNIV* (*tau x*)

lemma *sets-K*[*simp*]: *sets* (*K x*) = *UNIV* **and** *space-K*[*simp*]: *space* (*K x*) = *UNIV*
⟨*proof*⟩

lemma *AE-K*[*simp*]: (*AE x in K y. P x*) ⟷ *P* (*y* + *1*) ∧ *P* (*y* − *1*)
⟨*proof*⟩

lemma *emeasure-K*[*simp*]:
emeasure (*K x*) *X* = *1/3* ∗ *indicator X* (*x* + *1*) + *2/3* ∗ *indicator X* (*x* − *1*)
⟨*proof*⟩

interpretation *K*: *prob-space K x* **for** *x*
⟨*proof*⟩

We call the locale of the Markov chain *B*, hence all constants and theorems from this Markov chain get a *B* prefix.

interpretation *B*: *Discrete-Markov-Kernel UNIV K*
⟨*proof*⟩

7.2 Enabled, accessible and communicating states

For each step the predecessor and the successor are enabled (in the *0*::′*a* case, the predecessor is again *0*::′*a*. Hence every state is accessible from everywhere and every states is communicating with each other state. Finally we know that the state space is an essential class.

lemma *B-E-eq*: *B.E x* = {*x* − *1*, *x* + *1*}
⟨*proof*⟩

lemma *B-E-Suc*: *Suc x* ∈ *B.E x x* ∈ *B.E* (*Suc x*)
⟨*proof*⟩

lemma *B-accessible*[*intro*]: *B.accessible i j*
⟨*proof*⟩

lemma *B-communicating*[*intro*]: $(i, j) \in$ *B.communicating*
⟨*proof*⟩

lemma *B-essential*: *B.essential-class UNIV*
⟨*proof*⟩

7.3 B is aperiodic

lemma *B-aperiodic*: *B.aperiodic UNIV*
⟨*proof*⟩

7.4 The stationary distribution N

definition N = *point-measure UNIV* (λi. *ereal* (*1* / *2*) ^ *Suc i*)

lemma *borel-measurable-N*[*simp*]: $f \in$ *borel-measurable N*
⟨*proof*⟩

lemma *sets-N*[*simp*]: *sets N = UNIV* **and** *space-N*[*simp*]: *space N = UNIV*
⟨*proof*⟩

lemma *emeasure-N-finite*[*simp*]:
finite A $\Longrightarrow$ *emeasure N A* = ($\sum i \in A$. *ereal* (*1* / *2*)^*Suc i*)
⟨*proof*⟩

lemma *prob-space-N*: *prob-space N*
⟨*proof*⟩

lemma *stationary-distribution-N*: *B.stationary-distribution N*
⟨*proof*⟩

lemma *exclusive-N*: *measure N* (*UNIV* − *UNIV*) = *0*
⟨*proof*⟩

7.5 Limit behavior and recurrence times

lemma *limit*: (*B.p i j*) ----> (*1*/*2*)^*Suc j*
⟨*proof*⟩

lemma *pos-recurrent*: *B.pos-recurrent i*
⟨*proof*⟩

lemma *recurrence-time*: ($\int^{+}\omega$. *B.t i* ω ∂*B.paths i*) = *2*^*Suc i*
⟨*proof*⟩

end

References

[1] J. Hölzl. Analyzing discrete-time markov chains with countable state space in Isabelle/HOL. Draft (http://home.in.tum.de/hoelzl/classifying/hoelzl2013markov.pdf).