

The Cayley-Hamilton theorem

Stephan Adelsberger Stefan Hetzl Florian Pollak

April 17, 2016

Abstract

This document contains a proof of the Cayley-Hamilton theorem based on the development of matrices in `HOL/Multivariate_Analysis`.

Contents

1	Introduction	1
----------	---------------------	----------

1 Introduction

The Cayley-Hamilton theorem states that every square matrix is a zero of its own characteristic polynomial, in symbols: $\chi_A(A) = 0$. It is a central theorem of linear algebra and plays an important role for matrix normal form theory.

In this document we work with matrices over a commutative ring R and give a direct algebraic proof of the theorem. The starting point of the proof is the following fundamental property of the adjugate matrix

$$\text{adj}(B) \cdot B = B \cdot \text{adj}(B) = \det(B)I_n \tag{1}$$

where I_n denotes the $n \times n$ -identity matrix and $\det(B)$ the determinant of B . Recall that the characteristic polynomial is defined as $\chi_A(X) = \det(XI_n - A)$, i.e. as the determinant of a matrix whose entries are polynomials. Considering the adjugate of this matrix we obtain

$$(XI_n - A) \cdot \text{adj}(XI_n - A) = \chi_A(X)I_n \tag{2}$$

directly from (1). Now, $\text{adj}(XI_n - A)$ being a matrix of polynomials of degree at most $n - 1$ can be written as

$$\text{adj}(XI_n - A) = \sum_{i=0}^{n-1} X^i B_i \text{ for } B_i \in R^{n \times n}. \tag{3}$$

A straightforward calculation starting from (2) using (3) then shows that

$$\chi_A(X)I_n = X^n B_{n-1} + \sum_{i=1}^{n-1} X^i (B_{i-1} - A \cdot B_i) - A \cdot B_0. \quad (4)$$

Now let c_i be the coefficient of X^i in $\chi_A(X)$. Then equating the coefficients in (4) yields

$$\begin{aligned} B_{n-1} &= I_n, \\ B_{i-1} - A \cdot B_i &= c_i I_n \text{ for } 1 \leq i \leq n-1, \text{ and} \\ -A \cdot B_0 &= c_0 I_n. \end{aligned}$$

Multiplying the i -th equation with A^i from the left gives

$$\begin{aligned} A^n \cdot B_{n-1} &= A^n, \\ A^i \cdot B_{i-1} - A^{i+1} \cdot B_i &= c_i A^i \text{ for } 1 \leq i \leq n-1, \text{ and} \\ -A \cdot B_0 &= c_0 I_n \end{aligned}$$

which shows that

$$\chi_A(A)I_n = A^n + c_{n-1}A^{n-1} + \cdots + c_1 A + c_0 I_n = 0$$

and hence $\chi_A(A) = 0$ which finishes this proof sketch.

There are numerous other proofs of the Cayley-Hamilton theorem, in particular the one formalized in Coq by Sidi Ould Biha [1, 2]. This proof also starts with the fundamental property of the adjugate matrix but instead of the above calculation relies on the existence of a ring isomorphism between $\mathcal{M}_n(R[X])$, the matrices of polynomials over R , and $(\mathcal{M}_n(R))[X]$, the polynomials whose coefficients are matrices over R . On the upside, this permits a briefer and more abstract argument (once the background theory contains all prerequisites) but on the downside one has to deal with the mathematically subtle evaluation of polynomials over the non-commutative(!) ring $\mathcal{M}_n(R)$. As described nicely in [2] this evaluation is no longer a ring homomorphism. However, its use in the proof of the Cayley-Hamilton theorem is sufficiently restricted so that one can work around this problem.

Sections ??, ??, and ?? contain basic results about matrices and polynomials which are needed for the proof of the Cayley-Hamilton theorem in addition to the results which are available in the library. Section ?? contains basic results about matrices of polynomials, including the definition of the characteristic polynomial and proofs of some of its basic properties. Finally, Section ?? contains the proof of the Cayley-Hamilton theorem as outlined above.

theory *Square-Matrix*

```

imports
  ~~/src/HOL/Multivariate-Analysis/Multivariate-Analysis
begin

lemma smult-axis:  $x * s \text{ axis } i \ y = \text{axis } i \ (x * y) :: \text{mult-zero}$ 
  <proof>

typedef ('a, 'n) sq-matrix = UNIV :: ('n  $\Rightarrow$  'n  $\Rightarrow$  'a) set
  morphisms to-fun of-fun
  <proof>

syntax -sq-matrix :: type  $\Rightarrow$  type  $\Rightarrow$  type ((-  $\wedge$  / -) [15, 16] 15)

  <ML>

setup-lifting type-definition-sq-matrix

lift-definition map-sq-matrix :: ('a  $\Rightarrow$  'c)  $\Rightarrow$  'a $^{**}$ 'b  $\Rightarrow$  'c $^{**}$ 'b is
   $\lambda f \ M \ i \ j. f \ (M \ i \ j)$  <proof>

lift-definition from-vec :: 'a $^{**}$ 'n $^{**}$ 'n  $\Rightarrow$  'a $^{**}$ 'n is
   $\lambda M \ i \ j. M \ \$ \ i \ \$ \ j$  <proof>

lift-definition to-vec :: 'a $^{**}$ 'n  $\Rightarrow$  'a $^{**}$ 'n $^{**}$ 'n is
   $\lambda M. \chi \ i \ j. M \ i \ j$  <proof>

lemma from-vec-eq-iff: from-vec  $M = \text{from-vec } N \iff M = N$ 
  <proof>

lemma to-vec-from-vec[simp]: to-vec (from-vec  $M$ ) =  $M$ 
  <proof>

lemma from-vec-to-vec[simp]: from-vec (to-vec  $M$ ) =  $M$ 
  <proof>

lemma map-sq-matrix-compose[simp]: map-sq-matrix  $f \ (\text{map-sq-matrix } g \ M) =$ 
map-sq-matrix ( $\lambda x. f \ (g \ x)$ )  $M$ 
  <proof>

lemma map-sq-matrix-ident[simp]: map-sq-matrix ( $\lambda x. x$ )  $M = M$ 
  <proof>

lemma map-sq-matrix-cong:
   $M = N \implies (\bigwedge i \ j. f \ (\text{to-fun } N \ i \ j) = g \ (\text{to-fun } N \ i \ j)) \implies \text{map-sq-matrix } f \ M$ 
  = map-sq-matrix  $g \ N$ 
  <proof>

lift-definition diag :: 'a $^{**}$ zero  $\Rightarrow$  'a $^{**}$ 'n is
   $\lambda k \ i \ j. \text{if } i = j \ \text{then } k \ \text{else } 0$  <proof>

```

lemma *diag-eq-iff*: $\text{diag } x = \text{diag } y \longleftrightarrow x = y$
<proof>

lemma *map-sq-matrix-diag[simp]*: $f \ 0 = 0 \implies \text{map-sq-matrix } f \ (\text{diag } c) = \text{diag } (f \ c)$
<proof>

lift-definition *smult-sq-matrix* :: $'a::\text{times} \Rightarrow 'a^{n'} \Rightarrow 'a^{n'}$ (**infixr** $*_S$ 75) **is**
 $\lambda c \ M \ i \ j. \ c * M \ i \ j$ *<proof>*

lemma *smult-map-sq-matrix*:
 $(\bigwedge y. f \ (x * y) = z * f \ y) \implies \text{map-sq-matrix } f \ (x *_S A) = z *_S \text{map-sq-matrix } f \ A$
<proof>

lemma *map-sq-matrix-smult*: $c *_S \text{map-sq-matrix } f \ A = \text{map-sq-matrix } (\lambda x. \ c * f \ x) \ A$
<proof>

lemma *one-smult[simp]*: $(1::\text{monoid-mult}) *_S x = x$
<proof>

lemma *smult-diag*: $x *_S \text{diag } y = \text{diag } (x * y::\text{mult-zero})$
<proof>

instantiation *sq-matrix* :: $(\text{semigroup-add}, \text{finite}) \text{ semigroup-add}$
begin

lift-definition *plus-sq-matrix* :: $'a^{b'} \Rightarrow 'a^{b'}$ **is**
 $\lambda A \ B \ i \ j. \ A \ i \ j + B \ i \ j$ *<proof>*

instance
<proof>

end

lemma *map-sq-matrix-add*:
 $(\bigwedge a \ b. \ f \ (a + b) = f \ a + f \ b) \implies \text{map-sq-matrix } f \ (A + B) = \text{map-sq-matrix } f \ A + \text{map-sq-matrix } f \ B$
<proof>

lemma *add-map-sq-matrix*: $\text{map-sq-matrix } f \ A + \text{map-sq-matrix } g \ A = \text{map-sq-matrix } (\lambda x. \ f \ x + g \ x) \ A$
<proof>

instantiation *sq-matrix* :: $(\text{monoid-add}, \text{finite}) \text{ monoid-add}$
begin

lift-definition *zero-sq-matrix* :: 'a^^b is $\lambda i j. 0$ *<proof>*

instance
<proof>

end

lemma *diag-0*: $diag\ 0 = 0$
<proof>

lemma *diag-0-eq*: $diag\ x = 0 \longleftrightarrow x = 0$
<proof>

lemma *zero-map-sq-matrix*: $f\ 0 = 0 \implies map\text{-}sq\text{-}matrix\ f\ 0 = 0$
<proof>

lemma *map-sq-matrix-0[simp]*: $map\text{-}sq\text{-}matrix\ (\lambda x. 0)\ A = 0$
<proof>

instance *sq-matrix* :: (ab-semigroup-add, finite) ab-semigroup-add
<proof>

instantiation *sq-matrix* :: (minus, finite) minus
begin

lift-definition *minus-sq-matrix* :: 'a^^b \Rightarrow 'a^^b \Rightarrow 'a^^b is
 $\lambda A\ B\ i\ j. A\ i\ j - B\ i\ j$ *<proof>*

instance *<proof>*
end

instantiation *sq-matrix* :: (group-add, finite) group-add
begin

lift-definition *uminus-sq-matrix* :: 'a^^b \Rightarrow 'a^^b is
uminus *<proof>*

instance
<proof>

end

lemma *map-sq-matrix-diff*:
 $(\bigwedge a\ b. f\ (a - b) = f\ a - f\ b) \implies map\text{-}sq\text{-}matrix\ f\ (A - B) = map\text{-}sq\text{-}matrix\ f\ A - map\text{-}sq\text{-}matrix\ f\ B$
<proof>

lemma *smult-diff*: **fixes** *a* :: 'a::comm-ring-1 **shows** $a *_{\mathcal{S}} (A - B) = a *_{\mathcal{S}} A - a *_{\mathcal{S}} B$

<proof>

instance *sq-matrix* :: (*cancel-semigroup-add*, *finite*) *cancel-semigroup-add*
<proof>

instance *sq-matrix* :: (*cancel-ab-semigroup-add*, *finite*) *cancel-ab-semigroup-add*
<proof>

instance *sq-matrix* :: (*comm-monoid-add*, *finite*) *comm-monoid-add*
<proof>

lemma *map-sq-matrix-setsum*:
 $f\ 0 = 0 \implies (\bigwedge a\ b. f\ (a + b) = f\ a + f\ b) \implies$
 $map\text{-}sq\text{-}matrix\ f\ (\sum\ i \in I. A\ i) = (\sum\ i \in I. map\text{-}sq\text{-}matrix\ f\ (A\ i))$
<proof>

lemma *setsum-map-sq-matrix*: $(\sum\ i \in I. map\text{-}sq\text{-}matrix\ (f\ i)\ A) = map\text{-}sq\text{-}matrix$
 $(\lambda x. \sum\ i \in I. f\ i\ x)\ A$
<proof>

lemma *smult-zero[simp]*: **fixes** $a :: 'a::ring-1$ **shows** $a *_{\mathcal{S}} 0 = 0$
<proof>

lemma *smult-right-add*: **fixes** $a :: 'a::ring-1$ **shows** $a *_{\mathcal{S}} (x + y) = a *_{\mathcal{S}} x + a$
 $*_{\mathcal{S}} y$
<proof>

lemma *smult-setsum*: **fixes** $a :: 'a::ring-1$ **shows** $(\sum\ i \in I. a *_{\mathcal{S}} f\ i) = a *_{\mathcal{S}} (setsum$
 $f\ I)$
<proof>

instance *sq-matrix* :: (*ab-group-add*, *finite*) *ab-group-add*
<proof>

instantiation *sq-matrix* :: (*semiring-0*, *finite*) *semiring-0*
begin

lift-definition *times-sq-matrix* :: $'a^{**}b \Rightarrow 'a^{**}b \Rightarrow 'a^{**}b$ **is**
 $\lambda M\ N\ i\ j. \sum\ k \in UNIV. M\ i\ k * N\ k\ j$ *<proof>*

instance
<proof>
end

lemma *diag-mult*: $diag\ x * A = x *_{\mathcal{S}} A$
<proof>

lemma *mult-diag*:
fixes $x :: 'a::comm-ring-1$

shows $A * \text{diag } x = x *_S A$
 ⟨proof⟩

lemma *smult-mult1*: **fixes** $a :: 'a::\text{comm-ring-1}$ **shows** $a *_S (A * B) = (a *_S A) * B$
 ⟨proof⟩

lemma *smult-mult2*: **fixes** $a :: 'a::\text{comm-ring-1}$ **shows** $a *_S (A * B) = A * (a *_S B)$
 ⟨proof⟩

lemma *map-sq-matrix-mult*:
fixes $f :: 'a::\text{semiring-1} \Rightarrow 'b::\text{semiring-1}$
assumes $f: \bigwedge a b. f (a + b) = f a + f b \bigwedge a b. f (a * b) = f a * f b f 0 = 0$
shows $\text{map-sq-matrix } f (A * B) = \text{map-sq-matrix } f A * \text{map-sq-matrix } f B$
 ⟨proof⟩

lemma *from-vec-mult[simp]*: $\text{from-vec } (M ** N) = \text{from-vec } M * \text{from-vec } N$
 ⟨proof⟩

instantiation *sq-matrix* :: $(\text{semiring-1}, \text{finite}) \text{ semiring-1}$
begin

lift-definition *one-sq-matrix* :: $'a^{''}b$ **is**
 $\lambda i j. \text{if } i = j \text{ then } 1 \text{ else } 0$ ⟨proof⟩

instance
 ⟨proof⟩
end

instance *sq-matrix* :: $(\text{semiring-1}, \text{finite}) \text{ numeral}$ ⟨proof⟩

lemma *diag-1*: $\text{diag } 1 = 1$
 ⟨proof⟩

lemma *diag-1-eq*: $\text{diag } x = 1 \iff x = 1$
 ⟨proof⟩

instance *sq-matrix* :: $(\text{ring-1}, \text{finite}) \text{ ring-1}$
 ⟨proof⟩

interpretation *sq-matrix*: *vector-space smult-sq-matrix*
 ⟨proof⟩

instantiation *sq-matrix* :: $(\text{real-vector}, \text{finite}) \text{ real-vector}$
begin

lift-definition *scaleR-sq-matrix* :: $\text{real} \Rightarrow 'a^{''}b \Rightarrow 'a^{''}b$ **is**
 $\lambda r A i j. r *_R A i j$ ⟨proof⟩

instance

<proof>

end

instance *sq-matrix* :: (semiring-1, finite) Rings.dvd *<proof>*

lift-definition *transpose* :: 'a^{'n} ⇒ 'a^{'n} **is**

$\lambda M\ i\ j. M\ j\ i$ *<proof>*

lemma *transpose-transpose[simp]*: *transpose (transpose A) = A*

<proof>

lemma *transpose-diag[simp]*: *transpose (diag c) = diag c*

<proof>

lemma *transpose-zero[simp]*: *transpose 0 = 0*

<proof>

lemma *transpose-one[simp]*: *transpose 1 = 1*

<proof>

lemma *transpose-add[simp]*: *transpose (A + B) = transpose A + transpose B*

<proof>

lemma *transpose-minus[simp]*: *transpose (A - B) = transpose A - transpose B*

<proof>

lemma *transpose-uminus[simp]*: *transpose (- A) = - transpose A*

<proof>

lemma *transpose-mult[simp]*:

*transpose (A * B :: 'a::comm-semiring-0^{'n}) = transpose B * transpose A*

<proof>

lift-definition *trace* :: 'a::comm-monoid-add^{'n} ⇒ 'a **is**

$\lambda M. \sum_{i \in UNIV}. M\ i\ i$ *<proof>*

lemma *trace-diag[simp]*: *trace (diag c :: 'a::semiring-1^{'n}) = of-nat CARD('n) * c*

<proof>

lemma *trace-0[simp]*: *trace 0 = 0*

<proof>

lemma *trace-1[simp]*: *trace (1 :: 'a::semiring-1^{'n}) = of-nat CARD('n)*

<proof>

lemma *trace-plus[simp]*: $\text{trace } (A + B) = \text{trace } A + \text{trace } B$
(*proof*)

lemma *trace-minus[simp]*: $\text{trace } (A - B) = (\text{trace } A - \text{trace } B)$ (*ab-group-add*)
(*proof*)

lemma *trace-uminus[simp]*: $\text{trace } (- A) = - (\text{trace } A)$ (*ab-group-add*)
(*proof*)

lemma *trace-smult[simp]*: $\text{trace } (s * A) = (s * \text{trace } A)$ (*semiring-0*)
(*proof*)

lemma *trace-transpose[simp]*: $\text{trace } (\text{transpose } A) = \text{trace } A$
(*proof*)

lemma *trace-mult-symm*:
fixes $A B :: 'a::\text{comm-semiring-0}^{n \times n}$
shows $\text{trace } (A * B) = \text{trace } (B * A)$
(*proof*)

lift-definition *det* :: $'a::\text{comm-ring-1}^{n \times n} \Rightarrow 'a$ **is**
 $\lambda A. (\sum p | p \text{ permutes } UNIV. \text{of-int } (\text{sign } p) * (\prod i \in UNIV. A i (p i)))$ (*proof*)

lemma *det-eq*: $\text{det } A = (\sum p | p \text{ permutes } UNIV. \text{of-int } (\text{sign } p) * (\prod i \in UNIV. \text{to-fun } A i (p i)))$
(*proof*)

lemma *permutes-UNIV-permutation*: $p \text{ permutes } UNIV \iff p \text{ permutes } (UNIV)$ (*finite*)
(*proof*)

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

lemma *det-transpose*: $\text{det } (\text{transpose } A) = \text{det } A$
(*proof*)

lemma *det-diagonal*:
fixes $A :: 'a::\text{comm-ring-1}^{n \times n}$
shows $(\bigwedge i j. i \neq j \implies \text{to-fun } A i j = 0) \implies \text{det } A = (\prod i \in UNIV. \text{to-fun } A i i)$
(*proof*)

lemma *det-1[simp]*: $\text{det } (1 :: 'a::\text{comm-ring-1}^{n \times n}) = 1$
(*proof*)

lemma *det-lowerdiagonal*:
fixes $A :: 'a::\text{comm-ring-1}^{n \times n} :: \{\text{finite, wellorder}\}$
shows $(\bigwedge i j. i < j \implies \text{to-fun } A i j = 0) \implies \text{det } A = (\prod i \in UNIV. \text{to-fun } A i i)$

<proof>

lemma *det-upperdiagonal*:

fixes $A :: 'a::comm-ring-1^{n::\{finite, wellorder\}}$

shows $(\bigwedge i j. j < i \implies to_fun\ A\ i\ j = 0) \implies det\ A = (\prod_{i \in UNIV}. to_fun\ A\ i\ i)$

<proof>

lift-definition *perm-rows* :: $'a^{b \Rightarrow 'b} \Rightarrow ('b \Rightarrow 'b) \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ p\ i\ j. M\ (p\ i)\ j$ *<proof>*

lift-definition *perm-cols* :: $'a^{b \Rightarrow 'b} \Rightarrow ('b \Rightarrow 'b) \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ p\ i\ j. M\ i\ (p\ j)$ *<proof>*

lift-definition *upd-rows* :: $'a^{b \Rightarrow 'b\ set} \Rightarrow ('b \Rightarrow 'a^{b \Rightarrow 'b}) \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ S\ v\ i\ j. if\ i \in S\ then\ v\ i\ \$\ j\ else\ M\ i\ j$ *<proof>*

lift-definition *upd-cols* :: $'a^{b \Rightarrow 'b\ set} \Rightarrow ('b \Rightarrow 'a^{b \Rightarrow 'b}) \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ S\ v\ i\ j. if\ j \in S\ then\ v\ j\ \$\ i\ else\ M\ i\ j$ *<proof>*

lift-definition *upd-row* :: $'a^{b \Rightarrow 'b} \Rightarrow 'b \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ i'\ v\ i\ j. if\ i = i'\ then\ v\ \$\ j\ else\ M\ i\ j$ *<proof>*

lift-definition *upd-col* :: $'a^{b \Rightarrow 'b} \Rightarrow 'b \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ j'\ v\ i\ j. if\ j = j'\ then\ v\ \$\ i\ else\ M\ i\ j$ *<proof>*

lift-definition *row* :: $'a^{b \Rightarrow 'b} \Rightarrow 'b \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ i. \chi\ j. M\ i\ j$ *<proof>*

lift-definition *col* :: $'a^{b \Rightarrow 'b} \Rightarrow 'b \Rightarrow 'a^{b \Rightarrow 'b}$ **is**

$\lambda M\ j. \chi\ i. M\ i\ j$ *<proof>*

lemma *perm-rows-transpose*: $perm_rows\ (transpose\ M)\ p = transpose\ (perm_cols\ M\ p)$

<proof>

lemma *perm-cols-transpose*: $perm_cols\ (transpose\ M)\ p = transpose\ (perm_rows\ M\ p)$

<proof>

lemma *upd-row-transpose*: $upd_row\ (transpose\ M)\ i\ p = transpose\ (upd_col\ M\ i\ p)$

<proof>

lemma *upd-col-transpose*: $upd_col\ (transpose\ M)\ i\ p = transpose\ (upd_row\ M\ i\ p)$

<proof>

lemma *upd-rows-transpose*: $upd_rows\ (transpose\ M)\ i\ p = transpose\ (upd_cols\ M\ i\ p)$

<proof>

lemma *upd-cols-transpose*: $\text{upd-cols } (\text{transpose } M) \ i \ p = \text{transpose } (\text{upd-rows } M \ i \ p)$
 ⟨proof⟩

lemma *upd-rows-empty[simp]*: $\text{upd-rows } M \ \{\} \ f = M$
 ⟨proof⟩

lemma *upd-cols-empty[simp]*: $\text{upd-cols } M \ \{\} \ f = M$
 ⟨proof⟩

lemma *upd-rows-single[simp]*: $\text{upd-rows } M \ \{i\} \ f = \text{upd-row } M \ i \ (f \ i)$
 ⟨proof⟩

lemma *upd-cols-single[simp]*: $\text{upd-cols } M \ \{i\} \ f = \text{upd-col } M \ i \ (f \ i)$
 ⟨proof⟩

lemma *upd-rows-insert*: $\text{upd-rows } M \ (\text{insert } i \ I) \ f = \text{upd-row } (\text{upd-rows } M \ I \ f) \ i \ (f \ i)$
 ⟨proof⟩

lemma *upd-rows-insert-rev*: $\text{upd-rows } M \ (\text{insert } i \ I) \ f = \text{upd-rows } (\text{upd-row } M \ i \ (f \ i)) \ I \ f$
 ⟨proof⟩

lemma *upd-rows-upd-row-swap*: $i \notin I \implies \text{upd-rows } (\text{upd-row } M \ i \ x) \ I \ f = \text{upd-row } (\text{upd-rows } M \ I \ f) \ i \ x$
 ⟨proof⟩

lemma *upd-cols-insert*: $\text{upd-cols } M \ (\text{insert } i \ I) \ f = \text{upd-col } (\text{upd-cols } M \ I \ f) \ i \ (f \ i)$
 ⟨proof⟩

lemma *upd-cols-insert-rev*: $\text{upd-cols } M \ (\text{insert } i \ I) \ f = \text{upd-cols } (\text{upd-col } M \ i \ (f \ i)) \ I \ f$
 ⟨proof⟩

lemma *upd-cols-upd-col-swap*: $i \notin I \implies \text{upd-cols } (\text{upd-col } M \ i \ x) \ I \ f = \text{upd-col } (\text{upd-cols } M \ I \ f) \ i \ x$
 ⟨proof⟩

lemma *upd-rows-cong[cong]*:
 $M = N \implies T = S \implies (\bigwedge s. s \in S = \text{simp} \implies f \ s = g \ s) \implies \text{upd-rows } M \ T \ f = \text{upd-rows } N \ S \ g$
 ⟨proof⟩

lemma *upd-cols-cong[cong]*:
 $M = N \implies T = S \implies (\bigwedge s. s \in S = \text{simp} \implies f \ s = g \ s) \implies \text{upd-cols } M \ T \ f = \text{upd-cols } N \ S \ g$

<proof>

lemma *row-upd-row-If*: $\text{row } (\text{upd-row } M \ i \ x) \ j = (\text{if } i = j \text{ then } x \text{ else row } M \ j)$
<proof>

lemma *row-upd-row[simp]*: $\text{row } (\text{upd-row } M \ i \ x) \ i = x$
<proof>

lemma *col-upd-col-If*: $\text{col } (\text{upd-col } M \ i \ x) \ j = (\text{if } i = j \text{ then } x \text{ else col } M \ j)$
<proof>

lemma *col-upd-col[simp]*: $\text{col } (\text{upd-col } M \ i \ x) \ i = x$
<proof>

lemma *upd-row-row[simp]*: $\text{upd-row } M \ i \ (\text{row } M \ i) = M$
<proof>

lemma *upd-row-upd-row-cancel[simp]*: $\text{upd-row } (\text{upd-row } M \ i \ x) \ i \ y = \text{upd-row } M \ i \ y$
<proof>

lemma *upd-col-upd-col-cancel[simp]*: $\text{upd-col } (\text{upd-col } M \ i \ x) \ i \ y = \text{upd-col } M \ i \ y$
<proof>

lemma *upd-col-col[simp]*: $\text{upd-col } M \ i \ (\text{col } M \ i) = M$
<proof>

lemma *row-transpose*: $\text{row } (\text{transpose } M) \ i = \text{col } M \ i$
<proof>

lemma *col-transpose*: $\text{col } (\text{transpose } M) \ i = \text{row } M \ i$
<proof>

lemma *det-perm-cols*:

fixes $A :: 'a::\text{comm-ring-1}^{n \times n}$

assumes $p: p \text{ permutes } UNIV$

shows $\det (\text{perm-cols } A \ p) = \text{of-int } (\text{sign } p) * \det A$

<proof>

lemma *det-perm-rows*:

fixes $A :: 'a::\text{comm-ring-1}^{n \times n}$

assumes $p: p \text{ permutes } UNIV$

shows $\det (\text{perm-rows } A \ p) = \text{of-int } (\text{sign } p) * \det A$

<proof>

lemma *det-row-add*: $\det (\text{upd-row } M \ i \ (a + b)) = \det (\text{upd-row } M \ i \ a) + \det (\text{upd-row } M \ i \ b)$

<proof>

lemma *det-row-mul*: $\det (\text{upd-row } M \ i \ (c * s \ a)) = c * \det (\text{upd-row } M \ i \ a)$
 ⟨proof⟩

lemma *det-row-uminus*: $\det (\text{upd-row } M \ i \ (- \ a)) = - \det (\text{upd-row } M \ i \ a)$
 ⟨proof⟩

lemma *det-row-minus*: $\det (\text{upd-row } M \ i \ (a - b)) = \det (\text{upd-row } M \ i \ a) - \det (\text{upd-row } M \ i \ b)$
 ⟨proof⟩

lemma *det-row-0*: $\det (\text{upd-row } M \ i \ 0) = 0$
 ⟨proof⟩

lemma *det-row-setsum*: $\det (\text{upd-row } M \ i \ (\sum_{s \in S} a \ s)) = (\sum_{s \in S} \det (\text{upd-row } M \ i \ (a \ s)))$
 ⟨proof⟩

lemma *det-col-add*: $\det (\text{upd-col } M \ i \ (a + b)) = \det (\text{upd-col } M \ i \ a) + \det (\text{upd-col } M \ i \ b)$
 ⟨proof⟩

lemma *det-col-mul*: $\det (\text{upd-col } M \ i \ (c * s \ a)) = c * \det (\text{upd-col } M \ i \ a)$
 ⟨proof⟩

lemma *det-col-uminus*: $\det (\text{upd-col } M \ i \ (- \ a)) = - \det (\text{upd-col } M \ i \ a)$
 ⟨proof⟩

lemma *det-col-minus*: $\det (\text{upd-col } M \ i \ (a - b)) = \det (\text{upd-col } M \ i \ a) - \det (\text{upd-col } M \ i \ b)$
 ⟨proof⟩

lemma *det-col-0*: $\det (\text{upd-col } M \ i \ 0) = 0$
 ⟨proof⟩

lemma *det-col-setsum*: $\det (\text{upd-col } M \ i \ (\sum_{s \in S} a \ s)) = (\sum_{s \in S} \det (\text{upd-col } M \ i \ (a \ s)))$
 ⟨proof⟩

lemma *det-identical-cols*:

assumes $i \neq i'$ **shows** $\text{col } A \ i = \text{col } A \ i' \implies \det A = 0$
 ⟨proof⟩

lemma *det-identical-rows*: $i \neq i' \implies \text{row } A \ i = \text{row } A \ i' \implies \det A = 0$
 ⟨proof⟩

lemma *det-cols-setsum*:

$\det (\text{upd-cols } M \ T \ (\lambda i. \sum_{s \in S} a \ i \ s)) = (\sum_{f \in T \rightarrow_E S} \det (\text{upd-cols } M \ T \ (\lambda i. a \ i \ (f \ i))))$

<proof>

lemma *det-rows-setsum:*

$\det (\text{upd-rows } M \ T \ (\lambda i. \sum_{s \in S}. a \ i \ s)) = (\sum_{f \in T \rightarrow_E S}. \det (\text{upd-rows } M \ T \ (\lambda i. a \ i \ (f \ i))))$

<proof>

lemma *det-rows-mult:* $\det (\text{upd-rows } M \ T \ (\lambda i. c \ i \ *s \ a \ i)) = (\prod_{i \in T}. c \ i) * \det (\text{upd-rows } M \ T \ a)$

<proof>

lemma *det-cols-mult:* $\det (\text{upd-cols } M \ T \ (\lambda i. c \ i \ *s \ a \ i)) = (\prod_{i \in T}. c \ i) * \det (\text{upd-cols } M \ T \ a)$

<proof>

lemma *det-perm-rows-If:* $\det (\text{perm-rows } B \ f) = (\text{if } f \text{ permutes } UNIV \text{ then of-int } (\text{sign } f) * \det B \text{ else } 0)$

<proof>

lemma *det-mult:* $\det (A * B) = \det A * \det B$

<proof>

lift-definition *minor* :: $'a \ \wedge \wedge \ 'b \Rightarrow \ 'b \Rightarrow \ 'b \Rightarrow \ 'a::\text{semiring-1} \ \wedge \wedge \ 'b$ is

$\lambda A \ i \ j \ k \ l. \text{if } k = i \wedge l = j \text{ then } 1 \text{ else if } k = i \vee l = j \text{ then } 0 \text{ else } A \ k \ l$ *<proof>*

lemma *minor-transpose:* $\text{minor } (\text{transpose } A) \ i \ j = \text{transpose } (\text{minor } A \ j \ i)$

<proof>

lemma *minor-eq-row-col:* $\text{minor } M \ i \ j = \text{upd-row } (\text{upd-col } M \ j \ (\text{axis } i \ 1)) \ i \ (\text{axis } j \ 1)$

<proof>

lemma *minor-eq-col-row:* $\text{minor } M \ i \ j = \text{upd-col } (\text{upd-row } M \ i \ (\text{axis } j \ 1)) \ j \ (\text{axis } i \ 1)$

<proof>

lemma *row-minor:* $\text{row } (\text{minor } M \ i \ j) \ i = \text{axis } j \ 1$

<proof>

lemma *col-minor:* $\text{col } (\text{minor } M \ i \ j) \ j = \text{axis } i \ 1$

<proof>

lemma *det-minor-row':*

$\text{row } B \ i = \text{axis } j \ 1 \implies \det (\text{minor } B \ i \ j) = \det B$

<proof>

lemma *det-minor-row:* $\det (\text{minor } B \ i \ j) = \det (\text{upd-row } B \ i \ (\text{axis } j \ 1))$

<proof>

lemma *det-minor-col*: $\det (\text{minor } B \ i \ j) = \det (\text{upd-col } B \ j \ (\text{axis } i \ 1))$
 ⟨proof⟩

lift-definition *cofactor* :: 'a[^]'b ⇒ 'a::comm-ring-1[^]'b is
 $\lambda A \ i \ j. \det (\text{minor } A \ i \ j)$ ⟨proof⟩

lemma *cofactor-transpose*: $\text{cofactor } (\text{transpose } A) = \text{transpose } (\text{cofactor } A)$
 ⟨proof⟩

definition *adjugate* $A = \text{transpose } (\text{cofactor } A)$

lemma *adjugate-transpose*: $\text{adjugate } (\text{transpose } A) = \text{transpose } (\text{adjugate } A)$
 ⟨proof⟩

theorem *adjugate-mult-det*: $\text{adjugate } A * A = \text{diag } (\det A)$
 ⟨proof⟩

lemma *mult-adjugate-det*: $A * \text{adjugate } A = \text{diag } (\det A)$
 ⟨proof⟩

end

⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩⟨proof⟩

theorem *Cayley-Hamilton*:

fixes $A :: 'a::\text{comm-ring-1}^{\wedge\wedge} n$

shows *poly-mat* (*charpoly* A) $A = 0$

⟨proof⟩

Part 1

def $n \equiv \text{CARD}(n) - 1$

then have *d-charpoly*: $n + 1 = \text{degree } (\text{charpoly } A)$ **and**

d-adj: $n = \text{max-degree } (\text{adjugate } (\mathbf{X} - \mathbf{C} A))$

def $B \equiv \lambda i. \text{map-sq-matrix } (\lambda p. \text{coeff } p \ i) (\text{adjugate } (\mathbf{X} - \mathbf{C} A))$

have *A-eq-B*: $\text{adjugate } (\mathbf{X} - \mathbf{C} A) = (\sum_{i \leq n}. X^i *_S \mathbf{C} (B \ i))$

Part 2

have *charpoly* $A *_S 1 = X *_S \text{adjugate } (\mathbf{X} - \mathbf{C} A) - \mathbf{C} A *_S \text{adjugate } (\mathbf{X} - \mathbf{C} A)$

also have $\dots = (\sum_{i \leq n}. X^{(i+1)} *_S \mathbf{C} (B \ i)) - (\sum_{i \leq n}. X^i *_S \mathbf{C} (A * B \ i))$

also have $(\sum_{i \leq n}. X^{(i+1)} *_S \mathbf{C} (B \ i)) =$

$(\sum_{i < n}. X^{(i+1)} *_S \mathbf{C} (B \ i)) + X^{(n+1)} *_S \mathbf{C} (B \ n)$

also have $(\sum_{i \leq n}. X^i *_S \mathbf{C} (A * B \ i)) =$

$(\sum_{i < n}. X^{(i+1)} *_S \mathbf{C} (A * B \ (i+1))) + \mathbf{C} (A * B \ 0)$

finally have *diag-charpoly*:

charpoly $A *_S 1 = X^{(n+1)} *_S \mathbf{C} (B \ n) +$

$(\sum_{i < n}. X^{(i+1)} *_S \mathbf{C} (B \ i - A * B \ (i+1))) - \mathbf{C} (A * B \ 0)$

```

let ?p = λi. coeff (charpoly A) i *S A ^ i
let ?AB = λi. A ^ (i + 1) * B i
have (∑ i ≤ n + 1. ?p i) = ?p 0 + (∑ i < n. ?p (i + 1)) + ?p (n + 1)
also have ?p 0 = - ?AB 0
also have (∑ i < n. ?p (i + 1)) = (∑ i = 0 .. < n. ?AB i - ?AB (i + 1))
also have ... = ?AB 0 - ?AB n
also have ?AB n = ?p (n + 1)
also have coeff (charpoly A) (n + 1) = 1
finally show ?thesis
qed

```

References

- [1] S. Ould Biha. Formalisation des mathématiques : une preuve du théorème de Cayley-Hamilton. In *JFLA (Journées Francophones des Langages Applicatifs)*, pages 1–14. INRIA, 2008. available at <http://hal.inria.fr/inria-00202795/PDF/ouldbiha.pdf>.
- [2] S. Ould Biha. *Composants mathématiques pour la théorie des groupes*. PhD thesis, Université de Nice – Sophia Antipolis, 2010. available at <http://hal.inria.fr/tel-00493524>.