Interpolation Polynomials (in HOL-Algebra)

Emin Karayel

March 17, 2025

Abstract

A well known result from algebra is that, on any field, there is exactly one polynomial of degree less than n interpolating n points [1, §7].

This entry contains a formalization of the above result, as well as the following generalization in the case of finite fields F: There are $|F|^{m-n}$ polynomials of degree less than $m \geq n$ interpolating the same n points, where $|F|$ denotes the size of the domain of the field. To establish the result the entry also includes a formalization of Lagrange interpolation, which might be of independent interest.

The formalized results are defined on the algebraic structures from HOL-Algebra, which are distinct from the type-class based structures defined in HOL. Note that there is an existing formalization for polynomial interpolation and, in particular, Lagrange interpolation by Thiemann and Yamada [2] on the type-class based structures in HOL.

Contents

1 Bounded Degree Polynomials

This section contains a definition for the set of polynomials with a degree bound and establishes its cardinality.

theory *Bounded-Degree-Polynomials*
  imports *HOL−Algebra.Polynomial-Divisibility*
begin

lemma (**in** *ring*) *coeff-in-carrier*: $p \in carrier\ (poly\text{-}ring\ R) \Longrightarrow coeff\ p\ i \in carrier\ R$

using *poly-coeff-in-carrier carrier-is-subring* **by** (*simp add*: *univ-poly-carrier*)

definition *bounded-degree-polynomials*
where *bounded-degree-polynomials F n* = {*x*. *x* ∈ *carrier* (*poly-ring F*) ∧ (*degree x* < *n* ∨ *x* = [])}

Note: The definition for *bounded-degree-polynomials* includes the zero polynomial in *bounded-degree-polynomials F 0*. The reason for this adjustment is that, contrary to definition in HOL Algebra, most authors set the degree of the zero polynomial to $-\infty$ [1, §7.2.2]. That definition make some identities, such as $\deg(fg) = \deg f + \deg g$ for polynomials f and g unconditionally true. In particular, it prevents an unnecessary corner case in the statement of the results established in this entry.

lemma *bounded-degree-polynomials-length*:
bounded-degree-polynomials F n = {*x*. *x* ∈ *carrier* (*poly-ring F*) ∧ *length x* ≤ *n*}
unfolding *bounded-degree-polynomials-def* **using** *leI order-less-le-trans* **by** *fastforce*

lemma (**in** *ring*) *fin-degree-bounded*:
assumes *finite* (*carrier R*)
shows *finite* (*bounded-degree-polynomials R n*)
proof −
have *bounded-degree-polynomials R n* ⊆ {*p*. *set p* ⊆ *carrier R* ∧ *length p* ≤ *n*}
unfolding *bounded-degree-polynomials-length*
using *assms polynomial-incl univ-poly-carrier* **by** *blast*
thus *?thesis*
using *assms finite-lists-length-le finite-subset* **by** *fast*
qed

lemma (**in** *ring*) *non-empty-bounded-degree-polynomials*:
bounded-degree-polynomials R k ≠ {}
proof −
have $\mathbf{0}_{\textit{poly-ring } R}$ ∈ *bounded-degree-polynomials R k*
by (*simp add*: *bounded-degree-polynomials-def univ-poly-zero univ-poly-zero-closed*)
thus *?thesis* **by** *auto*
qed

lemma *in-image-by-witness*:
assumes ⋀*x*. *x* ∈ *A* ⟹ *g x* ∈ *B* ∧ *f* (*g x*) = *x*
shows *A* ⊆ *f* ‘ *B*
by (*metis assms image-eqI subsetI*)

lemma *card-mostly-constant-maps*:
assumes *y* ∈ *B*
shows *card* {*f*. *range f* ⊆ *B* ∧ (∀ *x*. *x* ≥ *n* ⟶ *f x* = *y*)} = *card B* ^ *n* (**is** *card ?A* = *?B*)
proof −
define *f* **where** *f* = (λ*f k*. *if k* < *n then f k else y*)

have *a*:*?A* ⊆ (*f* ‘ ({*0*..<*n*} →$_E$ *B*))
unfolding *f-def*
by (*rule in-image-by-witness*[**where** *g*=λ*f*. *restrict f* {*0*..<*n*}], *auto*)

have *b*:(*f* ‘ ({*0*..<*n*} →$_E$ *B*)) ⊆ *?A*
using *f-def assms* **by** *auto*

have *c*: *inj-on f* ({*0*..<*n*} →$_E$ *B*)
by (*rule inj-onI*, *metis PiE-E atLeastLessThan-iff ext f-def*)

have *card ?A* = *card* (*f* ‘ ({*0*..<*n*} →$_E$ *B*))
using *a b* **by** *auto*
also have ... = *card* ({*0*..<*n*} →$_E$ *B*)
by (*metis c card-image*)
also have ... = *card B* ^ *n*
by (*simp add*: *card-PiE*[*OF finite-atLeastLessThan*])
finally show *?thesis* **by** *simp*
qed

definition (**in** *ring*) *build-poly* **where**
build-poly f n = *normalize* (*rev* (*map f* [*0*..<*n*]))

lemma (**in** *ring*) *poly-degree-bound-from-coeff*:
assumes *x* ∈ *carrier* (*poly-ring R*)
assumes ⋀*k*. *k* ≥ *n* ⟹ *coeff x k* = **0**
shows *degree x* < *n* ∨ *x* = $\mathbf{0}_{poly\text{-}ring\ R}$
proof (*rule ccontr*)
assume *a*:¬(*degree x* < *n* ∨ *x* = $\mathbf{0}_{poly\text{-}ring\ R}$)
hence *b*:*lead-coeff x* ≠ $\mathbf{0}_R$
by (*metis assms*(*1*) *polynomial-def univ-poly-carrier univ-poly-zero*)
hence *coeff x* (*degree x*) ≠ **0**
by (*metis a lead-coeff-simp univ-poly-zero*)
moreover have *degree x* ≥ *n* **by** (*meson a not-le*)
ultimately show *False* **using** *assms*(*2*) **by** *blast*
qed

lemma (**in** *ring*) *poly-degree-bound-from-coeff-1*:
assumes *x* ∈ *carrier* (*poly-ring R*)
assumes ⋀*k*. *k* ≥ *n* ⟹ *coeff x k* = **0**
shows *x* ∈ *bounded-degree-polynomials R n*
using *poly-degree-bound-from-coeff*[*OF assms*]
by (*simp add*:*bounded-degree-polynomials-def univ-poly-zero assms*)

lemma (**in** *ring*) *length-build-poly*:
length (*build-poly f n*) ≤ *n*
by (*metis length-map build-poly-def normalize-length-le length-rev length-upt less-imp-diff-less linorder-not-less*)

lemma (**in** *ring*) *build-poly-degree*:
degree (*build-poly f n*) $\leq$ *n*−*1*
using *length-build-poly diff-le-mono* **by** *presburger*

lemma (**in** *ring*) *build-poly-poly*:
assumes $\bigwedge i.\ i < n \Longrightarrow f\ i \in$ *carrier R*
shows *build-poly f n* $\in$ *carrier* (*poly-ring R*)
unfolding *build-poly-def univ-poly-carrier*[*symmetric*]
by (*rule normalize-gives-polynomial*, *simp add*:*image-subset-iff Ball-def assms*)

lemma (**in** *ring*) *build-poly-coeff*:
coeff (*build-poly f n*) *i* = (*if i* < *n then f i else* **0**)
proof −
show *coeff* (*build-poly f n*) *i* = (*if i* < *n then f i else* **0**)
unfolding *build-poly-def normalize-coeff*[*symmetric*]
by (*cases i* < *n*, (*simp add*:*coeff-nth rev-nth coeff-length*)+)
qed

lemma (**in** *ring*) *build-poly-bounded*:
assumes $\bigwedge k.\ k < n \Longrightarrow f\ k \in$ *carrier R*
shows *build-poly f n* $\in$ *bounded-degree-polynomials R n*
unfolding *bounded-degree-polynomials-length*
using *build-poly-poly*[*OF assms*] *length-build-poly* **by** *auto*

The following establishes the total number of polynomials with a degree less than n. Unlike the results in the following sections, it is already possible to establish this property for polynomials with coefficients in a ring.

lemma (**in** *ring*) *bounded-degree-polynomials-card*:
card (*bounded-degree-polynomials R n*) = *card* (*carrier R*) ^ *n*
proof −
have *a*:*coeff* ‘ *bounded-degree-polynomials R n* $\subseteq$ {*f*. *range f* $\subseteq$ (*carrier R*) $\wedge$ ($\forall$ *k* $\geq$ *n*. *f k* = **0**)}
by (*rule image-subsetI*, *auto simp add*:*bounded-degree-polynomials-def coeff-length coeff-in-carrier*)

have *b*:{*f*. *range f* $\subseteq$ (*carrier R*) $\wedge$ ($\forall$ *k* $\geq$ *n*. *f k* = **0**)} $\subseteq$ *coeff* ‘ *bounded-degree-polynomials R n*
apply (*rule in-image-by-witness*[**where** *g*=λ*x*. *build-poly x n*])
by (*auto simp add*:*build-poly-coeff intro*:*build-poly-bounded*)

have *inj-on coeff* (*carrier* (*poly-ring R*))
by (*rule inj-onI*, *simp add*: *coeff-iff-polynomial-cond univ-poly-carrier*)

hence *coeff-inj*: *inj-on coeff* (*bounded-degree-polynomials R n*)
using *inj-on-subset bounded-degree-polynomials-def* **by** *blast*

have *card* (*bounded-degree-polynomials R n*) = *card* (*coeff* ‘ *bounded-degree-polynomials R n*)
using *coeff-inj card-image*[*symmetric*] **by** *blast*

also have ... = *card* $\{f.\ range\ f \subseteq (carrier\ R) \land (\forall k \geq n.\ f\ k = \mathbf{0})\}$
  by (*rule arg-cong*[**where** *f=card*], *rule order-antisym*[*OF a b*])
also have ... = *card* (*carrier R*)^*n*
  by (*rule card-mostly-constant-maps*, *simp*)
finally show *?thesis* **by** *simp*
qed

end

2 Lagrange Interpolation

This section introduces the function *interpolate*, which constructs the Lagrange interpolation polynomials for a given set of points, followed by a theorem of its correctness.

theory *Lagrange-Interpolation*
  imports *HOL−Algebra.Polynomial-Divisibility*
begin

A finite product in a domain is 0 if and only if at least one factor is. This could be added to *HOL−Algebra.FiniteProduct* or *HOL−Algebra.Ring*.

lemma (**in** *domain*) *finprod-zero-iff*:
  assumes *finite A*
  assumes $\bigwedge a.\ a \in A \Longrightarrow f\ a \in carrier\ R$
  shows $finprod\ R\ f\ A = \mathbf{0} \longleftrightarrow (\exists x \in A.\ f\ x = \mathbf{0})$
  using *assms*
proof (*induct A rule*: *finite-induct*)
  case *empty*
  then show *?case* **by** *simp*
next
  case (*insert y F*)
  moreover have $f \in F \to carrier\ R$ **using** *insert* **by** *blast*
  ultimately show *?case* **by** (*simp add*:*integral-iff*)
qed

lemma (**in** *ring*) *poly-of-const-in-carrier*:
  assumes $s \in carrier\ R$
  shows *poly-of-const s* ∈ *carrier* (*poly-ring R*)
  using *poly-of-const-def assms*
  by (*simp add*:*univ-poly-carrier*[*symmetric*] *polynomial-def*)

lemma (**in** *ring*) *eval-poly-of-const*:
  assumes $x \in carrier\ R$
  shows *eval* (*poly-of-const x*) $y = x$
  using *assms* **by** (*simp add*:*poly-of-const-def*)

lemma (**in** *ring*) *eval-in-carrier-2*:
  assumes $x \in carrier\ (poly\text{-}ring\ R)$

assumes $y \in$ *carrier R*
shows *eval x y* $\in$ *carrier R*
using *eval-in-carrier univ-poly-carrier polynomial-incl assms* **by** *blast*

lemma (**in** *domain*) *poly-mult-degree-le-1*:
assumes $x \in$ *carrier* (*poly-ring R*)
assumes $y \in$ *carrier* (*poly-ring R*)
shows *degree* $(x \otimes_{poly\text{-}ring\ R} y) \leq$ *degree x* + *degree y*
proof −
have *degree* $(x \otimes_{poly\text{-}ring\ R} y) = ($*if* $x = [] \vee y = []$ *then 0 else degree x* + *degree y*)
unfolding *univ-poly-mult*
by (*metis univ-poly-carrier assms*(*1*,*2*) *carrier-is-subring poly-mult-degree-eq*)
thus *?thesis* **by** (*metis nat-le-linear zero-le*)
qed

lemma (**in** *domain*) *poly-mult-degree-le*:
assumes $x \in$ *carrier* (*poly-ring R*)
assumes $y \in$ *carrier* (*poly-ring R*)
assumes *degree* $x \leq n$
assumes *degree* $y \leq m$
shows *degree* $(x \otimes_{poly\text{-}ring\ R} y) \leq n + m$
using *poly-mult-degree-le-1 assms add-mono* **by** *force*

lemma (**in** *domain*) *poly-add-degree-le*:
assumes $x \in$ *carrier* (*poly-ring R*) *degree* $x \leq n$
assumes $y \in$ *carrier* (*poly-ring R*) *degree* $y \leq n$
shows *degree* $(x \oplus_{poly\text{-}ring\ R} y) \leq n$
using *assms poly-add-degree*
by (*metis dual-order.trans max.bounded-iff univ-poly-add*)

lemma (**in** *domain*) *poly-sub-degree-le*:
assumes $x \in$ *carrier* (*poly-ring R*) *degree* $x \leq n$
assumes $y \in$ *carrier* (*poly-ring R*) *degree* $y \leq n$
shows *degree* $(x \ominus_{poly\text{-}ring\ R} y) \leq n$
proof −
interpret *x*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

show *?thesis*
unfolding *a-minus-def*
using *assms univ-poly-a-inv-degree carrier-is-subring poly-add-degree-le x.a-inv-closed*
by *simp*
qed

lemma (**in** *domain*) *poly-sum-degree-le*:
assumes *finite A*
assumes $\bigwedge x.\ x \in A \Longrightarrow$ *degree* $(f\ x) \leq n$
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x \in$ *carrier* (*poly-ring R*)

shows *degree* (*finsum* (*poly-ring R*) *f A*) $\leq$ *n*
using *assms*
proof (*induct A rule*:*finite-induct*)
case *empty*
interpret *x*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*
show *?case* **using** *empty* **by** (*simp add*:*univ-poly-zero*)
next
case (*insert x F*)
interpret *x*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*
have *a*: *degree* (*f x* $\oplus_{poly\text{-}ring\ R}$ *finsum* (*poly-ring R*) *f F*) $\leq$ *n*
using *insert poly-add-degree-le x.finsum-closed* **by** *auto*
show *?case* **using** *insert a* **by** *auto*
qed

definition (**in** *ring*) *lagrange-basis-polynomial-aux* **where**
lagrange-basis-polynomial-aux S =
($\bigotimes_{poly\text{-}ring\ R}$ *s* $\in$ *S*. *X* $\ominus_{poly\text{-}ring\ R}$ (*poly-of-const s*))

lemma (**in** *domain*) *lagrange-aux-eval*:
assumes *finite S*
assumes *S* $\subseteq$ *carrier R*
assumes *x* $\in$ *carrier R*
shows (*eval* (*lagrange-basis-polynomial-aux S*) *x*) = ($\bigotimes$ *s* $\in$ *S*. *x* $\ominus$ *s*)
proof −
interpret *x*:*ring-hom-cring poly-ring R R* (λ*p*. *eval p x*)
by (*rule eval-cring-hom*[*OF carrier-is-subring assms*(*3*)])

have $\bigwedge$*a*. *a* $\in$ *S* $\Longrightarrow$ *X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const a* $\in$ *carrier* (*poly-ring R*)
by (*meson poly-of-const-in-carrier carrier-is-subring assms*(*2*) *cring.cring-simprules*(*4*)
domain-def subsetD univ-poly-is-domain var-closed(*1*))

moreover have $\bigwedge$*s*. *s* $\in$ *S* $\Longrightarrow$ *eval* (*X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const s*) *x* = *x* $\ominus$ *s*
using *assms var-closed carrier-is-subring poly-of-const-in-carrier subsetD*[*OF assms*(*2*)]
by (*simp add*:*eval-var eval-poly-of-const*)

moreover have *a-minus R x* $\in$ *S* $\rightarrow$ *carrier R*
using *assms* **by** *blast*

ultimately show *?thesis*
by (*simp add*:*lagrange-basis-polynomial-aux-def x.hom-finprod cong*:*finprod-cong'*)
qed

lemma (**in** *domain*) *lagrange-aux-poly*:
assumes *finite S*
assumes *S* $\subseteq$ *carrier R*
shows *lagrange-basis-polynomial-aux S* $\in$ *carrier* (*poly-ring R*)

proof −
  have *a*:*subring* (*carrier R*) *R*
    using *carrier-is-subring assms* **by** *blast*

  have *b*: ⋀*a*. *a* ∈ *S* ⟹ *X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const a* ∈ *carrier* (*poly-ring R*)
    by (*meson poly-of-const-in-carrier a assms*(*2*) *cring.cring-simprules*(*4*) *domain-def subsetD*
      univ-poly-is-domain var-closed(*1*))

  interpret *x*:*cring poly-ring R*
    using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

  show *?thesis*
    using *lagrange-basis-polynomial-aux-def b x.finprod-closed*[*OF Pi-I*] **by** *simp*
qed

lemma (**in** *domain*) *poly-prod-degree-le*:
  assumes *finite A*
  assumes ⋀*x*. *x* ∈ *A* ⟹ *f x* ∈ *carrier* (*poly-ring R*)
  shows *degree* (*finprod* (*poly-ring R*) *f A*) ≤ (∑ *x* ∈ *A*. *degree* (*f x*))
  using *assms*
proof (*induct A rule*:*finite-induct*)
  case *empty*
  interpret *x*:*cring poly-ring R*
    using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*
  show *?case* **by** (*simp add*:*univ-poly-one*)
next
  case (*insert x F*)
  interpret *x*:*cring poly-ring R*
    using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*
  have *a*:*f* ∈ *F* → *carrier* (*poly-ring R*)
    using *insert* **by** *blast*
  have *b*:*f x* ∈ *carrier* (*poly-ring R*)
    using *insert* **by** *blast*
  have *degree* (*finprod* (*poly-ring R*) *f* (*insert x F*)) = *degree* (*f x* $\otimes_{poly\text{-}ring\ R}$ *finprod* (*poly-ring R*) *f F*)
    using *a b insert* **by** *simp*
  also have ... ≤ *degree* (*f x*) + *degree* (*finprod* (*poly-ring R*) *f F*)
    using *poly-mult-degree-le x.finprod-closed*[*OF a*] *b* **by** *auto*
  also have ... ≤ *degree* (*f x*) + (∑ *y* ∈ *F*. *degree* (*f y*))
    using *insert*(*3*) *a add-mono* **by** *auto*
  also have ... = (∑ *y* ∈ (*insert x F*). *degree* (*f y*)) **using** *insert* **by** *simp*
  finally show *?case* **by** *simp*
qed

lemma (**in** *domain*) *lagrange-aux-degree*:
  assumes *finite S*
  assumes *S* ⊆ *carrier R*
  shows *degree* (*lagrange-basis-polynomial-aux S*) ≤ *card S*

proof −
interpret *x*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

have *degree X* ≤ *1* **by** (*simp add*:*var-def*)
moreover have ⋀*y*. *y*∈ *S* ⟹ *degree* (*poly-of-const y*) ≤ *1* **by** (*simp add*:*poly-of-const-def*)
ultimately have *a*: ⋀*y*. *y*∈ *S* ⟹ *degree* (*X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const y*) ≤ *1*
by (*meson assms*(*2*) *in-mono poly-of-const-in-carrier poly-sub-degree-le var-closed*[*OF carrier-is-subring*])

have *b*:⋀*y*. *y* ∈ *S* ⟹ (*X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const y*) ∈ *carrier* (*poly-ring R*)
by (*meson subsetD x.minus-closed var-closed*(*1*)[*OF carrier-is-subring*] *poly-of-const-in-carrier assms*(*2*))

have *degree* (*lagrange-basis-polynomial-aux S*) ≤ (∑ *y* ∈ *S*. *degree* (*X* $\ominus_{poly\text{-}ring\ R}$ *poly-of-const y*))
using *lagrange-basis-polynomial-aux-def b poly-prod-degree-le*[*OF assms*(*1*)] **by** *auto*
also have ... ≤ (∑ *y* ∈ *S*. *1*)
using *sum-mono a* **by** *force*
also have ... = *card S* **by** *simp*
finally show *?thesis* **by** *simp*
qed

definition (**in** *ring*) *lagrange-basis-polynomial* **where**
lagrange-basis-polynomial S x = *lagrange-basis-polynomial-aux S*
$\otimes_{poly\text{-}ring\ R}$ (*poly-of-const* (inv_R (⨂ *s* ∈ *S*. *x* ⊖ *s*)))

lemma (**in** *field*)
assumes *finite S*
assumes *S* ⊆ *carrier R*
assumes *x* ∈ *carrier R* − *S*
shows
lagrange-one: *eval* (*lagrange-basis-polynomial S x*) *x* = **1** **and**
lagrange-degree: *degree* (*lagrange-basis-polynomial S x*) ≤ *card S* **and**
lagrange-zero: ⋀*s*. *s* ∈ *S* ⟹ *eval* (*lagrange-basis-polynomial S x*) *s* = **0** **and**
lagrange-poly: *lagrange-basis-polynomial S x* ∈ *carrier* (*poly-ring R*)
proof −
interpret *x*:*ring-hom-cring poly-ring R R* (λ*p*. *eval p x*)
using *assms carrier-is-subring eval-cring-hom* **by** *blast*

define *p* **where** *p* = *lagrange-basis-polynomial-aux S*
have *a*:*eval p x* = (⨂ *s* ∈ *S*. *x* ⊖ *s*)
using *assms* **by** (*simp add*:*p-def lagrange-aux-eval*)

have *b*:*p* ∈ *carrier* (*poly-ring R*) **using** *assms*
by (*simp add*:*p-def lagrange-aux-poly*)

have ⋀*y*. *y* ∈ *S* ⟹ *a-minus R x y* ∈ *carrier R*

```
    using assms by blast

  hence c:finprod R (a-minus R x) S ∈ Units R
    using finprod-closed[OF Pi-I] assms
    by (auto simp add:field-Units finprod-zero-iff)

  have eval (lagrange-basis-polynomial S x) x =
    (⨂s ∈ S. x ⊖ s) ⊗ eval (poly-of-const (inv finprod R (a-minus R x) S)) x
    using poly-of-const-in-carrier Units-inv-closed c p-def[symmetric]
    by (simp add: lagrange-basis-polynomial-def x.hom-mult[OF b] a)
  also have ... = 1
    using poly-of-const-in-carrier Units-inv-closed c eval-poly-of-const by simp
  finally show eval (lagrange-basis-polynomial S x) x = 1 by simp

  have degree (lagrange-basis-polynomial S x) ≤ degree p + degree (poly-of-const
(inv finprod R (a-minus R x) S))
    unfolding lagrange-basis-polynomial-def p-def[symmetric]
    using poly-mult-degree-le[OF b] poly-of-const-in-carrier Units-inv-closed c by
auto
  also have ... ≤ card S + 0
    using add-mono lagrange-aux-degree[OF assms(1) assms(2)] p-def poly-of-const-def
by auto
  finally show degree (lagrange-basis-polynomial S x) ≤ card S by simp

  show ⋀s. s ∈ S ⟹ eval (lagrange-basis-polynomial S x) s = 0
  proof -
    fix s
    assume d:s ∈ S

    interpret s:ring-hom-cring poly-ring R R (λp. eval p s)
      using eval-cring-hom carrier-is-subring assms d by blast

    have eval p s = finprod R (a-minus R s) S
      using subsetD[OF assms(2) d] assms
      by (simp add:p-def lagrange-aux-eval)
    also have ... = 0
      using subsetD[OF assms(2)] d assms by (simp add: finprod-zero-iff)
    finally have eval p s = 0_R by simp

    moreover have eval (poly-of-const (inv finprod R (a-minus R x) S)) s ∈ carrier
R
      using s.hom-closed poly-of-const-in-carrier Units-inv-closed c by blast

    ultimately show eval (lagrange-basis-polynomial S x) s = 0
      using poly-of-const-in-carrier Units-inv-closed c
    by (simp add:lagrange-basis-polynomial-def Let-def p-def[symmetric] s.hom-mult[OF
b])
  qed
```

interpret *r*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

show *lagrange-basis-polynomial S x* $\in$ *carrier* (*poly-ring R*)
using *lagrange-basis-polynomial-def p-def*[*symmetric*] *poly-of-const-in-carrier Units-inv-closed*
a b c **by** *simp*
qed

definition (**in** *ring*) *interpolate* **where**
interpolate S f =
($\bigoplus_{poly\text{-}ring\ R}$ *s* $\in$ *S*. *lagrange-basis-polynomial* (*S* − {*s*}) *s* $\otimes_{poly\text{-}ring\ R}$ (*poly-of-const* (*f s*)))

Let f be a function and S be a finite subset of the domain of the field. Then *interpolate S f* will return a polynomial with degree less than *card S* interpolating f on S.

theorem (**in** *field*)
assumes *finite S*
assumes *S* $\subseteq$ *carrier R*
assumes *f* ‘ *S* $\subseteq$ *carrier R*
shows
interpolate-poly: *interpolate S f* $\in$ *carrier* (*poly-ring R*) **and**
interpolate-degree: *degree* (*interpolate S f*) $\leq$ *card S* − *1* **and**
interpolate-eval: $\bigwedge$*s*. *s* $\in$ *S* $\Longrightarrow$ *eval* (*interpolate S f*) *s* = *f s*
proof −
interpret *r*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

have *a*:$\bigwedge$*x*. *x* $\in$ *S* $\Longrightarrow$ *lagrange-basis-polynomial* (*S* − {*x*}) *x* $\in$ *carrier* (*poly-ring R*)
by (*meson lagrange-poly assms Diff-iff finite-Diff in-mono insertI1 subset-insertI2 subset-insert-iff*)

have *b*:$\bigwedge$*x*. *x* $\in$ *S* $\Longrightarrow$ *f x* $\in$ *carrier R* **using** *assms* **by** *blast*

have *c*:$\bigwedge$*x*. *x* $\in$ *S* $\Longrightarrow$ *degree* (*lagrange-basis-polynomial* (*S* − {*x*}) *x*) $\leq$ *card S* − *1*
by (*metis* (*full-types*) *lagrange-degree DiffI Diff-insert-absorb assms*(*1*) *assms*(*2*)
card-Diff-singleton finite-insert insert-subset mk-disjoint-insert)

have *d*: $\bigwedge$*x*. *x* $\in$ *S* $\Longrightarrow$
degree (*lagrange-basis-polynomial* (*S* − {*x*}) *x* $\otimes_{poly\text{-}ring\ R}$ *poly-of-const* (*f x*)) $\leq$ (*card S* − *1*) + *0*
using *poly-of-const-in-carrier*[*OF b*] *poly-mult-degree-le*[*OF a*] *c poly-of-const-def* **by** *fastforce*

show *interpolate S f* $\in$ *carrier* (*poly-ring R*)
using *interpolate-def poly-of-const-in-carrier a b* **by** *simp*

show *degree* (*interpolate S f*) ≤ *card S* − *1*
using *poly-sum-degree-le*[*OF assms*(*1*) *d*] *poly-of-const-in-carrier*[*OF b*] *interpolate-def a* **by** *simp*

have *e*:*subring* (*carrier R*) *R*
using *carrier-is-subring assms* **by** *blast*

show $\bigwedge s.\ s \in S \Longrightarrow eval\ (interpolate\ S\ f)\ s = f\ s$
proof −
fix *s*
assume *f*:*s* ∈ *S*
interpret *s*:*ring-hom-cring poly-ring R R* (λ*p*. *eval p s*)
using *eval-cring-hom*[*OF e*] *assms f* **by** *blast*
have *g*:$\bigwedge i.\ i \in S \Longrightarrow$
eval (*lagrange-basis-polynomial* (*S* − {*i*}) *i* $\otimes_{poly\text{-}ring\ R}$ *poly-of-const* (*f i*)) *s* =
(*if s* = *i then f s else* **0**)
proof −
fix *i*
assume *i-in-S*: *i* ∈ *S*
have *eval* (*lagrange-basis-polynomial* (*S* − {*i*}) *i* $\otimes_{poly\text{-}ring\ R}$ *poly-of-const* (*f i*)) *s* =
eval (*lagrange-basis-polynomial* (*S* − {*i*}) *i*) *s* ⊗ *f i*
using *b i-in-S poly-of-const-in-carrier*
by (*simp add*: *s.hom-mult*[*OF a*] *eval-poly-of-const*)
also have ... = (*if s* = *i then f s else* **0**)
using *b i-in-S poly-of-const-in-carrier assms f*
apply (*cases s*=*i*, *simp*, *subst lagrange-one*, *auto*)
by (*subst lagrange-zero*, *auto*)
finally show
eval (*lagrange-basis-polynomial* (*S* − {*i*}) *i* $\otimes_{poly\text{-}ring\ R}$ *poly-of-const* (*f i*)) *s* =
(*if s* = *i then f s else* **0**) **by** *simp*
qed

have *eval* (*interpolate S f*) *s* =
($\bigoplus$*x*∈*S*. *eval* (*lagrange-basis-polynomial* (*S* − {*x*}) *x* $\otimes_{poly\text{-}ring\ R}$ *poly-of-const* (*f x*)) *s*)
using *poly-of-const-in-carrier*[*OF b*] *a e*
by (*simp add*: *interpolate-def s.hom-finsum*[*OF Pi-I*] *comp-def*)
also have ... = ($\bigoplus$*x*∈*S*. *if s* = *x then f s else* **0**)
using *b g* **by** (*simp cong*: *finsum-cong*)
also have ... = *f s*
using *finsum-singleton*[*OF f assms*(*1*)] *f assms* **by** *auto*
finally show *eval* (*interpolate S f*) *s* = *f s* **by** *simp*
qed
qed

end

3 Cardinalities of Interpolation Polynomials

This section establishes the cardinalities of the set of polynomials with a degree bound interpolating a given set of points.

theory *Interpolation-Polynomial-Cardinalities*
  imports *Bounded-Degree-Polynomials Lagrange-Interpolation*
begin

lemma (**in** *ring*) *poly-add-coeff*:
  assumes $x \in$ *carrier* (*poly-ring R*)
  assumes $y \in$ *carrier* (*poly-ring R*)
  shows *coeff* ($x \oplus_{poly\text{-}ring\ R} y$) k = *coeff* x $k \oplus$ *coeff* y k
  by (*metis assms univ-poly-carrier polynomial-incl univ-poly-add poly-add-coeff*)

lemma (**in** *domain*) *poly-neg-coeff*:
  assumes $x \in$ *carrier* (*poly-ring R*)
  shows *coeff* ($\ominus_{poly\text{-}ring\ R}$ x) k = $\ominus$*coeff* x k
proof −
  interpret *x*:*cring poly-ring R*
   using *assms cring-def carrier-is-subring domain.univ-poly-is-cring domain-axioms*
by *auto*

  have *a*:$\mathbf{0}_{poly\text{-}ring\ R} = x \ominus_{poly\text{-}ring\ R} x$
    by (*metis x.r-right-minus-eq assms*(*1*))

  have $\mathbf{0}$ = *coeff* ($\mathbf{0}_{poly\text{-}ring\ R}$) k **by** (*simp add*:*univ-poly-zero*)
  also have ... = *coeff* x $k \oplus$ *coeff* ($\ominus_{poly\text{-}ring\ R}$ x) k **using** *a assms*
    by (*simp add*:*a-minus-def poly-add-coeff*)
  finally have $\mathbf{0}$ = *coeff* x $k \oplus$ *coeff* ($\ominus_{poly\text{-}ring\ R}$ x) k **by** *simp*
  thus *?thesis*
      by (*metis local.minus-minus x.a-inv-closed sum-zero-eq-neg coeff-in-carrier assms*)
qed

lemma (**in** *domain*) *poly-substract-coeff*:
  assumes $x \in$ *carrier* (*poly-ring R*)
  assumes $y \in$ *carrier* (*poly-ring R*)
  shows *coeff* ($x \ominus_{poly\text{-}ring\ R} y$) k = *coeff* x $k \ominus$ *coeff* y k
proof −
  interpret *x*:*cring poly-ring R*
   using *assms cring-def carrier-is-subring domain.univ-poly-is-cring domain-axioms*
by *auto*
  show *?thesis*
    using *assms* **by** (*simp add*:*a-minus-def poly-add-coeff poly-neg-coeff*)
qed

A polynomial with more zeros than its degree is the zero polynomial.

lemma (**in** *field*) *max-roots*:
  assumes $p \in$ *carrier* (*poly-ring R*)
  assumes $K \subseteq$ *carrier R*
  assumes *finite K*
  assumes *degree* $p <$ *card K*
  assumes $\bigwedge x.\ x \in K \Longrightarrow$ *eval* $p\ x = \mathbf{0}$
  shows $p = \mathbf{0}_{poly\text{-}ring\ R}$
proof (*rule ccontr*)
  assume $p \neq \mathbf{0}_{poly\text{-}ring\ R}$
  hence $a{:}p \neq []$ **by** (*simp add*: *univ-poly-zero*)
  have $\bigwedge x.$ *count* (*mset-set K*) $x \leq$ *count* (*roots p*) x
  proof −
    fix x
    show *count* (*mset-set K*) $x \leq$ *count* (*roots p*) x
    proof (*cases* $x \in K$)
      case *True*
      hence *is-root p x*
        by (*meson a assms*(*2*,*5*) *is-ring is-root-def subsetD*)
      hence $x \in$ *set-mset* (*roots p*)
        using *assms*(*1*) *roots-mem-iff-is-root field-def* **by** *force*
      hence *1* $\leq$ *count* (*roots p*) x **by** *simp*
      moreover have *count* (*mset-set K*) $x =$ *1* **using** *True assms*(*3*) **by** *simp*
      ultimately show *?thesis* **by** *presburger*
    next
      case *False*
      hence *count* (*mset-set K*) $x =$ *0* **by** *simp*
      then show *?thesis* **by** *presburger*
    qed
  qed
  hence *mset-set K* $\subseteq\#$ *roots p*
    by (*simp add*: *subseteq-mset-def*)
  hence *card K* $\leq$ *size* (*roots p*)
    by (*metis size-mset-mono size-mset-set*)
  moreover have *size* (*roots p*) $\leq$ *degree p*
    using *a size-roots-le-degree assms* **by** *auto*
  ultimately show *False* **using** *assms*(*4*)
    by (*meson leD less-le-trans*)
qed

definition (**in** *ring*) *split-poly*
  where *split-poly K p* = (*restrict* (*eval p*) *K*, $\lambda k.$ *coeff p* (*k*+*card K*))

To establish the count of the number of polynomials of degree less than n interpolating a function f on K where $|K| \leq n$, the function *split-poly K* establishes a bijection between the polynomials of degree less than n and the values of the polynomials on K in combination with the coefficients of order $|K|$ and greater.

For the injectivity: Note that the difference of two polynomials whose coefficients of order $|K|$ and larger agree must have a degree less than $|K|$ and because their values agree on k points, it must have $|K|$ zeros and hence is the zero polynomial.

For the surjectivty: Let p be a polynomial whose coefficients larger than $|K|$ are chosen, and all other coefficients be 0. Now it is possible to find a polynomial q interpolating $f - p$ on K using Lagrange interpolation. Then $p + q$ will interpolate f on K and because the degree of q is less than $|K|$ its coefficients of order $|K|$ will be the same as those of p.

A tempting question is whether it would be easier to instead establish a bijection between the polynomials of degree less than n and its values on $K \cup K'$ where K' are arbitrarily chosen $n - |K|$ points in the field. This approach is indeed easier, however, it fails for the case where the size of the field is less than n.

```
lemma (in field) split-poly-inj:
  assumes finite K
  assumes K ⊆ carrier R
  shows inj-on (split-poly K) (carrier (poly-ring R))
proof
  fix x
  fix y
  assume a1:x ∈ carrier (poly-ring R)
  assume a2:y ∈ carrier (poly-ring R)
  assume a3:split-poly K x = split-poly K y

  interpret x:cring poly-ring R
    using carrier-is-subring domain.univ-poly-is-cring domain-axioms by auto

  have x-y-carrier: x ⊖poly-ring R y ∈ carrier (poly-ring R) using a1 a2 by simp
  have ⋀k. coeff x (k+card K) = coeff y (k+card K)
    using a3 by (simp add:split-poly-def, meson)
  hence ⋀k. coeff (x ⊖poly-ring R y) (k+card K) = 0
    using coeff-in-carrier a1 a2 by (simp add:poly-substract-coeff)
  hence degree (x ⊖poly-ring R y) < card K ∨ (x ⊖poly-ring R y) = 0poly-ring R
    by (metis poly-degree-bound-from-coeff add.commute le-iff-add x-y-carrier)
  moreover have ⋀k. k ∈ K ⟹ eval x k = eval y k
    using a3 by (simp add:split-poly-def restrict-def, meson)
  hence ⋀k. k ∈ K ⟹ eval x k ⊖ eval y k = 0
   by (metis eval-in-carrier univ-poly-carrier polynomial-incl a1 assms(2) in-mono
r-right-minus-eq)
  hence ⋀k. k ∈ K ⟹ eval (x ⊖poly-ring R y) k = 0
    using a1 a2 subsetD[OF assms(2)] carrier-is-subring
    by (simp add: ring-hom-cring.hom-sub[OF eval-cring-hom])
  ultimately have x ⊖poly-ring R y = 0poly-ring R
    using max-roots x-y-carrier assms by blast
  then show x = y
```

using *x.r-right-minus-eq*[*OF a1 a2*] **by** *simp*
qed

lemma (**in** *field*) *split-poly-image*:
assumes *finite K*
assumes $K \subseteq$ *carrier R*
shows *split-poly K ‘ carrier* (*poly-ring R*) $\supseteq$
$(K \rightarrow_E \textit{carrier } R) \times \{f.\ \textit{range } f \subseteq \textit{carrier } R \wedge (\exists\, n.\ \forall\, k \geq n.\ f\, k = \mathbf{0}_R)\}$
proof (*rule subsetI*)
fix *x*
assume *a*:$x \in (K \rightarrow_E \textit{carrier } R) \times \{f.\ \textit{range } f \subseteq \textit{carrier } R \wedge (\exists\,(n{::}nat).\ \forall\, k \geq n.\ f\, k = \mathbf{0})\}$
have *a1*: $\textit{fst } x \in (K \rightarrow_E \textit{carrier } R)$
using *a* **by** (*simp add*:*mem-Times-iff*)
obtain *n* **where** *a2*: $\textit{snd } x \in \{f.\ \textit{range } f \subseteq \textit{carrier } R \wedge (\forall\, k \geq n.\ f\, k = \mathbf{0})\}$
using *a mem-Times-iff* **by** *force*
have *a3*: $\bigwedge y.\ \textit{snd } x\ y \in \textit{carrier } R$ **using** *a2* **by** *blast*

define *w* **where** $w = \textit{build-poly}\ (\lambda i.\ \textit{if } i \geq \textit{card } K \textit{ then } (\textit{snd } x\ (i - \textit{card } K))\ \textit{else } \mathbf{0})\ (\textit{card } K + n)$

have *w-carr*: $w \in$ *carrier* (*poly-ring R*)
unfolding *w-def* **by** (*rule build-poly-poly*, *simp add*:*a3*)

have *w-eval-range*: $\bigwedge x.\ x \in \textit{carrier } R \Longrightarrow \textit{local.eval } w\ x \in \textit{carrier } R$
proof −
fix *x*
assume *w-eval-range-1*:$x \in$ *carrier R*
interpret *x*:*ring-hom-cring poly-ring R R* ($\lambda p.\ \textit{eval } p\ x$)
using *eval-cring-hom*[*OF carrier-is-subring*] *assms w-eval-range-1* **by** *blast*
show $\textit{eval } w\ x \in \textit{carrier } R$
by (*rule x.hom-closed*[*OF w-carr*])
qed

interpret *r*:*cring poly-ring R*
using *carrier-is-subring domain.univ-poly-is-cring domain-axioms* **by** *auto*

define *y* **where** $y = \textit{interpolate } K\ (\lambda k.\ \textit{fst } x\ k \ominus \textit{eval } w\ k)$
define *r* **where** $r = y \oplus_{\textit{poly-ring } R} w$

have *x-minus-w-in-carrier*: $\bigwedge z.\ z \in K \Longrightarrow \textit{fst } x\ z \ominus \textit{eval } w\ z \in \textit{carrier } R$
using *a1 PiE-def Pi-def minus-closed subsetD*[*OF assms*(*2*)] *w-eval-range* **by** *auto*

have *y-poly*: $y \in$ *carrier* (*poly-ring R*) **unfolding** *y-def*
using *x-minus-w-in-carrier interpolate-poly*[*OF assms*(*1*) *assms*(*2*)] *image-subsetI* **by** *force*

have *y-degree*: $\textit{degree } y \leq \textit{card } K - 1$

unfolding *y-def*
using *x-minus-w-in-carrier interpolate-degree*[*OF assms*(*1*) *assms*(*2*)] *image-subsetI*
by *force*

have *y-len*: *length y* ≤ *card K*
proof (*cases K*={})
case *True*
then show *?thesis*
by (*simp add*:*y-def interpolate-def univ-poly-zero*)
next
case *False*
then show *?thesis*
by (*metis y-degree Suc-le-D assms*(*1*) *card-gt-0-iff diff-Suc-1 not-less-eq-eq order.strict-iff-not*)
qed

have *r-poly*: *r* ∈ *carrier* (*poly-ring R*)
using *r-def y-poly w-carr* **by** *simp*

have *coeff-r*: ⋀*k*. *coeff r* (*k* + *card K*) = *snd x k*
proof −
fix *k* :: *nat*
have *y-len'*: *length y* ≤ *k* + *card K* **using** *y-len trans-le-add2* **by** *blast*
have *coeff r* (*k* + *card K*) = *coeff y* (*k* + *card K*) ⊕ *coeff w* (*k*+*card K*)
by (*simp add*:*r-def poly-add-coeff*[*OF y-poly w-carr*])
also have ... = **0** ⊕ *coeff w* (*k*+*card K*)
using *coeff-length*[*OF y-len'*] **by** *simp*
also have ... = *coeff w* (*k*+*card K*)
using *coeff-in-carrier*[*OF w-carr*] **by** *simp*
also have ... = *snd x k*
using *a2* **by** (*simp add*:*w-def build-poly-coeff not-less*)
finally show *coeff r* (*k* + *card K*) = *snd x k* **by** *simp*
qed

have *eval-r*: ⋀*k*. *k* ∈ *K* ⟹ *eval r k* = *fst x k*
proof −
fix *k*
assume *b*:*k* ∈ *K*
interpret *s*:*ring-hom-cring poly-ring R R* (λ*p*. *eval p k*)
using *eval-cring-hom*[*OF carrier-is-subring*] *assms b* **by** *blast*

have *k-carr*: *k* ∈ *carrier R* **using** *assms*(*2*) *b* **by** *blast*
have *fst-x-k-carr*: ⋀*k*. *k* ∈ *K* ⟹ *fst x k* ∈ *carrier R*
using *a1 PiE-def Pi-def* **by** *blast*
have *eval r k* = *eval y k* ⊕ *eval w k*
using *y-poly w-carr* **by** (*simp add*:*r-def*)
also have ... = *fst x k* ⊖ *local.eval w k* ⊕ *local.eval w k*
using *assms b x-minus-w-in-carrier*
by (*simp add*:*y-def interpolate-eval*[*OF - - image-subsetI*])

also have ... = *fst x k* ⊕ (⊖ *local.eval w k* ⊕ *local.eval w k*)
using *fst-x-k-carr*[*OF b*] *w-eval-range*[*OF k-carr*]
by (*simp add*:*a-minus-def a-assoc*)
also have ... = *fst x k*
using *fst-x-k-carr*[*OF b*] *w-eval-range*[*OF k-carr*]
by (*simp add*:*a-comm r-neg*)
finally show *eval r k* = *fst x k* **by** *simp*
qed

have $r \in$ (*carrier* (*poly-ring R*))
by (*metis r-poly*)
moreover have $\bigwedge y$. (*if* $y \in K$ *then eval r y else undefined*) = *fst x y*
using *a1 eval-r PiE-E* **by** *auto*
hence *split-poly K r* = *x*
by (*simp add*:*split-poly-def prod-eq-iff coeff-r restrict-def*)
ultimately show $x \in$ *split-poly K* ‘ (*carrier* (*poly-ring R*))
by *blast*
qed

This is like *card-vimage-inj* but supports *inj-on* instead.

lemma *card-vimage-inj-on*:
assumes *inj-on f B*
assumes $A \subseteq f$ ‘ B
shows *card* (f −‘ $A \cap B$) = *card A*
proof −
have $A = f$ ‘ (f −‘ $A \cap B$) **using** *assms*(*2*) **by** *auto*
thus *?thesis* **using** *assms card-image*
by (*metis inf-le2 inj-on-subset*)
qed

lemma *inv-subsetI*:
assumes $\bigwedge x.\ x \in A \Longrightarrow f\ x \in B \Longrightarrow x \in C$
shows f −‘ $B \cap A \subseteq C$
using *assms* **by** *force*

The following establishes the main result of this section: There are $|F|^{n-k}$ polynomials of degree less than n interpolating $k \leq n$ points.

lemma *restrict-eq-imp*:
assumes *restrict f A* = *restrict g A*
assumes $x \in A$
shows $f\ x = g\ x$
by (*metis restrict-def assms*)

theorem (**in** *field*) *interpolating-polynomials-card*:
assumes *finite K*
assumes $K \subseteq$ *carrier R*
assumes f ‘ $K \subseteq$ *carrier R*
shows *card* {$\omega \in$ *bounded-degree-polynomials R* (*card K* + *n*). ($\forall k \in K$. *eval* ω $k = f\ k$)} = *card* (*carrier R*)^*n*

(**is** *card ?A* = *?B*)
proof −
define *z* **where** *z* = *restrict f K*
define *M* **where** *M* = {*f*. *range f* ⊆ *carrier R* ∧ (∀ *k* ≥ *n*. *f k* = **0**)}

hence *inj-on-bounded*: *inj-on* (*split-poly K*) (*carrier* (*poly-ring R*))
using *split-poly-inj*[*OF assms*(*1*) *assms*(*2*)] **by** *blast*

have *?A* ⊆ *split-poly K* −' ({*z*} × *M*)
unfolding *split-poly-def z-def M-def bounded-degree-polynomials-length*
by (*rule subsetI*, *auto intro*!:*coeff-in-carrier coeff-length*)
moreover have *?A* ⊆ *carrier* (*poly-ring R*)
unfolding *bounded-degree-polynomials-length* **by** *blast*
ultimately have *a*:*?A* ⊆ *split-poly K* −' ({*z*} × *M*) ∩ *carrier* (*poly-ring R*)
by *blast*

have ⋀*x k* . (λ*k*. *coeff x* (*k* + *card K*)) ∈ *M* ⟹ *k* ≥ *n* + *card K* ⟹ *coeff x k* = **0**
by (*simp add*:*M-def*, *metis Nat.le-diff-conv2 Nat.le-imp-diff-is-add add-leD2*)
hence *split-poly K* −' ({*z*} × *M*) ∩ *carrier* (*poly-ring R*) ⊆ *bounded-degree-polynomials R* (*card K* + *n*)
unfolding *split-poly-def z-def* **using** *poly-degree-bound-from-coeff-1 inv-subsetI* **by** *force*
moreover have ⋀ω *k*. ω ∈ *split-poly K* −' ({*z*} × *M*) ∩ *carrier* (*poly-ring R*) ⟹ *k* ∈ *K* ⟹ *eval* ω *k* = *f k*
unfolding *split-poly-def z-def* **using** *restrict-eq-imp* **by** *fastforce*
ultimately have *b*:*split-poly K* −' ({*z*} × *M*) ∩ *carrier* (*poly-ring R*) ⊆ *?A*
by *blast*

have *z* ∈ *K* $\rightarrow_E$ *carrier R*
unfolding *z-def* **using** *assms*(*3*) **by** *auto*
moreover have *M* ⊆ {*f*. *range f* ⊆ *carrier R* ∧ (∃ *n*. (∀ *k* ≥ *n*. *f k* = **0**))}
unfolding *M-def* **by** *blast*
ultimately have *c*:{*z*} × *M* ⊆ *split-poly K* ' *carrier* (*poly-ring R*)
using *split-poly-image*[*OF assms*(*1*) *assms*(*2*)] **by** *fast*

have *card ?A* = *card* (*split-poly K* −' ({*z*} × *M*) ∩ *carrier* (*poly-ring R*))
using *order-antisym*[*OF a b*] **by** *simp*
also have ... = *card* ({*z*} × *M*)
using *card-vimage-inj-on*[*OF inj-on-bounded*] *c* **by** *blast*
also have ... = *card* (*carrier R*)^*n*
by (*simp add*:*card-cartesian-product M-def card-mostly-constant-maps*)
finally show *?thesis* **by** *simp*
qed

A corollary is the classic result [1, Theorem 7.15] that there is exactly one polynomial of degree less than n interpolating n points:

corollary (**in** *field*) *interpolating-polynomial-one*:
assumes *finite K*

assumes *K* ⊆ *carrier R*
assumes *f ‘ K* ⊆ *carrier R*
shows *card* {*ω* ∈ *bounded-degree-polynomials R* (*card K*). (∀ *k* ∈ *K*. *eval ω k* = *f k*)} = *1*
using *interpolating-polynomials-card*[*OF assms*(*1*) *assms*(*2*) *assms*(*3*), **where** *n*=*0*]
by *simp*

In the case of fields with infinite carriers, it is possible to conclude that there are infinitely many polynomials of degree less than n interpolating $k < n$ points.

corollary (**in** *field*) *interpolating-polynomial-inf*:
assumes *infinite* (*carrier R*)
assumes *finite K K* ⊆ *carrier R f ‘ K* ⊆ *carrier R*
assumes *n* > *0*
shows *infinite* {*ω* ∈ *bounded-degree-polynomials R* (*card K* + *n*). (∀ *k* ∈ *K*. *eval ω k* = *f k*)}
(**is** *infinite ?A*)
proof −
have {} ⊂ {*ω* ∈ *bounded-degree-polynomials R* (*card K*). (∀ *k* ∈ *K*. *eval ω k* = *f k*)}
using *interpolating-polynomial-one*[*OF assms*(*2*) *assms*(*3*) *assms*(*4*)] **by** *fastforce*
also have ... ⊆ *?A*
unfolding *bounded-degree-polynomials-def* **by** *auto*
finally have *a*:*?A* ≠ {} **by** *auto*

have *card ?A* = *card* (*carrier R*)^*n*
using *interpolating-polynomials-card*[*OF assms*(*2*) *assms*(*3*) *assms*(*4*), **where** *n*=*n*] **by** *simp*
also have ... = *0*
using *assms*(*1*) *assms*(*5*) **by** *simp*
finally have *b*:*card ?A* = *0* **by** *simp*

show *?thesis* **using** *a b card-0-eq* **by** *blast*
qed

The following is an additional independent result: The evaluation homomorphism is injective for degree one polynomials.

lemma (**in** *field*) *eval-inj-if-degree-1*:
assumes *p* ∈ *carrier* (*poly-ring R*) *degree p* = *1*
shows *inj-on* (*eval p*) (*carrier R*)
proof −
obtain *u v* **where** *p-def*: *p* = [*u*,*v*] **using** *assms*
by (*cases p*, *cases* (*tl p*), *auto*)

have *u* ∈ *carrier R* − {**0**} **using** *p-def assms* **by** *blast*
moreover have *v* ∈ *carrier R* **using** *p-def assms* **by** *blast*
ultimately show *?thesis* **by** (*simp add*:*p-def field-Units inj-on-def*)

qed

end

References

[1] V. Shoup. *A Computational Introduction to Number theory and Algebra.* Cambridge university press, 2009.

[2] R. Thiemann and A. Yamada. Polynomial interpolation. *Archive of Formal Proofs*, Jan. 2016. https://isa-afp.org/entries/Polynomial_Interpolation.html, Formal proof development.