The Chevalley–Warning Theorem

Arthur Freitas Ramos David Barros Hulak
Ruy J. G. B. de Queiroz

July 16, 2026

Abstract

This development formalizes the Chevalley–Warning theorem for finite fields: if finitely many multivariate polynomials over a finite field have total degree sum strictly smaller than the number of variables, then the number of their common zeros is divisible by the characteristic [1, 3]. The formalization reuses the concrete multivariate-polynomial representation and operations from the AFP *Polynomials* entry [2], and the proof follows the standard finite-field power-sum argument. AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

Contents

theory *Chevalley-Warning*
imports
HOL−Number-Theory.Residues
HOL−Computational-Algebra.Polynomial
HOL−Library.FuncSet
Polynomials.Polynomials
begin

1 The Chevalley-Warning theorem

We reuse the concrete multivariate-polynomial representation from the AFP entry *Polynomials.Polynomials*. Thus a polynomial has type $('v, 'a)$ *Polynomials.poly* and consists of AFP monomials paired with coefficients. The theorem takes an explicit finite set of variables, so it applies directly to the usual natural-number-indexed AFP polynomials.

definition *variables-poly* ::

$'v$::*linorder set* $\Rightarrow$ ($'v$, $'a$) *poly* $\Rightarrow$ *bool* **where**
variables-poly V p $\longleftrightarrow$ *poly-vars p* $\subseteq$ *V*

definition *total-degree-poly-le* :: ($'v$::*linorder*, $'a$) *poly* $\Rightarrow$ *nat* $\Rightarrow$ *bool* **where**
total-degree-poly-le p d $\longleftrightarrow$ *poly-degree p* $\leq$ *d*

definition *total-degree-poly-less* :: ($'v$::*linorder*, $'a$) *poly* $\Rightarrow$ *nat* $\Rightarrow$ *bool* **where**
total-degree-poly-less p d $\longleftrightarrow$ *poly-degree p* $<$ *d*

definition *common-zeros* ::
$'v$::*linorder set* $\Rightarrow$
($'v$, $'a$::*comm-semiring-1*) *poly list* $\Rightarrow$ ($'v \Rightarrow 'a$) *set* **where**
common-zeros V ps =
$\{\alpha \in$ *PiE V* (λ-. *UNIV*). *list-all* (λp. *eval-poly* α $p = 0$) *ps*$\}$

lemma *variables-poly-UNIV* [*simp*]: *variables-poly UNIV p*
⟨*proof*⟩

lemma *monom-vars-subset-poly-vars*:
assumes (m, c) $\in$ *set p*
shows *monom-vars m* $\subseteq$ *poly-vars p*
⟨*proof*⟩

lemma *poly-degree-leI*:
assumes $\bigwedge m\ c.$ (m, c) $\in$ *set p* $\Longrightarrow$ *monom-degree m* $\leq$ *d*
shows *poly-degree p* $\leq$ *d*
⟨*proof*⟩

lemma *total-degree-poly-leD*:
assumes *total-degree-poly-le p d* (m, c) $\in$ *set p*
shows *monom-degree m* $\leq$ *d*
⟨*proof*⟩

lemma *total-degree-poly-lessD*:
assumes *total-degree-poly-less p d* (m, c) $\in$ *set p*
shows *monom-degree m* $<$ *d*
⟨*proof*⟩

lemma *total-degree-poly-le-mono*:
assumes *total-degree-poly-le p d* $d \leq e$
shows *total-degree-poly-le p e*
⟨*proof*⟩

lemma *total-degree-poly-less-of-le*:
assumes *total-degree-poly-le p d* $d < e$
shows *total-degree-poly-less p e*
⟨*proof*⟩

1.1 Polynomial operations

fun *poly-prod* :: ('*v*::*linorder*, '*a*::*comm-semiring-1*) *poly list* ⇒ ('*v*, '*a*) *poly* **where**
poly-prod [] = *one-poly*
| *poly-prod* (*p* # *ps*) = *poly-mult p* (*poly-prod ps*)

definition *indicator-factor* ::
nat ⇒ ('*v*::*linorder*, '*a*::*comm-ring-1*) *poly* ⇒ ('*v*, '*a*) *poly* **where**
indicator-factor q p = *poly-minus* (*poly-const 1*) (*poly-power p* (*q* − *1*))

definition *indicator-poly* ::
nat ⇒ ('*v*::*linorder*, '*a*::*comm-ring-1*) *poly list* ⇒ ('*v*, '*a*) *poly* **where**
indicator-poly q ps = *poly-prod* (*map* (*indicator-factor q*) *ps*)

lemma *monom-list-degree-mult*:
monom-list-degree (*monom-mult-list m n*) =
monom-list-degree m + *monom-list-degree n*
⟨*proof*⟩

lemma *monom-degree-mult* [*simp*]:
monom-degree (*m* ∗ *n*) = *monom-degree m* + *monom-degree n*
⟨*proof*⟩

lemma *monom-degree-one* [*simp*]:
monom-degree (*1*::'*v*::*linorder monom*) = *0*
⟨*proof*⟩

lemma *monom-vars-mult* [*simp*]:
monom-vars (*m* ∗ *n*) = *monom-vars m* ∪ *monom-vars n*
⟨*proof*⟩

lemma *monom-mult-poly-monoms*:
poly-monoms (*monom-mult-poly* (*m*, *c*) *p*) ⊆ (∗) *m* ‘ *poly-monoms p*
⟨*proof*⟩

lemma *variables-poly-const* [*simp*]:
variables-poly V (*poly-const c* :: ('*v*::*linorder*, '*a*::*zero*) *poly*)
⟨*proof*⟩

lemma *total-degree-poly-le-const* [*simp*]:
total-degree-poly-le (*poly-const c* :: ('*v*::*linorder*, '*a*::*zero*) *poly*) *0*
⟨*proof*⟩

lemma *variables-poly-add*:
assumes *p*: *variables-poly V p* **and** *q*: *variables-poly V q*
shows *variables-poly V* (*poly-add p q*)
⟨*proof*⟩

lemma *total-degree-poly-le-add*:
assumes *p*: *total-degree-poly-le p d* **and** *q*: *total-degree-poly-le q d*

shows *total-degree-poly-le* (*poly-add p q*) *d*
⟨*proof*⟩

lemma *total-degree-monom-mult-poly*:
assumes *m*: *monom-degree m* $\leq$ *d* **and** *p*: *total-degree-poly-le p e*
shows *total-degree-poly-le* (*monom-mult-poly* (*m*, *c*) *p*) (*d* + *e*)
⟨*proof*⟩

lemma *variables-monom-mult-poly*:
assumes *m*: *monom-vars m* $\subseteq$ *V* **and** *p*: *variables-poly V p*
shows *variables-poly V* (*monom-mult-poly* (*m*, *c*) *p*)
⟨*proof*⟩

lemma *variables-poly-mult*:
assumes *p*: *variables-poly V p* **and** *q*: *variables-poly V q*
shows *variables-poly V* (*poly-mult p q*)
⟨*proof*⟩

lemma *total-degree-poly-le-mult*:
assumes *p*: *total-degree-poly-le p d* **and** *q*: *total-degree-poly-le q e*
shows *total-degree-poly-le* (*poly-mult p q*) (*d* + *e*)
⟨*proof*⟩

lemma *variables-poly-minus*:
assumes *p*: *variables-poly V p* **and** *q*: *variables-poly V q*
shows *variables-poly V* (*poly-minus p q*)
⟨*proof*⟩

lemma *total-degree-poly-le-minus*:
assumes *p*: *total-degree-poly-le p d* **and** *q*: *total-degree-poly-le q d*
shows *total-degree-poly-le* (*poly-minus p q*) *d*
⟨*proof*⟩

lemma *variables-poly-power*:
assumes *p*: *variables-poly V p*
shows *variables-poly V* (*poly-power p k*)
⟨*proof*⟩

lemma *total-degree-poly-le-power*:
assumes *p*: *total-degree-poly-le p d*
shows *total-degree-poly-le* (*poly-power p k*) (*k* $*$ *d*)
⟨*proof*⟩

lemma *eval-poly-prod*:
eval-poly α (*poly-prod ps*) = *prod-list* (*map* (*eval-poly* α) *ps*)
⟨*proof*⟩

lemma *variables-poly-prod*:
assumes *list-all* (*variables-poly V*) *ps*

shows *variables-poly V (poly-prod ps)*
⟨*proof*⟩

lemma *total-degree-poly-prod*:
assumes *list-all2 total-degree-poly-le ps ds*
shows *total-degree-poly-le (poly-prod ps) (sum-list ds)*
⟨*proof*⟩

lemma *eval-indicator-factor*:
eval-poly α *(indicator-factor q p)* = *1* − *eval-poly* α *p* ^ *(q* − *1)*
⟨*proof*⟩

lemma *eval-indicator-poly*:
eval-poly α *(indicator-poly q ps)* =
prod-list (map (λ*p. 1* − *eval-poly* α *p* ^ *(q* − *1)) ps)*
⟨*proof*⟩

lemma *variables-indicator-factor*:
assumes *variables-poly V p*
shows *variables-poly V (indicator-factor q p)*
⟨*proof*⟩

lemma *variables-indicator-poly*:
assumes *list-all (variables-poly V) ps*
shows *variables-poly V (indicator-poly q ps)*
⟨*proof*⟩

lemma *total-degree-indicator-factor*:
fixes *p* :: *('v::linorder, 'a::comm-ring-1) poly*
assumes *p*: *total-degree-poly-le p d*
shows *total-degree-poly-le (indicator-factor q p) ((q* − *1)* ∗ *d)*
⟨*proof*⟩

lemma *total-degree-indicator-poly*:
assumes *list-all2 total-degree-poly-le ps ds*
shows *total-degree-poly-le (indicator-poly q ps) ((q* − *1)* ∗ *sum-list ds)*
⟨*proof*⟩

1.2 Finite-field power sums

lemma *finite-field-card-gt-one*:
1 < *card (UNIV* :: *'a::finite-field set)*
⟨*proof*⟩

lemma *finite-field-power-card-minus-one*:
fixes *x* :: *'a::finite-field*
assumes *x* ≠ *0*
shows *x* ^ *(card (UNIV* :: *'a set)* − *1)* = *1*
⟨*proof*⟩

lemma *finite-field-exists-power-ne-one*:
  fixes k :: *nat*
  assumes *k-pos*: $0 < k$
  assumes *k-lt*: $k < card\ (UNIV :: 'a::finite\text{-}field\ set) - 1$
  shows $\exists\, a::'a.\ a \neq 0 \wedge a \mathbin{\hat{}} k \neq 1$
⟨*proof*⟩

lemma *finite-field-power-sum-less*:
  assumes *k-lt*: $k < card\ (UNIV :: 'a::finite\text{-}field\ set) - 1$
  shows $(\sum x{\in}UNIV.\ x \mathbin{\hat{}} k :: 'a) = 0$
⟨*proof*⟩

lemma *eval-monom-list-prod-sum-var*:
  fixes $\alpha :: 'v::linorder \Rightarrow 'a::comm\text{-}semiring\text{-}1$
  assumes *finite*: *finite V* **and** *vars*: $fst\ `\ set\ m \subseteq V$
  shows $eval\text{-}monom\text{-}list\ \alpha\ m =$
    $(\prod x{\in}V.\ \alpha\ x \mathbin{\hat{}} sum\text{-}var\text{-}list\ m\ x)$
  ⟨*proof*⟩

lemma *eval-monom-prod-sum-var*:
  fixes $\alpha :: 'v::linorder \Rightarrow 'a::comm\text{-}semiring\text{-}1$
  assumes *finite V* $monom\text{-}vars\ m \subseteq V$
  shows $eval\text{-}monom\ \alpha\ m = (\prod x{\in}V.\ \alpha\ x \mathbin{\hat{}} sum\text{-}var\ m\ x)$
  ⟨*proof*⟩

lemma *monom-list-degree-sum-var*:
  fixes $m :: 'v::linorder\ monom\text{-}list$
  assumes *finite V* $fst\ `\ set\ m \subseteq V$
  shows $monom\text{-}list\text{-}degree\ m = (\sum x{\in}V.\ sum\text{-}var\text{-}list\ m\ x)$
  ⟨*proof*⟩

lemma *monom-degree-sum-var*:
  fixes $m :: 'v::linorder\ monom$
  assumes *finite V* $monom\text{-}vars\ m \subseteq V$
  shows $monom\text{-}degree\ m = (\sum x{\in}V.\ sum\text{-}var\ m\ x)$
  ⟨*proof*⟩

lemma *sum-monom-assignments*:
  fixes $m :: 'v::linorder\ monom$
  assumes *finite*: *finite V* **and** *vars*: $monom\text{-}vars\ m \subseteq V$
  shows $(\sum \alpha{\in}PiE\ V\ (\lambda\text{-}.\ UNIV :: 'a::finite\text{-}field\ set).$
    $eval\text{-}monom\ \alpha\ m) =$
   $(\prod x{\in}V.\ \sum y{\in}(UNIV :: 'a\ set).\ y \mathbin{\hat{}} sum\text{-}var\ m\ x)$
⟨*proof*⟩

lemma *exists-variable-degree-lt*:
  assumes *finite*: *finite V*
  assumes $(\sum x{\in}V.\ f\ x) < card\ V * k$

shows $\exists x \in V.\ f\ x < k$
⟨*proof*⟩

lemma *monomial-sum-zero-if-degree-lt*:
fixes m :: *'v::linorder monom*
assumes *finite*: *finite V* **and** *vars*: *monom-vars m* $\subseteq$ *V*
assumes *deg*:
monom-degree m <
card V $*$ (*card* (*UNIV* :: *'a::finite-field set*) − *1*)
shows ($\sum \alpha \in$*PiE V* (λ-. *UNIV* :: *'a set*). *eval-monom* α *m*) = *0*
⟨*proof*⟩

lemma *finite-field-sum-poly-zero*:
assumes *finite*: *finite V* **and** *vars*: *variables-poly V p*
assumes *deg*:
total-degree-poly-less p
(*card V* $*$ (*card* (*UNIV* :: *'a::finite-field set*) − *1*))
shows ($\sum \alpha \in$*PiE V* (λ-. *UNIV* :: *'a set*). *eval-poly* α *p*) = *0*
⟨*proof*⟩

1.3 The theorem

lemma *finite-field-indicator-factor-value*:
fixes *y* :: *'a::finite-field*
shows *1* − *y* ^(*card* (*UNIV* :: *'a set*) − *1*) = *of-bool* (*y* = *0*)
⟨*proof*⟩

lemma *prod-list-of-bool-list-all*:
prod-list (*map* (λ*x*. *of-bool* (*P x*) :: *'a::comm-semiring-1*) *xs*) =
of-bool (*list-all P xs*)
⟨*proof*⟩

lemma *eval-indicator-poly-value*:
fixes *ps* :: (*'v::linorder*, *'a::finite-field*) *poly list*
shows *eval-poly* α (*indicator-poly* (*card* (*UNIV* :: *'a set*)) *ps*) =
of-bool (*list-all* (λ*p*. *eval-poly* α *p* = *0*) *ps*)
⟨*proof*⟩

lemma *list-all2-variables*:
assumes
list-all2 (λ*p d*. *variables-poly V p* $\wedge$ *total-degree-poly-le p d*) *ps ds*
shows *list-all* (*variables-poly V*) *ps*
⟨*proof*⟩

lemma *list-all2-degrees*:
assumes
list-all2 (λ*p d*. *variables-poly V p* $\wedge$ *total-degree-poly-le p d*) *ps ds*
shows *list-all2 total-degree-poly-le ps ds*
⟨*proof*⟩

theorem *chevalley-warning*:
  fixes V :: $'v$::*linorder set*
    and *ps* :: ($'v$, $'a$::*finite-field*) *poly list*
  assumes *finite*: *finite V*
  assumes *polys*:
    list-all2 (λp d. *variables-poly V p* $\wedge$ *total-degree-poly-le p d*) *ps ds*
  assumes *degree-sum*: *sum-list ds* $<$ *card V*
  shows *CHAR*($'a$) *dvd card* (*common-zeros V ps*)
⟨*proof*⟩

corollary *chevalley-warning-finite-type*:
  fixes *ps* :: ($'v$::{*finite*,*linorder*}, $'a$::*finite-field*) *poly list*
  assumes *polys*: *list-all2 total-degree-poly-le ps ds*
  assumes *degree-sum*: *sum-list ds* $<$ *card* (*UNIV* :: $'v$ *set*)
  shows *CHAR*($'a$) *dvd card* (*common-zeros UNIV ps*)
⟨*proof*⟩

end

References

[1] C. Chevalley. Démonstration d'une hypothèse de M. Artin. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):73–75, 1935. DOI: `https://doi.org/10.1007/BF02940714`.

[2] C. Sternagel, R. Thiemann, A. Maletzky, F. Immler, F. Haftmann, A. Lochbihler, and A. Bentkamp. Executable multivariate polynomials. *Archive of Formal Proofs*, August 2010. `https://isa-afp.org/entries/Polynomials.html`, Formal proof development.

[3] E. Warning. Bemerkung zur vorstehenden Arbeit von Herrn Chevalley. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 11(1):76–83, 1935. DOI: `https://doi.org/10.1007/BF02940715`.