The Cayley-Hamilton theorem

Stephan Adelsberger Stefan Hetzl Florian Pollak

March 17, 2025

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

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

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
HOL−Analysis.Determinants
HOL−Analysis.Cartesian-Euclidean-Space
begin

lemma *smult-axis*: $x *s axis\ i\ y = axis\ i\ (x * y$::-::*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* ⇒ *type* ⇒ *type* (‹(- ^^/ -)› [15, 16] 15)

syntax-types -*sq-matrix* ⇌ *sq-matrix*

⟨*ML*⟩

setup-lifting *type-definition-sq-matrix*

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

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

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

lemma *from-vec-eq-iff*: *from-vec* $M =$ *from-vec* $N \longleftrightarrow M = N$
⟨*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 (*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*:
$M = N \Longrightarrow (\bigwedge i\ j.\ f\ (to\text{-}fun\ N\ i\ j) = g\ (to\text{-}fun\ N\ i\ j)) \Longrightarrow map\text{-}sq\text{-}matrix\ f\ M =$ *map-sq-matrix* $g\ N$
⟨*proof*⟩

lift-definition *diag* :: *'a*::*zero* ⇒ *'a*^^*'n* **is**
λk i j. if i = j then k else 0 ⟨*proof*⟩

lemma *diag-eq-iff*: *diag x = diag y ⟷ x = y*
⟨*proof*⟩

lemma *map-sq-matrix-diag*[*simp*]: *f 0 = 0 ⟹ map-sq-matrix f (diag c) = diag (f c)*
⟨*proof*⟩

lift-definition *smult-sq-matrix* :: *'a*::*times* ⇒ *'a*^^*'n* ⇒ *'a*^^*'n* (**infixr** ‹∗$_S$› *75*) **is**
λc M i j. c ∗ M i j ⟨*proof*⟩

lemma *smult-map-sq-matrix*:
(⋀y. f (x ∗ y) = z ∗ f y) ⟹ map-sq-matrix f (x ∗$_S$ A) = z ∗$_S$ map-sq-matrix f A
⟨*proof*⟩

lemma *map-sq-matrix-smult*: *c ∗$_S$ map-sq-matrix f A = map-sq-matrix (λx. c ∗ f x) A*
⟨*proof*⟩

lemma *one-smult*[*simp*]: *(1*::*-*::*monoid-mult) ∗$_S$ x = x*
⟨*proof*⟩

lemma *smult-diag*: *x ∗$_S$ diag y = diag (x ∗ y*::*-*::*mult-zero)*
⟨*proof*⟩

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

lift-definition *plus-sq-matrix* :: *'a*^^*'b ⇒ 'a*^^*'b ⇒ 'a*^^*'b* **is**
λA B i j. A i j + B i j ⟨*proof*⟩

instance
⟨*proof*⟩

end

lemma *map-sq-matrix-add*:
(⋀a b. f (a + b) = f a + f b) ⟹ map-sq-matrix f (A + B) = map-sq-matrix f A + map-sq-matrix f B
⟨*proof*⟩

lemma *add-map-sq-matrix*: *map-sq-matrix f A + map-sq-matrix g A = map-sq-matrix (λx. f x + g x) A*
⟨*proof*⟩

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

lift-definition *zero-sq-matrix* :: $'a\,\widehat{}\,'b$ **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*: $f\ 0 = 0 \Longrightarrow map\text{-}sq\text{-}matrix\ f\ 0 = 0$
⟨*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* :: $'a\,\widehat{}\,'b \Rightarrow 'a\,\widehat{}\,'b \Rightarrow 'a\,\widehat{}\,'b$ **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* :: $'a\,\widehat{}\,'b \Rightarrow 'a\,\widehat{}\,'b$ **is**
uminus ⟨*proof*⟩

instance
⟨*proof*⟩

end

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

⟨*proof*⟩

lemma *smult-diff*: **fixes** $a :: 'a$::*comm-ring-1* **shows** $a *_S (A - B) = a *_S A - a *_S B$
⟨*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-sum*:
$f\ 0 = 0 \Longrightarrow (\bigwedge a\ b.\ f\ (a + b) = f\ a + f\ b) \Longrightarrow$
map-sq-matrix $f\ (\sum i{\in}I.\ A\ i) = (\sum i{\in}I.$ *map-sq-matrix* $f\ (A\ i))$
⟨*proof*⟩

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

lemma *smult-zero*[*simp*]: **fixes** $a :: 'a$::*ring-1* **shows** $a *_S 0 = 0$
⟨*proof*⟩

lemma *smult-right-add*: **fixes** $a :: 'a$::*ring-1* **shows** $a *_S (x + y) = a *_S x + a *_S y$
⟨*proof*⟩

lemma *smult-sum*: **fixes** $a :: 'a$::*ring-1* **shows** $(\sum i{\in}I.\ a *_S f\ i) = a *_S (sum\ f\ I)$
⟨*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* :: $'a^{\frown\frown}'b \Rightarrow 'a^{\frown\frown}'b \Rightarrow 'a^{\frown\frown}'b$ **is**
$\lambda M\ N\ i\ j.\ \sum k{\in}UNIV.\ M\ i\ k * N\ k\ j$ ⟨*proof*⟩

instance
⟨*proof*⟩
end

lemma *diag-mult*: *diag* $x * A = x *_S A$
⟨*proof*⟩

lemma *mult-diag*:
fixes x :: *'a*::*comm-ring-1*
shows $A * diag\ x = x *_S A$
⟨*proof*⟩

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

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

lemma *map-sq-matrix-mult*:
fixes f :: *'a*::*semiring-1* ⇒ *'b*::*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 *map-sq-matrix f* $(A * B)$ = *map-sq-matrix f A* * *map-sq-matrix f B*
⟨*proof*⟩

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

instantiation *sq-matrix* :: (*semiring-1*, *finite*) *semiring-1*
begin

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

instance
⟨*proof*⟩
end

instance *sq-matrix* :: (*semiring-1*, *finite*) *numeral* ⟨*proof*⟩

lemma *diag-1*: *diag 1 = 1*
⟨*proof*⟩

lemma *diag-1-eq*: *diag* $x = 1 \longleftrightarrow x = 1$
⟨*proof*⟩

instance *sq-matrix* :: (*ring-1*, *finite*) *ring-1*
⟨*proof*⟩

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

instantiation *sq-matrix* :: (*real-vector*, *finite*) *real-vector*
begin

lift-definition *scaleR-sq-matrix* :: *real* ⇒ *'a^~'b* ⇒ *'a^~'b* **is**
λr 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**
λM 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**
λM. ∑*i*∈*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*]: *trace* (*A* + *B*) = *trace A* + *trace B*
⟨*proof*⟩

lemma *trace-minus*[*simp*]: *trace* (*A* − *B*) = (*trace A* − *trace B*::-::*ab-group-add*)
⟨*proof*⟩

lemma *trace-uminus*[*simp*]: *trace* (− *A*) = − (*trace A*::-::*ab-group-add*)
⟨*proof*⟩

lemma *trace-smult*[*simp*]: *trace* (*s* ∗*S A*) = (*s* ∗ *trace A*::-::*semiring-0*)
⟨*proof*⟩

lemma *trace-transpose*[*simp*]: *trace* (*transpose A*) = *trace A*
⟨*proof*⟩

lemma *trace-mult-symm*:
fixes *A B* :: *'a*::*comm-semiring-0*^^*'n*
shows *trace* (*A* ∗ *B*) = *trace* (*B* ∗ *A*)
⟨*proof*⟩

lift-definition *det* :: *'a*::*comm-ring-1*^^*'n* ⇒ *'a* **is**
λ*A*. (∑ *p*|*p permutes UNIV*. *of-int* (*sign p*) ∗ (∏ *i*∈*UNIV*. *A i* (*p i*))) ⟨*proof*⟩

lemma *det-eq*: *det A* = (∑ *p*|*p permutes UNIV*. *of-int* (*sign p*) ∗ (∏ *i*∈*UNIV*. *to-fun A i* (*p i*)))
⟨*proof*⟩

lemma *permutes-UNIV-permutation*: *permutation p* ⟷ *p permutes* (*UNIV*::-::*finite*)
⟨*proof*⟩

lemma *det-0*[*simp*]: *det 0* = *0*
⟨*proof*⟩

lemma *det-transpose*: *det* (*transpose A*) = *det A*
⟨*proof*⟩

lemma *det-diagonal*:
fixes *A* :: *'a*::*comm-ring-1*^^*'n*
shows (⋀*i j*. *i* ≠ *j* ⟹ *to-fun A i j* = *0*) ⟹ *det A* = (∏ *i*∈*UNIV*. *to-fun A i i*)
⟨*proof*⟩

lemma *det-1*[*simp*]: *det* (*1*::*'a*::*comm-ring-1*^^*'n*) = *1*
⟨*proof*⟩

lemma *det-lowerdiagonal*:
fixes *A* :: *'a*::*comm-ring-1*^^*'n*::{*finite*,*wellorder*}

shows $(\bigwedge i\ j.\ i < j \Longrightarrow \textit{to-fun}\ A\ i\ j = 0) \Longrightarrow \textit{det}\ A = (\prod i{\in}\textit{UNIV}.\ \textit{to-fun}\ A\ i\ i)$
⟨*proof*⟩

lemma *det-upperdiagonal*:
fixes $A :: {'a}{::}\textit{comm-ring-1}^{\frown\frown}{'n}{::}\{\textit{finite}, \textit{wellorder}\}$
shows $(\bigwedge i\ j.\ j < i \Longrightarrow \textit{to-fun}\ A\ i\ j = 0) \Longrightarrow \textit{det}\ A = (\prod i{\in}\textit{UNIV}.\ \textit{to-fun}\ A\ i\ i)$
⟨*proof*⟩

lift-definition *perm-rows* $:: {'a}^{\frown\frown}{'b} \Rightarrow ({'b} \Rightarrow {'b}) \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ p\ i\ j.\ M\ (p\ i)\ j$ ⟨*proof*⟩

lift-definition *perm-cols* $:: {'a}^{\frown\frown}{'b} \Rightarrow ({'b} \Rightarrow {'b}) \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ p\ i\ j.\ M\ i\ (p\ j)$ ⟨*proof*⟩

lift-definition *upd-rows* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b}\ \textit{set} \Rightarrow ({'b} \Rightarrow {'a}^{\frown}{'b}) \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ S\ v\ i\ j.\ \textit{if}\ i \in S\ \textit{then}\ v\ i\ \$\ j\ \textit{else}\ M\ i\ j$ ⟨*proof*⟩

lift-definition *upd-cols* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b}\ \textit{set} \Rightarrow ({'b} \Rightarrow {'a}^{\frown}{'b}) \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ S\ v\ i\ j.\ \textit{if}\ j \in S\ \textit{then}\ v\ j\ \$\ i\ \textit{else}\ M\ i\ j$ ⟨*proof*⟩

lift-definition *upd-row* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b} \Rightarrow {'a}^{\frown}{'b} \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ i'\ v\ i\ j.\ \textit{if}\ i = i'\ \textit{then}\ v\ \$\ j\ \textit{else}\ M\ i\ j$ ⟨*proof*⟩

lift-definition *upd-col* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b} \Rightarrow {'a}^{\frown}{'b} \Rightarrow {'a}^{\frown\frown}{'b}$ **is**
$\lambda M\ j'\ v\ i\ j.\ \textit{if}\ j = j'\ \textit{then}\ v\ \$\ i\ \textit{else}\ M\ i\ j$ ⟨*proof*⟩

lift-definition *row* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b} \Rightarrow {'a}^{\frown}{'b}$ **is**
$\lambda M\ i.\ \chi\ j.\ M\ i\ j$ ⟨*proof*⟩

lift-definition *col* $:: {'a}^{\frown\frown}{'b} \Rightarrow {'b} \Rightarrow {'a}^{\frown}{'b}$ **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*: *upd-cols* (*transpose M*) *i p* = *transpose* (*upd-rows M i p*)
⟨*proof*⟩

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

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

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

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

lemma *upd-rows-insert*: *upd-rows M* (*insert i I*) *f* = *upd-row* (*upd-rows M I f*) *i* (*f i*)
⟨*proof*⟩

lemma *upd-rows-insert-rev*: *upd-rows M* (*insert i I*) *f* = *upd-rows* (*upd-row M i* (*f i*)) *I f*
⟨*proof*⟩

lemma *upd-rows-upd-row-swap*: *i* ∉ *I* ⟹ *upd-rows* (*upd-row M i x*) *I f* = *upd-row* (*upd-rows M I f*) *i x*
⟨*proof*⟩

lemma *upd-cols-insert*: *upd-cols M* (*insert i I*) *f* = *upd-col* (*upd-cols M I f*) *i* (*f i*)
⟨*proof*⟩

lemma *upd-cols-insert-rev*: *upd-cols M* (*insert i I*) *f* = *upd-cols* (*upd-col M i* (*f i*)) *I f*
⟨*proof*⟩

lemma *upd-cols-upd-col-swap*: *i* ∉ *I* ⟹ *upd-cols* (*upd-col M i x*) *I f* = *upd-col* (*upd-cols M I f*) *i x*
⟨*proof*⟩

lemma *upd-rows-cong*[*cong*]:
M = *N* ⟹ *T* = *S* ⟹ (⋀*s*. *s* ∈ *S* =*simp*=> *f s* = *g s*) ⟹ *upd-rows M T f* = *upd-rows N S g*
⟨*proof*⟩

lemma *upd-cols-cong*[*cong*]:
M = *N* ⟹ *T* = *S* ⟹ (⋀*s*. *s* ∈ *S* =*simp*=> *f s* = *g s*) ⟹ *upd-cols M T f* = *upd-cols N S g*
⟨*proof*⟩

lemma *row-upd-row-If*: *row* (*upd-row M i x*) *j* = (*if i* = *j then x else row M j*)
⟨*proof*⟩

lemma *row-upd-row*[*simp*]: *row* (*upd-row M i x*) *i* = *x*
⟨*proof*⟩

lemma *col-upd-col-If*: *col* (*upd-col M i x*) *j* = (*if i* = *j then x else col M j*)
⟨*proof*⟩

lemma *col-upd-col*[*simp*]: *col* (*upd-col M i x*) *i* = *x*
⟨*proof*⟩

lemma *upd-row-row*[*simp*]: *upd-row M i* (*row M i*) = *M*
⟨*proof*⟩

lemma *upd-row-upd-row-cancel*[*simp*]: *upd-row* (*upd-row M i x*) *i y* = *upd-row M i y*
⟨*proof*⟩

lemma *upd-col-upd-col-cancel*[*simp*]: *upd-col* (*upd-col M i x*) *i y* = *upd-col M i y*
⟨*proof*⟩

lemma *upd-col-col*[*simp*]: *upd-col M i* (*col M i*) = *M*
⟨*proof*⟩

lemma *row-transpose*: *row* (*transpose M*) *i* = *col M i*
⟨*proof*⟩

lemma *col-transpose*: *col* (*transpose M*) *i* = *row M i*
⟨*proof*⟩

lemma *det-perm-cols*:
fixes *A* :: *'a*::*comm-ring-1*^^*'n*
assumes *p*: *p permutes UNIV*
shows *det* (*perm-cols A p*) = *of-int* (*sign p*) ∗ *det A*
⟨*proof*⟩

lemma *det-perm-rows*:
fixes *A* :: *'a*::*comm-ring-1*^^*'n*
assumes *p*: *p permutes UNIV*
shows *det* (*perm-rows A p*) = *of-int* (*sign p*) ∗ *det A*
⟨*proof*⟩

lemma *det-row-add*: *det* (*upd-row M i* (*a* + *b*)) = *det* (*upd-row M i a*) + *det* (*upd-row M i b*)
⟨*proof*⟩

lemma *det-row-mul*: *det* (*upd-row M i* (*c* ∗*s a*)) = *c* ∗ *det* (*upd-row M i a*)

⟨*proof*⟩

lemma *det-row-uminus*: *det* (*upd-row M i* (− *a*)) = − *det* (*upd-row M i a*)
⟨*proof*⟩

lemma *det-row-minus*: *det* (*upd-row M i* (*a* − *b*)) = *det* (*upd-row M i a*) − *det* (*upd-row M i b*)
⟨*proof*⟩

lemma *det-row-0*: *det* (*upd-row M i 0*) = *0*
⟨*proof*⟩

lemma *det-row-sum*: *det* (*upd-row M i* (∑ *s*∈*S*. *a s*)) = (∑ *s*∈*S*. *det* (*upd-row M i* (*a s*)))
⟨*proof*⟩

lemma *det-col-add*: *det* (*upd-col M i* (*a* + *b*)) = *det* (*upd-col M i a*) + *det* (*upd-col M i b*)
⟨*proof*⟩

lemma *det-col-mul*: *det* (*upd-col M i* (*c* ∗*s a*)) = *c* ∗ *det* (*upd-col M i a*)
⟨*proof*⟩

lemma *det-col-uminus*: *det* (*upd-col M i* (− *a*)) = − *det* (*upd-col M i a*)
⟨*proof*⟩

lemma *det-col-minus*: *det* (*upd-col M i* (*a* − *b*)) = *det* (*upd-col M i a*) − *det* (*upd-col M i b*)
⟨*proof*⟩

lemma *det-col-0*: *det* (*upd-col M i 0*) = *0*
⟨*proof*⟩

lemma *det-col-sum*: *det* (*upd-col M i* (∑ *s*∈*S*. *a s*)) = (∑ *s*∈*S*. *det* (*upd-col M i* (*a s*)))
⟨*proof*⟩

lemma *det-identical-cols*:
assumes *i* ≠ *i′* **shows** *col A i* = *col A i′* ⟹ *det A* = *0*
⟨*proof*⟩

lemma *det-identical-rows*: *i* ≠ *i′* ⟹ *row A i* = *row A i′* ⟹ *det A* = *0*
⟨*proof*⟩

lemma *det-cols-sum*:
det (*upd-cols M T* (λ*i*. ∑ *s*∈*S*. *a i s*)) = (∑ *f*∈*T* →*E S*. *det* (*upd-cols M T* (λ*i*. *a i* (*f i*))))
⟨*proof*⟩

lemma *det-rows-sum*:
det (*upd-rows M T* (λ*i*. ∑ *s*∈*S*. *a i s*)) = (∑ *f*∈*T* →$_E$ *S*. *det* (*upd-rows M T* (λ*i*. *a i* (*f i*))))
⟨*proof*⟩

lemma *det-rows-mult*: *det* (*upd-rows M T* (λ*i*. *c i* ∗*s a i*)) = (∏ *i*∈*T*. *c i*) ∗ *det* (*upd-rows M T a*)
⟨*proof*⟩

lemma *det-cols-mult*: *det* (*upd-cols M T* (λ*i*. *c i* ∗*s a i*)) = (∏ *i*∈*T*. *c i*) ∗ *det* (*upd-cols M T a*)
⟨*proof*⟩

lemma *det-perm-rows-If*: *det* (*perm-rows B f*) = (*if f permutes UNIV then of-int* (*sign f*) ∗ *det B else 0*)
⟨*proof*⟩

lemma *det-mult*: *det* (*A* ∗ *B*) = *det A* ∗ *det B*
⟨*proof*⟩

lift-definition *minor* :: *'a*^^*'b* ⇒ *'b* ⇒ *'b* ⇒ *'a*::*semiring-1*^^*'b* **is**
λ*A i j k l*. *if k* = *i* ∧ *l* = *j then 1 else if k* = *i* ∨ *l* = *j then 0 else A k l* ⟨*proof*⟩

lemma *minor-transpose*: *minor* (*transpose A*) *i j* = *transpose* (*minor A j i*)
⟨*proof*⟩

lemma *minor-eq-row-col*: *minor M i j* = *upd-row* (*upd-col M j* (*axis i 1*)) *i* (*axis j 1*)
⟨*proof*⟩

lemma *minor-eq-col-row*: *minor M i j* = *upd-col* (*upd-row M i* (*axis j 1*)) *j* (*axis i 1*)
⟨*proof*⟩

lemma *row-minor*: *row* (*minor M i j*) *i* = *axis j 1*
⟨*proof*⟩

lemma *col-minor*: *col* (*minor M i j*) *j* = *axis i 1*
⟨*proof*⟩

lemma *det-minor-row′*:
row B i = *axis j 1* ⟹ *det* (*minor B i j*) = *det B*
⟨*proof*⟩

lemma *det-minor-row*: *det* (*minor B i j*) = *det* (*upd-row B i* (*axis j 1*))
⟨*proof*⟩

lemma *det-minor-col*: *det* (*minor B i j*) = *det* (*upd-col B j* (*axis i 1*))

⟨*proof*⟩

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

lemma *cofactor-transpose*: *cofactor* (*transpose* A) = *transpose* (*cofactor* A)
⟨*proof*⟩

definition *adjugate* A = *transpose* (*cofactor* A)

lemma *adjugate-transpose*: *adjugate* (*transpose* A) = *transpose* (*adjugate* A)
⟨*proof*⟩

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

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

end

⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*proof*⟩⟨*p
theorem *Cayley-Hamilton*:
fixes A :: $'a$::*comm-ring-1* $^{\frown\frown}$ $'n$
shows *poly-mat* (*charpoly* A) $A = 0$
⟨*proof*⟩

Part 1

define n **where** $n = CARD('n) - 1$
then have *d-charpoly*: $n + 1$ = *degree* (*charpoly* A) **and**
d-adj: n = *max-degree* (*adjugate* ($\mathbf{X} - \mathbf{C}\ A$))

define B **where** $B\ i$ = *map-sq-matrix* ($\lambda p.$ *coeff* $p\ i$) (*adjugate* ($\mathbf{X} - \mathbf{C}\ A$)) **for** i
have *A-eq-B*: *adjugate* ($\mathbf{X} - \mathbf{C}\ A$) = ($\sum i \le n.\ X\hat{}i *_S \mathbf{C}\ (B\ i)$)

Part 2

have *charpoly* $A *_S 1 = X *_S$ *adjugate* ($\mathbf{X} - \mathbf{C}\ A$) $- \mathbf{C}\ A *$ *adjugate* ($\mathbf{X} - \mathbf{C}\ A$)
also have $\ldots = (\sum i \le n.\ X\hat{}(i + 1) *_S \mathbf{C}\ (B\ i)) - (\sum i \le n.\ X\hat{}i *_S \mathbf{C}\ (A * B\ i))$
also have $(\sum i \le n.\ X\hat{}(i + 1) *_S \mathbf{C}\ (B\ i)) =$
$(\sum i < n.\ X\hat{}(i + 1) *_S \mathbf{C}\ (B\ i)) + X\hat{}(n + 1) *_S \mathbf{C}\ (B\ n)$
also have $(\sum i \le n.\ X\hat{}i *_S \mathbf{C}\ (A * B\ i)) =$
$(\sum i < n.\ X\hat{}(i + 1) *_S \mathbf{C}\ (A * B\ (i + 1))) + \mathbf{C}\ (A * B\ 0)$
finally have *diag-charpoly*:
charpoly $A *_S 1 = X\hat{}(n + 1) *_S \mathbf{C}\ (B\ n) +$
$(\sum i < n.\ X\hat{}(i + 1) *_S \mathbf{C}\ (B\ i - A * B\ (i + 1))) - \mathbf{C}\ (A * B\ 0)$

let *?p* = λi. *coeff* (*charpoly A*) *i* $*_S$ *A*^*i*
let *?AB* = λi. *A*^(*i* + *1*) $*$ *B i*
have ($\sum$ *i*≤*n*+*1*. *?p i*) = *?p 0* + ($\sum$ *i*<*n*. *?p* (*i* + *1*)) + *?p* (*n* + *1*)
also have *?p 0* = − *?AB 0*
also have ($\sum$ *i*<*n*. *?p* (*i* + *1*)) = ($\sum$ *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.