Cauchy’s Mean Theorem and the Cauchy-Schwarz Inequality

Benjamin Porter

May 27, 2015
## Contents

1 Cauchy’s Mean Theorem 3
   1.1 Abstract .................................................. 3
   1.2 Formal proof .............................................. 4
      1.2.1 Collection sum and product ......................... 4
      1.2.2 Auxiliary lemma ..................................... 7
      1.2.3 Mean and GMean ..................................... 8
      1.2.4 list-neq, list-eq ................................. 10
      1.2.5 Element selection ................................. 13
      1.2.6 Abstract properties .............................. 15
      1.2.7 Existence of a new collection .................... 19
      1.2.8 Cauchy’s Mean Theorem .......................... 24

2 The Cauchy-Schwarz Inequality 27
   2.1 Abstract .................................................. 27
   2.2 Formal Proof ............................................. 27
      2.2.1 Vector, Dot and Norm definitions. ............... 27
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

description

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:

\[ \sqrt[n]{x_1x_2\ldots x_n} \leq \frac{x_1 + \ldots + x_n}{n} \]

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 $\text{gmean } E \leq \text{mean } E$. QED.


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

**definition**

\[
\text{listsum} :: (\text{real list}) \Rightarrow \text{real} (\sum : [999] 998) \text{ where }
\text{listsum } xs = \text{foldr op+ } xs 0
\]

**definition**

\[
\text{listprod} :: (\text{real list}) \Rightarrow \text{real} (\prod : [999] 998) \text{ where }
\text{listprod } xs = \text{foldr op* } xs 1
\]

**lemma** listsum-empty [simp]: $\sum : [] = 0$

unfolding listsum-def by simp

**lemma** listsum-cons [simp]: $\sum : (a \# b) = a + \sum : b$

unfolding listsum-def by (induct b) simp-all

**lemma** listprod-empty [simp]: $\prod : [] = 1$

unfolding listprod-def by simp

**lemma** listprod-cons [simp]: $\prod : (a \# b) = a \ast \prod : b$

unfolding listprod-def by (induct b) simp-all

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) $m \ast \text{length}(C)$.

**lemma** listsum-mono-lt [rule-format]:

fixes $xs :: \text{real list}$

shows $xs \neq [] \land (\forall x \in \text{set } xs. x < m)$

$\rightarrow ((\sum : xs) < (m \ast (\text{real } (\text{length } xs)))$

proof (induct $xs$)

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 ∑ (y # ys) < m * real (length ys) + y by simp
    with ylm have ∑ (y # ys) < m * (real (length ys) + 1) by (simp add: field-simps)
    with real-of-nat-Suc have ∑ (y # ys) < m * (real (length ys + 1))
      apply -
      apply (drule meta-spec [of - length ys])
      apply (subst (asm) eq-sym-conv)
      by simp
    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 listsum-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 listsum-rmv1:
  a ∈ set xs →⇒ ∑ (remove1 a xs) = ∑ xs − a
  by (induct xs) auto

A handy addition and division distribution law over collection sums.

lemma list-sum-distrib-aux:
  shows (∑ xs/n + ∑ xs) = (1 + (1/n)) * ∑ xs
  proof (induct xs)
    case Nil show ?case by simp
  next
    case (Cons x xs)

CHAPTER 1. CAUCHY’S MEAN THEOREM

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

lemma remove1-retains-prod:
fixes a::real and xs::real list
shows a : set xs \rightarrow \prod : xs = \prod : (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 \prod : (aa # list) = \prod : (remove1 a (aa # list)) * a
proof (cases)
assume aeq: a = aa
hence
\prod : (remove1 a (aa#list)) = list
by simp
hence
\prod : (remove1 a (aa#list)) = \prod : list
by simp
moreover with aeq have
\prod : (aa#list) = \prod : list * a
by simp
ultimately show
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** \( \text{el-gt0-imp-prod-gt0} \) [rule-format]:

fixes \( x :: \text{real list} \)

shows \( \forall y : \text{set xs} \rightarrow y > 0 \implies \Pi : \text{xs} > 0 \)

proof (induct \( \text{xs} \))

next

case Nil show \( \Pi :: \text{xs} > 0 \) by simp

next

case (Cons a xs)

have \( \Pi :: (a\#xs) = \Pi :: \text{xs} * a \) by simp

with Cons have \( a > 0 \) by simp

with \( \text{exp} \) 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** \( \text{prod-exp} \):

fixes \( x :: \text{real} \)

shows \( 4 * (x+y) = (x+y)^2 - (x-y)^2 \)

by (simp add: power2-diff power2-sum)

**Lemma** \( \text{abs-less-imp-sq-less} \) [rule-format]:

fixes \( x :: \text{real} \) and \( y :: \text{real} \) and \( z :: \text{real} \) and \( w :: \text{real} \)

assumes \( \text{diff} : \text{abs} (x-y) < \text{abs} (z-w) \)

shows \( (x-y)^2 < (z-w)^2 \)
proof cases
  assume \( x = y \)
  hence \( \text{abs} \ (x - y) = 0 \) by simp
  moreover with \( \text{diff} \) have \( \text{abs}(z - w) > 0 \) by simp
  hence \( (z - w)^2 > 0 \) by simp
  ultimately show ?thesis by auto
next
  assume \( x \neq y \)
  hence \( \text{abs} \ (x - y) > 0 \) by simp
  with \( \text{diff} \) have \( (\text{abs} \ (x - y))^2 < (\text{abs} \ (z - w))^2 \)
  by -(drule power-strict-mono [where \( a=\text{abs} \ (x - y) \) and \( n=2 \) and \( b=\text{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 \( x::\text{real} \) and \( y::\text{real} \) and \( z::\text{real} \) and \( w::\text{real} \)
  assumes \( \text{diff}: \text{abs} \ (x - y) < \text{abs} \ (z - w) \) and \( \text{sum}: x + y = z + w \)
  shows \( x \times y > z \times w \)
proof
  from \( \text{sum} \) have \( (x + y)^2 = (z + w)^2 \) by simp
  moreover from \( \text{diff} \) 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 \( x \times y > z \times w \) 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)\Rightarrow real where
  mean s = (\sum :s / real (length s))

Geometric mean
definition
  gmean :: (real list)\Rightarrow real where
  gmean s = root (length s) (\prod :s)
CHAPTER 1. CAUCHY'S MEAN THEOREM

Properties

Here we present some trivial properties of mean and gmean.

lemma list-sum-mean:

fixes xs::real list

shows \( \sum xs = ((\text{mean } xs) \times (\text{real } (\text{length } xs))) \)

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: (\( \text{length one} = \text{length two} \))

shows (\( \text{mean } one = \text{mean } two \))

proof -

from se le have

(\( \sum one / \text{real } (\text{length one}) \)) = (\( \sum two / \text{real } (\text{length two}) \))

by auto

thus ?thesis unfolding mean-def.

qed

lemma list-gmean-gt-iff:

fixes one::real list and two::real list

assumes gz1: \( \prod one > 0 \) and gz2: \( \prod two > 0 \) and
ne1: one \( \neq [] \) and ne2: two \( \neq [] \) and
pe: (\( \prod one > \prod two \)) and
le: (\( \text{length one} = \text{length two} \))

shows (\( \text{gmean } one > \text{gmean } two \))

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 \( \neq [] \) \( \rightarrow \) mean ((\( \text{mean } xs \))\#xs) = mean xs

proof

assume lne: xs \( \neq [] \)

obtain len where ld: len = real (\( \text{length } xs \)) by simp

with lne have lgt0: len > 0 by simp

hence lnez: len \( \neq 0 \) by simp

from lgt0 have lunez: len + 1 \( \neq 0 \) by simp
CHAPTER 1. CAUCHY’S MEAN THEOREM

from ld have mean: mean xs = ∑:xs / len unfolding mean-def by simp
with ld real-of-nat-add real-of-one mean-def
have mean ((mean xs)#xs) = (∑:xs/len + ∑:xs) / (1+len)
  by simp
also from list-sum-distrib-aux have
  ... = (1 + (1/len))∗∑:xs / (1+len) by simp
also with lnez have
  ... = (len + 1)∗∑:xs / (len * (1+len))
  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 ≠ [] ∧ 0 < x ∧ 0 < (mean xs) → 0 < (mean (x#xs))
proof
  assume a: xs ≠ [] ∧ 0 < x ∧ 0 < mean xs
  hence xgt0: 0 < x and mgt0: 0 < mean xs by auto
  from a have lxsgt0: length xs ≠ 0 by simp
  from mgt0 have xsgt0: 0 < ∑:xs
    proof
      have mean xs = ∑:xs / real (length xs) unfolding mean-def by simp
      hence ∑: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 ∑:(x#xs) > 0 by simp
  thus 0 < (mean (x#xs))
    proof
      assume 0 < ∑:(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) ⇒ '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`
  **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: \(\text{length } (\text{list-neq } (\text{remove1 } a \ xs) \ m) < \text{length } (\text{list-neq } xs \ m)\)
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 \(x-ne-m\): \(x\neq m\)
from a-ne-x have \(\text{remove1 } a \ (x\#xs) = x\#(\text{remove1 } a \ xs)\)
by simp
hence
\(\text{length } (\text{list-neq } (\text{remove1 } a \ (x\#xs)) \ m) = \text{length } (\text{list-neq } (x\#(\text{remove1 } a \ xs)) \ m)\)
by simp
also with \(x-ne-m\) have
\(\ldots = 1 + \text{length } (\text{list-neq } (\text{remove1 } a \ xs) \ m)\)
by simp
finally have
\(\text{length } (\text{list-neq } (\text{remove1 } a \ (x\#xs)) \ m) = 1 + \text{length } (\text{list-neq } (\text{remove1 } a \ xs) \ m)\)
by simp
moreover with \(x-ne-m\) a-ne-x have
\(\text{length } (\text{list-neq } (x\#xs) \ m) = 1 + \text{length } (\text{list-neq } xs \ m)\)
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 \(\text{list-eq}, \text{list-neq}, \text{length}, \text{sum}\) and product.

lemma list-eq-sum [simp]:
fixes \(xs\)::real list
shows \(\sum: (\text{list-eq } xs \ m) = (m \ast (\text{real } (\text{length } (\text{list-eq } xs \ m))))\)
apply (induct-tac \(xs\))
apply simp
apply clarsimp
apply (subst real-of-nat-Suc)
apply (simp add:field-simps)
done

lemma list-eq-prod [simp]:
fixes \(xs\)::real list
shows \(\Pi: (\text{list-eq } xs \ m) = (m \hat{\ast} (\text{length } (\text{list-eq } xs \ m)))\)
apply (induct-tac xs)
apply simp
apply clarsimp
done

lemma listsum-split:
  fixes xs::real list
  shows \( \sum xs = (\sum (\text{list-neq xs } m) + \sum (\text{list-eq xs } m)) \)
apply (induct xs)
apply simp
apply clarsimp
done

lemma listprod-split:
  fixes xs::real list
  shows \( \prod xs = (\prod (\text{list-neq xs } m) \ast \prod (\text{list-eq xs } m)) \)
apply (induct xs)
apply simp
apply clarsimp
done

lemma listsum-length-split:
  fixes xs::real list
  shows \( \text{length } xs = \text{length } (\text{list-neq xs } m) + \text{length } (\text{list-eq xs } m) \)
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 than (or less than) the mean.

lemma pick-one-gt:
  fixes xs::real list and m::real
  defines m: \( m \equiv (\text{mean } xs) \) and neq: \( \text{noteq } \equiv \text{list-neq } xs \ m \)
  assumes asum: \( \text{noteq } \neq [] \)
  shows \( \exists e. e : \text{set } \text{noteq } \land e > m \)
proof (rule ccontr)
  let ?m = (mean xs)
  let ?noteq = list-neq xs ?m
  let ?eq = list-eq xs ?m
  from list-eq-sum have \( (\sum:?eq) = ?m \ast (\text{real } (\text{length } ?eq)) \) by simp
  from asum have neq-ne: \( \text{noteq } \neq [] \) unfolding m neq .
  assume not-el: \( \neg(\exists e. e : \text{set } \text{noteq } \land \text{m } < e) \)
  hence not-el-exp: \( \neg(\exists e. e : \text{set } \text{noteq } \land \text{m } < e) \) unfolding m neq .
  hence \( \forall e. \neg(e : \text{set } \text{noteq } \lor \neg(e > ?m) \) by simp
  hence \( \forall e. e : \text{set } \text{noteq } \longrightarrow \neg(e > ?m) \) by blast
  hence \( \forall e. e : \text{set } \text{noteq } \longrightarrow e \leq ?m \) by (simp add: linorder-not-less)
  hence \( \forall e. e : \text{set } \text{noteq } \longrightarrow e < ?m \) by (simp add:order-le-less)
with assms listsum-mono-lt have \((\sum:\text{?neq}) < \text{?m} \ast (\text{real length \text{?neq}}))\) by blast
hence
\((\sum:\text{?neq}) + (\sum:\text{?eq}) < \text{?m} \ast (\text{real length \text{?neq}})) + (\sum:\text{?eq})\) by simp
also have
\(\ldots = (\text{?m} \ast ((\text{real length \text{?neq}}) + (\text{real length \text{?eq}}))))\)
by (simp add:field-simps)
also have
\(\ldots = \sum:\text{xs}\) by (simp only: listsum-split [symmetric])
... = \sum_{xs} \, \text{by (simp add: list-sum-mean [symmetric])}
also from not-el calculation show False by (simp only: listsum-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

definition
het :: real list \Rightarrow nat where
het l = length (list-neq l (mean l))

lemma het-gt-0-imp-noteq-ne: het l > 0 \Rightarrow list-neq l (mean l) \neq []
  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-neq xs (mean xs) = [] by simp
  from arg-cong[OF this, of set] have mean: \\\_ 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
\(\gamma\)-eq :: ((real list)* (real list)) \Rightarrow bool where
\(\gamma\)-eq a \leftrightarrow mean (fst a) = mean (snd a) \land length (fst a) = length (snd a)

\(\gamma\)-eq is transitive and symmetric.

lemma \(\gamma\)-eq-sym: \(\gamma\)-eq (a,b) = \(\gamma\)-eq (b,a)
  unfolding \(\gamma\)-eq-def by auto

lemma \(\gamma\)-eq-trans:
\(\gamma\)-eq (x,y) \Longrightarrow \(\gamma\)-eq (y,z) \Longrightarrow \(\gamma\)-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 ⇒ 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

proof
(simp add: split-if, rule impI)
assume xsne: xs ≠ []
hence pxs-simp:
pos xs = (∀ e : set xs → e > 0)
unfolding pos-def by simp
show
(0 < x =⇒ pos (x # xs) = pos xs) ∧
(¬ 0 < x =⇒ ¬ pos (x # xs))
proof
{ assume xgt0: 0 < x
  { assume pxs: pos xs
    with pxs-simp have ∀ e : set xs → e > 0 by simp
    with xgt0 have ∀ e : set (x # xs) → e > 0 by simp
    hence pos (x # xs) unfolding pos-def by simp
  }
moreover
{ assume pxs: pos (x # xs)
  hence ∀ e : set (x # xs) → e > 0 unfolding pos-def by simp
  hence ∀ 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 : set xs → e > 0 by simp
  }
}
with \( x > 0 \) have \( \forall e. e : set (x #xs) \rightarrow e > 0 \) by auto

hence \( \neg (\text{pos } (x #xs)) \) unfolding pos-def by simp

} moreover

{ assume \( \neg \text{pos } xs \)
with \( \text{xsne} \) have \( \forall e. e : set xs \rightarrow e > 0 \) unfolding pos-def by simp

hence \( \neg (\text{pos } (x #xs)) \) by auto

hence \( \neg (\text{pos } (x #xs)) \) unfolding pos-def by simp

} ultimately have \( \neg \text{pos } (x #xs) \) by auto

} thus \( \neg 0 < x \rightarrow \neg \text{pos } (x #xs) \) by simp

qed

qed

Properties

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

Two lemmas regarding \textit{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.

\textbf{lemma pos-imp-rmv-pos:}

assumes \( (\text{remove1 } a \text{ } xs) \neq [] \) \text{pos } xs shows \( \text{pos } (\text{remove1 } a \text{ } xs) \)

\textbf{proof –}

from \( \text{assms} \) have \( \text{pl } \) \text{pos } xs and \( \text{rmvne: } (\text{remove1 } a \text{ } xs) \neq [] \) by auto

from \( \text{pl } \) have \( \text{xsne } \neq [] \) by (rule pos-imp-ne)

with \( \text{pl } \) \text{pos-def } \forall x. x : set xs \rightarrow x > 0 \) by simp

hence \( \forall x. x : set (\text{remove1 } a \text{ } xs) \rightarrow x > 0 \) by (simp

using set-remove1-subset[of \text{xs}] blast

with \( \text{rmvne } \) show \( \text{pos } (\text{remove1 } a \text{ } xs) \) unfolding pos-def by simp

qed

\textbf{lemma pos-mean:}

\( \text{pos } xs \rightarrow mean \text{xs } > 0 \)

\textbf{proof (induct xs)}

\textit{case Nil} thus \( ?\text{case } \) by (simp add: pos-def)

\textbf{next}

\textit{case (Cons } x \text{xs)}

\textbf{show } ?\text{case}

\textbf{proof cases}

\textit{assume } zse: \( xs = [] \)

hence \( \text{pos } (x #xs) = (x > 0) \) by simp

with \( \text{Cons}(2) \) have \( x > 0 \) by (simp)

with \( \text{zse } \) have \( 0 < \text{mean } (x #xs) \) by (auto simp: mean-def)

thus \( ?\text{thesis } \) by simp

\textbf{next}

\textit{assume } zsne: \( xs \neq [] \)

\textbf{show } ?\text{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: ¬pos xs
  with xsne pos-cons have pos (x#xs) = False by simp
  thus ?thesis using Cons(2) by simp
  qed
  qed

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

lemma listprod-het0:
  shows \( x \neq [] \land \text{het } x = 0 \implies \prod : x = (\text{mean } x)^{\text{length } x} \)
proof
  assume x̸[] ∧ het x = 0
  hence xne: x̸[] and hetx: het x = 0 by auto
  from hetx have l2: length (list-neq x (mean x)) = 0 unfolding het-def .
  hence \( \prod : (\text{list-neq } x (\text{mean } x)) = 1 \) by simp
  with listprod-split have \( \prod : x = \prod : (\text{list-eq } x (\text{mean } x)) \)
    apply
    apply (drule meta-spec [of - x])
    apply (drule meta-spec [of - mean x])
    by simp
  also with listeq-prod have
    \( \ldots = (\text{mean } x)^{\text{length } (\text{list-eq } x (\text{mean } x))} \) by simp
  also with calculation l2 listsum-length-split have
    \( \prod : x = (\text{mean } x)^{\text{length } x} \)
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 ∧ het x = 0 ⇒ gmean x = mean x
proof −
assume ass: pos x ∧ het x = 0
hence
xne: x≠[] 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 lgt0: length x > 0 by simp
with ass listprod-het0 have
root (length x) (∏:x) = root (length x) ((mean x) ^ (length x))
by simp
also from lgt0 mxgt0 real-root-power-cancel have ... = 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 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: ∀ y, y : set xs → y > 0 by (unfold pos-def, simp)
with el-gt0-imp-prod-gt0 have pos-asn: ∏:xs > 0 by simp
Pick two elements from \( xs \), one greater than \( m \), one less than \( m \).

---

---
have \[ \sum : \text{new-list} = \sum : \text{xs} \]
apply clarsimp
apply (subst listsum-rmv1)
apply simp
apply (subst listsum-rmv1)
apply simp
apply clarsimp
done
moreover from lo nl \( \beta \)-mem \( \alpha \)-mem mem have
\[ \text{leq: length new-list} = \text{length xs} \]
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: \[ \text{mean new-list} = \text{mean xs} \] by (rule list-mean-eq-iff)

— finally show that the new list has a greater gmean than the old list
have \( \text{gt-gmean: gmean new-list} > \text{gmean xs} \)
proof –
from bdef \( \alpha \)-gt \( \beta \)-lt have \[ \text{abs} (m - b) < \text{abs} (\alpha - \beta) \] 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
\[ \prod : \text{new-list} = \prod : \text{left-over} * (m*b) \] by auto
moreover from lo \( \alpha \)-mem \( \beta \)-mem mem remove1-retains-prod have
\[ \text{xsp:prod: } \prod : \text{xs} = \prod : \text{left-over} * (\alpha*\beta) \] by auto
moreover from xsne have
\[ \text{xs} \neq [] . \]
moreover from nl have
\( \text{nlne: } \text{new-list} \neq [] \) by simp
moreover from pos-asn lo have
\[ \prod : \text{left-over} > 0 \]
proof –
from pos-asn have \( \prod : \text{xs} > 0 . \)
moreover from xspprod have \( \prod : \text{xs} = \prod : \text{left-over} * (\alpha*\beta) . \)
ultimately have \( \prod : \text{left-over} * (\alpha*\beta) > 0 \) 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 \( \prod : \text{left-over} > 0 \)
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 zsne 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-\( \alpha\): 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
... = length (list-neq (m#b#(left-over)) m)
by simp
also with bm have
... = length (list-neq left-over m)
by simp
also with bm have
lo \( \beta\)-def \( \alpha\)-def have
... = length (list-neq (remove1 \( \beta \) (remove1 \( \alpha \) xs)) m)
by simp
also from \( \beta-ne-m \) \( \beta\)-mem-rmv-\( \alpha\) rmv-\( \alpha\)-ne have
... < length (list-neq (remove1 \( \alpha \) xs) m)
apply
apply (rule list-neq-remove1)
by simp
also from \( \alpha\)-mem \( \alpha-ne-m \) zsne have
... < length (list-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.
next
assume bnm: \( b\neq m \)
with het-def have
het new-list = length (list-neq new-list (mean new-list))
by simp
also with m nl eq-mean have
... = length (list-neq (m#b#(left-over)) m)
by simp
also with bnm have
... = length (b#(list-neq left-over m))
by simp
also have
... = 1 + length (list-neq left-over m)
by simp
also with lo β-def α-def have
... = 1 + length (list-neq (remove1 β (remove1 α xs)) m)
by simp
also from β-ne-m β-mem-rmv-α rmv-α-ne have
... < 1 + length (list-neq (remove1 α xs) m)
apply –
apply (simp only: nat-add-left-cancel-less)
apply (rule list-neq-remove1)
by simp
finally have
het new-list ≤ length (list-neq (remove1 α xs) m)
by simp
also from α-mem α-ne-m xsnr have \ldots < length (list-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.
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 γ-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 clar simp
      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 \( gmean z \leq mean z \)

proof cases
  assume \( het \ z = 0 \)
  with \( \langle pos \ z \rangle \) \( z ne \) \( het-base \) have \( gmean z = mean z \) by simp
  thus \(?thesis\) by simp

next
  assume \( het \ z \neq 0 \)
  hence \( het \ z > 0 \) by simp
  moreover obtain \( k \) where \( k = \) \( het \ z \) by simp
  moreover with \( calculation \) \( \langle pos \ z \rangle \) \( existence-of-het0 \) have
  \( \exists y. \ gmean y > gmean z \land \gamma-eq (z,y) \land het y = 0 \land pos y \) by auto
  then obtain \( \alpha \) where
  \( gmean \ \alpha > gmean z \land \gamma-eq (z,\alpha) \land het \ \alpha = 0 \land pos \ \alpha \) ..
  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 \(?thesis\) 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 \( gmean z = mean z \leftarrow\rightarrow het \ z = 0 \)

proof
  assume \( het \ z = 0 \)
  with \( het-base[\langle pos \ z \rangle] \) \( show \ gmean z = mean z \) 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 \( calculation \) \( \langle pos \ z \rangle \) \( existence-of-het0 \) have
    \( \exists y. \ gmean y > gmean z \land \gamma-eq (z,y) \land het y = 0 \land pos y \) by auto
    then obtain \( \alpha \) where
    \( gmean \ \alpha > gmean z \land \gamma-eq (z,\alpha) \land het \ \alpha = 0 \land pos \ \alpha \) ..
    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
corollary CauchysMeanTheorem-Less:
  fixes $z :: \text{real list}$
  assumes $\text{pos } z$ and $\text{het } z > 0$
  shows $\text{gmean } z < \text{mean } z$
  using
    CauchysMeanTheorem[OF $\langle \text{pos } z \rangle$]
    CauchysMeanTheorem-Eq[OF $\langle \text{pos } z \rangle$]
  $\langle \text{het } z > 0 \rangle$
  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 $\mathbb{R}^n$. 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 $\textit{nat} \Rightarrow \textit{real}$. We also define some accessor functions and appropriate notation.

type-synonym vector = (nat⇒real) * nat
CHAPTER 2. THE CAUCHY-SCHWARZ INEQUALITY

**Definition**

\[
\text{ith} :: \text{vector} \Rightarrow \text{nat} \Rightarrow \text{real} \\ \text{where}
\]
\[
\text{ith} \ v \ i = \text{fst} \ v \ i
\]

**Definition**

\[
\text{vlen} :: \text{vector} \Rightarrow \text{nat} \\ \text{where}
\]
\[
\text{vlen} \ v = \text{snd} \ v
\]

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**

\[
\text{dot} :: \text{vector} \Rightarrow \text{vector} \Rightarrow \text{real} \quad (\text{infixr} \cdot \ 60) \ \text{where}
\]
\[
\text{dot} \ a \ b = \left( \sum_{j \in \{1..(\text{vlen} \ a)\}} a_j \cdot b_j \right)
\]

**Definition**

\[
\text{norm} :: \text{vector} \Rightarrow \text{real} \quad (\|\cdot\| \ 100) \ \text{where}
\]
\[
\text{norm} \ v = \sqrt{\left( \sum_{j \in \{1..(\text{vlen} \ v)\}} v_j^2 \right)}
\]

**Notation** (HTML output)

\[
\text{norm} \ (\|\cdot\| \ 100)
\]

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

**Lemma** \( \text{norm-dot} \):

\[
\|v\| = \sqrt{v \cdot v}
\]

**Proof**

- have \( \sqrt{v \cdot v} = \sqrt{\left( \sum_{j \in \{1..(\text{vlen} \ v)\}} v_j \cdot v_j \right)} \) unfolding \( \text{dot-def} \) by simp
- also with \( \text{real-sq} \) have \( \ldots = \sqrt{\left( \sum_{j \in \{1..(\text{vlen} \ v)\}} v_j^2 \right)} \) by simp
- also have \( \ldots = \|v\| \) unfolding \( \text{norm-def} \) by simp
- finally show ?thesis ..

**QED**

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

**Lemma** \( \text{norm-pos} \):

\[
\|v\| \geq 0
\]

**Proof**

- have \( \forall j. \ v_j \cdot 2 \geq 0 \) unfolding \( \text{ith-def} \) by auto
- hence \( \forall j \in \{1..(\text{vlen} \ v)\}. \ v_j^2 \geq 0 \) by simp
- with \( \text{setsum-nonneg} \) have \( \left( \sum_{j \in \{1..(\text{vlen} \ v)\}} v_j^2 \right) \geq 0 \).
- with \( \text{real-sqrt-ge-zero} \) have \( \sqrt{\left( \sum_{j \in \{1..(\text{vlen} \ v)\}} v_j^2 \right)} \geq 0 \).
- thus ?thesis unfolding \( \text{norm-def} \).

**QED**

We now prove an intermediary lemma regarding double summation.

**Lemma** \( \text{double-sum-aux} \):
CHAPTER 2. THE CAUCHY-SCHWARZ INEQUALITY

fixes \( f :: \text{nat} \Rightarrow \text{real} \)
shows
\[
\left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j) \right) = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (f_k * g_j + f_j * g_k)) / 2 \right)
\]
proof
have \( 2 * (\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j)) = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j) + (\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j)) \right) \) by simp
also have
\[
\cdots = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j + f_j * g_k) \right)
\]
by (simp only: double-sum-eq)
also have
\[
\cdots = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j + f_j * g_k) \right)
\]
by (auto simp add: setsum.distrib)
finally have
\[
2 * (\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j)) = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j + f_j * g_k) \right).
\]
hence
\[
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} f_k * g_j)) = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (f_k * g_j + f_j * g_k))) * (1/2) \right)
\]
by auto
also have
\[
\cdots = \left( \sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (f_k * g_j + f_j * g_k)) * (1/2) \right)
\]
by (simp add: setsum-right-distrib setsum.commute)
finally show \( \text{thesis} \) 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 \( x :: \text{vector} \)
assumes \( \text{vlen} \ x = \text{vlen} \ y \)
shows \( |x^\cdot y| \leq \|x\| * \|y\| \)
proof
have \( |x^\cdot y| ^2 \leq (\|x\| * \|y\|) ^2 \)
proof
obtain \( n \) where \( nx : n = \text{vlen} \ x \) by simp
with \( \text{vlen} \ x = \text{vlen} \ y \) have \( ny : n = \text{vlen} \ y \) by simp

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

have \( (\|x\| * \|y\|) ^2 - |x^\cdot y| ^2 \geq 0 \)
proof
obtain \( n \) where \( nx : n = \text{vlen} \ x \) by simp
with \( \text{vlen} \ x = \text{vlen} \ y \) have \( ny : n = \text{vlen} \ y \) by simp

CHAPTER 2. THE CAUCHY-SCHWARZ INEQUALITY

Some preliminary simplification rules.

have \( \forall j \in \{1..n\}. \ x_j \cdot 2 \geq 0 \) by simp

hence (\( \sum j \in \{1..n\}. \ x_j \cdot 2 \)) \( \geq 0 \) by (rule setsum-nonneg)

hence \( x \cdot p \): (\( \sqrt{\sum j \in \{1..n\}. \ x_j \cdot 2} \)) \( \cdot 2 \) = (\( \sum j \in \{1..n\}. \ x_j \cdot 2 \))

by (rule real-sqrt-pow2)

have \( \forall j \in \{1..n\}. \ y_j \cdot 2 \geq 0 \) by simp

hence (\( \sum j \in \{1..n\}. \ y_j \cdot 2 \)) \( \geq 0 \) by (rule setsum-nonneg)

hence \( y \cdot p \): (\( \sqrt{\sum j \in \{1..n\}. \ y_j \cdot 2} \)) \( \cdot 2 \) = (\( \sum j \in \{1..n\}. \ y_j \cdot 2 \))

by (rule real-sqrt-pow2)

The main result of this section is that (\( \|x\| \cdot \|y\| \)) \( \cdot 2 \) can be written as a double sum.

have

\( (\|x\| \cdot \|y\|) \cdot 2 = \|x\| \cdot 2 \cdot \|y\| \cdot 2 \)

by (simp add: real-sq-exp)

also from \( \forall x \ y \) have

\( \ldots = (\sqrt{\sum j \in \{1..n\}. \ x_j \cdot 2}) \cdot 2 \cdot (\sqrt{\sum j \in \{1..n\}. \ y_j \cdot 2}) \cdot 2 \)

unfolding norm-def by auto

also from \( x \cdot p \ y \) have

\( \ldots = (\sum j \in \{1..n\}. \ x_j \cdot 2) \cdot (\sum j \in \{1..n\}. \ y_j \cdot 2) \)

by simp

also from setsum-product have

\( \ldots = (\sum k \in \{1..n\}. \ (\sum j \in \{1..n\}. \ (x_k \cdot 2) \cdot (y_j \cdot 2))) \cdot . \)

finally have

\( (\|x\| \cdot \|y\|) \cdot 2 = (\sum k \in \{1..n\}. \ (\sum j \in \{1..n\}. \ (x_k \cdot 2) \cdot (y_j \cdot 2))) \cdot . \)

} moreover

We also show that \( |x \cdot y| \cdot 2 \) can be expressed as a double sum.

have

\( |x \cdot y| \cdot 2 = (x \cdot y) \cdot 2 \)

by simp

also from \( \forall x \ y \) have

\( \ldots = (\sum j \in \{1..n\}. \ x_j \cdot y_j) \cdot 2 \)

unfolding dot-def by simp

also from real-sq have

\( \ldots = (\sum j \in \{1..n\}. \ x_j \cdot y_j) \cdot (\sum j \in \{1..n\}. \ x_j \cdot y_j) \)

by simp

also from setsum-product have

\( \ldots = (\sum k \in \{1..n\}. \ (\sum j \in \{1..n\}. \ (x_k \cdot y_k) \cdot (x_j \cdot y_j))) \cdot . \)

finally have

\( |x \cdot y| \cdot 2 = (\sum k \in \{1..n\}. \ (\sum j \in \{1..n\}. \ (x_k \cdot y_k) \cdot (x_j \cdot y_j))) \cdot . \)

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

ultimately have

\( (\|x\| \cdot \|y\|) \cdot 2 - |x \cdot y| \cdot 2 =

(\sum k \in \{1..n\}. \ (\sum j \in \{1..n\}. \ (x_k \cdot 2) \cdot (y_j \cdot 2))) -

\)
\[(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_k) \cdot (x_j \cdot y_j))) \]

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} ((x_k \cdot y_j \cdot y_j) + (x_j \cdot y_k \cdot y_k))/2)) - 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_k) \cdot (x_j \cdot y_j)))
\]
by (simp only: double-sum-aux)

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} ((x_k \cdot y_j \cdot y_j) + (x_j \cdot y_k \cdot y_k))/2) - (x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (auto simp add: setsum-subtractf)

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by auto

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (simp only: mult.assoc)

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (auto simp add: distrib-right mult.assoc ac-simps)

also have
\[
\ldots = 
(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (simp only: mult.assoc, simp)

also have
\[
\ldots = 
(inverse 2) \cdot (\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (simp only: setsum-right-distrib)

also have
\[
\ldots = 
(inverse 2) \cdot (\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k \cdot y_j \cdot y_j) - x_k \cdot y_k) \cdot (x_j \cdot y_j))
\]
by (simp only: power2-diff real-sq-exp, auto simp add: ac-simps)

also have \(\ldots \geq 0\)

proof
\[
\begin{proof}
\end{proof}
\]
by \((\text{rule setsum-nonneg})\)
\textbf{thus} \(\text{thesis by simp}\)
\textbf{qed}

finally show \((\|x\|\|y\|)^2 - |x \cdot y|^2 \geq 0\).
\textbf{qed}
\textbf{thus} \(\text{thesis by simp}\)
\textbf{qed}

moreover have \(0 \leq \|x\|\|y\|\)
\textbf{by} \((\text{auto simp add: norm-pos})\)

ultimately show \(\text{thesis by (rule power2-le-imp-le)}\)
\textbf{qed}

\textbf{end}