Cauchy's Mean Theorem and the Cauchy-Schwarz Inequality

Benjamin Porter

March 19, 2025

Contents

Abstract

This document presents the mechanised proofs of two popular theorems attributed to Augustin Louis Cauchy - Cauchy's Mean Theorem and the Cauchy-Schwarz Inequality.

Chapter 1

Cauchy's Mean Theorem

theory *CauchysMeanTheorem*
imports *Complex-Main*
begin

1.1 Abstract

The following document presents a proof of Cauchy's Mean theorem formalised in the Isabelle/Isar theorem proving system.

Theorem: For any collection of positive real numbers the geometric mean is always less than or equal to the arithmetic mean. In mathematical terms:

$$\sqrt[n]{x_1 x_2 \ldots x_n} \leq \frac{x_1 + \ldots + x_n}{n}$$

We will use the term *mean* to denote the arithmetic mean and *gmean* to denote the geometric mean.

Informal Proof:

This proof is based on the proof presented in [1]. First we need an auxiliary lemma (the proof of which is presented formally below) that states:

Given two pairs of numbers of equal sum, the pair with the greater product is the pair with the least difference. Using this lemma we now present the proof -

Given any collection C of positive numbers with mean M and product P and with some element not equal to M we can choose two elements from the collection, a and b where $a > M$ and $b < M$. Remove these elements from the collection and replace them with two new elements, a' and b' such that $a' = M$ and $a' + b' = a + b$. This new collection C' now has a greater product P' but equal mean with respect to C. We can continue in this fashion until we have a collection C_n such that $P_n > P$ and $M_n = M$, but C_n has all its elements equal to M and thus $P_n = M^n$. Using the definition of geometric and arithmetic means above we can see that for any collection of positive elements E it is always true that gmean E $\leq$ mean E. QED.

[1] Dorrie, H. "100 Great Problems of Elementary Mathematics." 1965, Dover.

1.2 Formal proof

1.2.1 Collection sum and product

The finite collections of numbers will be modelled as lists. We then define sum and product operations over these lists.

Sum and product definitions

notation (*input*) *sum-list* (‹∑:-› [*999*] *998*)

notation (*input*) *prod-list* (‹∏:-› [*999*] *998*)

Properties of sum and product

We now present some useful properties of sum and product over collections.

These lemmas just state that if all the elements in a collection C are less (greater than) than some value m, then the sum will less than (greater than) $m * length(C)$.

```
lemma sum-list-mono-lt [rule-format]:
  fixes xs::real list
  shows xs ≠ [] ∧ (∀ x∈ set xs. x < m)
        ⟶ ((∑:xs) < (m∗(real (length xs))))
proof (induct xs)
  case Nil show ?case by simp
next
  case (Cons y ys)
  {
    assume ant: y#ys ≠ [] ∧ (∀ x∈set(y#ys). x < m)
    hence ylm: y < m by simp
    have ∑:(y#ys) < m ∗ real (length (y#ys))
    proof cases
      assume ys ≠ []
      moreover with ant have ∀ x∈set ys. x < m by simp
      moreover with calculation Cons have ∑:ys < m∗real (length ys) by simp
      hence ∑:ys + y < m∗real(length ys) + y by simp
     with ylm have ∑:(y#ys) < m∗(real(length ys) + 1) by(simp add:field-simps)
      then have ∑:(y#ys) < m∗(real(length ys + 1))
        by (simp add: algebra-simps)
      hence ∑:(y#ys) < m∗(real (length(y#ys))) by simp
      thus ?thesis .
    next
```

assume ¬ (*ys* ≠ [])
hence *ys* = [] **by** *simp*
with *ylm* **show** *?thesis* **by** *simp*
qed
}
thus *?case* **by** *simp*
qed

lemma *sum-list-mono-gt* [*rule-format*]:
fixes *xs*::*real list*
shows *xs* ≠ [] ∧ (∀ *x*∈*set xs*. *x* > *m*)
⟶ ((∑:*xs*) > (*m*∗(*real* (*length xs*))))

proof omitted

qed

If a is in C then the sum of the collection D where D is C with a removed is the sum of C minus a.

lemma *sum-list-rmv1*:
a ∈ *set xs* ⟹ ∑:(*remove1 a xs*) = ∑:*xs* − (*a* :: ′*a* :: *ab-group-add*)
by (*induct xs*) *auto*

A handy addition and division distribution law over collection sums.

lemma *list-sum-distrib-aux*:
shows (∑:*xs*/(*n* :: ′*a* :: *archimedean-field*) + ∑:*xs*) = (*1* + (*1*/*n*)) ∗ ∑:*xs*
proof (*induct xs*)
case *Nil* **show** *?case* **by** *simp*
next
case (*Cons x xs*)
show *?case*
proof −
have
∑:(*x*#*xs*)/*n* = *x*/*n* + ∑:*xs*/*n*
by (*simp add*: *add-divide-distrib*)
also with *Cons* **have**
... = *x*/*n* + (*1*+*1*/*n*)∗∑:*xs* − ∑:*xs*
by *simp*
finally have
∑:(*x*#*xs*) / *n* + ∑:(*x*#*xs*) = *x*/*n* + (*1*+*1*/*n*)∗∑:*xs* − ∑:*xs* + ∑:(*x*#*xs*)
by *simp*
also have
... = *x*/*n* + (*1*+(*1*/*n*)− *1*)∗∑:*xs* + ∑:(*x*#*xs*)
by (*subst mult-1-left* [*symmetric*, *of* ∑:*xs*]) (*simp add*: *field-simps*)
also have
... = *x*/*n* + (*1*/*n*)∗∑:*xs* + ∑:(*x*#*xs*)
by *simp*
also have
... = (*1*/*n*)∗∑:(*x*#*xs*) + *1*∗∑:(*x*#*xs*) **by**(*simp add*: *divide-simps*)

```
    finally show ?thesis by (simp add: field-simps)
  qed
qed

lemma remove1-retains-prod:
  fixes a and xs::'a :: comm-ring-1 list
  shows a ∈ set xs ⟶ ∏:xs = ∏:(remove1 a xs) * a
  (is ?P xs)
proof (induct xs)
  case Nil
  show ?case by simp
next
  case (Cons aa list)
  assume plist: ?P list
  show ?P (aa#list)
  proof
    assume aml: a ∈ set(aa#list)
    show ∏:(aa # list) = ∏:remove1 a (aa # list) * a
    proof (cases)
      assume aeq: a = aa
      hence
        remove1 a (aa#list) = list
        by simp
      hence
        ∏:(remove1 a (aa#list)) = ∏:list
        by simp
      moreover with aeq have
        ∏:(aa#list) = ∏:list * a
        by simp
      ultimately show
        ∏:(aa#list) = ∏:remove1 a (aa # list) * a
        by simp
    next
      assume naeq: a ≠ aa
      with aml have aml2: a ∈ set list by simp
      from naeq have
        remove1 a (aa#list) = aa#(remove1 a list)
        by simp
      moreover hence
        ∏:(remove1 a (aa#list)) = aa * ∏:(remove1 a list)
        by simp
      moreover from aml2 plist have
        ∏:list = ∏:(remove1 a list) * a
        by simp
      ultimately show
        ∏:(aa#list) = ∏:remove1 a (aa # list) * a
        by simp
    qed
  qed
```

qed

The final lemma of this section states that if all elements are positive and non-zero then the product of these elements is also positive and non-zero.

lemma *el-gt0-imp-prod-gt0* [*rule-format*]:
fixes *xs*::'*a* :: *archimedean-field list*
shows $\forall\, y.\ y \in set\ xs \longrightarrow y > 0 \Longrightarrow \prod{:}xs > 0$
proof (*induct xs*)
case *Nil* **show** *?case* **by** *simp*
next
case (*Cons a xs*)
have *exp*: $\prod{:}(a\#xs) = \prod{:}xs * a$ **by** *simp*
with *Cons* **have** $a > 0$ **by** *simp*
with *exp Cons* **show** *?case* **by** *simp*
qed

1.2.2 Auxiliary lemma

This section presents a proof of the auxiliary lemma required for this theorem.

lemma *prod-exp*:
fixes *x*::*real*
shows $4*(x*y) = (x+y)\hat{\ }2 - (x-y)\hat{\ }2$
by (*simp add*: *power2-diff power2-sum*)

lemma *abs-less-imp-sq-less* [*rule-format*]:
fixes *x*::*real* **and** *y*::*real* **and** *z*::*real* **and** *w*::*real*
assumes *diff*: $abs\ (x-y) < abs\ (z-w)$
shows $(x-y)\hat{\ }2 < (z-w)\hat{\ }2$
proof *cases*
assume $x=y$
hence $abs\ (x-y) = 0$ **by** *simp*
moreover with *diff* **have** $abs(z-w) > 0$ **by** *simp*
hence $(z-w)\hat{\ }2 > 0$ **by** *simp*
ultimately show *?thesis* **by** *auto*
next
assume $x \neq y$
hence $abs\ (x - y) > 0$ **by** *simp*
with *diff* **have** $(abs\ (x-y))\hat{\ }2 < (abs\ (z-w))\hat{\ }2$
by − (*drule power-strict-mono* [**where** $a=abs\ (x-y)$ **and** $n=2$ **and** $b=abs\ (z-w)$], *auto*)
thus *?thesis* **by** *simp*
qed

The required lemma (phrased slightly differently than in the informal proof.) Here we show that for any two pairs of numbers with equal sums the pair with the least difference has the greater product.

lemma *le-diff-imp-gt-prod* [*rule-format*]:

```
  fixes x::real and y::real and z::real and w::real
  assumes diff: abs (x−y) < abs (z−w) and sum: x+y = z+w
  shows x∗y > z∗w
proof −
  from sum have (x+y)^2 = (z+w)^2 by simp
  moreover from diff have (x−y)^2 < (z−w)^2 by (rule abs-less-imp-sq-less)
  ultimately have (x+y)^2 − (x−y)^2 > (z+w)^2 − (z−w)^2 by auto
  thus x∗y > z∗w by (simp only: prod-exp [symmetric])
qed
```

1.2.3 Mean and GMean

Now we introduce definitions and properties of arithmetic and geometric means over collections of real numbers.

Definitions

Arithmetic mean

```
definition
  mean :: (real list)⇒real where
  mean s = (∑:s / real (length s))
```

Geometric mean

```
definition
  gmean :: (real list)⇒real where
  gmean s = root (length s) (∏:s)
```

Properties

Here we present some trivial properties of *mean* and *gmean*.

```
lemma list-sum-mean:
  fixes xs::real list
  shows ∑:xs = ((mean xs) ∗ (real (length xs)))
  by (induct xs) (auto simp: mean-def)

lemma list-mean-eq-iff:
  fixes one::real list and two::real list
  assumes
    se: ( ∑:one = ∑:two ) and
    le: (length one = length two)
  shows (mean one = mean two)
proof −
  from se le have
    (∑:one / real (length one)) = (∑:two / real (length two))
    by auto
  thus ?thesis unfolding mean-def .
qed
```

```
lemma list-gmean-gt-iff:
  fixes one::real list and two::real list
  assumes
    gz1: ∏:one > 0 and gz2: ∏:two > 0 and
    ne1: one ≠ [] and ne2: two ≠ [] and
    pe: (∏:one > ∏:two) and
    le: (length one = length two)
  shows (gmean one > gmean two)
  unfolding gmean-def
  using le ne2 pe by simp
```

This slightly more complicated lemma shows that for every non-empty collection with mean M, adding another element a where $a = M$ results in a new list with the same mean M.

```
lemma list-mean-cons [rule-format]:
  fixes xs::real list
  shows xs ≠ [] ⟶ mean ((mean xs)#xs) = mean xs
proof
  assume lne: xs ≠ []
  obtain len where ld: len = real (length xs) by simp
  with lne have lgt0: len > 0 by simp
  hence lnez: len ≠ 0 by simp
  from lgt0 have l1nez: len + 1 ≠ 0 by simp
  from ld have mean: mean xs = ∑:xs / len unfolding mean-def by simp
  with ld of-nat-add of-int-1 mean-def
  have mean ((mean xs)#xs) = (∑:xs/len + ∑:xs) / (1+len)
    by simp
  also from list-sum-distrib-aux[of xs] have
    ... = (1 + (1/len))*∑:xs / (1+len) by simp
  also have
    ... = (len + 1)*∑:xs / (len * (1+len))
    by (smt (verit, best) lnez add-divide-distrib divide-divide-eq-left
        nonzero-divide-mult-cancel-left times-divide-eq-left)
  also from l1nez have ... = ∑:xs / len
    by (simp add: add.commute)
  finally show mean ((mean xs)#xs) = mean xs by (simp add: mean)
qed
```

For a non-empty collection with positive mean, if we add a positive number to the collection then the mean remains positive.

```
lemma mean-gt-0:
  assumes xs ≠ [] 0 < x and mgt0: 0 < mean xs
  shows 0 < mean (x # xs)
proof -
  have lxsgt0: length xs ≠ 0
    using assms by simp
  from mgt0 have xsgt0: 0 < ∑:xs
    by (simp add: assms list-sum-mean)
```

```
  with ‹x>0› have ∑:(x#xs) > 0 by simp
  thus ?thesis
    using mean-def by force
qed
```

1.2.4 *list-neq*, *list-eq*

This section presents a useful formalisation of the act of removing all the elements from a collection that are equal (not equal) to a particular value. We use this to extract all the non-mean elements from a collection as is required by the proof.

Definitions

list-neq and *list-eq* just extract elements from a collection that are not equal (or equal) to some value.

```
abbreviation
  list-neq :: ('a list) ⇒ 'a ⇒ ('a list) where
  list-neq xs el == filter (λx. x≠el) xs

abbreviation
  list-eq :: ('a list) ⇒ 'a ⇒ ('a list) where
  list-eq xs el == filter (λx. x=el) xs
```

Properties

This lemma just proves a required fact about *list-neq*, *remove1* and *length*.

```
lemma list-neq-remove1 [rule-format]:
  shows a≠m ∧ a ∈ set xs
  ⟶ length (list-neq (remove1 a xs) m) < length (list-neq xs m)
  (is ?A xs ⟶ ?B xs is ?P xs)
proof (induct xs)
  case Nil show ?case by simp
next
  case (Cons x xs)
  note ‹?P xs›
  {
    assume a: ?A (x#xs)
    hence
      a-ne-m: a≠m and
      a-mem-x-xs: a ∈ set(x#xs)
      by auto
    have b: ?B (x#xs)
    proof cases
      assume xs = []
      with a-ne-m a-mem-x-xs show ?thesis
        by simp
```

```
    next
      assume xs-ne: xs ≠ []
      with a-ne-m a-mem-x-xs show ?thesis
      proof cases
        assume a=x with a-ne-m show ?thesis by simp
      next
        assume a-ne-x: a≠x
        with a-mem-x-xs xs-ne a-ne-m Cons have
          length (list-neq (remove1 a xs) m) < length (list-neq xs m)
          by simp
        then show ?thesis
          by (simp add: a-ne-x)
      qed
    qed
  }
  thus ?P (x#xs) by simp
qed
```

We now prove some facts about *list-eq*, *list-neq*, length, sum and product.

```
lemma list-eq-sum [simp]:
  fixes xs::real list
  shows ∑:(list-eq xs m) = (m * (real (length (list-eq xs m))))
  by (induct xs) (auto simp:field-simps)

lemma list-eq-prod [simp]:
  fixes xs::real list
  shows ∏:(list-eq xs m) = (m ^ (length (list-eq xs m)))
  by (induct xs) auto

lemma sum-list-split:
  fixes xs::real list
  shows ∑:xs = (∑:(list-neq xs m) + ∑:(list-eq xs m))
  by (induct xs) auto

lemma prod-list-split:
  fixes xs::real list
  shows ∏:xs = (∏:(list-neq xs m) * ∏:(list-eq xs m))
  by (induct xs) auto

lemma sum-list-length-split:
  fixes xs::real list
  shows length xs = length (list-neq xs m) + length (list-eq xs m)
  by (induct xs) auto
```

1.2.5 Element selection

We now show that given after extracting all the elements not equal to the mean there exists one that is greater then (or less than) the mean.

```
lemma pick-one-gt:
```

fixes *xs*::*real list* **and** *m*::*real*
defines *m*: $m \equiv (mean\ xs)$ **and** *neq*: $noteq \equiv list\text{-}neq\ xs\ m$
assumes *asum*: $noteq \neq []$
shows $\exists e.\ e \in set\ noteq \wedge e > m$
proof (*rule ccontr*)
let $?m = (mean\ xs)$
let $?neq = list\text{-}neq\ xs\ ?m$
let $?eq = list\text{-}eq\ xs\ ?m$
from *list-eq-sum* **have** $(\sum:?eq) = ?m * (real\ (length\ ?eq))$ **by** *simp*
from *asum* **have** *neq-ne*: $?neq \neq []$ **unfolding** *m neq* **.**
assume *not-el*: $\neg(\exists e.\ e \in set\ noteq \wedge m < e)$
hence *not-el-exp*: $\neg(\exists e.\ e \in set\ ?neq \wedge ?m < e)$ **unfolding** *m neq* **.**
hence $\forall e.\ \neg(e \in set\ ?neq) \vee \neg(e > ?m)$ **by** *simp*
hence $\forall e.\ e \in set\ ?neq \longrightarrow \neg(e > ?m)$ **by** *blast*
hence $\forall e.\ e \in set\ ?neq \longrightarrow e \leq ?m$ **by** (*simp add*: *linorder-not-less*)
hence $\forall e.\ e \in set\ ?neq \longrightarrow e < ?m$ **by** (*simp add*:*order-le-less*)
with *assms sum-list-mono-lt* **have** $(\sum:?neq) < ?m * (real\ (length\ ?neq))$ **by** *blast*
hence $(\sum:?neq) + (\sum:?eq) < ?m * (real\ (length\ ?neq)) + (\sum:?eq)$ **by** *simp*
also have $\ldots = (?m * ((real\ (length\ ?neq) + (real\ (length\ ?eq)))))$
by (*simp add*:*field-simps*)
also have $\ldots = (?m * (real\ (length\ xs)))$
by (*metis of-nat-add sum-list-length-split*)
also have $\ldots = \sum:xs$
by (*simp add*: *list-sum-mean* [*symmetric*])
finally show *False*
by (*metis nless-le sum-list-split*)
qed

lemma *pick-one-lt*:
fixes *xs*::*real list* **and** *m*::*real*
defines *m*: $m \equiv (mean\ xs)$ **and** *neq*: $noteq \equiv list\text{-}neq\ xs\ m$
assumes *asum*: $noteq \neq []$
shows $\exists e.\ e \in set\ noteq \wedge e < m$
proof (*rule ccontr*) — reductio ad absurdum
let $?m = (mean\ xs)$
let $?neq = list\text{-}neq\ xs\ ?m$
let $?eq = list\text{-}eq\ xs\ ?m$
from *list-eq-sum* **have** $(\sum:?eq) = ?m * (real\ (length\ ?eq))$ **by** *simp*
from *asum* **have** *neq-ne*: $?neq \neq []$ **unfolding** *m neq* **.**
assume *not-el*: $\neg(\exists e.\ e \in set\ noteq \wedge m > e)$
hence *not-el-exp*: $\neg(\exists e.\ e \in set\ ?neq \wedge ?m > e)$ **unfolding** *m neq* **.**
hence $\forall e.\ \neg(e \in set\ ?neq) \vee \neg(e < ?m)$ **by** *simp*
hence $\forall e.\ e \in set\ ?neq \longrightarrow \neg(e < ?m)$ **by** *blast*
hence $\forall e.\ e \in set\ ?neq \longrightarrow e \geq ?m$ **by** (*simp add*: *linorder-not-less*)
hence $\forall e.\ e \in set\ ?neq \longrightarrow e > ?m$ **by** (*auto simp*: *order-le-less*)
with *assms sum-list-mono-gt* **have** $(\sum:?neq) > ?m * (real\ (length\ ?neq))$ **by** *blast*
hence

    (∑:?neq) + (∑:?eq) > ?m ∗ (real (length ?neq)) + (∑:?eq) by simp
  also have
    (?m ∗ (real (length ?neq)) + (∑:?eq)) =
     (?m ∗ (real (length ?neq)) + (?m ∗ (real (length ?eq))))
    by simp
  also have ... = (?m ∗ ((real (length ?neq) + (real (length ?eq)))))
    by (simp add:field-simps)
  also have ... = (?m ∗ (real (length xs)))
    by (metis of-nat-add sum-list-length-split)
  also have ... = ∑:xs
    by (simp add: list-sum-mean [symmetric])
  finally show False
    by (metis less-irrefl sum-list-split)
qed

1.2.6 Abstract properties

In order to maintain some comprehension of the following proofs we now introduce some properties of collections.

Definitions

het: The heterogeneity of a collection is the number of elements not equal to its mean. A heterogeneity of zero implies the all the elements in the collection are the same (i.e. homogeneous).

definition
  het :: *real list* ⇒ *nat* **where**
  het l = *length* (*list-neq l* (*mean l*))

lemma *het-gt-0-imp-noteq-ne*: *het l* > *0* ⟹ *list-neq l* (*mean l*) ≠ []
  unfolding *het-def* **by** *simp*

lemma *het-gt-0I*:
  assumes *a* ∈ *set xs b* ∈ *set xs a* ≠ *b*
  shows *het xs* > *0*
  unfolding *het-def*
  by (*metis* (*mono-tags*, *lifting*) *assms filter-empty-conv length-greater-0-conv*)

γ−eq: Two lists are *γ*-equivalent if and only if they both have the same number of elements and the same arithmetic means.

definition
  γ-eq :: ((*real list*)∗(*real list*)) ⇒ *bool* **where**
  γ-eq a ⟷ *mean* (*fst a*) = *mean* (*snd a*) ∧ *length* (*fst a*) = *length* (*snd a*)

γ-eq is transitive and symmetric.

lemma *γ-eq-sym*: *γ-eq* (*a*,*b*) = *γ-eq* (*b*,*a*)
  unfolding *γ-eq-def* **by** *auto*

```
lemma γ-eq-trans:
  γ-eq (x,y) ⟹ γ-eq (y,z) ⟹ γ-eq (x,z)
  unfolding γ-eq-def by simp
```

pos: A list is positive if all its elements are greater than 0.

```
definition
  pos :: real list ⇒ bool where
  pos l ⟷ (if l=[] then False else ∀ e. e ∈ set l ⟶ e > 0)

lemma pos-empty [simp]: pos [] = False unfolding pos-def by simp
lemma pos-single [simp]: pos [x] = (x > 0) unfolding pos-def by simp
lemma pos-imp-ne: pos xs ⟹ xs≠[] unfolding pos-def by auto

lemma pos-cons [simp]:
  xs ≠ [] ⟹ pos (x#xs) = (if (x>0) then pos xs else False)
  by (auto simp: pos-def)
```

Properties

Here we prove some non-trivial properties of the abstract properties.

Two lemmas regarding *pos*. The first states the removing an element from a positive collection (of more than 1 element) results in a positive collection. The second asserts that the mean of a positive collection is positive.

```
lemma pos-imp-rmv-pos:
  assumes (remove1 a xs)≠[] pos xs shows pos (remove1 a xs)
  by (metis assms notin-set-remove1 pos-def)

lemma pos-mean: pos xs ⟹ mean xs > 0
proof (induct xs)
  case Nil thus ?case by(simp add: pos-def)
next
  case (Cons x xs)
  then show ?case
  proof cases
    assume xse: xs = []
    thus ?thesis
      using Cons.prems mean-def by auto
  next
    assume xsne: xs ≠ []
    show ?thesis
      by (meson Cons.hyps Cons.prems mean-gt-0 pos-cons xsne)
  qed
qed
```

We now show that homogeneity of a non-empty collection x implies that its product is equal to (*mean x*)^(*length x*).

```
lemma prod-list-het0:
  shows x≠[] ∧ het x = 0 ⟹ ∏:x = (mean x) ^(length x)
proof −
  assume x≠[] ∧ het x = 0
  hence xne: x≠[] and hetx: het x = 0 by auto
  from hetx have lz: length (list-neq x (mean x)) = 0 unfolding het-def .
  hence ∏:(list-neq x (mean x)) = 1 by simp
  with prod-list-split have ∏:x = ∏:(list-eq x (mean x))
    by (metis mult-1)
  also have ... = (mean x) ^(length (list-eq x (mean x))) by simp
  finally have ∏:x = (mean x) ^(length x)
    by (metis add-0 lz sum-list-length-split)
  thus ?thesis by simp
qed
```

Furthermore we present an important result - that a homogeneous collection has equal geometric and arithmetic means.

```
lemma het-base:
  assumes pos x het x = 0
  shows gmean x = mean x
proof −
  have root (length x) (∏:x) = root (length x) ((mean x)^(length x))
    by (simp add: assms pos-imp-ne prod-list-het0)
  also have ... = mean x
    by (simp add: ‹pos x› order.order-iff-strict pos-imp-ne pos-mean
        real-root-power-cancel)
  finally show gmean x = mean x unfolding gmean-def .
qed
```

1.2.7 Existence of a new collection

We now present the largest and most important proof in this document. Given any positive and non-homogeneous collection of real numbers there exists a new collection that is γ-equivalent, positive, has a strictly lower heterogeneity and a greater geometric mean.

```
lemma new-list-gt-gmean:
  fixes xs :: real list and m :: real
  and neq and eq
  defines
    m: m ≡ mean xs and
    neq: noteq ≡ list-neq xs m and
    eq: eq ≡ list-eq xs m
  assumes pos-xs: pos xs and het-gt-0: het xs > 0
  shows
  ∃ xs'. gmean xs' > gmean xs ∧ γ-eq (xs',xs) ∧
        het xs' < het xs ∧ pos xs'
proof −
  from pos-xs pos-imp-ne have
```

pos-els: $\forall\, y.\ y \in$ *set xs* $\longrightarrow y > 0$ **by** (*unfold pos-def*, *simp*)
with *el-gt0-imp-prod-gt0* [*of xs*] **have** *pos-asm*: $\prod$:*xs* > 0 **by** *simp*
from *neq het-gt-0 het-gt-0-imp-noteq-ne m* **have**
neqne: *noteq* $\neq$ [] **by** *simp*

Pick two elements from xs, one greater than m, one less than m.

from *assms pick-one-gt neqne* **obtain** α **where**
α-*def*: $\alpha \in$ *set noteq* $\wedge\ \alpha > m$ **unfolding** *neq m* **by** *auto*
from *assms pick-one-lt neqne* **obtain** β **where**
β-*def*: $\beta \in$ *set noteq* $\wedge\ \beta < m$ **unfolding** *neq m* **by** *auto*
from α-*def* β-*def* **have** α-*gt*: $\alpha > m$ **and** β-*lt*: $\beta < m$ **by** *auto*
from α-*def* β-*def* **have** *el-neq*: $\beta \neq \alpha$ **by** *simp*
from *neqne neq* **have** *xsne*: *xs* $\neq$ [] **by** *auto*

from β-*def* **have** β-*mem*: $\beta \in$ *set xs* **by** (*auto simp*: *neq*)
from α-*def* **have** α-*mem*: $\alpha \in$ *set xs* **by** (*auto simp*: *neq*)

from *pos-xs pos-def xsne* α-*mem* β-*mem* α-*def* β-*def* **have**
α-*pos*: $\alpha > 0$ **and** β-*pos*: $\beta > 0$ **by** *auto*

— remove these elements from xs, and insert two new elements
obtain *left-over* **where** *lo*: *left-over* = (*remove1* β (*remove1* α *xs*)) **by** *simp*
obtain *b* **where** *bdef*: $m + b = \alpha + \beta$
by (*drule meta-spec* [*of* - $\alpha + \beta - m$], *simp*)

from *m pos-xs pos-def pos-mean* **have** *m-pos*: $m > 0$ **by** *simp*
with *bdef* α-*pos* β-*pos* α-*gt* β-*lt* **have** *b-pos*: $b > 0$ **by** *simp*

obtain *new-list* **where** *nl*: *new-list* = *m#b#*(*left-over*) **by** *auto*

from *el-neq* β-*mem* α-*mem* **have** $\beta \in$ *set xs* $\wedge\ \alpha \in$ *set xs* $\wedge\ \beta \neq \alpha$ **by** *simp*
have *mem* : $\alpha \in$ *set*(*remove1* β *xs*) $\wedge\ \beta \in$ *set*(*remove1* α *xs*) $\wedge$ *remove1* α *xs* $\neq$ [] $\wedge$ (*remove1* β *xs*) $\neq$ []
by (*metis* α-*mem* β-*mem el-neq empty-iff in-set-remove1 list.set*(*1*))
— prove that new list is positive
from *nl* **have** *nl-pos*: *pos new-list*
by (*metis b-pos lo m-pos mem pos-cons pos-imp-rmv-pos pos-single pos-xs*)

— now show that the new list has the same mean as the old list
with *mem nl lo bdef* α-*mem* β-*mem*
have *s-eq-s*: $\sum$:*new-list* = $\sum$:*xs*
by (*simp add*: *sum-list-rmv1*)
then have *eq-mean*: *mean new-list* = *mean xs*
by (*metis One-nat-def Suc-pred* α-*mem length-Cons length-pos-if-in-set length-remove1 list-mean-eq-iff lo mem nl*)

— finally show that the new list has a greater gmean than the old list
have *gt-gmean*: *gmean new-list* > *gmean xs*
proof −

have *mb-gt-gt*: $m{*}b > \alpha{*}\beta$
using *α-gt β-lt bdef le-diff-imp-gt-prod* **by** *force*
moreover from *nl* **have**
$\prod$:*new-list* = $\prod$:*left-over* ∗ (*m*∗*b*) **by** *auto*
moreover
from *lo α-mem β-mem mem remove1-retains-prod*[**where** *'a* = *real*] **have**
xsprod: $\prod$:*xs* = $\prod$:*left-over* ∗ (α∗β) **by** *auto*
moreover from *nl* **have**
nlne: *new-list* ≠ [] **by** *simp*
moreover from *pos-asm lo* **have**
$\prod$:*left-over* > *0*
using *α-pos β-pos mult-pos-pos xsprod zero-less-mult-pos2* **by** *auto*
ultimately show *gmean new-list* > *gmean xs*
using *s-eq-s eq-mean list-gmean-gt-iff list-sum-mean m*
m-pos pos-asm xsne **by** *force*
qed

— auxiliary info
from *β-lt* **have** *β-ne-m*: β ≠ *m* **by** *simp*
from *mem* **have**
β-mem-rmv-α: β ∈ *set* (*remove1* α *xs*) **and** *rmv-α-ne*: (*remove1* α *xs*) ≠ [] **by** *auto*

from *α-def* **have** *α-ne-m*: α ≠ *m* **by** *simp*

— now show that new list is more homogeneous
have *lt-het*: *het new-list* < *het xs*
proof *cases*
assume *bm*: *b*=*m*
with *het-def* **have**
het new-list = *length* (*list-neq left-over m*)
using *assms*(*1*) *eq-mean nl* **by** *auto*
also have
... < *length* (*list-neq* (*remove1* α *xs*) *m*)
by (*metis β-ne-m list-neq-remove1 lo mem*)
also have ... < *length* (*list-neq xs m*)
by (*metis α-mem α-ne-m list-neq-remove1*)
also have ... = *het xs*
using *het-def m* **by** *presburger*
finally show *het new-list* < *het xs* **.**
next
assume *bnm*: *b*≠*m*
with *het-def* **have**
het new-list = *length* (*b*#(*list-neq left-over m*))
using *eq-mean m nl* **by** *force*
also have ... = *1* + *length* (*list-neq* (*remove1* β (*remove1* α *xs*)) *m*)
using *lo* **by** *auto*
also have ... < *1* + *length* (*list-neq* (*remove1* α *xs*) *m*)
by (*metis β-ne-m add-strict-left-mono list-neq-remove1 mem*)

```
    finally have het new-list ≤ length (list-neq (remove1 α xs) m)
      by simp
    also have ... < length (list-neq xs m)
      by (metis α-mem α-ne-m list-neq-remove1)
    also have ... = het xs
      using het-def m by presburger
    finally show ?thesis .
  qed
  then show ?thesis
     — thus thesis by existence of newlist
    using γ-eq-def eq-mean gt-gmean list-sum-mean nl-pos pos-mean s-eq-s
    by fastforce
qed
```

Furthermore we show that for all non-homogeneous positive collections there exists another collection that is γ-equivalent, positive, has a greater geometric mean *and* is homogeneous.

```
lemma existence-of-het0:
  shows p = het x ⟹ p > 0 ⟹ pos x ⟹
  (∃ y. gmean y > gmean x ∧ γ-eq (x,y) ∧ het y = 0 ∧ pos y)
proof (induct p arbitrary: x rule: nat-less-induct)
  case (1 n x)
  then have het x > 0 and pos x by auto
  with new-list-gt-gmean  obtain β where
    β-def: gmean β > gmean x ∧ γ-eq (x,β) ∧ het β < het x ∧ pos β
    using γ-eq-sym by blast
  then obtain b where bdef: b = het β by simp
  with 1 β-def have b < n by auto
  then show ?case
    by (smt (verit, best) 1.hyps β-def γ-eq-trans bdef not-gr-zero)
qed
```

1.2.8 Cauchy's Mean Theorem

We now present the final proof of the theorem. For any positive collection we show that its geometric mean is less than or equal to its arithmetic mean.

```
theorem CauchysMeanTheorem:
  fixes z::real list
  assumes pos z
  shows gmean z ≤ mean z
proof –
  from ‹pos z› have zne: z≠[] by (rule pos-imp-ne)
  show gmean z ≤ mean z
  proof cases
    assume het z = 0
    with ‹pos z› zne het-base have gmean z = mean z by simp
    thus ?thesis by simp
  next
```

```
    assume het z ≠ 0
    hence het z > 0 by simp
    moreover obtain k where k = het z by simp
    moreover with calculation ‹pos z› existence-of-het0 have
      ∃ y. gmean y > gmean z ∧ γ-eq (z,y) ∧ het y = 0 ∧ pos y by auto
    then obtain α where
      gmean α > gmean z ∧ γ-eq (z,α) ∧ het α = 0 ∧ pos α ..
    with het-base γ-eq-def pos-imp-ne have
      mean z = mean α and
      gmean α > gmean z and
      gmean α = mean α by auto
    hence gmean z < mean z by simp
    thus ?thesis by simp
  qed
qed
```

In the equality version we prove that the geometric mean is identical to the arithmetic mean iff the collection is homogeneous.

```
theorem CauchysMeanTheorem-Eq:
  fixes z::real list
  assumes pos z
  shows gmean z = mean z ⟷ het z = 0
proof
  assume het z = 0
  with het-base[of z] ‹pos z› show gmean z = mean z by auto
next
  assume eq: gmean z = mean z
  show het z = 0
  proof (rule ccontr)
    assume het z ≠ 0
    hence het z > 0 by auto
    moreover obtain k where k = het z by simp
    ultimately obtain α where
      gmean α > gmean z ∧ γ-eq (z,α) ∧ het α = 0 ∧ pos α
      using assms existence-of-het0 by blast
    with het-base γ-eq-def pos-imp-ne
    have mean z = mean α and gmean α > gmean z and gmean α = mean α
      by auto
    hence gmean z < mean z by simp
    thus False using eq by auto
  qed
qed

corollary CauchysMeanTheorem-Less:
  fixes z::real list
  assumes pos z and het z > 0
  shows gmean z < mean z
  by (metis CauchysMeanTheorem CauchysMeanTheorem-Eq assms nless-le)
```

end

Chapter 2

The Cauchy-Schwarz Inequality

theory *CauchySchwarz*
imports *Complex-Main*
begin

2.1 Abstract

The following document presents a formalised proof of the Cauchy-Schwarz Inequality for the specific case of R^n. The system used is Isabelle/Isar.

Theorem: Take V to be some vector space possessing a norm and inner product, then for all $a, b \in V$ the following inequality holds: $|a \cdot b| \leq \|a\| * \|b\|$. Specifically, in the Real case, the norm is the Euclidean length and the inner product is the standard dot product.

2.2 Formal Proof

2.2.1 Vector, Dot and Norm definitions.

This section presents definitions for a real vector type, a dot product function and a norm function.

Vector

We now define a vector type to be a tuple of (function, length). Where the function is of type *nat* ⇒ *real*. We also define some accessor functions and appropriate notation.

type-synonym *vector* = (*nat*⇒*real*) ∗ *nat*

```
definition
  ith :: vector ⇒ nat ⇒ real (‹((-)_)› [80,100] 100) where
  ith v i = fst v i
```

```
definition
  vlen :: vector ⇒ nat where
  vlen v = snd v
```

Now to access the second element of some vector v the syntax is v_2.

Dot and Norm

We now define the dot product and norm operations.

```
definition
  dot :: vector ⇒ vector ⇒ real (infixr ‹·› 60) where
  dot a b = (∑j∈{1..(vlen a)}. a_j*b_j)
```

```
definition
  norm :: vector ⇒ real                  (‹‖-‖› 100) where
  norm v = sqrt (∑j∈{1..(vlen v)}. v_j^2)
```

Another definition of the norm is $\|v\| = sqrt\ (v \cdot v)$. We show that our definition leads to this one.

```
lemma norm-dot: ‖v‖ = sqrt (v·v)
  using dot-def norm-def real-sq by presburger
```

A further important property is that the norm is never negative.

```
lemma norm-pos:
  ‖v‖ ≥ 0
  by (simp add: norm-def sum-nonneg)
```

We now prove an intermediary lemma regarding double summation.

```
lemma double-sum-aux:
  fixes f::nat ⇒ real
  shows
  (∑k∈{1..n}. (∑j∈{1..n}. f k * g j)) =
   (∑k∈{1..n}. (∑j∈{1..n}. (f k * g j + f j * g k) / 2))
proof −
  have
    2 * (∑k∈{1..n}. (∑j∈{1..n}. f k * g j)) =
    (∑k∈{1..n}. (∑j∈{1..n}. f k * g j)) +
    (∑k∈{1..n}. (∑j∈{1..n}. f k * g j))
    by simp
  also have
    ... =
    (∑k∈{1..n}. (∑j∈{1..n}. f k * g j)) +
    (∑k∈{1..n}. (∑j∈{1..n}. f j * g k))
    using sum.swap by force
```

```
  also have
    ... =
    (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j + f j * g k))
    by (auto simp add: sum.distrib)
  finally have
    2 * (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
    (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j + f j * g k)) .
  hence
    (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
     (∑ k∈{1..n}. (∑ j∈{1..n}. (f k * g j + f j * g k)))*(1/2)
    by auto
  also have
    ... =
     (∑ k∈{1..n}. (∑ j∈{1..n}. (f k * g j + f j * g k)*(1/2)))
    by (simp add: sum-distrib-left mult.commute)
  finally show ?thesis by (auto simp add: inverse-eq-divide)
qed
```

The final theorem can now be proven. It is a simple forward proof that uses properties of double summation and the preceding lemma.

```
theorem CauchySchwarzReal:
  fixes x::vector
  assumes vlen x = vlen y
  shows |x·y| ≤ ‖x‖*‖y‖
proof −
  have |x·y|^2 ≤ (‖x‖*‖y‖)^2
  proof −
```

We can rewrite the goal in the following form ...

```
    have (‖x‖*‖y‖)^2 − |x·y|^2 ≥ 0
    proof −
      obtain n where nx: n = vlen x by simp
      with ‹vlen x = vlen y› have ny: n = vlen y by simp
      {
```

Some preliminary simplification rules.

```
        have (∑ j∈{1..n}. x_j^2) ≥ 0 by (simp add: sum-nonneg)
        hence xp: (sqrt (∑ j∈{1..n}. x_j^2))^2 = (∑ j∈{1..n}. x_j^2)
          by (rule real-sqrt-pow2)

        have (∑ j∈{1..n}. y_j^2) ≥ 0 by (simp add: sum-nonneg)
        hence yp: (sqrt (∑ j∈{1..n}. y_j^2))^2 = (∑ j∈{1..n}. y_j^2)
          by (rule real-sqrt-pow2)
```

The main result of this section is that (‖x‖*‖y‖)^2 can be written as a double sum.

```
        have (‖x‖*‖y‖)^2 = ‖x‖^2 * ‖y‖^2
          by (simp add: real-sq-exp)
        also from nx ny have
          ... = (sqrt (∑ j∈{1..n}. x_j^2))^2 * (sqrt (∑ j∈{1..n}. y_j^2))^2
```

unfolding *norm-def* **by** *auto*
also from *xp yp* **have**
$\ldots = (\sum j \in \{1..n\}.\ x_j \hat{}2) * (\sum j \in \{1..n\}.\ y_j \hat{}2)$
by *simp*
also from *sum-product* **have**
$\ldots = (\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k \hat{}2) * (y_j \hat{}2)))$ **.**
finally have
$(\|x\| * \|y\|)\hat{}2 = (\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k \hat{}2) * (y_j \hat{}2)))$ **.**
}
moreover

We also show that $|x \cdot y|\hat{}2$ can be expressed as a double sum.

have $|x \cdot y|\hat{}2 = (\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k * y_k) * (x_j * y_j)))$
by (*metis* (*no-types*) *dot-def nx power2-abs real-sq sum-product*)

We now manipulate the double sum expressions to get the required inequality.

ultimately have
$(\|x\| * \|y\|)\hat{}2 - |x \cdot y|\hat{}2 =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k \hat{}2) * (y_j \hat{}2))) -$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k * y_k) * (x_j * y_j)))$
by *simp*
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ ((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2))/2)) -$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (x_k * y_k) * (x_j * y_j)))$
by (*simp only*: *double-sum-aux*)
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ ((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2))/2 - (x_k * y_k) * (x_j * y_j)))$
by (*auto simp add*: *sum-subtractf*)
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (inverse\ 2) * 2 *$
$(((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2)) * (1/2) - (x_k * y_k) * (x_j * y_j))))$
by *auto*
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (inverse\ 2) * (2 *$
$(((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2)) * (1/2) - (x_k * y_k) * (x_j * y_j)))))$
by (*simp only*: *mult.assoc*)
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (inverse\ 2) *$
$((((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2)) * 2 * (inverse\ 2) - 2 * (x_k * y_k) * (x_j * y_j)))))$
by (*auto simp add*: *distrib-right mult.assoc ac-simps*)
also have
$\ldots =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (inverse\ 2) *$
$((((x_k \hat{}2 * y_j \hat{}2) + (x_j \hat{}2 * y_k \hat{}2)) - 2 * (x_k * y_k) * (x_j * y_j)))))$

```
      unfolding mult.assoc by simp
    also have
      ... =
       (inverse 2)*(∑ k∈{1..n}. (∑ j∈{1..n}.
       (((x_k^2*y_j^2) + (x_j^2*y_k^2)) − 2*(x_k*y_k)*(x_j*y_j))))
      by (simp only: sum-distrib-left)
    also have
      ... =
       (inverse 2)*(∑ k∈{1..n}. (∑ j∈{1..n}. (x_k*y_j − x_j*y_k)^2))
      by (simp only: power2-diff real-sq-exp, auto simp add: ac-simps)
    also have ... ≥ 0
      by (simp add: sum-nonneg)
    finally show (‖x‖*‖y‖)^2 − |x·y|^2 ≥ 0 .
  qed
  thus ?thesis by simp
 qed
 moreover have 0 ≤ ‖x‖*‖y‖
  by (auto simp add: norm-pos)
 ultimately show ?thesis by (rule power2-le-imp-le)
qed

end
```