Cauchy-Schwarz for ACL2(r) Abstract Vector Spaces

(How `Smtlink` makes proofs about algebraic structures easy)

Carl Kwan, Yan Peng, Mark R. Greenstreet

16th International Workshop on the ACL2 Theorem Prover and Its Applications

Introduction

Previously:

- Proof of Cauchy-Schwarz for real vector spaces
- Proven "by hand" in ACL2(r)
- 54 ACL2(r) lemmas involving algebraic manipulations

Today:

- Proof of Cauchy-Schwarz for abstract vector spaces
- Proven with the help of `Smtlink` in ACL2(r)
- 6 ACL2(r) lemmas involving algebraic manipulations

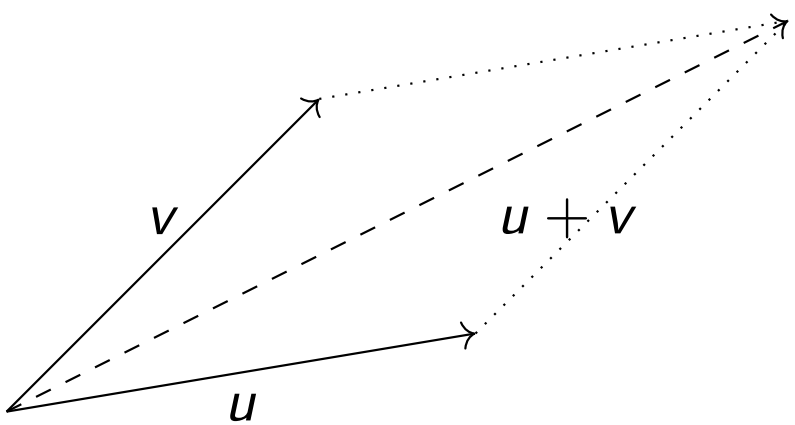
Outline:

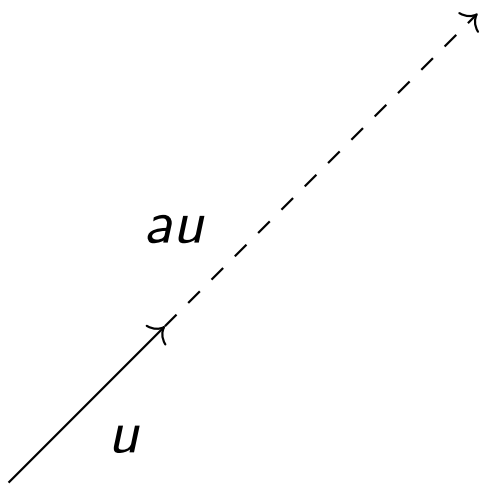
- Quick review of inner product spaces and Cauchy-Schwarz
- `Smtlink` proof
  - If you can understand the hand proof, you can write the `Smtlink` proof
- Conclusion

Real vector spaces

Real vector space $(\mathbb{R}^n, \mathbb{R}, \cdot, +)$:

- $+ : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is associative and commutative
- Identity elements: $\vec{0} + v = v$ and $1v = v$
- Inverse elements: $v + (-v) = \vec{0}$
- Compatibility: $a(bv) = (ab)v$
- Distributivity (two ways):
  $a(u + v) = au + uv$ and $(a + b)v = av + bv$





Inner Product Spaces

Inner product space = Vector space + Inner product
$\langle -, - \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

- Positive-definiteness: $\langle u, u \rangle \geq 0$ and $\langle u, u \rangle = 0 \iff u = 0$
- Symmetry[1]: $\langle u, v \rangle = \langle v, u \rangle$
- Linearity of the first coordinate:
  $\langle au + v, w \rangle = a\langle u, w \rangle + \langle v, w \rangle$

For $\mathbb{R}^n$ and $u = (u_i)_{i=1}^n$, $v = (v_i)_{i=1}^n$, use the dot product:

$$\langle u, v \rangle = \sum_{i=1}^{n} u_i v_i$$

Replace $\mathbb{R}^n$ with V by encapsulating formalisation of $(\mathbb{R}^n, \mathbb{R}, \cdot, +, \langle -, - \rangle)$. In particular, definitions of $\cdot$, $+$, and $\langle -, - \rangle$ are suppressed.

[1] when over $\mathbb{R}$

Cauchy-Schwarz

Theorem 1 (The Cauchy-Schwarz Inequality)

Let $u, v \in \mathbb{R}^n$. Then

$$|\langle u, v \rangle|^2 \leq \langle u, u \rangle \langle v, v \rangle \tag{CS1}$$

or, equivalently,

$$|\langle u, v \rangle| \leq \|u\| \cdot \|v\| \tag{CS2}$$

with equality iff u, v are linearly dependent. Here $\|u\| := \sqrt{\langle u, u \rangle}$.

How to prove it? Clever set-up + basic algebraic manipulations

Proof of $|\langle u, v\rangle|^2 \leq \langle u, u\rangle\langle v, v\rangle$

How to prove it? Clever set-up + basic algebraic manipulations:

Proof (sketch).

From positive-definiteness:

$$0 \leq \langle u - av, u - av\rangle = \langle u, u\rangle - 2a\langle u, v\rangle + a^2\langle v, v\rangle.$$

Set $a = \frac{\langle u,v\rangle}{\langle v,v\rangle}$ and rearrange (a bunch) to get

$$0 \leq \cdots = \langle u, u\rangle + \langle u, v\rangle \left(-2\frac{\langle u, v\rangle}{\langle v, v\rangle} + \frac{\langle u, v\rangle}{\langle v, v\rangle}\right) = \langle u, u\rangle - \frac{\langle u, v\rangle^2}{\langle v, v\rangle}.$$

□

Only requires properties of $\cdot$, $+$, and $\langle -, -\rangle$ (`inner-prod`).

```
(encapsulate
  (((scalar-vector-prod * *) => *)
   ((vector-add * *) => *)
   ((inner-prod * *) => *) ...) ...)
```

Smtlink example

If $x \in \mathbb{R}$ with $0 < x < 1$, then $x^2 < x$.

```
(defun x^2 (x)
  (* x x))

(defthm x^2-<-x
  (implies (and (realp x) (< 0 x) (< x 1))
           (< (x^2 x) x))
  :hints (("Goal" :smtlink nil)))
```

Idea: provide Smtlink with properties of $(V, \mathbb{R}, \cdot, +, \langle -, - \rangle)$ and let it "search" for the desired expression (by showing the negation of the desired statement is unsat).

Smtlink proof

$$u, v \in V \ \wedge \ a \in \mathbb{R} \ \wedge \ u = av \implies \langle u, v \rangle^2 = \langle u, u \rangle \langle v, v \rangle$$

```
:smtlink(...
          :functions(
           (scalar-vector-prod
             :formals ((a realp) (v a-vec-p))
             :returns ((prod a-vec-p)) ...)
            (inner-prod
             :formals ((u a-vec-p) (v a-vec-p))
             :returns ((prod realp)) ...)
          :hypotheses(
            ((equal (inner-prod (scalar-vector-prod a v)
                                (scalar-vector-prod a v))
                    (* (* a a) (inner-prod v v))))
            ((equal (inner-prod (scalar-vector-prod a v) v)
                    (* a (inner-prod v v)))))) ...
```

- `:smtlink` hints are all that is passed to Z3
- `:hypotheses` list statements Z3 can assume to be true but must be true in the ACL2 logical world
- If you can write the hand proof, the `Smtlink` proof is easy

Smtlink proof

Smtlink requires type recognizers in the hypothesis for each variable that shows up in the conclusion. But V is abstract! Encapsulate a type recogniser

```
(local (define a-vec-p (v) ... (real-vec-p v)))
```

and use Z3's theory of uninterpreted functions and sorts to reason about abstract vectors. Basic example[2]:

```
(encapsulate
  (((abstract-p *) => *))
  (local
    (defun abstract-p (x)
        (any-p x))))

(defthm abstract-example
  (implies (abstract-p x)
           (equal x x))
  :hints (("GOAL" :smtlink (:abstract (abstract-p))))
  :rule-classes nil)
```

[2] [books]/projects/smtlink/examples/examples.lisp

Results:

- Proof of Cauchy-Schwarz for ACL2 abstract vector spaces + conditions for equality. Other theorem provers[3]:

HOL Light	Isabelle	Metamath	Mizar	Proof Power	PVS
Real	Real	Hilbert?	Abstract?	?	?

- SMT techniques offers dramatic benefits even (especially) when the theorems to be proved are over user-defined structures (6 lemmas vs. 54 lemmas)
- Encapsulation and theories of uninterpreted functions/sorts enable the use of `Smtlink` and SMT solvers for algebraicly reasoning about abstract structures.
- `Smtlink` makes proofs about algebraic structures easy

[3] `https://www.cs.ru.nl/~freek/100/`

Future

- Cauchy-Schwarz appears in many pure and applied areas (e.g. probability theory, extremal graph theory, machine learning, functional analysis, financial mathematics, etc.)
- `Smtlink` will be very helpful for proving identities in abstract structures we want to explore in the future (e.g. matrix algebra, numerical linear algebra, etc.)

Statement of Cauchy-Schwarz

$$\langle u, v \rangle^2 \leq \langle u, u \rangle \langle v, v \rangle$$

```
(defthm cs1
  (implies (vector-compatible u v)
           (b* ((uu (inner-prod u u))
                (uv (inner-prod u v))
                (vv (inner-prod v v)))
             (<= (* uv uv) (* uu vv))))
  :hints(("Goal" :cases ((vector-zero-p v)))))
```

Statement of Cauchy-Schwarz

$$\langle u, v \rangle^2 \leq \langle u, u \rangle \langle v, v \rangle$$

```
(defthm cs1
  (implies (vector-compatible u v)
           (b* ((uu (inner-prod u u))
                (uv (inner-prod u v))
                (vv (inner-prod v v)))
             (<= (* uv uv) (* uu vv))))
  :hints(("Goal" :cases ((vector-zero-p v)))))
```

Thank You!

```
(defthm cs2
 (implies (vector-compatible u v)
          (b* ((uv (inner-prod u v))
               (uu (inner-prod u u))
               (vv (inner-prod v v)))
              (<= (abs uv) (* (acl2-sqrt uu) (acl2-sqrt vv)))))
 :hints (("GOAL" :use ((:instance cs2-iff-cs1)))))

(defthm cs1-equality-implies-linear-dependence
 (b* ((uu (inner-prod u u))
      (uv (inner-prod u v))
      (vv (inner-prod v v)))
     (implies (and (vector-compatible u v)
                   (equal (* uv uv)
                          (* uu vv)))
              (or (vector-zero-p v)
                  (vector-zero-p (vector-add u
                                             (scalar-vector-prod (- (/ uv vv))
                                                                 v)))))))

 (defthm cs2-equality-implies-linear-dependence
   (b* ((uu (inner-prod u u))
        (uv (inner-prod u v))
        (vv (inner-prod v v)))
       (implies (and (vector-compatible u v)
                     (equal (abs uv)
                            (* (acl2-sqrt uu) (acl2-sqrt vv))))
                (or (vector-zero-p v)
                    (vector-zero-p (vector-add u
                                               (scalar-vector-prod (- (/ uv vv))
                                                                   v))))))
   :hints (("GOAL" :use ((:instance cs2-equality-iff-cs1-equality)))))
```

```
(defthm linear-dependence-implies-cs1-equality
 (implies (and (vector-compatible u v)
               (a-vec-p u)
               (a-vec-p v)
               (realp a)
               (equal u (scalar-vector-prod a v)))
          (b* ((uu (inner-prod u u))
               (uv (inner-prod u v))
               (vv (inner-prod v v)))
                     (equal (* uv uv) (* uu vv))))
 :hints
  (("Goal"
    :smtlink(:abstract (a-vec-p)
              :functions((vector-add :formals ((u a-vec-p) (v a-vec-p))
                                     :returns ((sum a-vec-p))
                                     :level 0)
                         (scalar-vector-prod :formals ((a realp) (v a-vec-p))
                                             :returns ((prod a-vec-p))
                                             :level 0)
                         (inner-prod :formals ((u a-vec-p) (v a-vec-p))
                                     :returns ((prod realp))
                                     :level 0)
                         (vector-zero-p :formals ((v a-vec-p))
                                        :returns ((is-z booleanp))
                                        :level 0)
                         (vector-compatible :formals ((u a-vec-p) (v a-vec-p))
                                            :returns ((ok booleanp))
                                            :level 0))
              :hypotheses(((equal (inner-prod (scalar-vector-prod a v)
                                              (scalar-vector-prod a v))
                                  (* (* a a) (inner-prod v v))))
                          ((equal (inner-prod (scalar-vector-prod a v)
                                              v)
                                  (* a (inner-prod v v)))))))))
```

```
(defthm linear-dependence-implies-cs2-equality
 (implies (and (vector-compatible u v)
               (a-vec-p u)
               (a-vec-p v)
               (realp a)
               (equal u (scalar-vector-prod a v)))
          (equal (abs (inner-prod u v))
                 (* (acl2-sqrt (inner-prod u u))
                    (acl2-sqrt (inner-prod v v)))))
 :hints
  (("Goal"
    :smtlink(:abstract (a-vec-p)
             :functions((vector-add :formals ((u a-vec-p) (v a-vec-p))
                                    :returns ((sum a-vec-p))
                                    :level 0)
                        (scalar-vector-prod :formals ((a realp) (v a-vec-p))
                                            :returns ((prod a-vec-p))
                                            :level 0)
                        (inner-prod :formals ((u a-vec-p) (v a-vec-p))
                                    :returns ((prod realp))
                                    :level 0)
                        (vector-zero-p :formals ((v a-vec-p))
                                       :returns ((is-z booleanp))
                                       :level 0)
                        (vector-compatible :formals ((u a-vec-p) (v a-vec-p))
                                           :returns ((ok booleanp))
                                           :level 0))
                        (acl2-sqrt :formals ((sq realp))
                                   :returns ((rt realp))
                                   :level 0))
             :hypotheses(((equal (acl2-sqrt (* (inner-prod u v) (inner-prod u v)))
                                 (abs (inner-prod u v))))
                         ((equal (acl2-sqrt (* (inner-prod u u) (inner-prod v v)))
                                 (* (acl2-sqrt (inner-prod u u))
                                    (acl2-sqrt (inner-prod v v)))))
                         ((equal (* (inner-prod u v) (inner-prod u v))
                                 (* (inner-prod u u) (inner-prod v v)))
                          :hints (:use linear-dependence-implies-cs1-equality))))))
```

Real vs. rational

What if we did this proof in ACL2? (no '(r)')

Yes, we can (and did)! But with some differences.

	$\langle -, - \rangle$	$\lvert\langle u, v\rangle\rvert^2 \leq \langle u, u\rangle\langle v, v\rangle$	$\lvert\langle u, v\rangle\rvert \leq \lVert u\rVert\lVert v\rVert$
ACL2(r)	$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$	Yes	Yes
ACL2	$\mathbb{Q}^n \times \mathbb{Q}^n \to \mathbb{Q}$	Yes	No

Replace `realp` with `rationalp` everywhere inside the encapsulation. Recall $\lVert u\rVert = \sqrt{\langle u, u\rangle}$.