Vosper's Theorem and the Cauchy–Davenport Theorem via the Polynomial Method

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

June 12, 2026

Abstract

This entry formalizes the Cauchy–Davenport lower bound and Vosper's theorem for sumsets over finite fields of prime cardinality. It combines a polynomial-method proof of Cauchy–Davenport with a Davenport-transform proof of Vosper's classification of the nontrivial equality case, and it provides modular-representative lemmas connecting the abstract prime-field results to normalized representatives of $\mathbb{Z}/p\mathbb{Z}$. AI assistance was used for proof engineering. The final definitions, statements, and proofs are checked by Isabelle.

1 Overview

For nonempty finite subsets A and B of a prime field $\mathbb{F}_p$, the Cauchy–Davenport theorem gives the lower bound

$$|A + B| \geq \min(p, |A| + |B| - 1).$$

Vosper's theorem refines the extremal case: if $|A|, |B| \geq 2$, the sumset is not all of $\mathbb{F}_p$, and equality holds in the Cauchy–Davenport bound, then both sets are arithmetic progressions with a common difference. These results are standard cornerstones of additive combinatorics.

2 Formalization

The Isabelle/HOL development first establishes generic finite-sumset facts for translations, complements, and cardinalities. The proof of Cauchy–Davenport is then carried out by the polynomial method, using ordinary Isabelle polynomials to package the bivariate degree argument over finite fields of prime cardinality. Vosper's theorem is proved via Davenport transforms: the formalization derives one-sided progression results under normalization hypotheses and combines them to obtain the final shared-difference classification. The entry also supplies concrete modular-translation facts over normalized representatives of $\mathbb{Z}/p\mathbb{Z}$.

3 Relation to existing AFP entries

The Cauchy–Davenport bound has been formalized in the AFP before. The entry *Kneser's Theorem and the Cauchy–Davenport Theorem* [3] derives it as a corollary of Kneser's addition theorem for abelian groups, and *A Generalization of the Cauchy–Davenport Theorem* [2] formalizes DeVos's generalization to (possibly non-abelian) groups. Both developments are set in an abstract group structure and build on further AFP material.

The present entry is intended to complement, not duplicate, this work. Its principal new contribution is *Vosper's theorem*, the critical-pair (inverse) theorem characterizing the equality case of the Cauchy–Davenport bound, which neither of the entries above provides. The Cauchy–Davenport bound itself is established here by a different route, the *polynomial method*

(Combinatorial Nullstellensatz) following Alon, Nathanson, and Ruzsa [1]: a single explicit polynomial is shown to be non-vanishing on a grid built from A and B. To the best of our knowledge this proof of Cauchy–Davenport is not otherwise present in the AFP. It is carried out directly for finite fields of prime cardinality, modelled by the `finite_field` type class together with a primality hypothesis on the field's cardinality, and it depends only on the Isabelle/HOL distribution (`HOL-Computational_Algebra` and `HOL-Number_Theory`). Keeping this self-contained prime-field bound as the base case lets the Vosper development proceed without translating between the type-class and locale-based group representations.

4 Sources

The Cauchy–Davenport proof follows the polynomial-method perspective of Alon, Nathanson, and Ruzsa [1]. The additive-combinatorics background follows standard sources such as Nathanson [4] and Tao and Vu [5]. The inverse theorem formalized here is Vosper's critical-pair theorem for groups of prime order [6].

Contents

theory *Sumset-Basics*
imports *Main*
begin

definition *sumset* :: ('*a*::*comm-monoid-add*) *set* => '*a set* => '*a set* **where**
sumset A B = {*x*. ∃ *a*∈*A*. ∃ *b*∈*B*. *x* = *a* + *b*}

lemma *sumset-iff*:
x ∈ *sumset A B* ⟷ (∃ *a*∈*A*. ∃ *b*∈*B*. *x* = *a* + *b*)
⟨*proof*⟩

lemma *sumsetI* [*intro*]:

assumes $a \in A$ $b \in B$
shows $a + b \in sumset\ A\ B$
⟨*proof*⟩

lemma *sumsetE* [*elim*]:
assumes $x \in sumset\ A\ B$
obtains $a\ b$ **where** $a \in A$ $b \in B$ $x = a + b$
⟨*proof*⟩

lemma *sumset-as-image*:
$sumset\ A\ B = case\text{-}prod\ (+)\ `\ (A \times B)$
⟨*proof*⟩

lemma *sumset-commute*:
$sumset\ A\ B = sumset\ B\ A$
⟨*proof*⟩

lemma *empty-sumset-left* [*simp*]:
$sumset\ \{\}\ B = \{\}$
⟨*proof*⟩

lemma *empty-sumset-right* [*simp*]:
$sumset\ A\ \{\} = \{\}$
⟨*proof*⟩

lemma *sumset-assoc*:
fixes $A\ B\ C :: ('a{::}comm\text{-}monoid\text{-}add)\ set$
shows $sumset\ (sumset\ A\ B)\ C = sumset\ A\ (sumset\ B\ C)$
⟨*proof*⟩

lemma *sumset-zero-right* [*simp*]:
fixes $A :: ('a{::}comm\text{-}monoid\text{-}add)\ set$
shows $sumset\ A\ \{0\} = A$
⟨*proof*⟩

lemma *sumset-zero-left* [*simp*]:
fixes $A :: ('a{::}comm\text{-}monoid\text{-}add)\ set$
shows $sumset\ \{0\}\ A = A$
⟨*proof*⟩

definition *translate* :: $'a{::}comm\text{-}monoid\text{-}add \Rightarrow 'a\ set \Rightarrow 'a\ set$ **where**
$translate\ c\ A = ((+)\ c)\ `\ A$

lemma *translate-iff*:
$x \in translate\ c\ A \longleftrightarrow (\exists\, a{\in}A.\ x = c + a)$
⟨*proof*⟩

lemma *translate-empty* [*simp*]:
$translate\ c\ \{\} = \{\}$
⟨*proof*⟩

lemma *translate-compose*:
$translate\ c\ (translate\ d\ A) = translate\ (c + d)\ A$
⟨*proof*⟩

lemma *translate-as-sumset*:
$translate\ c\ A = sumset\ \{c\}\ A$
⟨*proof*⟩

lemma *finite-sumset* [*intro*]:
assumes *finite A finite B*
shows *finite* (*sumset A B*)
⟨*proof*⟩

lemma *card-sumset-le*:
assumes *finite A finite B*
shows *card* (*sumset A B*) ≤ *card A* ∗ *card B*
⟨*proof*⟩

lemma *card-translate-eq*:
fixes *A* :: ('*a*::*cancel-comm-monoid-add*) *set*
assumes *finite A*
shows *card* (*translate c A*) = *card A*
⟨*proof*⟩

lemma *translate-Compl*:
fixes *A* :: ('*a*::*ab-group-add*) *set*
shows *translate c* (*UNIV* − *A*) = *UNIV* − *translate c A*
⟨*proof*⟩

lemma *card-complement-translate-eq*:
fixes *A* :: ('*a*::{*finite*,*ab-group-add*}) *set*
shows *card* (*UNIV* − *translate c A*) = *card* (*UNIV* − *A*)
⟨*proof*⟩

lemma *sumset-pair-zero*:
fixes *A* :: ('*a*::*comm-monoid-add*) *set*
shows *sumset A* {*0*, *d*} = *A* ∪ *translate d A*
⟨*proof*⟩

end
theory *Modular-Sumsets*
imports
Sumset-Basics
HOL−Number-Theory.Number-Theory
begin

5 Modular sumsets over integer representatives of $\mathbb{Z}/p\mathbb{Z}$

This theory provides a concrete, self-contained companion to the abstract prime-field development. It works with the canonical integer representatives *0*, ..., *p* − *1* of $\mathbb{Z}/p\mathbb{Z}$ and records the basic cardinality behaviour of modular translations and singleton sumsets in that representation.

definition *residues* :: *nat* => *int set* **where**
residues p = {*0*..<*int p*}

definition *mod-translate* :: *nat* => *int* => *int set* => *int set* **where**
mod-translate p c A = ((λ*x*. (*x* + *c*) *mod int p*) ‘ *A*)

definition *mod-sumset* :: *nat* => *int set* => *int set* => *int set* **where**
mod-sumset p A B = {*x*. ∃ *a*∈*A*. ∃ *b*∈*B*. *x* = (*a* + *b*) *mod int p*}

lemma *residues-finite* [*simp*]:
finite (*residues p*)
⟨*proof*⟩

lemma *card-residues* [*simp*]:
assumes $0 < p$
shows *card* (*residues p*) = *p*
⟨*proof*⟩

lemma *mod-translate-iff*:
$x \in$ *mod-translate p c A* $\longleftrightarrow$ $(\exists\, a{\in}A.\ x = (a + c)$ *mod int p*$)$
⟨*proof*⟩

lemma *mod-sumset-iff*:
$x \in$ *mod-sumset p A B* $\longleftrightarrow$ $(\exists\, a{\in}A.\ \exists\, b{\in}B.\ x = (a + b)$ *mod int p*$)$
⟨*proof*⟩

lemma *mod-sumset-as-image*:
mod-sumset p A B = $((\lambda x.\ x$ *mod int p*$)$ ‘ *sumset A B*$)$
⟨*proof*⟩

lemma *mod-translate-subset-residues*:
assumes $0 < p$
shows *mod-translate p c A* ⊆ *residues p*
⟨*proof*⟩

lemma *mod-sumset-subset-residues*:
assumes $0 < p$
shows *mod-sumset p A B* ⊆ *residues p*
⟨*proof*⟩

lemma *mod-sumset-singleton-left*:
mod-sumset p {*a*} *B* = *mod-translate p a B*
⟨*proof*⟩

lemma *mod-sumset-singleton-right*:
mod-sumset p A {*b*} = *mod-translate p b A*
⟨*proof*⟩

lemma *mod-translate-inj-on-residues*:
assumes $0 < p$
shows *inj-on* $(\lambda x.\ (x + c)$ *mod int p*$)$ (*residues p*)
⟨*proof*⟩

lemma *card-mod-translate-eq*:
assumes $0 < p$
assumes $A \subseteq$ *residues p*
shows *card* (*mod-translate p c A*) = *card A*
⟨*proof*⟩

lemma *card-mod-sumset-singleton-left*:
assumes $0 < p$
assumes $B \subseteq$ *residues p*
shows *card* (*mod-sumset p* {*a*} *B*) = *card B*
⟨*proof*⟩

lemma *card-mod-sumset-singleton-right*:
assumes $0 < p$
assumes $A \subseteq$ *residues p*
shows *card* (*mod-sumset p A* {*b*}) = *card A*
⟨*proof*⟩

These lemmas complement the abstract prime-field proofs of Cauchy-Davenport and Vosper by

providing a convenient API for expressing the corresponding cardinality facts over the standard residue representatives of prime cyclic groups.

end
theory *Cauchy-Davenport-Prime-Field*
imports
Sumset-Basics
HOL−Computational-Algebra.Computational-Algebra
HOL−Number-Theory.Number-Theory
begin

6 Cauchy-Davenport over prime fields

This theory proves the prime-field Cauchy-Davenport theorem by a polynomial-method argument. The core technical work packages the bivariate degree bookkeeping needed to show that a suitably chosen polynomial cannot vanish on too large a grid unless the expected lower bound holds.

6.1 Polynomial infrastructure

definition *total-degree-le* :: *'a::comm-semiring-1 poly poly ⇒ nat ⇒ bool* **where**
total-degree-le P d ⟷
(∀ i. if i ≤ d then degree (poly.coeff P i) ≤ d − i else poly.coeff P i = 0)

definition *sum-factor* :: *'a::comm-ring-1 ⇒ 'a poly poly* **where**
sum-factor c = [: [:−c, 1:], [:1:] :]

lemma *coeff-poly-const*:
fixes *P* :: *'a::comm-semiring-1 poly poly*
shows *poly.coeff (poly P [:a:]) j =*
*(∑ i≤degree P. poly.coeff (poly.coeff P i) j * a ^ i)*
⟨*proof*⟩

lemma *total-degree-le-eval-const*:
fixes *P* :: *'a::comm-semiring-1 poly poly*
assumes *td*: *total-degree-le P d*
shows *degree (poly P [:a:]) ≤ d*
⟨*proof*⟩

lemma *total-degree-le-eval-const-top*:
fixes *P* :: *'a::comm-semiring-1 poly poly*
assumes *td*: *total-degree-le P d*
shows *poly.coeff (poly P [:a:]) d = poly.coeff (poly.coeff P 0) d*
⟨*proof*⟩

lemma *coeff-sum-factor-mult-0*:
*poly.coeff (sum-factor c * P) 0 = [:−c, 1:] * poly.coeff P 0*
⟨*proof*⟩

lemma *coeff-sum-factor-mult-Suc*:
*poly.coeff (sum-factor c * P) (Suc i) =*
*[:−c, 1:] * poly.coeff P (Suc i) + poly.coeff P i*
⟨*proof*⟩

lemma *total-degree-le-sum-factor-mult*:
fixes *P* :: *'a::field poly poly*
assumes *td*: *total-degree-le P d*
shows *total-degree-le (sum-factor c * P) (Suc d)*
⟨*proof*⟩

lemma *total-degree-le-prod-sum-factor*:
fixes $C :: 'a$::*field set*
assumes *finite C*
shows *total-degree-le* (*prod sum-factor C*) (*card C*)
⟨*proof*⟩

lemma *coeff-coeff-sum-factor-mult-top*:
fixes $P :: 'a$::*field poly poly*
assumes *td*: *total-degree-le P d*
assumes *ij*: $i + j = Suc\ d$
shows *poly.coeff* (*poly.coeff* (*sum-factor c* ∗ *P*) *i*) *j* =
(*if i* = *0 then 0 else poly.coeff* (*poly.coeff P* (*i* − *1*)) *j*) +
(*if j* = *0 then 0 else poly.coeff* (*poly.coeff P i*) (*j* − *1*))
⟨*proof*⟩

lemma *coeff-coeff-prod-sum-factor-top*:
fixes $C :: 'a$::*field set*
assumes *fin*: *finite C*
assumes *ij*: $i + j = card\ C$
shows *poly.coeff* (*poly.coeff* (*prod sum-factor C*) *i*) *j* = *of-nat* (*card C choose i*)
⟨*proof*⟩

lemma *poly-sum-factor*:
poly (*poly* (*sum-factor c*) [:*x*:]) *y* = *x* + *y* − *c*
⟨*proof*⟩

lemma *prime-not-dvd-binomial*:
assumes *prime p k* ≤ *n n* < *p*
shows ¬ *p dvd* (*n choose k*)
⟨*proof*⟩

lemma *finite-subset-with-card*:
assumes *finite U n* ≤ *card U*
shows ∃ *V*. *V* ⊆ *U* ∧ *card V* = *n*
⟨*proof*⟩

lemma *prime-card-eq-char*:
assumes *prime-card*: *prime* (*card* (*UNIV* :: $'a$::*finite-field set*))
shows *CHAR*($'a$) = *card* (*UNIV* :: $'a$ *set*)
⟨*proof*⟩

lemma *of-nat-binomial-ne-zero*:
assumes *prime-card*: *prime* (*card* (*UNIV* :: $'a$::*finite-field set*))
assumes *k* ≤ *n n* < *card* (*UNIV* :: $'a$ *set*)
shows *of-nat* (*n choose k*) ≠ (*0* :: $'a$)
⟨*proof*⟩

lemma *coeff-x-factor-plus-const-0*:
poly.coeff ([:−[:*a*:], *1*:] ∗ *Q* + [:*R*:]) *0* = − [:*a*:] ∗ *poly.coeff Q 0* + *R*
⟨*proof*⟩

lemma *coeff-x-factor-plus-const-Suc*:
poly.coeff ([:−[:*a*:], *1*:] ∗ *Q* + [:*R*:]) (*Suc i*) =
− [:*a*:] ∗ *poly.coeff Q* (*Suc i*) + *poly.coeff Q i*
⟨*proof*⟩

lemma *total-degree-le-factor-out*:
fixes *P Q* :: $'a$::*field poly poly*

assumes *td*: *total-degree-le P* (*Suc d*)
assumes *decomp*: $P = [:-[:a:],\ 1:] * Q + [:R:]$
shows *total-degree-le Q d*
⟨*proof*⟩

lemma *coeff-coeff-x-factor-top*:
fixes $Q :: 'a::field\ poly\ poly$
assumes *td*: *total-degree-le Q d*
assumes *mn*: $m + n = d$
shows *poly.coeff* (*poly.coeff* ($[:-[:a:],\ 1:] * Q + [:R:]$) (*Suc m*)) *n* =
poly.coeff (*poly.coeff Q m*) *n*
⟨*proof*⟩

lemma *grid-nonzero-from-top-coeff*:
fixes $P :: 'a::field\ poly\ poly$
assumes *cardA*: *card A* = *Suc m*
assumes *cardB*: *card B* = *Suc n*
assumes *td*: *total-degree-le P* $(m + n)$
assumes *top*: *poly.coeff* (*poly.coeff P m*) $n \neq 0$
shows $\exists\, a{\in}A.\ \exists\, b{\in}B.$ *poly* (*poly P* $[:a:]$) $b \neq 0$
⟨*proof*⟩

lemma *sumset-eq-UNIV-if-large*:
fixes *A B* :: *'a::finite-field set*
assumes *card* (*UNIV* :: *'a set*) < *card A* + *card B* − *1*
assumes $A \neq \{\}$ $B \neq \{\}$
shows *sumset A B* = *UNIV*
⟨*proof*⟩

6.2 The main lower bound

The final statement is the abstract prime-field version of Cauchy-Davenport: for nonempty finite subsets of a field whose cardinality is prime, the sumset has size at least the expected truncated lower bound.

theorem *cauchy-davenport-prime-field*:
fixes *A B* :: *'a::finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes $A \neq \{\}$ $B \neq \{\}$
shows *card* (*sumset A B*) ≥ *min* (*card* (*UNIV* :: *'a set*)) (*card A* + *card B* − *1*)
⟨*proof*⟩

end
theory *Vosper-Support*
imports *Cauchy-Davenport-Prime-Field*
begin

7 Arithmetic progressions in prime fields

Vosper's theorem is formulated in terms of arithmetic progressions with a shared common difference. This theory isolates the corresponding progression notation, translation lemmas, and the overlap criterion that recovers a progression from near-invariance under translation.

7.1 Basic progression infrastructure

definition *ap-segment* :: *'a::semiring-1* ⇒ *'a* ⇒ *nat* ⇒ *'a set* **where**
ap-segment a d n = $(\lambda i.\ a +$ *of-nat i* $* d)$ ‘ $\{..<n\}$

definition *arithmetic-progression* :: *'a::semiring-1 set* ⇒ *bool* **where**
arithmetic-progression A ⟷ (∃ *a d n*. *A* = *ap-segment a d n*)

lemma *arithmetic-progressionI*:
A = *ap-segment a d n* ⟹ *arithmetic-progression A*
⟨*proof*⟩

lemma *arithmetic-progressionE*:
assumes *arithmetic-progression A*
obtains *a d n* **where** *A* = *ap-segment a d n*
⟨*proof*⟩

lemma *finite-ap-segment* [*simp*]:
finite (*ap-segment a d n*)
⟨*proof*⟩

lemma *ap-segment-empty* [*simp*]:
ap-segment a d 0 = {}
⟨*proof*⟩

lemma *ap-segment-zero-step*:
fixes *a* :: *'a::semiring-1*
assumes *0* < *n*
shows *ap-segment a 0 n* = {*a*}
⟨*proof*⟩

lemma *ap-segment-translate*:
fixes *a d c* :: *'a::semiring-1*
shows *translate c* (*ap-segment a d n*) = *ap-segment* (*c* + *a*) *d n*
⟨*proof*⟩

lemma *arithmetic-progression-translate* [*simp*]:
fixes *c* :: *'a::ring-1*
shows *arithmetic-progression* (*translate c A*) ⟷ *arithmetic-progression A*
⟨*proof*⟩

lemma *sumset-ap-segment-pair-suc*:
fixes *a d* :: *'a::semiring-1*
shows *sumset* (*ap-segment a d* (*Suc n*)) {*0*, *d*} = *ap-segment a d* (*Suc* (*Suc n*))
⟨*proof*⟩

lemma *of-nat-eq-below-prime-card*:
fixes *i j* :: *nat*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a::finite-field set*))
assumes *i-lt*: *i* < *card* (*UNIV* :: *'a set*)
assumes *j-lt*: *j* < *card* (*UNIV* :: *'a set*)
assumes *eq*: (*of-nat i* :: *'a*) = *of-nat j*
shows *i* = *j*
⟨*proof*⟩

lemma *inj-on-ap-segment-index*:
fixes *a d* :: *'a::finite-field*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a::finite-field set*))
assumes *d-nonzero*: *d* ≠ *0*
assumes *n-le*: *n* ≤ *card* (*UNIV* :: *'a set*)
shows *inj-on* (λ*i*. *a* + *of-nat i* ∗ *d*) {..<*n*}
⟨*proof*⟩

lemma *card-ap-segment*:

fixes *a d* :: *'a*::*finite-field*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a*::*finite-field set*))
assumes *d-nonzero*: $d \neq 0$
assumes *n-le*: $n \leq$ *card* (*UNIV* :: *'a set*)
shows *card* (*ap-segment a d n*) = *n*
⟨*proof*⟩

lemma *arithmetic-progression-obtain-card*:
fixes *A* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *ap*: *arithmetic-progression A*
assumes *card-ge2*: $2 \leq$ *card A*
assumes *small*: *card A* < *card* (*UNIV* :: *'a set*)
obtains *a d* **where** $d \neq 0$ *A* = *ap-segment a d* (*card A*)
⟨*proof*⟩

lemma *step-image-UNIV*:
fixes *a d* :: *'a*::*finite-field*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a*::*finite-field set*))
assumes *d-nonzero*: $d \neq 0$
shows (λ*i*. *a* + *of-nat i* ∗ *d*) ‘ {..<*card* (*UNIV* :: *'a set*)} = *UNIV*
⟨*proof*⟩

lemma *predecessor-closed-initial-segment*:
fixes *S* :: *nat set*
assumes *fin*: *finite S*
assumes *zero*: $0 \in S$
assumes *pred*: $\bigwedge k.$ *Suc k* $\in S \Longrightarrow k \in S$
shows *S* = {..<*card S*}
⟨*proof*⟩

7.2 Recovering a progression from maximal overlap

The following lemmas package the structural step used repeatedly in the Vosper proof: if a set is almost preserved by translation with a nonzero step, then it must itself be an arithmetic progression with that step.

lemma *ap-segment-from-maximal-shift-overlap*:
fixes *A* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *d-nonzero*: $d \neq 0$
assumes *small*: *card A* < *card* (*UNIV* :: *'a set*)
assumes *overlap*: *card* (*A* ∩ *translate d A*) = *card A* − *1*
obtains *a* **where** *A* = *ap-segment a d* (*card A*)
⟨*proof*⟩

lemma *arithmetic-progression-from-maximal-shift-overlap*:
fixes *A* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *d-nonzero*: $d \neq 0$
assumes *small*: *card A* < *card* (*UNIV* :: *'a set*)
assumes *overlap*: *card* (*A* ∩ *translate d A*) = *card A* − *1*
shows *arithmetic-progression A*
⟨*proof*⟩

end
theory *Vosper-Prime-Field*
imports *Vosper-Support*
begin

8 Vosper over prime fields

This theory formalizes Vosper's theorem in the prime-field setting. The proof is organized around Davenport transforms, first deriving one-sided progression results under a normalization hypothesis and then combining the normalized arguments to obtain the full shared-difference conclusion.

8.1 Davenport transforms and translation lemmas

definition *davenport-transform* ::
  'a::ring-1 set ⇒ 'a set ⇒ 'a ⇒ 'a set
where
  davenport-transform A B e = $\{b \in B.\ e - b \notin sumset\ A\ B\}$

definition *davenport-remainder* ::
  'a::ring-1 set ⇒ 'a set ⇒ 'a ⇒ 'a set
where
  davenport-remainder A B e = $\{b \in B.\ e - b \in sumset\ A\ B\}$

lemma *sumset-mono-left*:
  assumes $A \subseteq A'$
  shows *sumset A B* $\subseteq$ *sumset A' B*
  ⟨*proof*⟩

lemma *sumset-mono-right*:
  assumes $B \subseteq B'$
  shows *sumset A B* $\subseteq$ *sumset A B'*
  ⟨*proof*⟩

lemma *sumset-translate-right*:
  fixes *A B* :: (*'a::comm-monoid-add*) *set*
  shows *sumset A* (*translate c B*) = *translate c* (*sumset A B*)
⟨*proof*⟩

lemma *sumset-translate-left*:
  fixes *A B* :: (*'a::comm-monoid-add*) *set*
  shows *sumset* (*translate c A*) *B* = *translate c* (*sumset A B*)
⟨*proof*⟩

lemma *card-uminus-image-eq* [*simp*]:
  fixes *A* :: (*'a::ab-group-add*) *set*
  assumes *finite A*
  shows *card* ((*uminus*) ` *A*) = *card A*
  ⟨*proof*⟩

lemma *card-2-zero-obtain*:
  fixes *B* :: (*'a::zero*) *set*
  assumes *fin*: *finite B*
  assumes *zero-in*: *0* ∈ *B*
  assumes *card2*: *card B* = *2*
  obtains *d* **where** *d* ≠ *0 B* = {*0*, *d*}
⟨*proof*⟩

lemma *davenport-partition*:
  davenport-transform A B e ∪ *davenport-remainder A B e* = *B*
  davenport-transform A B e ∩ *davenport-remainder A B e* = {}
  ⟨*proof*⟩

lemma *davenport-transform-subset* [*simp*]:

davenport-transform A B e ⊆ *B*
⟨*proof*⟩

lemma *davenport-remainder-subset* [*simp*]:
davenport-remainder A B e ⊆ *B*
⟨*proof*⟩

lemma *card-lt-univ-if-not-univ*:
fixes *S* :: *'a*::*finite set*
assumes *S* ≠ *UNIV*
shows *card S* < *card* (*UNIV* :: *'a set*)
⟨*proof*⟩

8.2 One-sided progression results

The first main step shows that equality in Cauchy-Davenport, together with a normalization such as *0* ∈ *B*, forces the opposite set to be an arithmetic progression.

theorem *vosper-progression-left-zero*:
fixes *A B* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *zero-in-B*: *0* ∈ *B*
assumes *B-ge2*: *2* ≤ *card B*
assumes *comp-ge2*: *2* ≤ *card* (*UNIV* − *sumset A B*)
assumes *eq*: *card* (*sumset A B*) = *card A* + *card B* − *1*
shows *arithmetic-progression A*
⟨*proof*⟩

theorem *vosper-progression-right-zero*:
fixes *A B* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *zero-in-A*: *0* ∈ *A*
assumes *A-ge2*: *2* ≤ *card A*
assumes *comp-ge2*: *2* ≤ *card* (*UNIV* − *sumset A B*)
assumes *eq*: *card* (*sumset A B*) = *card A* + *card B* − *1*
shows *arithmetic-progression B*
⟨*proof*⟩

lemma *vosper-left-with-right-ap-zero-two*:
fixes *A* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *d-nonzero*: *d* ≠ *0*
assumes *A-ge2*: *2* ≤ *card A*
assumes *comp-ge2*: *2* ≤ *card* (*UNIV* − *sumset A* (*ap-segment 0 d* (*Suc* (*Suc 0*))))
assumes *eq*: *card* (*sumset A* (*ap-segment 0 d* (*Suc* (*Suc 0*)))) = *card A* + *Suc 0*
shows ∃ *a*. *A* = *ap-segment a d* (*card A*)
⟨*proof*⟩

lemma *vosper-left-with-right-ap-zero-aux*:
fixes *A* :: *'a*::*finite-field set*
assumes *prime-card*: *prime* (*card* (*UNIV* :: *'a set*))
assumes *A-ge2*: *2* ≤ *card A*
assumes *d-nonzero*: *d* ≠ *0*
assumes *comp-ge2*: *2* ≤ *card* (*UNIV* − *sumset A* (*ap-segment 0 d* (*Suc* (*Suc m*))))
assumes *eq*: *card* (*sumset A* (*ap-segment 0 d* (*Suc* (*Suc m*)))) = *card A* + *Suc m*
shows ∃ *a*. *A* = *ap-segment a d* (*card A*)
⟨*proof*⟩

theorem *vosper-left-with-right-ap-zero*:

fixes $A :: 'a::finite\text{-}field\ set$
assumes *prime-card*: $prime\ (card\ (UNIV :: 'a\ set))$
assumes *A-ge2*: $2 \leq card\ A$
assumes *d-nonzero*: $d \neq 0$
assumes *n-ge2*: $2 \leq n$
assumes *comp-ge2*: $2 \leq card\ (UNIV - sumset\ A\ (ap\text{-}segment\ 0\ d\ n))$
assumes *eq*: $card\ (sumset\ A\ (ap\text{-}segment\ 0\ d\ n)) = card\ A + n - 1$
shows $\exists a.\ A = ap\text{-}segment\ a\ d\ (card\ A)$
⟨*proof*⟩

8.3 The full Vosper theorem

After normalizing one side by translation and propagating the common difference through an explicit arithmetic progression on the other side, we recover the classical prime-field conclusion: both extremal sets are arithmetic progressions with the same nonzero step.

theorem *vosper-prime-field*:
fixes $A\ B :: 'a::finite\text{-}field\ set$
assumes *prime-card*: $prime\ (card\ (UNIV :: 'a\ set))$
assumes *A-ge2*: $2 \leq card\ A$
assumes *B-ge2*: $2 \leq card\ B$
assumes *comp-ge2*: $2 \leq card\ (UNIV - sumset\ A\ B)$
assumes *eq*: $card\ (sumset\ A\ B) = card\ A + card\ B - 1$
shows $\exists a\ b\ d.\ d \neq 0 \wedge A = ap\text{-}segment\ a\ d\ (card\ A) \wedge B = ap\text{-}segment\ b\ d\ (card\ B)$
⟨*proof*⟩

end
theory *Cauchy-Davenport-Entry*
imports
Modular-Sumsets
Cauchy-Davenport-Prime-Field
Vosper-Prime-Field
begin

9 Overview

This section records the scope of the combined Cauchy-Davenport and Vosper development. The session combines abstract prime-field sumset theorems with concrete representative lemmas for modular sumsets. The resulting theory stack connects generic finite-sumset infrastructure, the polynomial-method Cauchy-Davenport bound, and the final Vosper classification theorem.

end

References

[1] N. Alon, M. B. Nathanson, and I. Z. Ruzsa. The polynomial method and restricted sums of congruence classes. *Journal of Number Theory*, 56(2):404–417, 1996. DOI: https://doi.org/10.1006/jnth.1996.0029.

[2] M. Bakšys. A generalization of the cauchy–davenport theorem. *Archive of Formal Proofs*, Nov. 2024. https://isa-afp.org/entries/Generalized_Cauchy_Davenport.html, Formal proof development.

[3] M. Bakšys and A. Koutsoukou-Argyraki. Kneser's theorem and the cauchy–davenport theorem. *Archive of Formal Proofs*, Nov. 2022. https://isa-afp.org/entries/Kneser_Cauchy_Davenport.html, Formal proof development.

[4] M. B. Nathanson. *Additive Number Theory: Inverse Problems and the Geometry of Sumsets*, volume 165 of *Graduate Texts in Mathematics*. Springer-Verlag, 1996.

[5] T. Tao and V. H. Vu. *Additive Combinatorics*, volume 105 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2006. DOI: https://doi.org/10.1017/CBO9780511755149.

[6] A. G. Vosper. The critical pairs of subsets of a group of prime order. *Journal of the London Mathematical Society*, 31(2):200–205, 1956. DOI: https://doi.org/10.1112/jlms/s1-31.2.200.