UNIVERSITY OF CAMBRIDGE

Formalisation of Kneser's Theorem in Lean and Isabelle/HOL

Mantas Bakšys | Yaël Dillies | Angeliki Koutsoukou-Argyraki

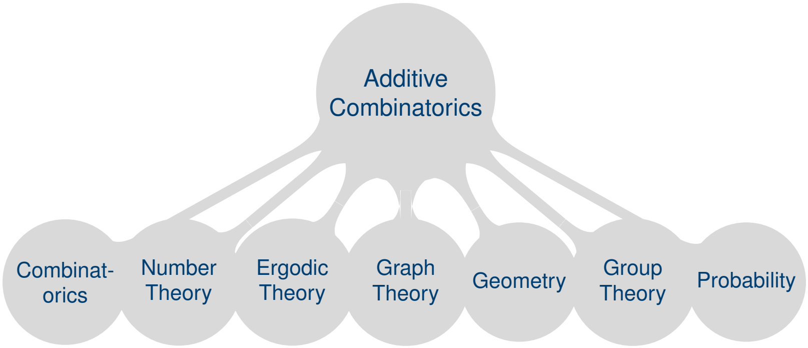
mb2412@cam.ac.uk | yyd23@cam.ac.uk | ak2110@cam.ac.uk

University of Cambridge, UK

4 February 2024

Additive Combinatorics

Additive combinatorics is, at heart, the study of combinatorial questions involving *the additive structure of sets*



Preliminary Definitions

Given an additive abelian group G and finite subsets A and B we define:

- **Sumset**: $A + B = \{a + b | a \in A, b \in B\}$.
- **Difference Set**: $A - B = \{a - b | a \in A, b \in B\}$.
- **Stabilizer**: $\mathcal{S}(A) = \{g \in G \, | g + A = A\}$.

Simple and broad concepts lead to many questions

E.g. What are the bounds on the cardinality of sumsets?

How close are sumsets to forming subgroups?

$\cdots$

Kneser's Theorem

Theorem (Cauchy-Davenport)

Let p be a prime and $A, B \subseteq \mathbb{Z}/p\mathbb{Z}$ be non-empty subsets, then

$$|A + B| \geq \min\{p, |A| + |B| - 1\}$$

A natural generalisation of the Cauchy-Davenport theorem for arbitrary abelian groups is a theorem of Kneser:

Theorem (Kneser)

Let G be an abelian group with finite non-empty subsets $A, B \subseteq G$ and $K = \mathcal{S}(A + B)$, then

$$|A + B| \geq |A + K| + |B + K| - |K|$$

Cauchy-Davenport from Kneser

Theorem

Kneser's theorem implies Cauchy-Davenport.

Proof.

$\mathbb{Z}/p\mathbb{Z}$ has prime order, so $K = \mathcal{S}(A + B)$ is either

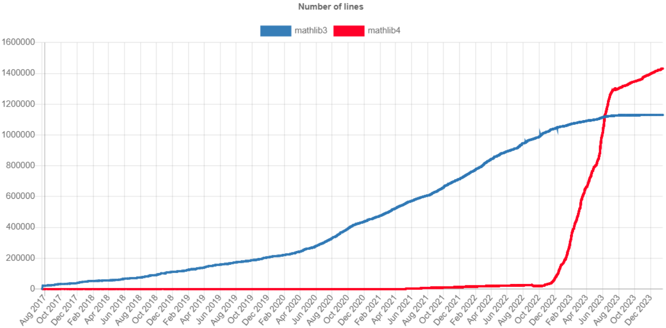
- $\mathbb{Z}/p\mathbb{Z}$ and $A + B = \mathbb{Z}/p\mathbb{Z}$
- $\{0\}$ and Kneser tells us

$$|A + B| \geq |A + K| + |B + K| - |K| = |A| + |B| - 1$$

□

Lean

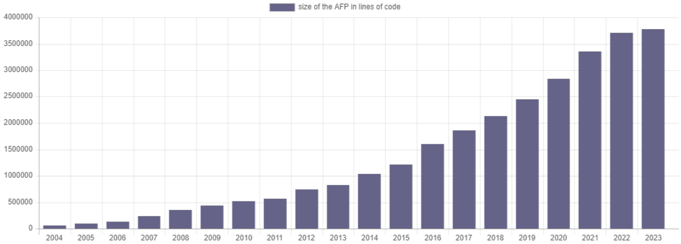
- Lean is an interactive theorem prover based on a version of Dependent Type Theory called Calculus of Inductive Constructions
- A non-trivial proportion of the modern literature formalised in Mathlib, the mathematics library



THEOREM PROVER

Isabelle/HOL and the Archive of Formal Proofs

- Isabelle/HOL is a modern interactive theorem prover based on Simple Type Theory
- Features a strong automation suite with Sledgehammer and human-readable proofs with Isar
- Many substantial theorems formalised in the fast-growing Isabelle Archive of Formal Proofs (AFP) library



Kneser's Theorem: A Blueprint

Kneser's theorem

$|A+B| \geq |A+K| + |B+K| - |K|$, where $K = \mathcal{S}(A+B)$

Induct on $|A+B| + |A|$

Is $\mathcal{S}(A+B) = \{0\}$?

No!
Apply induction hyp. in
$G \big/ \mathcal{S}(A+B)$

Yes!

Define **convergent** sets $C \subseteq A+B$

$|C| + |\mathcal{S}(C)| \geq |A \cap B| + |(A \cup B) + \mathcal{S}(C)|$

Find the **convergent** C with $|\mathcal{S}(C)|$ **minimal**

Complete the proof via pointwise-set manipulations

Type Universes

DeVos' proof of Kneser's theorem runs induction on the quantity

$$|A + B| + |A|$$

Induction hypothesis is applied to the quotient group $G/\mathcal{S}(A+B)$

$\implies$ Non-trivial argument to formalise, which requires the induction argument to **quantify** over all abelian groups.

Type Universes

DeVos' proof of Kneser's theorem runs induction on the quantity

$$|A + B| + |A|$$

Induction hypothesis is applied to the quotient group ${}^{G}\!/_{\mathcal{S}(A+B)}$

$\implies$ Non-trivial argument to formalise, which requires the induction argument to **quantify** over all abelian groups.

$\implies$ We ought to find a type β such that for any type α, which can be made into an abelian group:

$$\alpha : \beta$$

Type Universes

DeVos' proof of Kneser's theorem runs induction on the quantity

$$|A + B| + |A|$$

Induction hypothesis is applied to the quotient group ${}^{G}\!/\!_{\mathcal{S}(A+B)}$

$\implies$ Non-trivial argument to formalise, which requires the induction argument to **quantify** over all abelian groups.

$\implies$ We ought to find a type β such that for any type α, which can be made into an abelian group:

$$\alpha : \beta$$

$\implies$ Can we do this in each system?

Type Universes - Isabelle

The Type Theory of Isabelle/HOL (STP) does not support Type Universes.

Type Universes - Isabelle

The Type Theory of Isabelle/HOL (STP) does not support Type Universes.

$\Longrightarrow$ We cannot quantify over types

(there is no type which contains all *abelian groups* as terms)

Type Universes - Isabelle

The Type Theory of Isabelle/HOL (STP) does not support Type Universes.

$\Longrightarrow$ We cannot quantify over types

(there is no type which contains all *abelian groups* as terms)

$\Longrightarrow$ Is there a workaround?

Type Universes - Isabelle

The Type Theory of Isabelle/HOL (STP) does not support Type Universes.

$\implies$ We cannot quantify over types

(there is no type which contains all *abelian groups* as terms)

$\implies$ Is there a workaround? Yes!

In this case, just re-embed the quotient group ${}^{G}\!/_{\mathcal{S}(A+B)}$ into G by taking coset representatives.

Workaround for Isabelle/HOL

Preliminary definitions

```
definition ϕ :: 'a set ⟹ 'a where
  ϕ = ( x. if x ∈ G // K then
      (SOME a. a  G  x = a ·| K) else undefined)

definition quot-comp-alt :: 'a ⟹ 'a ⟹ 'a where
  quot-comp-alt a b = ϕ ((a · b) ·| K)
```

Excerpt from Kneser's proof:

```
let ?ϕ = K.Class
let ?K-repr = K.ϕ ‘ K.Partition
then interpret K-repr: additive-abelian-group ?K-repr
  K.quot-comp-alt K.ϕ ?K by <proof>
```

Type Universes - Lean

Lean's Type Theory contains ω-many Universe levels

Type Universes - Lean

Lean's Type Theory contains ω-many Universe levels
They are denoted by Type 0, Type 1, Type 2, ...
Each type α in Lean has a **universe level** $u \in \mathbb{N}$ such that

$$\alpha : \text{Type } u$$

Type Universes - Lean

Lean's Type Theory contains ω-many Universe levels
They are denoted by Type 0, Type 1, Type 2, ...
Each type α in Lean has a **universe level** $u \in \mathbb{N}$ such that

$$\alpha : \text{Type } u$$

Lean also implements universe polymorphism, which means that u above may be taken as a *variable*.

Type Universes - Lean

Lean's Type Theory contains ω-many Universe levels
They are denoted by Type 0, Type 1, Type 2, ...
Each type α in Lean has a **universe level** $u \in \mathbb{N}$ such that

$$\alpha : \text{Type } u$$

Lean also implements universe polymorphism, which means that u above may be taken as a *variable*.

Induction argument in Lean code

```
induction' n using Nat.strong_induction_on with n ih
  generalizing G
```

Stabilizers - different definitions

On pen-and-paper:

$$\mathcal{S}(A) = \{g \in G | g + A = A\}$$

In Isabelle:

```
definition stabilizer:: 'a set ⟹ 'a set where
  stabilizer S ≡ {x ∈ G. sumset {x} (S ∩ G) = S ∩ G}
```

In Lean:

```
def mulStab (s : Finset G) : Finset G :=
    (s / s).filter fun a => a • s = s
```

Stabilizers - different definitions

- `Finset` in Lean vs `Set` in Isabelle
- Use of `filter` and s/s in Lean. Why?
- What is the stabilizer of $\emptyset$?

Stabilizers - different definitions

- `Finset` in Lean vs `Set` in Isabelle
- Use of `filter` and s/s in Lean. Why?
- What is the stabilizer of $\emptyset$? Depends on the system!
- What could we have done differently?

Handling algebraic set expressions - Motivation

Additive combinatorics uses identities of the form:

- $A + B = B + A$
- $-(-A) = A$
- $-(A - B) = B - A$
- $A - (B - C) = A + C - B$
- $(A - B) + (C - D) = (A + C) - (B + D)$
- $2(A - 3B) + 3(B - 2C) = 2(A - 3C) + 3(2B - B)$,

where A, B, C, D are sets in an abelian group.

Handling algebraic set expressions - Problem

Easily derivable from $\mathrm{AddGroup}$ lemmas. But $\mathrm{Finset}\ G$ is not a group even if G is.

$$\mathrm{AddGroup}\ G \not\Rightarrow \mathrm{AddGroup}\ (\mathrm{Finset}\ G)$$

Handling algebraic set expressions - Problem

Easily derivable from $\mathrm{AddGroup}$ lemmas. But $\mathrm{Finset}\ G$ is not a group even if G is.

$$\mathrm{AddGroup}\ G \not\Rightarrow \mathrm{AddGroup}\,(\mathrm{Finset}\ G)$$

Isabelle solution: Extensionality every time + automation bash
Lean solution: Generalise relevant lemmas to something weaker than $\mathrm{AddGroup}$ that $\mathrm{Finset}\ G$ respects

Handling algebraic set expressions - Idea

The $\mathrm{AddGroup}$ identities that hold for $\mathrm{Finset}\ G$ are exactly the ones where **each variable (sign included) appears the same number of times on both sides**.

$$A \neq -A$$

$$A - A \neq 0$$

$$A(B + C) \neq AB + AC$$

Homework: Check this is the case for the identities two slides ago.

Handling algebraic set expressions - Idea

Addition identities are already covered by Monoid. So look at the most basic identities involving negation and subtraction:

$$A - B = A + (-B)$$
$$-(-A) = A$$
$$-(A + B) = (-B) + (-A)$$

This is enough to get all lemmas we care about on $\mathrm{Finset}\ G$!

Handling algebraic set expressions - Definition

```
class SubtractionMonoid (G : Type u)
    extends AddMonoid G, Neg G, Sub G where
  sub_eq_add_neg (a b : G) : a - b = a + -b
  neg_neg (a : G) : -(-a) = a
  neg_add_rev (a b : G) : -(a + b) = -b + -a
```

$$\text{SubtractionMonoid } G \implies \text{SubtractionMonoid } (\text{Finset } G)$$

Handling algebraic set expressions - Bonus

Mathlib used to prove lemmas like

$$\left(\frac{a}{b}\right)^{-1} = \frac{b}{a}$$
$$\frac{a}{\frac{b}{c}} = \frac{ac}{b}$$
$$\frac{a}{b}\frac{c}{d} = \frac{ac}{bd}$$

separately for Group and $\mathrm{GroupWithZero}$. $\mathrm{DivisionMonoid}$ unifies both versions!

Handling algebraic set expressions - Bonus

Mathlib used to prove lemmas like

$$\left(\frac{a}{b}\right)^{-1} = \frac{b}{a}$$
$$\frac{a}{\frac{b}{c}} = \frac{ac}{b}$$
$$\frac{a}{b}\frac{c}{d} = \frac{ac}{bd}$$

separately for Group and $\mathrm{GroupWithZero}$. $\mathrm{DivisionMonoid}$ unifies both versions!
This extra axiom lets us unify even more lemmas:

$$AB = 1 \implies A^{-1} = B$$

Concluding remarks

Kneser's theorem	Paper	Lean	Isabelle
.zip size (bytes)	2 829	7 236	10 611
De Bruijn factor	1	2.56	3.75

Additive Combinatorics is an area suitable in any modern proof assistant!

Acknowledgements and Contacts

Mantas[1]
mb2412@cam.ac.uk

Yaël[2]
yyd23@cam.ac.uk

Angeliki[3]
ak2110@cam.ac.uk

Source code:

- Isabelle AFP Entry: `https://www.isa-afp.org/entries/Kneser_Cauchy_Davenport.html`
- Lean formalisation: `https://yaeldillies.github.io/LeanCamCombi/docs/LeanCamCombi/Kneser/Kneser.html`

Funding: This work was funded by the ERC Advanced Grant ALEXANDRIA (Project GA 742178)[1,3], the Cambridge Mathematics Placements (CMP) Internship Programme[1]