

Cauchy's Mean Theorem and the Cauchy-Schwarz Inequality

Benjamin Porter

December 12, 2009

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	Auxillary lemma	7
1.2.3	Mean and GMean	8
1.2.4	<i>list-neq, list-eq</i>	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	26
2.1	Abstract	26
2.2	Formal Proof	26
2.2.1	Vector, Dot and Norm definitions.	26

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 auxillary 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] 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

definition

$\text{listsum} :: (\text{real list}) \Rightarrow \text{real}$ ($\sum :- [999] 1000$) **where**
 $\text{listsum } xs = \text{foldr } op+ \text{ } xs \ 0$

definition

$\text{listprod} :: (\text{real list}) \Rightarrow \text{real}$ ($\prod :- [999] 1000$) **where**
 $\text{listprod } xs = \text{foldr } op* \text{ } xs \ 1$

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*]: $\prod : (a \# b) = a * \prod : b$
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) $m * \text{length}(C)$.

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

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

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

$\longrightarrow ((\sum : xs) < (m * (\text{real } (\text{length } xs))))$

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:ring-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)

```

show *?case*
proof –
have
 $\sum:(x\#xs)/n = x/n + \sum:xs/n$
by (*simp add: add-divide-distrib*)
also with *Cons* **have**
 $\dots = x/n + (1+1/n)*\sum:xs - \sum:xs$
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
 $\dots = x/n + (1+(1/n)- 1)*\sum:xs + \sum:(x\#xs)$
by (*subst real-mult-1 [symmetric, of $\sum:xs$], simp only: ring-simps*)
also have
 $\dots = x/n + (1/n)*\sum:xs + \sum:(x\#xs)$
by *simp*
also have
 $\dots = (1/n)*\sum:(x\#xs) + 1*\sum:(x\#xs)$ **by** (*simp add:ring-simps*)
finally show *?thesis* **by** (*simp only: ring-simps*)
qed
qed

lemma *remove1-retains-prod*:
fixes *a::real and xs::real list*
shows $a : set\ xs \longrightarrow \prod:xs = \prod:(remove1\ a\ xs) * a$
(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
 $remove1\ a\ (aa\ \# list) = list$
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

```

     $\prod:(aa\#list) = \prod:remove1\ a\ (aa\ \#list) * a$ 
    by simp
  next
  assume naeq:  $a \neq aa$ 
  with aml have aml2:  $a : set\ list$  by simp
  from naeq have
     $remove1\ a\ (aa\ \#list) = aa\ \#(remove1\ a\ list)$ 
    by simp
  moreover hence
     $\prod:(remove1\ a\ (aa\ \#list)) = aa * \prod:(remove1\ a\ list)$ 
    by simp
  moreover from aml2 plist have
     $\prod:list = \prod:(remove1\ a\ list) * a$ 
    by simp
  ultimately show
     $\prod:(aa\ \#list) = \prod:remove1\ a\ (aa\ \#list) * a$ 
    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::real list
  shows  $\forall y. y : set\ xs \longrightarrow y > 0 \implies \prod:xs > 0$ 
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 add: mult-pos-pos)
qed
```

1.2.2 Auxillary lemma

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

```

lemma prod-exp:
  fixes x::real
  shows  $4*(x*y) = (x+y)^2 - (x-y)^2$ 
apply (simp only: diff-minus)
apply (simp add: real-sum-squared-expand)
done
```

```

lemma abs-less-imp-sq-less [rule-format]:
  fixes x::real and y::real and z::real and w::real
```

assumes *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 *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 *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 *diff*: $\text{abs } (x-y) < \text{abs } (z-w)$ **and** *sum*: $x+y = z+w$
shows $x*y > z*w$
proof -
from *sum* **have** $(x+y)^2 = (z+w)^2$ **by** *simp*
moreover from *diff* **have** $(x-y)^2 < (z-w)^2$ **by** (*rule abs-less-imp-sq-less*)
ultimately have $(x+y)^2 - (x-y)^2 > (z+w)^2 - (z-w)^2$ **by** *auto*
thus $x*y > z*w$ **by** (*simp only: prod-exp* [*symmetric*])
qed

1.2.3 Mean and GMean

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

Definitions

Arithmetic mean

definition

$\text{mean} :: (\text{real list}) \Rightarrow \text{real}$ **where**
 $\text{mean } s = (\sum :s / \text{real } (\text{length } s))$

Geometric mean

definition

$\text{gmean} :: (\text{real list}) \Rightarrow \text{real}$ **where**
 $\text{gmean } s = \text{root } (\text{length } s) (\prod :s)$

Properties

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

lemma *list-sum-mean*:

```

fixes xs::real list
shows  $\sum :xs = ((\text{mean } xs) * (\text{real } (\text{length } xs)))$ 
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  $xs \neq [] \longrightarrow \text{mean } ((\text{mean } xs)\#xs) = \text{mean } xs$ 
proof
  assume lne:  $xs \neq []$ 
  obtain len where ld:  $len = \text{real } (\text{length } xs)$  by simp
  with lne have lgt0:  $len > 0$  by simp
  hence lnez:  $len \neq 0$  by simp
  from lgt0 have l1nez:  $len + 1 \neq 0$  by simp

```

```

from ld have mean:  $\text{mean } xs = \sum :xs / \text{len}$  unfolding mean-def by simp
with ld real-of-nat-add real-of-one mean-def
have mean  $((\text{mean } xs)\#xs) = (\sum :xs/\text{len} + \sum :xs) / (1+\text{len})$ 
  by simp
also from list-sum-distrib-aux have
   $\dots = (1 + (1/\text{len})) * \sum :xs / (1+\text{len})$  by simp
also with lnez have
   $\dots = (\text{len} + 1) * \sum :xs / (\text{len} * (1+\text{len}))$ 
  apply  $-$ 
  apply (drule mult-divide-mult-cancel-left
    [symmetric, where  $c=\text{len}$  and  $a=(1 + 1 / \text{len}) * \sum :xs$  and  $b=1+\text{len}$ ])
  apply (clarsimp simp:ring-simps)
  done
also from l1nez have  $\dots = \sum :xs / \text{len}$ 
  apply (subst real-mult-commute [where  $z=\text{len}$ ])
  apply (drule mult-divide-mult-cancel-left
    [where  $c=\text{len}+1$  and  $a=\sum :xs$  and  $b=\text{len}$ ])
  by (simp add: mult-ac add-ac)
finally show  $\text{mean } ((\text{mean } xs)\#xs) = \text{mean } xs$  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*]:

$xs \neq [] \wedge 0 < x \wedge 0 < (\text{mean } xs) \longrightarrow 0 < (\text{mean } (x\#xs))$

proof

assume a : $xs \neq [] \wedge 0 < x \wedge 0 < \text{mean } xs$

hence $xgt0$: $0 < x$ **and** $mgt0$: $0 < \text{mean } xs$ **by** *auto*

from a **have** $lxsgt0$: $\text{length } xs \neq 0$ **by** *simp*

from $mgt0$ **have** $xsgt0$: $0 < \sum :xs$

proof $-$

have $\text{mean } xs = \sum :xs / \text{real } (\text{length } xs)$ **unfolding** *mean-def* **by** *simp*

hence $\sum :xs = \text{mean } xs * \text{real } (\text{length } xs)$ **by** *simp*

moreover from $lxsgt0$ **have** $\text{real } (\text{length } xs) > 0$ **by** *simp*

moreover with *calculation lxsgt0 mgt0 real-mult-order* **show** *?thesis* **by** *auto*

qed

with $xgt0$ **have** $\sum :(x\#xs) > 0$ **by** *simp*

thus $0 < (\text{mean } (x\#xs))$

proof $-$

assume $0 < \sum :(x\#xs)$

moreover have $\text{real } (\text{length } (x\#xs)) > 0$ **by** *simp*

ultimately show *?thesis* **unfolding** *mean-def* **by** (*rule divide-pos-pos*)

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 $a \neq m \wedge a : \text{set } xs$
 $\longrightarrow \text{length } (\text{list-neq } (\text{remove1 } a \text{ } xs) \text{ } m) < \text{length } (\text{list-neq } xs \text{ } m)$
(is ?A xs \longrightarrow ?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: $a \neq m$ **and**

a-mem-x-xs: $a : \text{set}(x\#xs)$

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*: $xs \neq []$

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*: $a \neq x$

with *a-mem-x-xs* **have** *a-mem-xs*: $a : \text{set } xs$ **by simp**


```

apply (induct-tac xs)
apply simp
apply clarsimp
done

```

```

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

```

```

lemma listprod-split:
  fixes xs::real list
  shows  $\prod :xs = (\prod :(list-neq\ xs\ m) * \prod :(list-eq\ xs\ m))$ 
apply (induct xs)
apply simp
apply clarsimp
done

```

```

lemma listsum-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:  $m \equiv (mean\ xs)$  and neq:  $noteq \equiv list-neq\ xs\ m$ 
  assumes asum:  $noteq \neq []$ 
  shows  $\exists e. e : set\ noteq \wedge e > m$ 
proof (rule ccontr)
  let ?m =  $(mean\ xs)$ 
  let ?neq =  $list-neq\ xs\ ?m$ 
  let ?eq =  $list-eq\ xs\ ?m$ 
  from list-eq-sum have  $(\sum :?eq) = ?m * (real\ (length\ ?eq))$  by simp
  from asum have neq-ne:  $?neq \neq []$  unfolding m neq .
  assume not-el:  $\neg(\exists e. e : set\ noteq \wedge m < e)$ 
  hence not-el-exp:  $\neg(\exists e. e : set\ ?neq \wedge ?m < e)$  unfolding m neq .
  hence  $\forall e. \neg(e : set\ ?neq) \vee \neg(e > ?m)$  by simp
  hence  $\forall e. e : set\ ?neq \longrightarrow \neg(e > ?m)$  by blast
  hence  $\forall e. e : set\ ?neq \longrightarrow e \leq ?m$  by (simp add: linorder-not-less)
  hence  $\forall e. e : set\ ?neq \longrightarrow e < ?m$  by (simp add: order-le-less)

```

with *prems* *listsum-mono-lt* **have** $(\sum : ?neg) < ?m * (\text{real } (\text{length } ?neg))$ **by** *blast*
hence
 $(\sum : ?neg) + (\sum : ?eq) < ?m * (\text{real } (\text{length } ?neg)) + (\sum : ?eq)$ **by** *simp*
also have
 $\dots = (?m * ((\text{real } (\text{length } ?neg) + (\text{real } (\text{length } ?eq))))$
by (*simp add:ring-simps*)
also have
 $\dots = (?m * (\text{real } (\text{length } xs)))$
apply (*subst real-of-nat-add [symmetric]*)
by (*simp add: listsum-length-split [symmetric]*)
also have
 $\dots = \sum : xs$
by (*simp add: list-sum-mean [symmetric]*)
also from *not-el* **calculation** **show** *False* **by** (*simp only: listsum-split [symmetric]*)
qed

lemma *pick-one-lt*:

fixes *xs::real list* **and** *m::real*
defines *m*: $m \equiv (\text{mean } xs)$ **and** *neg*: $\text{noteq} \equiv \text{list-ineq } xs \ m$
assumes *asum*: $\text{noteq} \neq []$
shows $\exists e. e : \text{set } \text{noteq} \wedge e < m$
proof (*rule ccontr*) — *reductio ad absurdum*
let $?m = (\text{mean } xs)$
let $?neg = \text{list-ineq } xs \ ?m$
let $?eq = \text{list-eq } xs \ ?m$
from *list-eq-sum* **have** $(\sum : ?eq) = ?m * (\text{real } (\text{length } ?eq))$ **by** *simp*
from *asum* **have** $\text{neg-ne: } ?neg \neq []$ **unfolding** *m neg* .
assume *not-el*: $\neg(\exists e. e : \text{set } \text{noteq} \wedge m > e)$
hence *not-el-exp*: $\neg(\exists e. e : \text{set } ?neg \wedge ?m > e)$ **unfolding** *m neg* .
hence $\forall e. \neg(e : \text{set } ?neg) \vee \neg(e < ?m)$ **by** *simp*
hence $\forall e. e : \text{set } ?neg \longrightarrow \neg(e < ?m)$ **by** *blast*
hence $\forall e. e : \text{set } ?neg \longrightarrow e \geq ?m$ **by** (*simp add: linorder-not-less*)
hence $\forall e. e : \text{set } ?neg \longrightarrow e > ?m$ **by** (*auto simp: order-le-less*)
with *prems listsum-mono-gt* **have** $(\sum : ?neg) > ?m * (\text{real } (\text{length } ?neg))$ **by**
blast
hence
 $(\sum : ?neg) + (\sum : ?eq) > ?m * (\text{real } (\text{length } ?neg)) + (\sum : ?eq)$ **by** *simp*
also have
 $(?m * (\text{real } (\text{length } ?neg)) + (\sum : ?eq)) =$
 $(?m * (\text{real } (\text{length } ?neg)) + (?m * (\text{real } (\text{length } ?eq))))$
by *simp*
also have
 $\dots = (?m * ((\text{real } (\text{length } ?neg) + (\text{real } (\text{length } ?eq))))$
by (*simp add:ring-simps*)
also have
 $\dots = (?m * (\text{real } (\text{length } xs)))$
apply (*subst real-of-nat-add [symmetric]*)
by (*simp add: listsum-length-split [symmetric]*)
also have

```

... =  $\sum :xs$ 
  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

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  $\Rightarrow$  nat where
het l = length (list-neq l (mean l))

```

lemma *het-gt-0-imp-noteq-ne*: $het\ l > 0 \implies list\ neq\ l\ (mean\ l) \neq []$

unfolding *het-def* **by** *simp*

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

definition

```

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

```

γ -*eq* is transitive and symmetric.

lemma *γ -eq-sym*: γ -eq (a,b) = γ -eq (b,a)

unfolding *γ -eq-def* **by** *auto*

lemma *γ -eq-trans*:

```

 $\gamma$ -eq (x,y)  $\implies$   $\gamma$ -eq (y,z)  $\implies$   $\gamma$ -eq (x,z)

```

unfolding *γ -eq-def* **by** *simp*

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

definition

```

pos :: real list  $\Rightarrow$  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 \implies xs $\neq []$ **unfolding** *pos-def* **by** *auto*

lemma *pos-cons* [simp]:

```

xs  $\neq [] \longrightarrow$  pos (x#xs) =

```

(if $(x > 0)$ then $\text{pos } xs$ else False)

(is $?P x xs$ is $?A xs \longrightarrow ?S x xs$)

proof (*simp add: split-if, rule impI*)

assume $xsne: xs \neq []$

hence $pxs\text{-simp}$:

$\text{pos } xs = (\forall e. e : \text{set } xs \longrightarrow e > 0)$

unfolding pos-def **by** *simp*

show

$(0 < x \longrightarrow \text{pos } (x \# xs) = \text{pos } xs) \wedge$

$(\neg 0 < x \longrightarrow \neg \text{pos } (x \# xs))$

proof

{

assume $xgt0: 0 < x$

{

assume $pxs: \text{pos } xs$

with $pxs\text{-simp}$ **have** $\forall e. e : \text{set } xs \longrightarrow e > 0$ **by** *simp*

with $xgt0$ **have** $\forall e. e : \text{set } (x \# xs) \longrightarrow e > 0$ **by** *simp*

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

}

moreover

{

assume $pxxs: \text{pos } (x \# xs)$

hence $\forall e. e : \text{set } (x \# xs) \longrightarrow e > 0$ **unfolding** pos-def **by** *simp*

hence $\forall e. e : \text{set } xs \longrightarrow e > 0$ **by** *simp*

with $xsne$ **have** $\text{pos } xs$ **unfolding** pos-def **by** *simp*

}

ultimately **have** $\text{pos } (x \# xs) = \text{pos } xs$

apply $-$

apply (*rule iffI*)

apply *auto*

done

}

thus $0 < x \longrightarrow \text{pos } (x \# xs) = \text{pos } xs$ **by** *simp*

next

{

assume $xngt0: \neg (0 < x)$

{

assume $pxs: \text{pos } xs$

with $pxs\text{-simp}$ **have** $\forall e. e : \text{set } xs \longrightarrow e > 0$ **by** *simp*

with $xngt0$ **have** $\neg (\forall e. e : \text{set } (x \# xs) \longrightarrow e > 0)$ **by** *auto*

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

}

moreover

{

assume $pxxs: \neg \text{pos } xs$

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

hence $\neg (\forall e. e : \text{set } (x \# xs) \longrightarrow e > 0)$ **by** *auto*

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

}

```

      ultimately have  $\neg \text{pos } (x\#xs)$  by auto
    }
  thus  $\neg 0 < x \longrightarrow \neg \text{pos } (x \# xs)$  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 $(\text{remove1 } a \ xs) \neq []$ *pos xs* **shows** $\text{pos } (\text{remove1 } a \ xs)$

proof –

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

from *pl* **have** $xs \neq []$ **by** (*rule pos-imp-ne*)

with *pl* *pos-def* **have** $\forall x. x : \text{set } xs \longrightarrow x > 0$ **by** *simp*

hence $\forall x. x : \text{set } (\text{remove1 } a \ xs) \longrightarrow x > 0$

using *set-remove1-subset[of - xs]* **by**(*blast*)

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

qed

lemma *pos-mean*: $\text{pos } xs \implies \text{mean } xs > 0$

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

with *Cons(2)* **have** $x > 0$ **by**(*simp*)

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

thus *?thesis* **by** *simp*

next

assume *xsne*: $xs \neq []$

show *?thesis*

proof *cases*

assume *pxs*: $\text{pos } xs$

with *Cons(1)* **have** *z-le-mxs*: $0 < \text{mean } xs$ **by**(*simp*)

{

assume *ass*: $x > 0$

with *ass z-le-mxs xsne* **have** $0 < \text{mean } (x\#xs)$

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  $\neg(0 < x)$ 
    with xsne pos-cons have  $\text{pos } (x\#xs) = \text{False}$  by simp
    with Cons(2) show ?thesis by simp
  qed
}
ultimately have  $0 < \text{mean } (x\#xs)$  by simp
thus ?thesis by simp
next
  assume npxs:  $\neg \text{pos } xs$ 
  with xsne pos-cons have  $\text{pos } (x\#xs) = \text{False}$  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 $(\text{mean } x) ^{(\text{length } x)}$.

lemma *listprod-het0*:

shows $x \neq [] \wedge \text{het } x = 0 \implies \prod :x = (\text{mean } x) ^{(\text{length } x)}$

proof –

assume $x \neq [] \wedge \text{het } x = 0$

hence *xne: $x \neq []$* **and** *hetx: $\text{het } x = 0$* **by auto**

from *hetx* **have** *lz: $\text{length } (\text{list-neq } x (\text{mean } x)) = 0$* **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 *list-eq-prod* **have**

$\dots = (\text{mean } x) ^{(\text{length } (\text{list-eq } x (\text{mean } x)))}$ **by simp**

also with *calculation lz 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 $pos\ x \wedge x \neq [] \wedge het\ x = 0 \implies gmean\ x = mean\ x$
proof –
 assume $ass: pos\ x \wedge x \neq [] \wedge het\ x = 0$
 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 *listprod-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 $\dots = mean\ x$ 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
 $m: m \equiv mean\ xs$ and
 $neq: noteq \equiv list\ neq\ xs\ m$ and
 $eq: eq \equiv list\ eq\ xs\ m$
 assumes $pos\ xs: pos\ xs$ and $het\ gt\ 0: het\ xs > 0$
 shows
 $\exists xs'. gmean\ xs' > gmean\ xs \wedge \gamma\ eq\ (xs', xs) \wedge$
 $het\ xs' < het\ xs \wedge pos\ xs'$

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* 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 $prems$ *pick-one-gt neqne* obtain α where
 $\alpha\ def: \alpha : set\ noteq \wedge \alpha > m$ **unfolding** *neq m* by *auto*
 from $prems$ *pick-one-lt neqne* obtain β 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 $prems$ have $el\ neq: \beta \neq \alpha$ by *simp*

from *neqne neq* **have** *xsne*: $xs \neq []$ **by** *auto*

from *prems* **have** *β mem*: $\beta : \text{set } xs$ **by** (*auto simp: neq*)

from *prems* **have** *α mem*: $\alpha : \text{set } xs$ **by** (*auto simp: neq*)

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

— remove these elements from *xs*, and insert two new elements

obtain *left-over* **where** *lo*: $\text{left-over} = (\text{remove1 } \beta (\text{remove1 } \alpha xs))$ **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 α -pos β -pos α -gt β -lt* **have** *b-pos*: $b > 0$ **by** *simp*

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

from *el-neq β mem α mem* **have** $\beta : \text{set } xs \wedge \alpha : \text{set } xs \wedge \beta \neq \alpha$ **by** *simp*

hence $\alpha : \text{set } (\text{remove1 } \beta xs) \wedge \beta : \text{set } (\text{remove1 } \alpha xs)$ **by** (*auto simp add: in-set-remove1*)

moreover **hence** $(\text{remove1 } \alpha xs) \neq [] \wedge (\text{remove1 } \beta xs) \neq []$ **by** (*auto*)

ultimately **have**

mem : $\alpha : \text{set } (\text{remove1 } \beta xs) \wedge \beta : \text{set } (\text{remove1 } \alpha xs) \wedge$
 $(\text{remove1 } \alpha xs) \neq [] \wedge (\text{remove1 } \beta xs) \neq []$ **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 α 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 prems lo bdef α mem β mem*

have $\sum : \text{new-list} = \sum : xs$

apply *clarsimp*

apply (*subst listsum-rmv1*)

apply *simp*

apply (*subst listsum-rmv1*)

apply *simp*

apply *clarsimp*

done

moreover **from** *lo nl β mem α mem mem* **have**

leq: $\text{length new-list} = \text{length } xs$

```

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: 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 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 (rule real-mult-order)

```

```

  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

```

```

apply clarsimp

```

```

apply (rule real-mult-less-mono2)

```

```

apply assumption

```

```

apply assumption

```

```

done

```

```

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

```

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

from α -def **have** α -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**

\dots = *length* (*list-neq* ($m\#b\#(\text{left-over})$) m)

by *simp*

also with *bm* **have**

\dots = *length* (*list-neq left-over* m)

by *simp*

also with *lo* β -def α -def **have**

\dots = *length* (*list-neq* (*remove1* β (*remove1* α xs)) m)

by *simp*

also from β -ne- m β -mem-rmv- α $\text{rmv-}\alpha$ -ne **have**

\dots < *length* (*list-neq* (*remove1* α xs) m)

apply —

apply (*rule list-neq-remove1*)

by *simp*

also from α -mem α -ne- m *xsne* **have**

\dots < *length* (*list-neq xs* m)

apply —

apply (*rule list-neq-remove1*)

by *simp*

also with m *het-def* **have** \dots = *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**

\dots = *length* (*list-neq* ($m\#b\#(\text{left-over})$) m)

by *simp*

also with *bnm* **have**

\dots = *length* ($b\#(\text{list-neq left-over } m)$)

by *simp*

also have

```

... = 1 + length (list-neq left-over m)
by simp
also with lo  $\beta$ -def  $\alpha$ -def have
... = 1 + length (list-neq (remove1  $\beta$  (remove1  $\alpha$  xs)) m)
by simp
also from  $\beta$ -ne-m  $\beta$ -mem-rmv- $\alpha$  rmv- $\alpha$ -ne have
... < 1 + length (list-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-list  $\leq$  length (list-neq (remove1  $\alpha$  xs) m)
by simp
also from  $\alpha$ mem  $\alpha$ -ne-m xsne 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 .
qed

```

— thus thesis by existence of newlist

```

from  $\gamma$ -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  $\forall x. p = \text{het } x \wedge p > 0 \wedge \text{pos } x \longrightarrow$ 
 $(\exists y. \text{gmean } y > \text{gmean } x \wedge \gamma\text{-eq } (x,y) \wedge \text{het } y = 0 \wedge \text{pos } y)$ 
(is ?Q p is  $\forall x. (?A x p \longrightarrow ?S x)$ )

```

proof (*induct p rule: nat-less-induct*)

```
fix n
```

```
assume ind:  $\forall m < n. ?Q m$ 
```

```
{
```

```
  fix x
```

```
  assume ass: ?A x n
```

```
  hence het x > 0 and pos x by auto
```

```
  with new-list-gt-gmean have
```

```
     $\exists y. \text{gmean } y > \text{gmean } x \wedge \gamma\text{-eq } (x,y) \wedge \text{het } y < \text{het } x \wedge \text{pos } y$ 
```

```
  apply -
```

```
  apply (drule meta-spec [of - x])
```

```
  apply (drule meta-mp)
```

```
    apply assumption
```

```
  apply (drule meta-mp)
```

```
    apply assumption
```

```
  apply (subst(asm)  $\gamma$ -eq-sym)
```

```

    apply simp
  done
  then obtain  $\beta$  where
     $\beta$ -def:  $gmean\ \beta > gmean\ x \wedge \gamma$ -eq  $(x, \beta) \wedge het\ \beta < het\ x \wedge pos\ \beta ..$ 
  then obtain  $b$  where  $b$ -def:  $b = het\ \beta$  by simp
  with  $ass\ \beta$ -def have  $b < n$  by auto
  with  $ind$  have  $?Q\ b$  by simp
  with  $\beta$ -def have
     $ind2: b = het\ \beta \wedge 0 < b \wedge pos\ \beta \longrightarrow$ 
     $(\exists y. gmean\ \beta < gmean\ y \wedge \gamma$ -eq  $(\beta, y) \wedge het\ y = 0 \wedge pos\ y)$  by simp
  {
    assume  $\neg(0 < b)$ 
    hence  $b=0$  by simp
    with  $b$ -def have  $het\ \beta = 0$  by simp
    with  $\beta$ -def have  $?S\ x$  by auto
  }
  moreover
  {
    assume  $0 < b$ 
    with  $b$ -def  $ind2\ \beta$ -def have  $?S\ \beta$  by simp
    then obtain  $\gamma$  where
       $gmean\ \beta < gmean\ \gamma \wedge \gamma$ -eq  $(\beta, \gamma) \wedge het\ \gamma = 0 \wedge pos\ \gamma ..$ 
    with  $\beta$ -def have  $gmean\ x < gmean\ \gamma \wedge \gamma$ -eq  $(x, \gamma) \wedge het\ \gamma = 0 \wedge pos\ \gamma$ 
      apply clarsimp
      apply (rule  $\gamma$ -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 $gmean\ z \leq mean\ z$

proof –

from $\langle pos\ z \rangle$ have $zne: z \neq []$ by (rule *pos-imp-ne*)

show $gmean\ z \leq mean\ z$

proof cases

assume $het\ z = 0$

with $\langle pos\ z \rangle\ zne\ het$ -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**
 $\exists y. gmean\ y > gmean\ z \wedge \gamma\text{-eq}\ (z, y) \wedge het\ y = 0 \wedge pos\ y$ **by** *auto*
then obtain α **where**
 $gmean\ \alpha > gmean\ z \wedge \gamma\text{-eq}\ (z, \alpha) \wedge het\ \alpha = 0 \wedge pos\ \alpha ..$
with *het-base* $\gamma\text{-eq-def}$ *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
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.

```
types vector = (nat $\Rightarrow$ real) * nat
```

definition

$ith :: vector \Rightarrow nat \Rightarrow real$ ($((-)) [80,100] 100$) **where**
 $ith\ v\ i = fst\ v\ i$

definition

$vlen :: vector \Rightarrow nat$ **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

$dot :: vector \Rightarrow vector \Rightarrow real$ (**infixr** \cdot 60) **where**
 $dot\ a\ b = (\sum\ j \in \{1..(vlen\ a)\}. a_j * b_j)$

definition

$norm :: vector \Rightarrow real$ ($(||-\|$ 100) **where**
 $norm\ v = sqrt\ (\sum\ j \in \{1..(vlen\ v)\}. v_j^2)$

notation (HTML output)

$norm\ (||-\| 100)$

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** $\dots = sqrt\ (\sum\ j \in \{1..(vlen\ v)\}. v_j^2)$ **by simp**
also have $\dots = \|v\|$ **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 $\forall j. v_j^2 \geq 0$ **unfolding ith-def by auto**
hence $\forall j \in \{1..(vlen\ v)\}. v_j^2 \geq 0$ **by simp**
with $setsum-nonneg$ **have** $(\sum\ j \in \{1..(vlen\ v)\}. v_j^2) \geq 0$.
with $real-sqrt-ge-zero$ **have** $sqrt\ (\sum\ j \in \{1..(vlen\ v)\}. v_j^2) \geq 0$.
thus $?thesis$ **unfolding norm-def** .

qed

We now prove an intermediary lemma regarding double summation.

lemma double-sum-aux:

```

fixes f::nat ⇒ real
shows
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
  (∑ k∈{1..n}. (∑ j∈{1..n}. (f k * g j + f j * g k) / 2))
proof -
have
  2 * (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) +
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j))
  by simp
also have
  ... =
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) +
  (∑ k∈{1..n}. (∑ j∈{1..n}. f j * g k))
  by (simp only: double-sum-equiv)
also have
  ... =
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j + f j * g k))
  by (auto simp add: setsum-addf)
finally have
  2 * (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j + f j * g k)) .
hence
  (∑ k∈{1..n}. (∑ j∈{1..n}. f k * g j)) =
  (∑ k∈{1..n}. (∑ j∈{1..n}. (f k * g j + f j * g k)))*(1/2)
  by auto
also have
  ... =
  (∑ k∈{1..n}. (∑ j∈{1..n}. (f k * g j + f j * g k)*(1/2)))
  by (simp add: setsum-right-distrib real-mult-commute)
finally show ?thesis by (auto simp add: inverse-eq-divide)
qed

```

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

theorem *CauchySchwarzReal*:

```

fixes x::vector
assumes vlen x = vlen y
shows |x·y| ≤ ||x||*||y||
proof -
have 0 ≤ |x·y| by simp
moreover have 0 ≤ ||x||*||y||
  by (auto simp add: norm-pos intro: mult-nonneg-nonneg)
moreover have |x·y|^2 ≤ (||x||*||y||)^2
proof -

```

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

```

have (||x||*||y||)^2 - |x·y|^2 ≥ 0
proof -

```

```

obtain  $n$  where  $nx: n = \text{vlen } x$  by simp
with  $(\text{vlen } x = \text{vlen } y)$  have  $ny: n = \text{vlen } y$  by simp
{

```

Some preliminary simplification rules.

```

have  $\forall j \in \{1..n\}. x_j^2 \geq 0$  by simp
hence  $(\sum j \in \{1..n\}. x_j^2) \geq 0$  by (rule setsum-nonneg)
hence  $xp: (\text{sqrt } (\sum j \in \{1..n\}. x_j^2))^2 = (\sum j \in \{1..n\}. x_j^2)$ 
by (rule real-sqrt-pow2)

```

```

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

```

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

```

have
   $(\|x\|* \|y\|)^2 = \|x\|^2 * \|y\|^2$ 
  by (simp add: real-sq-exp)
also from  $nx ny$  have
   $\dots = (\text{sqrt } (\sum j \in \{1..n\}. x_j^2))^2 * (\text{sqrt } (\sum j \in \{1..n\}. y_j^2))^2$ 
  unfolding norm-def by auto
also from  $xp yp$  have
   $\dots = (\sum j \in \{1..n\}. x_j^2) * (\sum j \in \{1..n\}. y_j^2)$ 
  by simp
also from setsum-product-expand have
   $\dots = (\sum k \in \{1..n\}. (\sum j \in \{1..n\}. (x_k^2) * (y_j^2)))$  .
finally have
   $(\|x\|* \|y\|)^2 = (\sum k \in \{1..n\}. (\sum j \in \{1..n\}. (x_k^2) * (y_j^2)))$  .
}
moreover
{

```

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

```

have
   $|x \cdot y|^2 = (x \cdot y)^2$ 
  by simp
also from  $nx$  have
   $\dots = (\sum j \in \{1..n\}. x_j * y_j)^2$ 
  unfolding dot-def by simp
also from real-sq have
   $\dots = (\sum j \in \{1..n\}. x_j * y_j) * (\sum j \in \{1..n\}. x_j * y_j)$ 
  by simp
also from setsum-product-expand have
   $\dots = (\sum k \in \{1..n\}. (\sum j \in \{1..n\}. (x_k * y_k) * (x_j * y_j)))$ 
  by simp
finally have
   $|x \cdot y|^2 = (\sum k \in \{1..n\}. (\sum j \in \{1..n\}. (x_k * y_k) * (x_j * y_j)))$  .
}

```

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

ultimately have

$$\begin{aligned} & (\|x\|*|y|)^2 - |x \cdot y|^2 = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (x_k^2)*(y_j^2))) - \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (x_k*y_k)*(x_j*y_j))) \end{aligned}$$

by simp

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. ((x_k^2*y_j^2) + (x_j^2*y_k^2))/2)) - \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (x_k*y_k)*(x_j*y_j))) \end{aligned}$$

by (simp only: double-sum-ax)

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. ((x_k^2*y_j^2) + (x_j^2*y_k^2))/2 - (x_k*y_k)*(x_j*y_j))) \end{aligned}$$

by (auto simp add: setsum-subtractf)

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (inverse 2)*2* \\ & (((x_k^2*y_j^2) + (x_j^2*y_k^2))*(1/2) - (x_k*y_k)*(x_j*y_j)))) \end{aligned}$$

by auto

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (inverse 2)*(2* \\ & (((x_k^2*y_j^2) + (x_j^2*y_k^2))*(1/2) - (x_k*y_k)*(x_j*y_j)))))) \end{aligned}$$

by (simp only: real-mult-assoc)

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (inverse 2)* \\ & (((x_k^2*y_j^2) + (x_j^2*y_k^2))*2*(inverse 2) - 2*(x_k*y_k)*(x_j*y_j)))) \end{aligned}$$

by (auto simp add: real-add-mult-distrib real-mult-assoc mult-ac)

also have

$$\begin{aligned} \dots = \\ & (\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (inverse 2)* \\ & (((x_k^2*y_j^2) + (x_j^2*y_k^2)) - 2*(x_k*y_k)*(x_j*y_j)))) \end{aligned}$$

by (simp only: real-mult-assoc, simp)

also have

$$\begin{aligned} \dots = \\ & (inverse 2)*(\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. \\ & (((x_k^2*y_j^2) + (x_j^2*y_k^2)) - 2*(x_k*y_k)*(x_j*y_j)))) \end{aligned}$$

by (simp only: setsum-right-distrib)

also have

$$\begin{aligned} \dots = \\ & (inverse 2)*(\sum_{k \in \{1..n\}}. (\sum_{j \in \{1..n\}}. (x_k*y_j - x_j*y_k)^2)) \end{aligned}$$

also have ... ≥ 0

proof –

{
fix k::nat
have $\forall j \in \{1..n\}. (x_k*y_j - x_j*y_k)^2 \geq 0$ **by simp**

hence $(\sum_{j \in \{1..n\}} (x_k * y_j - x_j * y_k)^2) \geq 0$ **by** (*rule setsum-nonneg*)
 }
 hence $\forall k \in \{1..n\}. (\sum_{j \in \{1..n\}} (x_k * y_j - x_j * y_k)^2) \geq 0$ **by** *simp*
 hence $(\sum_{k \in \{1..n\}} (\sum_{j \in \{1..n\}} (x_k * y_j - x_j * y_k)^2)) \geq 0$
 by (*rule setsum-nonneg*)
 thus *?thesis* **by** *simp*
 qed
 finally show $(\|x\| * \|y\|)^2 - |x \cdot y|^2 \geq 0$.
 qed
 thus *?thesis* **by** *simp*
 qed
 ultimately show *?thesis* **by** (*rule real-sq-order*)
 qed
 end