Stochastic Matrices and the Perron–Frobenius Theorem*

René Thiemann

December 14, 2021

Abstract

Stochastic matrices are a convenient way to model discrete-time and finite state Markov chains. The Perron–Frobenius theorem tells us something about the existence and uniqueness of non-negative eigenvectors of a stochastic matrix.

In this entry, we formalize stochastic matrices, link the formalization to the existing AFP-entry on Markov chains, and apply the Perron–Frobenius theorem to prove that stationary distributions always exist, and they are unique if the stochastic matrix is irreducible.

Contents

1 Introduction

In their AFP entry Markov Models [2], Hölzl and Nipkow provide a framework for specifying discrete- and continuous-time Markov chains.

In the following, we instantiate their framework by formalizing right-stochastic matrices and stochastic vectors. These vectors encode probability

*Supported by FWF (Austrian Science Fund) project Y757.

mass functions over a finite set of states, whereas stochastic matrices can be utilized to model discrete-time and finite space Markov chains.

The formulation of Markov chains as matrices has the advantage that certain concepts can easily be expressed via matrices. For instance, a stationary distribution is nothing else than a non-negative real eigenvector of the transition matrix for eigenvalue 1. As a consequence, we can derive certain properties on Markov chains using results on matrices. To be more precise, we utilize the formalization of the Perron–Frobenius theorem [1] to prove that a stationary distribution always exists, and that it is unique if the transition matrix is irreducible.

2 Stochastic Matrices

We define a type for stochastic vectors and right-stochastic matrices, i.e., non-negative real vectors and matrices where the sum of each column is 1. For this type we define a matrix-vector multplication, i.e., we show that $A*v$ is a stochastic vector, if A is a right-stochastic matrix and v a stochastic vector.

theory *Stochastic-Matrix*
  imports *Perron-Frobenius.Perron-Frobenius-Aux*
begin

definition *non-neg-vec* :: *'a :: linordered-idom ^ 'n ⇒ bool* **where**
  non-neg-vec A ≡ (∀ i. A $ i ≥ 0)

definition *stoch-vec* :: *'a :: comm-ring-1 ^ 'n ⇒ bool* **where**
  stoch-vec v = (sum (λ i. v $ i) UNIV = 1)

definition *right-stoch-mat* :: *'a :: comm-ring-1 ^ 'n ^ 'm ⇒ bool* **where**
  right-stoch-mat a = (∀ j. stoch-vec (column j a))

typedef *'i st-mat = { a :: real ^ 'i ^ 'i. non-neg-mat a ∧ right-stoch-mat a}*
  morphisms *st-mat Abs-st-mat*
  by *(rule exI[of - χ i j. if i = undefined then 1 else 0],*
    auto simp: non-neg-mat-def elements-mat-h-def right-stoch-mat-def stoch-vec-def column-def)

setup-lifting *type-definition-st-mat*

typedef *'i st-vec = { v :: real ^ 'i. non-neg-vec v ∧ stoch-vec v}*
  morphisms *st-vec Abs-st-vec*
  by *(rule exI[of - χ i. if i = undefined then 1 else 0],*
    auto simp: non-neg-vec-def stoch-vec-def)

setup-lifting *type-definition-st-vec*

lift-definition *transition-vec-of-st-mat* :: *'i :: finite st-mat ⇒ 'i ⇒ 'i st-vec*

is λ *a i. column i a*
by (*auto simp*: *right-stoch-mat-def non-neg-mat-def stoch-vec-def elements-mat-h-def non-neg-vec-def column-def*)

lemma *non-neg-vec-st-vec*: *non-neg-vec* (*st-vec v*)
by (*transfer*, *auto*)

lemma *non-neg-mat-mult-non-neg-vec*: *non-neg-mat a* $\Longrightarrow$ *non-neg-vec v* $\Longrightarrow$ *non-neg-vec* (*a* $*v$ *v*)
unfolding *non-neg-mat-def non-neg-vec-def elements-mat-h-def*
by (*auto simp*: *matrix-vector-mult-def intro*!: *sum-nonneg*)

lemma *right-stoch-mat-mult-stoch-vec*: **assumes** *right-stoch-mat a* **and** *stoch-vec v*
shows *stoch-vec* (*a* $*v$ *v*)
proof −
note $*$ = *assms*[*unfolded right-stoch-mat-def column-def stoch-vec-def*, *simplified*]
have *stoch-vec* (*a* $*v$ *v*) = (($\sum$ *i*$\in$*UNIV*. $\sum$ *j*$\in$*UNIV*. *a* \$ *i* \$ *j* $*$ *v* \$ *j*) = *1*)
(**is** - = (*?sum* = *1*))
unfolding *stoch-vec-def matrix-vector-mult-def* **by** *auto*
also have *?sum* = ($\sum$ *j*$\in$*UNIV*. $\sum$ *i*$\in$*UNIV*. *a* \$ *i* \$ *j* $*$ *v* \$ *j*)
by (*rule sum.swap*)
also have ... = ($\sum$ *j*$\in$*UNIV*. *v* \$ *j*)
by (*rule sum.cong*[*OF refl*], *insert* $*$, *auto simp*: *sum-distrib-right*[*symmetric*])
also have ... = *1* **using** $*$ **by** *auto*
finally show *?thesis* **by** *simp*
qed

lift-definition *st-mat-times-st-vec* :: *'i* :: *finite st-mat* $\Rightarrow$ *'i st-vec* $\Rightarrow$ *'i st-vec*
(**infixl** $*$*st 70*) **is** ($*v$)
using *right-stoch-mat-mult-stoch-vec non-neg-mat-mult-non-neg-vec* **by** *auto*

lift-definition *to-st-vec* :: *real* ^ *'i* $\Rightarrow$ *'i st-vec* **is**
λ *x. if stoch-vec x* $\wedge$ *non-neg-vec x then x else* (χ *i. if i* = *undefined then 1 else 0*)
by (*auto simp*: *non-neg-vec-def stoch-vec-def*)

lemma *right-stoch-mat-st-mat*: *right-stoch-mat* (*st-mat A*)
by *transfer auto*

lemma *non-neg-mat-st-mat*: *non-neg-mat* (*st-mat A*)
by (*transfer*, *auto simp*: *non-neg-mat-def elements-mat-h-def*)

lemma *st-mat-mult-st-vec*: *st-mat A* $*v$ *st-vec X* = *st-vec* (*A* $*$*st X*) **by** (*transfer*, *auto*)

lemma *st-vec-nonneg*[*simp*]: *st-vec x* \$ *i* $\geq$ *0*
using *non-neg-vec-st-vec*[*of x*] **by** (*auto simp*: *non-neg-vec-def*)

lemma *st-mat-nonneg*[*simp*]: *st-mat x* $ *i* $*h j* ≥ *0*
using *non-neg-mat-st-mat*[*of x*] **by** (*auto simp*: *non-neg-mat-def elements-mat-h-def*)

end

3 Stochastic Vectors and Probability Mass Functions

We prove that over a finite type, stochastic vectors and probability mass functions are essentially the same thing: one can convert between both representations.

theory *Stochastic-Vector-PMF*
imports *Stochastic-Matrix*
HOL−Probability.Probability-Mass-Function
begin

lemma *sigma-algebra-UNIV-finite*[*simp*]: *sigma-algebra* (*UNIV* :: *'a* :: *finite set*) *UNIV*
proof (*unfold-locales*, *goal-cases*)
case (*4 a b*)
show *?case* **by** (*intro exI*[*of* - {*a*−*b*}], *auto*)
qed *auto*

definition *measure-of-st-vec'* :: *'a st-vec* ⇒ *'a* :: *finite set* ⇒ *ennreal* **where**
measure-of-st-vec' x I = *sum* (λ *i*. *st-vec x* $ *i*) *I*

lemma *positive-measure-of-st-vec'*[*simp*]: *positive A* (*measure-of-st-vec' x*)
unfolding *measure-of-st-vec'-def positive-def* **by** *auto*

lemma *measure-space-measure-of-st-vec'*: *measure-space UNIV UNIV* (*measure-of-st-vec' x*)
unfolding *measure-space-def*
proof (*simp*, *simp add*: *countably-additive-def measure-of-st-vec'-def disjoint-family-on-def*, *clarify*, *goal-cases*)
case (*1 A*)
let *?x* = *st-vec x*
define *N* **where** *N* = {*i*. *A i* ≠ {}}
let *?A* = ⋃(*A* ' *N*)
have *finite B* ⟹ *B* ⊆ *?A* ⟹ ∃ *K*. *finite K* ∧ *K* ⊆ *N* ∧ *B* ⊆ ⋃(*A* ' *K*) **for** *B*
proof (*induct rule*: *finite-induct*)
case (*insert b B*)
from *insert*(*3*−*4*) **obtain** *K* **where** *K*: *finite K K* ⊆ *N B* ⊆ ⋃(*A* ' *K*) **by** *auto*
from *insert*(*4*) **obtain** *a* **where** *a*: *a* ∈ *N b* ∈ *A a* **by** *auto*
show *?case* **by** (*intro exI*[*of* - *insert a K*], *insert a K*, *auto*)
qed *auto*
from *this*[*OF* - *subset-refl*] **obtain** *K* **where** ∗: *finite K K* ⊆ *N* ⋃(*A* ' *K*) = *?A*

by *auto*
  {
    assume $K \subset N$
    then obtain n **where** $**$: $n \in N$ $n \notin K$ **by** *auto*
    from *this*[*unfolded N-def*] **obtain** a **where** a: $a \in A\ n$ **by** *auto*
    with $**$ $*$ **obtain** k **where** $***$: $k \in K$ $a \in A\ k$ **by** *force*
    from $**$ $***$ **have** $n \neq k$ **by** *auto*
    from *1*[*rule-format*, *OF this*] **have** $A\ n \cap A\ k = \{\}$ **by** *auto*
    with $***$ a **have** *False* **by** *auto*
  }
  with $*$ **have** *fin*: *finite N* **by** *auto*
  have *id*: $\bigcup (A$ ‘ *UNIV*$)$ = *?A* **unfolding** *N-def* **by** *auto*
  show ($\sum$ *i*. *ennreal* (*sum* ((\$*h*) *?x*) (*A i*))) =
    ennreal (*sum* ((\$*h*) *?x*) ($\bigcup$(*A* ‘ *UNIV*))) **unfolding** *id*
    apply (*subst suminf-finite*[*OF fin*], (*auto simp*: *N-def*)[*1*])
   apply (*subst sum-ennreal*, (*insert non-neg-vec-st-vec*[*of x*], *auto simp*: *non-neg-vec-def*
intro!: *sum-nonneg*)[*1*])
    apply (*rule arg-cong*[*of - - ennreal*])
    apply (*subst sum.UNION-disjoint*[*OF fin*], *insert 1*, *auto*)
    done
qed

context begin
setup-lifting *type-definition-measure*

lift-definition *measure-of-st-vec* :: *'a st-vec* ⇒ *'a* :: *finite measure* **is**
  λ *x*. (*UNIV*, *UNIV*, *measure-of-st-vec' x*)
  by (*auto simp*: *measure-space-measure-of-st-vec'*)

lemma *sets-measure-of-st-vec*[*simp*]: *sets* (*measure-of-st-vec x*) = *UNIV*
  unfolding *sets-def* **by** (*transfer*, *auto*)

lemma *space-measure-of-st-vec*[*simp*]: *space* (*measure-of-st-vec x*) = *UNIV*
  unfolding *space-def* **by** (*transfer*, *auto*)

lemma *emeasure-measure-of-st-vec*[*simp*]: *emeasure* (*measure-of-st-vec x*) *I* =
  sum (λ *i*. *st-vec x* \$ *i*) *I*
  unfolding *emeasure-def* **by** (*transfer'*, *auto simp*: *measure-of-st-vec'-def*)

lemma *prob-space-measure-of-st-vec*: *prob-space* (*measure-of-st-vec x*)
  by (*unfold-locales*, *intro exI*[*of - UNIV*], *auto*, *transfer*, *auto simp*: *stoch-vec-def*)
end

lift-definition *st-vec-of-pmf* :: *'i* :: *finite pmf* ⇒ *'i st-vec* **is**
  λ *pmF*. *vec-lambda* (*pmf pmF*)
proof (*intro conjI*, *goal-cases*)
  case (*2 pmF*)
  show *stoch-vec* (*vec-lambda* (*pmf pmF*))
    unfolding *stoch-vec-def*

```
    apply auto
    apply (unfold measure-pmf-UNIV[of pmF, symmetric])
    by (simp add: measure-pmf-conv-infsetsum)
qed (auto simp: non-neg-vec-def stoch-vec-def)

context pmf-as-measure
begin
lift-definition pmf-of-st-vec :: 'a :: finite st-vec ⇒ 'a pmf is measure-of-st-vec
proof (goal-cases)
  case (1 x)
  show ?case
    by (auto simp: prob-space-measure-of-st-vec measure-def)
      (rule AE-I[where N = {i. st-vec x $ i = 0}], auto)
qed

lemma st-vec-st-vec-of-pmf[simp]:
  st-vec (st-vec-of-pmf x) $ i = pmf x i
  by (simp add: st-vec-of-pmf.rep-eq)

lemma pmf-pmf-of-st-vec[simp]: pmf (pmf-of-st-vec x) i = st-vec x $ i
  by (transfer, auto simp: measure-def)

lemma st-vec-of-pmf-pmf-of-st-vec[simp]: st-vec-of-pmf (pmf-of-st-vec x) = x
proof –
  have st-vec (st-vec-of-pmf (pmf-of-st-vec x)) = st-vec x
    unfolding vec-eq-iff by auto
  thus ?thesis using st-vec-inject by blast
qed

lemma pmf-of-st-vec-inj: (pmf-of-st-vec x = pmf-of-st-vec y) = (x = y)
  by (metis st-vec-of-pmf-pmf-of-st-vec)
end
end
```

4 Stochastic Matrices and Markov Models

We interpret stochastic matrices as Markov chain with discrete time and finite state and prove that the bind-operation on probability mass functions is precisely matrix-vector multiplication. As a consequence, the notion of stationary distribution is equivalent to being an eigenvector with eigenvalue 1.

```
theory Stochastic-Matrix-Markov-Models
imports
  Markov-Models.Classifying-Markov-Chain-States
  Stochastic-Vector-PMF
begin

definition transition-of-st-mat :: 'i st-mat ⇒ 'i :: finite ⇒ 'i pmf where
```

transition-of-st-mat a i = *pmf-as-measure.pmf-of-st-vec* (*transition-vec-of-st-mat a i*)

lemma *st-vec-transition-vec-of-st-mat*[*simp*]:
st-vec (*transition-vec-of-st-mat A a*) \$ *i* = *st-mat A* \$ *i* \$ *a*
by (*transfer*, *auto simp*: *column-def*)

locale *transition-matrix* = *pmf-as-measure* +
fixes *A* :: *'i* :: *finite st-mat*
begin
sublocale *MC-syntax transition-of-st-mat A* **.**

lemma *measure-pmf-of-st-vec*[*simp*]: *measure-pmf* (*pmf-of-st-vec x*) = *measure-of-st-vec x*
by (*rule pmf-as-measure.pmf-of-st-vec.rep-eq*)

lemma *pmf-transition-of-st-mat*[*simp*]: *pmf* (*transition-of-st-mat A a*) *i* = *st-mat A* \$ *i* \$ *a*
unfolding *transition-of-st-mat-def*
by (*transfer*, *auto simp*: *measure-def*)

lemma *bind-is-matrix-vector-mult*: (*bind-pmf x* (*transition-of-st-mat A*)) =
pmf-as-measure.pmf-of-st-vec (*A* ∗*st st-vec-of-pmf x*)
proof (*rule pmf-eqI*, *goal-cases*)
case (*1 i*)
define *X* **where** *X* = *st-vec-of-pmf x*
have *pmf* (*bind-pmf x* (*transition-of-st-mat A*)) *i* =
($\sum$ *a*∈*UNIV*. *pmf x a* $*_R$ *pmf* (*transition-of-st-mat A a*) *i*)
unfolding *pmf-bind* **by** (*subst integral-measure-pmf*[*of UNIV*], *auto*)
also have ... = ($\sum$ *a*∈*UNIV*. *st-mat A* \$ *i* \$ *a* ∗ *st-vec X* \$ *a*)
by (*rule sum.cong*[*OF refl*], *auto simp*: *X-def*)
also have ... = (*st-mat A* ∗*v st-vec X*) \$ *i*
unfolding *matrix-vector-mult-def* **by** *auto*
also have ... = *st-vec* (*A* ∗*st X*) \$ *i* **unfolding** *st-mat-mult-st-vec* **by** *simp*
also have ... = *pmf* (*pmf-of-st-vec* (*A* ∗*st X*)) *i* **by** *simp*
finally show *?case* **by** (*simp add*: *X-def*)
qed

lemmas *stationary-distribution-alt-def* =
stationary-distribution-def[*unfolded bind-is-matrix-vector-mult*]

lemma *stationary-distribution-implies-pmf-of-st-vec*:
assumes *stationary-distribution N*
shows ∃ *x*. *N* = *pmf-of-st-vec x*
proof −
from *assms*[*unfolded stationary-distribution-alt-def*] **show** *?thesis* **by** *auto*
qed

lemma *stationary-distribution-pmf-of-st-vec*:

stationary-distribution (*pmf-of-st-vec x*) = (*A* ∗*st x* = *x*)
unfolding *stationary-distribution-alt-def pmf-of-st-vec-inj* **by** *auto*
end
end

5 Eigenspaces

Using results on Jordan-Normal forms, we prove that the geometric multiplicity of an eigenvalue (i.e., the dimension of the eigenspace) is bounded by the algebraic multiplicity of an eigenvalue (i.e., the multiplicity as root of the characteristic polynomial.). As a consequence we derive that any two eigenvectors of some eigenvalue with multiplicity 1 must be scalar multiples of each other.

theory *Eigenspace*
imports
Jordan-Normal-Form.Jordan-Normal-Form-Uniqueness
Perron-Frobenius.Perron-Frobenius-Aux
begin
hide-const (**open**) *Coset.order*

The dimension of every generalized eigenspace is bounded by the algebraic multiplicity of an eigenvalue. Hence, in particular the geometric multiplicity is smaller than the algebraic multiplicity.

lemma *dim-gen-eigenspace-order-char-poly*: **assumes** *jnf*: *jordan-nf A n-as*
shows *dim-gen-eigenspace A lam k* ≤ *order lam* (*char-poly A*)
unfolding *jordan-nf-order*[*OF jnf*] *dim-gen-eigenspace*[*OF jnf*]
by (*induct n-as*, *auto*)

Every eigenvector is contained in the eigenspace.

lemma *eigenvector-mat-kernel-char-matrix*: **assumes** *A*: *A* ∈ *carrier-mat n n*
and *ev*: *eigenvector A v lam*
shows *v* ∈ *mat-kernel* (*char-matrix A lam*)
using *ev*[*unfolded eigenvector-char-matrix*[*OF A*]] *A*
unfolding *mat-kernel-def* **by** (*auto simp*: *char-matrix-def*)

If the algebraic multiplicity is one, then every two eigenvectors are scalar multiples of each other.

lemma *unique-eigenvector-jnf*: **assumes** *jnf*: *jordan-nf* (*A* :: *'a* :: *field mat*) *n-as*
and *ord*: *order lam* (*char-poly A*) = *1*
and *ev*: *eigenvector A v lam eigenvector A w lam*
shows ∃ *a*. *v* = *a* $\cdot_v$ *w*
proof −
let *?cA* = *char-matrix A lam*
from *similar-matD jnf*[*unfolded jordan-nf-def*] **obtain** *n* **where**
A: *A* ∈ *carrier-mat n n* **by** *auto*
from *dim-gen-eigenspace-order-char-poly*[*OF jnf*, *of lam 1*, *unfolded ord*]
have *dim*: *kernel-dim ?cA* ≤ *1*

unfolding *dim-gen-eigenspace-def* **by** *auto*
from *eigenvector-mat-kernel-char-matrix*[*OF A ev*(*1*)]
have *vk*: $v \in$ *mat-kernel ?cA* **.**
from *eigenvector-mat-kernel-char-matrix*[*OF A ev*(*2*)]
have *wk*: $w \in$ *mat-kernel ?cA* **.**
from *ev*[*unfolded eigenvector-def*] *A* **have**
v: $v \in$ *carrier-vec n* $v \neq 0_v$ *n* **and**
w: $w \in$ *carrier-vec n* $w \neq 0_v$ *n* **by** *auto*
have *cA*: *?cA* $\in$ *carrier-mat n n* **using** *A*
unfolding *char-matrix-def* **by** *auto*
interpret *kernel n n ?cA*
by (*unfold-locales*, *rule cA*)
from *kernel-basis-exists*[*OF A*] **obtain** *B* **where** *B*: *finite B basis B* **by** *auto*
from *this*[*unfolded Ker.basis-def*] **have** *basis*: *mat-kernel ?cA* = *span B* **by** *auto*
{
assume *card B* = *0*
with *B basis* **have** *bas*: *mat-kernel ?cA* = *local.span* {} **by** *auto*
also have ... = $\{0_v\ n\}$ **unfolding** *Ker.span-def* **by** *auto*
finally have *False* **using** *v vk* **by** *auto*
}
with *Ker.dim-basis*[*OF B*] *dim* **have** *card B* = *Suc 0* **by** (*cases card B*, *auto*)
hence $\exists$ *b*. *B* = {*b*} **using** *card-eq-SucD* **by** *blast*
then obtain *b* **where** *Bb*: *B* = {*b*} **by** *blast*
from *B*(*2*)[*unfolded Bb Ker.basis-def*] **have** *bk*: $b \in$ *mat-kernel ?cA* **by** *auto*
hence *b*: $b \in$ *carrier-vec n* **using** *cA mat-kernelD*(*1*) **by** *blast*
from *Bb basis* **have** *mat-kernel ?cA* = *span* {*b*} **by** *auto*
also have ... = *NC.span* {*b*}
by (*rule span-same*, *insert bk*, *auto*)
also have ... $\subseteq$ { $a \cdot_v b$ | *a*. *True*}
proof −
{
fix *x*
assume $x \in$ *NC.span* {*b*}
from *this*[*unfolded NC.span-def*] **obtain** *a A*
where *x*: *x* = *NC.lincomb a A* **and** *A*: $A \subseteq \{b\}$ **by** *auto*
hence *A* = {} $\vee$ *A* = {*b*} **by** *auto*
hence $\exists$ *a*. $x = a \cdot_v b$
proof
assume *A* = {} **thus** *?thesis* **unfolding** *x* **using** *b* **by** (*intro exI*[*of - 0*], *auto*)
next
assume *A* = {*b*} **thus** *?thesis* **unfolding** *x* **using** *b*
by (*intro exI*[*of - a b*], *auto simp*: *NC.lincomb-def*)
qed
}
thus *?thesis* **by** *auto*
qed
finally obtain *vv ww* **where** *vb*: $v = vv \cdot_v b$ **and** *wb*: $w = ww \cdot_v b$ **using** *vk wk* **by** *force+*

from *wb w b* **have** *ww*: $ww \neq 0$ **by** *auto*
from *arg-cong*[*OF wb*, *of* λ *x*. *inverse ww* $\cdot_v$ *x*] *w ww b* **have** *b* = *inverse ww* $\cdot_v$ *w*
by (*auto simp*: *smult-smult-assoc*)
from *vb*[*unfolded this smult-smult-assoc*] **show** *?thesis* **by** *auto*
qed

Getting rid of the JNF-assumption for complex matrices.

lemma *unique-eigenvector-complex*: **assumes** *A*: $A \in$ *carrier-mat n n*
and *ord*: *order lam* (*char-poly A* :: *complex poly*) = *1*
and *ev*: *eigenvector A v lam eigenvector A w lam*
shows ∃ *a*. *v* = *a* $\cdot_v$ *w*
proof −
from *jordan-nf-exists*[*OF A*] *char-poly-factorized*[*OF A*] **obtain** *n-as*
where *jordan-nf A n-as* **by** *auto*
from *unique-eigenvector-jnf*[*OF this ord ev*] **show** *?thesis* **.**
qed

Convert the result to real matrices via homomorphisms.

lemma *unique-eigenvector-real*: **assumes** *A*: $A \in$ *carrier-mat n n*
and *ord*: *order lam* (*char-poly A* :: *real poly*) = *1*
and *ev*: *eigenvector A v lam eigenvector A w lam*
shows ∃ *a*. *v* = *a* $\cdot_v$ *w*
proof −
let *?c* = *complex-of-real*
let *?A* = *map-mat ?c A*
from *A* **have** *cA*: *?A* ∈ *carrier-mat n n* **by** *auto*
have *ord*: *order* (*?c lam*) (*char-poly ?A*) = *1*
unfolding *of-real-hom.char-poly-hom*[*OF A*]
by (*subst map-poly-inj-idom-divide-hom.order-hom*, *unfold-locales*, *rule ord*)
note *evc* = *of-real-hom.eigenvector-hom*[*OF A*]
from *unique-eigenvector-complex*[*OF cA ord evc evc*, *OF ev*] **obtain** *a* :: *complex*

where *id*: *map-vec ?c v* = *a* $\cdot_v$ *map-vec ?c w* **by** *auto*

from *ev*[*unfolded eigenvector-def*] *A* **have** *carr*: *v* ∈ *carrier-vec n w* ∈ *carrier-vec n*
$v \neq 0_v$ *n* **by** *auto*
then obtain *i* **where** *i*: *i* < *n v* $ *i* ≠ *0* **unfolding** *Matrix.vec-eq-iff* **by** *auto*
from *arg-cong*[*OF id*, *of* λ *x*. *x* $ *i*] *carr i*
have *?c* (*v* $ *i*) = *a* ∗ *?c* (*w* $ *i*)
by *auto*
with *i*(*2*) **have** *a* ∈ *Reals*
by (*metis Reals-cnj-iff complex-cnj-complex-of-real complex-cnj-mult mult-cancel-right mult-eq-0-iff of-real-hom.hom-zero of-real-hom.injectivity*)
then obtain *b* **where** *a*: *a* = *?c b* **by** (*rule Reals-cases*)
from *id*[*unfolded a*] **have** *map-vec ?c v* = *map-vec ?c* (*b* $\cdot_v$ *w*) **by** *auto*
hence *v* = *b* $\cdot_v$ *w* **by** (*rule of-real-hom.vec-hom-inj*)
thus *?thesis* **by** *auto*

qed

Finally, the statement converted to HMA-world.

lemma *unique-eigen-vector-real*: **assumes** *ord*: *order lam* (*charpoly A* :: *real poly*) = *1*
 and *ev*: *eigen-vector A v lam eigen-vector A w lam*
shows ∃ *a*. *v* = *a* ∗*s w* **using** *assms*
proof (*transfer*, *goal-cases*)
 case (*1 lam A v w*)
 show *?case*
  by (*rule unique-eigenvector-real*[*OF 1*(*1*−*2*,*4*,*6*)])
qed

end

6 Stochastic Matrices and the Perron–Frobenius Theorem

Since a stationary distribution corresponds to a non-negative real eigenvector of the stochastic matrix, we can apply the Perron–Frobenius theorem. In this way we easily derive that every stochastic matrix has a stationary distribution, and moreover that this distribution is unique, if the matrix is irreducible, i.e., if the graph of the matrix is strongly connected.

theory *Stochastic-Matrix-Perron-Frobenius*
imports
 Perron-Frobenius.Perron-Frobenius-Irreducible
 Stochastic-Matrix-Markov-Models
 Eigenspace
begin

hide-const (**open**) *Coset.order*

lemma *pf-nonneg-mat-st-mat*: *pf-nonneg-mat* (*st-mat A*)
 by (*unfold-locales*, *auto simp*: *non-neg-mat-st-mat*)

lemma *stoch-non-neg-vec-norm1*: **assumes** *stoch-vec* (*v* :: *real* ^ *'n*) *non-neg-vec v*
 shows *norm1 v* = *1*
 unfolding *assms*(*1*)[*unfolded stoch-vec-def*, *symmetric*] *norm1-def*
 by (*rule sum.cong*, *insert assms*(*2*)[*unfolded non-neg-vec-def*], *auto*)

lemma *stationary-distribution-exists*: ∃ *v*. *A* ∗*st v* = *v*
proof −
 let *?A* = *st-mat A*
 let *?c* = *complex-of-real*
 let *?B* = χ *i j*. *?c* (*?A* $ *i* $ *j*)
 have *real-non-neg-mat ?B* **using** *non-neg-mat-st-mat*[*of A*]

unfolding *real-non-neg-mat-def elements-mat-h-def non-neg-mat-def*
by *auto*
from *Perron-Frobenius.perron-frobenius-both*[*OF this*] **obtain** *v a* **where**
ev: *eigen-vector ?B v* (*?c a*) **and** *nn*: *real-non-neg-vec v*
and *a*: *a* = *HMA-Connect.spectral-radius ?B* **by** *auto*
from *spectral-radius-ev*[*of ?B*, *folded a*] **have** *a0*: *a* ≥ *0* **by** *auto*
define *w* **where** *w* = (χ *i*. *Re* (*v* $ *i*))
from *nn* **have** *vw*: *v* = (χ *i*. *?c* (*w* $ *i*)) **unfolding** *real-non-neg-vec-def w-def*
by (*auto simp*: *vec-elements-h-def*)
from *ev*[*unfolded eigen-vector-def*] **have** *v0*: *v* ≠ *0* **and** *ev*: *?B* ∗*v v* = *?c a* ∗*s*
v **by** *auto*
from *v0* **have** *w0*: *w* ≠ *0* **unfolding** *vw* **by** (*auto simp*: *Finite-Cartesian-Product.vec-eq-iff*)
{
fix *i*
from *ev* **have** *Re* ((*?B* ∗*v v*) $ *i*) = *Re* ((*?c a* ∗*s v*) $ *i*) **by** *simp*
also have *Re* ((*?c a* ∗*s v*) $ *i*) = (*a* ∗*s w*) $ *i* **unfolding** *vw* **by** *simp*
also have *Re* ((*?B* ∗*v v*) $ *i*) = (*?A* ∗*v w*) $ *i* **unfolding** *vw*
by (*simp add*: *matrix-vector-mult-def*)
also note *calculation*
}
hence *ev*: *?A* ∗*v w* = *a* ∗*s w* **by** (*auto simp*: *Finite-Cartesian-Product.vec-eq-iff*)
from *nn* **have** *nn*: *non-neg-vec w*
unfolding *vw* **by** (*auto simp*: *real-non-neg-vec-def non-neg-vec-def vec-elements-h-def*)

let *?n* = *norm1 w*
from *w0* **have** *n0*: *?n* ≠ *0* **by** *auto*
hence *n-pos*: *?n* > *0* **using** *norm1-ge-0*[*of w*] **by** *linarith*
define *u* **where** *u* = *inverse ?n* ∗*s w*
have *nn*: *non-neg-vec u* **using** *nn n-pos* **unfolding** *u-def non-neg-vec-def* **by**
auto
have *nu*: *norm1 u* = *1* **unfolding** *u-def scalar-mult-eq-scaleR norm1-scaleR*
using *n-pos*
by (*auto simp*: *field-simps*)
have *1*: *stoch-vec u* **unfolding** *stoch-vec-def nu*[*symmetric*] *norm1-def*
by (*rule sum.cong*, *insert nn*[*unfolded non-neg-vec-def*], *auto*)
from *arg-cong*[*OF ev*, *of* λ *x*. *inverse ?n* ∗*s x*]
have *ev*: *?A* ∗*v u* = *a* ∗*s u* **unfolding** *u-def*
by (*auto simp*: *ac-simps vector-smult-distrib matrix-vect-scaleR*)
from *right-stoch-mat-mult-stoch-vec*[*OF right-stoch-mat-st-mat*[*of A*] *1*, *unfolded*
ev]
have *st*: *stoch-vec* (*a* ∗*s u*) **.**
from *non-neg-mat-mult-non-neg-vec*[*OF non-neg-mat-st-mat*[*of A*] *nn*, *unfolded*
ev]
have *nn′*: *non-neg-vec* (*a* ∗*s u*) **.**
from *stoch-non-neg-vec-norm1*[*OF st nn′*, *unfolded scalar-mult-eq-scaleR norm1-scaleR*
nu] *a0*
have *a* = *1* **by** *auto*
with *ev st* **have** *ev*: *?A* ∗*v u* = *u* **and** *st*: *stoch-vec u* **by** *auto*
show *?thesis* **using** *ev st nn*

by (*intro exI*[*of* - *to-st-vec u*], *transfer*, *auto*)
qed

lemma *stationary-distribution-unique*:
assumes *fixed-mat.irreducible* (*st-mat A*)
shows $\exists!$ *v*. *A* $*$*st v* = *v*
proof −
from *stationary-distribution-exists* **obtain** *v* **where** *ev*: *A* $*$*st v* = *v* **by** *auto*
show *?thesis*
proof (*intro ex1I*, *rule ev*)
fix *w*
assume *A* $*$*st w* = *w*
thus *w* = *v* **using** *ev assms*
proof (*transfer*, *goal-cases*)
case (*1 A w v*)
interpret *perron-frobenius A*
by (*unfold-locales*, *insert 1*, *auto*)
from *1* **have** $*$: *eigen-vector A v 1 le-vec 0 v eigen-vector A w 1*
by (*auto simp*: *eigen-vector-def stoch-vec-def non-neg-vec-def*)
from *nonnegative-eigenvector-has-ev-sr*[*OF* $*$(*1*−*2*)] **have** *sr1*: *sr* = *1* **by** *auto*
from *multiplicity-sr-1*[*unfolded sr1*] **have** *order 1* (*charpoly A*) = *1* **.**
from *unique-eigen-vector-real*[*OF this* $*$(*1*,*3*)] **obtain** *a* **where**
vw: *v* = *a* $*$*s w* **by** *auto*
from *1*(*2*,*4*)[*unfolded stoch-vec-def*] **have** *sum* (($*h*) *v*) *UNIV* = *sum* (($*h*) *w*) *UNIV* **by** *auto*
also have *sum* (($*h*) *v*) *UNIV* = *a* $*$ *sum* (($*h*) *w*) *UNIV* **unfolding** *vw*
by (*auto simp*: *sum-distrib-left*)
finally have *a* = *1* **using** *1*(*2*)[*unfolded stoch-vec-def*] **by** *auto*
with *vw* **show** *v* = *w* **by** *auto*
qed
qed
qed

Let us now convert the stationary distribution results from matrices to Markov chains.

context *transition-matrix*
begin

lemma *stationary-distribution-exists*:
$\exists$ *x*. *stationary-distribution* (*pmf-of-st-vec x*)
proof −
from *stationary-distribution-exists* **obtain** *x* **where** *ev*: *A* $*$*st x* = *x* **by** *auto*
show *?thesis*
by (*intro exI*[*of* - *x*], *unfold stationary-distribution-pmf-of-st-vec*,
simp add: *ev*)
qed

lemma *stationary-distribution-unique*: **assumes** *fixed-mat.irreducible* (*st-mat A*)

shows ∃! *N*. *stationary-distribution N*
proof −
from *stationary-distribution-exists* **obtain** *x* **where**
st: *stationary-distribution* (*pmf-of-st-vec x*) **by** *blast*
show *?thesis*
proof (*rule ex1I*, *rule st*)
fix *N*
assume *st'*: *stationary-distribution N*
from *stationary-distribution-implies-pmf-of-st-vec*[*OF this*] **obtain** *y* **where**
N: *N* = *pmf-of-st-vec y* **by** *auto*
from *st'*[*unfolded N*] *st*
have *A* ∗*st x* = *x A* ∗*st y* = *y* **unfolding** *stationary-distribution-pmf-of-st-vec*
by *auto*
from *stationary-distribution-unique*[*OF assms*] *this* **have** *x* = *y* **by** *auto*
with *N* **show** *N* = *pmf-of-st-vec x* **by** *auto*
qed
qed
end
end

References

[1] J. Divasón, O. Kunar, R. Thiemann, and A. Yamada. Perron-frobenius theorem for spectral radius analysis. *Archive of Formal Proofs*, May 2016. http://isa-afp.org/entries/Perron_Frobenius.html, Formal proof development.

[2] J. Hölzl and T. Nipkow. Markov models. *Archive of Formal Proofs*, Jan. 2012. http://isa-afp.org/entries/Markov_Models.html, Formal proof development.