log-fun-ineq-e-weak

$$\forall x \in (0, 12), y \in (-\infty, \infty)$$
$$xy \leq \frac{1}{5} + x \ln(x) + e^{y-1}$$

sin-3425b

$$\forall x \in (0, \infty), y \in (-\infty, \infty)$$
$$(x < y \wedge y^2 < 6) \Rightarrow \frac{\sin(y)}{\sin(x)} \leq 10^{-4} + \frac{y - \frac{1}{6}y^3 + \frac{1}{120}y^5}{x - \frac{1}{6}x^3 + \frac{1}{120}x^5}$$

CONVOI2-sincos

$$\forall t \in (0, \infty), v \in (0, \infty)$$
$$((1.565 + 0.313\,v)\cos(1.16\,t) + (0.01340 + 0.00268\,v)\sin(1.16\,t))\,e^{-1.34\,t}$$
$$-(6.55 + 1.31\,v)\,e^{-0.318\,t} + v + 10 \geq 0$$

ACL2 Proofs of Nonlinear Inequalities with Imandra

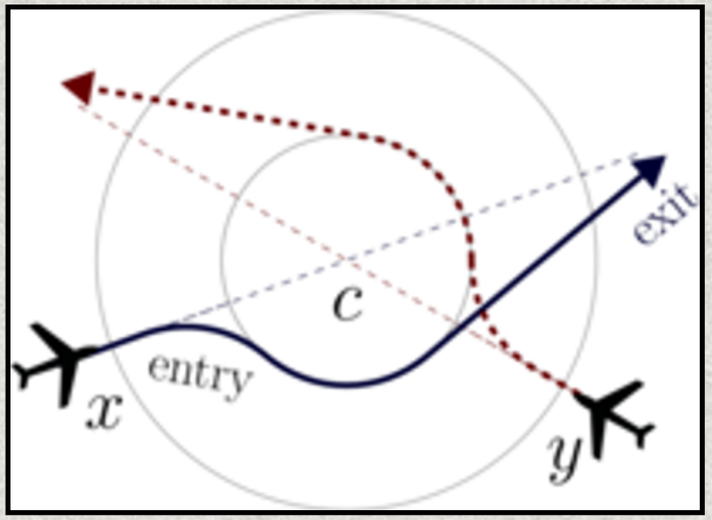
Grant Passmore, Imandra Inc.

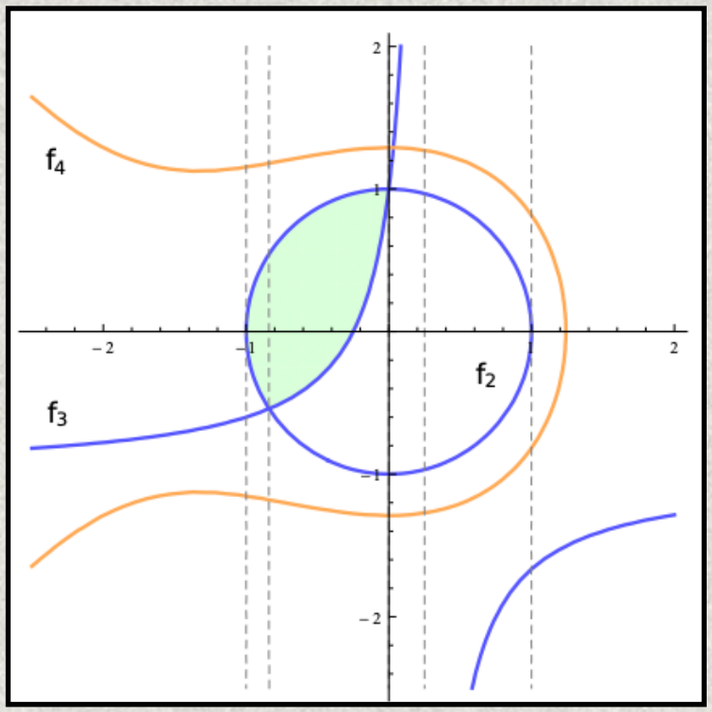
The Big Picture

- A new tool for proving nonlinear (real) inequalities automatically in ACL2
- Leverages a fundamental result in real algebraic geometry - the *Krivine-Stengle Positivstellensatz*
- Implemented and relied upon in Imandra, and now exports checkable ACL2 proof scripts
- Many extensions possible - Would love feedback from anyone who finds it useful!

Nonlinear real arithmetic

- Nonlinear real arithmetic is important in verification of safety-critical systems (and much more besides!)
- Unlike nonlinear arithmetic over N, Z or Q, nonlinear real arithmetic is decidable
- The most widely used decision procedure - CAD - does not produce easily checkable proof objects
- However, we can exploit the Positivstellensatz and convex optimization (SDP) to decide the universal fragment and produce easily checkable proofs





A simple example

Consider one direction of the discriminant criterion for solubility of a quadratic equation:

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

A simple example

Consider one direction of the discriminant criterion for solubility of a quadratic equation:

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

How might we prove it?

Goal

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

Goal

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Goal

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

Goal

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

Consider the polynomial

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

Goal

$$\forall x, a, b, c \in \mathbb{R} \left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0 \right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

Consider the polynomial

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

By assumption, it must be >0.

Goal

$$\forall x, a, b, c \in \mathbb{R}\left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0\right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

Consider the polynomial

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

By assumption, it must be >0.
But it normalizes to 0.

Goal

$$\forall x, a, b, c \in \mathbb{R}\left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0\right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

Consider the polynomial

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

By assumption, it must be >0.
But it normalizes to 0.
Contradiction!

Goal

$$\forall x, a, b, c \in \mathbb{R}\left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0\right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

We call this a *Positivstellensatz certificate:*

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

Goal

$$\forall x, a, b, c \in \mathbb{R}\left(ax^2 + bx + c = 0 \implies b^2 - 4ac \geq 0\right).$$

iff

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is unsat.

Assume

$$ax^2 + bx + c = 0 \wedge 4ac - b^2 > 0$$

is sat.

We call this a *Positivstellensatz certificate:*

$$(4ac - b^2) + (2ax + b)^2 + (-4a)(ax^2 + bx + c)$$

And the Positivstellensatz guarantees their existence! We just have to find them…

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;;
;; Theorem (over R):
;;
;; (IMPLIES (= (+ (* A X X) (* B X) C) 0)
;;          (>= (- (* B B) (* 4 A C)) 0)).
;;
;; Proof found by Imandra in 0.179818 secs.
;; Questions? Contact Grant Passmore (grant@imandra.ai).
;;

(encapsulate ()

;; Preamble

(SET-IGNORE-OK T)
 (SET-IRRELEVANT-FORMALS-OK T)

 (LOCAL (DEFMACRO NEQ (X Y)
          `(OR (< ,X ,Y) (> ,X ,Y))))

 (LOCAL (DEFUN SQUARE (X)
          (* X X)))

 (LOCAL (DEFTHM SQUARE-PSD
          (IMPLIES (RATIONALP X)
                   (>= (SQUARE X) 0))
          :RULE-CLASSES (:LINEAR)))

 (LOCAL (DEFTHM SQUARE-TYPE
          (IMPLIES (RATIONALP X)
                   (RATIONALP (SQUARE X)))
          :RULE-CLASSES (:TYPE-PRESCRIPTION)))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;;
;; Theorem (over R):
;;
;; (IMPLIES (= (+ (* A X X) (* B X) C) 0)
;;          (>= (- (* B B) (* 4 A C)) 0)).
;;
;; Proof found by Imandra in 0.179818 secs.
;; Questions? Contact Grant Passmore (grant@imandra.ai).
;;

(encapsulate ()

;; Preamble

(SET-IGNORE-OK T)
(SET-IRRELEVANT-FORMALS-OK T)

(LOCAL (DEFMACRO NEQ (X Y)
         `(OR (< ,X ,Y) (> ,X ,Y))))

(LOCAL (DEFUN SQUARE (X)
         (* X X)))

(LOCAL (DEFTHM SQUARE-PSD
         (IMPLIES (RATIONALP X)
                  (>= (SQUARE X) 0))
         :RULE-CLASSES (:LINEAR)))

(LOCAL (DEFTHM SQUARE-TYPE
         (IMPLIES (RATIONALP X)
                  (RATIONALP (SQUARE X)))
         :RULE-CLASSES (:TYPE-PRESCRIPTION)))

(LOCAL (IN-THEORY (DISABLE SQUARE)))

(LOCAL (include-book "arithmetic-5/top" :dir :system))

;; Normalized problem polynomials

(LOCAL (DEFUND PROB-0 (A B C X)
         (+ (* A (* X X)) (+ (* B X) C))))

(LOCAL (DEFUND PROB-1 (A B C X)
         (- 0 (- (* B B) (* 4 (* A C))))))

;; Normalized goal expressed using problem polynomials

(LOCAL (DEFUN GOAL (A B C X)
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (NOT (AND (= (PROB-0 A B C X) 0)
                            (> (PROB-1 A B C X) 0))))))

;; Ideal cofactors

(LOCAL (DEFUND IDEAL-CF-0 (A B C X)
         (* -4 A)))

(LOCAL (DEFTHM IDEAL-CF-0-TYPE
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (RATIONALP (IDEAL-CF-0 A B C X)))
         :hints
         (("Goal" :in-theory (enable IDEAL-CF-0)))))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;;
;; Theorem (over R):
;;
;; (IMPLIES (= (+ (* A X X) (* B X) C) 0)
;;          (>= (- (* B B) (* 4 A C)) 0)).
;;
;; Proof found by Imandra in 0.179818 secs.
;; Questions? Contact Grant Passmore (grant@imandra.ai).
;;

(encapsulate ()

;; Preamble

(SET-IGNORE-OK T)
(SET-IRRELEVANT-FORMALS-OK T)

(LOCAL (DEFMACRO NEQ (X Y)
         `(OR (< ,X ,Y) (> ,X ,Y))))

(LOCAL (DEFUN SQUARE (X)
         (* X X)))

(LOCAL (DEFTHM SQUARE-PSD
         (IMPLIES (RATIONALP X)
                  (>= (SQUARE X) 0))
         :RULE-CLASSES (:LINEAR)))

(LOCAL (DEFTHM SQUARE-TYPE
         (IMPLIES (RATIONALP X)
                  (RATIONALP (SQUARE X)))
         :RULE-CLASSES (:TYPE-PRESCRIPTION)))

(LOCAL (IN-THEORY (DISABLE SQUARE)))

(LOCAL (include-book "arithmetic-5/top" :dir :system))

;; Normalized problem polynomials

(LOCAL (DEFUND PROB-0 (A B C X)
         (+ (* A (* X X)) (+ (* B X) C))))

(LOCAL (DEFUND PROB-1 (A B C X)
         (- 0 (- (* B B) (* 4 (* A C))))))

;; Normalized goal expressed using problem polynomials

(LOCAL (DEFUN GOAL (A B C X)
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (NOT (AND (= (PROB-0 A B C X) 0)
                            (> (PROB-1 A B C X) 0))))))

;; Ideal cofactors

(LOCAL (DEFUND IDEAL-CF-0 (A B C X)
         (* -4 A)))

(LOCAL (DEFTHM IDEAL-CF-0-TYPE
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (RATIONALP (IDEAL-CF-0 A B C X)))
         :hints
         (("Goal" :in-theory (enable IDEAL-CF-0)))))

;; Cone cofactors

(LOCAL (DEFUND CONE-CF-0 (A B C X)
         (SQUARE (+ (* 2 (* A X)) B))))

(LOCAL (DEFTHM CONE-CF-0-TYPE
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (RATIONALP (CONE-CF-0 A B C X)))
         :hints
         (("Goal" :in-theory (enable CONE-CF-0)))))

(LOCAL (DEFTHM CONE-CF-0-PSD
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (>= (CONE-CF-0 A B C X) 0))
         :hints
         (("Goal" :in-theory
                  (enable CONE-CF-0 PROB-0 PROB-1)))
         :rule-classes (:linear )))

;; Monoid cofactors

(LOCAL (DEFUND MONOID-CF-0 (A B C X)
         (- 0 (- (* B B) (* 4 (* A C))))))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;; Positivstellensatz certificate

 (LOCAL (DEFUN CERT (A B C X)
          (+ (MONOID-CF-0 A B C X)
             (CONE-CF-0 A B C X)
             (* (IDEAL-CF-0 A B C X) (PROB-0 A B C X))))))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;; Contradictory results on the sign of the certificate

 (LOCAL (DEFTHMD CERT-KEY
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (= (CERT A B C X) 0))
         :hints
         (("Goal" :in-theory
                  (enable SQUARE
                          CERT
                          PROB-0
                          PROB-1
                          IDEAL-CF-0 CONE-CF-0 MONOID-CF-0)))))

(LOCAL (DEFTHM CERT-CONTRA-M-0
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (> (MONOID-CF-0 A B C X) 0))
         :hints
         (("Goal" :in-theory
                  (enable SQUARE
                          CERT
                          PROB-0
                          PROB-1
                          IDEAL-CF-0 CONE-CF-0 MONOID-CF-0)))
         :rule-classes (:linear )))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;; Contradictory results on the sign of the certificate

 (LOCAL (DEFTHMD CERT-KEY
         (IMPLIES (AND (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (= (CERT A B C X) 0))
         :hints
         (("Goal" :in-theory
                  (enable SQUARE
                          CERT
                          PROB-0
                          PROB-1
                          IDEAL-CF-0 CONE-CF-0 MONOID-CF-0)))))

(LOCAL (DEFTHM CERT-CONTRA-M-0
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (> (MONOID-CF-0 A B C X) 0))
         :hints
         (("Goal" :in-theory
                  (enable SQUARE
                          CERT
                          PROB-0
                          PROB-1
                          IDEAL-CF-0 CONE-CF-0 MONOID-CF-0)))
         :rule-classes (:linear )))
```

```
(LOCAL (DEFTHM CERT-CONTRA-C-0
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (>= (CONE-CF-0 A B C X) 0))
         :rule-classes (:linear )))

(LOCAL (DEFTHM CERT-CONTRA-I-0
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (= (* (IDEAL-CF-0 A B C X)
                        (PROB-0 A B C X))
                     0))
         :hints
         (("Goal" :in-theory
                  (enable SQUARE
                          CERT
                          PROB-0
                          PROB-1
                          IDEAL-CF-0 CONE-CF-0 MONOID-CF-0)))
         :rule-classes (:linear )))

(LOCAL (DEFTHM CERT-CONTRA
         (IMPLIES (AND (NOT (GOAL A B C X))
                       (RATIONALP A)
                       (RATIONALP B)
                       (RATIONALP C) (RATIONALP X))
                  (NEQ (CERT A B C X) 0))
         :rule-classes nil))
```

The Imandra/ACL2 tool

```
(IMPLIES (= (+ (* A X X) (* B X) C) 0)
         (>= (- (* B B) (* 4 A C)) 0))
```

```
;; Main lemma

 (LOCAL (DEFTHM MAIN
          (IMPLIES (AND (RATIONALP A)
                        (RATIONALP B)
                        (RATIONALP C) (RATIONALP X))
                   (GOAL A B C X))
          :hints
          (("Goal" :in-theory
                   (disable GOAL)
                   :use (CERT-KEY CERT-CONTRA)))
          :rule-classes nil))

;; Final theorem

 (DEFTHM FINAL
   (IMPLIES (AND (RATIONALP A)
                 (RATIONALP B)
                 (RATIONALP C)
                 (RATIONALP X) (= (+ (* A X X) (* B X) C) 0))
            (>= (- (* B B) (* 4 A C)) 0))
   :hints
   (("Goal" :in-theory
            (enable GOAL PROB-0 PROB-1) :use (MAIN )))
   :rule-classes nil)
```

The Positivstellensatz

$$\left(\bigwedge_i^{k_0} p_i = 0\right) \wedge \left(\bigwedge_i^{k_1} q_i \geq 0\right) \wedge \left(\bigwedge_i^{k_2} r_i \neq 0\right) \quad \text{s.t.} \quad p_i, q_i, r_i \in \mathbb{Q}[\vec{x}]$$

is unsatisfiable over $\mathbb{R}$ iff

$$\exists \mathrm{P} \in Ideal(p_1, \ldots, p_{k_0})$$

$$\exists \mathrm{Q} \in Cone(q_1, \ldots, q_{k_1})$$

$$\exists \mathrm{R} \in Monoid(r_1, \ldots, r_{k_2})$$

s.t.

$$\mathrm{P} + \mathrm{Q} + \mathrm{R}^2 = 0$$

where

$$Ideal(a_1, \ldots, a_m) = \left\{ \sum_{i=1}^{m} a_i b_i \mid b_i \in \mathbb{Q}[\vec{x}] \right\}$$

$$Cone(a_1, \ldots, a_m) = \left\{ r + \sum_{i=1}^{m} t_i u_i \mid r, t_i \in \sum (\mathbb{Q}[\vec{x}])^2, u_i \in Monoid(a_1, \ldots, a_m) \right\}$$

$$Monoid(a_1, \ldots, a_m) = \left\{ \prod_{i=1}^{m} (a_i)^j \mid j \in \mathbb{N} \right\}$$

$$\sum (\mathbb{Q}[\vec{x}])^2 = \left\{ \sum_{i=1}^{v} (p_i)^2 \mid p_i \in \mathbb{Q}[\vec{x}] \ \wedge \ v \in \mathbb{N} \right\}.$$

The Positivstellensatz

$$\left(\bigwedge_i^{k_0} p_i = 0\right) \wedge \left(\bigwedge_i^{k_1} q_i \geq 0\right) \wedge \left(\bigwedge_i^{k_2} r_i \neq 0\right) \quad \text{s.t.} \quad p_i, q_i, r_i \in \mathbb{Q}[\vec{x}]$$

is unsatisfiable over $\mathbb{R}$ iff

$$\exists \mathrm{P} \in Ideal(p_1, \ldots, p_{k_0})$$

$$\exists \mathrm{Q} \in Cone(q_1, \ldots, q_{k_1})$$

$$\exists \mathrm{R} \in Monoid(r_1, \ldots, r_{k_2})$$

s.t.

$$\mathrm{P} + \mathrm{Q} + \mathrm{R}^2 = 0$$

CERTIFICATE

where

$$Ideal(a_1, \ldots, a_m) = \left\{ \sum_{i=1}^{m} a_i b_i \mid b_i \in \mathbb{Q}[\vec{x}] \right\}$$

$$Cone(a_1, \ldots, a_m) = \left\{ r + \sum_{i=1}^{m} t_i u_i \mid r, t_i \in \sum (\mathbb{Q}[\vec{x}])^2, u_i \in Monoid(a_1, \ldots, a_m) \right\}$$

$$Monoid(a_1, \ldots, a_m) = \left\{ \prod_{i=1}^{m} (a_i)^j \mid j \in \mathbb{N} \right\}$$

$$\sum (\mathbb{Q}[\vec{x}])^2 = \left\{ \sum_{i=1}^{v} (p_i)^2 \mid p_i \in \mathbb{Q}[\vec{x}] \ \wedge \ v \in \mathbb{N} \right\}.$$

The Positivstellensatz

$$\left(\bigwedge_i^{k_0} p_i = 0\right) \wedge \left(\bigwedge_i^{k_1} q_i \geq 0\right) \wedge \left(\bigwedge_i^{k_2} r_i \neq 0\right) \quad \text{s.t.} \quad p_i, q_i, r_i \in \mathbb{Q}[\vec{x}]$$

is unsatisfiable over $\mathbb{R}$ iff

$$\exists \mathrm{P} \in Ideal(p_1, \ldots, p_{k_0})$$

$$\exists \mathrm{Q} \in Cone(q_1, \ldots, q_{k_1})$$

$$\exists \mathrm{R} \in Monoid(r_1, \ldots, r_{k_2})$$

s.t.

$$\mathrm{P} + \mathrm{Q} + \mathrm{R}^2 = 0$$

CERTIFICATE

=0

where

$$Ideal(a_1, \ldots, a_m) = \left\{ \sum_{i=1}^{m} a_i b_i \mid b_i \in \mathbb{Q}[\vec{x}] \right\}$$

$$Cone(a_1, \ldots, a_m) = \left\{ r + \sum_{i=1}^{m} t_i u_i \mid r, t_i \in \sum (\mathbb{Q}[\vec{x}])^2, u_i \in Monoid(a_1, \ldots, a_m) \right\}$$

$$Monoid(a_1, \ldots, a_m) = \left\{ \prod_{i=1}^{m} (a_i)^j \mid j \in \mathbb{N} \right\}$$

$$\sum (\mathbb{Q}[\vec{x}])^2 = \left\{ \sum_{i=1}^{v} (p_i)^2 \mid p_i \in \mathbb{Q}[\vec{x}] \ \wedge \ v \in \mathbb{N} \right\}.$$

The Positivstellensatz

$$\left(\bigwedge_i^{k_0} p_i = 0\right) \wedge \left(\bigwedge_i^{k_1} q_i \geq 0\right) \wedge \left(\bigwedge_i^{k_2} r_i \neq 0\right) \quad \text{s.t.} \quad p_i, q_i, r_i \in \mathbb{Q}[\vec{x}]$$

is unsatisfiable over $\mathbb{R}$ iff

$$\exists \mathrm{P} \in Ideal(p_1, \ldots, p_{k_0})$$

$$\exists \mathrm{Q} \in Cone(q_1, \ldots, q_{k_1})$$

$$\exists \mathrm{R} \in Monoid(r_1, \ldots, r_{k_2})$$

s.t.

$$\mathrm{P} + \mathrm{Q} + \mathrm{R}^2 = 0$$

CERTIFICATE

where

$$Ideal(a_1, \ldots, a_m) = \left\{ \sum_{i=1}^{m} a_i b_i \mid b_i \in \mathbb{Q}[\vec{x}] \right\}$$

=0

$$Cone(a_1, \ldots, a_m) = \left\{ r + \sum_{i=1}^{m} t_i u_i \mid r, t_i \in \sum (\mathbb{Q}[\vec{x}])^2, u_i \in Monoid(a_1, \ldots, a_m) \right\}$$

>=0

$$Monoid(a_1, \ldots, a_m) = \left\{ \prod_{i=1}^{m} (a_i)^j \mid j \in \mathbb{N} \right\}$$

$$\sum (\mathbb{Q}[\vec{x}])^2 = \left\{ \sum_{i=1}^{v} (p_i)^2 \mid p_i \in \mathbb{Q}[\vec{x}] \ \wedge \ v \in \mathbb{N} \right\}.$$

The Positivstellensatz

$$\left(\bigwedge_i^{k_0} p_i = 0\right) \wedge \left(\bigwedge_i^{k_1} q_i \geq 0\right) \wedge \left(\bigwedge_i^{k_2} r_i \neq 0\right) \quad \text{s.t.} \quad p_i, q_i, r_i \in \mathbb{Q}[\vec{x}]$$

is unsatisfiable over $\mathbb{R}$ iff

$$\exists \mathrm{P} \in Ideal(p_1, \ldots, p_{k_0})$$

$$\exists \mathrm{Q} \in Cone(q_1, \ldots, q_{k_1})$$

$$\exists \mathrm{R} \in Monoid(r_1, \ldots, r_{k_2})$$

s.t.

$$\mathrm{P} + \mathrm{Q} + \mathrm{R}^2 = 0$$

CERTIFICATE

where

$$Ideal(a_1, \ldots, a_m) = \left\{ \sum_{i=1}^{m} a_i b_i \mid b_i \in \mathbb{Q}[\vec{x}] \right\}$$

=0

$$Cone(a_1, \ldots, a_m) = \left\{ r + \sum_{i=1}^{m} t_i u_i \mid r, t_i \in \sum (\mathbb{Q}[\vec{x}])^2, u_i \in Monoid(a_1, \ldots, a_m) \right\}$$

>=0

$$Monoid(a_1, \ldots, a_m) = \left\{ \prod_{i=1}^{m} (a_i)^j \mid j \in \mathbb{N} \right\}$$

<>0

$$\sum (\mathbb{Q}[\vec{x}])^2 = \left\{ \sum_{i=1}^{v} (p_i)^2 \mid p_i \in \mathbb{Q}[\vec{x}] \ \wedge \ v \in \mathbb{N} \right\}.$$

Semi-definite Programming for Positivstellensatz Search

- SDP: Convex optimization of a linear objective function modulo PSD matrix
- Pioneered by Parrilo in his 2000 Caltech PhD: *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*
- Builds on the *Gram Matrix* ***Sums of Squares*** approach of Powers and Woermann
- Implemented in HOL-Light by John Harrison (2007): *we build heavily on his work!*

Semi-definite Programming for Positivstellensatz Search

Mathematical Programming manuscript No.
(will be inserted by the editor)

Pablo A. Parrilo

Semidefinite programming relaxations for semialgebraic problems

Abstract. A hierarchy of convex relaxations for semialgebraic problems is introduced. For questions reducible to a finite number of polynomial equalities and inequalities, it is shown how to construct a complete family of polynomially sized semidefinite programming conditions that prove infeasibility. The main tools employed are a semidefinite programming formulation of the sum of squares decomposition for multivariate polynomials, and some results from real algebraic geometry. The techniques provide a constructive approach for finding bounded degree solutions to the Positivstellensatz, and are illustrated with examples from diverse application fields.

Key words. Semidefinite programming – convex optimization – sums of squares – polynomial equations – real algebraic geometry.

1. Introduction

Numerous questions in applied mathematics can be formally expressed using a finite number of polynomial equalities and inequalities. Well-known examples are optimization problems with polynomial objective and constraints, such as quadratic, linear, and boolean programming. This is a fairly broad class, including problems with a combination of continuous and discrete variables, and easily seen to be NP-hard in the general case.

In this paper we introduce a new approach to the formulation of computable relaxations for this kind of problems. The crucial enabling fact is the computational tractability of the sum of squares decomposition for multivariate polynomials, coupled with powerful results from semialgebraic geometry. As a result, a whole new class of convex approximations for semialgebraic problems is obtained. The results generalize in a very natural way existing successful approaches, including the well-known semidefinite relaxations for combinatorial optimization problems.

The paper includes notions from traditionally separated research areas, namely numerical optimization and real algebra. In the interest of achieving the broadest possible communication of the main ideas, we have tried to make this article as self-contained as possible, providing a brief introduction to both semidefinite programming and real algebra. It is our belief that there is a lot of potential in the interaction between these fields, particularly with regard to practical applications. Most of the material in the pa-

Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), CH-8092 Zurich, Switzerland. e-mail: parrilo@aut.ee.ethz.ch. The majority of this research has been carried out while the author was with the Control & Dynamical Systems Department, California Institute of Technology, Pasadena, CA 91125, USA.

Verifying nonlinear real formulas via sums of squares

John Harrison

Intel Corporation, JF1-13
2111 NE 25th Avenue, Hillsboro OR 97124, USA
johnh@ichips.intel.com

Abstract. Techniques based on sums of squares appear promising as a general approach to the universal theory of reals with addition and multiplication, i.e. verifying Boolean combinations of equations and inequalities. A particularly attractive feature is that suitable 'sum of squares' certificates can be found by sophisticated numerical methods such as semidefinite programming, yet the actual verification of the resulting proof is straightforward even in a highly foundational theorem prover. We will describe our experience with an implementation in HOL Light, noting some successes as well as difficulties. We also describe a new approach to the univariate case that can handle some otherwise difficult examples.

1 Verifying nonlinear formulas over the reals

Over the real numbers, there are algorithms that can in principle perform quantifier elimination from arbitrary first-order formulas built up using addition, multiplication and the usual equality and inequality predicates. A classic example of such a quantifier elimination equivalence is the criterion for a quadratic equation to have a real root:

$$\forall a\ b\ c.\ (\exists x.\ ax^2 + bx + c = 0) \Leftrightarrow a = 0 \wedge (b = 0 \Rightarrow c = 0) \vee a \neq 0 \wedge b^2 \geq 4ac$$

The first quantifier elimination algorithm for this theory was developed by Tarski [32],[1] who actually demonstrated completeness and quantifier elimination just for the theory of real-closed fields, which can be characterized as ordered fields where all non-negative elements have square roots ($\forall x.\ 0 \leq x \Rightarrow \exists y.\ x = y^2$) and all non-trivial polynomials of odd degree have a root. There are several interesting models of these axioms besides the reals (e.g. the algebraic reals, the computable reals, the hyperreals) yet Tarski's result shows that these different models satisfy exactly the same properties in the first-order language under consideration.

However, Tarski's procedure is complicated and inefficient. Many alternative decision methods were subsequently proposed; two that are significantly simpler were given by Seidenberg [30] and Cohen [8], while the CAD algorithm [9], apparently the first ever to be implemented, is significantly more efficient, though relatively complicated. Cohen's ideas were recast by Hörmander [17] into a relatively simple algorithm. However, even CAD has poor worst-case complexity (doubly exponential), and the Cohen-Hörmander algorithm is generally still slower. Thus, there has been limited progress on

[1] Tarski actually discovered the procedure in 1930, but it remained unpublished for many years afterwards.

Semi-definite Programming for Positivstellensatz Search

Sums of Squares Methods Explained: Part I

Grant Olney Passmore
grant.passmore@cl.cam.ac.uk
15 JJ Thomson Ave., University of Cambridge, CB3 0FD, UK

1 Sums of Squares Methods

From a high level, these methods rely upon the following two observations:

1. The question as to whether or not a real polynomial $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ is a sum of squares (SOS) of real polynomials can be reduced to a semidefinite programming problem, and
2. The search for a Positivstellensatz refutation certifying the emptiness of a semialgebraic set defined by an RCF constraint system can be reduced to a finite sequence of searches for SOS decompositions.

Below we present an expository account of the difficult part of the first observation, due to Powers and Wörmann [PW99] and building upon the key insights of Choi, Lam, and Reznick [MDCR95]. An expository account of the second observation, due to Parrilo [Par03], will be the subject of a second note.

Given a PSD real polynomial $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ that is a sum of squares of real polynomials, we seek an algorithm that will compute $p_1(\vec{x}), \ldots, p_n(\vec{x}) \in \mathbb{R}[\vec{x}]$ s.t. $p(\vec{x}) = \sum_{i=1}^{n} p_i^2(\vec{x})$.

Remark. From now on, unless specified otherwise, when we write "sum of squares" or "SOS" we mean "sum of squares of real polynomials in $\mathbb{R}[x_1, \ldots, x_n]$".

1.1 The Powers-Wörmann SOS Decomposition

Let $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ be SOS in t real polynomial squares. Then, $p(\vec{x})$ must have even degree. Let $deg(p(\vec{x})) = 2k$. Then, $\exists q_1, \ldots, q_t \in \mathbb{R}[\vec{x}]$ s.t. $deg(q_i(\vec{x})) \leq k$ and $p(\vec{x}) = \sum_{i=1}^{t} q_i^2(\vec{x})$.

A key observation is that we can now exactly characterise the finitely many possible power-products that could occur in each $q_i(\vec{x})$.

Definition 1.1. Let $\Lambda_n(d) = \{\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle \in \mathbb{N}^n \mid \alpha_1 + \ldots + \alpha_n \leq d\}$.

Then, as $deg(q_i(\vec{x})) \leq k$ $(\forall\ 1 \leq i \leq t)$, we see that the exponent vector α for each monomial occurring in each $q_i(\vec{x})$ must be a member of $\Lambda_n(k)$.

Semi-definite Programming for Positivstellensatz Search

Sums of Squares Methods Explained: Part I

Grant Olney Passmore
grant.passmore@cl.cam.ac.uk
15 JJ Thomson Ave., University of Cambridge, CB3 0FD, UK

1 Sums of Squares Methods

From a high level, these methods rely upon the following two observations:

1. The question as to whether or not a real polynomial $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ is a sum of squares (SOS) of real polynomials can be reduced to a semidefinite programming problem, and
2. The search for a Positivstellensatz refutation certifying the emptiness of a semialgebraic set defined by an RCF constraint system can be reduced to a finite sequence of searches for SOS decompositions.

Below we present an expository account of the difficult part of the first observation, due to Powers and Wörmann [PW99] and building upon the key insights of Choi, Lam, and Reznick [MDCR95]. An expository account of the second observation, due to Parrilo [Par03], will be the subject of a second note.

Given a PSD real polynomial $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ that is a sum of squares of real polynomials, we seek an algorithm that will compute $p_1(\vec{x}), \ldots, p_n(\vec{x}) \in \mathbb{R}[\vec{x}]$ s.t. $p(\vec{x}) = \sum_{i=1}^{n} p_i^2(\vec{x})$.

Remark. From now on, unless specified otherwise, when we write "sum of squares" or "SOS" we mean "sum of squares of real polynomials in $\mathbb{R}[x_1, \ldots, x_n]$".

1.1 The Powers-Wörmann SOS Decomposition

Let $p(\vec{x}) \in \mathbb{R}[\vec{x}]$ be SOS in t real polynomial squares. Then, $p(\vec{x})$ must have even degree. Let $deg(p(\vec{x})) = 2k$. Then, $\exists q_1, \ldots, q_t \in \mathbb{R}[\vec{x}]$ s.t. $deg(q_i(\vec{x})) \leq k$ and $p(\vec{x}) = \sum_{i=1}^{t} q_i^2(\vec{x})$.

A key observation is that we can now exactly characterise the finitely many possible power-products that could occur in each $q_i(\vec{x})$.

Definition 1.1. Let $\Lambda_n(d) = \{\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle \in \mathbb{N}^n \mid \alpha_1 + \ldots + \alpha_n \leq d\}$.

Then, as $deg(q_i(\vec{x})) \leq k$ $(\forall\ 1 \leq i \leq t)$, we see that the exponent vector α for each monomial occurring in each $q_i(\vec{x})$ must be a member of $\Lambda_n(k)$.

Example 1.1. Let $p(x_1, x_2) = 4x_1^2 + x_2^2$. Then,

1. $dim(p) = 2$, $deg(p) = 2 = 2k$ with $k = 1$,
2. $\Lambda_2(2k) = \Lambda_2(2) = \{\langle 0,0 \rangle, \langle 1,0 \rangle, \langle 0,1 \rangle, \langle 1,1 \rangle, \langle 2,0 \rangle, \langle 0,2 \rangle\}$, which are the exponent vectors for the following power-products (in this order):
$$\{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\}.$$
These are all of the power-products that could occur in an arbitrary 2-dimensional polynomial of degree 2.
3. $\Lambda_2(k) = \Lambda_2(1) = \{\langle 0,0 \rangle, \langle 1,0 \rangle, \langle 0,1 \rangle\}$ which are the exponent vectors for the following power-products (in this order):
$$\{1, x_1, x_2\}.$$
These are all of the power-products that could occur in SOS co-factors of an arbitrary 2-dimensional polynomial of degree 2.
4. We set $u = |\Lambda_2(k)| = 3$ and fix an order upon $\Lambda_2(k)$ by setting:
$$\beta_1 = \langle 0,0 \rangle,\ \beta_2 = \langle 1,0 \rangle,\ \beta_3 = \langle 0,1 \rangle.$$
5. We then set
$$\vec{\zeta} = \begin{pmatrix} \vec{x}^{\beta_1} \\ \vec{x}^{\beta_2} \\ \vec{x}^{\beta_3} \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$
6. Now, $p(x_1, x_2)$ is *SOS* iff we can exhibit a $u \times u$ $(= 3 \times 3)$ real, symmetric, PSD matrix B s.t.
$$p(x_1, x_2) \ = \ \vec{\zeta}^T B \vec{\zeta}.$$
That is, we are looking for some real, symmetric, PSD
$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \ s.t. \ \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \ = \ 4x_1^2 + x_2^2.$$
By multiplying through, we then see that:
$$\begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \ =$$
$$\begin{pmatrix} b_{1,1} + b_{2,1}x_1 + b_{3,1}x_2 & b_{1,2} + b_{2,2}x_1 + b_{3,2}x_2 & b_{1,3} + b_{2,3}x_1 + b_{3,3}x_2 \end{pmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \ =$$
$$(b_{1,2} + b_{2,1})x_1 + (b_{1,3} + b_{3,1})x_2 + (b_{2,3} + b_{3,2})x_1x_2 + b_{2,2}x_1^2 + b_{3,3}x_2^2 + b_{1,1}.$$

More example problems

```
(IMPLIES (= (+ (* X X) (* Y Y) (* Z Z)) 1)
         (<= (* (+ X Y Z) (+ X Y Z)) 3))

(IMPLIES (= (+ (* W W) (* X X) (* Y Y) (* Z Z)) 1)
         (<= (* (+ W X Y Z) (+ W X Y Z)) 4))

(IMPLIES (AND (<= 0 X) (<= 0 Y) (= (* X Y) 1))
         (<= (+ X Y) (+ (* X X) (* Y Y))))

(IMPLIES (AND (>= X 1) (>= Y 1))
         (>= (* X Y) (- (+ X Y) 1)))

(IMPLIES (AND (<= 0 X) (<= 0 Y))
              (<= (* X Y (EXPT (+ X Y) 2))
                  (EXPT (+ (* X X) (* Y Y)) 2)))

(IMPLIES (AND (<= 0 A) (<= 0 B) (<= 0 C)
              (<= (* C (EXPT (+ (* 2 A) B) 3)) (* 27 X)))
          (<= (* C A A B) X))
```

Conclusion

```
grant@logicBOX imandra-sos % dune exec ./main.exe

          .__                                .___
          |__|  _____  _____      ____     __| _/_______ _____
   ______ |  | /     \ \__  \    /    \   / __ | \_  __ \\__  \    ______
  /_____/ |  ||  Y Y  \ / __ \_ |   |  \ / /_/ |  |  | \/ / __ \_ /_____/
          |__||__|_|  /(____  /|___|  /\____ |  |__|   (____  /
                    \/      \/      \/      \/              \/
--------------------------------------------------------------------------------
 Imandra/ACL2 for nonlinear real arithmetic v0.1a - (c) Imandra Inc., 2023
--------------------------------------------------------------------------------
- Goal (over ℝ):

   (IMPLIES (AND (<= 0 a)
                 (<= 0 b)
                 (<= 0 c)
                 (<= (* c (EXPT (+ (* 2 a) b) 3)) (* 27 x)))
            (<= (* c a a b) x)).

.:. Proof found!

- Constructing ACL2 proof from Imandra's Positivstellensatz refutation:

;;
;; Theorem (over ℝ):
;;
;; (IMPLIES (AND (<= 0 a)
;;               (<= 0 b)
;;               (<= 0 c)
;;               (<= (* c (EXPT (+ (* 2 a) b) 3)) (* 27 x)))
;;          (<= (* c a a b) x)).
;;
;; Proof found by Imandra in 1.728803 secs.
;; Questions? Contact Grant Passmore (grant@imandra.ai).
;;
```

```
ACL2 !>>>(DEFTHM
          FINAL
          (IMPLIES (AND (RATIONALP A)
                        (RATIONALP B)
                        (RATIONALP C)
                        (RATIONALP X)
                        (<= 0 A)
                        (<= 0 B)
                        (<= 0 C)
                        (<= (* C (EXPT (+ (* 2 A) B) 3))
                            (* 27 X)))
                   (<= (* C A A B) X))
          :HINTS
          (("Goal"
                :IN-THEORY (ENABLE GOAL PROB-0 PROB-1 PROB-2 PROB-3 PROB-4)
                :USE (MAIN)))
          :RULE-CLASSES NIL)
Goal'
Goal''

Q.E.D.
```

- A new automatic proof procedure for nonlinear real inequalities in ACL2
- Uses Imandra's implementation of SDP-based Positivstellensatz search, exports ACL2-checkable proofs
- In practice can be very useful on hard algebraic problems in, e.g., <=10 variables
- As dimension grows, search becomes exponentially harder
- Many extensions possible — please send me your problems! Thank you!