Kneser's Theorem and the Cauchy–Davenport Theorem

Mantas Bakšys and Angeliki Koutsoukou-Argyraki
University of Cambridge
{mb2412, ak2110}@cam.ac.uk

December 1, 2022

Abstract

We formalise Kneser's Theorem in combinatorics [2, 3] following the proof from the 2014 paper "A short proof of Knesers addition theorem for abelian groups" by Matt DeVos [1]. We also show a strict version of Kneser's Theorem as well as the Cauchy–Davenport Theorem as a corollary of Kneser's Theorem.

Contents

Acknowledgements

Angeliki Koutsoukou-Argyraki is funded by the ERC Advanced Grant ALEXANDRIA (Project GA 742178) funded by the European Research Council and led by Lawrence C. Paulson (University of Cambridge, Department of Computer Science and Technology). Mantas Bakšys received funding for his internship supervised by Koutsoukou-Argyraki by the Cambridge Mathematics Placements (CMP) Programme and by the ALEXANDRIA Project. We wish to thank Manuel Eberl for his useful suggestion for treating induction when there is a type discrepancy between the induction hypothesis and the induction step.

1 Preliminaries

theory *Kneser-Cauchy-Davenport-preliminaries*
imports
Complex-Main
Pluennecke-Ruzsa-Inequality.Pluennecke-Ruzsa-Inequality
HOL−Number-Theory.Prime-Powers

begin

context *subgroup-of-group*

begin

interpretation *left*: *left-translations-of-group* **..**
interpretation *right*: *right-translations-of-group* **..**

interpretation *transformation-group left.translation ‘ H G* **..**

lemma *Right-Coset-eq-iff*:
assumes $x \in G$ **and** $y \in G$
shows $H \mid\cdot\ x = (H \mid\cdot\ y) \longleftrightarrow H \mid\cdot\ x \cap (H \mid\cdot\ y) \neq \{\}$
using *assms Right-Coset-is-orbit*
by (*metis Int-absorb orbit.disjoint orbit.natural.map-closed orbit.non-vacuous*)

end

context *additive-abelian-group*

begin

1.1 Elementary lemmas on sumsets

lemma *sumset-translate-eq-right*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** $x \in G$
shows (*sumset* A $\{x\}$ = *sumset* B $\{x\}$) $\longleftrightarrow$ $A = B$ **using** *assms*
by (*smt* (*verit*, *best*) *Diff-Int-distrib2 Diff-eq-empty-iff*
Int-Un-eq(*1*) *Int-absorb2 Un-Diff-cancel2 Un-commute insert-disjoint*(*2*)
subset-refl sumset-is-empty-iff sumsetdiff-sing)

lemma *sumset-translate-eq-left*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** $x \in G$
shows (*sumset* $\{x\}$ A = *sumset* $\{x\}$ B) $\longleftrightarrow$ $A = B$ **using** *assms*
by (*simp add*: *sumset-commute sumset-translate-eq-right*)

lemma *differenceset-translate-eq-right*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** $x \in G$
shows (*differenceset* A $\{x\}$ = *differenceset* B $\{x\}$) $\longleftrightarrow$ $A = B$ **using** *assms*

by (*metis Int-absorb2 differenceset-commute minus-minusset minusset-subset-carrier*

sumset-translate-eq-left)

lemma *differenceset-translate-eq-left*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** $x \in G$
shows (*differenceset* $\{x\}$ A = *differenceset* $\{x\}$ B) $\longleftrightarrow$ $A = B$ **using** *assms*
by (*metis differenceset-commute differenceset-translate-eq-right*)

lemma *sumset-inter-union-subset*:
sumset $(A \cap B)$ $(A \cup B) \subseteq$ *sumset* A B
by (*metis Int-Diff-Un Int-Un-eq*(*2*) *Un-subset-iff sumset-commute sumset-subset-Un*(*2*)

sumset-subset-Un2)

lemma *differenceset-group-eq*:
G = *differenceset* G G
using *equalityE minusset-eq minusset-triv subset-antisym sumset-D*(*1*) *sumset-subset-carrier*

sumset-mono image-mono Int-absorb **by** *metis*

lemma *card-sumset-singleton-subset-eq*:
assumes $a \in G$ **and** $A \subseteq G$
shows *card* (*sumset* $\{a\}$ A) = *card* A
using *assms card-sumset-singleton-eq card.infinite card-sumset-0-iff′ le-iff-inf sumset-commute*
by *metis*

lemma *card-differenceset-singleton-mem-eq*:
assumes $a \in G$ **and** $A \subseteq G$
shows *card* A = *card* (*differenceset* A $\{a\}$)
using *assms* **by** (*metis card-minusset′ card-sumset-singleton-subset-eq difference-set-commute*
minusset-subset-carrier)

lemma *card-singleton-differenceset-eq*:
assumes $a \in G$ **and** $A \subseteq G$
shows *card* A = *card* (*differenceset* $\{a\}$ A)
using *assms* **by** (*metis card-minusset′ card-sumset-singleton-subset-eq minus-set-subset-carrier*)

lemma *sumset-eq-Union-left*:
assumes $A \subseteq G$
shows *sumset* A B = ($\bigcup$ $a \in A$. *sumset* $\{a\}$ B)
proof
show *sumset* A $B \subseteq$ ($\bigcup$ $a \in A$. *sumset* $\{a\}$ B)
using *assms sumset.cases Int-absorb2 Int-iff UN-iff singletonI sumset.sumsetI*
by (*smt* (*verit*, *del-insts*) *subsetI*)
next

show ($\bigcup$ $a \in A$. *sumset* $\{a\}$ B) $\subseteq$ *sumset* A B
using *sumset* **by** *auto*
qed

lemma *sumset-eq-Union-right*:
assumes $B \subseteq G$
shows *sumset* A B = ($\bigcup$ $b \in B$. *sumset* A $\{b\}$)
using *assms sumset-commute sumset-eq-Union-left* **by** (*metis* (*no-types*, *lifting*) *Sup.SUP-cong*)

lemma *sumset-singletons-eq*:
assumes $a \in G$ **and** $b \in G$
shows *sumset* $\{a\}$ $\{b\}$ = $\{a \oplus b\}$
using *assms sumset.simps subset-antisym* **by** *auto*

lemma *sumset-eq-subset-differenceset*:
assumes $K \subseteq G$ **and** $K \neq \{\}$ **and** $A \subseteq G$ **and** *sumset* A K = *sumset* B K
shows $A \subseteq$ *differenceset* (*sumset* B K) K
proof
fix a **assume** *ha*: $a \in A$
obtain k **where** *hk*: $k \in K$ **using** *assms*(*2*) **by** *blast*
then have $a \oplus k \in$ *sumset* B K **using** *assms sumset.sumsetI ha* **by** *blast*
then have $a \oplus (k \ominus k) \in$ *differenceset* (*sumset* B K) K **using** *hk assms ha minusset.minussetI*
subset-iff sumset.sumsetI **by** (*smt* (*verit*) *associative composition-closed inverse-closed*)
then show $a \in$ *differenceset* (*sumset* B K) K **using** *hk ha subsetD assms right-unit*
by (*metis invertible invertible-right-inverse*)
qed

end

locale *subgroup-of-additive-abelian-group* =
subgroup-of-abelian-group H G ($\oplus$) $\mathbf{0}$ + *additive-abelian-group* G ($\oplus$) $\mathbf{0}$
for H G **and** *addition* (**infixl** $\oplus$ *65*) **and** *zero* ($\mathbf{0}$)

begin

notation *Left-Coset* (**infixl** $\cdot|$ *70*)

lemma *Left-Coset-eq-sumset*:
assumes $x \in G$
shows *sumset* $\{x\}$ H = $x \cdot| H$
using *assms Left-Coset-memI sumset.simps* **by** *fastforce*

lemma *sumset-subgroup-eq-iff*:
assumes $a \in G$ **and** $b \in G$

shows *sumset* $\{a\}$ *H* = *sumset* $\{b\}$ *H* $\longleftrightarrow$
(*sumset* $\{a\}$ *H*) $\cap$ (*sumset* $\{b\}$ *H*) $\neq \{\}$
using *Right-Coset-eq-iff assms Left-Coset-eq-sumset Left-equals-Right-coset* **by** *presburger*

lemma *card-divide-sumset*:
assumes $A \subseteq G$
shows *card H dvd card* (*sumset A H*)
proof(*cases finite H* $\wedge$ *finite A*)
case *hfin*: *True*
then have *hfinsum*: $\bigwedge$ *X*. $X \in ((\lambda\ a.\ sumset\ \{a\}\ H)\ `\ A) \Longrightarrow$ *finite X*
using *finite-sumset* **by** *force*
moreover have *pairwise disjnt* ((λ *a. sumset* $\{a\}$ *H*) ‘ *A*)
using *pairwise-imageI disjnt-def sumset-subgroup-eq-iff subset-eq assms* **by** (*smt* (*verit, best*))
moreover have *card H dvd sum card* ((λ *a. sumset* $\{a\}$ *H*) ‘ *A*)
proof(*intro dvd-sum*)
fix *X* **assume** $X \in (\lambda\ a.\ sumset\ \{a\}\ H)\ `\ A$
then show *card H dvd card X* **using** *dvd-refl*
using *Left-equals-Right-coset Right-Coset-cardinality assms Left-Coset-eq-sumset* **by** *auto*
qed
ultimately show *?thesis* **using** *assms sumset-eq-Union-left card-Union-disjoint* **by** *metis*
next
case *False*
then show *?thesis* **using** *assms card-sumset-0-iff* **by** (*metis card-eq-0-iff dvd-0-right sub subsetI*)
qed

lemma *sumset-subgroup-eq-Class-Union*:
assumes $A \subseteq G$
shows *sumset A H* = ($\bigcup$ (*Class* ‘ *A*))
proof
show *sumset A H* $\subseteq \bigcup$ (*Class* ‘ *A*)
proof
fix *x* **assume** $x \in$ *sumset A H*
then obtain *a b* **where** *ha*: $a \in A$ **and** $b \in H$ **and** $x = a \oplus b$
using *sumset.cases* **by** *blast*
then have $x \in$ *Class a* **using** *Left-Coset-Class-unit Left-Coset-eq-sumset assms* **by** *blast*
thus $x \in \bigcup$ (*Class* ‘ *A*) **using** *ha* **by** *blast*
qed
next
show $\bigcup$ (*Class* ‘ *A*) $\subseteq$ *sumset A H*
proof(*intro Union-least*)
fix *X* **assume** $X \in$ *Class* ‘ *A*
then obtain *a* **where** $a \in A$ **and** *X* = *Class a* **by** *blast*
moreover hence $\{a\} \subseteq A$ **by** *auto*

ultimately show $X \subseteq$ *sumset A H* **using** *Left-Coset-Class-unit Left-Coset-eq-sumset assms sumset-mono subset-refl in-mono* **by** *metis*
qed
qed

lemma *Class-image-sumset-subgroup-eq*:
assumes $A \subseteq G$
shows *Class ‘ (sumset A H) = Class ‘ A*
proof
show *Class ‘ sumset A H* $\subseteq$ *Class ‘ A*
proof
fix x **assume** $x \in$ *Class ‘ sumset A H*
then obtain c **where** *hc*: $c \in$ *sumset A H* **and** $x =$ *Class c* **by** *blast*
moreover obtain a b **where** *ha*: $a \in A$ **and** $b \in H$ **and** $c = a \oplus b$ **using** *hc sumset.cases*
by *blast*
ultimately show $x \in$ *Class ‘ A* **using** *ha Class-eq CongruenceI assms composition-closed*
sumset.cases Partition-def commutative image-eqI left-unit sub unit-closed
by *(smt (verit, ccfv-threshold) Block-self Class-cong Normal-def)*
qed
next
show *Class ‘ A* $\subseteq$ *Class ‘ sumset A H* **using** *assms right-unit subsetD subsetI sumset.sumsetI*
unit-closed **by** *(smt (verit, del-insts) image-subset-iff sub-unit-closed)*
qed

lemma *Class-cover-imp-subset-or-disj*:
assumes $A = (\bigcup\ (Class\ ‘\ C))$ **and** $x \in G$ **and** $C \subseteq G$
shows *Class x* $\subseteq A \vee$ *Class x* $\cap A = \{\}$
proof*(intro disjCI)*
assume *Class x* $\cap A \neq \{\}$
then obtain c **where** $c \in C$ **and** *Class x* $\cap$ *Class c* $\neq \{\}$ **using** *assms* **by** *blast*
then show *Class x* $\subseteq A$ **using** *assms not-disjoint-implies-equal Sup-upper imageI subset-iff*
by *blast*
qed

end

context *additive-abelian-group*

begin

1.2 Stabilizer and basic properties

We define the stabilizer or group of periods of a nonempty subset of an abelian group.

definition *stabilizer*::*'a set* $\Rightarrow$ *'a set* **where**

stabilizer S ≡ {*x* ∈ *G*. *sumset* {*x*} (*S* ∩ *G*) = *S* ∩ *G*}

lemma *stabilizer-is-subgroup*: **fixes** *S* :: *'a set*
  shows *subgroup* (*stabilizer S*) *G* (⊕) (**0**)
proof (*intro subgroupI*)
  show *stabilizer S* ⊆ *G* **using** *stabilizer-def* **by** *auto*
next
  show 0 ∈ *stabilizer S* **using** *stabilizer-def* **by** (*simp add*: *Int-absorb1 Int-commute*)
next
  fix *a b* **assume** *haS*: *a* ∈ *stabilizer S* **and** *hbS*: *b* ∈ *stabilizer S*
  then have *haG*: *a* ∈ *G* **and** *hbG*: *b* ∈ *G* **using** *stabilizer-def* **by** *auto*
  have *sumset* {*a* ⊕ *b*} (*S* ∩ *G*) = *sumset* {*a*} (*sumset* {*b*} (*S* ∩ *G*))
  proof
    show *sumset* {*a* ⊕ *b*} (*S* ∩ *G*) ⊆ *sumset* {*a*} (*sumset* {*b*} (*S* ∩ *G*)) **using** *haG* *hbG*
      empty-subsetI insert-subset subsetI sumset.simps sumset-assoc sumset-mono
      by *metis*
    show *sumset* {*a*} (*sumset* {*b*} (*S* ∩ *G*)) ⊆ *sumset* {*a* ⊕ *b*} (*S* ∩ *G*)
      using *empty-iff insert-iff sumset.simps sumset-assoc* **by** (*smt* (*verit*, *best*) *subsetI*)
  qed
  then show *a* ⊕ *b* ∈ *stabilizer S* **using** *haS hbS stabilizer-def* **by** *auto*
next
  fix *g* **assume** *g* ∈ *stabilizer S*
  thus *invertible g* **using** *stabilizer-def* **by** *auto*
next
  fix *g* **assume** *hgS*: *g* ∈ *stabilizer S*
  then have *hinvsum* : *inverse g* ⊕ *g* = **0** **using** *stabilizer-def* **by** *simp*
  have *sumset* {*inverse g*} (*sumset* {*g*} (*S* ∩ *G*)) = (*S* ∩ *G*)
  proof
    show *sumset* {*inverse g*} (*sumset* {*g*} (*S* ∩ *G*)) ⊆ (*S* ∩ *G*) **using**
      empty-iff insert-iff sumset.simps sumset-assoc subsetI left-unit hinvsum
      by (*smt* (*verit*, *ccfv-threshold*))
    show (*S* ∩ *G*) ⊆ *sumset* {*inverse g*} (*sumset* {*g*} (*S* ∩ *G*))
    proof
      fix *s* **assume** *hs*: *s* ∈ (*S* ∩ *G*)
      then have *inverse g* ⊕ *g* ⊕ *s* ∈ *sumset* {*inverse g*} (*sumset* {*g*} (*S* ∩ *G*)) **using**
        hgS stabilizer-def additive-abelian-group.sumset.sumsetI
      additive-abelian-group-axioms associative in-mono inverse-closed mem-Collect-eq *singletonI*
        by (*smt* (*z3*) *IntD2*)
      thus *s* ∈ *sumset* {*inverse g*} (*sumset* {*g*} (*S* ∩ *G*)) **using** *hinvsum hs*
        by *simp*
    qed
  qed
  thus *inverse g* ∈ *stabilizer S* **using** *hgS stabilizer-def* **by** *auto*
qed

interpretation *subgroup-of-additive-abelian-group stabilizer A G* (⊕) **0**
using *stabilizer-is-subgroup subgroup-of-abelian-group-def*
by (*metis abelian-group-axioms additive-abelian-group-axioms group-axioms subgroup-of-additive-abelian-group-def subgroup-of-group-def*)

lemma *zero-mem-stabilizer*: **0** ∈ *stabilizer A* **..**

lemma *stabilizer-is-nonempty*:
shows *stabilizer S* ≠ {}
using *sub-unit-closed* **by** *blast*

lemma *Left-Coset-eq-sumset-stabilizer*:
assumes *x* ∈ *G*
shows *sumset* {*x*} (*stabilizer B*) = *x* ·| (*stabilizer B*)
by (*simp add*: *Left-Coset-eq-sumset assms*)

lemma *stabilizer-subset-difference-singleton*:
assumes *S* ⊆ *G* **and** *s* ∈ *S*
shows *stabilizer S* ⊆ *differenceset S* {*s*}
proof
fix *x* **assume** *hx*: *x* ∈ *stabilizer S*
then obtain *t* **where** *ht*: *t* ∈ *S* **and** *x* ⊕ *s* = *t* **using** *assms stabilizer-def* **by** *blast*
then have *x* = *t* ⊖ *s* **using** *hx stabilizer-def assms associative*
by (*metis* (*no-types*, *lifting*) *in-mono inverse-closed invertible invertible-right-inverse sub sub.right-unit*)
thus *x* ∈ *differenceset S* {*s*} **using** *ht assms*
by (*simp add*: *minusset.minussetI subsetD sumset.sumsetI*)
qed

lemma *stabilizer-subset-singleton-difference*:
assumes *S* ⊆ *G* **and** *s* ∈ *S*
shows *stabilizer S* ⊆ *differenceset* {*s*} *S*
proof−
have *stabilizer S* ⊆ *minusset* (*stabilizer S*) **using** *assms Int-absorb2 minusset-eq*

subgroup.image-of-inverse submonoid.sub subset-eq
by (*smt* (*verit*) *stabilizer-is-subgroup stabilizer-subset-difference-singleton sumset-subset-carrier*)
moreover have *minusset* (*stabilizer S*) ⊆ *minusset* (*differenceset S* {*s*})
proof
fix *x* **assume** *hx1*: *x* ∈ *minusset* (*stabilizer S*)
then have *hx*: *inverse x* ∈ *stabilizer S*
by (*metis invertible invertible-inverse-inverse minusset.cases*)
then obtain *t* **where** *ht*: *t* ∈ *S* **and** *inverse x* ⊕ *s* = *t* **using** *assms stabilizer-def* **by** *blast*
then have *hx2*: *inverse x* = *t* ⊖ *s* **using** *hx stabilizer-def assms*
by (*smt* (*verit*, *ccfv-threshold*) *commutative in-mono inverse-closed invertible*

invertible-left-inverse2 sub)
thus $x \in$ *minusset* (*differenceset S* $\{s\}$) **using** *ht assms*
by (*smt* (*verit*, *best*) *hx1 additive-abelian-group.sumset.sumsetI additive-abelian-group-axioms*

inverse-closed invertible invertible-inverse-inverse minusset.cases minusset.minussetI
singletonI subset-iff)
qed
ultimately show *?thesis* **using** *differenceset-commute assms* **by** *blast*
qed

lemma *stabilizer-subset-nempty*:
assumes $S \neq \{\}$ **and** $S \subseteq G$
shows *stabilizer S* $\subseteq$ *differenceset S S*
proof
fix x **assume** *hx*: $x \in$ *stabilizer S*
then obtain s **where** *hs*: $s \in S \cap G$ **using** *assms* **by** *blast*
then have $x \oplus s \in S \cap G$ **using** *stabilizer-def assms hx mem-Collect-eq singletonI*

sumset.sumsetI sumset-Int-carrier **by** *blast*
then obtain t **where** *ht*: $t \in S$ **and** $x \oplus s = t$ **by** *blast*
then have $x = t \ominus s$ **using** *hx stabilizer-def assms*(*2*) *hs ht associative*
by (*metis IntD2 inverse-closed invertible invertible-right-inverse sub sub.right-unit*)
thus $x \in$ *differenceset S S* **using** *ht hs* **using** *assms*(*2*) **by** *blast*
qed

lemma *stabilizer-coset-subset*:
assumes $A \subseteq G$ **and** $x \in A$
shows *sumset* $\{x\}$ (*stabilizer A*) $\subseteq A$
proof
fix y **assume** $y \in$ *sumset* $\{x\}$ (*stabilizer A*)
moreover hence *stabilizer A* $\subseteq$ *differenceset A* $\{x\}$ **using** *assms*
stabilizer-subset-difference-singleton **by** *auto*
moreover have *sumset* $\{x\}$ (*differenceset A* $\{x\}$) $\subseteq A$
proof
fix z **assume** $z \in$ *sumset* $\{x\}$ (*differenceset A* $\{x\}$)
then obtain a **where** $a \in A$ **and** $z = x \oplus (a \ominus x)$
using *additive-abelian-group.sumset.cases additive-abelian-group-axioms singletonD*
minusset.cases assms subsetD **by** (*smt* (*verit*, *ccfv-SIG*))
thus $z \in A$ **using** *assms*
by (*metis additive-abelian-group.inverse-closed additive-abelian-group-axioms*
commutative in-mono invertible invertible-right-inverse2)
qed
ultimately show $y \in A$ **by** (*meson in-mono subset-singleton-iff sumset-mono*)

qed

lemma *stabilizer-subset-stabilizer-dvd*:

```
  assumes stabilizer A ⊆ stabilizer B
  shows card (stabilizer A) dvd card (stabilizer B)
proof(cases finite (stabilizer B))
  case hB: True
  interpret H: subgroup-of-group stabilizer A stabilizer B (⊕) 0
  proof(unfold-locales)
    show stabilizer A ⊆ stabilizer B using assms by blast
  next
    show ⋀a b. a ∈ stabilizer A ⟹ b ∈ stabilizer A ⟹ a ⊕ b ∈ stabilizer A
      using stabilizer-is-subgroup group-axioms by simp
  next
    show 0 ∈ stabilizer A using sub-unit-closed by blast
  qed
  show ?thesis using H.lagrange hB by auto
next
  case False
  then show ?thesis by simp
qed

lemma stabilizer-coset-Un:
  assumes A ⊆ G
  shows (⋃ x ∈ A. sumset {x} (stabilizer A)) = A
proof
  show (⋃x∈A. sumset {x} (stabilizer A)) ⊆ A
    using stabilizer-coset-subset assms by blast
next
  show A ⊆ (⋃x∈A. sumset {x} (stabilizer A))
  proof
    fix x assume hx: x ∈ A
    then have x ∈ sumset {x} (stabilizer A) using sub-unit-closed assms
      by (metis in-mono right-unit singletonI sumset.sumsetI unit-closed)
    thus x ∈ (⋃x∈A. sumset {x} (stabilizer A)) using hx by blast
  qed
qed

lemma stabilizer-empty: stabilizer {} = G
  using sumset-empty Int-empty-left stabilizer-def subset-antisym
 by (smt (verit, best) mem-Collect-eq subsetI sumset-Int-carrier-eq(1) sumset-commute)

lemma stabilizer-finite:
  assumes S ⊆ G and S ≠ {} and finite S
  shows finite (stabilizer S)
  using stabilizer-subset-nempty assms
  by (meson finite-minusset finite-sumset rev-finite-subset)

lemma stabilizer-subset-group:
  shows stabilizer S ⊆ G using stabilizer-def by blast

lemma sumset-stabilizer-eq-iff:
```

assumes $a \in G$ **and** $b \in G$
shows *sumset* $\{a\}$ (*stabilizer* A) = *sumset* $\{b\}$ (*stabilizer* A) $\longleftrightarrow$
(*sumset* $\{a\}$ (*stabilizer* A)) $\cap$ (*sumset* $\{b\}$ (*stabilizer* A)) $\neq \{\}$
by (*simp add*: *assms sumset-subgroup-eq-iff*)

lemma *sumset-stabilizer-eq-Class-Union*:
assumes $A \subseteq G$
shows *sumset* A (*stabilizer* B) = ($\bigcup$ (*Class* B ' A))
by (*simp add*: *assms sumset-subgroup-eq-Class-Union*)

lemma *card-stabilizer-divide-sumset*:
assumes $A \subseteq G$
shows *card* (*stabilizer* B) *dvd card* (*sumset* A (*stabilizer* B))
by (*simp add*: *assms card-divide-sumset*)

lemma *Class-image-sumset-stabilizer-eq*:
assumes $A \subseteq G$
shows *Class* B ' (*sumset* A (*stabilizer* B)) = *Class* B ' A
by (*simp add*: *Class-image-sumset-subgroup-eq assms*)

lemma *Class-cover-imp-subset-or-disj*:
assumes $A = (\bigcup$ (*Class* B ' C)) **and** $x \in G$ **and** $C \subseteq G$
shows *Class* B $x \subseteq A \lor$ *Class* B $x \cap A = \{\}$
by (*simp add*: *Class-cover-imp-subset-or-disj assms*)

lemma *stabilizer-sumset-disjoint*:
fixes *S1 S2* :: *'a set*
assumes *stabilizer S1* $\cap$ *stabilizer S2* = $\{\mathbf{0}\}$ **and** *S1* $\subseteq G$ **and** *S2* $\subseteq G$
and *finite S1* **and** *finite S2* **and** *S1* $\neq \{\}$ **and** *S2* $\neq \{\}$
shows *card* (*sumset* (*stabilizer S1*) (*stabilizer S2*)) =
card (*stabilizer S1*) $*$ *card* (*stabilizer S2*)
proof−
have *inj-on* : *inj-on* (λ (a, b). $a \oplus b$) (*stabilizer S1* $\times$ *stabilizer S2*)
proof(*intro inj-onI*)
fix x y **assume** $x \in$ *stabilizer S1* $\times$ *stabilizer S2* **and** $y \in$ *stabilizer S1* $\times$ *stabilizer S2* **and**
(*case* x *of* (a, b) $\Rightarrow a \oplus b$) = (*case* y *of* (a, b) $\Rightarrow a \oplus b$)
then obtain a b c d **where** *hx*: $x = (a, b)$ **and** *hy*: $y = (c, d)$ **and** *ha*: $a \in$ *stabilizer S1* **and**
hb: $b \in$ *stabilizer S2* **and** *hc*: $c \in$ *stabilizer S1* **and** *hd*: $d \in$ *stabilizer S2* **and**
habcd: $a \oplus b = c \oplus d$ **by** *auto*
then have *haG*: $a \in G$ **using** *stabilizer-def* **by** *blast*
have *hbG*: $b \in G$ **using** *hb stabilizer-def* **by** *blast*
have *hcG*: $c \in G$ **using** *hc stabilizer-def* **by** *blast*
have *hdG*: $d \in G$ **using** *hd stabilizer-def* **by** *blast*
then have $a \ominus c = d \ominus b$ **using** *habcd haG hbG hcG hdG*
by (*metis* (*full-types*) *associative commutative composition-closed inverse-equality invertible*
invertible-def invertible-left-inverse2)

moreover have *a* ⊖ *c* ∈ *stabilizer S1* **using** *ha hc stabilizer-is-subgroup subgroup.axioms*(*1*) *submonoid.sub-composition-closed*
by (*metis group.invertible group-axioms hcG subgroup.subgroup-inverse-iff*)
moreover have *d* ⊖ *b* ∈ *stabilizer S2* **using** *hd hb stabilizer-is-subgroup subgroup.axioms*(*1*) *submonoid.sub-composition-closed*
by (*metis group.invertible group-axioms hbG subgroup.subgroup-inverse-iff*)
ultimately have *a* ⊖ *c* = **0 and** *d* ⊖ *b* = **0 using** *assms*(*1*) **by** *auto*
thus *x* = *y* **using** *hx hy haG hbG hcG hdG*
by (*metis inverse-closed invertible invertible-right-cancel invertible-right-inverse*)

qed
moreover have *himage* : (λ (*a*, *b*). *a* ⊕ *b*) ‘ (*stabilizer S1* × *stabilizer S2*) = *sumset* (*stabilizer S1*) (*stabilizer S2*)
proof
show (λ(*a*, *b*). *a* ⊕ *b*) ‘ (*stabilizer S1* × *stabilizer S2*) ⊆ *sumset* (*stabilizer S1*) (*stabilizer S2*)
using *stabilizer-subset-group* **by** *force*

next
show *sumset* (*stabilizer S1*) (*stabilizer S2*) ⊆ (λ(*a*, *b*). *a* ⊕ *b*) ‘ (*stabilizer S1* × *stabilizer S2*)
proof
fix *x* **assume** *x* ∈ *sumset* (*stabilizer S1*) (*stabilizer S2*)
then obtain *s1 s2* **where** *hs1*: *s1* ∈ *stabilizer S1* **and** *hs2*: *s2* ∈ *stabilizer S2* **and**
x = *s1* ⊕ *s2* **by** (*meson sumset.cases*)
thus *x* ∈ (λ(*a*, *b*). *a* ⊕ *b*) ‘ (*stabilizer S1* × *stabilizer S2*) **using** *hs1 hs2* **by** *auto*
qed
qed
ultimately show *?thesis* **using** *card-image card-cartesian-product* **by** *fastforce*
qed

lemma *stabilizer-sub-sumset-left*:
stabilizer A ⊆ *stabilizer* (*sumset A B*)
proof
fix *x* **assume** *hx*: *x* ∈ *stabilizer A*
then have *sumset* {*x*} (*sumset A B*) = *sumset A B* **using** *stabilizer-def sumset-assoc*
mem-Collect-eq **by** (*smt* (*verit*, *del-insts*) *sumset-Int-carrier-eq*(*1*) *sumset-commute*)
thus *x* ∈ *stabilizer* (*sumset A B*) **using** *hx stabilizer-def*
by (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq sumset-Int-carrier*)
qed

lemma *stabilizer-sub-sumset-right*:
stabilizer B ⊆ *stabilizer* (*sumset A B*)
using *stabilizer-sub-sumset-left sumset-commute* **by** *fastforce*

lemma *not-mem-stabilizer-obtain*:

assumes $A \neq \{\}$ **and** $x \notin$ *stabilizer* A **and** $x \in G$ **and** $A \subseteq G$ **and** *finite* A
obtains a **where** $a \in A$ **and** $x \oplus a \notin A$
proof−
have *sumset* $\{x\}$ $A \neq A$ **using** *assms stabilizer-def*
by (*metis* (*mono-tags*, *lifting*) *inf.orderE mem-Collect-eq*)
moreover have *card* (*sumset* $\{x\}$ A) = *card* A **using** *assms*
by (*metis card-sumset-singleton-eq inf.orderE sumset-commute*)
ultimately obtain y **where** $y \in$ *sumset* $\{x\}$ A **and** $y \notin A$ **using** *assms*
by (*meson card-subset-eq subsetI*)
then obtain a **where** $a \in A$ **and** $x \oplus a \notin A$ **using** *assms*
by (*metis singletonD sumset.cases*)
thus *?thesis* **using** *that* **by** *blast*
qed

lemma *sumset-eq-sub-stabilizer*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** *finite* B
shows *sumset* A $B = B \Longrightarrow A \subseteq$ *stabilizer* B
proof
fix x **assume** *hsum*: *sumset* A $B = B$ **and** *hx*: $x \in A$
have *sumset* $\{x\}$ $B = B$
proof−
have *sumset* $\{x\}$ $B \subseteq B$ **using** *hsum hx*
by (*metis empty-subsetI equalityE insert-subset sumset-mono*)
moreover have *card* (*sumset* $\{x\}$ B) = *card* B **using** *assms*
by (*metis IntD1 Int-absorb1 card-sumset-singleton-eq hx inf-commute sum-set-commute*)
ultimately show *?thesis* **using** *card-subset-eq assms*(*3*) **by** *auto*
qed
thus $x \in$ *stabilizer* B **using** *hx assms*(*1*) *stabilizer-def*
by (*metis* (*mono-tags*, *lifting*) *assms*(*2*) *inf.orderE mem-Collect-eq subsetD*)
qed

lemma *sumset-stabilizer-eq*:
shows *sumset* (*stabilizer* A) (*stabilizer* A) = *stabilizer* A
proof
show *sumset* (*stabilizer* A) (*stabilizer* A) $\subseteq$ *stabilizer* A
using *stabilizer-is-subgroup subgroup.axioms*(*1*) *subsetI*
by (*metis* (*mono-tags*, *lifting*) *additive-abelian-group.sumset.simps additive-abelian-group-axioms*
submonoid.sub-composition-closed)
next
show *stabilizer* $A \subseteq$ *sumset* (*stabilizer* A) (*stabilizer* A)
using *Left-Coset-eq-sumset stabilizer-is-nonempty*
stabilizer-subset-group sub-unit-closed additive-abelian-group-axioms right-unit
subset-iff sumsetI **by** (*smt* (*verit*, *best*))
qed

lemma *differenceset-stabilizer-eq*:
  shows *differenceset* (*stabilizer A*) (*stabilizer A*) = *stabilizer A*
proof
  show *differenceset* (*stabilizer A*) (*stabilizer A*) ⊆ *stabilizer A*
  proof
    fix *x* **assume** *x* ∈ *differenceset* (*stabilizer A*) (*stabilizer A*)
    then obtain *a b* **where** *a* ∈ *stabilizer A* **and** *b* ∈ *stabilizer A* **and** *x* = *a* ⊖ *b*
      by (*metis minusset.cases sumset.cases*)
    thus *x* ∈ *stabilizer A* **using** *stabilizer-is-subgroup subgroup.axioms*(*1*)
      by (*smt* (*verit*, *ccfv-threshold*) *in-mono invertible stabilizer-subset-group*
        subgroup-inverse-iff sub-composition-closed)
  qed
next
  show *stabilizer A* ⊆ *differenceset* (*stabilizer A*) (*stabilizer A*)
  proof
    fix *x* **assume** *hx*: *x* ∈ *stabilizer A*
    then have *x* ⊖ **0** ∈ *differenceset* (*stabilizer A*) (*stabilizer A*) **by** *blast*
    then show *x* ∈ *differenceset* (*stabilizer A*) (*stabilizer A*) **using** *hx* **by** *simp*
  qed
qed

lemma *stabilizer2-sub-stabilizer*:
  shows *stabilizer*(*stabilizer A*) ⊆ *stabilizer A*
proof(*cases A* ≠ {})
  case *True*
  then have *stabilizer*(*stabilizer A*) ⊆ *differenceset* (*stabilizer A*) (*stabilizer A*)
   by (*simp add*: *stabilizer-is-nonempty stabilizer-subset-group stabilizer-subset-nempty*)
  thus *?thesis* **using** *differenceset-stabilizer-eq* **by** *blast*
next
  case *False*
  then show *?thesis* **by** (*simp add*: *stabilizer-empty stabilizer-subset-group*)
qed

lemma *stabilizer-left-sumset-invariant*:
  assumes *a* ∈ *G* **and** *A* ⊆ *G*
  shows *stabilizer* (*sumset* {*a*} *A*) = *stabilizer A*

proof
  show *stabilizer* (*sumset* {*a*} *A*) ⊆ *stabilizer A*
  proof
    fix *x* **assume** *hx*: *x* ∈ *stabilizer* (*sumset* {*a*} *A*)
    then have *hxG*: *x* ∈ *G* **using** *stabilizer-def* **by** *blast*
    have *sumset* {*x*} (*sumset* {*a*} *A*) = *sumset* {*a*} *A* **using** *stabilizer-def hx*
      by (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq sumset-Int-carrier*)
    then have *sumset* {*x*} *A* = *A* **using** *assms*
      by (*metis* (*full-types*) *sumset-assoc sumset-commute sumset-subset-carrier*
        sumset-translate-eq-right)
    thus *x* ∈ *stabilizer A* **using** *hxG stabilizer-def*

by (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq sumset-Int-carrier*)
qed
next
show *stabilizer A* ⊆ *stabilizer* (*sumset* {*a*} *A*) **using** *stabilizer-def*
using *stabilizer-sub-sumset-right* **by** *meson*
qed

lemma *stabilizer-right-sumset-invariant*:
assumes *a* ∈ *G* **and** *A* ⊆ *G*
shows *stabilizer* (*sumset A* {*a*}) = *stabilizer A*
using *sumset-commute stabilizer-left-sumset-invariant assms* **by** *simp*

lemma *stabilizer-right-differenceset-invariant*:
assumes *b* ∈ *G* **and** *A* ⊆ *G*
shows *stabilizer* (*differenceset A* {*b*}) = *stabilizer A*
using *assms minusset-eq stabilizer-right-sumset-invariant* **by** *auto*

lemma *stabilizer-unchanged*:
assumes *a* ∈ *G* **and** *b* ∈ *G*
shows *stabilizer* (*sumset A B*) = *stabilizer* (*sumset A* (*sumset* (*differenceset B* {*b*}) {*a*}))

proof−
have *sumset A* (*sumset* (*differenceset B* {*b*}) {*a*}) = *sumset* (*differenceset* (*sumset A B*) {*b*}) {*a*}
by (*simp add*: *sumset-assoc*)
thus *?thesis* **using** *stabilizer-right-sumset-invariant*
stabilizer-right-differenceset-invariant assms sumset-subset-carrier **by** *simp*
qed

lemma *subset-stabilizer-of-subset-sumset*:
assumes *A* ⊆ *sumset* {*x*} (*stabilizer B*) **and** *x* ∈ *G* **and** *A* ≠ {} **and** *A* ⊆ *G*
shows *stabilizer A* ⊆ *stabilizer B*
proof−
obtain *a* **where** *ha*: *a* ∈ *A* **using** *assms* **by** *blast*
moreover then obtain *b* **where** *hb*: *b* ∈ *stabilizer B* **and** *haxb*: *a* = *x* ⊕ *b* **using** *sumset.cases assms* **by** *blast*
ultimately have *stabilizer A* ⊆ *differenceset A* {*a*} **using** *assms sumset-subset-carrier*
stabilizer-subset-difference-singleton **by** (*meson subset-trans*)
also have ... = *sumset* {*inverse a*} *A* **using** *sumset-commute ha assms*(*4*) *inverse-closed*
subsetD minusset-eq **by** *auto*
also have ... ⊆ *sumset* {*inverse x* ⊕ *inverse b*} (*sumset* {*x*} (*stabilizer B*))
using *assms sumset-mono haxb inverse-closed hb stabilizer-subset-group subsetD*
commutative inverse-composition-commute **by** (*metis invertible subset-singleton-iff*)
also have ... = *sumset* {*inverse b*} (*stabilizer B*) **using** *sumset-singletons-eq*

commutative
assms sumset-assoc hb stabilizer-subset-group inverse-closed invertible
by (*metis composition-closed invertible-right-inverse2 sub*)
also have ... = *stabilizer B* **using** *hb Left-Coset-eq-sumset sub-unit-closed sub subset-iff*
additive-abelian-group-axioms calculation disjoint-iff-not-equal factor-unit inverse-closed
sumset-subgroup-eq-iff **by** (*smt* (*verit*, *del-insts*))
finally show *?thesis* **.**
qed

lemma *sumset-stabilizer-eq-self*:
assumes $A \subseteq G$
shows *sumset* (*stabilizer A*) $A = A$
using *assms sumset-eq-Union-left*[*OF stabilizer-subset-group*]
Int-absorb2 stabilizer-coset-Un sumset-commute sumset-eq-Union-left **by** *presburger*

lemma *stabilizer-neq-subset-sumset*:
assumes $A \subseteq$ *sumset* $\{x\}$ (*stabilizer B*) **and** $x \in A$ **and** $\neg$ *sumset* $\{x\}$ (*stabilizer B*) $\subseteq C$ **and**
$A \subseteq C$ **and** $C \subseteq G$
shows *stabilizer A* $\neq$ *stabilizer B*
proof
assume *heq*: *stabilizer A* = *stabilizer B*
obtain *a* **where** $a \in$ *sumset* $\{x\}$ (*stabilizer B*) **and**
$a \notin C$ **using** *assms* **by** *blast*
moreover then obtain *b* **where** $b \in$ *stabilizer B* **and** $a = x \oplus b$ **using** *sumset.cases* **by** *blast*
ultimately have $b \oplus x \notin A$ **using** *commutative stabilizer-subset-group assms in-mono* **by** *metis*
thus *False* **using** *assms heq stabilizer-coset-subset subset-trans* **by** *metis*
qed

lemma *subset-stabilizer-Un*:
shows *stabilizer A* $\cap$ *stabilizer B* $\subseteq$ *stabilizer* $(A \cup B)$
proof
fix *x* **assume** *hx*: $x \in$ *stabilizer A* $\cap$ *stabilizer B*
then have *sumset* $\{x\}$ $(A \cap G) = A \cap G$ **using** *stabilizer-def* **by** *blast*
moreover have *sumset* $\{x\}$ $(B \cap G) = (B \cap G)$ **using** *stabilizer-def hx* **by** *blast*
ultimately have *sumset* $\{x\}$ $((A \cup B) \cap G) = (A \cup B) \cap G$ **using** *sumset-subset-Un2*
boolean-algebra.conj-disj-distrib2 **by** *auto*
then show $x \in$ *stabilizer* $(A \cup B)$ **using** *hx stabilizer-subset-group stabilizer-def* **by** *blast*
qed

lemma *mem-stabilizer-Un-and-left-imp-right*:
assumes *finite B* **and** $x \in$ *stabilizer* $(A \cup B)$ **and** $x \in$ *stabilizer A* **and** *disjnt A*

B
shows $x \in$ *stabilizer B*
proof−
have $(A \cap G) \cup$ *sumset* $\{x\}$ $(B \cap G) = (A \cap G) \cup (B \cap G)$
using *assms*(*2*) *sumset-subset-Un2*[*of* $\{x\}$ $A \cap G$ $B \cap G$] *stabilizer-def*[*of* $A \cup$ *B*]
Int-Un-distrib2[*of A B G*] *assms*(*3*) *stabilizer-def*
by (*metis* (*mono-tags*, *lifting*) *mem-Collect-eq*)
then have $B \cap G \subseteq$ *sumset* $\{x\}$ $(B \cap G)$ **using** *assms*(*4*) *disjnt-def Int-Un-distrib2 Int-commute*
sumset-subset-Un1 **by** (*smt* (*verit*, *del-insts*) *Int-assoc Un-Int-eq*(*2*) *inf.orderI insert-is-Un*
sumset-empty(*2*))
then show $x \in$ *stabilizer B* **using** *stabilizer-def*[*of B*] *assms*(*1*) *assms*(*3*) *card-subset-eq*

card-sumset-singleton-subset-eq finite.emptyI finite.insertI finite-Int finite-sumset

inf.cobounded2 stabilizer-subset-group subsetD **by** (*smt* (*verit*) *mem-Collect-eq*)
qed

lemma *mem-stabilizer-Un-and-right-imp-left*:
assumes *finite A* **and** $x \in$ *stabilizer* $(A \cup B)$ **and** $x \in$ *stabilizer B* **and** *disjnt A B*
shows $x \in$ *stabilizer A*
using *mem-stabilizer-Un-and-left-imp-right Un-commute assms disjnt-sym* **by** *metis*

lemma *Union-stabilizer-Class-eq*:
assumes $A \subseteq G$
shows $A = (\bigcup$ (*Class A* ‘ *A*)) **using** *assms sumset-commute sumset-subgroup-eq-Class-Union*

sumset-stabilizer-eq-self **by** *presburger*

lemma *card-stabilizer-sumset-divide-sumset*:
card (*stabilizer* (*sumset A B*)) *dvd card* (*sumset A B*) **using** *card-divide-sumset*
sumset-commute sumset-stabilizer-eq-self sumset-subset-carrier **by** *metis*

lemma *card-stabilizer-le*:
assumes $A \subseteq G$ **and** *finite A* **and** $A \neq \{\}$
shows *card* (*stabilizer A*) $\leq$ *card A* **using** *assms*
by (*metis card-le-sumset finite.cases insertCI insert-subset stabilizer-finite*
stabilizer-subset-group sumset-commute sumset-stabilizer-eq-self)

lemma *sumset-Inter-subset-sumset*:
assumes $a \in G$ **and** $b \in G$
shows *sumset* ($A \cap$ *sumset* $\{a\}$ (*stabilizer C*)) ($B \cap$ *sumset* $\{b\}$ (*stabilizer C*)) $\subseteq$
sumset $\{a \oplus b\}$ (*stabilizer C*) (**is** *sumset ?A ?B* $\subseteq$ -)
proof

fix x **assume** $x \in$ *sumset ?A ?B*
then obtain *d1 d2* **where** $d1 \in$ *sumset* $\{a\}$ (*stabilizer C*) **and**
$d2 \in$ *sumset* $\{b\}$ (*stabilizer C*) **and** $x = d1 \oplus d2$ **by** (*meson IntD2 sumset.cases*)
then obtain *c1 c2* **where** *hc1*: $c1 \in$ *stabilizer C* **and** *hc2*: $c2 \in$ *stabilizer C* **and**
$x = (a \oplus c1) \oplus (b \oplus c2)$ **using** *sumset.simps* **by** *auto*
then have $x = (a \oplus b) \oplus (c1 \oplus c2)$ **using** *hc1 hc2 assms associative commutative*

stabilizer-subset-group **by** *simp*
thus $x \in$ *sumset* $\{a \oplus b\}$ (*stabilizer C*) **using** *stabilizer-is-subgroup hc1 hc2*
stabilizer-subset-group sumset.simps sumset-stabilizer-eq assms **by** *blast*
qed

1.3 Convergent

definition *convergent* :: $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool* **where**
convergent C A B $\equiv$ $C \subseteq$ *sumset A B* $\wedge$ $C \neq \{\}$ $\wedge$
card C + *card* (*stabilizer C*) $\geq$ *card* $(A \cap B)$ + *card* (*sumset* $(A \cup B)$ (*stabilizer C*))

definition *convergent-set* :: $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set set* **where**
convergent-set A B = *Collect* (λ *C*. *convergent C A B*)

lemma *convergent-set-sub-powerset*:
convergent-set A B $\subseteq$ *Pow* (*sumset A B*) **using** *convergent-set-def convergent-def* **by** *blast*

lemma *finite-convergent-set*:
assumes *finite A* **and** *finite B*
shows *finite* (*convergent-set A B*)
using *convergent-set-sub-powerset finite-Pow-iff finite-sumset assms finite-subset* **by** *metis*

1.4 Technical lemmas from DeVos's proof of Kneser's Theorem

The following lemmas correspond to intermediate arguments in the proof of Kneser's Theorem by DeVos that we will be following [1].

lemma *stabilizer-sumset-psubset-stabilizer*:
assumes $a \in G$ **and** $b \in G$ **and** $A \cap$ *sumset* $\{a\}$ (*stabilizer C*) $\neq \{\}$ **and**
$B \cap$ *sumset* $\{b\}$ (*stabilizer C*) $\neq \{\}$ **and** *hnotsub*: $\neg$ *sumset* $\{a \oplus b\}$ (*stabilizer C*) $\subseteq$ *sumset A B*
shows *stabilizer* (*sumset* ($A \cap$ *sumset* $\{a\}$ (*stabilizer C*)) ($B \cap$ *sumset* $\{b\}$ (*stabilizer C*))) $\subset$
stabilizer C (**is** *?H* $\subset$ -)
proof
have *sumset* ($A \cap$ *sumset* $\{a\}$ (*stabilizer C*)) ($B \cap$ *sumset* $\{b\}$ (*stabilizer C*)) $\neq \{\}$
using *assms* **by** (*simp add*: *inf-assoc*)

then show *?H* ⊆ *stabilizer C*
by (*meson assms*(*1*) *assms*(*2*) *composition-closed subset-stabilizer-of-subset-sumset sumset-Inter-subset-sumset sumset-subset-carrier*)
next
obtain *c1 c2* **where** *a* ⊕ *c1* ∈ *A* **and** *b* ⊕ *c2* ∈ *B* **and** *hc1*: *c1* ∈ *stabilizer C* **and** *hc2*: *c2* ∈ *stabilizer C*
using *assms*(*1*, *2*, *3*, *4*) *Left-Coset-eq-sumset-stabilizer* **by** *fastforce*
then have *hac1mem*: (*a* ⊕ *c1*) ∈ *A* ∩ *sumset* {*a*} (*stabilizer C*) **and** *hac1G*: *a* ⊕ *c1* ∈ *G* **and**
hbc2mem: (*b* ⊕ *c2*) ∈ *B* ∩ *sumset* {*b*} (*stabilizer C*) **and** *hbc2G*: *b* ⊕ *c2* ∈ *G*
by (*auto simp add*: *assms*(*1*, *2*) *sumset.sumsetI*)
have (*a* ⊕ *c1*) ⊕ (*b* ⊕ *c2*) ∈ *sumset* {*a* ⊕ *b*} (*stabilizer C*) **using** *assms hc1 hc2*
by (*smt* (*verit*) *associative commutative composition-closed insertI1 sub sub-composition-closed sumset.sumsetI*)
then have *sumset* {*a* ⊕ *b*} (*stabilizer C*) ∩ *sumset* {(*a* ⊕ *c1*) ⊕ (*b* ⊕ *c2*)} (*stabilizer C*) ≠ {}
using *zero-mem-stabilizer* **by** (*smt* (*verit*, *ccfv-threshold*) *composition-closed disjoint-iff-not-equal hac1G hbc2G insertCI right-unit sumset.sumsetI unit-closed*)
then have *hsumeq*: *sumset* {*a* ⊕ *b*} (*stabilizer C*) = *sumset* {(*a* ⊕ *c1*) ⊕ (*b* ⊕ *c2*)} (*stabilizer C*)
using *sumset-stabilizer-eq-iff assms hac1G hbc2G composition-closed* **by** *presburger*
have (*sumset* (*A* ∩ *sumset* {*a*} (*stabilizer C*)) (*B* ∩ *sumset* {*b*} (*stabilizer C*))) ⊆ *sumset* {*a* ⊕ *b*} (*stabilizer C*)
by (*simp add*: *assms*(*1*, *2*) *sumset-Inter-subset-sumset*)
have *hsummem*: (*a* ⊕ *c1*) ⊕ (*b* ⊕ *c2*) ∈ *sumset* (*A* ∩ *sumset* {*a*} (*stabilizer C*)) (*B* ∩ *sumset* {*b*} (*stabilizer C*))
using *hac1mem hbc2mem hac1G hbc2G sumset.sumsetI* **by** *blast*
show *?H* ≠ *stabilizer C*
using *stabilizer-neq-subset-sumset*[*OF - hsummem*] *hnotsub hsumeq sumset-Inter-subset-sumset assms*
sumset-subset-carrier composition-closed sumset-mono sumset.sumsetI zero-mem-stabilizer

inf.cobounded1 right-unit unit-closed **by** *metis*
qed

lemma *stabilizer-eq-stabilizer-union*:
assumes *a* ∈ *G* **and** *b* ∈ *G* **and***A* ∩ *sumset* {*a*} (*stabilizer C*) ≠ {} **and**
B ∩ *sumset* {*b*} (*stabilizer C*) ≠ {} **and** *hnotsub*: ¬ *sumset* {*a* ⊕ *b*} (*stabilizer C*) ⊆ *sumset A B* **and**
C ⊆ *sumset A B* **and** *finite C* **and**
C ∩ *sumset* (*A* ∩ *sumset* {*a*} (*stabilizer C*)) (*B* ∩ *sumset* {*b*} (*stabilizer C*)) = {} **and** *C* ≠ {} **and**
finite A **and** *finite B*
shows *stabilizer* (*sumset* (*A* ∩ *sumset* {*a*} (*stabilizer C*)) (*B* ∩ *sumset* {*b*} (*stabilizer C*))) =
stabilizer (*C* ∪ *sumset* (*A* ∩ *sumset* {*a*} (*stabilizer C*)) (*B* ∩ *sumset* {*b*} (*stabilizer C*))) (**is** *stabilizer ?H* = *stabilizer ?K*)
proof

show *stabilizer ?H* $\subseteq$ *stabilizer ?K* **using** *subset-stabilizer-Un Int-absorb1 stabilizer-sumset-psubset-stabilizer psubset-imp-subset assms* **by** *metis*
next
have *hCG* : $C \subseteq G$ **using** *assms(6) sumset-subset-carrier* **by** *force*
show *stabilizer ?K* $\subseteq$ *stabilizer ?H*
proof
fix x **assume** *hxC1*: $x \in$ *stabilizer ?K*
moreover have $x \in$ *stabilizer C*
proof−
have *hC-Un*: $C = (\bigcup$ *(Class C ' C))* **using** *Union-stabilizer-Class-eq hCG* **by** *simp*
have *hCsumx*: *sumset* $\{x\}$ $C = (\bigcup$ $y \in$ *Class C ' C. sumset* $\{x\}$ $y)$
proof
show *sumset* $\{x\}$ $C \subseteq \bigcup$ *(sumset* $\{x\}$ *' Class C ' C)*
proof
fix y **assume** *hy*: $y \in$ *sumset* $\{x\}$ C
then obtain c **where** *hc*: $c \in C$ **and** *hyxc*: $y = x \oplus c$ **using** *sumset.cases* **by** *blast*
then obtain K **where** *hK*: $K \in$ *Class C ' C* **and** $c \in K$ **using** *hC-Un* **by** *blast*
then have $y \in$ *sumset* $\{x\}$ K **using** *hyxc hc* **by** *(metis sumset.cases sumset.sumsetI hCG hy singletonD subset-iff)*
then show $y \in \bigcup$ *(sumset* $\{x\}$ *' Class C ' C)* **using** *hK* **by** *auto*
qed
next
show $\bigcup$ *(sumset* $\{x\}$ *' Class C ' C)* $\subseteq$ *sumset* $\{x\}$ C
proof(*intro Union-least*)
fix X **assume** $X \in$ *sumset* $\{x\}$ *' Class C ' C*
then obtain K **where** $K \in$ *Class C ' C* **and** $X =$ *sumset* $\{x\}$ K **by** *blast*
then show $X \subseteq$ *sumset* $\{x\}$ C **using** *sumset-mono[of* $\{x\}$ $\{x\}$ *K C] hC-Un subset-refl* **by** *blast*
qed
qed
have $x \notin$ *stabilizer C* $\Longrightarrow$ *False*
proof−
assume *hxC*: $x \notin$ *stabilizer C*
then have *hxG*: $x \in G$ **using** *hxC1 stabilizer-subset-group* **by** *blast*
then have *hxCne*: *sumset* $\{x\}$ $C \neq C$ **using** *stabilizer-def[of C] hCG Int-absorb2 hxC* **by** *(metis (mono-tags, lifting) mem-Collect-eq)*
moreover have *hxsplit*: *sumset* $\{x\}$ $C \cup$ *sumset* $\{x\}$ *?H* $= C \cup$ *?H* **using** *hxC1 stabilizer-def[of ?K] sumset-subset-carrier assms(6) sumset-subset-Un2* **by** *force*
have *sumset* $\{x\}$ $C \cap$ *?H* $\neq \{\}$
proof
assume *sumset* $\{x\}$ $C \cap$ *?H* $= \{\}$
then have *sumset* $\{x\}$ $C \subset C$ **using** *hxsplit hxCne* **by** *blast*
thus *False* **using** *hCG assms(6) assms(7) hxC1 stabilizer-subset-group psubset-card-mono*

by (*metis card-sumset-singleton-eq sumset-Int-carrier sumset-commute sumset-stabilizer-eq-self hxG less-irrefl-nat*)
qed
then obtain *c* **where** *hc*: $c \in C$ **and**
hxcne: *sumset* $\{x\}$ (*Class C c*) $\cap$ *?H* $\neq$ $\{\}$ **using** *hCsumx* **by** *blast*
then have *hxc*: *sumset* $\{x\}$ (*Class C c*) = *Class C* ($x \oplus c$)
using *hxG assms*(*6*) *Left-Coset-Class-unit Left-Coset-eq-sumset-stabilizer sumset-assoc*
sumset-singletons-eq composition-closed sumset.cases sumset-stabilizer-eq-self hCG **by** (*smt* (*verit*))
have *hClassCempty*: *Class C* ($x \oplus c$) $\cap$ *C* = $\{\}$
proof−
have ¬ *Class C* ($x \oplus c$) $\subseteq$ *C* **using** *hxc hxcne assms*(*8*) **by** *blast*
then show *?thesis* **using** *Class-cover-imp-subset-or-disj*[*OF hC-Un - hCG*]
by (*meson composition-closed hCG hc hxG subsetD*)
qed
have *Class C* ($x \oplus c$) $\subseteq$ *sumset* $\{x\}$ *C* **using** *hCsumx hc hxc* **by** *blast*
then have *Class C* ($x \oplus c$) $\subseteq$ *?H* **using** *hClassCempty hxsplit* **by** *auto*
moreover have *card* (*Class C* ($x \oplus c$)) = *card* (*stabilizer C*) **using** *hxG hc hCG*
composition-closed Right-Coset-Class-unit Right-Coset-cardinality sumset-Int-carrier
Class-cover-imp-subset-or-disj assms **by** *auto*
ultimately have *card* (*stabilizer C*) $\leq$ *card ?H* **using** *card-mono finite-sumset assms*(*10*, *11*) *finite-Int* **by** *metis*
moreover have *card ?H* < *card* (*sumset* $\{a \oplus b\}$ (*stabilizer C*))
proof (*intro psubset-card-mono psubsetI sumset-Inter-subset-sumset assms*(*1*) *assms*(*2*))
show *finite* (*sumset* $\{a \oplus b\}$ (*stabilizer C*))
using *stabilizer-finite assms finite-sumset* **by** (*simp add*: *hCG*)
next
show *?H* $\neq$ *sumset* $\{a \oplus b\}$ (*stabilizer C*)
using *hnotsub sumset-mono* **by** (*metis Int-lower1*)
qed
ultimately show *False*
using *assms*(*1*, *2*) *stabilizer-subset-group* **by** (*simp add*: *card-sumset-singleton-subset-eq*)

qed
then show *?thesis* **by** *auto*
qed
moreover have *finite ?H* **using** *finite-sumset assms*(*10*, *11*) *finite-Int* **by** *simp*
ultimately show $x \in$ *stabilizer ?H* **using** *mem-stabilizer-Un-and-right-imp-left*[*of ?H x C*]
disjnt-def assms Un-commute **by** (*metis disjoint-iff-not-equal*)
qed
qed

lemma *sumset-inter-ineq*:
assumes *B* $\cap$ *sumset* $\{a\}$ (*stabilizer C*) = $\{\}$ **and** *stabilizer* (*sumset* (*A* $\cap$ *sumset*

```
{a} (stabilizer C)) (B ∩ sumset {b} (stabilizer C))) ⊂ stabilizer C and
  a ∈ A and a ∈ G and finite A and finite B and A ≠ {} and B ≠ {} and finite
(stabilizer C)
   shows int (card (sumset (A ∪ B) (stabilizer C))) − card (sumset (A ∪ B)
(stabilizer (sumset (A ∩ sumset {a} (stabilizer C)) (B ∩ sumset {b} (stabilizer
C))))) ≥
      int (card (stabilizer C)) − card (sumset (A ∩ sumset {a} (stabilizer C))
(stabilizer (sumset (A ∩ sumset {a} (stabilizer C)) (B ∩ sumset {b} (stabilizer
C)))))
  (is int (card (sumset (A ∪ B) (stabilizer C))) − card (sumset (A ∪ B) ?H1) ≥
                int (card (stabilizer C)) − card (sumset ?A1 ?H1))
proof−
  have hfinsumH1:finite (sumset (A ∪ B) ?H1)
    using finite-sumset assms by (meson finite-Un psubsetE rev-finite-subset)
  have hsubsumH1: sumset (A ∪ B) ?H1 ⊆ sumset (A ∪ B) (stabilizer C)
    using sumset.cases assms by (meson psubsetE subset-refl sumset-mono)
   have hsumH1card-le: card (sumset (A ∪ B) ?H1) ≤ card (sumset (A ∪ B)
(stabilizer C))
    using card-mono finite-sumset stabilizer-finite assms
    by (metis equalityE finite-UnI psubset-imp-subset sumset-mono)
  have hsub: sumset ?A1 ?H1 ⊆ sumset {a} (stabilizer C)
  proof
    fix x assume x ∈ sumset ?A1 ?H1
    then obtain h1 f where h1 ∈ ?A1 and hf: f ∈ ?H1 and
      hx: x = h1 ⊕ f by (meson sumset.cases)
    then obtain c where hc: c ∈ stabilizer C and hac: h1 = a ⊕ c
      by (metis Int-iff empty-iff insert-iff sumset.cases)
    then have hcf: c ⊕ f ∈ stabilizer C using hf assms(2) stabilizer-is-subgroup
      subgroup-def monoid-axioms Group-Theory.group.axioms(1)
      Group-Theory.monoid-def subset-iff psubset-imp-subset
      by (smt (verit) stabilizer-subset-group sumset.sumsetI sumset-stabilizer-eq)
    have hcG: c ∈ G using hc stabilizer-subset-group by auto
    have hfG: f ∈ G using hf stabilizer-subset-group by auto
    show x ∈ sumset {a} (stabilizer C) using hx hac assms stabilizer-subset-group
hcf
      using Left-Coset-eq-sumset-stabilizer Left-Coset-memI associative hcG hfG
by presburger
  qed
  moreover have finite (sumset ?A1 ?H1) using finite-sumset assms stabilizer-finite
finite-subset
    by (metis finite.simps hsub)
  ultimately have card (sumset {a} (stabilizer C)) − card (sumset ?A1 ?H1) =
    card (sumset {a} (stabilizer C) − sumset ?A1 ?H1)
    using card-Diff-subset by metis
  moreover have card (sumset ?A1 ?H1) ≤ card (sumset {a} (stabilizer C))
    using card-mono hsub finite-sumset assms by (metis finite.simps)
   ultimately have int (card (sumset {a} (stabilizer C))) − card (sumset ?A1
?H1) =
    card (sumset {a} (stabilizer C) − sumset ?A1 ?H1) by linarith
```

also have ... ≤ *card* ((*sumset* (*A* ∪ *B*) (*stabilizer C*)) − (*sumset* (*A* ∪ *B*) *?H1*))
proof−
have *sumset* {*a*} (*stabilizer C*) − *sumset ?A1 ?H1* ⊆ *sumset* (*A* ∪ *B*) (*stabilizer C*) − *sumset* (*A* ∪ *B*) *?H1*
proof
fix *x* **assume** *hx*: *x* ∈ *sumset* {*a*} (*stabilizer C*) − *sumset ?A1 ?H1*
then obtain *c* **where** *hxac*: *x* = *a* ⊕ *c* **and** *hc*: *c* ∈ *stabilizer C* **and** *hcG*: *c* ∈ *G*
using *sumset.cases* **by** *blast*
then have *x* ∈ *sumset* (*A* ∪ *B*) (*stabilizer C*) **using** *assms sumset.cases* **by** *blast*
moreover have *x* ∉ *sumset* (*A* ∪ *B*) *?H1*
proof
assume *x* ∈ *sumset* (*A* ∪ *B*) *?H1*
then obtain *y h1* **where** *hy*: *y* ∈ *A* ∪ *B* **and** *hyG*: *y* ∈ *G* **and** *hh1G*: *h1* ∈ *G*
and *hh1*: *h1* ∈ *?H1* **and** *hxy*: *x* = *y* ⊕ *h1* **by** (*meson sumset.cases*)
then have *y* = *a* ⊕ (*c* ⊕ *inverse h1*) **using** *hxac hxy assms associative commutative composition-closed*
inverse-closed invertible invertible-left-inverse2 **by** (*metis hcG*)
moreover have *h1* ∈ *stabilizer C* **using** *hh1 assms* **by** *auto*
moreover hence *c* ⊕ *inverse h1* ∈ *stabilizer C* **using** *hc stabilizer-is-subgroup subgroup-def*
group-axioms invertible subgroup.subgroup-inverse-iff submonoid.sub-composition-closed hh1G **by** *metis*
ultimately have *y* ∈ *sumset* {*a*} (*stabilizer C*)
using *assms hcG hh1G* **by** *blast*
moreover hence *y* ∈ *A* **using** *assms*(*1*) *hy* **by** *auto*
ultimately have *x* ∈ *sumset ?A1 ?H1* **using** *hxy hh1*
by (*simp add*: *hyG hh1G sumset.sumsetI*)
thus *False* **using** *hx* **by** *auto*
qed
ultimately show *x* ∈ *sumset* (*A* ∪ *B*) (*stabilizer C*) − *sumset* (*A* ∪ *B*) *?H1*
by *simp*
qed
thus *?thesis* **using** *card-mono finite-Diff finite-sumset assms*
by (*metis finite-UnI nat-int-comparison*(*3*))
qed
also have ... = *int* (*card* (*sumset* (*A* ∪ *B*) (*stabilizer C*))) −
card (*sumset* (*A* ∪ *B*) *?H1*)
using *card-Diff-subset*[*OF hfinsumH1 hsubsumH1*] *hsumH1card-le* **by** *linarith*
finally show *int* (*card* (*sumset* (*A* ∪ *B*) (*stabilizer C*))) − *card* (*sumset* (*A* ∪ *B*) *?H1*) ≥
int (*card* (*stabilizer C*)) − *card* (*sumset ?A1 ?H1*)
using *assms* **by** (*metis card-sumset-singleton-subset-eq stabilizer-subset-group*)
qed

lemma *exists-convergent-min-stabilizer*:
assumes *hind*: ∀ *m*<*n*. ∀ *C D*. *C* ⊆ *G* ⟶ *D* ⊆ *G* ⟶ *finite C* ⟶ *finite D* ⟶

$C \neq \{\} \longrightarrow$
$D \neq \{\} \longrightarrow$ *card* (*sumset C D*) + *card C* = *m* $\longrightarrow$
card (*sumset C* (*stabilizer* (*sumset C D*))) + *card* (*sumset D* (*stabilizer* (*sumset C D*))) −
card ((*stabilizer* (*sumset C D*)))
$\leq$ *card* (*sumset C D*) **and** *hAG*: $A \subseteq G$ **and** *hBG*: $B \subseteq G$ **and** *hA*: *finite A* **and**
hB: *finite B* **and** *hAne*: $A \neq \{\}$ **and** $A \cap B \neq \{\}$ **and**
hcardsum: *card* (*sumset A B*) + *card A* = *n* **and** *hintercardA*: *card* ($A \cap B$) < *card A*
obtains *X* **where** *convergent X A B* **and** $\bigwedge$ *Y*. *Y* $\in$ *convergent-set A B* $\Longrightarrow$
card (*stabilizer Y*) $\geq$ *card* (*stabilizer X*)
proof−
let *?C0* = *sumset* ($A \cap B$) ($A \cup B$)
have *hC0ne*: *?C0* $\neq \{\}$ **using** *assms* **by** *fast*
moreover have *finite ?C0* **using** *sumset-inter-union-subset finite-sumset assms* **by** *auto*
ultimately have *finite* (*stabilizer ?C0*) **using** *stabilizer-finite*
using *sumset-subset-carrier* **by** *presburger*
then have *hcard-sumset-le*: *card* ($A \cap B$) $\leq$ *card* (*sumset* ($A \cap B$) (*stabilizer ?C0*))
using *card-le-sumset sumset-commute sub-unit-closed assms*
by (*metis Int-Un-eq*(*3*) *Un-subset-iff finite-Int unit-closed*)
have *card ?C0* $\leq$ *card* (*sumset A B*)
using *card-mono sumset-inter-union-subset finite-sumset assms*
by (*simp add*: *card-mono finite-sumset hA hB sumset-inter-union-subset*)
then have *card ?C0* + *card* ($A \cap B$) < *card* (*sumset A B*) + *card A*
using *hintercardA* **by** *auto*
then obtain *m* **where** *m* < *n* **and** *card ?C0* + *card* ($A \cap B$) = *m* **using** *hcardsum* **by** *auto*
then have *card* (*sumset* ($A \cap B$) (*stabilizer ?C0*)) +
card (*sumset* ($A \cup B$) (*stabilizer ?C0*)) − *card* (*stabilizer ?C0*) $\leq$ *card ?C0*
using *assms finite-Un finite-Int*
by (*metis Int-Un-eq*(*4*) *Un-empty Un-subset-iff*)
then have *card ?C0* + *card* (*stabilizer ?C0*) $\geq$
card ($A \cap B$) + *card* (*sumset* ($A \cup B$) (*stabilizer ?C0*)) **using** *hcard-sumset-le*
by *auto*
then have *?C0* $\in$ *convergent-set A B* **using** *convergent-set-def convergent-def*
sumset-inter-union-subset hC0ne **by** *auto*
then have *hconvergent-ne*: *convergent-set A B* $\neq \{\}$ **by** *auto*
define *KS* **where** *KS* $\equiv$ (λ *X*. *card* (*stabilizer X*)) ‘ *convergent-set A B*
define *K* **where** *K* $\equiv$ *Min KS*
define *C* **where** *C* $\equiv$ @*C*. *C* $\in$ *convergent-set A B* $\wedge$ *K* = *card* (*stabilizer C*)
obtain *KS*: *finite KS KS* $\neq \{\}$
using *hconvergent-ne finite-convergent-set assms KS-def* **by** *auto*
then have *K* $\in$ *KS* **using** *K-def Min-in* **by** *blast*
then have $\exists$ *X*. *X* $\in$ *convergent-set A B* $\wedge$ *K* = *card* (*stabilizer X*)
using *KS-def* **by** *auto*
then obtain *C* $\in$ *convergent-set A B* **and** *Keq*: *K* = *card* (*stabilizer C*)

by (*metis* (*mono-tags*, *lifting*) *C-def someI-ex*)
then have *hC*: $C \subseteq$ *sumset A B* **and** *hCne*: $C \neq \{\}$ **and**
hCcard: *card C* + *card* (*stabilizer C*) $\geq$
card ($A \cap B$) + *card* (*sumset* ($A \cup B$) (*stabilizer C*))
using *convergent-set-def convergent-def* **by** *auto*
have *hCmin*: $\bigwedge$ *Y*. *Y* $\in$ *convergent-set A B* $\Longrightarrow$
card (*stabilizer Y*) $\geq$ *card* (*stabilizer C*)
using *K-def KS-def Keq Min-le KS*(*1*) **by** *auto*
show *?thesis* **using** *hCmin hC hCcard hCne local.convergent-def that* **by** *presburger*
qed

end

context *normal-subgroup*
begin

1.5 A function that picks coset representatives randomly

definition φ :: *'a set* $\Rightarrow$ *'a* **where**
φ = (λ *x*. *if x* $\in$ *G // K then* (*SOME a*. *a* $\in$ *G* $\wedge$ *x* = *a* ·| *K*) *else undefined*)

definition *quot-comp-alt* :: *'a* $\Rightarrow$ *'a* $\Rightarrow$ *'a* **where** *quot-comp-alt a b* = φ ((*a* · *b*) ·| *K*)

lemma *phi-eq-coset*:
assumes φ *x* = *a* **and** *a* $\in$ *G* **and** *x* $\in$ *G // K*
shows *x* = *a* ·| *K*
proof−
have (*SOME a*. *a* $\in$ *G* $\wedge$ *x* = *a* ·| *K*) = *a* **using** φ*-def assms* **by** *simp*
then show *?thesis* **using** *some-eq-ex representant-exists Left-Coset-Class-unit assms*
by (*metis* (*mono-tags*, *lifting*))
qed

lemma *phi-coset-mem*:
assumes *a* $\in$ *G*
shows φ (*a* ·| *K*) $\in$ *a* ·| *K*
proof−
obtain *x* **where** *hx*: *x* = φ (*a* ·| *K*) **by** *auto*
then have *x* = (*SOME x*. *x* $\in$ *G* $\wedge$ *a* ·| *K* = *x* ·| *K*) **using** φ*-def assms*
Class-in-Partition Left-Coset-Class-unit **by** *presburger*
then show *?thesis* **using** φ*-def Class-self Left-Coset-Class-unit hx assms*
by (*smt* (*verit*, *ccfv-SIG*) *tfl-some*)
qed

lemma *phi-coset-eq*:
assumes *a* $\in$ *G* **and** φ *x* = *a* **and** *x* $\in$ *G // K*
shows φ (*a* ·| *K*) = *a* **using** *phi-eq-coset assms* **by** *metis*

lemma *phi-inverse-right*:
assumes $g \in G$
shows *quot-comp-alt* g (φ (*inverse* $g \cdot| K$)) $= \varphi\ K$
proof –
have $g \cdot (\varphi$ (*inverse* $g \cdot| K$)) $\in$ ($g \cdot$ (*inverse* g) $\cdot| K$)
using *phi-coset-mem assms* **by** (*smt* (*z3*) *Left-Coset-memE factor-unit invertible invertible-right-inverse invertible-inverse-closed invertible-inverse-inverse sub invertible-left-inverse2*)
then have $g \cdot (\varphi$ (*inverse* $g \cdot| K$)) $\cdot| K = K$
using *Block-self Left-Coset-Class-unit Normal-def quotient.unit-closed sub*
by (*metis assms composition-closed invertible invertible-inverse-closed invertible-right-inverse*)
then show *?thesis* **using** *quot-comp-alt-def* **by** *auto*
qed

lemma *phi-inverse-left*:
assumes $g \in G$
shows *quot-comp-alt* (φ (*inverse* $g \cdot| K$)) $g = \varphi\ K$
proof –
have (φ (*inverse* $g \cdot| K$)) $\cdot\ g \in$ ((*inverse* g) $\cdot\ g$) $\cdot| K$ **using** *phi-coset-mem assms*
by (*metis Left-Coset-memE factor-unit invertible invertible-inverse-closed invertible-left-inverse normal*)
then have (φ (*inverse* $g \cdot| K$)) $\cdot\ g \cdot| K = K$ **using** *Block-self Left-Coset-Class-unit Normal-def*
quotient.unit-closed sub **by** (*smt* (*verit, best*) *assms composition-closed invertible invertible-inverse-closed*
invertible-left-inverse)
then show *?thesis* **using** *quot-comp-alt-def* **by** *auto*
qed

lemma *phi-mem-coset-eq*:
assumes $a \in G\ //\ K$ **and** $b \in G$
shows $\varphi\ a \in b \cdot| K \Longrightarrow a = (b \cdot| K)$
proof –
assume $\varphi\ a \in b \cdot| K$
then have $a \cap (b \cdot| K) \neq \{\}$
by (*metis Class-closed Class-is-Left-Coset Int-iff assms empty-iff phi-coset-mem phi-eq-coset*)
then show $a = b \cdot| K$ **by** (*metis Class-in-Partition Class-is-Left-Coset assms disjoint*)
qed

lemma *forall-unique-repr*:
$\forall\ x \in G\ //\ K.\ \exists!\ k \in \varphi$ ‘ ($G\ //\ K$). $x = k \cdot| K$
proof
fix x **assume** *hx*: $x \in G\ //\ K$
then have $\varphi\ x \cdot| K = x$

by (*metis Class-is-Left-Coset block-closed phi-coset-mem phi-eq-coset representant-exists*)
then have *hex*: $\exists\ k \in \varphi$ ' $(G\ //\ K).\ x = k \cdot|\ K$ **using** *hx* **by** *blast*
moreover have $\bigwedge a\ b.\ a \in \varphi$ ' $(G\ //\ K) \Longrightarrow x = a \cdot|\ K \Longrightarrow b \in \varphi$ ' $(G\ //\ K) \Longrightarrow x = b \cdot|\ K \Longrightarrow$
$a = b$
proof−
fix *a b* **assume** $a \in \varphi$ ' $(G\ //\ K)$ **and** *hxa*: $x = a \cdot|\ K$ **and** $b \in \varphi$ ' $(G\ //\ K)$ **and**
hxb: $x = b \cdot|\ K$
then obtain *z w* **where** $a = \varphi\ (z \cdot|\ K)$ **and** $b = \varphi\ (w \cdot|\ K)$ **and** $z \in G$ **and** $w \in G$
using *representant-exists Left-Coset-Class-unit* **by** *force*
then show $a = b$ **using** *hxa hxb*
by (*metis Class-in-Partition Class-is-Left-Coset block-closed phi-coset-mem phi-eq-coset*)
qed
ultimately show $\exists!\ k \in \varphi$ ' $(G\ //\ K).\ x = k \cdot|\ K$ **by** *blast*
qed

lemma *phi-inj-on*:
shows *inj-on* $\varphi\ (G\ //\ K)$
proof(*intro inj-onI*)
fix *x y* **assume** $x \in G\ //\ K$ **and** *hy*: $y \in G\ //\ K$ **and** *hxy*: $\varphi\ x = \varphi\ y$
then obtain *a b* **where** $x = a \cdot|\ K$ **and** $y = b \cdot|\ K$ **and** $a \in G$ **and** $b \in G$
using *representant-exists Left-Coset-Class-unit* **by** *metis*
then show $x = y$ **using** *hxy hy* **by** (*metis phi-coset-mem phi-mem-coset-eq*)
qed

lemma *phi-coset-eq-self*:
assumes $a \in G\ //\ K$
shows $\varphi\ a \cdot|\ K = a$
by (*metis Class-closed Class-is-Left-Coset assms phi-coset-mem phi-eq-coset representant-exists*)

lemma *phi-coset-comp-eq*:
assumes $a \in G\ //\ K$ **and** $b \in G\ //\ K$
shows $\varphi\ a \cdot \varphi\ b \cdot|\ K = a\ [\cdot]\ b$ **using** *assms phi-coset-eq-self*
by (*metis Class-is-Left-Coset block-closed factor-composition phi-coset-mem representant-exists*)

lemma *phi-comp-eq*:
assumes $a \in G\ //\ K$ **and** $b \in G\ //\ K$
shows $\varphi\ (a\ [\cdot]\ b) =$ *quot-comp-alt* $(\varphi\ a)\ (\varphi\ b)$
using *phi-coset-comp-eq quot-comp-alt-def assms* **by** *auto*

lemma *phi-image-subset*:
φ ' $(G\ //\ K) \subseteq G$
proof(*intro image-subsetI*, *simp add*: φ-*def*)

fix *x* **assume** $x \in G\ //\ K$
then show $(SOME\ a.\ a \in G \wedge x = a \cdot|\ K) \in G$
using *Left-Coset-Class-unit representant-exists someI-ex* **by** (*metis* (*mono-tags*, *lifting*))
qed

lemma *phi-image-group*:
Group-Theory.group $(\varphi\ `\ (G\ //\ K))$ *quot-comp-alt* $(\varphi\ K)$
proof−
have *hmonoid*: *Group-Theory.monoid* $(\varphi\ `\ (G\ //\ K))$ *quot-comp-alt* $(\varphi\ K)$
proof
show $\bigwedge a\ b.\ a \in \varphi\ `\ (G\ //\ K) \Longrightarrow b \in \varphi\ `\ (G\ //\ K) \Longrightarrow$
quot-comp-alt a b $\in \varphi\ `\ (G\ //\ K)$ **using** *quot-comp-alt-def imageI phi-image-subset*
by (*metis Class-in-Partition Left-Coset-Class-unit composition-closed subset-iff*)
next
show $(\varphi\ K) \in \varphi\ `\ (G\ //\ K)$ **using** *φ-def Left-Coset-Class-unit imageI Normal-def* **by** *blast*
next
show $\bigwedge a\ b\ c.\ a \in \varphi\ `\ Partition \Longrightarrow b \in \varphi\ `\ Partition \Longrightarrow c \in \varphi\ `\ Partition \Longrightarrow$
quot-comp-alt (*quot-comp-alt a b*) *c* = *quot-comp-alt a* (*quot-comp-alt b c*)
proof−
fix *a b c* **assume** *ha*: $a \in \varphi\ `\ (G\ //\ K)$ **and** *hb*: $b \in \varphi\ `\ (G\ //\ K)$ **and** *hc*: $c \in \varphi\ `\ (G\ //\ K)$
have *habc*: $a \cdot b \cdot c \in G$ **using** *ha hb hc composition-closed phi-image-subset* **by** (*meson subsetD*)
have *hab*: *quot-comp-alt a b* $\in (a \cdot b) \cdot|\ K$ **using** *phi-image-subset quot-comp-alt-def ha hb*
by (*metis composition-closed phi-coset-mem subsetD*)
then have *quot-comp-alt* (*quot-comp-alt a b*) *c* $\in (a \cdot b \cdot c) \cdot|\ K$ **using** *quot-comp-alt-def phi-image-subset ha hb hc*
by (*smt* (*z3*) *Block-self Class-closed Class-in-Partition Left-Coset-Class-unit composition-closed*
natural.commutes-with-composition phi-coset-mem subset-iff)
moreover have *hbc*: *quot-comp-alt b c* $\in (b \cdot c) \cdot|\ K$ **using** *hb hc phi-image-subset quot-comp-alt-def*
by (*metis composition-closed phi-coset-mem subset-iff*)
moreover hence *quot-comp-alt a* (*quot-comp-alt b c*) $\in (a \cdot b \cdot c) \cdot|\ K$ **using** *quot-comp-alt-def phi-image-subset ha hb hc*
by (*smt* (*verit*, *del-insts*) *Block-self Class-closed Class-in-Partition Left-Coset-Class-unit*

associative composition-closed natural.commutes-with-composition phi-coset-mem subset-iff)
moreover have $a \cdot$ (*quot-comp-alt b c*) $\cdot|\ K \in G\ //\ K$ **using** *ha hb hc phi-image-subset*
by (*metis Class-closed Class-in-Partition Class-is-Left-Coset hbc composition-closed in-mono subset-eq*)
moreover have (*quot-comp-alt a b*) $\cdot\ c \cdot|\ K \in G\ //\ K$ **using** *ha hb hc*

phi-image-subset
by (*metis Class-closed Class-in-Partition Left-Coset-Class-unit hab composition-closed in-mono*)
ultimately show *quot-comp-alt* (*quot-comp-alt a b*) *c* = *quot-comp-alt a* (*quot-comp-alt b c*)
using *phi-mem-coset-eq*[*OF - habc*] *quot-comp-alt-def* **by** *metis*
qed
next
show $\bigwedge a.\ a \in \varphi$ ‘ *Partition* $\Longrightarrow$ *quot-comp-alt* (φ *K*) *a* = *a* **using** *quot-comp-alt-def* φ*-def*
phi-image-subset image-iff phi-coset-eq subsetD **by** (*smt* (*z3*) *Normal-def Partition-def*
natural.image.sub-unit-closed phi-comp-eq quotient.left-unit)
next
show $\bigwedge a.\ a \in \varphi$ ‘ *Partition* $\Longrightarrow$ *quot-comp-alt a* (φ *K*) = *a* **using** *quot-comp-alt-def* φ*-def*
phi-image-subset image-iff phi-coset-eq subsetD **by** (*smt* (*verit*) *Normal-def*
factor-composition factor-unit normal-subgroup.phi-coset-eq-self normal-subgroup-axioms

quotient.unit-closed right-unit unit-closed)
qed
moreover show *Group-Theory.group* (φ ‘ (*G* // *K*)) *quot-comp-alt* (φ *K*)
proof(*simp add*: *group-def group-axioms-def hmonoid*)
show $\forall u.\ u \in \varphi$ ‘ *Partition* $\longrightarrow$ *monoid.invertible* (φ ‘ *Partition*) *quot-comp-alt* (φ *K*) *u*
proof(*intro allI impI*)
fix *g* **assume** *hg*: $g \in \varphi$ ‘ (*G* // *K*)
then have *quot-comp-alt g* (φ ((*inverse g*) ·| *K*)) = (φ *K*)
and *quot-comp-alt* (φ ((*inverse g*) ·| *K*)) *g* = (φ *K*)
using *phi-image-subset phi-inverse-right phi-inverse-left* **by** *auto*
moreover have φ ((*inverse g*) ·| *K*) $\in \varphi$ ‘ (*G* // *K*) **using** *imageI hg phi-image-subset*
by (*metis* (*no-types*, *opaque-lifting*) *Class-in-Partition Left-Coset-Class-unit in-mono*
invertible invertible-inverse-closed)
ultimately show *monoid.invertible* (φ ‘ *Partition*) *quot-comp-alt* (φ *K*) *g*
using *monoid.invertibleI*[*OF hmonoid*] *hg* **by** *presburger*
qed
qed
qed

lemma *phi-map*: *Set-Theory.map* φ *Partition* (φ ‘ *Partition*)
by (*auto simp add*: *Set-Theory.map-def* φ*-def*)

lemma *phi-image-isomorphic*:
group-isomorphism φ (*G* // *K*) ([·]) (*Class* **1**) (φ ‘ (*G* // *K*)) *quot-comp-alt* (φ *K*)
proof −
have *bijective-map* φ *Partition* (φ ‘ *Partition*)

using *bijective-map-def bijective-def bij-betw-def phi-inj-on phi-map* **by** *blast*
moreover have *Group-Theory.monoid* (φ ‘ *Partition*) *quot-comp-alt* (φ *K*)
using *phi-image-group group-def* **by** *metis*
moreover have φ (*Class* **1**) = φ *K* **using** *Left-Coset-Class-unit Normal-def* **by** *auto*
ultimately show *?thesis*
by (*auto simp add*: *group-isomorphism-def group-homomorphism-def monoid-homomorphism-def*
phi-image-group quotient.monoid-axioms quotient.group-axioms monoid-homomorphism-axioms-def
phi-comp-eq phi-map)
qed

end

context *subgroup-of-additive-abelian-group*

begin

lemma *Union-Coset-card-eq*:
assumes *hSG*: $S \subseteq G$ **and** *hSU*: ($\bigcup$ (*Class* ‘ *S*)) = *S*
shows *card S* = *card H* ∗ *card* (*Class* ‘ *S*)
proof(*cases finite H*)
case *hH*: *True*
have *hfin*: $\bigwedge A.\ A \in$ *Class* ‘ *S* $\Longrightarrow$ *finite A* **using** *hSG Right-Coset-Class-unit Right-Coset-cardinality hH card-eq-0-iff empty-iff sub-unit-closed subsetD*
by (*smt* (*verit*, *del-insts*) *imageE*)
have *card S* = *card H* ∗ *card* (*Class* ‘ *S*) **when** *hS*: *finite S*
proof−
have *hdisj*: *pairwise* ($\lambda s\ t.$ *disjnt s t*) (*Class* ‘ *S*)
proof (*intro pairwiseI*)
fix *x y* **assume** $x \in$ *Class* ‘ *S* **and** $y \in$ *Class* ‘ *S* **and** *hxy*: $x \neq y$
then obtain *a b* **where** *x* = *Class a* **and** *y* = *Class b* **and**
$a \in S$ **and** $b \in S$ **by** *blast*
then show *disjnt x y* **using** *disjnt-def hxy*
by (*smt* (*verit*, *ccfv-threshold*) *not-disjoint-implies-equal hSG subsetD*)
qed
then have *card* ($\bigcup$ (*Class* ‘ *S*)) = *sum card* (*Class* ‘ *S*) **using** *card-Union-disjoint hfin* **by** *blast*
moreover have *finite* (*Class* ‘ *S*) **using** *hS* **by** *blast*
ultimately have *card* ($\bigcup$ (*Class* ‘ *S*)) = ($\sum$ *a* $\in$ *Class* ‘ *S*. *card a*)
using *sum-card-image hdisj* **by** *blast*
moreover have $\bigwedge$ *a*. $a \in$ *Class* ‘ *S* $\Longrightarrow$ *card a* = *card H*
using *hSG Right-Coset-Class-unit Right-Coset-cardinality* **by** *auto*
ultimately show *card S* = *card H* ∗ *card* (*Class* ‘ *S*)
using *hSU* **by** *simp*
qed
moreover have *card S* = *card H* ∗ *card* (*Class* ‘ *S*) **when** *hS*: ¬ *finite S*
using *finite-Union hfin hS hSU* **by** (*metis card-eq-0-iff mult-0-right*)

ultimately show *?thesis* **by** *blast*
next
case *hH*: *False*
have *card S = card H ∗ card (Class ‘ S)* **when** *S* = {}
by (*simp add*: *that*)
then have *hinf*: ⋀ *A*. *A* ∈ *Class ‘ S* ⟹ *infinite A* **using** *hSG Right-Coset-Class-unit*

Right-Coset-cardinality hH card-eq-0-iff empty-iff sub-unit-closed subsetD
by (*smt* (*verit*) *Class-self imageE*)
moreover have *card S = card H ∗ card (Class ‘ S)* **when** *S* ≠ {} **using** *hSU*
by (*metis Class-closed2*
Normal-def card.infinite card-sumset-0-iff hH hSG mult-is-0 sumset-subgroup-eq-Class-Union unit-closed)
ultimately show *?thesis* **by** *fastforce*
qed

end

context *subgroup-of-abelian-group*
begin

interpretation *GH*: *additive-abelian-group G // H* ([·]) *Class* **1**
proof
fix *x y* **assume** *x* ∈ *G // H* **and** *y* ∈ *G // H*
then show *x* [·] *y* = *y* [·] *x* **using** *Class-commutes-with-composition commutative representant-exists*
by *metis*
qed

interpretation *GH-repr*: *additive-abelian-group φ ‘ (G // H) quot-comp-alt φ H*
proof(*simp add*: *additive-abelian-group-def abelian-group-def phi-image-group commutative-monoid-def commutative-monoid-axioms-def*, *intro conjI allI impI*)
show *Group-Theory.monoid (φ ‘ Partition) quot-comp-alt (φ H)*
using *phi-image-group group-def* **by** *metis*
next
show ⋀ *x y*. *x* ∈ *φ ‘ Partition* ⟹ *y* ∈ *φ ‘ Partition* ⟹ *quot-comp-alt x y* = *quot-comp-alt y x*
by (*auto*) (*metis GH.commutative phi-comp-eq*)
qed

lemma *phi-image-sumset-eq*:
assumes *A* ⊆ *G // H* **and** *B* ⊆ *G // H*
shows *φ ‘ (GH.sumset A B) = GH-repr.sumset (φ ‘ A) (φ ‘ B)*
proof(*intro subset-antisym image-subsetI subsetI*)
fix *x* **assume** *x* ∈ *GH.sumset A B*
then obtain *c d* **where** *x = quotient-composition c d* **and** *hc*: *c* ∈ *A* **and** *hd*: *d* ∈ *B*
using *GH.sumset.cases* **by** *blast*
then have *φ x = quot-comp-alt (φ c) (φ d)*

using *phi-comp-eq assms subsetD* **by** *blast*
then show φ *x* ∈ *GH-repr.sumset* (φ ‘ *A*) (φ ‘ *B*)
using *hc hd assms subsetD GH-repr.sumsetI imageI* **by** *auto*
next
fix *x* **assume** *x* ∈ *GH-repr.sumset* (φ ‘ *A*) (φ ‘ *B*)
then obtain *a b* **where** *x* = *quot-comp-alt a b* **and** *ha*: *a* ∈ φ ‘ *A* **and** *hb*: *b* ∈ φ ‘ *B*
using *GH-repr.sumset.cases* **by** *metis*
moreover obtain *c d* **where** *a* = φ *c* **and** *b* = φ *d* **and** *c* ∈ *A* **and** *d* ∈ *B*
using *ha hb* **by** *blast*
ultimately show *x* ∈ φ ‘ *GH.sumset A B* **using** *phi-comp-eq assms imageI GH.sumsetI*
by (*smt* (*verit*, *del-insts*) *subsetD*)
qed

lemma *phi-image-stabilizer-eq*:
assumes *A* ⊆ *G* // *H*
shows φ ‘ (*GH.stabilizer A*) = *GH-repr.stabilizer* (φ ‘ *A*)
proof(*intro subset-antisym image-subsetI subsetI*)
fix *x* **assume** *x* ∈ *GH.stabilizer A*
then have *GH.sumset* {*x*} *A* = *A* **and** *hx*: *x* ∈ *G* // *H* **using** *GH.stabilizer-def assms* **by** *auto*
then have *GH-repr.sumset* (φ ‘ {*x*}) (φ ‘ *A*) = φ ‘ *A* **using** *assms phi-image-sumset-eq*
by (*metis empty-subsetI insert-subset*)
then show φ *x* ∈ *GH-repr.stabilizer* (φ ‘ *A*) **using** *GH-repr.stabilizer-def assms*
by (*smt* (*z3*) *GH-repr.sumset-Int-carrier hx image-empty image-eqI image-insert mem-Collect-eq*)
next
fix *x* **assume** *x* ∈ *GH-repr.stabilizer* (φ ‘ *A*)
then have *hstab*: *GH-repr.sumset* {*x*} (φ ‘ *A*) = (φ ‘ *A*) **and** *hx*: *x* ∈ φ ‘ (*G* // *H*)
using *GH-repr.stabilizer-def assms phi-image-subset* **by** *auto*
then obtain *B* **where** *hB*: *B* ∈ *G* // *H* **and** *hBx*: φ *B* = *x* **by** *blast*
then have *GH-repr.sumset* (φ ‘ {*B*}) (φ ‘ *A*) = φ ‘ *A* **using** *hstab* **by** *auto*
then have *GH.sumset* {*B*} *A* = *A* **using** *phi-image-sumset-eq phi-inj-on assms hB*
GH.sumset-subset-carrier **by** (*smt* (*z3*) *GH.sumset-singletons-eq inj-on-image-eq-iff*

quotient.right-unit quotient.unit-closed)
then show *x* ∈ φ ‘ (*GH.stabilizer A*) **using** *assms hBx GH.stabilizer-def*
by (*smt* (*z3*) *GH.sumset-Int-carrier hB image-iff mem-Collect-eq*)
qed

end

1.6 Useful group-theoretic results

lemma *residue-group*: *abelian-group* {*0*..(*m* :: *nat*)−*1*} (λ *x y*. ((*x* + *y*) *mod m*)) (*0* :: *int*)

proof(*cases m > 1*)
case *hm*: *True*
then have *hmonoid*: *Group-Theory.monoid* {*0..m−1*} (λ *x y*. ((*x* + *y*) *mod m*)) (*0* :: *int*)
by (*unfold-locales*, *auto simp add*: *of-nat-diff*, *presburger*)
moreover have *monoid.invertible* {*0..int* (*m* − *1*)} (λ*x y*. (*x* + *y*) *mod int m*) *0 u* **if** *u* ∈ {*0..int* (*m* − *1*)} **for** *u*
proof(*cases u* = *0*)
case *True*
then show *?thesis* **using** *monoid.invertible-def*[*OF hmonoid that*] *monoid.unit-invertible*[*OF hmonoid*] **by** *simp*
next
case *hx*: *False*
then have ((*m* − *u*) + *u*) *mod m* = *0* **and** (*u* + (*m* − *u*)) *mod m* = *0* **and** *m* − *u* ∈ {*0..int*(*m*−*1*)}
using *atLeastAtMost-iff hx that* **by** *auto*
then show *?thesis* **using** *monoid.invertible-def*[*OF hmonoid that*] **by** *metis*
qed
moreover have *commutative-monoid* {*0..m−1*} (λ *x y*. ((*x* + *y*) *mod m*)) (*0* :: *int*)
using *hmonoid commutative-monoid-def commutative-monoid-axioms-def* **by** (*smt* (*verit*))
ultimately show *?thesis* **by** (*simp add*: *abelian-group-def group-def group-axioms-def*

hmonoid)
next
case *hm*: *False*
moreover have *hmonoid*: *Group-Theory.monoid* {*0*} (λ *x y*. ((*x* + *y*) *mod m*)) (*0* :: *int*)
by (*unfold-locales*, *auto*)
moreover have *monoid.invertible* {*0*} (λ*x y*. (*x* + *y*) *mod int m*) *0 0* **using** *monoid.invertible-def*[*OF hmonoid*]
monoid.unit-invertible[*OF hmonoid*] *hm* **by** *simp*
ultimately show *?thesis* **by** (*unfold-locales*, *auto*)
qed

lemma (**in** *subgroup-of-group*) *prime-order-simple*:
assumes *prime* (*card G*)
shows *H* = {**1**} ∨ *H* = *G*
proof−
have *card H dvd card G* **using** *lagrange assms card.infinite dvdI not-prime-0* **by** *fastforce*
then have *card H* = *1* ∨ *card H* = *card G* **using** *assms prime-nat-iff* **by** *blast*
then show *?thesis* **using** *card-1-singletonE sub-unit-closed card.infinite card-subset-eq sub*
assms not-prime-0 subsetI insertE empty-iff **by** *metis*
qed

lemma *residue-group-simple*:

assumes *prime p* **and** *subgroup H* $\{0..(p :: nat)-1\}$ $(\lambda\ x\ y.\ ((x + y)\ mod\ p))$ $(0 :: int)$
shows $H = \{0\} \lor H = \{0..int(p-1)\}$
proof−
have *hprime*: *prime* (*card* $\{0..int(p-1)\}$) **using** *card-atLeastAtMost-int assms int-ops* **by** *auto*
moreover have *hsub*:*subgroup-of-group H* $\{0..(p :: nat)-1\}$ $(\lambda\ x\ y.\ ((x + y)$ $mod\ p))$ $(0 :: int)$
using *subgroup-of-group-def assms abelian-group-def residue-group* **by** *fast*
ultimately show *?thesis* **using** *assms subgroup-of-group.prime-order-simple*[*OF hsub hprime*] **by** *blast*
qed

end

2 Kneser's Theorem and the CauchyDavenport Theorem: main proofs

theory *Kneser-Cauchy-Davenport-main-proofs*
imports
Kneser-Cauchy-Davenport-preliminaries

begin

context *additive-abelian-group*

begin

2.1 Proof of Kneser's Theorem

The proof we formalise follows the paper [1]. This version of Kneser's Theorem corresponds to Theorem 3.2 in [3], or to Theorem 4.3 in [2].

theorem *Kneser*:
assumes $A \subseteq G$ **and** $B \subseteq G$ **and** *finite A* **and** *finite B* **and** *hAne*: $A \neq \{\}$ **and** *hBne*: $B \neq \{\}$
shows *card* (*sumset A B*) $\geq$ *card* (*sumset A* (*stabilizer* (*sumset A B*))) +
card (*sumset B* (*stabilizer* (*sumset A B*))) − *card* (*stabilizer* (*sumset A B*))
proof−
have $\bigwedge$ *n A B*. *additive-abelian-group G* $(\oplus)$ $\mathbf{0} \Longrightarrow A \subseteq G \Longrightarrow B \subseteq G \Longrightarrow$
finite A $\Longrightarrow$ *finite B* $\Longrightarrow A \neq \{\} \Longrightarrow B \neq \{\} \Longrightarrow$ *card* (*sumset A B*) + *card* $A = n \Longrightarrow$
card (*sumset A B*) $\geq$ *card* (*sumset A* (*stabilizer* (*sumset A B*))) +
card (*sumset B* (*stabilizer* (*sumset A B*))) − *card* ((*stabilizer* (*sumset A B*)))
proof−
fix *n*
show $\bigwedge$ *A B*. *additive-abelian-group G* $(\oplus)$ $\mathbf{0} \Longrightarrow A \subseteq G \Longrightarrow B \subseteq G \Longrightarrow$
finite A $\Longrightarrow$ *finite B* $\Longrightarrow A \neq \{\} \Longrightarrow B \neq \{\} \Longrightarrow$ *card* (*sumset A B*) + *card* $A = n \Longrightarrow$

card (*sumset* *A* *B*) ≥ *card* (*sumset* *A* (*stabilizer* (*sumset* *A* *B*))) +
card (*sumset* *B* (*stabilizer* (*sumset* *A* *B*))) − *card* ((*stabilizer* (*sumset* *A* *B*)))
proof(*induction n arbitrary* : *G* (⊕) **0** *rule*: *nat-less-induct*)
fix *n*
fix *A B G* :: ′*a set*
fix *add* (**infixl** [⊕] *65*)
fix *zero* ([**0**])
assume *hind*: ∀ *m*<*n*. ∀ *x xa xb* :: ′*a set*. ∀ *xc xd*.
additive-abelian-group xb xc xd ⟶ *x* ⊆ *xb* ⟶
xa ⊆ *xb* ⟶ *finite x* ⟶ *finite xa* ⟶ *x* ≠ {} ⟶ *xa* ≠ {} ⟶
card (*additive-abelian-group.sumset xb xc x xa*) + *card x* = *m* ⟶
card (*additive-abelian-group.sumset xb xc x* (*additive-abelian-group.stabilizer xb xc*
(*additive-abelian-group.sumset xb xc x xa*))) + *card* (*additive-abelian-group.sumset xb xc xa*
(*additive-abelian-group.stabilizer xb xc* (*additive-abelian-group.sumset xb xc x xa*))) −
card (*additive-abelian-group.stabilizer xb xc* (*additive-abelian-group.sumset xb xc x xa*))
≤ *card* (*additive-abelian-group.sumset xb xc x xa*) **and**
hGroupG: *additive-abelian-group G* ([⊕]) [**0**] **and** *hAG*: *A* ⊆ *G* **and** *hBG*: *B* ⊆ *G* **and**
hA: *finite A* **and** *hB*: *finite B* **and** *hAne*: *A* ≠ {} **and** *hBne*: *B* ≠ {} **and**
hcardsum: *card* (*additive-abelian-group.sumset G* ([⊕]) *A B*) + *card A* = *n*
interpret *G*: *additive-abelian-group G* ([⊕]) [**0**] **using** *hGroupG* **by** *simp*
have *hindG*: ∀ *m*<*n*. ∀ *C D*. *C* ⊆ *G* ⟶
D ⊆ *G* ⟶ *finite C* ⟶ *finite D* ⟶ *C* ≠ {} ⟶ *D* ≠ {} ⟶
card (*G.sumset C D*) + *card C* = *m* ⟶
card (*G.sumset C* (*G.stabilizer*
(*G.sumset C D*))) + *card* (*G.sumset D*
(*G.stabilizer* (*G.sumset C D*))) −
card (*G.stabilizer* (*G.sumset C D*))
≤ *card* (*G.sumset C D*) **using** *hind hGroupG* **by** *blast*
show *card* (*G.sumset A* (*G.stabilizer* (*G.sumset A B*))) +
card (*G.sumset B* (*G.stabilizer* (*G.sumset A B*))) −
card (*G.stabilizer* (*G.sumset A B*)) ≤ *card* (*G.sumset A B*)
proof(*cases G.stabilizer* (*G.sumset A B*) = {[**0**]})
case *hstab0*: *True*
show *?thesis*
proof (*cases card A* = *1*)
case *True*
then obtain *a* **where** *A* = {*a*} **and** *a* ∈ *G*
by (*metis hAG card-1-singletonE insert-subset*)
then show *?thesis* **using** *G.card-sumset-singleton-subset-eq*
G.stabilizer-left-sumset-invariant G.stabilizer-subset-group G.sumset-commute

G.sumset-stabilizer-eq-self hBG **by** (*metis diff-add-inverse eq-imp-le*)
next
case *False*

```
          then have card A ≥ 2 using Suc-1 order-antisym-conv
                 by (metis Suc-eq-plus1 bot.extremum card-seteq hA hAne le-add2
not-less-eq-eq)
         then obtain a a′ where haA: {a′, a} ⊆ A and hanot: a′ ≠ a and ha1G:
a ∈ G and
            ha2G: a′ ∈ G
            by (smt (verit, ccfv-threshold) card-2-iff hAG insert-subset
               obtain-subset-with-card-n subset-trans)
       then have (a′ [⊕] (G.inverse a)) ∉ G.stabilizer B using G.stabilizer-sub-sumset-right

            hstab0 subset-singletonD by (metis G.commutative G.inverse-closed
            G.invertible G.invertible-right-inverse2 G.right-unit empty-iff insert-iff)
          then obtain b where hb: b ∈ B and (a′ [⊕] (G.inverse a)) [⊕] b ∉ B
            using G.stabilizer-def G.not-mem-stabilizer-obtain hBG hB hBne ha1G
ha2G
            G.composition-closed G.inverse-closed by (metis (no-types, lifting))
          then have habG: a′ [⊕] b [⊕] G.inverse a ∉ B
               using hb hBG ha1G ha2G by (metis G.associative G.commutative
G.inverse-closed subset-iff)
          have hbG: b ∈ G using hb hBG by auto
          let ?B′ = G.sumset (G.differenceset B {b}) {a}
          have hB′: finite ?B′ using hB
            by (simp add: G.finite-minusset G.finite-sumset)
          have hB′G: ?B′ ⊆ G using G.sumset-subset-carrier by blast
         have hB′ne: ?B′ ≠ {} using hBne hbG ha1G sumset-is-empty-iff hBG by
auto
           have hstabeq: G.stabilizer (G.sumset A B) = G.stabilizer (G.sumset A
?B′)
            using hbG ha1G hAG hBG G.stabilizer-unchanged by blast
         have hstab0′: G.stabilizer (G.sumset A ?B′) = {[0]} using hstab0 hstabeq
by blast
          have ha1B′: a ∈ ?B′
          proof−
                have (b [⊕] G.inverse b) [⊕] a ∈ ?B′ using hBG hbG ha1G hb
G.minusset.minussetI by blast
            thus a ∈ ?B′ by (simp add: hbG ha1G)
          qed
          then have hinter-nempty: A ∩ ?B′ ≠ {} using ha1B′ haA by blast
          have ha2B′: a′ ∉ ?B′
          proof−
            have h1: (a′ [⊕] b [⊕] G.inverse a) [⊕] G.inverse b ∉ G.differenceset B
{b}
            proof
              assume (a′ [⊕] b [⊕] G.inverse a) [⊕] G.inverse b ∈ G.differenceset B
{b}
              then obtain b′ where (a′ [⊕] b [⊕] G.inverse a) [⊕] G.inverse b = b′
[⊕] G.inverse b
                 and b′ ∈ B using hbG G.minusset.simps G.sumset.cases by force
              then have (a′ [⊕] b [⊕] G.inverse a) ∈ B using hbG
```

by (*smt* (*verit*) *G.composition-closed hBG ha1G ha2G G.inverse-closed G.invertible*
G.invertible-right-cancel subsetD)
thus *False* **using** *habG* **by** *auto*
qed
have $((a' [\oplus] b [\oplus] G.inverse\ a) [\oplus] G.inverse\ b) [\oplus] a \notin ?B'$
proof
assume $((a' [\oplus] b [\oplus] G.inverse\ a) [\oplus] G.inverse\ b) [\oplus] a \in ?B'$
then obtain b' **where** $((a' [\oplus] b [\oplus] G.inverse\ a) [\oplus] G.inverse\ b) [\oplus] a = b' [\oplus] a$
and $b' \in G.differenceset\ B\ \{b\}$ **using** *ha1G G.sumset.simps* **by** *auto*
then have $((a' [\oplus] b [\oplus] G.inverse\ a) [\oplus] G.inverse\ b) \in G.differenceset\ B\ \{b\}$
using *ha1G* **by** (*smt* (*z3*) *G.sumset.simps G.additive-abelian-group-axioms*

G.composition-closed ha2G hbG G.inverse-closed G.invertible G.invertible-right-cancel)
thus *False* **using** *h1* **by** *auto*
qed
thus $a' \notin ?B'$ **using** *ha1G hbG*
by (*smt* (*verit, del-insts*) *G.associative G.commutative G.composition-closed ha2G*
G.inverse-closed G.invertible G.invertible-left-inverse G.right-unit)
qed
have *hinterA*: $A \cap ?B' \neq A$ **using** *haA ha2B'* **by** *auto*
have *hintercard0*: $0 < card\ (A \cap ?B')$
using *hA hB hinter-nempty card-gt-0-iff* **by** *blast*
have *hintercardA*: $card\ (A \cap ?B') < card\ A$ **using** *hA hB hinterA card-mono*
by (*simp add*: *psubsetI psubset-card-mono*)
have *inj*: *inj-on* $(\lambda\ x.\ x\ [\oplus]\ G.inverse\ b\ [\oplus]\ a)$ *G* **using** *inj-onI ha1G hb G.invertible*
G.inverse-closed G.composition-closed **by** (*smt* (*verit*) *G.invertible-right-cancel hbG*)
have *1*: $card\ (G.sumset\ A\ B) = card\ (G.sumset\ A\ ?B')$
using *G.card-differenceset-singleton-mem-eq G.card-sumset-singleton-subset-eq hAG hB'G*
ha1G hbG G.sumset-assoc G.sumset-commute G.sumset-subset-carrier
by (*smt* (*verit, del-insts*))
obtain *C* **where** *hCconv*: $G.convergent\ C\ A\ ?B'$ **and** *hCmin*: $\bigwedge\ Y.\ Y \in G.convergent\text{-}set\ A\ ?B'$
$\Longrightarrow card\ (G.stabilizer\ Y) \geq card\ (G.stabilizer\ C)$
using *G.exists-convergent-min-stabilizer*[*of n A ?B'*]
hindG hA hB' hAG hB'G hinter-nempty hAne hcardsum hintercardA 1 hGroupG **by** *auto*
have *hC*: $C \subseteq G.sumset\ A\ ?B'$ **and** *hCne*: $C \neq \{\}$ **and**
hCcard: $card\ C + card\ (G.stabilizer\ C) \geq$
$card\ (A \cap ?B') + card\ (G.sumset\ (A \cup ?B')\ (G.stabilizer\ C))$
using *G.convergent-def hCconv* **by** *auto*
then have *hCfinite*: *finite C* **using** *hC G.finite-sumset hA hB'*

by (*meson finite-subset*)
have *htranslate*: *card* (*G.sumset A* {[**0**]}) + *card* (*G.sumset ?B′* {[**0**]}) − *card* {[**0**]} ≤
card (*G.sumset A ?B′*)
proof(*cases G.stabilizer C* = {[**0**]})
case *hCstab0*: *True*
have *card* (*G.sumset A* {[**0**]}) + *card* (*G.sumset ?B′* {[**0**]}) − *card* {[**0**]} = *card A* + *card ?B′* −
card {[**0**]} **using** *hAG hB′G G.card-minusset′* **by** *fastforce*
also have ... = *card* (*A* ∩ *?B′*) + *card* (*A* ∪ *?B′*) − *card* {[**0**]}
using *hA hB′ card-Un-Int* **by** *fastforce*
also have ... = *card* (*A* ∩ *?B′*) + *card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) − *card* {[**0**]}
by (*simp add*: *Int-absorb1 Int-commute hAG G.sumset-subset-carrier*)
also have ... ≤ *card C* **using** *hCcard hCstab0* **by** *auto*
also have ... ≤ *card* (*G.sumset A ?B′*)
using *hC card-mono G.finite-sumset hA hB′* **by** *metis*
finally show *?thesis* **by** *simp*
next
case *hCstab-ne0*: *False*
have *hCG*: *C* ⊆ *G* **using** *hC* **by** (*meson subset-trans G.sumset-subset-carrier*)
then have *hstabC*: *finite* (*G.stabilizer C*) **using** *G.stabilizer-finite hCne hC*
G.finite-sumset hA hB′ **by** (*metis Nat.add-0-right add-leE card.infinite hCcard*
hintercard0 le-0-eq not-gr0)
then have *hcardstabC-gt-1*: *card* (*G.stabilizer C*) > *1* **using** *G.zero-mem-stabilizer*
hCstab-ne0 hCG **by** (*metis card-1-singletonE card-gt-0-iff diffs0-imp-equal empty-iff*
gr-zeroI insertE less-one zero-less-diff)
have *G.sumset* (*G.stabilizer C*) (*G.sumset A ?B′*) ≠ *G.sumset A ?B′*
using *G.finite-sumset G.stabilizer-is-nonempty G.stabilizer-subset-group*
G.sumset-eq-sub-stabilizer G.sumset-subset-carrier hA hB′ hCstab-ne0 hstab0′
subset-singletonD **by** *metis*
moreover have *card* (*G.sumset* (*G.stabilizer C*) (*G.sumset A ?B′*)) ≥ *card* (*G.sumset A ?B′*)
using *G.card-le-sumset G.finite-sumset hA hB′ hstabC*
by (*meson hCG G.sumset-subset-carrier G.unit-closed G.zero-mem-stabilizer*)
ultimately have ¬ *G.sumset* (*G.stabilizer C*) (*G.sumset A ?B′*) ⊆ *G.sumset A ?B′*
using *G.finite-sumset hA hB′ card-seteq* **by** *metis*
then obtain *x* **where** *hx1*: *x* ∈ *G.sumset* (*G.stabilizer C*) (*G.sumset A ?B′*) **and**
hx2: *x* ∉ *G.sumset A ?B′* **by** *auto*
then obtain *a1 b1 c* **where** *x* = *c* [⊕] (*a1* [⊕] *b1*) **and** *c* ∈ *G.stabilizer C* **and**
ha1A: *a1* ∈ *A* **and** *hb1B′*: *b1* ∈ *?B′* **and** *ha1memG*: *a1* ∈ *G* **and** *hb1memG*: *b1* ∈ *G*

by (*metis* (*no-types*, *lifting*) *G.sumset.cases*)
then have *hx*: $x \in$ *G.sumset* (*G.stabilizer C*) {*a1* [$\oplus$] *b1*}
using *hx1* **by** (*meson G.composition-closed hAG hB′G insertI1 G.stabilizer-subset-group*
subsetD G.sumset.sumsetI)
then have *hnotsubAB*: ¬ *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ⊆ *G.sumset A ?B′*
using *hx2 G.sumset-commute* **by** *auto*
let *?A1* = *A* ∩ (*G.sumset* {*a1*} (*G.stabilizer C*))
let *?A2* = *A* ∩ (*G.sumset* {*b1*} (*G.stabilizer C*))
let *?B1* = *?B′* ∩ (*G.sumset* {*b1*} (*G.stabilizer C*))
let *?B2* = *?B′* ∩ (*G.sumset* {*a1*} (*G.stabilizer C*))
let *?C1* = *C* ∪ (*G.sumset ?A1 ?B1*)
let *?C2* = *C* ∪ (*G.sumset ?A2 ?B2*)
let *?H1* = *G.stabilizer* (*G.sumset ?A1 ?B1*)
let *?H2* = *G.stabilizer* (*G.sumset ?A2 ?B2*)
have *hA1ne*: *?A1* ≠ {} **using** *ha1A G.zero-mem-stabilizer hCG*
by (*metis* (*full-types*) *disjoint-iff-not-equal hAG insertCI G.right-unit subset-eq*
G.sumset.sumsetI G.unit-closed)
have *hB1ne*: *?B1* ≠ {} **using** *hb1B′ G.zero-mem-stabilizer hCG*
by (*metis G.composition-closed disjoint-iff-not-equal insertCI G.left-unit*

G.sumset.cases G.sumset.sumsetI G.sumset-commute G.unit-closed)
have *hnotsubC*: ¬ *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ⊆ *C* **using** *hnotsubAB hC* **by** *blast*
have *habstabempty*: *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ∩ *C* = {}
proof(*rule ccontr*)
assume *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ∩ *C* ≠ {}
then obtain *z* **where**
hz: *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ∩ *G.sumset* {*z*} (*G.stabilizer C*) ≠ {} **and**
hzC: *z* ∈ *C* **using** *G.stabilizer-coset-Un hCG* **by** *blast*
then have *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) = *G.sumset* {*z*} (*G.stabilizer C*) **using** *hz*
G.sumset-stabilizer-eq-iff hCG G.sumset.simps hx **by** *auto*
then have *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) ⊆ *C* **using** *hzC*
by (*simp add*: *hCG G.stabilizer-coset-subset*)
thus *False* **using** *hnotsubC* **by** *simp*
qed
have *hA1B1sub*: *G.sumset ?A1 ?B1* ⊆ *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*) **and**
hB2A2sub: *G.sumset ?B2 ?A2* ⊆ *G.sumset* {*a1* [$\oplus$] *b1*} (*G.stabilizer C*)
using *G.sumset-Inter-subset-sumset ha1memG hb1memG* **by** *auto*
then have *hA1B1Cempty*: *G.sumset ?A1 ?B1* ∩ *C* = {} **using** *habstabempty* **by** *blast*
then have *hcardC1*: *card ?C1* = *card C* + *card* (*G.sumset ?A1 ?B1*) **using** *card-Un-disjoint*

G.finite-sumset hA hB′ finite-Int hC finite-subset Int-commute **by** *metis*
have *hA1B1cardless*: *card* (*G.sumset ?A1 ?B1*) < *card* (*G.sumset A B*)
proof−
have *G.sumset ?A1 ?B1* ⊆ *G.sumset A ?B′* **using** *G.sumset-mono* **by** *auto*
moreover have *G.sumset ?A1 ?B1* ≠ *G.sumset A ?B′*
using *hA1B1Cempty hC hCne hA1B1sub* **by** *auto*
ultimately show *card* (*G.sumset ?A1 ?B1*) < *card* (*G.sumset A B*)
using *G.finite-sumset hA hB′ psubset-card-mono psubset-eq 1* **by** *metis*
qed
have *hB2A2Cempty*: *G.sumset ?B2 ?A2* ∩ *C* = {} **using** *habstabempty hB2A2sub* **by** *blast*
then have *hcardC2*: *card ?C2* = *card C* + *card* (*G.sumset ?B2 ?A2*) **using** *card-Un-disjoint*
G.finite-sumset hA hB′ finite-Int hC finite-subset Int-commute G.sumset-commute **by** *metis*
have *hA2B2cardless*: *card* (*G.sumset ?A2 ?B2*) < *card* (*G.sumset A B*)
proof−
have *G.sumset ?A2 ?B2* ⊂ *G.sumset A ?B′*
using *G.sumset-mono hB2A2Cempty hC hCne hB2A2sub G.sumset-commute psubset-eq*
by (*metis Int-absorb1 Int-lower1*)
then show *?thesis* **by** (*simp add*: *1 G.finite-sumset hA hB′ psubset-card-mono*)
qed
have *card ?A1* ≤ *card A* **using** *card-mono hA* **by** (*metis Int-lower1*)
then have *card* (*G.sumset ?A1 ?B1*) + *card ?A1* < *card* (*G.sumset A B*) + *card A*
using *hA1B1cardless* **by** *linarith*
then obtain *l* **where** *hln*: *l* < *n* **and** *hind1*: *card* (*G.sumset ?A1 ?B1*) + *card ?A1* = *l*
using *hcardsum* **by** *auto*
have *card ?A2* ≤ *card A* **using** *card-mono hA Int-lower1* **by** *metis*
then have *card* (*G.sumset ?A2 ?B2*) + *card ?A2* < *card* (*G.sumset A B*) + *card A*
using *hA2B2cardless* **by** *linarith*
then obtain *k* **where** *hkn*: *k* < *n* **and** *hind2*: *card* (*G.sumset ?A2 ?B2*) + *card ?A2* = *k*
using *hcardsum* **by** *auto*
have *hH1stabC*: *?H1* ⊂ *G.stabilizer C* **using** *G.stabilizer-sumset-psubset-stabilizer*
hA1ne hB1ne ha1memG hb1memG hnotsubAB **by** *presburger*
then have *card ?H1* < *card* (*G.stabilizer C*) **using** *psubset-card-mono hstabC* **by** *auto*
moreover have *hC1H1*: *?H1* = *G.stabilizer ?C1* **using** *G.stabilizer-eq-stabilizer-union*
by (*metis Int-commute hA hA1B1Cempty hA1ne hB′ hB1ne hC hCfinite hCne ha1memG hb1memG hnotsubAB*)
ultimately have *hC1notconv*: ¬ *G.convergent ?C1 A ?B′* **using** *hCmin*

G.convergent-set-def
le-antisym less-imp-le-nat less-not-refl2 **by** *fastforce*
have *hC1ne*: *?C1* ≠ {} **and** *hC2ne*: *?C2* ≠ {} **using** *hCne* **by** *auto*
have *hC1AB′*: *?C1* ⊆ *G.sumset A ?B′* **and** *hC2AB′*: *?C2* ⊆ *G.sumset A ?B′* **using** *hC*
by (*auto simp add*: *G.sumset-mono*)
have *hA1G*: *?A1* ⊆ *G* **and** *hA1* : *finite ?A1* **and** *hB1G*: *?B1* ⊆ *G* **and** *hB1*: *finite ?B1*
using *hAG hB′G hA hB′ finite-Int* **by** *auto*
then have *hindA1B1*: *card* (*G.sumset ?A1 ?H1*) + *card* (*G.sumset ?B1 ?H1*) − *card ?H1* ≤
card (*G.sumset ?A1 ?B1*) **using** *hindG hGroupG hA1ne hB1ne hind1 hln hAG hB′G*
hA hB′ **by** *metis*
have *hC1notconv-ineq*:
(*int* (*card ?C1*) + *card ?H1* − *card* (*A* ∩ *?B′*)) < *int* (*card* (*G.sumset* (*A* ∪ *?B′*) *?H1*))
using *hC1notconv hC1ne hC1AB′ hC1H1 G.convergent-def* **by** *auto*
have *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*)
≤ (*int* (*card C*) + *card* (*G.stabilizer C*) − *card* (*A* ∩ *?B′*)) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*)
using *hCcard* **by** *linarith*
then have *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*) <
(*int* (*card C*) + *card* (*G.stabilizer C*) − *card* (*A* ∩ *?B′*)) −
(*int* (*card ?C1*) + *card ?H1* − *card* (*A* ∩ *?B′*)) **using** *hC1notconv-ineq* **by** *linarith*
also have ... = *int* (*card* (*G.stabilizer C*)) − *card ?H1* − *card* (*G.sumset ?A1 ?B1*)
using *hcardC1* **by** *presburger*
also have ... ≤ *int* (*card* (*G.stabilizer C*)) − *card* (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*)
using *hindA1B1* **by** *linarith*

Finally, we deduce the inequality that is referred to as inequality (1) in [1] for *A* ∩ *G.sumset* {*a1*} (*G.stabilizer C*) and *G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*b1*} (*G.stabilizer C*).

finally have *hA1B1ineq*: *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) −
card (*G.sumset* (*A* ∪ *?B′*) *?H1*) < *int* (*card* (*G.stabilizer C*)) −
card (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*) **by** *simp*
have *hB2ne*: *?B2* ≠ {} **using** *G.sumset-inter-ineq hA1B1ineq ha1A ha1G hA hB′ hAne hB′ne*
hstabC hH1stabC ha1memG of-nat-0-le-iff **by** (*smt* (*verit*, *del-insts*))
have *hA2ne*: *?A2* ≠ {} **using** *G.sumset-inter-ineq*[*of A b1 C ?B′ a1*] *hA1B1ineq*
hb1B′ hb1memG hA hB′ hAne hB′ne hstabC hH1stabC of-nat-0-le-iff G.sumset-commute Un-commute **by** (*smt* (*verit*, *ccfv-SIG*))

have *hH2stabC*: *?H2* ⊂ *G.stabilizer C* **using** *G.stabilizer-sumset-psubset-stabilizer*

G.commutative hA2ne hB2ne ha1memG hb1memG hnotsubAB **by** *presburger*
then have *card ?H2* < *card* (*G.stabilizer C*) **using** *psubset-card-mono hstabC* **by** *auto*
moreover have *hC2H2*: *?H2* = *G.stabilizer ?C2* **using** *G.stabilizer-eq-stabilizer-union*
by (*smt* (*verit*, *ccfv-threshold*) *G.sumset-commute Int-commute hA hA2ne hB′ hB2A2Cempty*
hB2ne hC hCfinite hCne ha1memG hb1memG hnotsubAB)
ultimately have *hC2notconv*: ¬ *G.convergent ?C2 A ?B′*
using *hCmin G.convergent-set-def le-antisym less-imp-le-nat less-not-refl2*

by *fastforce*
have *hA2G*: *?A2* ⊆ *G* **and** *hA2* : *finite ?A2* **and** *hB2G*: *?B2* ⊆ *G* **and** *hB2*: *finite ?B2*
using *hAG hB′G hA hB′ finite-Int* **by** *auto*
then have *hindA2B2*: *card* (*G.sumset ?A2 ?H2*) + *card* (*G.sumset ?B2 ?H2*) − *card ?H2* ≤
card (*G.sumset ?A2 ?B2*)
using *hindG hGroupG hA2ne hB2ne hind2 hkn hAG hB′G hA hB′* **by** *metis*
have *hC2notconv-ineq*:
(*int* (*card ?C2*) + *card ?H2* − *card* (*A* ∩ *?B′*)) < *int* (*card* (*G.sumset* (*A* ∪ *?B′*) *?H2*))
using *hC2notconv hC2ne hC2AB′ hC2H2 G.convergent-def* **by** *auto*
have *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H2*)
≤ (*int* (*card C*) + *card* (*G.stabilizer C*) − *card* (*A* ∩ *?B′*)) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H2*)
using *hCcard* **by** *linarith*
then have *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H2*) <
(*int* (*card C*) + *card* (*G.stabilizer C*) − *card* (*A* ∩ *?B′*)) −
(*int* (*card ?C2*) + *card ?H2* − *card* (*A* ∩ *?B′*)) **using** *hC2notconv-ineq* **by** *linarith*
also have ... = *int* (*card* (*G.stabilizer C*)) − *card ?H2* − *card* (*G.sumset ?A2 ?B2*)
using *hcardC2 G.sumset-commute* **by** *simp*
also have ... ≤ *int* (*card* (*G.stabilizer C*)) − *card* (*G.sumset ?A2 ?H2*) − *card* (*G.sumset ?B2 ?H2*)
using *hindA2B2* **by** *linarith*

Analogously, we deduce the inequality that is referred to as inequality (1) in [1] for *A* ∩ *G.sumset* {*b1*} (*G.stabilizer C*) and *G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*a1*} (*G.stabilizer C*).

finally have *hA2B2ineq*: *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) −
card (*G.sumset* (*A* ∪ *?B′*) *?H2*) < *int* (*card* (*G.stabilizer C*)) −

card (*G.sumset ?A2 ?H2*) − *card* (*G.sumset ?B2 ?H2*) **by** *simp*
have *hfinsumH2*:*finite* (*G.sumset* (*A* ∪ *?B′*) *?H2*)
using *G.finite-sumset hA hB′ finite-UnI hstabC hH2stabC finite-subset psubset-imp-subset*
by *metis*
have *hsubsumH2*: *G.sumset* (*A* ∪ *?B′*) *?H2* ⊆ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*)
using *G.sumset.cases hAG hB′G G.stabilizer-subset-group hH2stabC psubset-imp-subset*
by (*smt* (*verit*, *best*) *subset-Un-eq G.sumset-commute G.sumset-subset-Un1*)
then have *hsumH2card-le*: *card* (*G.sumset* (*A* ∪ *?B′*) *?H2*) ≤
card (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))
using *card-mono G.finite-sumset G.stabilizer-finite hC hCne hCG hA hB′*
by (*metis finite-UnI hstabC*)
have *hfinsumH1*: *finite* (*G.sumset* (*A* ∪ *?B′*) *?H1*)
using *finite-sumset finite-Un psubsetE* **by** (*metis G.finite-sumset G.stabilizer-finite*
G.sumset-subset-carrier Int-Un-eq(*4*) *hA hA1 hB′ hB1 hC1H1 hH1stabC habstabempty*)
have *hsubsumH1*: *G.sumset* (*A* ∪ *?B′*) *?H1* ⊆ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*)
using *G.sumset.cases psubsetE subset-refl G.sumset-mono hH1stabC* **by** *simp*
have *hsumH1card-le*: *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*) ≤ *card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))
using *card-mono G.finite-sumset G.stabilizer-finite* **by** (*metis finite-UnI hA hB′ hstabC hsubsumH1*)
have *ha1b1stabCne*: *G.sumset* {*a1*} (*G.stabilizer C*) ≠ *G.sumset* {*b1*} (*G.stabilizer C*)
proof
assume *ha1b1*: *G.sumset* {*a1*} (*G.stabilizer C*) = *G.sumset* {*b1*} (*G.stabilizer C*)
have *hfin*: *finite* (*G.sumset ?A1 ?H1* ∪ *G.sumset ?B1 ?H1*)
using *finite-UnI G.finite-sumset hA1 hB1 hH1stabC hstabC psubset-imp-subset*
by (*metis finite-subset*)
have *int* (*card* (*G.sumset* {*a1*} (*G.stabilizer C*))) − *card* (*G.sumset ?A1 ?H1*) −
card (*G.sumset ?B1 ?H1*) ≤ *int* (*card* (*G.sumset* {*a1*} (*G.stabilizer C*))) −
card (*G.sumset ?A1 ?H1* ∪ *G.sumset ?B1 ?H1*)
using *card-Un-le*[*of G.sumset ?A1 ?H1 G.sumset ?B1 ?H1*] **by** *linarith*
also have ... ≤ *card* (*G.sumset* {*a1*} (*G.stabilizer C*) − (*G.sumset ?A1 ?H1* ∪ *G.sumset ?B1 ?H1*))
using *diff-card-le-card-Diff*[*of G.sumset ?A1 ?H1* ∪ *G.sumset ?B1 ?H1*
G.sumset {*a1*} (*G.stabilizer C*)] *hfin* **by** *linarith*
also have ... ≤ *card* (*G.sumset* {*a1*} (*G.stabilizer C*) − *G.sumset* (*?A1*

∪ *?B1*) *?H1*)
using *G.sumset-subset-Un1* **by** *auto*
also have ... ≤ *card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) − *G.sumset* (*A* ∪ *?B′*) *?H1*)
proof−
have *hsub*: *G.sumset* {*a1*} (*G.stabilizer C*) − *G.sumset* (*?A1* ∪ *?B1*) *?H1* ⊆
G.sumset (*A* ∪ *?B′*) (*G.stabilizer C*) − *G.sumset* (*A* ∪ *?B′*) *?H1*
proof
fix *x* **assume** *hx*: *x* ∈ *G.sumset* {*a1*} (*G.stabilizer C*) − *G.sumset* (*?A1* ∪ *?B1*) *?H1*
then obtain *c* **where** *hxac*: *x* = *a1* [⊕] *c* **and** *hc*: *c* ∈ *G.stabilizer C*
using *G.sumset.cases* **by** *blast*
then have *x* ∈ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) **using** *ha1A hAG*
G.stabilizer-subset-group **by** (*simp add*: *subset-iff G.sumset.sumsetI*)
moreover have *x* ∉ *G.sumset* (*A* ∪ *?B′*) *?H1*
proof
assume *hx1*: *x* ∈ *G.sumset* (*A* ∪ *?B′*) *?H1*
then obtain *y h1* **where** *hxy*: *x* = *y* [⊕] *h1* **and** *hy*: *y* ∈ *A* ∪ *?B′* **and**
hh1: *h1* ∈ *?H1* **using** *G.sumset.cases* **by** *blast*
then have *hyG*: *y* ∈ *G* **and** *hcG*: *c* ∈ *G* **and** *hh1G*: *h1* ∈ *G*
using *hAG hB′G G.stabilizer-subset-group hc* **by** *auto*
then have *y* = *a1* [⊕] (*c* [⊕] *G.inverse h1*) **using** *hxac hxy ha1A hAG*
by (*metis G.associative G.commutative G.composition-closed in-mono*
G.inverse-closed G.invertible G.invertible-left-inverse2)
moreover have *h1* ∈ *G.stabilizer C* **using** *hh1 hH1stabC* **by** *auto*
moreover hence *c* [⊕] *G.inverse h1* ∈ *G.stabilizer C* **using** *hc*
G.stabilizer-is-subgroup subgroup-def G.group-axioms
group.invertible subgroup.subgroup-inverse-iff
submonoid.sub-composition-closed hh1G **by** *metis*
ultimately have *y* ∈ *G.sumset* {*a1*} (*G.stabilizer C*) **using** *ha1G hcG hAG ha1A hh1G*
by *blast*
then have *y* ∈ *?A1* ∪ *?B1* **using** *hy* **by** (*simp add*: *ha1b1*)
thus *False* **using** *hx hxy hh1 hh1G hyG* **by** *auto*
qed
ultimately show *x* ∈ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) − *G.sumset* (*A* ∪ *?B′*) *?H1* **by** *simp*
qed
moreover have *finite* (*G.sumset* {*a1*} (*G.stabilizer C*) − *G.sumset* (*?A1* ∪ *?B1*) *?H1*)
using *finite-subset G.finite-sumset hstabC* **by** *simp*
moreover hence *finite* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) − *G.sumset* (*A* ∪ *?B′*) *?H1*)

using *finite-subset G.finite-sumset hstabC hA hB′ finite-UnI* **by** *simp*
moreover have *card* (*G.sumset* {*a1*} (*G.stabilizer C*) − *G.sumset* (*?A1* ∪ *?B1*) *?H1*) ≤
card (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) − *G.sumset* (*A* ∪ *?B′*) *?H1*)
using *card-mono hsub calculation*(*3*) **by** *auto*
ultimately show *?thesis* **using** *card-Diff-subset* **by** *linarith*
qed
also have ... = *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*)
using *card-Diff-subset*[*OF hfinsumH1 hsubsumH1*] *hsumH1card-le* **by** *linarith*
finally have *int* (*card* (*G.stabilizer C*)) − *card* (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*)
≤ *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) − *card* (*G.sumset* (*A* ∪ *?B′*) *?H1*)
using *hCG subset-iff G.card-sumset-singleton-subset-eq G.stabilizer-subset-group*
hAG ha1A **by** *auto*
thus *False* **using** *hA1B1ineq* **by** *linarith*
qed
have *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) −
card (*G.sumset* (*A* ∪ *?B′*) *?H1*) ≥ *0* **by** (*simp add*: *hsumH1card-le*)
then have *int* (*card* (*G.stabilizer C*)) −
card (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*) > *0* **using** *hA1B1ineq* **by** *linarith*
moreover have *int* (*card ?H1*) *dvd int* (*card* (*G.stabilizer C*)) −
card (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*) **using**
G.stabilizer-subset-stabilizer-dvd hH1stabC psubset-imp-subset int-dvd-int-iff dvd-diff
G.card-stabilizer-divide-sumset[*OF hA1G*] *G.card-stabilizer-divide-sumset*[*OF hB1G*]
by *fastforce*
ultimately have *int* (*card* (*G.stabilizer C*)) −
card (*G.sumset ?A1 ?H1*) − *card* (*G.sumset ?B1 ?H1*) ≥ *int* (*card ?H1*)
using *zdvd-imp-le* **by** *blast*
moreover have *hA1-le-sum*: *card ?A1* ≤ *card* (*G.sumset ?A1 ?H1*)
using *G.sumset-commute G.card-le-sumset G.zero-mem-stabilizer hA1G hA1 hH1stabC hstabC*
by (*metis finite-subset psubset-imp-subset G.unit-closed*)
moreover have *hB1-le-sum*: *card ?B1* ≤ *card* (*G.sumset ?B1 ?H1*)
using *G.sumset-commute G.card-le-sumset G.zero-mem-stabilizer hB1G hB1 hH1stabC hstabC*
by (*metis finite-subset psubset-imp-subset G.unit-closed*)

The above facts combined allow us to deduce the inequality that is referred to as inequality (2) in [1] for *A* ∩ *G.sumset* {*a1*} (*G.stabilizer C*), *G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*b1*} (*G.stabilizer C*) and

G.stabilizer (*G.sumset* ($A \cap$ *G.sumset* {*a1*} (*G.stabilizer* *C*)) (*G.sumset* (*G.differenceset* *B* {*b*}) {*a*} $\cap$ *G.sumset* {*b1*} (*G.stabilizer* *C*))).

ultimately have *21*: *int* (*card* (*G.stabilizer* *C*)) $\geq$ *int* (*card* *?H1*) + *card* *?A1* + *card* *?B1*
by *linarith*
have *int* (*card* (*G.sumset* ($A \cup$ *?B′*) (*G.stabilizer* *C*))) −
card (*G.sumset* ($A \cup$ *?B′*) *?H2*) $\geq$ *0* **by** (*simp add*: *hsumH2card-le*)
then have *int* (*card* (*G.stabilizer* *C*)) −
card (*G.sumset* *?A2* *?H2*) − *card* (*G.sumset* *?B2* *?H2*) > *0* **using** *hA2B2ineq* **by** *linarith*
moreover have *int* (*card* *?H2*) *dvd* *int* (*card* (*G.stabilizer* *C*)) −
card (*G.sumset* *?A2* *?H2*) − *card* (*G.sumset* *?B2* *?H2*) **using** *psubset-imp-subset*
G.stabilizer-subset-stabilizer-dvd *hH2stabC* *int-dvd-int-iff* *dvd-diff*
G.card-stabilizer-divide-sumset[*OF* *hA2G*] *G.card-stabilizer-divide-sumset*[*OF* *hB2G*]
by *fastforce*
ultimately have *int* (*card* (*G.stabilizer* *C*)) −
card (*G.sumset* *?A2* *?H2*) − *card* (*G.sumset* *?B2* *?H2*) $\geq$ *int* (*card* *?H2*)
using *zdvd-imp-le* **by** *blast*
moreover have *hA2-le-sum*: *card* *?A2* $\leq$ *card* (*G.sumset* *?A2* *?H2*)
using *G.sumset-commute* *G.card-le-sumset* *G.zero-mem-stabilizer* *G.stabilizer-subset-group*
hA2G *hA2* *hH2stabC* *hstabC* *psubset-imp-subset* **by** (*metis* *finite-subset* *G.unit-closed*)
moreover have *hB2-le-sum*: *card* *?B2* $\leq$ *card* (*G.sumset* *?B2* *?H2*)
using *G.sumset-commute* *G.card-le-sumset* *G.zero-mem-stabilizer* *G.stabilizer-subset-group*
hB2G *hB2* *hH2stabC* *hstabC* *psubset-imp-subset* **by** (*metis* *finite-subset* *G.unit-closed*)

Analogously, the above facts combined allow us to deduce the inequality that is referred to as inequality (2) in [1] for $A \cap$ *G.sumset* {*b1*} (*G.stabilizer* *C*), *G.sumset* (*G.differenceset* *B* {*b*}) {*a*} $\cap$ *G.sumset* {*a1*} (*G.stabilizer* *C*) and *G.stabilizer* (*G.sumset* ($A \cap$ *G.sumset* {*b1*} (*G.stabilizer* *C*)) (*G.sumset* (*G.differenceset* *B* {*b*}) {*a*} $\cap$ *G.sumset* {*a1*} (*G.stabilizer* *C*))).

ultimately have *22*: *int* (*card* (*G.stabilizer* *C*)) $\geq$ *int* (*card* *?H2*) + *card* *?A2* + *card* *?B2*
by *linarith*
let *?S* = *G.sumset* {*a1*} (*G.stabilizer* *C*) − (*?A1* $\cup$ *?B2*)
let *?T* = *G.sumset* {*b1*} (*G.stabilizer* *C*) − (*?A2* $\cup$ *?B1*)
have *hS* : *finite* *?S* **and** *hT*: *finite* *?T* **using** *G.finite-sumset* *hstabC* **by** *auto*
have *hST*: *?S* $\cap$ *?T* = {} **using** *ha1b1stabCne* *Diff-Int2* *Diff-Int-distrib2* *Int-Diff* *Int-Un-eq*(*4*)
Int-absorb *Int-commute* *Int-empty-right* *Int-insert-right* *Un-empty* *empty-Diff* *hA1ne*

hA2ne G.sumset-commute G.sumset-is-empty-iff G.sumset-stabilizer-eq-iff
by (*smt* (*verit*, *ccfv-threshold*) *G.sumset-assoc*)
have *hSTcard-le*: *card ?S* + *card ?T* + *card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]})
≤
card (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))
proof−
have *G.sumset* {*a1*} (*G.stabilizer C*) ⊆ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*) **and**
G.sumset {*b1*} (*G.stabilizer C*) ⊆ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*)
using *G.sumset-mono ha1A hb1B′ subset-refl empty-subsetI insert-subset* **by** *auto*
moreover have (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) ⊆ *G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*)
using *G.sumset-mono subset-refl empty-subsetI insert-subset G.zero-mem-stabilizer*
by *metis*
ultimately have *hsub*: *?S* ∪ *?T* ∪ (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) ⊆
G.sumset (*A* ∪ *?B′*) (*G.stabilizer C*) **by** *blast*
have *?S* ∩ *G.sumset* (*A* ∪ *?B′*) {[**0**]} = {} **by** *auto*
moreover have (*?S* ∪ *?T*) ∩ *G.sumset* (*A* ∪ *?B′*) {[**0**]} = {} **by** *auto*
ultimately have *card ?S* + *card ?T* + *card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) =
card (*?S* ∪ *?T* ∪ (*G.sumset* (*A* ∪ *?B′*) {[**0**]})) **using** *card-Un-disjoint hS hT*
G.finite-sumset finite-UnI hA hB′ hST **by** (*metis finite.emptyI finite.insertI*)
also have ... ≤ *card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*)) **using** *card-mono*
G.finite-sumset finite-UnI hA hB′ hstabC hsub **by** *metis*
finally show *?thesis* **by** *simp*
qed
have *hAB-not-conv*: ¬ *G.convergent* (*G.sumset A ?B′*) *A ?B′* **using** *hCmin hstab0*
G.convergent-set-def hcardstabC-gt-1 hstab0′ **by** *fastforce*
then have *card* (*G.sumset A ?B′*) + *card* {[**0**]} < *card* (*A* ∩ *?B′*) +
card (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) **using** *G.convergent-def hAne hB′ne hAG hB′G hstab0′*
subset-refl **by** *auto*
then have *hAB′sum*: *int* (*card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]})) + *card* (*A* ∩ *?B′*) −
card (*G.sumset A ?B′*) > *1* **by** *simp*
moreover have *int* (*card ?A1*) + *card ?B1* ≤ *int* (*card* (*G.sumset ?A1 ?H1*)) +
card (*G.sumset ?B1 ?H1*) **using** *hA1-le-sum hB1-le-sum* **by** *linarith*
moreover hence *int* (*card ?A1*) + *card ?B1* − *card ?H1* ≤ *card* (*G.sumset ?A1 ?B1*)
using *hindA1B1* **by** *linarith*
ultimately have *int* (*card ?S*) + *card ?T* + *card ?A1* + *card ?B1* −

card ?H1 <
int (*card ?S*) + *card ?T* + *card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) + *card* (*A* ∩ *?B′*) −
card (*G.sumset A ?B′*) + *card* (*G.sumset ?A1 ?B1*) **by** *linarith*
also have ... ≤ *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) + *card* (*A* ∩ *?B′*) − *card C*
proof−
have *G.sumset ?A1 ?B1* ∪ *C* ⊆ *G.sumset A ?B′* **using** *hC G.sumset-mono* **by** *auto*
then have *card* (*G.sumset ?A1 ?B1*) + *card C* ≤
card (*G.sumset A ?B′*) **using** *card-Un-disjoint hCfinite G.finite-sumset hA1 hB1 hA1B1Cempty*
card-mono hA hB′ **by** *metis*
then show *?thesis* **using** *hSTcard-le* **by** *linarith*
qed

From this, inequality (3) [1] follows for *A* ∩ *G.sumset* {*a1*} (*G.stabilizer C*), *G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*b1*} (*G.stabilizer C*) and *G.stabilizer* (*G.sumset* (*A* ∩ *G.sumset* {*a1*} (*G.stabilizer C*)) (*G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*b1*} (*G.stabilizer C*))).

finally have *31*: *int* (*card ?S*) + *card ?T* + *card ?A1* + *card ?B1* − *card ?H1* <
card (*G.stabilizer C*) **using** *hCcard* **by** *linarith*
have *int* (*card ?A2*) + *card ?B2* ≤ *int* (*card* (*G.sumset ?A2 ?H2*)) +
card (*G.sumset ?B2 ?H2*) **using** *hA2-le-sum hB2-le-sum* **by** *linarith*
moreover hence *int* (*card ?A2*) + *card ?B2* − *card ?H2* ≤ *card* (*G.sumset ?A2 ?B2*)
using *hindA2B2* **by** *linarith*
ultimately have *int* (*card ?S*) + *card ?T* + *card ?A2* + *card ?B2* − *card ?H2* <
int (*card ?S*) + *card ?T* + *card* (*G.sumset* (*A* ∪ *?B′*) {[**0**]}) + *card* (*A* ∩ *?B′*) −
card (*G.sumset A ?B′*) + *card* (*G.sumset ?A2 ?B2*) **using** *hAB′sum* **by** *linarith*
also have ... ≤ *int* (*card* (*G.sumset* (*A* ∪ *?B′*) (*G.stabilizer C*))) + *card* (*A* ∩ *?B′*) − *card C*
proof−
have *G.sumset ?A2 ?B2* ∪ *C* ⊆ *G.sumset A ?B′* **using** *hC G.sumset-mono* **by** *auto*
then have *card* (*G.sumset ?A2 ?B2*) + *card C* ≤
card (*G.sumset A ?B′*) **using** *card-Un-disjoint hCfinite G.finite-sumset hA2 hB2 hA1B1Cempty*
card-mono hA hB′ hB2A2Cempty G.sumset-commute **by** *metis*
then show *?thesis* **using** *hSTcard-le* **by** *linarith*
qed

From this, inequality (3) [1] follows for *A* ∩ *G.sumset* {*b1*} (*G.stabilizer C*), *G.sumset* (*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*a1*} (*G.stabilizer C*) and *G.stabilizer* (*G.sumset* (*A* ∩ *G.sumset* {*b1*} (*G.stabilizer C*)) (*G.sumset*

(*G.differenceset B* {*b*}) {*a*} ∩ *G.sumset* {*a1*} (*G.stabilizer C*))).

finally have *32*: *int* (*card ?S*) + *card ?T* + *card ?A2* + *card ?B2* − *card ?H2* <
card (*G.stabilizer C*) **using** *hCcard* **by** *linarith*

Adding together the four inequalities obtained by versions of inequalities (2) and (3) and dividing by 2 gives the following inequality:

have *4*: *2* ∗ *int* (*card* (*G.stabilizer C*)) > *int* (*card ?A1*) + *card ?B2* + *card ?S* + *card ?T* + *card ?B1* + *card ?A2*
using *21 22 31 32* **by** *linarith*
have *G.sumset* {*a1*} (*G.stabilizer C*) = (*?S* ∪ *?A1*) ∪ *?B2* **and**
hb1T: *G.sumset* {*b1*} (*G.stabilizer C*) = (*?T* ∪ *?A2*) ∪ *?B1* **by** *auto*
then have *card* (*G.stabilizer C*) ≤ *card ?S* + *card ?A1* + *card ?B2*
using *card-Un-le*[*of ?S ?A1*] *card-Un-le*[*of ?S* ∪ *?A1 ?B2*]
G.card-sumset-singleton-subset-eq G.stabilizer-subset-group ha1A hAG
by *auto*
moreover have *card* (*G.stabilizer C*) ≤ *card ?T* + *card ?A2* + *card ?B1*
using *hb1T card-Un-le*[*of ?T ?A2*] *card-Un-le*[*of ?T* ∪ *?A2 ?B1*]
G.card-sumset-singleton-subset-eq G.stabilizer-subset-group hb1B′ hB′G
by *auto*

Combining the inequality labelled *4*::′*b* above with the above facts, the claim follows:

ultimately show *?thesis* **using** *4* **by** *linarith*
qed

It remains to transfer the statement of inequality labelled *1*::′*b* into an analogous one, which replaces *G.sumset* (*G.differenceset B* {*b*}) {*a*} with *B*.

have *2*: *card* (*G.sumset A* (*G.stabilizer* (*G.sumset A B*))) =
card (*G.sumset A* (*G.stabilizer* (*G.sumset A ?B′*)))
using *hstabeq* **by** *auto*
have *3*: *card* (*G.sumset B* (*G.stabilizer* (*G.sumset A B*))) =
card (*G.sumset ?B′* (*G.stabilizer* (*G.sumset A ?B′*))) **using** *hstabeq hstab0′ G.sumset-commute*
by (*metis G.card-differenceset-singleton-mem-eq G.card-sumset-singleton-subset-eq*
G.sumset-D(*1*) *G.sumset-commute G.sumset-subset-carrier Int-absorb1 Int-commute hBG ha1G hbG*)
then show *?thesis* **using** *1 2 3 hstabeq hstab0′ htranslate* **by** *auto*
qed
next
case *hstabne0*: *False*
let *?K* = *G.stabilizer* (*G.sumset A B*)
have *hcardK-gt-1*: *card ?K* > *1* **using** *G.stabilizer-finite G.sumset-subset-carrier G.finite-sumset*
hA hB hstabne0 G.zero-mem-stabilizer **by** (*metis card-0-eq card-1-singletonE G.card-sumset-0-iff*

empty-iff hAG hAne hBG hBne insertE less-one linorder-neqE-nat)
interpret *K*: *subgroup-of-additive-abelian-group ?K G* ([⊕]) [**0**]
using *G.stabilizer-is-subgroup subgroup-of-additive-abelian-group-def*
by (*metis G.abelian-group-axioms G.group-axioms hGroupG subgroup-of-abelian-group-def*

subgroup-of-group.intro)
let *?φ* = *K.Class*
have *hφ*: ⋀ *a*. *a* ∈ *G* ⟹ *?φ a* = (λ *x*. *G.sumset* {*x*} *?K*) *a*
using *K.Left-Coset-Class-unit G.Left-Coset-eq-sumset-stabilizer* **by** *simp*
interpret *GK*: *additive-abelian-group G.Factor-Group G ?K K.quotient-composition K.Class* [**0**]
proof
fix *x y* **assume** *x* ∈ *K.Partition* **and** *y* ∈ *K.Partition*
then obtain *a b* **where** *x* = *K.Class a* **and** *y* = *K.Class b* **and** *a* ∈ *G* **and** *b* ∈ *G*
by (*meson K.representant-exists*)
then show *K.quotient-composition x y* = *K.quotient-composition y x*
using *K.Class-commutes-with-composition G.commutative* **by** *presburger*
qed
have *hGroupGK*: *additive-abelian-group* (*G.Factor-Group G ?K*) *K.quotient-composition* (*K.Class* [**0**]) **..**

Here, we specialize the induction hypothesis to the factor group.

let *?K-repr* = *K.φ* ‘ *K.Partition*
have ⋀ *x y*. *x* ∈ *?K-repr* ⟹ *y* ∈ *?K-repr* ⟹ *K.quot-comp-alt x y* = *K.quot-comp-alt y x*
using *K.quot-comp-alt-def G.commutative K.phi-image-subset subsetD* **by** (*metis* (*full-types*))
then interpret *K-repr*: *additive-abelian-group ?K-repr K.quot-comp-alt K.φ ?K*
using *Group-Theory.group.axioms*(*1*)[*OF K.phi-image-group*]
by (*auto simp add*: *additive-abelian-group-def abelian-group-def K.phi-image-group commutative-monoid-axioms-def commutative-monoid-def*)
have *hindrepr*: ⋀ *m C D*. *m* < *n* ⟶ *C* ⊆ *?K-repr* ⟶ *D* ⊆ *?K-repr* ⟶ *finite C* ⟶
finite D ⟶ *C* ≠ {} ⟶ *D* ≠ {} ⟶ *card* (*K-repr.sumset C D*) + *card C* = *m* ⟶
card (*K-repr.sumset C* (*K-repr.stabilizer* (*K-repr.sumset C D*))) +
card (*K-repr.sumset D* (*K-repr.stabilizer* (*K-repr.sumset C D*))) − *card* (*K-repr.stabilizer* (*K-repr.sumset C D*)) ≤
card (*K-repr.sumset C D*) **using** *hind K-repr.additive-abelian-group-axioms* **by** *blast*
have *hindfactor*: ⋀ *m C D*. *m* < *n* ⟶ *C* ⊆ *K.Partition* ⟶ *D* ⊆ *K.Partition* ⟶ *finite C* ⟶
finite D ⟶ *C* ≠ {} ⟶ *D* ≠ {} ⟶ *card* (*GK.sumset C D*) + *card C* = *m* ⟶
card (*GK.sumset C* (*GK.stabilizer* (*GK.sumset C D*))) +
card (*GK.sumset D* (*GK.stabilizer* (*GK.sumset C D*))) − *card* (*GK.stabilizer* (*GK.sumset C D*)) ≤

card (*GK.sumset C D*)
proof(*intro impI*)
fix *m C D* **assume** *hmn*: $m < n$ **and** *hCK*: $C \subseteq K.Partition$ **and** *hDK*: $D \subseteq K.Partition$ **and**
hC: *finite C* **and** *hD*: *finite D* **and** *hCne*: $C \neq \{\}$ **and** *hDne*: $D \neq \{\}$ **and**

hCDcard: *card* (*GK.sumset C D*) + *card C* = *m*
let *?C* = *K.φ ‘ C* **and** *?D* = *K.φ ‘ D*
have *hCrepr*: *?C* ⊆ *?K-repr* **and** *hDrepr*: *?D* ⊆ *?K-repr* **using** *hCK hDK* **by** *auto*
have *hCfin*: *finite ?C* **and** *hDfin*: *finite ?D* **and** *hCne-1*: *?C* ≠ {} **and** *hDne-1*: *?D* ≠ {} **using** *hC hD hCne hDne* **by** *auto*
have *hcardC*: *card ?C* = *card C* **using** *K.phi-inj-on hC card-image inj-on-subset hCK* **by** *metis*
have *card* (*GK.sumset C D*) = *card* (*K-repr.sumset ?C ?D*)
using *card-image K.phi-inj-on inj-on-subset K.phi-image-sumset-eq GK.sumset-subset-carrier hCK hDK* **by** (*smt* (*verit*, *best*))
then have *card* (*K-repr.sumset ?C ?D*) + *card ?C* = *m* **using** *hCDcard hcardC* **by** *presburger*
then have *card* (*K-repr.sumset ?C* (*K-repr.stabilizer* (*K-repr.sumset ?C ?D*))) +
card (*K-repr.sumset ?D* (*K-repr.stabilizer* (*K-repr.sumset ?C ?D*))) − *card* (*K-repr.stabilizer* (*K-repr.sumset ?C ?D*)) ≤
card (*K-repr.sumset ?C ?D*)
using *hindrepr hCfin hDfin hCne-1 hDne-1 hCrepr hDrepr hmn* **by** *blast*
then show *card* (*GK.sumset C* (*GK.stabilizer* (*GK.sumset C D*))) +
card (*GK.sumset D* (*GK.stabilizer* (*GK.sumset C D*))) − *card* (*GK.stabilizer* (*GK.sumset C D*)) ≤
card (*GK.sumset C D*) **using** *K.phi-image-sumset-eq K.phi-image-stabilizer-eq K.phi-inj-on inj-on-subset hCK hDK card-image*
by (*smt* (*z3*) *GK.stabilizer-subset-group GK.sumset-subset-carrier*)
qed
have *hstab0*: *GK.stabilizer* (*?φ ‘* (*G.sumset A B*)) = {*K.Class* [**0**]}
proof
show *GK.stabilizer* (*?φ ‘ G.sumset A B*) ⊆ {*K.Class* [**0**]}
proof
fix *x* **assume** *hx*: *x* ∈ *GK.stabilizer* (*?φ ‘ G.sumset A B*)
moreover have *?φ ‘ G.sumset A B* ⊆ *K.Partition*
using *K.natural.map-closed G.sumset-subset-carrier* **by** *blast*
ultimately have *hsum*: *GK.sumset* {*x*} (*?φ ‘ G.sumset A B*) = *?φ ‘ G.sumset A B*
using *GK.stabilizer-def* **by** *auto*
obtain *x′* **where** *hxφ*: *x* = *?φ x′* **and** *hx′G*: *x′* ∈ *G*
using *hx GK.stabilizer-subset-group K.representant-exists* **by** *force*
have *hsumset*: *GK.sumset* {*x*} (*?φ ‘ G.sumset A B*) = (λ *a*. *?φ* (*x′* [⊕] *a*)) ‘ *G.sumset A B*
proof
show *GK.sumset* {*x*} (*?φ ‘ G.sumset A B*) ⊆ (λ*a*. *?φ* (*x′* [⊕] *a*)) ‘ *G.sumset A B*

proof
fix *y* **assume** $y \in GK.sumset\ \{x\}\ (?\varphi\ `\ G.sumset\ A\ B)$
then obtain *z* **where** $z \in G.sumset\ A\ B$ **and** $y = K.quotient\text{-}composition$ $(?\varphi\ x')\ (?\varphi\ z)$
using *GK.sumset.cases hxφ* **by** *blast*
then show $y \in (\lambda a.\ ?\varphi\ (x'\ [\oplus]\ a))\ `\ G.sumset\ A\ B$
using *K.Class-commutes-with-composition G.composition-closed hx′G G.sumset.cases imageI* **by** *metis*
qed
next
show $(\lambda a.\ ?\varphi\ (x'\ [\oplus]\ a))\ `\ G.sumset\ A\ B \subseteq GK.sumset\ \{x\}\ (?\varphi\ `\ G.sumset\ A\ B)$
proof
fix *y* **assume** $y \in (\lambda a.\ ?\varphi\ (x'\ [\oplus]\ a))\ `\ G.sumset\ A\ B$
then obtain *z* **where** *hz*: $z \in G.sumset\ A\ B$ **and** $y = ?\varphi\ (x'\ [\oplus]\ z)$ **by** *blast*
then have $y = K.quotient\text{-}composition\ (?\varphi\ x')\ (?\varphi\ z)$ **using** *K.Class-commutes-with-composition G.composition-closed hx′G G.sumset.cases* **by** *metis*
then show $y \in GK.sumset\ \{x\}\ (?\varphi\ `\ G.sumset\ A\ B)$
using *hxφ hz imageI hx GK.sumset.sumsetI K.natural.map-closed*
by (*metis G.composition-closed hx′G insertCI G.sumset.cases*)
qed
qed
have $G.sumset\ \{x'\}\ (G.sumset\ A\ B) \subseteq G.sumset\ (G.sumset\ A\ B)\ ?K$
proof
fix *y* **assume** $y \in G.sumset\ \{x'\}\ (G.sumset\ A\ B)$
then obtain *z* **where** *hz*: $z \in G.sumset\ A\ B$ **and** *hy*: $y = x'\ [\oplus]\ z$ **using** *G.sumset.cases* **by** *blast*
then have $?\varphi\ (x'\ [\oplus]\ z) \in ?\varphi\ `\ G.sumset\ A\ B$ **using** *hsum hsumset* **by** *blast*
then obtain *w* **where** *hw*: $\{w\} \subseteq G.sumset\ A\ B$ **and** $w \in G.sumset\ A\ B$
and $?\varphi\ (x'\ [\oplus]\ z) = ?\varphi\ w$ **by** *auto*
then have $(x'\ [\oplus]\ z) \in G.differenceset\ (G.sumset\ \{w\}\ ?K)\ ?K$
using *hφ G.sumset-subset-carrier hx′G hz G.sumset-eq-subset-differenceset G.composition-closed G.stabilizer-is-nonempty G.stabilizer-subset-group G.sumset.cases K.Class-self G.differenceset-stabilizer-eq G.sumset-assoc* **by** *metis*
moreover have $G.differenceset\ (G.sumset\ \{w\}\ ?K)\ ?K \subseteq G.sumset\ \{w\}\ ?K$
using *hw* **by** (*simp add*: *G.differenceset-stabilizer-eq G.sumset-assoc*)
ultimately show $y \in G.sumset\ (G.sumset\ A\ B)\ ?K$ **using** *hy hw G.sumset-mono subsetD subset-refl* **by** *blast*
qed
moreover have $G.sumset\ (G.sumset\ A\ B)\ ?K = G.sumset\ A\ B$

using *G.sumset-commute G.sumset-stabilizer-eq-self G.sumset-subset-carrier*
by *auto*
ultimately have *G.sumset* $\{x'\}$ (*G.sumset A B*) = *G.sumset A B*
by (*metis G.finite-sumset G.sumset-subset-carrier card-subset-eq*
G.card-sumset-singleton-subset-eq hA hB hx'G)
then have $x' \in$ *?K* **using** *hx'G* **by** (*meson empty-subsetI G.finite-sumset*
hA hB insert-subset
G.sumset-eq-sub-stabilizer G.sumset-subset-carrier)
then show $x \in \{$*K.Class* [**0**]$\}$ **using** *hxφ*
by (*metis K.Block-self K.Normal-def K.quotient.unit-closed insertCI*)
qed
next
show $\{$*K.Class* [**0**]$\} \subseteq$ *GK.stabilizer* (*K.Class* ' *G.sumset A B*)
using *GK.zero-mem-stabilizer* **by** *auto*
qed
interpret *group-epimorphism ?φ G* ([⊕]) [**0**] *G.Factor-Group G ?K*
K.quotient-composition K.Class [**0**] **..**
interpret *GKN*: *normal-subgroup-in-kernel K.Class G* ([⊕]) [**0**] *G.Factor-Group*
G ?K
K.quotient-composition K.Class [**0**] *?K*
proof
show *?K* ⊆ *Ker* **using** *K.Block-self K.Normal-def K.quotient.unit-closed*
by *blast*
qed
have *hsumK*: *card* (*G.sumset A B*) = *card ?K* ∗ *card* (*?φ* ' (*G.sumset A*
B))
using *G.finite-sumset hA hB G.sumset-subset-carrier G.Union-stabilizer-Class-eq*

G.sumset-subset-carrier K.Union-Coset-card-eq **by** *simp*
have *hGKsumset*: *GK.sumset* (*?φ* ' *A*) (*?φ* ' *B*) = *?φ* ' (*G.sumset A B*)
proof
show *GK.sumset* (*?φ* ' *A*) (*?φ* ' *B*) ⊆ *?φ* ' *G.sumset A B*
proof
fix *x* **assume** $x \in$ *GK.sumset* (*?φ* ' *A*) (*?φ* ' *B*)
then obtain *a b* **where** *ha*: $a \in A$ **and** *hb*: $b \in B$ **and**
$x =$ *K.quotient-composition* (*?φ a*) (*?φ b*) **using** *GK.sumset.cases* **by**
blast
then have $x =$ *?φ* (*a* [⊕] *b*) **by** (*meson K.Class-commutes-with-composition*
hAG hBG in-mono)
then show $x \in$ *?φ* ' *G.sumset A B* **using** *ha hb hAG hBG* **by** *blast*
qed
next
show *?φ* ' *G.sumset A B* ⊆ *GK.sumset* (*?φ* ' *A*) (*?φ* ' *B*)
proof
fix *x* **assume** $x \in$ *?φ* ' *G.sumset A B*
then obtain *c* **where** $c \in$ *G.sumset A B* **and** $x =$ *?φ c* **by** *blast*
then obtain *a b* **where** $a \in A$ **and** $b \in B$ **and** $x =$ *?φ* (*a* [⊕] *b*)
using *G.sumset.cases* **by** *metis*
then show $x \in$ *GK.sumset* (*?φ* ' *A*) (*?φ* ' *B*) **using** *GK.sumset.cases*

K.Class-commutes-with-composition hAG hBG in-mono
by (*smt* (*verit*, *best*) *GK.sumset.simps K.natural.map-closed imageI*)
qed
qed
have *hAK*: *card* (*G.sumset A ?K*) = *card ?K* ∗ *card* (*?φ ‘ A*) **using** *hAG K.Union-Coset-card-eq hA*
G.sumset-stabilizer-eq-Class-Union G.Class-image-sumset-stabilizer-eq
by (*smt* (*verit*, *ccfv-threshold*) *card-0-eq G.card-sumset-0-iff G.finite-sumset hAne hB hBG hBne*
G.stabilizer-finite G.sumset-subset-carrier)
have *hBK*: *card* (*G.sumset B ?K*) = *card ?K* ∗ *card* (*?φ ‘ B*) **using** *hBG K.Union-Coset-card-eq hB*
G.sumset-stabilizer-eq-Class-Union G.Class-image-sumset-stabilizer-eq
by (*smt* (*verit*, *ccfv-SIG*) *card-0-eq G.card-sumset-0-iff G.finite-sumset hA hAG hAne hBne*
G.stabilizer-finite G.sumset-subset-carrier)
have *card* (*?φ ‘ A*) ≤ *card A* **by** (*simp add*: *card-image-le hA*)
moreover have *card* (*?φ ‘* (*G.sumset A B*)) < *card* (*G.sumset A B*) **using** *hsumK hcardK-gt-1*
G.card-sumset-0-iff hA hB hAne hBne **by** (*metis card-eq-0-iff card-image-le hAG hBG*
le-neq-implies-less less-not-refl3 mult-cancel2 nat-mult-1)
ultimately have *card* (*GK.sumset* (*?φ ‘ A*) (*?φ ‘ B*)) + *card* (*?φ ‘ A*) < *card* (*G.sumset A B*) + *card A*
using *hGKsumset* **by** *auto*
then obtain *m* **where** *m* < *n* **and** *card* (*GK.sumset* (*?φ ‘ A*) (*?φ ‘ B*)) + *card* (*?φ ‘ A*) = *m*
using *hcardsum* **by** *blast*
moreover have *hφAsub*: *?φ ‘ A* ⊆ *G.Factor-Group G ?K*
proof
fix *x* **assume** *x* ∈ *?φ ‘ A* **then obtain** *a* **where** *a* ∈ *G* **and** *?φ a* = *x* **using** *hAG* **by** *blast*
then show *x* ∈ *G.Factor-Group G ?K* **by** *blast*
qed
moreover have *hφBsub*: *?φ ‘ B* ⊆ *G.Factor-Group G ?K*
proof
fix *x* **assume** *x* ∈ *?φ ‘ B* **then obtain** *b* **where** *b* ∈ *G* **and** *?φ b* = *x* **using** *hBG* **by** *blast*
then show *x* ∈ *G.Factor-Group G ?K* **by** *blast*
qed
moreover have *hφA*: *finite* (*?φ ‘ A*) **and** *hφB*: *finite* (*?φ ‘ B*) **and** *hφAne*: *?φ ‘ A* ≠ {} **and**
hφBne: *?φ ‘ B* ≠ {} **using** *hA hB hAne hBne* **by** *auto*
moreover have *additive-abelian-group* (*G.Factor-Group G ?K*) *K.quotient-composition*
(*K.Class* [**0**]) **..**
moreover have *GK.stabilizer* (*GK.sumset* (*?φ ‘ A*) (*?φ ‘ B*)) = {*K.Class* [**0**]}
using *hstab0 hGKsumset* **by** *auto*

ultimately have *hindφ*: *card* (*GK.sumset* (*?φ* ‘ *A*) (*?φ* ‘ *B*)) ≥
card (*GK.sumset* (*?φ* ‘ *A*) {*K.Class* [**0**]}) + *card* (*GK.sumset* (*?φ* ‘ *B*) {*K.Class* [**0**]}) − *1*
using *hindfactor*[*of m ?φ* ‘ *A ?φ* ‘ *B*] **by** *simp*
have *hφsumA*: *GK.sumset* (*?φ* ‘ *A*) {*K.Class* [**0**]} = *?φ* ‘ *A*
by (*simp add*: *Int-absorb1 Int-commute hφAsub*)
have *hφsumB*: *GK.sumset* (*?φ* ‘ *B*) {*K.Class* [**0**]} = *?φ* ‘ *B*
by (*simp add*: *Int-absorb1 Int-commute hφBsub*)
have *card* (*G.sumset A ?K*) + *card* (*G.sumset B ?K*) − *card ?K* =
card ?K ∗ (*card* (*?φ* ‘ *A*) + *card* (*?φ* ‘ *B*) − *1*)
using *hAK hBK add-mult-distrib2 diff-mult-distrib2 nat-mult-1-right* **by** *presburger*
also have ... ≤ *card ?K* ∗ *card* (*GK.sumset* (*?φ* ‘ *A*) (*?φ* ‘ *B*))
using *hindφ hφsumA hφsumB* **by** *simp*
finally show *?thesis* **by** (*simp add*: *hGKsumset hsumK*)
qed
qed
qed
thus *?thesis* **using** *assms hAne hBne additive-abelian-group-axioms* **by** *blast*
qed

2.2 Strict version of Kneser's Theorem

We show a strict version of Kneser's Theorem as presented in Theorem 3.2 of [3].

theorem *Kneser-strict-aux*: **fixes** *A* **and** *B* **assumes** *hAG*: *A* ⊆ *G* **and** *hBG*: *B* ⊆ *G* **and** *hA*: *finite A*
and *hB*: *finite B* **and** *hAne*: *A* ≠ {} **and** *hBne*: *B* ≠ {} **and**
hineq: *card* (*sumset A B*) > *card* (*sumset A* (*stabilizer* (*sumset A B*))) +
card (*sumset B* (*stabilizer* (*sumset A B*))) − *card* (*stabilizer* (*sumset A B*))
shows *card* (*sumset A B*) ≥ *card A* + *card B*

proof−
let *?H* = *stabilizer* (*sumset A B*)
have *hfin*: *finite ?H* **using** *stabilizer-subset-group stabilizer-finite sumset-subset-carrier*
finite-sumset assms sumset-is-empty-iff sumset-stabilizer-eq-self **by** *metis*
have *card ?H dvd card* (*sumset A B*) **and** *card ?H dvd* (*card* (*sumset A* (*stabilizer* (*sumset A B*))) +
card (*sumset B* (*stabilizer* (*sumset A B*))) − *card ?H*)
using *card-stabilizer-divide-sumset hAG hBG card-stabilizer-sumset-divide-sumset* **by** *auto*
then have *card* (*sumset A B*) ≥ *card* (*sumset A ?H*) + *card* (*sumset B ?H*) **using** *hineq*
by (*metis diff-le-mono2 dvd-add-right-iff dvd-imp-le le-diff-conv less-imp-add-positive*)
moreover have *card A* + *card B* ≤ *card* (*sumset A ?H*) + *card* (*sumset B ?H*)
using *card-le-sumset sumset-commute assms stabilizer-subset-group stabilizer-is-nonempty*
Int-emptyI inf.orderE add-mono hfin **by** *metis*

ultimately show *?thesis* **by** *linarith*
qed

theorem *Kneser-strict*: **fixes** *A* **and** *B* **assumes** $A \subseteq G$ **and** $B \subseteq G$ **and** *finite A* **and** *finite B*
and *stabilizer* (*sumset A B*) = *H* **and** $A \neq \{\}$ **and** $B \neq \{\}$ **and** *card* (*sumset A B*) < *card A* + *card B*
shows *card* (*sumset A B*) = *card* (*sumset A* (*stabilizer* (*sumset A B*))) +
card (*sumset B* (*stabilizer* (*sumset A B*))) − *card* (*stabilizer* (*sumset A B*))
using *Kneser Kneser-strict-aux assms le-antisym nat-less-le* **by** *metis*

2.3 The CauchyDavenport Theorem

We show the CauchyDavenport Theorem as a corollary of Kneser's Theorem, following a comment on Theorem 3.2 in [3].

interpretation *Z-p*: *additive-abelian-group* {*0..int* ((*p* :: *nat*)−*1*)} (λ *x y*. ((*x* + *y*) *mod int p*)) *0*::*int*
using *additive-abelian-group-def residue-group*[*of p*] **by** *fastforce*

theorem *Cauchy-Davenport*:
fixes *p* :: *nat*
assumes *prime p* **and** $A \neq \{\}$ **and** $B \neq \{\}$ **and** *finite A* **and** *finite B* **and**
$A \subseteq \{0..p-1\}$ **and** $B \subseteq \{0..p-1\}$
shows *card* (*Z-p.sumset p A B*) $\geq$ *Min* {*p*, *card A* + *card B* −*1*}

proof(*cases Z-p.stabilizer p* (*Z-p.sumset p A B*) = {*0*})
case *True*
moreover have *Z-p.sumset p A* {*0*} = *A* **and** *Z-p.sumset p B* {*0*} = *B* **using** *assms Z-p.sumset-D*(*1*) **by** *auto*
ultimately show *?thesis* **using** *Z-p.Kneser*[*of A p B*] *assms* **by** *fastforce*
next
case *hne*: *False*
let *?H* = *Z-p.stabilizer p* (*Z-p.sumset p A B*)
have *?H* = {*0..int*(*p*−*1*)} **using** *hne Z-p.stabilizer-is-subgroup*[*of p Z-p.sumset p A B*]
residue-group-simple[*OF assms*(*1*)] **by** *blast*
moreover have $p \geq 2$ **using** *assms*(*1*) **by** (*simp add*: *prime-ge-2-nat*)
ultimately have *card ?H* = *p* **using** *card-atLeastAtMost* **by** (*simp add*: *of-nat-diff*)
then have $p \leq$ *card* (*Z-p.sumset p A B*)
using *Z-p.card-stabilizer-le card-0-eq assms Z-p.card-sumset-0-iff Z-p.sumset.cases*

Z-p.sumset-subset-carrier Z-p.finite-sumset **by** *metis*
then show *?thesis* **by** *auto*
qed

end
end

References

[1] M. DeVos. A short proof of kneser's addition theorem for abelian groups. In M. B. Nathanson, editor, *Combinatorial and Additive Number Theory*, pages 39–41, New York, NY, 2014. Springer New York.

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

[3] I. Z. Ruzsa. Sumsets and structure, 2008. Course notes, available on https://www.math.cmu.edu/users/af1p/Teaching/AdditiveCombinatorics/Additive-Combinatorics.pdf.