Khovanskii's Theorem

Angeliki Koutsoukou-Argyraki and Lawrence C. Paulson

March 17, 2025

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

1 Product Operator for Commutative Monoids **3**
1.1 Products over Finite Sets 3
1.2 Results for Abelian Groups 6

2 Khovanskii's Theorem **6**
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* $x \equiv$ *if* $x \in M$ *then* x *else* **1**

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

lemma *fincomp-empty* [*simp*]: *fincomp* f $\{\} =$ **1**
  ⟨*proof*⟩

lemma *fincomp-infinite*[*simp*]: *infinite* $A \Longrightarrow$ *fincomp* f $A =$ **1**
  ⟨*proof*⟩

lemma *left-commute*: $[\![\ a \in M;\ b \in M;\ c \in M\ ]\!] \Longrightarrow b \cdot (a \cdot c) = a \cdot (b \cdot c)$
  ⟨*proof*⟩

lemma *comp-fun-commute-onI*:
  assumes $f \in F \to M$
  shows *comp-fun-commute-on* F $(\lambda x\ y.\ f\ x \cdot M\text{-}ify\ y)$
  ⟨*proof*⟩

lemma *fincomp-closed* [*simp*]:
  assumes $f \in F \to M$
  shows *fincomp* f $F \in M$
⟨*proof*⟩

lemma *fincomp-insert* [*simp*]:
  assumes F: *finite* F $a \notin F$ **and** f: $f \in F \to M$ $f\ a \in M$
  shows *fincomp* f (*insert* a F) $= f\ a \cdot$ *fincomp* f F
⟨*proof*⟩

lemma *fincomp-unit-eqI*: $(\bigwedge x.\ x \in A \Longrightarrow f\ x = \mathbf{1}) \Longrightarrow$ *fincomp* f $A =$ **1**
⟨*proof*⟩

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

lemma *funcset-Int-left* [*simp*, *intro*]:
  $[\![f \in A \to C;\ f \in B \to C]\!] \Longrightarrow f \in A\ Int\ B \to C$

$\langle proof \rangle$

lemma *funcset-Un-left* [*iff*]:
$(f \in A\ Un\ B \to C) = (f \in A \to C \wedge f \in B \to C)$
$\langle proof \rangle$

lemma *fincomp-Un-Int*:
$[\![finite\ A;\ finite\ B;\ g \in A \to M;\ g \in B \to M]\!] \Longrightarrow$
$fincomp\ g\ (A \cup B) \cdot fincomp\ g\ (A \cap B) =$
$fincomp\ g\ A \cdot fincomp\ g\ B$
— The reversed orientation looks more natural, but LOOPS as a simprule!
$\langle proof \rangle$

lemma *fincomp-Un-disjoint*:
$[\![finite\ A;\ finite\ B;\ A \cap B = \{\};\ g \in A \to M;\ g \in B \to M]\!]$
$\Longrightarrow fincomp\ g\ (A \cup B) = fincomp\ g\ A \cdot fincomp\ g\ B$
$\langle proof \rangle$

lemma *fincomp-comp*:
$[\![f \in A \to M;\ g \in A \to M]\!] \Longrightarrow fincomp\ (\lambda x.\ f\ x \cdot g\ x)\ A = (fincomp\ f\ A \cdot fincomp\ g\ A)$
$\langle proof \rangle$

lemma *fincomp-cong'*:
assumes $A = B\ g \in B \to M\ \bigwedge i.\ i \in B \Longrightarrow f\ i = g\ i$
shows $fincomp\ f\ A = fincomp\ g\ B$
$\langle proof \rangle$

lemma *fincomp-cong*:
assumes $A = B\ g \in B \to M\ \bigwedge i.\ i \in B =simp=> f\ i = g\ i$
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 $g \in B \to M$ 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*]:
$f \in \{0::nat\} \to M \Longrightarrow fincomp\ f\ \{..0\} = f\ 0$
$\langle proof \rangle$

lemma *fincomp-0'*: $f \in \{..n\} \to M \Longrightarrow (f\ 0) \cdot fincomp\ f\ \{Suc\ 0..n\} = fincomp\ f\ \{..n\}$
$\langle proof \rangle$

lemma *fincomp-Suc* [*simp*]:
$f \in \{..Suc\ n\} \to M \Longrightarrow fincomp\ f\ \{..Suc\ n\} = (f\ (Suc\ n) \cdot fincomp\ f\ \{..n\})$
$\langle proof \rangle$

lemma *fincomp-Suc2*:
$f \in \{..Suc\ n\} \to M \Longrightarrow fincomp\ f\ \{..Suc\ n\} = (fincomp\ (\%i.\ f\ (Suc\ i))\ \{..n\} \cdot f\ 0)$
⟨*proof*⟩

lemma *fincomp-Suc3*:
assumes $f \in \{..n :: nat\} \to M$
shows $fincomp\ f\ \{..\ n\} = (f\ n) \cdot fincomp\ f\ \{..< n\}$
⟨*proof*⟩

lemma *fincomp-reindex*:
$f \in (h\ `\ A) \to M \Longrightarrow$
$inj\text{-}on\ h\ A \Longrightarrow fincomp\ f\ (h\ `\ A) = fincomp\ (\lambda x.\ f\ (h\ x))\ A$
⟨*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-in-A*: $i \in A$ **and** *fin-A*: *finite A* **and** *f-Pi*: $f \in A \to M$
shows $fincomp\ (\lambda j.\ if\ i = j\ then\ f\ j\ else\ \mathbf{1})\ A = f\ i$
⟨*proof*⟩

lemma *fincomp-singleton-swap*:
assumes *i-in-A*: $i \in A$ **and** *fin-A*: *finite A* **and** *f-Pi*: $f \in A \to M$
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*: $\bigwedge i.\ i \in B - A \Longrightarrow h\ i = \mathbf{1}$
and *gh*: $\bigwedge x.\ x \in A \Longrightarrow g\ x = h\ x$
and *h*: $h \in B \to M$
shows $fincomp\ g\ A = fincomp\ h\ B$
⟨*proof*⟩

lemma *fincomp-mono-neutral-cong-right*:
assumes *finite B*
and $A \subseteq B$ $\bigwedge i.\ i \in B - A \Longrightarrow g\ i = \mathbf{1}$ $\bigwedge x.\ x \in A \Longrightarrow g\ x = h\ x$ $g \in B \to M$
shows $fincomp\ g\ B = fincomp\ h\ A$
⟨*proof*⟩

lemma *fincomp-mono-neutral-cong*:
assumes [*simp*]: *finite B finite A*
and $*$: $\bigwedge i.\ i \in B - A \Longrightarrow h\ i = \mathbf{1}$ $\bigwedge i.\ i \in A - B \Longrightarrow g\ i = \mathbf{1}$
and *gh*: $\bigwedge x.\ x \in A \cap B \Longrightarrow g\ x = h\ x$

and *g*: $g \in A \to M$
and *h*: $h \in B \to M$
shows *fincomp g A = fincomp h B*
⟨*proof*⟩

lemma *fincomp-UN-disjoint*:
assumes
finite I $\bigwedge i.\ i \in I \Longrightarrow$ *finite* $(A\ i)$ *pairwise* $(\lambda i\ j.$ *disjnt* $(A\ i)\ (A\ j))\ I$
$\bigwedge i\ x.\ i \in I \Longrightarrow x \in A\ i \Longrightarrow g\ x \in M$
shows *fincomp g* $(\bigcup (A\ `\ I))$ = *fincomp* $(\lambda i.$ *fincomp g* $(A\ i))\ I$
⟨*proof*⟩

lemma *fincomp-Union-disjoint*:
⟦*finite C*; $\bigwedge A.\ A \in C \Longrightarrow$ *finite A* $\wedge (\forall x{\in}A.\ f\ x \in M)$; *pairwise disjnt C*⟧ $\Longrightarrow$
fincomp f $(\bigcup C)$ = *fincomp* (*fincomp f*) *C*
⟨*proof*⟩

end

1.2 Results for Abelian Groups

context *abelian-group* **begin**

lemma *fincomp-inverse*:
$f \in A \to G \Longrightarrow$ *fincomp* $(\lambda x.$ *inverse* $(f\ x))\ A$ = *inverse* (*fincomp f A*)
⟨*proof*⟩

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*]: *finite G*
and *a* [*simp*]: $a \in G$
shows *rec-nat* **1** $(\lambda u.\ (\cdot)\ a)$ (*card G*) = **1**
⟨*proof*⟩

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
Pluennecke-Ruzsa-Inequality.Pluennecke-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 $\sigma \equiv$ *sum-list*

Related to the nsets of Ramsey.thy, but simpler

definition *finsets* :: ['*a set*, *nat*] ⇒ '*a set set*
where *finsets A n* ≡ {*N*. *N* ⊆ *A* ∧ *card N* = *n*}

lemma *card-finsets*: *finite N* ⟹ *card* (*finsets N k*) = *card N choose k*
⟨*proof*⟩

lemma *sorted-map-plus-iff* [*simp*]:
fixes *a*::'*a*::*linordered-cancel-ab-semigroup-add*
shows *sorted* (*map* ((+) *a*) *xs*) ⟷ *sorted xs*
⟨*proof*⟩

lemma *distinct-map-plus-iff* [*simp*]:
fixes *a*::'*a*::*linordered-cancel-ab-semigroup-add*
shows *distinct* (*map* ((+) *a*) *xs*) ⟷ *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* ≡ *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 $i < length\ xs\ i < length\ ys$
shows $(xs+ys)!i = xs!i + ys!i$
⟨*proof*⟩

instantiation *list* :: (*minus*) *minus*
begin
definition *minus-list* ≡ *map2* (−)
instance⟨*proof*⟩
end

lemma *length-minus-list* [*simp*]:
fixes *xs* :: *'a::minus list*
shows *length* (*xs*−*ys*) = *min* (*length xs*) (*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 $i < length\ xs\ i < length\ ys$
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 (≤) for his pointwise ordering. In our development, ≤ is the lexicographic ordering and ⊴ is the pointwise ordering.

definition *pointwise-le* :: [*nat list*, *nat list*] ⇒ *bool* (**infixr** ‹⊴› *50*)
where *x* ⊴ *y* ≡ *list-all2* (≤) *x y*

definition *pointwise-less* :: [*nat list*, *nat list*] ⇒ *bool* (**infixr** ‹⊲› *50*)
where *x* ⊲ *y* ≡ *x* ⊴ *y* ∧ *x* ≠ *y*

lemma *pointwise-le-iff-nth*:
x ⊴ *y* ⟷ *length x* = *length y* ∧ (∀ *i* < *length x*. *x*!*i* ≤ *y*!*i*)
⟨*proof*⟩

lemma *pointwise-le-iff*:
$x \trianglelefteq y \longleftrightarrow \mathit{length}\ x = \mathit{length}\ y \wedge (\forall (i,j) \in \mathit{set}\ (\mathit{zip}\ x\ y).\ i{\leq}j)$
⟨*proof*⟩

lemma *pointwise-append-le-iff* [*simp*]: $u@x \trianglelefteq u@y \longleftrightarrow x \trianglelefteq y$
⟨*proof*⟩

lemma *pointwise-le-refl* [*iff*]: $x \trianglelefteq x$
⟨*proof*⟩

lemma *pointwise-le-antisym*: $[\![x \trianglelefteq y;\ y \trianglelefteq x]\!] \Longrightarrow x{=}y$
⟨*proof*⟩

lemma *pointwise-le-trans*: $[\![x \trianglelefteq y;\ y \trianglelefteq z]\!] \Longrightarrow x \trianglelefteq z$
⟨*proof*⟩

lemma *pointwise-le-Nil* [*simp*]: $\mathit{Nil} \trianglelefteq x \longleftrightarrow x = \mathit{Nil}$
⟨*proof*⟩

lemma *pointwise-le-Nil2* [*simp*]: $x \trianglelefteq \mathit{Nil} \longleftrightarrow x = \mathit{Nil}$
⟨*proof*⟩

lemma *pointwise-le-iff-less-equal*: $x \trianglelefteq y \longleftrightarrow x \triangleleft y \vee x = y$
⟨*proof*⟩

lemma *pointwise-less-iff*:
$x \triangleleft y \longleftrightarrow x \trianglelefteq y \wedge (\exists (i,j) \in \mathit{set}\ (\mathit{zip}\ x\ y).\ i{<}j)$
⟨*proof*⟩

lemma *pointwise-less-iff2*: $x \triangleleft y \longleftrightarrow x \trianglelefteq y \wedge (\exists k < \mathit{length}\ x.\ x!k < y\ !\ k)$
⟨*proof*⟩

lemma *pointwise-less-Nil* [*simp*]: $\neg\ \mathit{Nil} \triangleleft x$
⟨*proof*⟩

lemma *pointwise-less-Nil2* [*simp*]: $\neg\ x \triangleleft \mathit{Nil}$
⟨*proof*⟩

lemma *zero-pointwise-le-iff* [*simp*]: $\mathit{replicate}\ r\ 0 \trianglelefteq x \longleftrightarrow \mathit{length}\ x = r$
⟨*proof*⟩

lemma *pointwise-le-imp-σ*:
assumes $xs \trianglelefteq ys$ **shows** $\sigma\ xs \leq \sigma\ ys$
⟨*proof*⟩

lemma *sum-list-plus*:
fixes $xs :: {}'a{::}\mathit{comm\text{-}monoid\text{-}add}\ \mathit{list}$
assumes $\mathit{length}\ xs = \mathit{length}\ ys$ **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 $min\text{-}pointwise \equiv \lambda r\ U.\ map\ (\lambda i.\ Min\ ((\lambda u.\ u!i)\ `\ U))\ [0..<r]$

lemma *min-pointwise-le*: $[\![u \in U;\ finite\ U]\!] \Longrightarrow min\text{-}pointwise\ (length\ u)\ U \trianglelefteq u$
⟨*proof*⟩

lemma *min-pointwise-ge-iff*:
assumes $finite\ U\ U \neq \{\}\ \bigwedge u.\ u \in U \Longrightarrow length\ u = r\ length\ x = r$
shows $x \trianglelefteq min\text{-}pointwise\ r\ U \longleftrightarrow (\forall u \in U.\ x \trianglelefteq u)$
⟨*proof*⟩

definition *max-pointwise* :: $[nat,\ nat\ list\ set] \Rightarrow nat\ list$
where $max\text{-}pointwise \equiv \lambda r\ U.\ map\ (\lambda i.\ Max\ ((\lambda u.\ u!i)\ `\ U))\ [0..<r]$

lemma *max-pointwise-ge*: $[\![u \in U;\ finite\ U]\!] \Longrightarrow u \trianglelefteq max\text{-}pointwise\ (length\ u)\ U$
⟨*proof*⟩

lemma *max-pointwise-le-iff*:
assumes $finite\ U\ U \neq \{\}\ \bigwedge u.\ u \in U \Longrightarrow length\ u = r\ length\ x = r$
shows $max\text{-}pointwise\ r\ U \trianglelefteq x \longleftrightarrow (\forall u \in U.\ u \trianglelefteq x)$
⟨*proof*⟩

lemma *max-pointwise-mono*:
assumes $X' \subseteq X\ finite\ X\ X' \neq \{\}$
shows $max\text{-}pointwise\ r\ X' \trianglelefteq max\text{-}pointwise\ r\ X$
⟨*proof*⟩

lemma *pointwise-le-plus*: $[\![xs \trianglelefteq ys;\ length\ ys \leq length\ zs]\!] \Longrightarrow xs \trianglelefteq ys+zs$
⟨*proof*⟩

lemma *pairwise-minus-cancel*: $[\![z \trianglelefteq x;\ z \trianglelefteq y;\ x - z = y - z]\!] \Longrightarrow x = y$
⟨*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* ⇒ *nat* ⇒ *'a*
where *Gmult a n* ≡ (((⊕)*a*) ^^ *n*) **0**

lemma *Gmult-0* [*simp*]: *Gmult a 0* = **0**
⟨*proof*⟩

lemma *Gmult-1* [*simp*]: *a* ∈ *G* ⟹ *Gmult a* (*Suc 0*) = *a*
⟨*proof*⟩

lemma *Gmult-Suc* [*simp*]: *Gmult a* (*Suc n*) = *a* ⊕ *Gmult a n*
⟨*proof*⟩

lemma *Gmult-in-G* [*simp*,*intro*]: *a* ∈ *G* ⟹ *Gmult a n* ∈ *G*
⟨*proof*⟩

lemma *Gmult-add-add*:
assumes *a* ∈ *G*
shows *Gmult a* (*m*+*n*) = *Gmult a m* ⊕ *Gmult a n*
⟨*proof*⟩

lemma *Gmult-add-diff*:
assumes *a* ∈ *G*
shows *Gmult a* (*n*+*k*) ⊖ *Gmult a n* = *Gmult a k*
⟨*proof*⟩

lemma *Gmult-diff*:
assumes *a* ∈ *G n*≤*m*
shows *Gmult a m* ⊖ *Gmult a n* = *Gmult a* (*m*−*n*)
⟨*proof*⟩

Mapping elements of A to their numeric subscript

abbreviation *idx* ≡ *to-nat-on A*

The elements of A in order

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

definition α :: *nat list* ⇒ *'a*
where α ≡ λ*x*. *fincomp* (λ*i*. *Gmult* (*aA*!*i*) (*x*!*i*)) {..<*card A*}

The underlying assumption is *length y* = *length x*

definition *useless*:: *nat list* ⇒ *bool*
where *useless x* ≡ ∃ *y* < *x*. σ *y* = σ *x* ∧ α *y* = α *x* ∧ *length y* = *length x*

abbreviation *useful x* ≡ ¬ *useless x*

lemma *alpha-replicate-0* [*simp*]: α (*replicate* (*card A*) *0*) = **0**
⟨*proof*⟩

lemma *idx-less-cardA*:
  assumes $a \in A$ **shows** $idx\ a < card\ A$
  ⟨*proof*⟩

lemma *aA-idx-eq* [*simp*]:
  assumes $a \in A$ **shows** $aA\ !\ (idx\ a) = a$
  ⟨*proof*⟩

lemma *set-aA*: $set\ aA = A$
  ⟨*proof*⟩

lemma *nth-aA-in-G* [*simp*]: $i < card\ A \Longrightarrow aA!i \in G$
  ⟨*proof*⟩

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

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

lemma *alpha-plus*:
  assumes $length\ x = card\ A$ $length\ y = card\ A$
  shows $\alpha\ (x + y) = \alpha\ x \oplus \alpha\ y$
⟨*proof*⟩

lemma *alpha-minus*:
  assumes $y \trianglelefteq x$ $length\ y = card\ A$
  shows $\alpha\ (x - y) = \alpha\ x \ominus \alpha\ y$
⟨*proof*⟩

2.5 Adding one to a list element

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

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

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

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

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

lemma *nth-le-list-incr*: $i < card\ A \Longrightarrow x!i \le list\text{-}incr\ (idx\ a)\ x!i$
  ⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

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

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

lemma *alpha-list-incr*:
  assumes $a \in A$ $x \in$ *length-sum-set* (*card* A) n
  shows α (*list-incr* (*idx* a) x) $= a \oplus \alpha\ x$
⟨*proof*⟩

lemma *sumset-iterated-enum*:
  defines $r \equiv$ *card* A
  shows *sumset-iterated* A $n = \alpha$ ‘ *length-sum-set* r n
⟨*proof*⟩

2.7 Lemma 2.7 in Gowers's notes

The following lemma corresponds to a key fact about the cardinality of the set of all sums of n many elements of A, stated before Gowers's Lemma 2.7.

lemma *card-sumset-iterated-length-sum-set-useful*:
  defines $r \equiv$ *card* A
  shows *card*(*sumset-iterated* A n) $=$ *card* (*length-sum-set* r $n \cap \{x.$ *useful* $x\}$)
    (**is** *card* ?L = *card* ?R)
⟨*proof*⟩

The following lemma corresponds to Lemma 2.7 in Gowers's notes.

lemma *useless-leq-useless*:
  defines $r \equiv$ *card* A
  assumes *useless* x **and** $x \trianglelefteq y$ **and** *length* $x = r$
  shows *useless* y
⟨*proof*⟩

inductive-set *minimal-elements* **for** U
  where $[\![x \in U; \bigwedge y.\ y \in U \Longrightarrow \neg\ y \lhd x]\!] \Longrightarrow x \in$ *minimal-elements* U

lemma *pointwise-less-imp-σ*:
  assumes $xs \lhd ys$ **shows** $\sigma\ xs < \sigma\ ys$
⟨*proof*⟩

lemma *wf-measure-σ*: *wf* (*inv-image less-than* σ)
  ⟨*proof*⟩

lemma *WFP*: *wfP* ($\lhd$)
  ⟨*proof*⟩

The following is a direct corollary of the above lemma, i.e. a corollary of Lemma 2.7 in Gowers's notes.

corollary *useless-iff*:
  assumes *length* $x =$ *card* A
  shows *useless* $x \longleftrightarrow (\exists x' \in$ *minimal-elements* (*Collect useless*). $x' \trianglelefteq x)$ (**is** -=?R)
⟨*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*: $\bigwedge x.\ x \in U \Longrightarrow \mathit{length}\ x = r$
  shows *finite* (*minimal-elements U*)
  ⟨*proof*⟩

2.9 Towards Lemma 2.9 in Gowers's notes

Increasing sequences

fun *augmentum* :: *nat list* ⇒ *nat list*
  where *augmentum* [] = []
  | *augmentum* (*n*#*ns*) = *n* # *map* ((+)*n*) (*augmentum ns*)

definition *dementum*:: *nat list* ⇒ *nat list*
  where *dementum xs* ≡ *xs* − (*0*#*xs*)

lemma *dementum-Nil* [*simp*]: *dementum* [] = []
  ⟨*proof*⟩

lemma *zero-notin-augmentum* [*simp*]: *0* ∉ *set ns* ⟹ *0* ∉ *set* (*augmentum ns*)
  ⟨*proof*⟩

lemma *length-augmentum* [*simp*]:*length* (*augmentum xs*) = *length xs*
  ⟨*proof*⟩

lemma *sorted-augmentum* [*simp*]: *0* ∉ *set ns* ⟹ *sorted* (*augmentum ns*)
  ⟨*proof*⟩

lemma *distinct-augmentum* [*simp*]: *0* ∉ *set ns* ⟹ *distinct* (*augmentum ns*)
  ⟨*proof*⟩

lemma *augmentum-subset-sum-list*: *set* (*augmentum ns*) ⊆ {..σ *ns*}
  ⟨*proof*⟩

lemma *sum-list-augmentum*: σ *ns* ∈ *set* (*augmentum ns*) ⟷ *length ns* > *0*
  ⟨*proof*⟩

lemma *length-dementum* [*simp*]: *length* (*dementum xs*) = *length xs*
  ⟨*proof*⟩

lemma *sorted-imp-pointwise*:
  assumes *sorted* (*xs*@[*n*])
  shows *0* # *xs* ⊴ *xs* @ [*n*]
  ⟨*proof*⟩

lemma *sum-list-dementum*:

assumes *sorted* $(xs@[n])$
shows σ $(dementum\ (xs@[n])) = n$
⟨*proof*⟩

lemma *augmentum-cancel*: *map* $((+)k)$ $(augmentum\ ns)$ $-$ $(k\ \#\ map\ ((+)k)$ $(augmentum\ ns)) = ns$
⟨*proof*⟩

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

lemma *dementum-nonzero*:
assumes *ns*: *sorted-wrt* $(<)$ *ns* **and** *0*: $0 \notin set\ ns$
shows $0 \notin set\ (dementum\ ns)$
⟨*proof*⟩

lemma *nth-augmentum* [*simp*]: $i < length\ ns \Longrightarrow augmentum\ ns!i = (\sum j{\le}i.\ ns!j)$
⟨*proof*⟩

lemma *augmentum-dementum* [*simp*]:
assumes $0 \notin set\ ns$ *sorted ns*
shows *augmentum* $(dementum\ ns) = ns$
⟨*proof*⟩

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*: $n \ge \sigma\ x'$ **and** *lenx'*: $length\ x' = r$
defines $S \equiv \{x.\ x' \trianglelefteq x \wedge \sigma\ x = n\}$
shows $card\ S = (n - \sigma\ x' + r - 1)\ choose\ (r-1)$
⟨*proof*⟩

2.10 Towards the main theorem

lemma *extend-tuple*:
assumes $\sigma\ xs \le n$ $length\ xs \ne 0$
obtains *ys* **where** $\sigma\ ys = n$ $xs \trianglelefteq ys$
⟨*proof*⟩

lemma *extend-preserving*:
assumes $\sigma\ xs \le n$ $length\ xs > 1$ $i < length\ xs$
obtains *ys* **where** $\sigma\ ys = n$ $xs \trianglelefteq ys$ $ys!i = xs!i$
⟨*proof*⟩

The proof of the main theorem will make use of the inclusion-exclusion formula, in addition to the previously shown results.

theorem *Khovanskii*:
  assumes *card* $A > 1$
  defines $f \equiv \lambda n.\ card(sumset\text{-}iterated\ A\ n)$
  obtains $N\ p$ **where** *real-polynomial-function* $p \bigwedge n.\ n \geq N \Longrightarrow real\ (f\ n) = p\ (real\ n)$
$\langle proof \rangle$

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.