# Cauchy's Mean Theorem and the Cauchy-Schwarz Inequality

Benjamin Porter

December 14, 2021

# 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:

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

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* ($\sum$:- [*999*] *998*)

**notation** (*input*) *prod-list* ($\prod$:- [*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** $xs \neq [] \wedge (\forall x{\in}set\ xs.\ x > m)$
    $\longrightarrow ((\sum{:}xs) > (m*(real\ (length\ xs))))$

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*:
  $a \in set\ xs \Longrightarrow \sum{:}(remove1\ a\ xs) = \sum{:}xs - (a :: 'a :: ab\text{-}group\text{-}add)$
**by** (*induct xs*) *auto*

A handy addition and division distribution law over collection sums.

**lemma** *list-sum-distrib-aux*:
  **shows** $(\sum{:}xs/(n :: 'a :: archimedean\text{-}field) + \sum{:}xs) = (1 + (1/n)) * \sum{:}xs$
**proof** (*induct xs*)
  **case** *Nil* **show** *?case* **by** *simp*
**next**
  **case** (*Cons x xs*)
  **show** *?case*
  **proof** −
    **have**
      $\sum{:}(x\#xs)/n = x/n + \sum{:}xs/n$
      **by** (*simp add*: *add-divide-distrib*)
    **also with** *Cons* **have**
      $\ldots = x/n + (1{+}1/n)*\sum{:}xs - \sum{:}xs$
      **by** *simp*
    **finally have**
      $\sum{:}(x\#xs)\ /\ n + \sum{:}(x\#xs) = x/n + (1{+}1/n)*\sum{:}xs - \sum{:}xs + \sum{:}(x\#xs)$
      **by** *simp*
    **also have**
      $\ldots = x/n + (1{+}(1/n){-}\ 1)*\sum{:}xs + \sum{:}(x\#xs)$
      **by** (*subst mult-1-left* [*symmetric*, *of* $\sum{:}xs$]) (*simp add*: *field-simps*)
    **also have**
      $\ldots = x/n + (1/n)*\sum{:}xs + \sum{:}(x\#xs)$
      **by** *simp*
    **also have**
      $\ldots = (1/n)*\sum{:}(x\#xs) + 1*\sum{:}(x\#xs)$ **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** $\forall\, y.\ y : set\ xs \longrightarrow y > 0 \Longrightarrow \prod{:}xs > 0$
**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** $(x+y)\hat{}2 = (z+w)\hat{}2$ **by** *simp*
**moreover from** *diff* **have** $(x-y)\hat{}2 < (z-w)\hat{}2$ **by** (*rule abs-less-imp-sq-less*)
**ultimately have** $(x+y)\hat{}2 - (x-y)\hat{}2 > (z+w)\hat{}2 - (z-w)\hat{}2$ **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* = ($\sum$:*s* / *real* (*length s*))

*Geometric mean*

**definition**
*gmean* :: (*real list*)⇒*real* **where**
*gmean s* = *root* (*length s*) ($\prod$:*s*)

#### Properties

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

**lemma** *list-sum-mean*:
**fixes** *xs*::*real list*
**shows** $\sum$:*xs* = ((*mean xs*) * (*real* (*length xs*)))
**apply** (*induct-tac xs*)
**apply** *simp*
**apply** *clarsimp*
**apply** (*unfold mean-def*)
**apply** *clarsimp*
**done**

**lemma** *list-mean-eq-iff*:
**fixes** *one*::*real list* **and** *two*::*real list*
**assumes**
*se*: ( $\sum$:*one* = $\sum$:*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 with lnez have
    ... = (len + 1)*∑:xs / (len * (1+len))
    apply −
    apply (drule mult-divide-mult-cancel-left
      [symmetric, where c=len and a=(1 + 1 / len) * ∑:xs and b=1+len])
    apply (clarsimp simp:field-simps)
    done
  also from l1nez have ... = ∑:xs / len
    apply (subst mult.commute [where a=len])
    apply (drule mult-divide-mult-cancel-left
      [where c=len+1 and a=∑:xs and b=len])
    by (simp add: ac-simps ac-simps)
  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* [*rule-format*]:
  $xs \neq [] \wedge 0 < x \wedge 0 < (mean\ xs) \longrightarrow 0 < (mean\ (x\#xs))$
**proof**
  **assume** *a*: $xs \neq [] \wedge 0 < x \wedge 0 < mean\ xs$
  **hence** *xgt0*: $0 < x$ **and** *mgt0*: $0 < mean\ xs$ **by** *auto*
  **from** *a* **have** *lxsgt0*: $length\ xs \neq 0$ **by** *simp*
  **from** *mgt0* **have** *xsgt0*: $0 < \sum{:}xs$
  **proof** −
    **have** $mean\ xs = \sum{:}xs\ /\ real\ (length\ xs)$ **unfolding** *mean-def* **by** *simp*
    **hence** $\sum{:}xs = mean\ xs * real\ (length\ xs)$ **by** *simp*
    **moreover from** *lxsgt0* **have** $real\ (length\ xs) > 0$ **by** *simp*
    **moreover with** *calculation lxsgt0 mgt0* **show** *?thesis* **by** *auto*
  **qed**
  **with** *xgt0* **have** $\sum{:}(x\#xs) > 0$ **by** *simp*
  **thus** $0 < (mean\ (x\#xs))$
  **proof** −
    **assume** $0 < \sum{:}(x\#xs)$
    **moreover have** $real\ (length\ (x\#xs)) > 0$ **by** *simp*
    **ultimately show** *?thesis* **unfolding** *mean-def* **by** *simp*
  **qed**
**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* :: $('a\ list) \Rightarrow 'a \Rightarrow ('a\ list)$ **where**
  $list\text{-}neq\ xs\ el == filter\ (\lambda x.\ x \neq el)\ xs$

**abbreviation**
  *list-eq* :: $('a\ list) \Rightarrow 'a \Rightarrow ('a\ list)$ **where**
  $list\text{-}eq\ xs\ el == filter\ (\lambda x.\ x = el)\ xs$

#### 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
        apply (cases x=a)
        by auto
    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 have a-mem-xs: a : set xs by simp
        with xs-ne a-ne-m Cons have
          rel: length (list-neq (remove1 a xs) m) < length (list-neq xs m)
          by simp
        show ?thesis
        proof cases
          assume x-e-m: x=m
          with Cons xs-ne a-ne-m a-mem-xs show ?thesis by simp
        next
          assume x-ne-m: x≠m
          from a-ne-x have
            remove1 a (x#xs) = x#(remove1 a xs)
            by simp
          hence
            length (list-neq (remove1 a (x#xs)) m) =
             length (list-neq (x#(remove1 a xs)) m)
            by simp
          also with x-ne-m have
            ... = 1 + length (list-neq (remove1 a xs) m)
            by simp
          finally have
            length (list-neq (remove1 a (x#xs)) m) =
```

```
          1 + length (list-neq (remove1 a xs) m)
          by simp
        moreover with x-ne-m a-ne-x have
          length (list-neq (x#xs) m) =
          1 + length (list-neq xs m)
          by simp
        moreover with rel show ?thesis by simp
      qed
    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))))
apply (induct-tac xs)
apply simp
apply (simp add:field-simps)
done

lemma list-eq-prod [simp]:
  fixes xs::real list
  shows ∏:(list-eq xs m) = (m ^(length (list-eq xs m)))
apply (induct-tac xs)
apply simp
apply clarsimp
done

lemma sum-list-split:
  fixes xs::real list
  shows ∑:xs = (∑:(list-neq xs m) + ∑:(list-eq xs m))
apply (induct xs)
apply simp
apply clarsimp
done

lemma prod-list-split:
  fixes xs::real list
  shows ∏:xs = (∏:(list-neq xs m) ∗ ∏:(list-eq xs m))
apply (induct xs)
apply simp
apply clarsimp
done

lemma sum-list-length-split:
  fixes xs::real list
```

**shows** *length xs* = *length* (*list-neq xs m*) + *length* (*list-eq xs m*)
**apply** (*induct xs*)
**apply** *simp*+
**done**

### 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* ≡ (*mean xs*) **and** *neq*: *noteq* ≡ *list-neq xs m*
**assumes** *asum*: *noteq*≠[]
**shows** ∃ *e*. *e* : *set noteq* ∧ *e* > *m*
**proof** (*rule ccontr*)
**let** *?m* = (*mean xs*)
**let** *?neq* = *list-neq xs ?m*
**let** *?eq* = *list-eq xs ?m*
**from** *list-eq-sum* **have** (∑:*?eq*) = *?m* ∗ (*real* (*length ?eq*)) **by** *simp*
**from** *asum* **have** *neq-ne*: *?neq* ≠ [] **unfolding** *m neq* **.**
**assume** *not-el*: ¬(∃ *e*. *e* : *set noteq* ∧ *m* < *e*)
**hence** *not-el-exp*: ¬(∃ *e*. *e* : *set ?neq* ∧ *?m* < *e*) **unfolding** *m neq* **.**
**hence** ∀ *e*. ¬(*e* : *set ?neq*) ∨ ¬(*e* > *?m*) **by** *simp*
**hence** ∀ *e*. *e* : *set ?neq* ⟶ ¬(*e* > *?m*) **by** *blast*
**hence** ∀ *e*. *e* : *set ?neq* ⟶ *e* ≤ *?m* **by** (*simp add*: *linorder-not-less*)
**hence** ∀ *e*. *e* : *set ?neq* ⟶ *e* < *?m* **by** (*simp add*:*order-le-less*)
**with** *assms sum-list-mono-lt* **have** (∑:*?neq*) < *?m* ∗ (*real* (*length ?neq*)) **by** *blast*
**hence**
(∑:*?neq*) + (∑:*?eq*) < *?m* ∗ (*real* (*length ?neq*)) + (∑:*?eq*) **by** *simp*
**also have**
... = (*?m* ∗ ((*real* (*length ?neq*) + (*real* (*length ?eq*)))))
**by** (*simp add*:*field-simps*)
**also have**
... = (*?m* ∗ (*real* (*length xs*)))
**apply** (*subst of-nat-add* [*symmetric*])
**by** (*simp add*: *sum-list-length-split* [*symmetric*])
**also have**
... = ∑:*xs*
**by** (*simp add*: *list-sum-mean* [*symmetric*])
**also from** *not-el calculation* **show** *False* **by** (*simp only*: *sum-list-split* [*symmetric*])
**qed**

**lemma** *pick-one-lt*:
**fixes** *xs*::*real list* **and** *m*::*real*
**defines** *m*: *m* ≡ (*mean xs*) **and** *neq*: *noteq* ≡ *list-neq xs m*
**assumes** *asum*: *noteq*≠[]
**shows** ∃ *e*. *e* : *set noteq* ∧ *e* < *m*
**proof** (*rule ccontr*) — reductio ad absurdum

**let** *?m* = (*mean xs*)
**let** *?neq* = *list-neq xs ?m*
**let** *?eq* = *list-eq xs ?m*
**from** *list-eq-sum* **have** ($\sum$:*?eq*) = *?m* ∗ (*real* (*length ?eq*)) **by** *simp*
**from** *asum* **have** *neq-ne*: *?neq* ≠ [] **unfolding** *m neq* **.**
**assume** *not-el*: ¬(∃ *e*. *e* : *set noteq* ∧ *m* > *e*)
**hence** *not-el-exp*: ¬(∃ *e*. *e* : *set ?neq* ∧ *?m* > *e*) **unfolding** *m neq* **.**
**hence** ∀ *e*. ¬(*e* : *set ?neq*) ∨ ¬(*e* < *?m*) **by** *simp*
**hence** ∀ *e*. *e* : *set ?neq* ⟶ ¬(*e* < *?m*) **by** *blast*
**hence** ∀ *e*. *e* : *set ?neq* ⟶ *e* ≥ *?m* **by** (*simp add*: *linorder-not-less*)
**hence** ∀ *e*. *e* : *set ?neq* ⟶ *e* > *?m* **by** (*auto simp*: *order-le-less*)
**with** *assms sum-list-mono-gt* **have** ($\sum$:*?neq*) > *?m* ∗ (*real* (*length ?neq*)) **by** *blast*
**hence**
($\sum$:*?neq*) + ($\sum$:*?eq*) > *?m* ∗ (*real* (*length ?neq*)) + ($\sum$:*?eq*) **by** *simp*
**also have**
(*?m* ∗ (*real* (*length ?neq*)) + ($\sum$:*?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*)))
**apply** (*subst of-nat-add* [*symmetric*])
**by** (*simp add*: *sum-list-length-split* [*symmetric*])
**also have**
... = $\sum$:*xs*
**by** (*simp add*: *list-sum-mean* [*symmetric*])
**also from** *not-el calculation* **show** *False* **by** (*simp only*: *sum-list-split* [*symmetric*])
**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*: $a \in set\ xs$ **and** *b*: $b \in set\ xs$ **and** *neq*: $a \neq b$
  **shows** $het\ xs > 0$
**proof** (*rule ccontr*)
  **assume** $\neg$ *?thesis*
  **hence** $het\ xs = 0$ **by** *auto*
  **from** *this*[*unfolded het-def*] **have** $list\text{-}neq\ xs\ (mean\ xs) = []$ **by** *simp*
  **from** *arg-cong*[*OF this*, *of set*] **have** *mean*: $\bigwedge x.\ x \in set\ xs \Longrightarrow x = mean\ xs$ **by** *auto*
  **from** *mean*[*OF a*] *mean*[*OF b*] *neq* **show** *False* **by** *auto*
**qed**

$\gamma-eq$: Two lists are $\gamma$-equivalent if and only if they both have the same number of elements and the same arithmetic means.

**definition**
  $\gamma\text{-}eq$ :: $((real\ list)*(real\ list)) \Rightarrow bool$ **where**
  $\gamma\text{-}eq\ a \longleftrightarrow mean\ (fst\ a) = mean\ (snd\ a) \wedge length\ (fst\ a) = length\ (snd\ a)$

$\gamma$-*eq* is transitive and symmetric.

**lemma** $\gamma$-*eq-sym*: $\gamma\text{-}eq\ (a,b) = \gamma\text{-}eq\ (b,a)$
  **unfolding** $\gamma$-*eq-def* **by** *auto*

**lemma** $\gamma$-*eq-trans*:
  $\gamma\text{-}eq\ (x,y) \Longrightarrow \gamma\text{-}eq\ (y,z) \Longrightarrow \gamma\text{-}eq\ (x,z)$
  **unfolding** $\gamma$-*eq-def* **by** *simp*

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

**definition**
  *pos* :: $real\ list \Rightarrow bool$ **where**
  $pos\ l \longleftrightarrow (if\ l{=}[]\ then\ False\ else\ \forall e.\ e : set\ l \longrightarrow e > 0)$

**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*: $pos\ xs \Longrightarrow xs{\neq}[]$ **unfolding** *pos-def* **by** *auto*

**lemma** *pos-cons* [*simp*]:
  $xs \neq [] \longrightarrow pos\ (x\#xs) =$
   $(if\ (x{>}0)\ then\ pos\ xs\ else\ False)$
  (**is** $?P\ x\ xs$ **is** $?A\ xs \longrightarrow ?S\ x\ xs$)
**proof** (*simp add*: *if-split*, *rule impI*)
  **assume** *xsne*: $xs \neq []$
  **hence** *pxs-simp*:
    $pos\ xs = (\forall e.\ e : set\ xs \longrightarrow e > 0)$
    **unfolding** *pos-def* **by** *simp*
  **show**
    $(0 < x \longrightarrow pos\ (x \# xs) = pos\ xs) \wedge$
     $(\neg\ 0 < x \longrightarrow \neg\ pos\ (x \# xs))$
  **proof**
    **{**

```
      assume xgt0: 0 < x
      {
        assume pxs: pos xs
        with pxs-simp have ∀ e. e : set xs ⟶ e > 0 by simp
        with xgt0 have ∀ e. e : set (x#xs) ⟶ e > 0 by simp
        hence pos (x#xs) unfolding pos-def by simp
      }
      moreover
      {
        assume pxxs: pos (x#xs)
        hence ∀ e. e : set (x#xs) ⟶ e > 0 unfolding pos-def by simp
        hence ∀ e. e : set xs ⟶ e > 0 by simp
        with xsne have pos xs unfolding pos-def by simp
      }
      ultimately have pos (x # xs) = pos xs
        apply −
        apply (rule iffI)
        apply auto
        done
    }
    thus 0 < x ⟶ pos (x # xs) = pos xs by simp
  next
    {
      assume xngt0: ¬ (0<x)
      {
        assume pxs: pos xs
        with pxs-simp have ∀ e. e : set xs ⟶ e > 0 by simp
        with xngt0 have ¬ (∀ e. e : set (x#xs) ⟶ e > 0) by auto
        hence ¬ (pos (x#xs)) unfolding pos-def by simp
      }
      moreover
      {
        assume pxxs: ¬pos xs
        with xsne have ¬ (∀ e. e : set xs ⟶ e > 0) unfolding pos-def by simp
        hence ¬ (∀ e. e : set (x#xs) ⟶ e > 0) by auto
        hence ¬ (pos (x#xs)) unfolding pos-def by simp
      }
      ultimately have ¬ pos (x#xs) by auto
    }
    thus ¬ 0 < x ⟶ ¬ pos (x # xs) by simp
  qed
qed
```

### 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)
proof −
  from assms have pl: pos xs and rmvne: (remove1 a xs)≠[] by auto
  from pl have xs ≠ [] by (rule pos-imp-ne)
  with pl pos-def have ∀x. x : set xs ⟶ x > 0 by simp
  hence ∀x. x : set (remove1 a xs) ⟶ x > 0
    using set-remove1-subset[of - xs] by(blast)
  with rmvne show pos (remove1 a xs) unfolding pos-def by simp
qed

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)
  show ?case
  proof cases
    assume xse: xs = []
    hence pos (x#xs) = (x > 0) by simp
    with Cons(2) have x>0 by(simp)
    with xse have 0 < mean (x#xs) by(auto simp:mean-def)
    thus ?thesis by simp
  next
    assume xsne: xs ≠ []
    show ?thesis
    proof cases
      assume pxs: pos xs
      with Cons(1) have z-le-mxs: 0 < mean xs by(simp)
      {
        assume ass: x > 0
        with ass z-le-mxs xsne have 0 < mean (x#xs)
          apply −
          apply (rule mean-gt-0)
          by simp
      }
      moreover
      {
        from xsne pxs have 0 < x
        proof cases
          assume 0 < x thus ?thesis by simp
        next
          assume ¬(0 < x)
          with xsne pos-cons have pos (x#xs) = False by simp
          with Cons(2) show ?thesis by simp
        qed
      }
      ultimately have 0 < mean (x#xs) by simp
```

**thus** *?thesis* **by** *simp*
**next**
**assume** *npxs*: $\neg$*pos xs*
**with** *xsne pos-cons* **have** *pos* $(x\#xs) = $ *False* **by** *simp*
**thus** *?thesis* **using** *Cons*(*2*) **by** *simp*
**qed**
**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** $x \neq [] \wedge het\ x = 0 \Longrightarrow \prod{:}x = (mean\ x)\ \hat{}\ (length\ x)$
**proof** –
**assume** $x \neq [] \wedge het\ x = 0$
**hence** *xne*: $x \neq []$ **and** *hetx*: $het\ x = 0$ **by** *auto*
**from** *hetx* **have** *lz*: *length* (*list-neq x* (*mean x*)) = *0* **unfolding** *het-def* **.**
**hence** $\prod{:}$(*list-neq x* (*mean x*)) = *1* **by** *simp*
**with** *prod-list-split* **have** $\prod{:}x = \prod{:}$(*list-eq x* (*mean x*))
**apply** –
**apply** (*drule meta-spec* [*of - x*])
**apply** (*drule meta-spec* [*of - mean x*])
**by** *simp*
**also with** *list-eq-prod* **have**
... = (*mean x*) ^(*length* (*list-eq x* (*mean x*))) **by** *simp*
**also with** *calculation lz sum-list-length-split* **have**
$\prod{:}x = (mean\ x)\ \hat{}\ (length\ x)$
**apply** –
**apply** (*drule meta-spec* [*of - x*])
**apply** (*drule meta-spec* [*of - mean x*])
**by** *simp*
**thus** *?thesis* **by** *simp*
**qed**

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

**lemma** *het-base*:
**shows** $pos\ x \wedge het\ x = 0 \Longrightarrow gmean\ x = mean\ x$
**proof** –
**assume** *ass*: $pos\ x \wedge het\ x = 0$
**hence**
*xne*: $x \neq []$ **and**
*hetx*: $het\ x = 0$ **and**
*posx*: *pos x*
**by** *auto*
**from** *posx pos-mean* **have** *mxgt0*: $mean\ x > 0$ **by** *simp*
**from** *xne* **have** *lxgt0*: $length\ x > 0$ **by** *simp*
**with** *ass prod-list-het0* **have**
*root* (*length x*) ($\prod{:}x$) = *root* (*length x*) ((*mean x*)^(*length x*))

**by** *simp*
**also from** *lxgt0 mxgt0 real-root-power-cancel* **have** $\ldots = mean\ x$ **by** *auto*
**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 $\gamma$-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*: $m \equiv mean\ xs$ **and**
*neq*: $noteq \equiv list\text{-}neq\ xs\ m$ **and**
*eq*: $eq \equiv list\text{-}eq\ xs\ m$
**assumes** *pos-xs*: *pos xs* **and** *het-gt-0*: $het\ xs > 0$
**shows**
$\exists\, xs'.\ gmean\ xs' > gmean\ xs \wedge \gamma\text{-}eq\ (xs',xs) \wedge$
$het\ xs' < het\ xs \wedge pos\ xs'$
**proof** −
**from** *pos-xs pos-imp-ne* **have**
*pos-els*: $\forall\, y.\ y : set\ xs \longrightarrow y > 0$ **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** $\alpha$ **where**
$\alpha$*-def*: $\alpha : set\ noteq \wedge \alpha > m$ **unfolding** *neq m* **by** *auto*
**from** *assms pick-one-lt neqne* **obtain** $\beta$ **where**
$\beta$*-def*: $\beta : set\ noteq \wedge \beta < m$ **unfolding** *neq m* **by** *auto*
**from** $\alpha$*-def* $\beta$*-def* **have** $\alpha$*-gt*: $\alpha > m$ **and** $\beta$*-lt*: $\beta < m$ **by** *auto*
**from** $\alpha$*-def* $\beta$*-def* **have** *el-neq*: $\beta \neq \alpha$ **by** *simp*
**from** *neqne neq* **have** *xsne*: $xs \neq []$ **by** *auto*

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

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

— remove these elements from xs, and insert two new elements
**obtain** *left-over* **where** *lo*: $left\text{-}over = (remove1\ \beta\ (remove1\ \alpha\ xs))$ **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* $\alpha$*-pos* $\beta$*-pos* $\alpha$*-gt* $\beta$*-lt* **have** *b-pos*: $b > 0$ **by** *simp*

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

**from** *el-neq* $\beta$*-mem* $\alpha$*-mem* **have** $\beta$ : *set xs* $\wedge$ $\alpha$ : *set xs* $\wedge$ $\beta \neq \alpha$ **by** *simp*
**hence** $\alpha$ : *set* (*remove1* $\beta$ *xs*) $\wedge$ $\beta$ : *set*(*remove1* $\alpha$ *xs*) **by** (*auto simp add*: *in-set-remove1*)
**moreover hence** (*remove1* $\alpha$ *xs*) $\neq$ [] $\wedge$ (*remove1* $\beta$ *xs*) $\neq$ [] **by** (*auto*)
**ultimately have**
*mem* : $\alpha$ : *set*(*remove1* $\beta$ *xs*) $\wedge$ $\beta$ : *set*(*remove1* $\alpha$ *xs*) $\wedge$
(*remove1* $\alpha$ *xs*) $\neq$ [] $\wedge$ (*remove1* $\beta$ *xs*) $\neq$ [] **by** *simp*
— prove that new list is positive
**from** *nl* **have** *nl-pos*: *pos new-list*
**proof** *cases*
**assume** *left-over* = []
**with** *nl b-pos m-pos* **show** *?thesis* **by** *simp*
**next**
**assume** *lone*: *left-over* $\neq$ []
**from** *mem pos-imp-rmv-pos pos-xs* **have** *pos* (*remove1* $\alpha$ *xs*) **by** *simp*
**with** *lo lone pos-imp-rmv-pos* **have** *pos left-over* **by** *simp*
**with** *lone mem nl m-pos b-pos* **show** *?thesis* **by** *simp*
**qed**

— now show that the new list has the same mean as the old list
**with** *mem nl lo bdef* $\alpha$*-mem* $\beta$*-mem*
**have** $\sum$:*new-list* = $\sum$:*xs*
**apply** *clarsimp*
**apply** (*subst sum-list-rmv1*)
**apply** *simp*
**apply** (*subst sum-list-rmv1*)
**apply** *simp*
**apply** *clarsimp*
**done**
**moreover from** *lo nl* $\beta$*-mem* $\alpha$*-mem mem* **have**
*leq*: *length new-list* = *length xs*
**apply** –
**apply** (*erule conjE*)+
**apply** (*clarsimp*)
**apply** (*subst length-remove1*, *simp*)
**apply** (*simp add*: *length-remove1*)
**apply** (*auto dest*!:*length-pos-if-in-set*)
**done**
**ultimately have** *eq-mean*: *mean new-list* = *mean xs* **by** (*rule list-mean-eq-iff*)

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

**from** *bdef* $\alpha$*-gt* $\beta$*-lt* **have** *abs* $(m - b) <$ *abs* $(\alpha - \beta)$ **by** *arith*
**moreover from** *bdef* **have** $m+b = \alpha+\beta$ **.**
**ultimately have** *mb-gt-gt*: $m*b > \alpha*\beta$ **by** (*rule le-diff-imp-gt-prod*)
**moreover from** *nl* **have**
$\prod$:*new-list* = $\prod$:*left-over* $*$ $(m*b)$ **by** *auto*
**moreover**
**from** *lo* $\alpha$*-mem* $\beta$*-mem mem remove1-retains-prod*[**where** *'a* = *real*] **have**
*xsprod*: $\prod$:*xs* = $\prod$:*left-over* $*$ $(\alpha*\beta)$ **by** *auto*
**moreover from** *xsne* **have**
*xs* $\neq$ [] **.**
**moreover from** *nl* **have**
*nlne*: *new-list* $\neq$ [] **by** *simp*
**moreover from** *pos-asm lo* **have**
$\prod$:*left-over* $>$ *0*
**proof** $-$
**from** *pos-asm* **have** $\prod$:*xs* $>$ *0* **.**
**moreover**
**from** *xsprod* **have** $\prod$:*xs* = $\prod$:*left-over* $*$ $(\alpha*\beta)$ **.**
**ultimately have** $\prod$:*left-over* $*$ $(\alpha*\beta) >$ *0* **by** *simp*
**moreover**
**from** *pos-els* $\alpha$*-mem* $\beta$*-mem* **have** $\alpha >$ *0* **and** $\beta >$ *0* **by** *auto*
**hence** $\alpha*\beta >$ *0* **by** *simp*
**ultimately show** $\prod$:*left-over* $>$ *0*
**apply** $-$
**apply** (*rule zero-less-mult-pos2* [**where** *a*=$(\alpha * \beta)$])
**by** *auto*
**qed**
**ultimately have** $\prod$:*new-list* $>$ $\prod$:*xs*
**by** *simp*
**moreover with** *pos-asm nl* **have** $\prod$:*new-list* $>$ *0* **by** *auto*
**moreover from** *calculation pos-asm xsne nlne leq list-gmean-gt-iff*
**show** *gmean new-list* $>$ *gmean xs* **by** *simp*
**qed**

— auxiliary info
**from** $\beta$*-lt* **have** $\beta$*-ne-m*: $\beta \neq m$ **by** *simp*
**from** *mem* **have**
$\beta$*-mem-rmv-*$\alpha$: $\beta$ : *set* (*remove1* $\alpha$ *xs*) **and** *rmv-*$\alpha$*-ne*: (*remove1* $\alpha$ *xs*) $\neq$ [] **by** *auto*

**from** $\alpha$*-def* **have** $\alpha$*-ne-m*: $\alpha \neq m$ **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 new-list* (*mean new-list*))
**by** *simp*

**also with** *m nl eq-mean* **have**
  $\ldots = length\ (list\text{-}neq\ (m\#b\#(left\text{-}over))\ m)$
  **by** *simp*
**also with** *bm* **have**
  $\ldots = length\ (list\text{-}neq\ left\text{-}over\ m)$
  **by** *simp*
**also with** *lo* $\beta$*-def* $\alpha$*-def* **have**
  $\ldots = length\ (list\text{-}neq\ (remove1\ \beta\ (remove1\ \alpha\ xs))\ m)$
  **by** *simp*
**also from** $\beta$*-ne-m* $\beta$*-mem-rmv-*$\alpha$ *rmv-*$\alpha$*-ne* **have**
  $\ldots < length\ (list\text{-}neq\ (remove1\ \alpha\ xs)\ m)$
  **apply** −
  **apply** (*rule list-neq-remove1*)
  **by** *simp*
**also from** $\alpha$*-mem* $\alpha$*-ne-m xsne* **have**
  $\ldots < length\ (list\text{-}neq\ xs\ m)$
  **apply** −
  **apply** (*rule list-neq-remove1*)
  **by** *simp*
**also with** *m het-def* **have** $\ldots = het\ xs$ **by** *simp*
**finally show** *het new-list* < *het xs* **.**
**next**
**assume** *bnm*: $b \neq m$
**with** *het-def* **have**
  $het\ new\text{-}list = length\ (list\text{-}neq\ new\text{-}list\ (mean\ new\text{-}list))$
  **by** *simp*
**also with** *m nl eq-mean* **have**
  $\ldots = length\ (list\text{-}neq\ (m\#b\#(left\text{-}over))\ m)$
  **by** *simp*
**also with** *bnm* **have**
  $\ldots = length\ (b\#(list\text{-}neq\ left\text{-}over\ m))$
  **by** *simp*
**also have**
  $\ldots = 1 + length\ (list\text{-}neq\ left\text{-}over\ m)$
  **by** *simp*
**also with** *lo* $\beta$*-def* $\alpha$*-def* **have**
  $\ldots = 1 + length\ (list\text{-}neq\ (remove1\ \beta\ (remove1\ \alpha\ xs))\ m)$
  **by** *simp*
**also from** $\beta$*-ne-m* $\beta$*-mem-rmv-*$\alpha$ *rmv-*$\alpha$*-ne* **have**
  $\ldots < 1 + length\ (list\text{-}neq\ (remove1\ \alpha\ xs)\ m)$
  **apply** −
  **apply** (*simp only*: *nat-add-left-cancel-less*)
  **apply** (*rule list-neq-remove1*)
  **by** *simp*
**finally have**
  $het\ new\text{-}list \leq length\ (list\text{-}neq\ (remove1\ \alpha\ xs)\ m)$
  **by** *simp*
**also from** $\alpha$*-mem* $\alpha$*-ne-m xsne* **have** $\ldots < length\ (list\text{-}neq\ xs\ m)$
  **apply** −

```
      apply (rule list-neq-remove1)
      by simp
    also with m het-def have ... = het xs by simp
    finally show het new-list < het xs .
  qed

    — thus thesis by existence of newlist
  from γ-eq-def lt-het gt-gmean eq-mean leq nl-pos show ?thesis by auto
qed
```

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

```
lemma existence-of-het0 [rule-format]:
  shows ∀ x. p = het x ∧ p > 0 ∧ pos x ⟶
  (∃ y. gmean y > gmean x ∧ γ-eq (x,y) ∧ het y = 0 ∧ pos y)
  (is ?Q p is ∀ x. (?A x p ⟶ ?S x))
proof (induct p rule: nat-less-induct)
  fix n
  assume ind: ∀ m<n. ?Q m
  {
    fix x
    assume ass: ?A x n
    hence het x > 0 and pos x by auto
    with new-list-gt-gmean have
      ∃ y. gmean y > gmean x ∧ γ-eq (x,y) ∧ het y < het x ∧ pos y
      apply −
      apply (drule meta-spec [of - x])
      apply (drule meta-mp)
        apply assumption
      apply (drule meta-mp)
        apply assumption
      apply (subst(asm) γ-eq-sym)
      apply simp
      done
    then obtain β where
      β-def: gmean β > gmean x ∧ γ-eq (x,β) ∧ het β < het x ∧ pos β ..
    then obtain b where bdef: b = het β by simp
    with ass β-def have b < n by auto
    with ind have ?Q b by simp
    with β-def have
      ind2: b = het β ∧ 0 < b ∧ pos β ⟶
      (∃ y. gmean β < gmean y ∧ γ-eq (β, y) ∧ het y = 0 ∧ pos y) by simp
    {
      assume ¬(0<b)
      hence b=0 by simp
      with bdef have het β = 0 by simp
      with β-def have ?S x by auto
    }
```

```
    moreover
    {
      assume 0 < b
      with bdef ind2 β-def have ?S β by simp
      then obtain γ where
        gmean β < gmean γ ∧ γ-eq (β, γ) ∧ het γ = 0 ∧ pos γ ..
      with β-def have gmean x < gmean γ ∧ γ-eq (x,γ) ∧ het γ = 0 ∧ pos γ
        apply clarsimp
        apply (rule γ-eq-trans)
        by auto
      hence ?S x by auto
    }
    ultimately have ?S x by auto
  }
  thus ?Q n by simp
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* $\longleftrightarrow$ *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* $\neq$ *0*
    **hence** *het z* > *0* **by** *auto*
    **moreover obtain** *k* **where** *k* = *het z* **by** *simp*
    **moreover with** *calculation* ‹*pos z*› *existence-of-het0* **have**
      $\exists$ *y*. *gmean y* > *gmean z* $\wedge$ $\gamma$-*eq* (*z*,*y*) $\wedge$ *het y* = *0* $\wedge$ *pos y* **by** *auto*
    **then obtain** $\alpha$ **where**
      *gmean* $\alpha$ > *gmean z* $\wedge$ $\gamma$-*eq* (*z*,$\alpha$) $\wedge$ *het* $\alpha$ = *0* $\wedge$ *pos* $\alpha$ **..**
    **with** *het-base* $\gamma$-*eq-def pos-imp-ne* **have**
      *mean z* = *mean* $\alpha$ **and**
      *gmean* $\alpha$ > *gmean z* **and**
      *gmean* $\alpha$ = *mean* $\alpha$ **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*
  **using**
    *CauchysMeanTheorem*[*OF* ‹*pos z*›]
    *CauchysMeanTheorem-Eq*[*OF* ‹*pos z*›]
    ‹*het z* > *0*›
    **by** *auto*

**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 $nat \Rightarrow real$. 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* = ($\sum j \in \{1..(vlen\ a)\}.\ a_j * b_j$)

**definition**
*norm* :: *vector* ⇒ *real* (‖-‖ *100*) **where**
*norm v* = *sqrt* ($\sum j \in \{1..(vlen\ v)\}.\ v_j\hat{}2$)

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

**lemma** *norm-dot*:
$\|v\| = sqrt\ (v \cdot v)$
**proof** –
**have** $sqrt\ (v \cdot v) = sqrt\ (\sum j \in \{1..(vlen\ v)\}.\ v_j * v_j)$ **unfolding** *dot-def* **by** *simp*
**also with** *real-sq* **have** $\ldots = sqrt\ (\sum j \in \{1..(vlen\ v)\}.\ v_j\hat{}2)$ **by** *simp*
**also have** $\ldots = \|v\|$ **unfolding** *norm-def* **by** *simp*
**finally show** *?thesis* **..**
**qed**

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

**lemma** *norm-pos*:
$\|v\| \geq 0$
**proof** –
**have** $\forall j.\ v_j\hat{}2 \geq 0$ **unfolding** *ith-def* **by** *auto*
**have** $(\sum j \in \{1..(vlen\ v)\}.\ v_j\hat{}2) \geq 0$ **by** (*simp add*: *sum-nonneg*)
**with** *real-sqrt-ge-zero* **have** $sqrt\ (\sum j \in \{1..(vlen\ v)\}.\ v_j\hat{}2) \geq 0$ **.**
**thus** *?thesis* **unfolding** *norm-def* **.**
**qed**

We now prove an intermediary lemma regarding double summation.

**lemma** *double-sum-aux*:
**fixes** *f*::*nat* ⇒ *real*
**shows**
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ f\ k * g\ j)) =$
$(\sum k \in \{1..n\}.\ (\sum j \in \{1..n\}.\ (f\ k * g\ j + f\ j * g\ k)\ /\ 2))$

**proof** −
**have**
$2 * (\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j)) =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j)) +$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j))$
**by** *simp*
**also have**
$\ldots =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j)) +$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ j * g\ k))$
**by** (*simp only*: *double-sum-equiv*)
**also have**
$\ldots =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j + f\ j * g\ k))$
**by** (*auto simp add*: *sum.distrib*)
**finally have**
$2 * (\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j)) =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j + f\ j * g\ k))$ **.**
**hence**
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ f\ k * g\ j)) =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ (f\ k * g\ j + f\ j * g\ k)))*(1/2)$
**by** *auto*
**also have**
$\ldots =$
$(\sum k{\in}\{1..n\}.\ (\sum j{\in}\{1..n\}.\ (f\ k * g\ j + f\ j * g\ k)*(1/2)))$
**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** $|x{\cdot}y| \le \|x\|*\|y\|$
**proof** −
**have** $|x{\cdot}y|\hat{}2 \le (\|x\|*\|y\|)\hat{}2$
**proof** −

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

**have** $(\|x\|*\|y\|)\hat{}2 - |x{\cdot}y|\hat{}2 \ge 0$
**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** $(\sum j{\in}\{1..n\}.\ x_j\hat{}2) \ge 0$ **by** (*simp add*: *sum-nonneg*)
**hence** *xp*: $(sqrt\ (\sum j{\in}\{1..n\}.\ x_j\hat{}2))\hat{}2 = (\sum j{\in}\{1..n\}.\ x_j\hat{}2)$

**by** (*rule real-sqrt-pow2*)

**have** $(\sum j\in\{1..n\}.\ y_j\hat{}2) \geq 0$ **by** (*simp add*: *sum-nonneg*)
**hence** *yp*: $(sqrt\ (\sum j\in\{1..n\}.\ y_j\hat{}2))\hat{}2 = (\sum j\in\{1..n\}.\ y_j\hat{}2)$
**by** (*rule real-sqrt-pow2*)

The main result of this section is that $(\|x\|*\|y\|)\hat{}2$ can be written as a double sum.

**have**
$(\|x\|*\|y\|)\hat{}2 = \|x\|\hat{}2 * \|y\|\hat{}2$
**by** (*simp add*: *real-sq-exp*)
**also from** *nx ny* **have**
$\ldots = (sqrt\ (\sum j\in\{1..n\}.\ x_j\hat{}2))\hat{}2 * (sqrt\ (\sum j\in\{1..n\}.\ y_j\hat{}2))\hat{}2$
**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**
$|x\cdot y|\hat{}2 = (x\cdot y)\hat{}2$
**by** *simp*
**also from** *nx* **have**
$\ldots = (\sum j\in\{1..n\}.\ x_j*y_j)\hat{}2$
**unfolding** *dot-def* **by** *simp*
**also from** *real-sq* **have**
$\ldots = (\sum j\in\{1..n\}.\ x_j*y_j)*(\sum j\in\{1..n\}.\ x_j*y_j)$
**by** *simp*
**also from** *sum-product* **have**
$\ldots = (\sum k\in\{1..n\}.\ (\sum j\in\{1..n\}.\ (x_k*y_k)*(x_j*y_j)))$ **.**
**finally have**
$|x\cdot y|\hat{}2 = (\sum k\in\{1..n\}.\ (\sum j\in\{1..n\}.\ (x_k*y_k)*(x_j*y_j)))$ **.**
**}**

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**
$\ldots =$
$(\sum k\in\{1..n\}.\ (\sum j\in\{1..n\}.\ ((x_k\hat{}2*y_j\hat{}2) + (x_j\hat{}2*y_k\hat{}2))/2)) -$

```
        (∑ k∈{1..n}. (∑ j∈{1..n}. (x_k*y_k)*(x_j*y_j)))
       by (simp only: double-sum-aux)
     also have
       ... =
     (∑ 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 (auto simp add: sum-subtractf)
     also have
       ... =
        (∑ k∈{1..n}. (∑ j∈{1..n}. (inverse 2)*2*
        (((x_k^2*y_j^2) + (x_j^2*y_k^2))*(1/2) − (x_k*y_k)*(x_j*y_j))))
       by auto
     also have
       ... =
        (∑ k∈{1..n}. (∑ j∈{1..n}. (inverse 2)*(2*
       (((x_k^2*y_j^2) + (x_j^2*y_k^2))*(1/2) − (x_k*y_k)*(x_j*y_j)))))
       by (simp only: mult.assoc)
     also have
       ... =
        (∑ k∈{1..n}. (∑ j∈{1..n}. (inverse 2)*
       ((((x_k^2*y_j^2) + (x_j^2*y_k^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
       ... =
       (∑ k∈{1..n}. (∑ j∈{1..n}. (inverse 2)*
       ((((x_k^2*y_j^2) + (x_j^2*y_k^2)) − 2*(x_k*y_k)*(x_j*y_j)))))
       by (simp only: mult.assoc, 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
     proof −
       have (∑ k∈{1..n}. (∑ j∈{1..n}. (x_k*y_j − x_j*y_k)^2)) ≥ 0
         by (simp add: sum-nonneg)
       thus ?thesis by simp
     qed
     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**