

# Khovanskii's Theorem

Angeliki Koutsoukou-Argyraki and Lawrence C. Paulson

February 6, 2026

## Abstract

We formalise the proof of an important theorem in additive combinatorics due to Khovanskii [2, 3], attesting that the cardinality of the set of all sums of  $n$  many elements of  $A$ , where  $A$  is a finite subset of an abelian group, is a polynomial in  $n$  for all sufficiently large  $n$ . We follow a proof of the theorem due to Nathanson and Ruzsa [4, 5] as presented in the notes “Introduction to Additive Combinatorics” by Timothy Gowers [1] for the University of Cambridge.

## Contents

<b>1</b>	<b>Product Operator for Commutative Monoids</b>	<b>3</b>
1.1	Products over Finite Sets . . . . .	3
1.2	Results for Abelian Groups . . . . .	6
<b>2</b>	<b>Khovanskii's Theorem</b>	<b>6</b>
2.1	Arithmetic operations on lists, pointwise on the elements . . .	7
2.2	The pointwise ordering on two equal-length lists of natural numbers . . . . .	8
2.3	Pointwise minimum and maximum of a set of lists . . . . .	10
2.4	A locale to fix the finite subset $A \subseteq G$ . . . . .	10
2.5	Adding one to a list element . . . . .	12
2.6	The set of all $r$ -tuples that sum to $n$ . . . . .	13
2.7	Lemma 2.7 in Gowers's notes . . . . .	14
2.8	The set of minimal elements of a set of $r$ -tuples is finite . . .	15
2.9	Towards Lemma 2.9 in Gowers's notes . . . . .	15
2.10	Towards the main theorem . . . . .	16

**Acknowledgements** The authors were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

# 1 Product Operator for Commutative Monoids

**theory** *FiniteProduct*

**imports**

*Jacobson-Basic-Algebra.Group-Theory*

**begin**

## 1.1 Products over Finite Sets

**context** *commutative-monoid* **begin**

**definition** *M-ify*  $x \equiv \text{if } x \in M \text{ then } x \text{ else } \mathbf{1}$

**definition** *fincomp*  $f A \equiv \text{if finite } A \text{ then } \text{Finite-Set.fold } (\lambda x y. f x \cdot M\text{-ify } y) \mathbf{1} A \text{ else } \mathbf{1}$

**lemma** *fincomp-empty* [*simp*]:  $\text{fincomp } f \ \{\} = \mathbf{1}$   
*<proof>*

**lemma** *fincomp-infinite*[*simp*]:  $\text{infinite } A \implies \text{fincomp } f A = \mathbf{1}$   
*<proof>*

**lemma** *left-commute*:  $\llbracket a \in M; b \in M; c \in M \rrbracket \implies b \cdot (a \cdot c) = a \cdot (b \cdot c)$   
*<proof>*

**lemma** *comp-fun-commute-onI*:

**assumes**  $f \in F \rightarrow M$

**shows** *comp-fun-commute-on*  $F (\lambda x y. f x \cdot M\text{-ify } y)$

*<proof>*

**lemma** *fincomp-closed* [*simp*]:

**assumes**  $f \in F \rightarrow M$

**shows**  $\text{fincomp } f F \in M$

*<proof>*

**lemma** *fincomp-insert* [*simp*]:

**assumes**  $F$ : *finite*  $F$   $a \notin F$  **and**  $f$ :  $f \in F \rightarrow M$   $f a \in M$

**shows**  $\text{fincomp } f (\text{insert } a F) = f a \cdot \text{fincomp } f F$

*<proof>*

**lemma** *fincomp-unit-eqI*:  $(\bigwedge x. x \in A \implies f x = \mathbf{1}) \implies \text{fincomp } f A = \mathbf{1}$

*<proof>*

**lemma** *fincomp-unit* [*simp*]:  $\text{fincomp } (\lambda i. \mathbf{1}) A = \mathbf{1}$

*<proof>*

**lemma** *funcset-Int-left* [*simp, intro*]:

$\llbracket f \in A \rightarrow C; g \in B \rightarrow C \rrbracket \implies f \in A \text{ Int } B \rightarrow C$

$\langle proof \rangle$

**lemma** *funcset-Un-left* [iff]:

$$(f \in A \text{ Un } B \rightarrow C) = (f \in A \rightarrow C \wedge f \in B \rightarrow C)$$

$\langle proof \rangle$

**lemma** *fincomp-Un-Int*:

$$\llbracket finite\ A; finite\ B; g \in A \rightarrow M; g \in B \rightarrow M \rrbracket \implies$$

$$fincomp\ g\ (A \cup B) \cdot fincomp\ g\ (A \cap B) =$$

$$fincomp\ g\ A \cdot fincomp\ g\ B$$

— The reversed orientation looks more natural, but LOOPS as a simprule!

$\langle proof \rangle$

**lemma** *fincomp-Un-disjoint*:

$$\llbracket finite\ A; finite\ B; A \cap B = \{\}; g \in A \rightarrow M; g \in B \rightarrow M \rrbracket$$

$$\implies fincomp\ g\ (A \cup B) = fincomp\ g\ A \cdot fincomp\ g\ B$$

$\langle proof \rangle$

**lemma** *fincomp-comp*:

$$\llbracket f \in A \rightarrow M; g \in A \rightarrow M \rrbracket \implies fincomp\ (\lambda x. f\ x \cdot g\ x)\ A = (fincomp\ f\ A \cdot fincomp\ g\ A)$$

$\langle proof \rangle$

**lemma** *fincomp-cong'*:

$$\text{assumes } A = B\ g \in B \rightarrow M \wedge i. i \in B \implies f\ i = g\ i$$

$$\text{shows } fincomp\ f\ A = fincomp\ g\ B$$

$\langle proof \rangle$

**lemma** *fincomp-cong*:

$$\text{assumes } A = B\ g \in B \rightarrow M \wedge i. i \in B = simp \implies f\ i = g\ i$$

$$\text{shows } fincomp\ f\ A = fincomp\ g\ B$$

$\langle proof \rangle$

Usually, if this rule causes a failed congruence proof error, the reason is that the premise  $g \in B \rightarrow M$  cannot be shown. Adding *Pi-def* to the simpset is often useful. For this reason, *fincomp-cong* is not added to the simpset by default.

**lemma** *fincomp-0* [simp]:

$$f \in \{0 :: nat\} \rightarrow M \implies fincomp\ f\ \{..0\} = f\ 0$$

$\langle proof \rangle$

**lemma** *fincomp-0'*:  $f \in \{..n\} \rightarrow M \implies (f\ 0) \cdot fincomp\ f\ \{Suc\ 0..n\} = fincomp\ f\ \{..n\}$

$\langle proof \rangle$

**lemma** *fincomp-Suc* [simp]:

$$f \in \{..Suc\ n\} \rightarrow M \implies fincomp\ f\ \{..Suc\ n\} = (f\ (Suc\ n)) \cdot fincomp\ f\ \{..n\}$$

$\langle proof \rangle$

**lemma** *fincomp-Suc2*:

$f \in \{..Suc\ n\} \rightarrow M \implies fincomp\ f\ \{..Suc\ n\} = (fincomp\ (\%i.\ f\ (Suc\ i))\ \{..n\} \cdot f\ 0)$   
*<proof>*

**lemma** *fincomp-Suc3*:

**assumes**  $f \in \{..n\ :: nat\} \rightarrow M$   
**shows**  $fincomp\ f\ \{..n\} = (f\ n) \cdot fincomp\ f\ \{..<\ n\}$   
*<proof>*

**lemma** *fincomp-reindex*:

$f \in (h\ 'A) \rightarrow M \implies$   
 $inj\text{-on}\ h\ A \implies fincomp\ f\ (h\ 'A) = fincomp\ (\lambda x.\ f\ (h\ x))\ A$   
*<proof>*

**lemma** *fincomp-const*:

**assumes**  $a\ [simp]: a \in M$   
**shows**  $fincomp\ (\lambda x.\ a)\ A = rec\text{-nat}\ \mathbf{1}\ (\lambda u.\ (\cdot)\ a)\ (card\ A)$   
*<proof>*

**lemma** *fincomp-singleton*:

**assumes**  $i\text{-in}\ A: i \in A$  **and**  $fin\text{-}A: finite\ A$  **and**  $f\text{-}Pi: f \in A \rightarrow M$   
**shows**  $fincomp\ (\lambda j.\ if\ i = j\ then\ f\ j\ else\ \mathbf{1})\ A = f\ i$   
*<proof>*

**lemma** *fincomp-singleton-swap*:

**assumes**  $i\text{-in}\ A: i \in A$  **and**  $fin\text{-}A: finite\ A$  **and**  $f\text{-}Pi: f \in A \rightarrow M$   
**shows**  $fincomp\ (\lambda j.\ if\ j = i\ then\ f\ j\ else\ \mathbf{1})\ A = f\ i$   
*<proof>*

**lemma** *fincomp-mono-neutral-cong-left*:

**assumes**  $finite\ B$   
**and**  $A \subseteq B$   
**and**  $1: \bigwedge i.\ i \in B - A \implies h\ i = \mathbf{1}$   
**and**  $gh: \bigwedge x.\ x \in A \implies g\ x = h\ x$   
**and**  $h: h \in B \rightarrow M$   
**shows**  $fincomp\ g\ A = fincomp\ h\ B$   
*<proof>*

**lemma** *fincomp-mono-neutral-cong-right*:

**assumes**  $finite\ B$   
**and**  $A \subseteq B \bigwedge i.\ i \in B - A \implies g\ i = \mathbf{1} \bigwedge x.\ x \in A \implies g\ x = h\ x\ g \in B \rightarrow M$   
**shows**  $fincomp\ g\ B = fincomp\ h\ A$   
*<proof>*

**lemma** *fincomp-mono-neutral-cong*:

**assumes**  $[simp]: finite\ B\ finite\ A$   
**and**  $*$ :  $\bigwedge i.\ i \in B - A \implies h\ i = \mathbf{1} \bigwedge i.\ i \in A - B \implies g\ i = \mathbf{1}$   
**and**  $gh: \bigwedge x.\ x \in A \cap B \implies g\ x = h\ x$

**and**  $g: g \in A \rightarrow M$   
**and**  $h: h \in B \rightarrow M$   
**shows**  $\text{fincomp } g \ A = \text{fincomp } h \ B$   
 $\langle \text{proof} \rangle$

**lemma** *fincomp-UN-disjoint*:

**assumes**

$\text{finite } I \wedge i. i \in I \implies \text{finite } (A \ i) \text{ pairwise } (\lambda i \ j. \text{disjnt } (A \ i) \ (A \ j)) \ I$

$\wedge i \ x. i \in I \implies x \in A \ i \implies g \ x \in M$

**shows**  $\text{fincomp } g \ (\bigcup (A \ ' \ I)) = \text{fincomp } (\lambda i. \text{fincomp } g \ (A \ i)) \ I$

$\langle \text{proof} \rangle$

**lemma** *fincomp-Union-disjoint*:

$\llbracket \text{finite } C; \wedge A. A \in C \implies \text{finite } A \wedge (\forall x \in A. f \ x \in M); \text{pairwise disjoint } C \rrbracket \implies$

$\text{fincomp } f \ (\bigcup C) = \text{fincomp } (\text{fincomp } f) \ C$

$\langle \text{proof} \rangle$

**end**

## 1.2 Results for Abelian Groups

**context** *abelian-group* **begin**

**lemma** *fincomp-inverse*:

$f \in A \rightarrow G \implies \text{fincomp } (\lambda x. \text{inverse } (f \ x)) \ A = \text{inverse } (\text{fincomp } f \ A)$

$\langle \text{proof} \rangle$

Jeremy Avigad. This should be generalized to arbitrary groups, not just Abelian ones, using Lagrange's theorem.

**lemma** *power-order-eq-one*:

**assumes** *fin [simp]*:  $\text{finite } G$

**and** *a [simp]*:  $a \in G$

**shows**  $\text{rec-nat } \mathbf{1} \ (\lambda u. (\cdot) \ a) \ (\text{card } G) = \mathbf{1}$

$\langle \text{proof} \rangle$

**end**

**end**

## 2 Khovanskii's Theorem

We formalise the proof of an important theorem in additive combinatorics due to Khovanskii, attesting that the cardinality of the set of all sums of  $n$  many elements of  $A$ , where  $A$  is a finite subset of an abelian group, is a polynomial in  $n$  for all sufficiently large  $n$ . We follow a proof due to Nathanson and Ruzsa as presented in the notes "Introduction to Additive Combinatorics" by Timothy Gowers for the University of Cambridge.

**theory** *Khovanskii*

**imports**

*FiniteProduct*

*Pluenecke-Ruzsa-Inequality.Pluenecke-Ruzsa-Inequality*

*Bernoulli.Bernoulli* — sums of a fixed power are polynomials

*HOL-Analysis.Weierstrass-Theorems* — needed for polynomial function

*HOL-Library.List-Lenlexorder* — lexicographic ordering for the type *nat*

*list*

**begin**

The sum of the elements of a list

**abbreviation**  $\sigma \equiv \text{sum-list}$

Related to the nsets of Ramsey.thy, but simpler

**definition** *finsets* :: [*'a set, nat*]  $\Rightarrow$  *'a set set*

**where** *finsets A n*  $\equiv \{N. N \subseteq A \wedge \text{card } N = n\}$

**lemma** *card-finsets*: *finite N*  $\implies \text{card } (\text{finsets } N \ k) = \text{card } N \ \text{choose } k$

*<proof>*

**lemma** *sorted-map-plus-iff* [*simp*]:

**fixes** *a*::*'a::linordered-cancel-ab-semigroup-add*

**shows** *sorted (map ((+) a) xs)*  $\longleftrightarrow$  *sorted xs*

*<proof>*

**lemma** *distinct-map-plus-iff* [*simp*]:

**fixes** *a*::*'a::linordered-cancel-ab-semigroup-add*

**shows** *distinct (map ((+) a) xs)*  $\longleftrightarrow$  *distinct xs*

*<proof>*

## 2.1 Arithmetic operations on lists, pointwise on the elements

Weak type class properties. Cancellation is difficult to arrange because of complications when lists differ in length.

**instantiation** *list* :: (*plus*) *plus*

**begin**

**definition** *plus-list*  $\equiv \text{map2 } (+)$

**instance***<proof>*

**end**

**lemma** *length-plus-list* [*simp*]:

**fixes** *xs* :: *'a::plus list*

**shows** *length (xs+ys)* = *min (length xs) (length ys)*

*<proof>*

**lemma** *plus-Nil* [*simp*]:  $[] + xs = []$

*<proof>*

**lemma** *plus-Cons*:  $(y \# ys) + (x \# xs) = (y+x) \# (ys+xs)$

*<proof>*

**lemma** *nth-plus-list* [*simp*]:  
 **fixes** *xs* :: 'a::plus list  
 **assumes**  $i < \text{length } xs$   $i < \text{length } ys$   
 **shows**  $(xs+ys)!i = xs!i + ys!i$   
 *<proof>*

**instantiation** *list* :: (*minus*) *minus*  
**begin**  
**definition** *minus-list*  $\equiv \text{map2 } (-)$   
**instance***<proof>*  
**end**

**lemma** *length-minus-list* [*simp*]:  
 **fixes** *xs* :: 'a::minus list  
 **shows**  $\text{length } (xs-ys) = \min (\text{length } xs) (\text{length } ys)$   
 *<proof>*

**lemma** *minus-Nil* [*simp*]:  $[] - xs = []$   
 *<proof>*

**lemma** *minus-Cons*:  $(y \# ys) - (x \# xs) = (y-x) \# (ys-xs)$   
 *<proof>*

**lemma** *nth-minus-list* [*simp*]:  
 **fixes** *xs* :: 'a::minus list  
 **assumes**  $i < \text{length } xs$   $i < \text{length } ys$   
 **shows**  $(xs-ys)!i = xs!i - ys!i$   
 *<proof>*

**instance** *list* :: (*ab-semigroup-add*) *ab-semigroup-add*  
 *<proof>*

## 2.2 The pointwise ordering on two equal-length lists of natural numbers

Gowers uses the usual symbol ( $\leq$ ) for his pointwise ordering. In our development,  $\leq$  is the lexicographic ordering and  $\sqsubseteq$  is the pointwise ordering.

**definition** *pointwise-le* :: [*nat list*, *nat list*]  $\Rightarrow$  *bool* (**infixr**  $\sqsubseteq$  50 )  
 **where**  $x \sqsubseteq y \equiv \text{list-all2 } (\leq) x y$

**definition** *pointwise-less* :: [*nat list*, *nat list*]  $\Rightarrow$  *bool* (**infixr**  $\sqtriangleleft$  50 )  
 **where**  $x \sqtriangleleft y \equiv x \sqsubseteq y \wedge x \neq y$

**lemma** *pointwise-le-iff-nth*:  
  $x \sqsubseteq y \iff \text{length } x = \text{length } y \wedge (\forall i < \text{length } x. x!i \leq y!i)$   
 *<proof>*

**lemma** *pointwise-le-iff*:

$x \sqsubseteq y \longleftrightarrow \text{length } x = \text{length } y \wedge (\forall (i,j) \in \text{set } (\text{zip } x \ y). i \leq j)$   
*<proof>*

**lemma** *pointwise-append-le-iff* [simp]:  $u @ x \sqsubseteq u @ y \longleftrightarrow x \sqsubseteq y$   
*<proof>*

**lemma** *pointwise-le-refl* [iff]:  $x \sqsubseteq x$   
*<proof>*

**lemma** *pointwise-le-antisym*:  $\llbracket x \sqsubseteq y; y \sqsubseteq x \rrbracket \Longrightarrow x = y$   
*<proof>*

**lemma** *pointwise-le-trans*:  $\llbracket x \sqsubseteq y; y \sqsubseteq z \rrbracket \Longrightarrow x \sqsubseteq z$   
*<proof>*

**lemma** *pointwise-le-Nil* [simp]:  $\text{Nil} \sqsubseteq x \longleftrightarrow x = \text{Nil}$   
*<proof>*

**lemma** *pointwise-le-Nil2* [simp]:  $x \sqsubseteq \text{Nil} \longleftrightarrow x = \text{Nil}$   
*<proof>*

**lemma** *pointwise-le-iff-less-equal*:  $x \sqsubseteq y \longleftrightarrow x \triangleleft y \vee x = y$   
*<proof>*

**lemma** *pointwise-less-iff*:

$x \triangleleft y \longleftrightarrow x \sqsubseteq y \wedge (\exists (i,j) \in \text{set } (\text{zip } x \ y). i < j)$   
*<proof>*

**lemma** *pointwise-less-iff2*:  $x \triangleleft y \longleftrightarrow x \sqsubseteq y \wedge (\exists k < \text{length } x. x ! k < y ! k)$   
*<proof>*

**lemma** *pointwise-less-Nil* [simp]:  $\neg \text{Nil} \triangleleft x$   
*<proof>*

**lemma** *pointwise-less-Nil2* [simp]:  $\neg x \triangleleft \text{Nil}$   
*<proof>*

**lemma** *zero-pointwise-le-iff* [simp]:  $\text{replicate } r \ 0 \sqsubseteq x \longleftrightarrow \text{length } x = r$   
*<proof>*

**lemma** *pointwise-le-imp-σ*:

**assumes**  $xs \sqsubseteq ys$  **shows**  $\sigma \ xs \leq \sigma \ ys$   
*<proof>*

**lemma** *sum-list-plus*:

**fixes**  $xs :: 'a::\text{comm-monoid-add list}$

**assumes**  $\text{length } xs = \text{length } ys$  **shows**  $\sigma \ (xs + ys) = \sigma \ xs + \sigma \ ys$

*<proof>*

**lemma** *sum-list-minus*:

**assumes**  $xs \trianglelefteq ys$  **shows**  $\sigma (ys - xs) = \sigma ys - \sigma xs$

*<proof>*

## 2.3 Pointwise minimum and maximum of a set of lists

**definition** *min-pointwise* ::  $[nat, nat\ list\ set] \Rightarrow nat\ list$

**where**  $min\_pointwise \equiv \lambda r\ U. map (\lambda i. Min ((\lambda u. u!i) ' U)) [0..<r]$

**lemma** *min-pointwise-le*:  $\llbracket u \in U; finite\ U \rrbracket \Longrightarrow min\_pointwise (length\ u)\ U \trianglelefteq u$

*<proof>*

**lemma** *min-pointwise-ge-iff*:

**assumes**  $finite\ U\ U \neq \{\}$   $\bigwedge u. u \in U \Longrightarrow length\ u = r\ length\ x = r$

**shows**  $x \trianglelefteq min\_pointwise\ r\ U \longleftrightarrow (\forall u \in U. x \trianglelefteq u)$

*<proof>*

**definition** *max-pointwise* ::  $[nat, nat\ list\ set] \Rightarrow nat\ list$

**where**  $max\_pointwise \equiv \lambda r\ U. map (\lambda i. Max ((\lambda u. u!i) ' U)) [0..<r]$

**lemma** *max-pointwise-ge*:  $\llbracket u \in U; finite\ U \rrbracket \Longrightarrow u \trianglelefteq max\_pointwise (length\ u)\ U$

*<proof>*

**lemma** *max-pointwise-le-iff*:

**assumes**  $finite\ U\ U \neq \{\}$   $\bigwedge u. u \in U \Longrightarrow length\ u = r\ length\ x = r$

**shows**  $max\_pointwise\ r\ U \trianglelefteq x \longleftrightarrow (\forall u \in U. u \trianglelefteq x)$

*<proof>*

**lemma** *max-pointwise-mono*:

**assumes**  $X' \subseteq X\ finite\ X\ X' \neq \{\}$

**shows**  $max\_pointwise\ r\ X' \trianglelefteq max\_pointwise\ r\ X$

*<proof>*

**lemma** *pointwise-le-plus*:  $\llbracket xs \trianglelefteq ys; length\ ys \leq length\ zs \rrbracket \Longrightarrow xs \trianglelefteq ys+zs$

*<proof>*

**lemma** *pairwise-minus-cancel*:  $\llbracket z \trianglelefteq x; z \trianglelefteq y; x - z = y - z \rrbracket \Longrightarrow x = y$

*<proof>*

## 2.4 A locale to fix the finite subset $A \subseteq G$

**locale** *Khovanskii* = *additive-abelian-group* +

**fixes**  $A :: 'a\ set$

**assumes**  $AsubG: A \subseteq G$  **and**  $finA: finite\ A$

**begin**

finite products of a group element

**definition**  $Gmult :: 'a \Rightarrow nat \Rightarrow 'a$   
**where**  $Gmult\ a\ n \equiv (((\oplus)a) \overset{\sim}{\sim} n)\ \mathbf{0}$

**lemma**  $Gmult-0$  [simp]:  $Gmult\ a\ 0 = \mathbf{0}$   
 $\langle proof \rangle$

**lemma**  $Gmult-1$  [simp]:  $a \in G \implies Gmult\ a\ (Suc\ 0) = a$   
 $\langle proof \rangle$

**lemma**  $Gmult-Suc$  [simp]:  $Gmult\ a\ (Suc\ n) = a \oplus Gmult\ a\ n$   
 $\langle proof \rangle$

**lemma**  $Gmult-in-G$  [simp,intro]:  $a \in G \implies Gmult\ a\ n \in G$   
 $\langle proof \rangle$

**lemma**  $Gmult-add-add$ :  
**assumes**  $a \in G$   
**shows**  $Gmult\ a\ (m+n) = Gmult\ a\ m \oplus Gmult\ a\ n$   
 $\langle proof \rangle$

**lemma**  $Gmult-add-diff$ :  
**assumes**  $a \in G$   
**shows**  $Gmult\ a\ (n+k) \ominus Gmult\ a\ n = Gmult\ a\ k$   
 $\langle proof \rangle$

**lemma**  $Gmult-diff$ :  
**assumes**  $a \in G\ n \leq m$   
**shows**  $Gmult\ a\ m \ominus Gmult\ a\ n = Gmult\ a\ (m-n)$   
 $\langle proof \rangle$

Mapping elements of  $A$  to their numeric subscript

**abbreviation**  $idx \equiv to-nat-on\ A$

The elements of  $A$  in order

**definition**  $aA :: 'a\ list$   
**where**  $aA \equiv map\ (from-nat-into\ A)\ [0..<card\ A]$

**definition**  $\alpha :: nat\ list \Rightarrow 'a$   
**where**  $\alpha \equiv \lambda x. fincomp\ (\lambda i. Gmult\ (aA!i)\ (x!i))\ \{..<card\ A\}$

The underlying assumption is  $length\ y = length\ x$

**definition**  $useless :: nat\ list \Rightarrow bool$   
**where**  $useless\ x \equiv \exists y < x. \sigma\ y = \sigma\ x \wedge \alpha\ y = \alpha\ x \wedge length\ y = length\ x$

**abbreviation**  $useful\ x \equiv \neg\ useless\ x$

**lemma**  $alpha-replicate-0$  [simp]:  $\alpha\ (replicate\ (card\ A)\ 0) = \mathbf{0}$   
 $\langle proof \rangle$

**lemma** *idx-less-cardA*:

**assumes**  $a \in A$  **shows**  $\text{idx } a < \text{card } A$

*<proof>*

**lemma** *aA-idx-eq* [simp]:

**assumes**  $a \in A$  **shows**  $aA ! (\text{idx } a) = a$

*<proof>*

**lemma** *set-aA*:  $\text{set } aA = A$

*<proof>*

**lemma** *nth-aA-in-G* [simp]:  $i < \text{card } A \implies aA!i \in G$

*<proof>*

**lemma** *alpha-in-G* [iff]:  $\alpha x \in G$

*<proof>*

**lemma** *Gmult-in-PiG* [simp]:  $(\lambda i. \text{Gmult } (aA!i) (f i)) \in \{.. < \text{card } A\} \rightarrow G$

*<proof>*

**lemma** *alpha-plus*:

**assumes**  $\text{length } x = \text{card } A$   $\text{length } y = \text{card } A$

**shows**  $\alpha (x + y) = \alpha x \oplus \alpha y$

*<proof>*

**lemma** *alpha-minus*:

**assumes**  $y \trianglelefteq x$   $\text{length } y = \text{card } A$

**shows**  $\alpha (x - y) = \alpha x \ominus \alpha y$

*<proof>*

## 2.5 Adding one to a list element

**definition** *list-incr* ::  $\text{nat} \Rightarrow \text{nat list} \Rightarrow \text{nat list}$

**where**  $\text{list-incr } i x \equiv x[i := \text{Suc } (x!i)]$

**lemma** *list-incr-Nil* [simp]:  $\text{list-incr } i [] = []$

*<proof>*

**lemma** *list-incr-Cons* [simp]:  $\text{list-incr } (Suc i) (k \# ks) = k \# \text{list-incr } i ks$

*<proof>*

**lemma** *sum-list-incr* [simp]:  $i < \text{length } x \implies \sigma (\text{list-incr } i x) = \text{Suc } (\sigma x)$

*<proof>*

**lemma** *length-list-incr* [simp]:  $\text{length } (\text{list-incr } i x) = \text{length } x$

*<proof>*

**lemma** *nth-le-list-incr*:  $i < \text{card } A \implies x!i \leq \text{list-incr } (idx a) x!i$

*<proof>*

**lemma** *list-incr-nth-diff*:  $i < \text{length } x \implies \text{list-incr } j \ x!i - x!i = (\text{if } i = j \text{ then } 1 \text{ else } 0)$   
 ⟨proof⟩

## 2.6 The set of all $r$ -tuples that sum to $n$

**definition** *length-sum-set* ::  $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat list set}$   
 where  $\text{length-sum-set } r \ n \equiv \{x. \text{length } x = r \wedge \sigma \ x = n\}$

**lemma** *length-sum-set-Nil* [*simp*]:  $\text{length-sum-set } 0 \ n = (\text{if } n=0 \text{ then } \{\}\ \text{else } \{\})$   
 ⟨proof⟩

**lemma** *length-sum-set-Suc* [*simp*]:  $k \# ks \in \text{length-sum-set } (\text{Suc } r) \ n \longleftrightarrow (\exists m. ks \in \text{length-sum-set } r \ m \wedge n = m+k)$   
 ⟨proof⟩

**lemma** *length-sum-set-Suc-egpoll*:  $\text{length-sum-set } (\text{Suc } r) \ n \approx \text{Sigma } \{..n\} \ (\lambda i. \text{length-sum-set } r \ (n-i))$  (is ?L  $\approx$  ?R)  
 ⟨proof⟩

**lemma** *finite-length-sum-set*:  $\text{finite } (\text{length-sum-set } r \ n)$   
 ⟨proof⟩

**lemma** *card-length-sum-set*:  $\text{card } (\text{length-sum-set } (\text{Suc } r) \ n) = (\sum i \leq n. \text{card } (\text{length-sum-set } r \ (n-i)))$   
 ⟨proof⟩

**lemma** *sum-up-index-split'*:  
 assumes  $N \leq n$  shows  $(\sum i \leq n. f \ i) = (\sum i \leq n-N. f \ i) + (\sum i = \text{Suc } (n-N)..n. f \ i)$   
 ⟨proof⟩

**lemma** *sum-invert*:  $N \leq n \implies (\sum i = \text{Suc } (n - N)..n. f \ (n - i)) = (\sum j < N. f \ j)$   
 ⟨proof⟩

**lemma** *real-polynomial-function-length-sum-set*:  
 $\exists p. \text{real-polynomial-function } p \wedge (\forall n > 0. \text{real } (\text{card } (\text{length-sum-set } r \ n)) = p \ (\text{real } n))$   
 ⟨proof⟩

**lemma** *all-zeroes-replicate*:  $\text{length-sum-set } r \ 0 = \{\text{replicate } r \ 0\}$   
 ⟨proof⟩

**lemma** *length-sum-set-Suc-eq-UN*:  $\text{length-sum-set } r \ (\text{Suc } n) = (\bigcup i < r. \text{list-incr } i \ ' \ \text{length-sum-set } r \ n)$   
 ⟨proof⟩



## 2.8 The set of minimal elements of a set of $r$ -tuples is finite

The following general finiteness claim corresponds to Lemma 2.8 in Gowers's notes and is key to the main proof.

**lemma** *minimal-elements-set-tuples-finite*:  
  **assumes**  $Ur: \bigwedge x. x \in U \implies \text{length } x = r$   
  **shows** *finite (minimal-elements U)*  
   $\langle \text{proof} \rangle$

## 2.9 Towards Lemma 2.9 in Gowers's notes

Increasing sequences

**fun** *augmentum* ::  $\text{nat list} \Rightarrow \text{nat list}$   
  **where** *augmentum* [] = []  
  | *augmentum* (n#ns) = n # map ((+)n) (*augmentum* ns)

**definition** *dementum*::  $\text{nat list} \Rightarrow \text{nat list}$   
  **where** *dementum* xs  $\equiv$  xs - (0#xs)

**lemma** *dementum-Nil* [*simp*]: *dementum* [] = []  
   $\langle \text{proof} \rangle$

**lemma** *zero-notin-augmentum* [*simp*]:  $0 \notin \text{set } ns \implies 0 \notin \text{set } (\text{augmentum } ns)$   
   $\langle \text{proof} \rangle$

**lemma** *length-augmentum* [*simp*]:  $\text{length } (\text{augmentum } xs) = \text{length } xs$   
   $\langle \text{proof} \rangle$

**lemma** *sorted-augmentum* [*simp*]:  $0 \notin \text{set } ns \implies \text{sorted } (\text{augmentum } ns)$   
   $\langle \text{proof} \rangle$

**lemma** *distinct-augmentum* [*simp*]:  $0 \notin \text{set } ns \implies \text{distinct } (\text{augmentum } ns)$   
   $\langle \text{proof} \rangle$

**lemma** *augmentum-subset-sum-list*:  $\text{set } (\text{augmentum } ns) \subseteq \{..\sigma \text{ } ns\}$   
   $\langle \text{proof} \rangle$

**lemma** *sum-list-augmentum*:  $\sigma \text{ } ns \in \text{set } (\text{augmentum } ns) \iff \text{length } ns > 0$   
   $\langle \text{proof} \rangle$

**lemma** *length-dementum* [*simp*]:  $\text{length } (\text{dementum } xs) = \text{length } xs$   
   $\langle \text{proof} \rangle$

**lemma** *sorted-imp-pointwise*:  
  **assumes** *sorted* (xs@[n])  
  **shows**  $0 \# xs \leq xs @ [n]$   
   $\langle \text{proof} \rangle$

**lemma** *sum-list-dementum*:

**assumes** *sorted* ( $xs@[n]$ )  
**shows**  $\sigma$  (*dementum* ( $xs@[n]$ )) =  $n$   
 $\langle$ *proof* $\rangle$

**lemma** *augmentum-cancel*:  $\text{map } ((+)k) (\text{augmentum } ns) - (k \# \text{map } ((+)k) (\text{augmentum } ns)) = ns$   
 $\langle$ *proof* $\rangle$

**lemma** *dementum-augmentum* [*simp*]:  
**assumes**  $0 \notin \text{set } ns$   
**shows** (*dementum*  $\circ$  *sorted-list-of-set*) ((*set*  $\circ$  *augmentum*)  $ns$ ) =  $ns$  (**is** ? $L$   $ns =$  -)  
 $\langle$ *proof* $\rangle$

**lemma** *dementum-nonzero*:  
**assumes**  $ns$ : *sorted-wrt* ( $<$ )  $ns$  **and**  $0$ :  $0 \notin \text{set } ns$   
**shows**  $0 \notin \text{set} (\text{dementum } ns)$   
 $\langle$ *proof* $\rangle$

**lemma** *nth-augmentum* [*simp*]:  $i < \text{length } ns \implies \text{augmentum } ns!i = (\sum_{j \leq i} ns!j)$   
 $\langle$ *proof* $\rangle$

**lemma** *augmentum-dementum* [*simp*]:  
**assumes**  $0 \notin \text{set } ns$  *sorted*  $ns$   
**shows** *augmentum* (*dementum*  $ns$ ) =  $ns$   
 $\langle$ *proof* $\rangle$

The following lemma corresponds to Lemma 2.9 in Gowers's notes. The proof involves introducing bijective maps between  $r$ -tuples that fulfill certain properties/ $r$ -tuples and subsets of naturals, so as to show the cardinality claim.

**lemma** *bound-sum-list-card*:  
**assumes**  $r > 0$  **and**  $n$ :  $n \geq \sigma x'$  **and**  $lenx'$ : *length*  $x' = r$   
**defines**  $S \equiv \{x. x' \trianglelefteq x \wedge \sigma x = n\}$   
**shows**  $\text{card } S = (n - \sigma x' + r - 1) \text{ choose } (r-1)$   
 $\langle$ *proof* $\rangle$

## 2.10 Towards the main theorem

**lemma** *extend-tuple*:  
**assumes**  $\sigma xs \leq n$  *length*  $xs \neq 0$   
**obtains**  $ys$  **where**  $\sigma ys = n$   $xs \trianglelefteq ys$   
 $\langle$ *proof* $\rangle$

**lemma** *extend-preserving*:  
**assumes**  $\sigma xs \leq n$  *length*  $xs > 1$   $i < \text{length } xs$   
**obtains**  $ys$  **where**  $\sigma ys = n$   $xs \trianglelefteq ys$   $ys!i = xs!i$   
 $\langle$ *proof* $\rangle$

The proof of the main theorem will make use of the inclusion-exclusion

formula, in addition to the previously shown results.

**theorem** *Khovanskii*:

**assumes**  $\text{card } A > 1$

**defines**  $f \equiv \lambda n. \text{card}(\text{sumset-iterated } A \ n)$

**obtains**  $N \ p$  **where** *real-polynomial-function*  $p \wedge n. n \geq N \implies \text{real } (f \ n) = p$   
(*real n*)

*<proof>*

**end**

**end**

## References

- [1] W. T. Gowers. Introduction to additive combinatorics. Lecture notes, University of Cambridge, 2022.
- [2] A. G. Khovanskii. Newton polyhedron, Hilbert polynomial, and sums of finite sets. *Functional Analysis and Its Applications*, 26(4):276–281, 1992.
- [3] A. G. Khovanskii. Sums of finite sets, orbits of commutative semigroups, and Hilbert functions. *Functional Analysis and Its Applications*, 29(2):102–112, 1995.
- [4] M. B. Nathanson and I. Z. Ruzsa. Polynomial growth of sumsets in abelian semigroups. *Journal de Théorie des Nombres de Bordeaux*, 14(2):553–560, 2002.
- [5] I. Z. Ruzsa. Sumsets and structure. Lecture notes, Institute of Mathematics, Budapest.