

Perron-Frobenius Theorem for Spectral Radius Analysis*

Jose Divasón, Ondrej Kunar, René Thiemann and Akihisa Yamada

August 7, 2022

Abstract

The spectral radius of a matrix A is the maximum norm of all eigenvalues of A . In previous work we already formalized that for a complex matrix A , the values in A^n grow polynomially in n if and only if the spectral radius is at most one. One problem with the above characterization is the determination of all *complex* eigenvalues. In case A contains only non-negative real values, a simplification is possible with the help of the Perron-Frobenius theorem, which tells us that it suffices to consider only the *real* eigenvalues of A , i.e., applying Sturm's method can decide the polynomial growth of A^n .

We formalize the Perron-Frobenius theorem based on a proof via Brouwer's fixpoint theorem, which is available in the HOL multivariate analysis (HMA) library. Since the results on the spectral radius is based on matrices in the Jordan normal form (JNF) library, we further develop a connection which allows us to easily transfer theorems between HMA and JNF. With this connection we derive the combined result: if A is a non-negative real matrix, and no real eigenvalue of A is strictly larger than one, then A^n is polynomially bounded in n .

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 2 |
| 2 | Elimination of CARD('n) | 3 |
| 3 | Connecting HMA-matrices with JNF-matrices | 4 |
| 3.1 | Bijections between index types of HMA and natural numbers | 4 |
| 3.2 | Transfer rules to convert theorems from JNF to HMA and vice-versa. | 6 |

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

| | | |
|-----------|---|-----------|
| 4 | Perron-Frobenius Theorem | 16 |
| 4.1 | Auxiliary Notions | 16 |
| 4.2 | Perron-Frobenius theorem via Brouwer’s fixpoint theorem. . . | 21 |
| 5 | Roots of Unity | 23 |
| 5.1 | The Perron Frobenius Theorem for Irreducible Matrices . . . | 27 |
| 5.2 | Handling Non-Irreducible Matrices as Well | 36 |
| 6 | Combining Spectral Radius Theory with Perron Frobenius theorem | 41 |
| 7 | The Jordan Blocks of the Spectral Radius are Largest | 44 |
| 8 | Homomorphisms of Gauss-Jordan Elimination, Kernel and More | 48 |
| 9 | Combining Spectral Radius Theory with Perron Frobenius theorem | 50 |
| 10 | An efficient algorithm to compute the growth rate of A^n. | 51 |

1 Introduction

The spectral radius of a matrix A over \mathbb{R} or \mathbb{C} is defined as

$$\rho(A) = \max \{|x| \mid \chi_A(x) = 0, x \in \mathbb{C}\}$$

where χ_A is the characteristic polynomial of A . It is a central notion related to the growth rate of matrix powers. A matrix A has polynomial growth, i.e., all values of A^n can be bounded polynomially in n , if and only if $\rho(A) \leq 1$. It is quite easy to see that $\rho(A) \leq 1$ is a necessary criterion,¹ but it is more complicated to argue about sufficiency. In previous work we formalized this statement via Jordan normal forms [4].

Theorem 1 (in JNF). *The values in A^n are polynomially bounded in n if $\rho(A) \leq 1$.*

In order to perform the proof via Jordan normal forms, we did not use the HMA library from the distribution to represent matrices. The reason is that already the definition of a Jordan normal form is naturally expressed via block-matrices, and arbitrary block-matrices are hard to express in HMA, if at all.

¹Let λ and v be some eigenvalue and eigenvector pair such that $|\lambda| > 1$. Then $|A^n v| = |\lambda^n v| = |\lambda|^n |v|$ grows exponentially in n , where $|w|$ denotes the component-wise application of $|\cdot|$ to vector elements of w .

The problem in applying Theorem 1 in concrete examples is the determination of all complex roots of the polynomial χ_A . For instance, one can utilize complex algebraic numbers for this purpose, which however are computationally expensive. To avoid this problem, in this work we formalize the Perron Frobenius theorem. It states that for non-negative real-valued matrices, $\rho(A)$ is an eigenvalue of A .

Theorem 2 (in HMA). *If $A \in \mathbb{R}_{\geq 0}^{k \times k}$, then $\chi_A(\rho(A)) = 0$.*

We decided to perform the formalization based on the HMA library, since there is a short proof of Theorem 2 via Brouwer’s fixpoint theorem [2, Section 5.2]. The latter is a well-known but complex theorem that is available in HMA, but not in the JNF library.

Eventually we want to combine both theorems to obtain:

Corollary 1. *If $A \in \mathbb{R}_{\geq 0}^{k \times k}$, then the values in A^n are polynomially bounded in n if χ_A has no real roots in the interval $(1, \infty)$.*

This criterion is computationally far less expensive – one invocation of Sturm’s method on χ_A suffices. Unfortunately, we cannot immediately combine both theorems. We first have to bridge the gap between the HMA-world and the JNF-world. To this end, we develop a setup for the transfer-tool which admits to translate theorems from JNF into HMA. Moreover, using a recent extension for local type definitions within proofs [1], we also provide a translation from HMA into JNF.

With the help of these translations, we prove Corollary 1 and make it available in both HMA and JNF. (In the formalization the corollary looks a bit more complicated as it also contains an estimation of the the degree of the polynomial growth.)

2 Elimination of CARD('n)

In the following theory we provide a method which modifies theorems of the form $P[\text{CARD}(n)]$ into $n! = 0 \implies P[n]$, so that they can more easily be applied.

Known issues: there might be problems with nested meta-implications and meta-quantification.

theory *Cancel-Card-Constraint*

imports

HOL-Types-To-Sets.Types-To-Sets

HOL-Library.Cardinality

begin

lemma *n-zero-nonempty*: $n \neq 0 \implies \{0 ..< n :: nat\} \neq \{\}$ *<proof>*

```

lemma type-impl-card-n: assumes  $\exists (Rep :: 'a \Rightarrow nat)$  Abs. type-definition Rep
Abs  $\{0 ..< n :: nat\}$ 
  shows class.finite (TYPE('a))  $\wedge$  CARD('a) = n
  <proof>

  <ML>

```

end

3 Connecting HMA-matrices with JNF-matrices

The following theories provide a connection between the type-based representation of vectors and matrices in HOL multivariate-analysis (HMA) with the set-based representation of vectors and matrices with integer indices in the Jordan-normal-form (JNF) development.

3.1 Bijections between index types of HMA and natural numbers

At the core of HMA-connect, there has to be a translation between indices of vectors and matrices, which are via index-types on the one hand, and natural numbers on the other hand.

We some unspecified bijection in our application, and not the conversions to-nat and from-nat in theory Rank-Nullity-Theorem/Mod-Type, since our definitions below do not enforce any further type constraints.

```

theory Bij-Nat
imports
  HOL-Library.Cardinality
  HOL-Library.Numeral-Type
begin

```

```

lemma finite-set-to-list:  $\exists xs :: 'a :: finite$  list. distinct xs  $\wedge$  set xs = Y
  <proof>

```

```

definition univ-list :: 'a :: finite list where
  univ-list = (SOME xs. distinct xs  $\wedge$  set xs = UNIV)

```

```

lemma univ-list: distinct (univ-list :: 'a list) set univ-list = (UNIV :: 'a :: finite
  set)
  <proof>

```

```

definition to-nat :: 'a :: finite  $\Rightarrow$  nat where

```

$to\text{-}nat\ a = (SOME\ i.\ univ\text{-}list\ !\ i = a \wedge i < length\ (univ\text{-}list\ ::\ 'a\ list))$

definition $from\text{-}nat\ ::\ nat \Rightarrow 'a\ ::\ finite$ **where**
 $from\text{-}nat\ i = univ\text{-}list\ !\ i$

lemma $length\text{-}univ\text{-}list\text{-}card$: $length\ (univ\text{-}list\ ::\ 'a\ ::\ finite\ list) = CARD('a)$
 $\langle proof \rangle$

lemma $to\text{-}nat\text{-}ex$: $\exists!\ i.\ univ\text{-}list\ !\ i = (a\ ::\ 'a\ ::\ finite) \wedge i < length\ (univ\text{-}list\ ::\ 'a\ list)$
 $\langle proof \rangle$

lemma $to\text{-}nat\text{-}less\text{-}card$: $to\text{-}nat\ (a\ ::\ 'a\ ::\ finite) < CARD('a)$
 $\langle proof \rangle$

lemma $to\text{-}nat\text{-}from\text{-}nat\text{-}id$:
assumes $i: i < CARD('a\ ::\ finite)$
shows $to\text{-}nat\ (from\text{-}nat\ i\ ::\ 'a) = i$
 $\langle proof \rangle$

lemma $from\text{-}nat\text{-}inj$: **assumes** $i: i < CARD('a\ ::\ finite)$
and $j: j < CARD('a\ ::\ finite)$
and $id: (from\text{-}nat\ i\ ::\ 'a) = from\text{-}nat\ j$
shows $i = j$
 $\langle proof \rangle$

lemma $from\text{-}nat\text{-}to\text{-}nat\text{-}id[simp]$:
 $(from\text{-}nat\ (to\text{-}nat\ a)) = (a\ ::\ 'a\ ::\ finite)$
 $\langle proof \rangle$

lemma $to\text{-}nat\text{-}inj[simp]$: **assumes** $to\text{-}nat\ a = to\text{-}nat\ b$
shows $a = b$
 $\langle proof \rangle$

lemma $range\text{-}to\text{-}nat$: $range\ (to\text{-}nat\ ::\ 'a\ ::\ finite \Rightarrow nat) = \{0 ..< CARD('a)\}$ (is
 $?l = ?r$)
 $\langle proof \rangle$

lemma $inj\text{-}to\text{-}nat$: $inj\ to\text{-}nat\ \langle proof \rangle$

lemma $bij\text{-}to\text{-}nat$: $bij\text{-}betw\ to\text{-}nat\ (UNIV\ ::\ 'a\ ::\ finite\ set)\ \{0 ..< CARD('a)\}$
 $\langle proof \rangle$

lemma $numeral\text{-}nat$: $(numeral\ m1\ ::\ nat) * numeral\ n1 \equiv numeral\ (m1 * n1)$
 $(numeral\ m1\ ::\ nat) + numeral\ n1 \equiv numeral\ (m1 + n1)\ \langle proof \rangle$

lemmas $card\text{-}num\text{-}simps =$
 $card\text{-}num1\ card\text{-}bit0\ card\text{-}bit1$

```

mult-num-simps
add-num-simps
eq-num-simps
mult-Suc-right mult-0-right One-nat-def add.right-neutral
numeral-nat Suc-numeral

```

end

3.2 Transfer rules to convert theorems from JNF to HMA and vice-versa.

```

theory HMA-Connect
imports
  Jordan-Normal-Form.Spectral-Radius
  HOL-Analysis.Determinants
  HOL-Analysis.Cartesian-Euclidean-Space
  Bij-Nat
  Cancel-Card-Constraint
  HOL-Eisbach.Eisbach
begin

```

Prefer certain constants and lemmas without prefix.

```

hide-const (open) Matrix.mat
hide-const (open) Matrix.row
hide-const (open) Determinant.det

```

```

lemmas mat-def = Finite-Cartesian-Product.mat-def
lemmas det-def = Determinants.det-def
lemmas row-def = Finite-Cartesian-Product.row-def

```

```

notation vec-index (infixl $v 90)
notation vec-nth (infixl $h 90)

```

Forget that *'a mat*, *'a Matrix.vec*, and *'a poly* have been defined via lifting

```

lifting-forget vec.lifting
lifting-forget mat.lifting

```

```

lifting-forget poly.lifting

```

Some notions which we did not find in the HMA-world.

```

definition eigen-vector :: 'a::comm-ring-1 ^ 'n ^ 'n => 'a ^ 'n => 'a => bool where
  eigen-vector A v ev = (v ≠ 0 ∧ A *v v = ev *s v)

```

```

definition eigen-value :: 'a::comm-ring-1 ^ 'n ^ 'n => 'a => bool where
  eigen-value A k = (∃ v. eigen-vector A v k)

```

```

definition similar-matrix-wit
  :: 'a::semiring-1 ^ 'n ^ 'n => 'a ^ 'n ^ 'n => 'a ^ 'n ^ 'n => 'a ^ 'n ^ 'n => bool
where

```

similar-matrix-wit $A B P Q = (P ** Q = \text{mat } 1 \wedge Q ** P = \text{mat } 1 \wedge A = P ** B ** Q)$

definition *similar-matrix*

$:: 'a :: \text{semiring-1} \wedge 'n \wedge 'n \Rightarrow 'a \wedge 'n \wedge 'n \Rightarrow \text{bool}$ **where**
similar-matrix $A B = (\exists P Q. \text{similar-matrix-wit } A B P Q)$

definition *spectral-radius* $:: \text{complex} \wedge 'n \wedge 'n \Rightarrow \text{real}$ **where**

spectral-radius $A = \text{Max} \{ \text{norm } ev \mid v \text{ ev. eigen-vector } A v \text{ ev} \}$

definition *Spectrum* $:: 'a :: \text{field} \wedge 'n \wedge 'n \Rightarrow 'a \text{ set}$ **where**

Spectrum $A = \text{Collect} (\text{eigen-value } A)$

definition *vec-elements-h* $:: 'a \wedge 'n \Rightarrow 'a \text{ set}$ **where**

vec-elements-h $v = \text{range} (\text{vec-nth } v)$

lemma *vec-elements-h-def'*: $\text{vec-elements-h } v = \{v \$h i \mid i. \text{True}\}$

<proof>

definition *elements-mat-h* $:: 'a \wedge 'nc \wedge 'nr \Rightarrow 'a \text{ set}$ **where**

elements-mat-h $A = \text{range} (\lambda (i,j). A \$h i \$h j)$

lemma *elements-mat-h-def'*: $\text{elements-mat-h } A = \{A \$h i \$h j \mid i j. \text{True}\}$

<proof>

definition *map-vector* $:: ('a \Rightarrow 'b) \Rightarrow 'a \wedge 'n \Rightarrow 'b \wedge 'n$ **where**

map-vector $f v \equiv \chi i. f (v \$h i)$

definition *map-matrix* $:: ('a \Rightarrow 'b) \Rightarrow 'a \wedge 'n \wedge 'm \Rightarrow 'b \wedge 'n \wedge 'm$ **where**

map-matrix $f A \equiv \chi i. \text{map-vector } f (A \$h i)$

definition *normbound* $:: 'a :: \text{real-normed-field} \wedge 'nc \wedge 'nr \Rightarrow \text{real} \Rightarrow \text{bool}$ **where**

normbound $A b \equiv \forall x \in \text{elements-mat-h } A. \text{norm } x \leq b$

lemma *spectral-radius-ev-def'*: $\text{spectral-radius } A = \text{Max} (\text{norm } '(\text{Collect} (\text{eigen-value } A)))$

<proof>

lemma *elements-mat*: $\text{elements-mat } A = \{A \$\$ (i,j) \mid i j. i < \text{dim-row } A \wedge j < \text{dim-col } A\}$

<proof>

definition *vec-elements* $:: 'a \text{ Matrix.vec} \Rightarrow 'a \text{ set}$

where *vec-elements* $v = \text{set } [v \$ i. i <- [0 ..< \text{dim-vec } v]]$

lemma *vec-elements*: $\text{vec-elements } v = \{v \$ i \mid i. i < \text{dim-vec } v\}$

<proof>

context includes *vec.lifting*

begin

end

definition *from-hma_v* :: 'a ^ 'n ⇒ 'a Matrix.vec **where**

from-hma_v v = Matrix.vec CARD('n) (λ i. v \$h from-nat i)

definition *from-hma_m* :: 'a ^ 'nc ^ 'nr ⇒ 'a Matrix.mat **where**

from-hma_m a = Matrix.mat CARD('nr) CARD('nc) (λ (i,j). a \$h from-nat i \$h from-nat j)

definition *to-hma_v* :: 'a Matrix.vec ⇒ 'a ^ 'n **where**

to-hma_v v = (χ i. v \$v to-nat i)

definition *to-hma_m* :: 'a Matrix.mat ⇒ 'a ^ 'nc ^ 'nr **where**

to-hma_m a = (χ i j. a \$\$ (to-nat i, to-nat j))

declare *vec-lambda-eta[simp]*

lemma *to-hma-from-hma_v[simp]*: *to-hma_v* (*from-hma_v* v) = v

⟨*proof*⟩

lemma *to-hma-from-hma_m[simp]*: *to-hma_m* (*from-hma_m* v) = v

⟨*proof*⟩

lemma *from-hma-to-hma_v[simp]*:

v ∈ carrier-vec (CARD('n)) ⇒ *from-hma_v* (*to-hma_v* v :: 'a ^ 'n) = v

⟨*proof*⟩

lemma *from-hma-to-hma_m[simp]*:

A ∈ carrier-mat (CARD('nr)) (CARD('nc)) ⇒ *from-hma_m* (*to-hma_m* A :: 'a ^ 'nc ^ 'nr) = A

⟨*proof*⟩

lemma *from-hma_v-inj[simp]*: *from-hma_v* x = *from-hma_v* y ⇔ x = y

⟨*proof*⟩

lemma *from-hma_m-inj[simp]*: *from-hma_m* x = *from-hma_m* y ⇔ x = y

⟨*proof*⟩

definition *HMA-V* :: 'a Matrix.vec ⇒ 'a ^ 'n ⇒ bool **where**

HMA-V = (λ v w. v = *from-hma_v* w)

definition *HMA-M* :: 'a Matrix.mat ⇒ 'a ^ 'nc ^ 'nr ⇒ bool **where**

HMA-M = (λ a b. a = *from-hma_m* b)

definition *HMA-I* :: nat ⇒ 'n :: finite ⇒ bool **where**

HMA-I = (λ i a. i = to-nat a)

context includes *lifting-syntax*

begin

lemma *Domainp-HMA-V* [*transfer-domain-rule*]:

$Domainp (HMA-V :: 'a Matrix.vec \Rightarrow 'a \wedge 'n \Rightarrow bool) = (\lambda v. v \in carrier-vec (CARD('n)))$
<proof>

lemma *Domainp-HMA-M* [*transfer-domain-rule*]:

$Domainp (HMA-M :: 'a Matrix.mat \Rightarrow 'a \wedge 'nc \wedge 'nr \Rightarrow bool)$
 $= (\lambda A. A \in carrier-mat CARD('nr) CARD('nc))$
<proof>

lemma *Domainp-HMA-I* [*transfer-domain-rule*]:

$Domainp (HMA-I :: nat \Rightarrow 'n :: finite \Rightarrow bool) = (\lambda i. i < CARD('n))$ (**is** *?! = ?r*)
<proof>

lemma *bi-unique-HMA-V* [*transfer-rule*]: *bi-unique HMA-V left-unique HMA-V right-unique HMA-V*

<proof>

lemma *bi-unique-HMA-M* [*transfer-rule*]: *bi-unique HMA-M left-unique HMA-M right-unique HMA-M*

<proof>

lemma *bi-unique-HMA-I* [*transfer-rule*]: *bi-unique HMA-I left-unique HMA-I right-unique HMA-I*

<proof>

lemma *right-total-HMA-V* [*transfer-rule*]: *right-total HMA-V*

<proof>

lemma *right-total-HMA-M* [*transfer-rule*]: *right-total HMA-M*

<proof>

lemma *right-total-HMA-I* [*transfer-rule*]: *right-total HMA-I*

<proof>

lemma *HMA-V-index* [*transfer-rule*]: $(HMA-V \implies HMA-I \implies (=))$ (*\$v*)

(*\$h*)

<proof>

We introduce the index function to have pointwise access to HMA-matrices by a constant. Otherwise, the transfer rule with $\lambda A i. (\$h) (A \$h i)$ instead of index is not applicable.

definition *index-hma* $A i j \equiv A \$h i \$h j$

lemma *HMA-M-index* [transfer-rule]:

(*HMA-M* \implies *HMA-I* \implies *HMA-I* \implies (=)) ($\lambda A i j. A \$\$ (i,j)$)
index-hma
(proof)

lemma *HMA-V-0* [transfer-rule]: *HMA-V* (0_v *CARD*('n)) ($0 :: 'a :: \text{zero} \hat{\ } ^n$)
(proof)

lemma *HMA-M-0* [transfer-rule]:

HMA-M (0_m *CARD*('nr) *CARD*('nc)) ($0 :: 'a :: \text{zero} \hat{\ } ^{nc} \hat{\ } ^{nr}$)
(proof)

lemma *HMA-M-1* [transfer-rule]:

HMA-M (1_m (*CARD*('n))) ($\text{mat } 1 :: 'a :: \{\text{zero,one}\} \hat{\ } ^n \hat{\ } ^n$)
(proof)

lemma *from-hma_v-add*: *from-hma_v* $v + \text{from-hma}_v w = \text{from-hma}_v (v + w)$
(proof)

lemma *HMA-V-add* [transfer-rule]: (*HMA-V* \implies *HMA-V* \implies *HMA-V*)
(+) (+)
(proof)

lemma *from-hma_v-diff*: *from-hma_v* $v - \text{from-hma}_v w = \text{from-hma}_v (v - w)$
(proof)

lemma *HMA-V-diff* [transfer-rule]: (*HMA-V* \implies *HMA-V* \implies *HMA-V*)
(-) (-)
(proof)

lemma *from-hma_m-add*: *from-hma_m* $a + \text{from-hma}_m b = \text{from-hma}_m (a + b)$
(proof)

lemma *HMA-M-add* [transfer-rule]: (*HMA-M* \implies *HMA-M* \implies *HMA-M*)
(+) (+)
(proof)

lemma *from-hma_m-diff*: *from-hma_m* $a - \text{from-hma}_m b = \text{from-hma}_m (a - b)$
(proof)

lemma *HMA-M-diff* [transfer-rule]: (*HMA-M* \implies *HMA-M* \implies *HMA-M*)
(-) (-)
(proof)

lemma *scalar-product*: **fixes** $v :: 'a :: \text{semiring-1} \hat{\ } ^n$

shows *scalar-prod* (*from-hma_v* v) (*from-hma_v* w) = *scalar-product* $v w$
(proof)

lemma [*simp*]:

$from-hma_m (y :: 'a \hat{~} 'nc \hat{~} 'nr) \in carrier_mat (CARD('nr)) (CARD('nc))$
 $dim_row (from-hma_m (y :: 'a \hat{~} 'nc \hat{~} 'nr)) = CARD('nr)$
 $dim_col (from-hma_m (y :: 'a \hat{~} 'nc \hat{~} 'nr)) = CARD('nc)$
 <proof>

lemma [simp]:
 $from-hma_v (y :: 'a \hat{~} 'n) \in carrier_vec (CARD('n))$
 $dim_vec (from-hma_v (y :: 'a \hat{~} 'n)) = CARD('n)$
 <proof>

declare *rel-funI* [intro!]

lemma *HMA-scalar-prod* [transfer-rule]:
 $(HMA-V \implies HMA-V \implies (=)) \text{ scalar-prod scalar-product}$
 <proof>

lemma *HMA-row* [transfer-rule]: $(HMA-I \implies HMA-M \implies HMA-V) (\lambda i$
a. Matrix.row a i) row
 <proof>

lemma *HMA-col* [transfer-rule]: $(HMA-I \implies HMA-M \implies HMA-V) (\lambda i$
a. col a i) column
 <proof>

definition *mk-mat* :: $('i \Rightarrow 'j \Rightarrow 'c) \Rightarrow 'c \hat{~} 'j \hat{~} 'i$ **where**
 $mk_mat\ f = (\chi\ i\ j.\ f\ i\ j)$

definition *mk-vec* :: $('i \Rightarrow 'c) \Rightarrow 'c \hat{~} 'i$ **where**
 $mk_vec\ f = (\chi\ i.\ f\ i)$

lemma *HMA-M-mk-mat*[transfer-rule]: $((HMA-I \implies HMA-I \implies (=)) \implies$
HMA-M)
 $(\lambda f.\ Matrix.mat (CARD('nr)) (CARD('nc)) (\lambda (i,j).\ f\ i\ j))$
 $(mk_mat :: (('nr \Rightarrow 'nc \Rightarrow 'a) \Rightarrow 'a \hat{~} 'nc \hat{~} 'nr))$
 <proof>

lemma *HMA-M-mk-vec*[transfer-rule]: $((HMA-I \implies (=)) \implies HMA-V)$
 $(\lambda f.\ Matrix.vec (CARD('n)) (\lambda i.\ f\ i))$
 $(mk_vec :: (('n \Rightarrow 'a) \Rightarrow 'a \hat{~} 'n))$
 <proof>

lemma *mat-mult-scalar*: $A ** B = mk_mat (\lambda i\ j.\ scalar_product (row\ i\ A) (column$
j B))
 <proof>

lemma *mult-mat-vec-scalar*: $A * v\ v = mk_vec (\lambda i.\ scalar_product (row\ i\ A) v)$
 <proof>

lemma *dim-row-transfer-rule*:

$HMA-M A (A' :: 'a \wedge 'nc \wedge 'nr) \implies (=) (dim-row A) (CARD('nr))$
 $\langle proof \rangle$

lemma *dim-col-transfer-rule*:

$HMA-M A (A' :: 'a \wedge 'nc \wedge 'nr) \implies (=) (dim-col A) (CARD('nc))$
 $\langle proof \rangle$

lemma *HMA-M-mult [transfer-rule]*: ($HMA-M \implies HMA-M \implies HMA-M$)

$((*) (**))$
 $\langle proof \rangle$

lemma *HMA-V-smult [transfer-rule]*: ($(=) \implies HMA-V \implies HMA-V$) (\cdot_v)

$((*s))$
 $\langle proof \rangle$

lemma *HMA-M-mult-vec [transfer-rule]*: ($HMA-M \implies HMA-V \implies HMA-V$)

$((*_v)) ((*_v))$
 $\langle proof \rangle$

lemma *HMA-det [transfer-rule]*: ($HMA-M \implies (=)$) *Determinant.det*

$(det :: 'a :: comm-ring-1 \wedge 'n \wedge 'n \Rightarrow 'a)$
 $\langle proof \rangle$

lemma *HMA-mat[transfer-rule]*: ($(=) \implies HMA-M$) ($\lambda k. k \cdot_m 1_m CARD('n)$)

$(Finite-Cartesian-Product.mat :: 'a::semiring-1 \Rightarrow 'a \wedge 'n \wedge 'n)$
 $\langle proof \rangle$

lemma *HMA-mat-minus[transfer-rule]*: ($HMA-M \implies HMA-M \implies HMA-M$)

$(\lambda A B. A + map-mat uminus B) ((-) :: 'a :: group-add \wedge 'nc \wedge 'nr \Rightarrow 'a \wedge 'nc \wedge 'nr$
 $\Rightarrow 'a \wedge 'nc \wedge 'nr)$
 $\langle proof \rangle$

definition *mat2matofpoly* **where** $mat2matofpoly A = (\chi i j. [A \$ i \$ j :])$

definition *charpoly* **where** *charpoly-def*: $charpoly A = det (mat (monom 1 (Suc 0)) - mat2matofpoly A)$

definition *erase-mat* :: $'a :: zero \wedge 'nc \wedge 'nr \Rightarrow 'nr \Rightarrow 'nc \Rightarrow 'a \wedge 'nc \wedge 'nr$

where $erase-mat A i j = (\chi i'. \chi j'. \text{if } i' = i \vee j' = j \text{ then } 0 \text{ else } A \$ i' \$ j')$

definition *sum-UNIV-type* :: $('n :: finite \Rightarrow 'a :: comm-monoid-add) \Rightarrow 'n \text{ itself}$
 $\Rightarrow 'a$ **where**

$sum-UNIV-type f - = sum f UNIV$

definition *sum-UNIV-set* :: (nat \Rightarrow 'a :: comm-monoid-add) \Rightarrow nat \Rightarrow 'a **where**
sum-UNIV-set f n = sum f {..*n*}

definition *HMA-T* :: nat \Rightarrow 'n :: finite itself \Rightarrow bool **where**
HMA-T n = (n = CARD('n))

lemma *HMA-mat2matofpoly*[*transfer-rule*]: (*HMA-M* \implies *HMA-M*) (λ x. map-mat
(λ a. [:a:]) x) mat2matofpoly
<proof>

lemma *HMA-char-poly* [*transfer-rule*]:
((*HMA-M* :: ('a :: comm-ring-1 mat \Rightarrow 'aⁿ \Rightarrow bool)) \implies (=)) *char-poly*
charpoly
<proof>

lemma *HMA-eigen-vector* [*transfer-rule*]: (*HMA-M* \implies *HMA-V* \implies (=))
eigenvector *eigen-vector*
<proof>

lemma *HMA-eigen-value* [*transfer-rule*]: (*HMA-M* \implies (=) \implies (=)) *eigen-*
value *eigen-value*
<proof>

lemma *HMA-spectral-radius* [*transfer-rule*]:
(*HMA-M* \implies (=)) *Spectral-Radius.spectral-radius* *spectral-radius*
<proof>

lemma *HMA-elements-mat*[*transfer-rule*]: ((*HMA-M* :: ('a mat \Rightarrow 'aⁿ \Rightarrow bool)) \implies (=))
elements-mat *elements-mat-h*
<proof>

lemma *HMA-vec-elements*[*transfer-rule*]: ((*HMA-V* :: ('a Matrix.vec \Rightarrow 'aⁿ \Rightarrow bool)) \implies (=))
vec-elements *vec-elements-h*
<proof>

lemma *norm-bound-elements-mat*: *norm-bound* A b = (\forall x \in *elements-mat* A.
norm x \leq b)
<proof>

lemma *HMA-normbound* [*transfer-rule*]:
((*HMA-M* :: 'a :: real-normed-field mat \Rightarrow 'aⁿ \Rightarrow bool) \implies (=))
 \implies (=))
norm-bound *normbound*
<proof>

lemma *HMA-map-matrix* [*transfer-rule*]:
 $((=) \implies \text{HMA-M} \implies \text{HMA-M}) \text{ map-mat map-matrix}$
<proof>

lemma *HMA-transpose-matrix* [*transfer-rule*]:
 $(\text{HMA-M} \implies \text{HMA-M}) \text{ transpose-mat transpose}$
<proof>

lemma *HMA-map-vector* [*transfer-rule*]:
 $((=) \implies \text{HMA-V} \implies \text{HMA-V}) \text{ map-vec map-vector}$
<proof>

lemma *HMA-similar-mat-wit* [*transfer-rule*]:
 $((\text{HMA-M} :: - \Rightarrow 'a :: \text{comm-ring-1} \wedge 'n \wedge 'n \Rightarrow -) \implies \text{HMA-M} \implies \text{HMA-M} \implies \text{HMA-M} \implies (=))$
similar-mat-wit similar-matrix-wit
<proof>

lemma *HMA-similar-mat* [*transfer-rule*]:
 $((\text{HMA-M} :: - \Rightarrow 'a :: \text{comm-ring-1} \wedge 'n \wedge 'n \Rightarrow -) \implies \text{HMA-M} \implies (=))$
similar-mat similar-matrix
<proof>

lemma *HMA-spectrum*[*transfer-rule*]: $(\text{HMA-M} \implies (=)) \text{ spectrum Spectrum}$
<proof>

lemma *HMA-M-erase-mat*[*transfer-rule*]: $(\text{HMA-M} \implies \text{HMA-I} \implies \text{HMA-I} \implies \text{HMA-M}) \text{ mat-erase erase-mat}$
<proof>

lemma *HMA-M-sum-UNIV*[*transfer-rule*]:
 $((\text{HMA-I} \implies (=)) \implies \text{HMA-T} \implies (=)) \text{ sum-UNIV-set sum-UNIV-type}$
<proof>

end

Setup a method to easily convert theorems from JNF into HMA.

method *transfer-hma* **uses** *rule* = (
(fold index-hma-def)?,
transfer,
rule rule,
(unfold carrier-vec-def carrier-mat-def)?,
auto)

Now it becomes easy to transfer results which are not yet proven in HMA, such as:

lemma *matrix-add-vect-distrib*: $(A + B) *v v = A *v v + B *v v$
<proof>

lemma *matrix-vector-right-distrib*: $M *v (v + w) = M *v v + M *v w$
 ⟨proof⟩

lemma *matrix-vector-right-distrib-diff*: $(M :: 'a :: ring-1 \hat{'}nr \hat{'}nc) *v (v - w)$
 $= M *v v - M *v w$
 ⟨proof⟩

lemma *eigen-value-root-charpoly*:
eigen-value $A k \longleftrightarrow poly (charpoly (A :: 'a :: field \hat{'}n \hat{'}n)) k = 0$
 ⟨proof⟩

lemma *finite-spectrum*: **fixes** $A :: 'a :: field \hat{'}n \hat{'}n$
shows *finite* (*Collect* (*eigen-value* A))
 ⟨proof⟩

lemma *non-empty-spectrum*: **fixes** $A :: complex \hat{'}n \hat{'}n$
shows *Collect* (*eigen-value* A) $\neq \{\}$
 ⟨proof⟩

lemma *charpoly-transpose*: $charpoly (transpose A :: 'a :: field \hat{'}n \hat{'}n) = charpoly$
 A
 ⟨proof⟩

lemma *eigen-value-transpose*: *eigen-value* ($transpose A :: 'a :: field \hat{'}n \hat{'}n$) $v =$
eigen-value $A v$
 ⟨proof⟩

lemma *matrix-diff-vect-distrib*: $(A - B) *v v = A *v v - B *v (v :: 'a :: ring-1 \hat{'}n \hat{'}n)$
 ⟨proof⟩

lemma *similar-matrix-charpoly*: *similar-matrix* $A B \implies charpoly A = charpoly B$
 ⟨proof⟩

lemma *pderiv-char-poly-erase-mat*: **fixes** $A :: 'a :: idom \hat{'}n \hat{'}n$
shows *monom* $1 1 * pderiv (charpoly A) = sum (\lambda i. charpoly (erase-mat A i$
 $i)) UNIV$
 ⟨proof⟩

lemma *degree-monic-charpoly*: **fixes** $A :: 'a :: comm-ring-1 \hat{'}n \hat{'}n$
shows *degree* ($charpoly A$) = $CARD('n) \wedge monic (charpoly A)$
 ⟨proof⟩

end

4 Perron-Frobenius Theorem

4.1 Auxiliary Notions

We define notions like non-negative real-valued matrix, both in JNF and in HMA. These notions will be linked via HMA-connect.

theory *Perron-Frobenius-Aux*
imports *HMA-Connect*
begin

definition *real-nonneg-mat* :: *complex mat* \Rightarrow *bool* **where**
real-nonneg-mat $A \equiv \forall a \in \text{elements-mat } A. a \in \mathbf{R} \wedge \text{Re } a \geq 0$

definition *real-nonneg-vec* :: *complex Matrix.vec* \Rightarrow *bool* **where**
real-nonneg-vec $v \equiv \forall a \in \text{vec-elements } v. a \in \mathbf{R} \wedge \text{Re } a \geq 0$

definition *real-non-neg-vec* :: *complex* \wedge *'n* \Rightarrow *bool* **where**
real-non-neg-vec $v \equiv (\forall a \in \text{vec-elements-h } v. a \in \mathbf{R} \wedge \text{Re } a \geq 0)$

definition *real-non-neg-mat* :: *complex* \wedge *'nr* \wedge *'nc* \Rightarrow *bool* **where**
real-non-neg-mat $A \equiv (\forall a \in \text{elements-mat-h } A. a \in \mathbf{R} \wedge \text{Re } a \geq 0)$

lemma *real-non-neg-matD*: **assumes** *real-non-neg-mat* A
shows $A \ \$h\ i\ \$h\ j \in \mathbf{R} \ \text{Re } (A \ \$h\ i\ \$h\ j) \geq 0$
<proof>

definition *nonneg-mat* :: *'a* :: *linordered-idom mat* \Rightarrow *bool* **where**
nonneg-mat $A \equiv \forall a \in \text{elements-mat } A. a \geq 0$

definition *non-neg-mat* :: *'a* :: *linordered-idom* \wedge *'nr* \wedge *'nc* \Rightarrow *bool* **where**
non-neg-mat $A \equiv (\forall a \in \text{elements-mat-h } A. a \geq 0)$

context includes *lifting-syntax*
begin

lemma *HMA-real-non-neg-mat* [*transfer-rule*]:
 $((\text{HMA-M} :: \text{complex mat} \Rightarrow \text{complex } \wedge \text{'nc } \wedge \text{'nr} \Rightarrow \text{bool}) \implies (=))$
real-nonneg-mat *real-non-neg-mat*
<proof>

lemma *HMA-real-non-neg-vec* [*transfer-rule*]:
 $((\text{HMA-V} :: \text{complex Matrix.vec} \Rightarrow \text{complex } \wedge \text{'n} \Rightarrow \text{bool}) \implies (=))$
real-nonneg-vec *real-non-neg-vec*
<proof>

lemma *HMA-non-neg-mat* [*transfer-rule*]:
 $((\text{HMA-M} :: \text{'a} :: \text{linordered-idom mat} \Rightarrow \text{'a } \wedge \text{'nc } \wedge \text{'nr} \Rightarrow \text{bool}) \implies (=))$
nonneg-mat *non-neg-mat*

$\langle \text{proof} \rangle$
end
primrec *matpow* :: 'a::semiring-1[^]n[^]n \Rightarrow nat \Rightarrow 'a[^]n[^]n **where**
matpow-0: *matpow* A 0 = *mat* 1 |
matpow-Suc: *matpow* A (Suc n) = (*matpow* A n) ** A
context includes *lifting-syntax*
begin
lemma *HMA-pow-mat*[*transfer-rule*]:
 ((*HMA-M* :: 'a::{semiring-1} mat \Rightarrow 'a[^]n[^]n \Rightarrow bool) \implies (=) \implies *HMA-M*)
pow-mat *matpow*
 $\langle \text{proof} \rangle$
end
lemma *trancl-image*:
 (*i,j* \in R^+ \implies (*f i, f j*) \in (*map-prod* *f f* ' R)⁺)
 $\langle \text{proof} \rangle$
lemma *inj-trancl-image*: **assumes** *inj*: *inj* *f*
shows (*f i, f j*) \in (*map-prod* *f f* ' R)⁺ = ((*i,j*) \in R^+) (**is** ?l = ?r)
 $\langle \text{proof} \rangle$
lemma *matrix-add-rdistrib*: ((*B + C*) ** *A*) = (*B* ** *A*) + (*C* ** *A*)
 $\langle \text{proof} \rangle$
lemma *norm-smult*: *norm* ((*a* :: real) *s *x*) = *abs* *a* * *norm* *x*
 $\langle \text{proof} \rangle$
lemma *nonneg-mat-mult*:
nonneg-mat *A* \implies *nonneg-mat* *B* \implies *A* \in *carrier-mat* nr *n*
 \implies *B* \in *carrier-mat* n *nc* \implies *nonneg-mat* (*A* * *B*)
 $\langle \text{proof} \rangle$
lemma *nonneg-mat-power*: **assumes** *A* \in *carrier-mat* n n *nonneg-mat* *A*
shows *nonneg-mat* (*A* [^]_{*m*} *k*)
 $\langle \text{proof} \rangle$
lemma *nonneg-matD*: **assumes** *nonneg-mat* *A*
and *i* < *dim-row* *A* **and** *j* < *dim-col* *A*
shows *A* \$\$ (*i,j*) \geq 0
 $\langle \text{proof} \rangle$
lemma (**in** *comm-ring-hom*) *similar-mat-wit-hom*: **assumes**
similar-mat-wit *A B C D*
shows *similar-mat-wit* (*mat*_{*h*} *A*) (*mat*_{*h*} *B*) (*mat*_{*h*} *C*) (*mat*_{*h*} *D*)
 $\langle \text{proof} \rangle$

lemma (in *comm-ring-hom*) *similar-mat-hom*:
similar-mat $A B \implies \text{similar-mat } (\text{mat}_h A) (\text{mat}_h B)$
 ⟨*proof*⟩

lemma *det-dim-1*: **assumes** $A: A \in \text{carrier-mat } n \ n$
and $n: n = 1$
shows $\text{Determinant.det } A = A \ \$\$ (0,0)$
 ⟨*proof*⟩

lemma *det-dim-2*: **assumes** $A: A \in \text{carrier-mat } n \ n$
and $n: n = 2$
shows $\text{Determinant.det } A = A \ \$\$ (0,0) * A \ \$\$ (1,1) - A \ \$\$ (0,1) * A \ \$\$ (1,0)$
 ⟨*proof*⟩

lemma *jordan-nf-root-char-poly*: **fixes** $A :: 'a :: \{\text{semiring-no-zero-divisors, idom}\}$
 mat
assumes *jordan-nf* $A \ n\text{-as}$
and $(m, \text{lam}) \in \text{set } n\text{-as}$
shows $\text{poly } (\text{char-poly } A) \ \text{lam} = 0$
 ⟨*proof*⟩

lemma *inverse-power-tendsto-zero*:
 $(\lambda x. \text{inverse } ((\text{of-nat } x :: 'a :: \text{real-normed-div-algebra}) \wedge \text{Suc } d)) \longrightarrow 0$
 ⟨*proof*⟩

lemma *inverse-of-nat-tendsto-zero*:
 $(\lambda x. \text{inverse } (\text{of-nat } x :: 'a :: \text{real-normed-div-algebra})) \longrightarrow 0$
 ⟨*proof*⟩

lemma *poly-times-exp-tendsto-zero*: **assumes** $b: \text{norm } (b :: 'a :: \text{real-normed-field}) < 1$
shows $(\lambda x. \text{of-nat } x \wedge k * b \wedge x) \longrightarrow 0$
 ⟨*proof*⟩

lemma (in *linorder-topology*) *tendsto-Min*: **assumes** $I: I \neq \{\}$ **and** *fin*: *finite* I
shows $(\bigwedge i. i \in I \implies (f \ i \longrightarrow a \ i) \ F) \implies ((\lambda x. \text{Min } ((\lambda i. f \ i \ x) \ 'I)) \longrightarrow (\text{Min } (a \ 'I) :: 'a)) \ F$
 ⟨*proof*⟩

lemma *tendsto-mat-mult* [*tendsto-intros*]:
 $(f \longrightarrow a) \ F \implies (g \longrightarrow b) \ F \implies ((\lambda x. f \ x ** g \ x) \longrightarrow a ** b) \ F$
for $f :: 'a \Rightarrow 'b :: \{\text{semiring-1, real-normed-algebra}\} \wedge 'n1 \wedge 'n2$
 ⟨*proof*⟩

lemma *tendsto-matpower* [*tendsto-intros*]: $(f \longrightarrow a) \ F \implies ((\lambda x. \text{matpow } (f \ x) \ n) \longrightarrow \text{matpow } a \ n) \ F$
for $f :: 'a \Rightarrow 'b :: \{\text{semiring-1, real-normed-algebra}\} \wedge 'n \wedge 'n$

<proof>

lemma *continuous-matpow*: *continuous-on* R ($\lambda A :: 'a :: \{\text{semiring-1}, \text{real-normed-algebra-1}\}$
 $\wedge 'n \wedge 'n. \text{matpow } A \ n$)
<proof>

lemma *vector-smult-distrib*: $(A * v ((a :: 'a :: \text{comm-ring-1}) * s \ x)) = a * s ((A * v \ x))$
<proof>

instance *real* :: *ordered-semiring-strict*
<proof>

lemma *poly-tendsto-pinfty*: **fixes** $p :: \text{real}$ *poly*
assumes $\text{lead-coeff } p > 0$ $\text{degree } p \neq 0$
shows $\text{poly } p \longrightarrow \infty$
<proof>

lemma *div-lt-nat*: $(j :: \text{nat}) < x * y \implies j \ \text{div} \ x < y$
<proof>

definition *diagvector* :: $('n \Rightarrow 'a :: \text{semiring-0}) \Rightarrow 'a \wedge 'n \wedge 'n$ **where**
 $\text{diagvector } x = (\chi \ i. \chi \ j. \text{if } i = j \ \text{then } x \ i \ \text{else } 0)$

lemma *diagvector-mult-vector[simp]*: $\text{diagvector } x * v \ y = (\chi \ i. x \ i * y \ \$ \ i)$
<proof>

lemma *diagvector-mult-left*: $\text{diagvector } x ** A = (\chi \ i \ j. x \ i * A \ \$ \ i \ \$ \ j)$ (**is** $?A = ?B$)
<proof>

lemma *diagvector-mult-right*: $A ** \text{diagvector } x = (\chi \ i \ j. A \ \$ \ i \ \$ \ j * x \ j)$ (**is** $?A = ?B$)
<proof>

lemma *diagvector-mult[simp]*: $\text{diagvector } x ** \text{diagvector } y = \text{diagvector } (\lambda \ i. x \ i * y \ i)$
<proof>

lemma *diagvector-const[simp]*: $\text{diagvector } (\lambda \ x. k) = \text{mat } k$
<proof>

lemma *diagvector-eq-mat*: $\text{diagvector } x = \text{mat } a \iff x = (\lambda \ x. a)$
<proof>

lemma *cmod-eq-Re*: **assumes** $\text{cmod } x = \text{Re } x$
shows $\text{of-real } (\text{Re } x) = x$
<proof>

hide-fact (open) *Matrix.vec-eq-iff*

no-notation

vec-index (infixl \$ 100)

lemma *spectral-radius-ev*:

$\exists ev v. \text{eigen-vector } A v \wedge \text{norm } ev = \text{spectral-radius } A$
(proof)

lemma *spectral-radius-max*: **assumes** *eigen-value* $A v$

shows $\text{norm } v \leq \text{spectral-radius } A$
(proof)

For Perron-Frobenius it is useful to use the linear norm, and not the Euclidean norm.

definition *norm1* :: 'a :: real-normed-field ^ 'n \Rightarrow real **where**
 $\text{norm1 } v = (\sum_{i \in UNIV} \text{norm } (v \ \$ \ i))$

lemma *norm1-ge-0*: $\text{norm1 } v \geq 0$ (proof)

lemma *norm1-0[simp]*: $\text{norm1 } 0 = 0$ (proof)

lemma *norm1-nonzero*: **assumes** $v \neq 0$

shows $\text{norm1 } v > 0$
(proof)

lemma *norm1-0-iff[simp]*: $(\text{norm1 } v = 0) = (v = 0)$

(proof)

lemma *norm1-scaleR[simp]*: $\text{norm1 } (r *_R v) = \text{abs } r * \text{norm1 } v$ (proof)

lemma *abs-norm1[simp]*: $\text{abs } (\text{norm1 } v) = \text{norm1 } v$ (proof)

lemma *normalize-eigen-vector*: **assumes** *eigen-vector* $(A :: 'a :: \text{real-normed-field} \wedge 'n \wedge 'n) v \text{ ev}$

shows *eigen-vector* $A ((1 / \text{norm1 } v) *_R v) \text{ ev}$ $\text{norm1 } ((1 / \text{norm1 } v) *_R v) = 1$
(proof)

lemma *norm1-cont[simp]*: *isCont* $\text{norm1 } v$ (proof)

lemma *norm1-ge-norm*: $\text{norm1 } v \geq \text{norm } v$ (proof)

The following continuity lemmas have been proven with hints from Fabian Immler.

lemma *tendsto-matrix-vector-mult[tendsto-intros]*:

$((*v) (A :: 'a :: \text{real-normed-algebra-1} \wedge 'n \wedge 'k) \longrightarrow A *v v)$ (at v within S)
(proof)

lemma *tendsto-matrix-matrix-mult*[*tendsto-intros*]:
 $((*) (A :: 'a :: \text{real-normed-algebra-1 } ^\wedge 'n \ ^\wedge 'k) \longrightarrow A ** B) \text{ (at } B \text{ within } S)$
 $\langle \text{proof} \rangle$

lemma *matrix-vect-scaleR*: $(A :: 'a :: \text{real-normed-algebra-1 } ^\wedge 'n \ ^\wedge 'k) * v (a *_R v)$
 $= a *_R (A * v v)$
 $\langle \text{proof} \rangle$

lemma (in *inj-semiring-hom*) *map-vector-0*: $(\text{map-vector hom } v = 0) = (v = 0)$
 $\langle \text{proof} \rangle$

lemma (in *inj-semiring-hom*) *map-vector-inj*: $(\text{map-vector hom } v = \text{map-vector hom } w) = (v = w)$
 $\langle \text{proof} \rangle$

lemma (in *semiring-hom*) *matrix-vector-mult-hom*:
 $(\text{map-matrix hom } A) * v (\text{map-vector hom } v) = \text{map-vector hom } (A * v v)$
 $\langle \text{proof} \rangle$

lemma (in *semiring-hom*) *vector-smult-hom*:
 $\text{hom } x * s (\text{map-vector hom } v) = \text{map-vector hom } (x * s v)$
 $\langle \text{proof} \rangle$

lemma (in *inj-comm-ring-hom*) *eigen-vector-hom*:
 $\text{eigen-vector } (\text{map-matrix hom } A) (\text{map-vector hom } v) (\text{hom } x) = \text{eigen-vector } A$
 $v x$
 $\langle \text{proof} \rangle$

end

4.2 Perron-Frobenius theorem via Brouwer's fixpoint theorem.

theory *Perron-Frobenius*
imports
HOL-Analysis.Brouwer-Fixpoint
Perron-Frobenius-Aux
begin

We follow the textbook proof of Serre [2, Theorem 5.2.1].

context
fixes $A :: \text{complex } ^\wedge 'n \ ^\wedge 'n :: \text{finite}$
assumes $\text{rnn}A: \text{real-non-neg-mat } A$
begin

private abbreviation(*input*) sr **where** $sr \equiv \text{spectral-radius } A$

private definition $\text{max-v-ev} :: (\text{complex } ^\wedge 'n) \times \text{complex}$ **where**

$max-v-ev = (SOME\ v-ev.\ eigen-vector\ A\ (fst\ v-ev)\ (snd\ v-ev))$
 $\wedge\ norm\ (snd\ v-ev) = sr$

private definition $max-v = (1 / norm1\ (fst\ max-v-ev)) *R\ fst\ max-v-ev$

private definition $max-ev = snd\ max-v-ev$

private lemma $max-v-ev$:

$eigen-vector\ A\ max-v\ max-ev$

$norm\ max-ev = sr$

$norm1\ max-v = 1$

$\langle proof \rangle$

In the definition of S , we use the linear norm instead of the default euclidean norm which is defined via the type-class. The reason is that S is not convex if one uses the euclidean norm.

private definition $B :: real\ ^n\ ^n$ **where** $B \equiv \chi\ i\ j.\ Re\ (A\ \$\ i\ \$\ j)$

private definition S **where** $S = \{v :: real\ ^n.\ norm1\ v = 1 \wedge (\forall\ i.\ v\ \$\ i \geq 0)\} \wedge$

$(\forall\ i.\ (B *v\ v)\ \$\ i \geq sr * (v\ \$\ i))\}$

private definition $f :: real\ ^n \Rightarrow real\ ^n$ **where**

$f\ v = (1 / norm1\ (B *v\ v)) *R\ (B *v\ v)$

private lemma $closedS$: $closed\ S$

$\langle proof \rangle$ **lemma** $boundedS$: $bounded\ S$

$\langle proof \rangle$ **lemma** $compactS$: $compact\ S$

$\langle proof \rangle$ **lemmas** $rnn = real-non-neg-matD[OF\ rnnA]$

lemma $B-norm$: $B\ \$\ i\ \$\ j = norm\ (A\ \$\ i\ \$\ j)$

$\langle proof \rangle$

lemma $mult-B-mono$: **assumes** $\bigwedge\ i.\ v\ \$\ i \geq w\ \$\ i$

shows $(B *v\ v)\ \$\ i \geq (B *v\ w)\ \$\ i$ $\langle proof \rangle$ **lemma** $non-emptyS$: $S \neq \{\}$

$\langle proof \rangle$ **lemma** $convexS$: $convex\ S$

$\langle proof \rangle$ **abbreviation** $(input)\ r :: real \Rightarrow complex$ **where**

$r \equiv of-real$

private abbreviation $rv :: real\ ^n \Rightarrow complex\ ^n$ **where**

$rv\ v \equiv \chi\ i.\ r\ (v\ \$\ i)$

private lemma $rv-0$: $(rv\ v = 0) = (v = 0)$

$\langle proof \rangle$ **lemma** $rv-mult$: $A *v\ rv\ v = rv\ (B *v\ v)$

$\langle proof \rangle$

context

assumes $zero-no-ev$: $\bigwedge\ v.\ v \in S \implies A *v\ rv\ v \neq 0$

begin

private lemma $normB-S$: **assumes** v : $v \in S$

shows $norm1\ (B *v\ v) \neq 0$

$\langle proof \rangle$ **lemma** $image-f$: $f\ 'S \subseteq S$

<proof> **lemma** *cont-f: continuous-on S f*
 <proof> **lemma** *perron-frobenius-positive-ev:*
 $\exists v. \text{eigen-vector } A \ v \ (r \ sr) \wedge \text{real-non-neg-vec } v$
 <proof>
end

qualified lemma *perron-frobenius-both:*
 $\exists v. \text{eigen-vector } A \ v \ (r \ sr) \wedge \text{real-non-neg-vec } v$
 <proof>
end

Perron Frobenius: The largest complex eigenvalue of a real-valued non-negative matrix is a real one, and it has a real-valued non-negative eigenvector.

lemma *perron-frobenius:*
assumes *real-non-neg-mat A*
shows $\exists v. \text{eigen-vector } A \ v \ (\text{of-real } (\text{spectral-radius } A)) \wedge \text{real-non-neg-vec } v$
 <proof>

And a version which ignores the eigenvector.

lemma *perron-frobenius-eigen-value:*
assumes *real-non-neg-mat A*
shows *eigen-value A (of-real (spectral-radius A))*
 <proof>

end

5 Roots of Unity

theory *Roots-Unity*

imports

Polynomial-Factorization.Order-Polynomial
HOL-Computational-Algebra.Fundamental-Theorem-Algebra
Polynomial-Interpolation.Ring-Hom-Poly

begin

lemma *cis-mult-cmod-id:* $\text{cis } (\text{Arg } x) * \text{of-real } (\text{cmod } x) = x$
 <proof>

lemma *rcis-mult-cis[simp]:* $\text{rcis } n \ a * \text{cis } b = \text{rcis } n \ (a + b)$ <proof>

lemma *rcis-div-cis[simp]:* $\text{rcis } n \ a / \text{cis } b = \text{rcis } n \ (a - b)$ <proof>

lemma *cis-plus-2pi[simp]:* $\text{cis } (x + 2 * \text{pi}) = \text{cis } x$ <proof>

lemma *cis-plus-2pi-neq-1:* **assumes** $x: 0 < x < 2 * \text{pi}$
shows $\text{cis } x \neq 1$
 <proof>

lemma *cis-times-2pi[simp]:* $\text{cis } (\text{of-nat } n * 2 * \text{pi}) = 1$
 <proof>

lemma *cis-add-pi[simp]*: $\text{cis } (\pi + x) = - \text{cis } x$
 ⟨proof⟩

lemma *cis-3-pi-2[simp]*: $\text{cis } (\pi * 3 / 2) = - i$
 ⟨proof⟩

lemma *rcis-plus-2pi[simp]*: $\text{rcis } y (x + 2 * \pi) = \text{rcis } y x$ ⟨proof⟩
lemma *rcis-times-2pi[simp]*: $\text{rcis } r (\text{of-nat } n * 2 * \pi) = \text{of-real } r$
 ⟨proof⟩

lemma *arg-rcis-cis*: **assumes** $n: n > 0$ **shows** $\text{Arg } (\text{rcis } n x) = \text{Arg } (\text{cis } x)$
 ⟨proof⟩

lemma *arg-eqD*: **assumes** $\text{Arg } (\text{cis } x) = \text{Arg } (\text{cis } y) - \pi < x \leq \pi - \pi < y \leq \pi$
shows $x = y$
 ⟨proof⟩

lemma *rcis-inj-on*: **assumes** $r: r \neq 0$ **shows** *inj-on* $(\text{rcis } r) \{0 ..< 2 * \pi\}$
 ⟨proof⟩

lemma *cis-inj-on*: *inj-on* $\text{cis } \{0 ..< 2 * \pi\}$
 ⟨proof⟩

definition *root-unity* :: $\text{nat} \Rightarrow 'a :: \text{comm-ring-1 poly}$ **where**
 $\text{root-unity } n = \text{monom } 1 n - 1$

lemma *poly-root-unity*: $\text{poly } (\text{root-unity } n) x = 0 \iff x^n = 1$
 ⟨proof⟩

lemma *degree-root-unity[simp]*: $\text{degree } (\text{root-unity } n) = n$ (**is** $\text{degree } ?p = -$)
 ⟨proof⟩

lemma *zero-root-unity[simp]*: $\text{root-unity } n = 0 \iff n = 0$ (**is** $?p = 0 \iff -$)
 ⟨proof⟩

definition *prod-root-unity* :: $\text{nat list} \Rightarrow 'a :: \text{idom poly}$ **where**
 $\text{prod-root-unity } ns = \text{prod-list } (\text{map } \text{root-unity } ns)$

lemma *poly-prod-root-unity*: $\text{poly } (\text{prod-root-unity } ns) x = 0 \iff (\exists k \in \text{set } ns. x^k = 1)$
 ⟨proof⟩

lemma *degree-prod-root-unity[simp]*: $0 \notin \text{set } ns \implies \text{degree } (\text{prod-root-unity } ns) = \text{sum-list } ns$
 ⟨proof⟩

lemma *zero-prod-root-unity[simp]*: $\text{prod-root-unity } ns = 0 \iff 0 \in \text{set } ns$

<proof>

lemma roots-of-unity: assumes $n: n \neq 0$
shows $(\lambda i. (cis (of-nat i * 2 * pi / n))) \text{ ' } \{0 ..< n\} = \{x :: complex. x \wedge n = 1\}$ **(is** $?prod = ?Roots$)
 $\{x. poly (root-unity n) x = 0\} = \{x :: complex. x \wedge n = 1\}$
 $card \{x :: complex. x \wedge n = 1\} = n$
<proof>

lemma poly-roots-dvd: fixes $p :: 'a :: field$ *poly*
assumes $p \neq 0$ **and** $degree p = n$
and $card \{x. poly p x = 0\} \geq n$ **and** $\{x. poly p x = 0\} \subseteq \{x. poly q x = 0\}$
shows $p \text{ dvd } q$
<proof>

lemma root-unity-decomp: assumes $n: n \neq 0$
shows $root-unity n =$
 $prod-list (map (\lambda i. [:-cis (of-nat i * 2 * pi / n), 1:]) [0 ..< n])$ **(is** $?u = ?p$)
<proof>

lemma order-monic-linear: order $x [y,1:] = (if y + x = 0 then 1 else 0)$
<proof>

lemma order-root-unity: fixes $x :: complex$ **assumes** $n: n \neq 0$
shows $order x (root-unity n) = (if x \wedge n = 1 then 1 else 0)$
(is $order - ?u = -$)
<proof>

lemma order-prod-root-unity: assumes $0: 0 \notin set ks$
shows $order (x :: complex) (prod-root-unity ks) = length (filter (\lambda k. x \wedge k = 1) ks)$
<proof>

lemma root-unity-witness: fixes $xs :: complex list$
assumes $prod-list (map (\lambda x. [:-x,1:]) xs) = monom 1 n - 1$
shows $x \wedge n = 1 \iff x \in set xs$
<proof>

lemma root-unity-explicit: fixes $x :: complex$
shows
 $(x \wedge 1 = 1) \iff x = 1$
 $(x \wedge 2 = 1) \iff (x \in \{1, -1\})$
 $(x \wedge 3 = 1) \iff (x \in \{1, Complex (-1/2) (sqrt 3 / 2), Complex (-1/2) (-sqrt 3 / 2)\})$
 $(x \wedge 4 = 1) \iff (x \in \{1, -1, i, -i\})$
<proof>

definition primitive-root-unity :: nat \Rightarrow 'a :: power \Rightarrow bool **where**
 $primitive-root-unity k x = (k \neq 0 \wedge x \wedge k = 1 \wedge (\forall k' < k. k' \neq 0 \longrightarrow x \wedge k' \neq 1))$

1))

lemma *primitive-root-unityD*: **assumes** *primitive-root-unity* k x
shows $k \neq 0 \ x \wedge k = 1 \ k' \neq 0 \implies x \wedge k' = 1 \implies k \leq k'$
(*proof*)

lemma *primitive-root-unity-exists*: **assumes** $k \neq 0 \ x \wedge k = 1$
shows $\exists k'. k' \leq k \wedge \text{primitive-root-unity } k' \ x$
(*proof*)

lemma *primitive-root-unity-dvd*: **fixes** $x :: \text{complex}$
assumes k : *primitive-root-unity* k x
shows $x \wedge n = 1 \longleftrightarrow k \text{ dvd } n$
(*proof*)

lemma *primitive-root-unity-simple-computation*:
primitive-root-unity k $x = (\text{if } k = 0 \text{ then } \text{False} \text{ else}$
 $x \wedge k = 1 \wedge (\forall i \in \{1 .. k\}. x \wedge i \neq 1))$
(*proof*)

lemma *primitive-root-unity-explicit*: **fixes** $x :: \text{complex}$
shows *primitive-root-unity* 1 $x \longleftrightarrow x = 1$
primitive-root-unity 2 $x \longleftrightarrow x = -1$
primitive-root-unity 3 $x \longleftrightarrow (x \in \{\text{Complex } (-1/2) (\text{sqrt } 3 / 2), \text{Complex}$
 $(-1/2) (-\text{sqrt } 3 / 2)\})$
primitive-root-unity 4 $x \longleftrightarrow (x \in \{i, -i\})$
(*proof*)

function *decompose-prod-root-unity-main* ::
 $'a :: \text{field poly} \Rightarrow \text{nat} \Rightarrow \text{nat list} \times 'a \text{ poly}$ **where**
decompose-prod-root-unity-main p $k =$ (
 if $k = 0$ *then* ($[], p$) *else*
 let $q = \text{root-unity } k$ *in* *if* $q \text{ dvd } p$ *then* *if* $p = 0$ *then* ($[], 0$) *else*
 map-prod (*Cons* k) *id* (*decompose-prod-root-unity-main* ($p \text{ div } q$) k) *else*
 decompose-prod-root-unity-main p ($k - 1$)
(*proof*)

termination (*proof*)

declare *decompose-prod-root-unity-main.simps*[*simp del*]

lemma *decompose-prod-root-unity-main*: **fixes** $p :: \text{complex poly}$
assumes p : $p = \text{prod-root-unity } ks * f$
and d : *decompose-prod-root-unity-main* p $k = (ks', g)$
and f : $\bigwedge x. \text{cmod } x = 1 \implies \text{poly } f \ x \neq 0$
and k : $\bigwedge k'. k' > k \implies \neg \text{root-unity } k' \text{ dvd } p$
shows $p = \text{prod-root-unity } ks' * f \wedge f = g \wedge \text{set } ks = \text{set } ks'$
(*proof*)

definition *decompose-prod-root-unity* $p = \text{decompose-prod-root-unity-main } p$ (degree p)

lemma *decompose-prod-root-unity*: **fixes** $p :: \text{complex poly}$

assumes $p: p = \text{prod-root-unity } ks * f$

and $d: \text{decompose-prod-root-unity } p = (ks', g)$

and $f: \bigwedge x. \text{cmod } x = 1 \implies \text{poly } f x \neq 0$

and $p0: p \neq 0$

shows $p = \text{prod-root-unity } ks' * f \wedge f = g \wedge \text{set } ks = \text{set } ks'$

$\langle \text{proof} \rangle$

lemma (in *comm-ring-hom*) *hom-root-unity*: $\text{map-poly } \text{hom} (\text{root-unity } n) = \text{root-unity } n$

$\langle \text{proof} \rangle$

lemma (in *idom-hom*) *hom-prod-root-unity*: $\text{map-poly } \text{hom} (\text{prod-root-unity } n) = \text{prod-root-unity } n$

$\langle \text{proof} \rangle$

lemma (in *field-hom*) *hom-decompose-prod-root-unity-main*:

$\text{decompose-prod-root-unity-main } (\text{map-poly } \text{hom } p) k = \text{map-prod id } (\text{map-poly } \text{hom})$

$(\text{decompose-prod-root-unity-main } p k)$

$\langle \text{proof} \rangle$

lemma (in *field-hom*) *hom-decompose-prod-root-unity*:

$\text{decompose-prod-root-unity } (\text{map-poly } \text{hom } p) = \text{map-prod id } (\text{map-poly } \text{hom})$

$(\text{decompose-prod-root-unity } p)$

$\langle \text{proof} \rangle$

end

5.1 The Perron Frobenius Theorem for Irreducible Matrices

theory *Perron-Frobenius-Irreducible*

imports

Perron-Frobenius

Roots-Unity

Rank-Nullity-Theorem.Miscellaneous

begin

lifting-forget *vec.lifting*

lifting-forget *mat.lifting*

lifting-forget *poly.lifting*

lemma *charpoly-of-real*: $\text{charpoly } (\text{map-matrix } \text{complex-of-real } A) = \text{map-poly of-real } (\text{charpoly } A)$

$\langle \text{proof} \rangle$

context includes *lifting-syntax*

begin

lemma *HMA-M-smult*[*transfer-rule*]: $((=) \implies \text{HMA-M} \implies \text{HMA-M}) (\cdot_m)$
 $((*k))$

<proof>

end

lemma *order-charpoly-smult*: **fixes** $A :: \text{complex } ^{\wedge} 'n \wedge 'n$

assumes $k: k \neq 0$

shows $\text{order } x (\text{charpoly } (k *k A)) = \text{order } (x / k) (\text{charpoly } A)$

<proof>

lemma *smult-eigen-vector*: **fixes** $a :: 'a :: \text{field}$

assumes *eigen-vector* $A v x$

shows *eigen-vector* $(a *k A) v (a * x)$

<proof>

lemma *smult-eigen-value*: **fixes** $a :: 'a :: \text{field}$

assumes *eigen-value* $A x$

shows *eigen-value* $(a *k A) (a * x)$

<proof>

locale *fixed-mat* = **fixes** $A :: 'a :: \text{zero } ^{\wedge} 'n \wedge 'n$

begin

definition $G :: 'n \text{ rel}$ **where**

$G = \{ (i,j). A \$ i \$ j \neq 0 \}$

definition *irreducible* :: *bool* **where**

irreducible = $(UNIV \subseteq G^{\wedge+})$

end

lemma *G-transpose*:

$\text{fixed-mat}.G (\text{transpose } A) = ((\text{fixed-mat}.G A))^{\wedge-1}$

<proof>

lemma *G-transpose-trancl*:

$(\text{fixed-mat}.G (\text{transpose } A))^{\wedge+} = ((\text{fixed-mat}.G A)^{\wedge+})^{\wedge-1}$

<proof>

locale *pf-nonneg-mat* = *fixed-mat* A **for**

$A :: 'a :: \text{linordered-idom } ^{\wedge} 'n \wedge 'n +$

assumes *non-neg-mat*: *non-neg-mat* A

begin

lemma *nonneg*: $A \$ i \$ j \geq 0$

<proof>

lemma *nonneg-matpow*: *matpow* $A n \$ i \$ j \geq 0$

<proof>

lemma *G-relpow-matpow-pos*: $(i,j) \in G \overset{\sim}{\sim} n \implies \text{matpow } A \ n \ \$ \ i \ \$ \ j > 0$
<proof>

lemma *matpow-mono*: **assumes** $B: \bigwedge i j. B \ \$ \ i \ \$ \ j \geq A \ \$ \ i \ \$ \ j$
shows $\text{matpow } B \ n \ \$ \ i \ \$ \ j \geq \text{matpow } A \ n \ \$ \ i \ \$ \ j$
<proof>

lemma *matpow-sum-one-mono*: $\text{matpow } (A + \text{mat } 1) \ (n + k) \ \$ \ i \ \$ \ j \geq \text{matpow } (A + \text{mat } 1) \ n \ \$ \ i \ \$ \ j$
<proof>

lemma *G-relpow-matpow-pos-ge*:
assumes $(i,j) \in G \overset{\sim}{\sim} m \ n \geq m$
shows $\text{matpow } (A + \text{mat } 1) \ n \ \$ \ i \ \$ \ j > 0$
<proof>
end

locale *perron-frobenius* = *pf-nonneg-mat* A
for $A :: \text{real} \ ^n \ ^n +$
assumes *irr*: *irreducible*
begin

definition N **where** $N = (\text{SOME } N. \forall ij. \exists n \leq N. ij \in G \overset{\sim}{\sim} n)$

lemma $N: \exists n \leq N. ij \in G \overset{\sim}{\sim} n$
<proof>

lemma *irreducible-matpow-pos*: **assumes** *irreducible*
shows $\text{matpow } (A + \text{mat } 1) \ N \ \$ \ i \ \$ \ j > 0$
<proof>

lemma *pf-transpose*: *perron-frobenius* (*transpose* A)
<proof>

abbreviation *le-vec* :: $\text{real} \ ^n \ \Rightarrow \ \text{real} \ ^n \ \Rightarrow \ \text{bool}$ **where**
 $\text{le-vec } x \ y \equiv (\forall i. x \ \$ \ i \ \leq \ y \ \$ \ i)$

abbreviation *lt-vec* :: $\text{real} \ ^n \ \Rightarrow \ \text{real} \ ^n \ \Rightarrow \ \text{bool}$ **where**
 $\text{lt-vec } x \ y \equiv (\forall i. x \ \$ \ i \ < \ y \ \$ \ i)$

definition $A1n = \text{matpow } (A + \text{mat } 1) \ N$

lemmas $A1n\text{-pos} = \text{irreducible-matpow-pos}[OF \ \text{irr}, \ \text{folded } A1n\text{-def}]$

definition $r :: \text{real} \ ^n \ \Rightarrow \ \text{real}$ **where**
 $r \ x = \text{Min } \{ (A * v \ x) \ \$ \ j / x \ \$ \ j \mid j. x \ \$ \ j \neq 0 \}$

definition $X :: (\text{real } ^n) \text{set}$ **where**
 $X = \{ x . \text{le-vec } 0 \ x \wedge x \neq 0 \}$

lemma *nonneg-Ax*: $x \in X \implies \text{le-vec } 0 \ (A *v \ x)$
 $\langle \text{proof} \rangle$

lemma *A-nonzero-fixed-i*: $\exists j. A \ \$ \ i \ \$ \ j \neq 0$
 $\langle \text{proof} \rangle$

lemma *A-nonzero-fixed-j*: $\exists i. A \ \$ \ i \ \$ \ j \neq 0$
 $\langle \text{proof} \rangle$

lemma *Ax-pos*: **assumes** $x: \text{lt-vec } 0 \ x$
shows $\text{lt-vec } 0 \ (A *v \ x)$
 $\langle \text{proof} \rangle$

lemma *nonzero-Ax*: **assumes** $x: x \in X$
shows $A *v \ x \neq 0$
 $\langle \text{proof} \rangle$

lemma *r-witness*: **assumes** $x: x \in X$
shows $\exists j. x \ \$ \ j > 0 \wedge r \ x = (A *v \ x) \ \$ \ j / x \ \$ \ j$
 $\langle \text{proof} \rangle$

lemma *rx-nonneg*: **assumes** $x: x \in X$
shows $r \ x \geq 0$
 $\langle \text{proof} \rangle$

lemma *rx-pos*: **assumes** $x: \text{lt-vec } 0 \ x$
shows $r \ x > 0$
 $\langle \text{proof} \rangle$

lemma *rx-le-Ax*: **assumes** $x: x \in X$
shows $\text{le-vec} \ (r \ x *s \ x) \ (A *v \ x)$
 $\langle \text{proof} \rangle$

lemma *rho-le-x-Ax-imp-rho-le-rx*: **assumes** $x: x \in X$
and $\varrho: \text{le-vec} \ (\varrho *s \ x) \ (A *v \ x)$
shows $\varrho \leq r \ x$
 $\langle \text{proof} \rangle$

lemma *rx-Max*: **assumes** $x: x \in X$
shows $r \ x = \text{Sup} \ \{ \varrho . \text{le-vec} \ (\varrho *s \ x) \ (A *v \ x) \}$ (**is** $= \text{Sup} \ ?S$)
 $\langle \text{proof} \rangle$

lemma *r-smult*: **assumes** $x: x \in X$
and $a: a > 0$
shows $r \ (a *s \ x) = r \ x$

<proof>

definition $X1 = (X \cap \{x. \text{norm } x = 1\})$

lemma *bounded-X1: bounded X1* *<proof>*

lemma *closed-X1: closed X1*

<proof>

lemma *compact-X1: compact X1* *<proof>*

definition $\text{pow-A-1 } x = A1n *v x$

lemma *continuous-pow-A-1: continuous-on R pow-A-1*

<proof>

definition $Y = \text{pow-A-1 } ' X1$

lemma *compact-Y: compact Y*

<proof>

lemma *Y-pos-main: assumes y: y ∈ pow-A-1 ' X*

shows $y \ \$ \ i > 0$

<proof>

lemma *Y-pos: assumes y: y ∈ Y*

shows $y \ \$ \ i > 0$

<proof>

lemma *Y-nonzero: assumes y: y ∈ Y*

shows $y \ \$ \ i \neq 0$

<proof>

definition $r' :: \text{real} \ ^n \Rightarrow \text{real}$ **where**

$r' x = \text{Min } (\text{range } (\lambda j. (A *v x) \ \$ \ j / x \ \$ \ j))$

lemma *r'-r: assumes x: x ∈ Y shows r' x = r x*

<proof>

lemma *continuous-Y-r: continuous-on Y r*

<proof>

lemma *X1-nonempty: X1 ≠ {}*

<proof>

lemma *Y-nonempty: Y ≠ {}*

<proof>

definition z where $z = (\text{SOME } z. z \in Y \wedge (\forall y \in Y. r y \leq r z))$

abbreviation $sr \equiv r z$

lemma $z: z \in Y$ and $sr\text{-max-}Y: \bigwedge y. y \in Y \implies r y \leq sr$
<proof>

lemma $Y\text{-subset-}X: Y \subseteq X$
<proof>

lemma $zX: z \in X$
<proof>

lemma $le\text{-vec-}mono\text{-left}$: **assumes** $B: \bigwedge i j. B \$ i \$ j \geq 0$
and $le\text{-vec } x y$
shows $le\text{-vec } (B *v x) (B *v y)$
<proof>

lemma $matpow\text{-}1\text{-commute}$: $matpow (A + mat 1) n ** A = A ** matpow (A + mat 1) n$
<proof>

lemma $A1n\text{-commute}$: $A1n ** A = A ** A1n$
<proof>

lemma $le\text{-vec-pow-}A\text{-}1$: **assumes** $le: le\text{-vec } (rho *s x) (A *v x)$
shows $le\text{-vec } (rho *s pow\text{-}A\text{-}1 x) (A *v pow\text{-}A\text{-}1 x)$
<proof>

lemma $r\text{-pow-}A\text{-}1$: **assumes** $x: x \in X$
shows $r x \leq r (pow\text{-}A\text{-}1 x)$
<proof>

lemma $sr\text{-max}$: **assumes** $x: x \in X$
shows $r x \leq sr$
<proof>

lemma $z\text{-pos}$: $z \$ i > 0$
<proof>

lemma $sr\text{-pos}$: $sr > 0$
<proof>

context **fixes** u
assumes $u: u \in X$ and $ru: r u = sr$
begin

lemma *sr-imp-eigen-vector-main*: $sr * s u = A * v u$
<proof>

lemma *sr-imp-eigen-vector*: *eigen-vector* $A u sr$
<proof>

lemma *sr-u-pos*: *lt-vec* $0 u$
<proof>
end

lemma *eigen-vector-z-sr*: *eigen-vector* $A z sr$
<proof>

lemma *eigen-value-sr*: *eigen-value* $A sr$
<proof>

abbreviation $c \equiv \text{complex-of-real}$

abbreviation $cA \equiv \text{map-matrix } c A$

abbreviation $\text{norm-v} \equiv \text{map-vector } (\text{norm} :: \text{complex} \Rightarrow \text{real})$

lemma *norm-v-ge-0*: *le-vec* $0 (\text{norm-v } v)$ *<proof>*

lemma *norm-v-eq-0*: $\text{norm-v } v = 0 \longleftrightarrow v = 0$ *<proof>*

lemma *cA-index*: $cA \$ i \$ j = c (A \$ i \$ j)$
<proof>

lemma *norm-cA[simp]*: $\text{norm } (cA \$ i \$ j) = A \$ i \$ j$
<proof>

context *fixes* αv

assumes *ev*: *eigen-vector* $cA v \alpha$

begin

lemma *evD*: $\alpha * s v = cA * v v v \neq 0$
<proof>

lemma *ev-alpha-norm-v*: $\text{norm-v } (\alpha * s v) = (\text{norm } \alpha * s \text{norm-v } v)$
<proof>

lemma *ev-A-norm-v*: $\text{norm-v } (cA * v v) \$ j \leq (A * v \text{norm-v } v) \$ j$
<proof>

lemma *ev-le-vec*: *le-vec* $(\text{norm } \alpha * s \text{norm-v } v) (A * v \text{norm-v } v)$
<proof>

lemma *norm-v-X*: $\text{norm-v } v \in X$
<proof>

lemma *ev-inequalities*: $\text{norm } \alpha \leq r (\text{norm-v } v) \ r (\text{norm-v } v) \leq sr$

<proof>

lemma *eigen-vector-norm-sr: norm $\alpha \leq sr$* *<proof>*
end

lemma *eigen-value-norm-sr: assumes eigen-value $cA \alpha$*
shows *norm $\alpha \leq sr$*
<proof>

lemma *le-vec-trans: le-vec $x y \implies le-vec y u \implies le-vec x u$*
<proof>

lemma *eigen-vector-z-sr-c: eigen-vector $cA (map-vector c z) (c sr)$*
<proof>

lemma *eigen-value-sr-c: eigen-value $cA (c sr)$*
<proof>

definition *w = perron-frobenius.z (transpose A)*

lemma *w: transpose A *v w = sr *s w lt-vec 0 w perron-frobenius.sr (transpose A) = sr*
<proof>

lemma *c-cmod-id: a $\in \mathbb{R} \implies Re a \geq 0 \implies c (cmod a) = a$* *<proof>*

lemma *pos-rowvector-mult-0: assumes lt: lt-vec 0 x*
and *0: (rowvector $x :: real ^ 'n ^ 'n$) *v $y = 0$ (is ?x *v - = 0) and le: le-vec 0 y*
shows *y = 0*
<proof>

lemma *pos-matrix-mult-0: assumes le: $\bigwedge i j. B \$ i \$ j \geq 0$*
and *lt: lt-vec 0 x*
and *0: B *v x = 0*
shows *B = 0*
<proof>

lemma *eigen-value-smaller-matrix: assumes B: $\bigwedge i j. 0 \leq B \$ i \$ j \wedge B \$ i \$ j \leq A \$ i \$ j$*
and *AB: $A \neq B$*
and *ev: eigen-value (map-matrix c B) sigma*
shows *cmod sigma < sr*
<proof>

lemma *charpoly-erase-mat-sr: 0 < poly (charpoly (erase-mat A i i)) sr*
<proof>

lemma *multiplicity-sr-1*: $\text{order } sr \text{ (charpoly } A) = 1$
 ⟨proof⟩

lemma *sr-spectral-radius*: $sr = \text{spectral-radius } cA$
 ⟨proof⟩

lemma *le-vec-A-mu*: **assumes** $y: y \in X$ **and** $le: le\text{-vec } (A *v y) (mu *s y)$
shows $sr \leq mu$ *lt-vec* $0 y$
 $mu = sr \vee A *v y = mu *s y \implies mu = sr \wedge A *v y = mu *s y$
 ⟨proof⟩

lemma *nonnegative-eigenvector-has-ev-sr*: **assumes** *eigen-vector* $A v mu$ **and** $le: le\text{-vec } 0 v$
shows $mu = sr$
 ⟨proof⟩

lemma *similar-matrix-rotation*: **assumes** *eigen-value* $cA \alpha$ **and** $\alpha: cmod \alpha = sr$
shows *similar-matrix* $(cis (Arg \alpha) *k cA) cA$
 ⟨proof⟩

lemma **assumes** *eigen-value* $cA \alpha$ **and** $\alpha: cmod \alpha = sr$
shows *maximal-eigen-value-order-1*: $\text{order } \alpha \text{ (charpoly } cA) = 1$
and *maximal-eigen-value-rotation*: *eigen-value* $cA (x * cis (Arg \alpha)) = \text{eigen-value } cA x$
 $\text{eigen-value } cA (x / cis (Arg \alpha)) = \text{eigen-value } cA x$
 ⟨proof⟩

lemma *maximal-eigen-values-group*: **assumes** $M: M = \{ev :: \text{complex. eigen-value } cA \mid ev \wedge cmod ev = sr\}$
and $a: rcis sr \alpha \in M$
and $b: rcis sr \beta \in M$
shows $rcis sr (\alpha + \beta) \in M$ $rcis sr (\alpha - \beta) \in M$ $rcis sr 0 \in M$
 ⟨proof⟩

lemma *maximal-eigen-value-roots-of-unity-rotation*:
assumes $M: M = \{ev :: \text{complex. eigen-value } cA \mid ev \wedge cmod ev = sr\}$
and $kM: k = \text{card } M$
shows $k \neq 0$
 $k \leq \text{CARD}('n)$
 $\exists f. \text{charpoly } A = (\text{monom } 1 k - [:sr \wedge k:]) * f$
 $\wedge (\forall x. \text{poly } (map\text{-poly } c f) x = 0 \implies cmod x < sr)$
 $M = (*) (c sr) \text{ ' } (\lambda i. (cis (of\text{-nat } i * 2 * pi / k))) \text{ ' } \{0 .. k\}$
 $M = (*) (c sr) \text{ ' } \{x :: \text{complex. } x \wedge k = 1\}$
 $(*) (cis (2 * pi / k)) \text{ ' } \text{Spectrum } cA = \text{Spectrum } cA$
 ⟨proof⟩

lemmas *pf-main* =
eigen-value-sr eigen-vector-z-sr

```

eigen-value-norm-sr
z-pos
multiplicity-sr-1
nonnegative-eigenvector-has-ev-sr
maximal-eigen-value-order-1
maximal-eigen-value-roots-of-unity-rotation

```

```

lemmas pf-main-connect = pf-main(1,3,5,7,8-10)[unfolded sr-spectral-radius]
sr-pos[unfolded sr-spectral-radius]
end

end

```

5.2 Handling Non-Irreducible Matrices as Well

```

theory Perron-Frobenius-General
imports Perron-Frobenius-Irreducible
begin

```

We will need to take sub-matrices and permutations of matrices where the former can best be done via JNF-matrices. So, we first need the Perron-Frobenius theorem in the JNF-world. So, we first define irreducibility of a JNF-matrix.

```

definition graph-of-mat where
graph-of-mat  $A = (\text{let } n = \text{dim-row } A; U = \{..<n\} \text{ in } \{ ij. A \ \$\$ ij \neq 0\} \cap U \times U)$ 

```

```

definition irreducible-mat where
irreducible-mat  $A = (\text{let } n = \text{dim-row } A \text{ in } (\forall i j. i < n \longrightarrow j < n \longrightarrow (i,j) \in (\text{graph-of-mat } A) \hat{+}))$ 

```

```

definition nonneg-irreducible-mat  $A = (\text{nonneg-mat } A \wedge \text{irreducible-mat } A)$ 

```

Next, we have to install transfer rules

```

context
includes lifting-syntax
begin
lemma HMA-irreducible[transfer-rule]:  $((\text{HMA-M} :: - \Rightarrow - \hat{\ } 'n \hat{\ } 'n \Rightarrow -) \text{ ===> } (=))$ 
irreducible-mat fixed-mat.irreducible
<proof>

lemma HMA-nonneg-irreducible-mat[transfer-rule]:  $(\text{HMA-M} \text{ ===> } (=)) \text{ non-neg-irreducible-mat perron-frobenius}$ 
<proof>
end

```

The main statements of Perron-Frobenius can now be transferred to JNF-matrices

lemma *perron-frobenius-irreducible*: fixes $A :: \text{real Matrix.mat}$ and $cA :: \text{complex Matrix.mat}$

assumes $A: A \in \text{carrier-mat } n \ n$ and $n: n \neq 0$ and **nonneg**: *nonneg-mat* A
and *irr*: *irreducible-mat* A
and $cA: cA = \text{map-mat of-real } A$
and $sr: sr = \text{Spectral-Radius.spectral-radius } cA$

shows

eigenvalue $A \ sr$
order $sr \ (\text{char-poly } A) = 1$
 $0 < sr$
eigenvalue $cA \ \alpha \implies \text{cmod } \alpha \leq sr$
eigenvalue $cA \ \alpha \implies \text{cmod } \alpha = sr \implies \text{order } \alpha \ (\text{char-poly } cA) = 1$
 $\exists k \ f. k \neq 0 \wedge k \leq n \wedge \text{char-poly } A = (\text{monom } 1 \ k - [:sr \wedge k:]) * f \wedge$
 $(\forall x. \text{poly } (\text{map-poly complex-of-real } f) \ x = 0 \longrightarrow \text{cmod } x < sr)$

<proof>

We now need permutations on matrices to show that a matrix if a matrix is not irreducible, then it can be turned into a four-block-matrix by a permutation, where the lower left block is 0.

definition *permutation-mat* :: $\text{nat} \Rightarrow (\text{nat} \Rightarrow \text{nat}) \Rightarrow 'a :: \text{semiring-1 mat}$ **where**
permutation-mat $n \ p = \text{Matrix.mat } n \ n \ (\lambda \ (i,j). \ (\text{if } i = p \ j \ \text{then } 1 \ \text{else } 0))$

no-notation *m-inv* (*inv1* - [81] 80)

lemma *permutation-mat-dim[simp]*: *permutation-mat* $n \ p \in \text{carrier-mat } n \ n$
dim-row (*permutation-mat* $n \ p$) = n
dim-col (*permutation-mat* $n \ p$) = n
<proof>

lemma *permutation-mat-row[simp]*: p permutes $\{..<n\}$ $\implies i < n \implies$
Matrix.row (*permutation-mat* $n \ p$) $i = \text{unit-vec } n \ (\text{inv } p \ i)$
<proof>

lemma *permutation-mat-col[simp]*: p permutes $\{..<n\}$ $\implies i < n \implies$
Matrix.col (*permutation-mat* $n \ p$) $i = \text{unit-vec } n \ (p \ i)$
<proof>

lemma *permutation-mat-left*: **assumes** $A: A \in \text{carrier-mat } n \ nc$ and $p: p$ permutes $\{..<n\}$
shows *permutation-mat* $n \ p * A = \text{Matrix.mat } n \ nc \ (\lambda \ (i,j). \ A \ \$\$ \ (\text{inv } p \ i, \ j))$
<proof>

lemma *permutation-mat-right*: **assumes** $A: A \in \text{carrier-mat } nr \ n$ and $p: p$ permutes $\{..<n\}$
shows $A * \text{permutation-mat } n \ p = \text{Matrix.mat } nr \ n \ (\lambda \ (i,j). \ A \ \$\$ \ (i, \ p \ j))$
<proof>

lemma *permutes-lt*: p permutes $\{..<n\}$ $\implies i < n \implies p \ i < n$
<proof>

lemma *permutes-iff*: p permutes $\{..<n\}$ $\implies i < n \implies j < n \implies p\ i = p\ j \longleftrightarrow i = j$
 ⟨proof⟩

lemma *permutation-mat-id-1*: **assumes** p : p permutes $\{..<n\}$
shows $\text{permutation-mat } n\ p * \text{permutation-mat } n\ (\text{inv } p) = 1_m\ n$
 ⟨proof⟩

lemma *permutation-mat-id-2*: **assumes** p : p permutes $\{..<n\}$
shows $\text{permutation-mat } n\ (\text{inv } p) * \text{permutation-mat } n\ p = 1_m\ n$
 ⟨proof⟩

lemma *permutation-mat-both*: **assumes** A : $A \in \text{carrier-mat } n\ n$ **and** p : p permutes $\{..<n\}$
shows $\text{permutation-mat } n\ p * \text{Matrix.mat } n\ n\ (\lambda\ (i,j).\ A\ \$\$ (p\ i,\ p\ j)) * \text{permutation-mat } n\ (\text{inv } p) = A$
 ⟨proof⟩

lemma *permutation-similar-mat*: **assumes** A : $A \in \text{carrier-mat } n\ n$ **and** p : p permutes $\{..<n\}$
shows $\text{similar-mat } A\ (\text{Matrix.mat } n\ n\ (\lambda\ (i,j).\ A\ \$\$ (p\ i,\ p\ j)))$
 ⟨proof⟩

lemma *det-four-block-mat-lower-left-zero*: **fixes** $A1 :: 'a :: \text{idom mat}$
assumes $A1$: $A1 \in \text{carrier-mat } n\ n$
and $A2$: $A2 \in \text{carrier-mat } n\ m$ **and** $A3$: $A3 = 0_m\ m\ n$
and $A4$: $A4 \in \text{carrier-mat } m\ m$
shows $\text{Determinant.det } (\text{four-block-mat } A1\ A2\ A3\ A4) = \text{Determinant.det } A1 * \text{Determinant.det } A4$
 ⟨proof⟩

lemma *char-poly-matrix-four-block-mat*: **assumes**
 $A1$: $A1 \in \text{carrier-mat } n\ n$
and $A2$: $A2 \in \text{carrier-mat } n\ m$
and $A3$: $A3 \in \text{carrier-mat } m\ n$
and $A4$: $A4 \in \text{carrier-mat } m\ m$
shows $\text{char-poly-matrix } (\text{four-block-mat } A1\ A2\ A3\ A4) =$
 $\text{four-block-mat } (\text{char-poly-matrix } A1)\ (\text{map-mat } (\lambda\ x.\ [-x:])\ A2)$
 $(\text{map-mat } (\lambda\ x.\ [-x:])\ A3)\ (\text{char-poly-matrix } A4)$
 ⟨proof⟩

lemma *char-poly-four-block-mat-lower-left-zero*: **fixes** $A :: 'a :: \text{idom mat}$
assumes A : $A = \text{four-block-mat } B\ C\ (0_m\ m\ n)\ D$
and B : $B \in \text{carrier-mat } n\ n$
and C : $C \in \text{carrier-mat } n\ m$
and D : $D \in \text{carrier-mat } m\ m$
shows $\text{char-poly } A = \text{char-poly } B * \text{char-poly } D$
 ⟨proof⟩

lemma *elements-mat-permutes*: **assumes** p : p permutes $\{..< n\}$
and A : $A \in \text{carrier-mat } n \ n$
and B : $B = \text{Matrix.mat } n \ n \ (\lambda \ (i,j). \ A \ \$\$ \ (p \ i, \ p \ j))$
shows $\text{elements-mat } A = \text{elements-mat } B$
 $\langle \text{proof} \rangle$

lemma *elements-mat-four-block-mat-supseteq*:
assumes $A1$: $A1 \in \text{carrier-mat } n \ n$
and $A2$: $A2 \in \text{carrier-mat } n \ m$
and $A3$: $A3 \in \text{carrier-mat } m \ n$
and $A4$: $A4 \in \text{carrier-mat } m \ m$
shows $\text{elements-mat } (\text{four-block-mat } A1 \ A2 \ A3 \ A4) \supseteq$
 $(\text{elements-mat } A1 \cup \text{elements-mat } A2 \cup \text{elements-mat } A3 \cup \text{elements-mat } A4)$
 $\langle \text{proof} \rangle$

lemma *non-irreducible-mat-split*:
fixes $A :: 'a :: \text{idom mat}$
assumes A : $A \in \text{carrier-mat } n \ n$
and not : $\neg \text{irreducible-mat } A$
and n : $n > 1$
shows $\exists \ n1 \ n2 \ B \ B1 \ B2 \ B4. \ \text{similar-mat } A \ B \wedge \ \text{elements-mat } A = \text{elements-mat } B \wedge$
 $B = \text{four-block-mat } B1 \ B2 \ (0_m \ n2 \ n1) \ B4 \wedge$
 $B1 \in \text{carrier-mat } n1 \ n1 \wedge B2 \in \text{carrier-mat } n1 \ n2 \wedge B4 \in \text{carrier-mat } n2$
 $n2 \wedge$
 $0 < n1 \wedge n1 < n \wedge 0 < n2 \wedge n2 < n \wedge n1 + n2 = n$
 $\langle \text{proof} \rangle$

lemma *non-irreducible-nonneg-mat-split*:
fixes $A :: 'a :: \text{linordered-idom mat}$
assumes A : $A \in \text{carrier-mat } n \ n$
and nonneg : $\text{nonneg-mat } A$
and not : $\neg \text{irreducible-mat } A$
and n : $n > 1$
shows $\exists \ n1 \ n2 \ A1 \ A2. \ \text{char-poly } A = \text{char-poly } A1 * \text{char-poly } A2$
 $\wedge \ \text{nonneg-mat } A1 \wedge \ \text{nonneg-mat } A2$
 $\wedge \ A1 \in \text{carrier-mat } n1 \ n1 \wedge \ A2 \in \text{carrier-mat } n2 \ n2$
 $\wedge \ 0 < n1 \wedge n1 < n \wedge 0 < n2 \wedge n2 < n \wedge n1 + n2 = n$
 $\langle \text{proof} \rangle$

The main generalized theorem. The characteristic polynomial of a non-negative real matrix can be represented as a product of roots of unitys (scaled by the the spectral radius sr) and a polynomial where all roots are smaller than the spectral radius.

theorem *perron-frobenius-nonneg*: **fixes** $A :: \text{real Matrix.mat}$
assumes A : $A \in \text{carrier-mat } n \ n$ **and** pos : $\text{nonneg-mat } A$ **and** n : $n \neq 0$
shows $\exists \ sr \ ks \ f.$

$sr \geq 0 \wedge$
 $0 \notin \text{set } ks \wedge ks \neq [] \wedge$
 $\text{char-poly } A = \text{prod-list } (\text{map } (\lambda k. \text{monom } 1 k - [:sr \wedge k:]) ks) * f \wedge$
 $(\forall x. \text{poly } (\text{map-poly complex-of-real } f) x = 0 \longrightarrow \text{cmod } x < sr)$
 <proof>

And back to HMA world via transfer.

theorem *perron-frobenius-non-neg*: **fixes** $A :: \text{real } ^\wedge n ^\wedge n$

assumes $pos: \text{non-neg-mat } A$

shows $\exists sr ks f.$

$sr \geq 0 \wedge$

$0 \notin \text{set } ks \wedge ks \neq [] \wedge$

$\text{charpoly } A = \text{prod-list } (\text{map } (\lambda k. \text{monom } 1 k - [:sr \wedge k:]) ks) * f \wedge$

$(\forall x. \text{poly } (\text{map-poly complex-of-real } f) x = 0 \longrightarrow \text{cmod } x < sr)$

<proof>

We now specialize the theorem for complexity analysis where we are mainly interested in the case where the spectral radius is at most 1. Note that this can be checked by testing that there are no real roots of the characteristic polynomial which exceed 1.

Moreover, here the existential quantifier over the factorization is replaced by *decompose-prod-root-unity*, an algorithm which computes this factorization in an efficient way.

lemma *perron-frobenius-for-complexity*: **fixes** $A :: \text{real } ^\wedge n ^\wedge n$ **and** $f :: \text{real poly}$

defines $cA \equiv \text{map-matrix complex-of-real } A$

defines $cf \equiv \text{map-poly complex-of-real } f$

assumes $pos: \text{non-neg-mat } A$

and $sr: \bigwedge x. \text{poly } (\text{charpoly } A) x = 0 \implies x \leq 1$

and $\text{decomp}: \text{decompose-prod-root-unity } (\text{charpoly } A) = (ks, f)$

shows $0 \notin \text{set } ks$

$\text{charpoly } A = \text{prod-root-unity } ks * f$

$\text{charpoly } cA = \text{prod-root-unity } ks * cf$

$\bigwedge x. \text{poly } (\text{charpoly } cA) x = 0 \implies \text{cmod } x \leq 1$

$\bigwedge x. \text{poly } cf x = 0 \implies \text{cmod } x < 1$

$\bigwedge x. \text{cmod } x = 1 \implies \text{order } x (\text{charpoly } cA) = \text{length } [k \leftarrow ks . x \wedge k = 1]$

$\bigwedge x. \text{cmod } x = 1 \implies \text{poly } (\text{charpoly } cA) x = 0 \implies \exists k \in \text{set } ks. x \wedge k = 1$

<proof>

and convert to JNF-world

lemmas *perron-frobenius-for-complexity-jnf* =

perron-frobenius-for-complexity[*unfolded atomize-imp atomize-all,*

untransferred, cancel-card-constraint, rule-format]

end

6 Combining Spectral Radius Theory with Perron Frobenius theorem

theory *Spectral-Radius-Theory*

imports

Polynomial-Factorization.Square-Free-Factorization

Jordan-Normal-Form.Spectral-Radius

Jordan-Normal-Form.Char-Poly

Perron-Frobenius

HOL-Computational-Algebra.Field-as-Ring

begin

abbreviation *spectral-radius* **where** *spectral-radius* \equiv *Spectral-Radius.spectral-radius*

hide-const (open) *Module.smult*

Via JNFs it has been proven that the growth of A^k is polynomially bounded, if all complex eigenvalues have a norm at most 1, i.e., the spectral radius must be at most 1. Moreover, the degree of the polynomial growth can be bounded by the order of those roots which have norm 1, cf. $\llbracket ?A \in \text{carrier-mat } ?n \ ?n; \text{Spectral-Radius-Theory.spectral-radius } ?A \leq 1; \bigwedge \text{ev } k. \llbracket \text{poly } (\text{char-poly } ?A) \text{ ev} = 0; \text{cmod ev} = 1 \rrbracket \implies \text{order ev } (\text{char-poly } ?A) \leq ?d \rrbracket \implies \exists c1 \ c2. \forall k. \text{norm-bound } (?A \widehat{m} k) (c1 + c2 * (\text{real } k)^{?d - 1})$.

Perron Frobenius theorem tells us that for a real valued non negative matrix, the largest eigenvalue is a real non-negative one. Hence, we only have to check, that all real eigenvalues are at most one.

We combine both theorems in the following. To be more precise, the set-based complexity results from JNFs with the type-based Perron Frobenius theorem in HMA are connected to obtain a set based complexity criterion for real-valued non-negative matrices, where one only investigated the real valued eigenvalues for checking the eigenvalue-at-most-1 condition. Here, in the precondition of the roots of the polynomial, the type-system ensures that we only have to look at real-valued eigenvalues, and can ignore the complex-valued ones.

The linkage between set-and type-based is performed via HMA-connect.

lemma *perron-frobenius-spectral-radius-complex*: **fixes** $A :: \text{complex mat}$

assumes $A: A \in \text{carrier-mat } n \ n$

and *real-nonneg*: $\text{real-nonneg-mat } A$

and *ev-le-1*: $\bigwedge x. \text{poly } (\text{char-poly } (\text{map-mat } \text{Re } A)) \ x = 0 \implies x \leq 1$

and *ev-order*: $\bigwedge x. \text{norm } x = 1 \implies \text{order } x \ (\text{char-poly } A) \leq d$

shows $\exists c1 \ c2. \forall k. \text{norm-bound } (A \widehat{m} k) (c1 + c2 * \text{real } k \widehat{(d - 1)})$

<proof>

The following lemma is the same as $\llbracket ?A \in \text{carrier-mat } ?n \ ?n; \text{real-nonneg-mat } ?A; \bigwedge x. \text{poly } (\text{char-poly } (\text{map-mat } \text{Re } ?A)) \ x = 0 \implies x \leq 1; \bigwedge x. \text{cmod } x = 1 \implies \text{order } x \ (\text{char-poly } ?A) \leq ?d \rrbracket \implies \exists c1 \ c2. \forall k. \text{norm-bound } (?A \widehat{m} k) (c1 + c2 * (\text{real } k)^{?d - 1})$, except that now the type *real* is used instead of *complex*.

lemma *perron-frobenius-spectral-radius*: **fixes** $A :: \text{real mat}$
assumes $A: A \in \text{carrier-mat } n \ n$
and *nonneg*: $\text{nonneg-mat } A$
and *ev-le-1*: $\forall x. \text{poly } (\text{char-poly } A) \ x = 0 \longrightarrow x \leq 1$
and *ev-order*: $\forall x :: \text{complex. norm } x = 1 \longrightarrow \text{order } x \ (\text{map-poly of-real } (\text{char-poly } A)) \leq d$
shows $\exists c1 \ c2. \forall k \ a. a \in \text{elements-mat } (A \widehat{\ }_m \ k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{real } k \widehat{\ }^{(d-1)})$
 $\langle \text{proof} \rangle$

We can also convert the set-based lemma $\llbracket ?A \in \text{carrier-mat } ?n \ ?n; \text{nonneg-mat } ?A; \forall x. \text{poly } (\text{char-poly } ?A) \ x = 0 \longrightarrow x \leq 1; \forall x. \text{cmod } x = 1 \longrightarrow \text{order } x \ (\text{map-poly complex-of-real } (\text{char-poly } ?A)) \leq ?d \rrbracket \Longrightarrow \exists c1 \ c2. \forall k \ a. a \in \text{elements-mat } (?A \widehat{\ }_m \ k) \longrightarrow |a| \leq c1 + c2 * (\text{real } k)^{?d-1}$ to a type-based version.

lemma *perron-frobenius-spectral-type-based*:
assumes *non-neg-mat* $(A :: \text{real } \widehat{\ }^n \ \widehat{\ }^n)$
and $\forall x. \text{poly } (\text{charpoly } A) \ x = 0 \longrightarrow x \leq 1$
and $\forall x :: \text{complex. norm } x = 1 \longrightarrow \text{order } x \ (\text{map-poly of-real } (\text{charpoly } A)) \leq d$
shows $\exists c1 \ c2. \forall k \ a. a \in \text{elements-mat-h } (\text{matpow } A \ k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{real } k \widehat{\ }^{(d-1)})$
 $\langle \text{proof} \rangle$

And of course, we can also transfer the type-based lemma back to a set-based setting, only that – without further case-analysis – we get the additional assumption $n \neq 0$.

lemma **assumes** $A \in \text{carrier-mat } n \ n$
and *nonneg-mat* A
and $\forall x. \text{poly } (\text{char-poly } A) \ x = 0 \longrightarrow x \leq 1$
and $\forall x :: \text{complex. norm } x = 1 \longrightarrow \text{order } x \ (\text{map-poly of-real } (\text{char-poly } A)) \leq d$
and $n \neq 0$
shows $\exists c1 \ c2. \forall k \ a. a \in \text{elements-mat } (A \widehat{\ }_m \ k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{real } k \widehat{\ }^{(d-1)})$
 $\langle \text{proof} \rangle$

Note that the precondition eigenvalue-at-most-1 can easily be formulated as a cardinality constraints which can be decided by Sturm’s theorem. And in order to obtain a bound on the order, one can perform a square-free-factorization (via Yun’s factorization algorithm) of the characteristic polynomial into $f_1^1 \cdot \dots \cdot f_d^d$ where each f_i has precisely the roots of order i .

context

fixes $A :: \text{real mat}$ **and** $c :: \text{real}$ **and** fis **and** $n :: \text{nat}$
assumes $A: A \in \text{carrier-mat } n \ n$
and *nonneg*: $\text{nonneg-mat } A$
and *yun*: $\text{yun-factorization gcd } (\text{char-poly } A) = (c, fis)$
and *ev-le-1*: $\text{card } \{x. \text{poly } (\text{char-poly } A) \ x = 0 \wedge x > 1\} = 0$

begin

Note that *yun-factorization* has an offset by 1, so the pair $(f_i, i) \in \text{set } \text{fis}$ encodes $f_i^{\text{Suc } i}$.

lemma *perron-frobenius-spectral-radius-yun:*

assumes *bnd*: $\bigwedge f_i i. (f_i, i) \in \text{set } \text{fis}$

$\implies (\exists x :: \text{complex. poly } (\text{map-poly of-real } f_i) x = 0 \wedge \text{norm } x = 1)$

$\implies \text{Suc } i \leq d$

shows $\exists c1 c2. \forall k a. a \in \text{elements-mat } (A \hat{=}^m k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{real } k \wedge^{(d-1)})$

<proof>

Note that the only remaining problem in applying $(\bigwedge f_i i. \llbracket (f_i, i) \in \text{set } \text{fis}; \exists x. \text{poly } (\text{map-poly complex-of-real } f_i) x = 0 \wedge \text{cmod } x = 1 \rrbracket \implies \text{Suc } i \leq ?d) \implies \exists c1 c2. \forall k a. a \in \text{elements-mat } (A \hat{=}^m k) \longrightarrow |a| \leq c1 + c2 * (\text{real } k)^{?d-1}$ is to check the condition $\exists x. \text{poly } (\text{map-poly complex-of-real } f_i) x = 0 \wedge \text{cmod } x = 1$. Here, there are at least three possibilities. First, one can just ignore this precondition and weaken the statement. Second, one can apply Sturm's theorem to determine whether all roots are real. This can be done by comparing the number of distinct real roots with the degree of f_i , since f_i is square-free. If all roots are real, then one can decide the criterion by checking the only two possible real roots with norm equal to 1, namely 1 and -1. If on the other hand there are complex roots, then we loose precision at this point. Third, one uses a factorization algorithm (e.g., via complex algebraic numbers) to precisely determine the complex roots and decide the condition.

The second approach is illustrated in the following theorem. Note that all preconditions – including the ones from the context – can easily be checked with the help of Sturm's method. This method is used as a fast approximative technique in CeTA [3]. Only if the desired degree cannot be ensured by this method, the more costly complex algebraic number based factorization is applied.

lemma *perron-frobenius-spectral-radius-yun-real-roots:*

assumes *bnd*: $\bigwedge f_i i. (f_i, i) \in \text{set } \text{fis}$

$\implies \text{card } \{ x. \text{poly } f_i x = 0 \} \neq \text{degree } f_i \vee \text{poly } f_i 1 = 0 \vee \text{poly } f_i (-1) = 0$

$\implies \text{Suc } i \leq d$

shows $\exists c1 c2. \forall k a. a \in \text{elements-mat } (A \hat{=}^m k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{real } k \wedge^{(d-1)})$

<proof>

end

end

7 The Jordan Blocks of the Spectral Radius are Largest

Consider a non-negative real matrix, and consider any Jordan-block of any eigenvalues whose norm is the spectral radius. We prove that there is a Jordan block of the spectral radius which has the same size or is larger.

theory *Spectral-Radius-Largest-Jordan-Block*

imports

Jordan-Normal-Form.Jordan-Normal-Form-Uniqueness

Perron-Frobenius-General

HOL-Real-Asymp.Real-Asymp

begin

lemma *poly-asymp-equiv*: $(\lambda x. \text{poly } p \text{ (real } x)) \sim[\text{at-top}] (\lambda x. \text{lead-coeff } p * \text{real } x \wedge (\text{degree } p))$

<proof>

lemma *sum-root-unity*: **fixes** $x :: 'a :: \{\text{comm-ring}, \text{division-ring}\}$

assumes $x \wedge n = 1$

shows $\text{sum } (\lambda i. x \wedge i) \{.. < n\} = (\text{if } x = 1 \text{ then of-nat } n \text{ else } 0)$

<proof>

lemma *sum-root-unity-power-pos-implies-1*:

assumes *sumpos*: $\bigwedge k. \text{Re } (\text{sum } (\lambda i. b \ i * x \ i \wedge k) \ I) > 0$

and *root-unity*: $\bigwedge i. i \in I \implies \exists d. d \neq 0 \wedge x \ i \wedge d = 1$

shows $1 \in x \ 'I$

<proof>

fun *j-to-jb-index* :: $(\text{nat} \times 'a)\text{list} \Rightarrow \text{nat} \Rightarrow \text{nat} \times \text{nat}$ **where**

j-to-jb-index $((n, a) \# n\text{-as}) \ i = (\text{if } i < n \text{ then } (0, i) \text{ else}$

$\text{let } \text{rec} = \text{j-to-jb-index } n\text{-as } (i - n) \text{ in } (\text{Suc } (\text{fst } \text{rec}), \text{snd } \text{rec}))$

fun *jb-to-j-index* :: $(\text{nat} \times 'a)\text{list} \Rightarrow \text{nat} \times \text{nat} \Rightarrow \text{nat}$ **where**

jb-to-j-index $n\text{-as } (0, j) = j$

$| \text{jb-to-j-index } ((n, -) \# n\text{-as}) (\text{Suc } i, j) = n + \text{jb-to-j-index } n\text{-as } (i, j)$

lemma *j-to-jb-index*: **assumes** $i < \text{sum-list } (\text{map } \text{fst } n\text{-as})$

and $j < \text{sum-list } (\text{map } \text{fst } n\text{-as})$

and *j-to-jb-index* $n\text{-as } i = (b_i, l_i)$

and *j-to-jb-index* $n\text{-as } j = (b_j, l_j)$

and $n\text{-as} \ ! \ b_j = (n, a)$

shows $((\text{jordan-matrix } n\text{-as}) \wedge_m r) \ \S\S (i, j) = (\text{if } b_i = b_j \text{ then } ((\text{jordan-block } n \ a) \wedge_m r) \ \S\S (l_i, l_j) \text{ else } 0)$

$\wedge (b_i = b_j \implies l_i < n \wedge l_j < n \wedge b_j < \text{length } n\text{-as} \wedge (n, a) \in \text{set } n\text{-as})$

<proof>

lemma *j-to-jb-index-rev*: **assumes** $j: \text{j-to-jb-index } n\text{-as } i = (b_i, l_i)$

and $i: i < \text{sum-list } (\text{map } \text{fst } n\text{-as})$

and $k: k \leq li$
shows $li \leq i \wedge j\text{-to-jb-index } n\text{-as } (i - k) = (bi, li - k) \wedge ($
 $j\text{-to-jb-index } n\text{-as } j = (bi, li - k) \longrightarrow j < \text{sum-list } (\text{map fst } n\text{-as}) \longrightarrow j = i - k)$
 $\langle \text{proof} \rangle$

locale *spectral-radius-1-jnf-max* =
fixes $A :: \text{real mat}$ **and** $n m :: \text{nat}$ **and** $\text{lam} :: \text{complex}$ **and** $n\text{-as}$
assumes $A: A \in \text{carrier-mat } n \ n$
and $\text{nonneg}: \text{nonneg-mat } A$
and $\text{jnf}: \text{jordan-nf } (\text{map-mat complex-of-real } A) \ n\text{-as}$
and $\text{mem}: (m, \text{lam}) \in \text{set } n\text{-as}$
and $\text{lam1}: \text{cmod } \text{lam} = 1$
and $\text{sr1}: \bigwedge x. \text{poly } (\text{char-poly } A) \ x = 0 \implies x \leq 1$
and $\text{max-block}: \bigwedge k \ \text{la}. (k, \text{la}) \in \text{set } n\text{-as} \implies \text{cmod } \text{la} \leq 1 \wedge (\text{cmod } \text{la} = 1 \longrightarrow$
 $k \leq m)$
begin

lemma $n\text{-as0}: 0 \notin \text{fst } \text{'set } n\text{-as}$
 $\langle \text{proof} \rangle$

lemma $m0: m \neq 0 \langle \text{proof} \rangle$

abbreviation cA **where** $cA \equiv \text{map-mat complex-of-real } A$
abbreviation J **where** $J \equiv \text{jordan-matrix } n\text{-as}$

lemma $\text{sim-A-J}: \text{similar-mat } cA \ J$
 $\langle \text{proof} \rangle$

lemma $\text{sumlist-nf}: \text{sum-list } (\text{map fst } n\text{-as}) = n$
 $\langle \text{proof} \rangle$

definition $p :: \text{nat} \Rightarrow \text{real poly}$ **where**
 $p \ s = (\prod i = 0..<s. [: - \text{of-nat } i / \text{of-nat } (s - i), 1 / \text{of-nat } (s - i) :])$

lemma $p\text{-binom}: \text{assumes } sk: s \leq k$
shows $\text{of-nat } (k \ \text{choose } s) = \text{poly } (p \ s) \ (\text{of-nat } k)$
 $\langle \text{proof} \rangle$

lemma $p\text{-binom-complex}: \text{assumes } sk: s \leq k$
shows $\text{of-nat } (k \ \text{choose } s) = \text{complex-of-real } (\text{poly } (p \ s) \ (\text{of-nat } k))$
 $\langle \text{proof} \rangle$

lemma $\text{deg-p}: \text{degree } (p \ s) = s \langle \text{proof} \rangle$

lemma $\text{lead-coeff-p}: \text{lead-coeff } (p \ s) = (\prod i = 0..<s. 1 / (\text{of-nat } s - \text{of-nat } i))$
 $\langle \text{proof} \rangle$

lemma $\text{lead-coeff-p-gt-0}: \text{lead-coeff } (p \ s) > 0 \langle \text{proof} \rangle$

definition $c = \text{lead-coeff } (p \ (m - 1))$

lemma $c\text{-gt-0}$: $c > 0$ $\langle \text{proof} \rangle$

lemma $c0$: $c \neq 0$ $\langle \text{proof} \rangle$

definition PP **where** $PP = (\text{SOME } PP. \text{similar-mat-wit } cA \ J \ (\text{fst } PP) \ (\text{snd } PP))$

definition P **where** $P = \text{fst } PP$

definition iP **where** $iP = \text{snd } PP$

lemma JNF : $P \in \text{carrier-mat } n \ n \ iP \in \text{carrier-mat } n \ n \ J \in \text{carrier-mat } n \ n$

$P * iP = 1_m \ n \ iP * P = 1_m \ n \ cA = P * J * iP$

$\langle \text{proof} \rangle$

definition $C :: \text{nat set}$ **where**

$C = \{j \mid j \text{ bj } lj \ nn \ la. \ j < n \wedge j\text{-to-jb-index } n\text{-as } j = (bj, lj)$

$\wedge n\text{-as } ! \text{ bj} = (nn, la) \wedge c\text{mod } la = 1 \wedge nn = m \wedge lj = nn - 1\}$

lemma $C\text{-nonempty}$: $C \neq \{\}$

$\langle \text{proof} \rangle$

lemma $C\text{-n}$: $C \subseteq \{..<n\}$ $\langle \text{proof} \rangle$

lemma root-unity-cmod-1 : **assumes** $la: la \in \text{snd } \text{'set } n\text{-as}$ **and** $1: c\text{mod } la = 1$

shows $\exists d. d \neq 0 \wedge la \wedge^d = 1$

$\langle \text{proof} \rangle$

definition d **where** $d = (\text{SOME } d. \forall la. la \in \text{snd } \text{'set } n\text{-as} \longrightarrow c\text{mod } la = 1 \longrightarrow$

$d \ la \neq 0 \wedge la \wedge^{(d \ la)} = 1)$

lemma d : **assumes** $(k, la) \in \text{set } n\text{-as}$ $c\text{mod } la = 1$

shows $la \wedge^{(d \ la)} = 1 \wedge d \ la \neq 0$

$\langle \text{proof} \rangle$

definition D **where** $D = \text{prod-list } (\text{map } (\lambda na. \text{if } c\text{mod } (\text{snd } na) = 1 \text{ then } d \ (\text{snd } na) \text{ else } 1) \ n\text{-as})$

lemma $D0$: $D \neq 0$ $\langle \text{proof} \rangle$

definition f **where** $f \ \text{off } k = D * k + (m-1) + \text{off}$

lemma mono-f : $\text{strict-mono } (f \ \text{off})$ $\langle \text{proof} \rangle$

definition inv-op **where** $\text{inv-op } \text{off } k = \text{inverse } (c * \text{real } (f \ \text{off } k) \wedge^{(m-1)})$

lemma $\text{limit-jordan-block}$: **assumes** $kla: (k, la) \in \text{set } n\text{-as}$

and $ij: i < k \ j < k$

shows $(\lambda N. (\text{jordan-block } k \text{ la } \widehat{m} (f \text{ off } N)) \text{ $$ } (i, j) * \text{inv-op off } N)$
 \longrightarrow $(\text{if } i = 0 \wedge j = k - 1 \wedge \text{cmod } la = 1 \wedge k = m \text{ then } la \widehat{\text{off}} \text{ else } 0)$
 $\langle \text{proof} \rangle$

definition *lambda* **where** $\text{lambda } i = \text{snd } (n\text{-as } ! \text{fst } (j\text{-to-jb-index } n\text{-as } i))$

lemma *cmod-lambda*: $i \in C \implies \text{cmod } (\text{lambda } i) = 1$
 $\langle \text{proof} \rangle$

lemma *R-lambda*: **assumes** $i: i \in C$
shows $(m, \text{lambda } i) \in \text{set } n\text{-as}$
 $\langle \text{proof} \rangle$

lemma *limit-jordan-matrix*: **assumes** $ij: i < n \ j < n$
shows $(\lambda N. (J \widehat{m} (f \text{ off } N)) \text{ $$ } (i, j) * \text{inv-op off } N)$
 \longrightarrow $(\text{if } j \in C \wedge i = j - (m - 1) \text{ then } (\text{lambda } j) \widehat{\text{off}} \text{ else } 0)$
 $\langle \text{proof} \rangle$

declare *sumlist-nf*[*simp*]

lemma *A-power-P*: $cA \widehat{m} k * P = P * J \widehat{m} k$
 $\langle \text{proof} \rangle$

lemma *inv-op-nonneg*: $\text{inv-op off } k \geq 0 \langle \text{proof} \rangle$

lemma *P-nonzero-entry*: **assumes** $j: j < n$
shows $\exists i < n. P \text{ $$ } (i, j) \neq 0$
 $\langle \text{proof} \rangle$

definition *j* **where** $j = (\text{SOME } j. j \in C)$

lemma *j*: $j \in C \langle \text{proof} \rangle$

lemma *j-n*: $j < n \langle \text{proof} \rangle$

definition *i* = $(\text{SOME } i. i < n \wedge P \text{ $$ } (i, j - (m - 1)) \neq 0)$

lemma *i*: $i < n$ **and** *P-ij0*: $P \text{ $$ } (i, j - (m - 1)) \neq 0$
 $\langle \text{proof} \rangle$

definition *w* = $P *_v \text{unit-vec } n \ j$

lemma *w*: $w \in \text{carrier-vec } n \langle \text{proof} \rangle$

definition *v* = $\text{map-vec } \text{cmod } w$

lemma *v*: $v \in \text{carrier-vec } n \langle \text{proof} \rangle$

definition *u* **where** $u = iP *_v \text{map-vec of-real } v$

lemma *u*: $u \in \text{carrier-vec } n$ *<proof>*

definition *a where* $a\ j = P\ \$\$ (i, j - (m - 1)) * u\ \$v\ j$ **for** *j*

lemma *main-step*: $0 < \text{Re} (\sum_{j \in C}. a\ j * \text{lambda } j \wedge l)$
<proof>

lemma *main-theorem*: $(m, 1) \in \text{set } n\text{-as}$
<proof>
end

lemma *nonneg-sr-1-largest-jb*:
 assumes *nonneg*: *nonneg-mat* *A*
 and *jnf*: *jordan-nf* (*map-mat* *complex-of-real* *A*) *n-as*
 and *mem*: $(m, \text{lam}) \in \text{set } n\text{-as}$
 and *lam1*: $\text{cmod } \text{lam} = 1$
 and *sr1*: $\bigwedge x. \text{poly } (\text{char-poly } A)\ x = 0 \implies x \leq 1$
 shows $\exists M. M \geq m \wedge (M, 1) \in \text{set } n\text{-as}$
<proof>
hide-const(**open**) *spectral-radius*

lemma (**in** *ring-hom*) *hom-smult-mat*: $\text{mat}_h (a \cdot_m A) = \text{hom } a \cdot_m \text{mat}_h A$
<proof>

lemma *root-char-poly-smult*: **fixes** *A* :: *complex mat*
 assumes *A*: *A* $\in \text{carrier-mat } n\ n$
 and *k*: $k \neq 0$
shows $(\text{poly } (\text{char-poly } (k \cdot_m A))\ x = 0) = (\text{poly } (\text{char-poly } A)\ (x / k) = 0)$
<proof>

theorem *real-nonneg-mat-spectral-radius-largest-jordan-block*:
 assumes *real-nonneg-mat* *A*
 and *jordan-nf* *A* *n-as*
 and $(m, \text{lam}) \in \text{set } n\text{-as}$
 and $\text{cmod } \text{lam} = \text{spectral-radius } A$
shows $\exists M \geq m. (M, \text{of-real } (\text{spectral-radius } A)) \in \text{set } n\text{-as}$
<proof>

end

8 Homomorphisms of Gauss-Jordan Elimination, Kernel and More

theory *Hom-Gauss-Jordan*
 imports *Jordan-Normal-Form.Matrix-Kernel*
 Jordan-Normal-Form.Jordan-Normal-Form-Uniqueness

begin

lemma (in *comm-ring-hom*) *similar-mat-wit-hom*: **assumes**

similar-mat-wit A B C D

shows *similar-mat-wit (mat_h A) (mat_h B) (mat_h C) (mat_h D)*

<proof>

lemma (in *comm-ring-hom*) *similar-mat-hom*:

similar-mat A B \implies similar-mat (mat_h A) (mat_h B)

<proof>

context *field-hom*

begin

lemma *hom-swaprows*: *i < dim-row A \implies j < dim-row A \implies*

swaprows i j (mat_h A) = mat_h (swaprows i j A)

<proof>

lemma *hom-gauss-jordan-main*: *A \in carrier-mat nr nc \implies B \in carrier-mat nr nc2 \implies*

gauss-jordan-main (mat_h A) (mat_h B) i j =

map-prod mat_h mat_h (gauss-jordan-main A B i j)

<proof>

lemma *hom-gauss-jordan*: *A \in carrier-mat nr nc \implies B \in carrier-mat nr nc2 \implies*

gauss-jordan (mat_h A) (mat_h B) = map-prod mat_h mat_h (gauss-jordan A B)

<proof>

lemma *hom-gauss-jordan-single[simp]*: *gauss-jordan-single (mat_h A) = mat_h (gauss-jordan-single A)*

<proof>

lemma *hom-pivot-positions-main-gen*: **assumes** *A: A \in carrier-mat nr nc*

shows *pivot-positions-main-gen 0 (mat_h A) nr nc i j = pivot-positions-main-gen 0 A nr nc i j*

<proof>

lemma *hom-pivot-positions[simp]*: *pivot-positions (mat_h A) = pivot-positions A*

<proof>

lemma *hom-kernel-dim[simp]*: *kernel-dim (mat_h A) = kernel-dim A*

<proof>

lemma *hom-char-matrix*: **assumes** *A: A \in carrier-mat n n*

shows *char-matrix (mat_h A) (hom x) = mat_h (char-matrix A x)*

<proof>

lemma *hom-dim-gen-eigenspace*: **assumes** *A: A \in carrier-mat n n*

shows *dim-gen-eigenspace (mat_h A) (hom x) = dim-gen-eigenspace A x*

<proof>

end
end

9 Combining Spectral Radius Theory with Perron Frobenius theorem

theory *Spectral-Radius-Theory-2*

imports

Spectral-Radius-Largest-Jordan-Block

Hom-Gauss-Jordan

begin

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

lemma *jnf-complexity-generic*: **fixes** $A :: \text{complex mat}$

assumes $A: A \in \text{carrier-mat } n \ n$

and $sr: \bigwedge x. \text{poly } (\text{char-poly } A) \ x = 0 \implies \text{cmod } x \leq 1$

and $1: \bigwedge x. \text{poly } (\text{char-poly } A) \ x = 0 \implies \text{cmod } x = 1 \implies$

$\text{order } x \ (\text{char-poly } A) > d + 1 \implies$

$(\forall \text{ bsize} \in \text{fst } ' \text{set } (\text{compute-set-of-jordan-blocks } A \ x). \text{ bsize} \leq d + 1)$

shows $\exists c1 \ c2. \forall k. \text{norm-bound } (A \widehat{m} \ k) \ (c1 + c2 * \text{of-nat } k \widehat{d})$

<proof>

lemma *norm-bound-complex-to-real*: **fixes** $A :: \text{real mat}$

assumes $A: A \in \text{carrier-mat } n \ n$

and $bnd: \exists c1 \ c2. \forall k. \text{norm-bound } ((\text{map-mat } \text{complex-of-real } A) \widehat{m} \ k) \ (c1 + c2 * \text{of-nat } k \widehat{d})$

shows $\exists c1 \ c2. \forall k \ a. a \in \text{elements-mat } (A \widehat{m} \ k) \longrightarrow \text{abs } a \leq (c1 + c2 * \text{of-nat } k \widehat{d})$

<proof>

lemma *dim-gen-eigenspace-max-jordan-block*: **assumes** $\text{jnf}: \text{jordan-nf } A \ n\text{-as}$

shows $\text{dim-gen-eigenspace } A \ l \ d = \text{order } l \ (\text{char-poly } A) \longleftrightarrow$

$(\forall n. (n, l) \in \text{set } n\text{-as} \longrightarrow n \leq d)$

<proof>

lemma *jnf-complexity-1-complex*: **fixes** $A :: \text{complex mat}$

assumes $A: A \in \text{carrier-mat } n \ n$

and $\text{nonneg}: \text{real-nonneg-mat } A$

and $sr: \bigwedge x. \text{poly } (\text{char-poly } A) \ x = 0 \implies \text{cmod } x \leq 1$

and $1: \text{poly } (\text{char-poly } A) \ 1 = 0 \implies$

$\text{order } 1 \ (\text{char-poly } A) > d + 1 \implies$

$\text{dim-gen-eigenspace } A \ 1 \ (d+1) = \text{order } 1 \ (\text{char-poly } A)$

shows $\exists c1 \ c2. \forall k. \text{norm-bound } (A \widehat{m} \ k) \ (c1 + c2 * \text{of-nat } k \widehat{d})$

<proof>

lemma *jnf-complexity-1-real*: **fixes** $A :: \text{real mat}$

assumes $A: A \in \text{carrier-mat } n \ n$

```

and nonneg: nonneg-mat A
and sr:  $\bigwedge x. \text{poly}(\text{char-poly } A) x = 0 \implies x \leq 1$ 
and jb:  $\text{poly}(\text{char-poly } A) 1 = 0 \implies$ 
  order 1 (char-poly A) > d + 1  $\implies$ 
  dim-gen-eigenspace A 1 (d+1) = order 1 (char-poly A)
shows  $\exists c1\ c2. \forall k\ a. a \in \text{elements-mat } (A \widehat{m} k) \implies |a| \leq c1 + c2 * \text{real } k \widehat{d}$ 
<proof>
end

```

10 An efficient algorithm to compute the growth rate of A^n .

```

theory Check-Matrix-Growth
imports
  Spectral-Radius-Theory-2
  Sturm-Sequences.Sturm-Method
begin

```

```

hide-const (open) Coset.order

```

```

definition check-matrix-complexity :: real mat  $\Rightarrow$  real poly  $\Rightarrow$  nat  $\Rightarrow$  bool where
  check-matrix-complexity A cp d = (count-roots-above cp 1 = 0
   $\wedge$  (poly cp 1 = 0  $\implies$  (let ord = order 1 cp in
  d + 1 < ord  $\implies$  kernel-dim ((A - 1m (dim-row A))  $\widehat{m}$  (d + 1)) = ord)))

```

```

lemma check-matrix-complexity: assumes A: A  $\in$  carrier-mat n n and nn: non-
neg-mat A
and check: check-matrix-complexity A (char-poly A) d
shows  $\exists c1\ c2. \forall k\ a. a \in \text{elements-mat } (A \widehat{m} k) \implies \text{abs } a \leq (c1 + c2 * \text{of-nat } k \widehat{d})$ 
<proof>
end

```

Acknowledgements We thank Fabian Immler for an introduction to continuity proving using HMA.

References

- [1] O. Kunar and A. Popescu. From types to sets by local type definitions in higher-order logic. In *Proc. ITP 2016*. Springer, 2016. To appear.
- [2] D. Serre. *Matrices: Theory and Applications*. Graduate texts in mathematics. Springer, 2002.
- [3] R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proc. TPHOLs'09*, LNCS 5674, pages 452–468. Springer, 2009.

- [4] R. Thiemann and A. Yamada. Formalizing Jordan normal forms in Isabelle/HOL. In *Proc. CPP 2016*, pages 88–99. ACM, 2016.