

Proofs of polynomial inequalities in Coq.

Loïc Pottier (INRIA Sophia Antipolis),

joint work with Assia Mahboubi (ENS Lyon)

Mechanize parts of proofs in real analysis in the Coq system:

Existing tactics:

Omega: linear inequalities with integer coefficients.

Ring,Field: equalities of polynomials, rational fractions:

$$A[X_1, \dots, X_n], k(X_1, \dots, X_n).$$

Fourier: linear inequalities with real coefficients.

Improvement: proof **polynomial inequalities** with real coefficients.

$$\forall x \in \mathbb{R}, x^2 - x - 1 = 0 \wedge x \geq 0 \Rightarrow x \geq 1, 6$$

$$\forall x, y \in \mathbb{R}, y \geq x^2 + 1 \wedge y \leq 0 \Rightarrow y \geq x + 1$$

Algorithm.

Suggested by M-F.Roy, from "Géométrie algébrique réelle", J.Bochnak, M.Coste, M-F.Roy, p15-19:"Le principe de Tarski-Seidenberg".

Following L.Hörmander "The Analysis of Linear Partial differential Operator II", Appendix A.2, p364-371.

Following a manuscript of P.Cohen.

Input: a family $f = \{f_1, \dots, f_s\}$ of polynomials in $\mathbb{Q}[X_1, \dots, X_n]$

Output (simplified): the set of all possible signs of f , given real values to the X_i .

Example:

$$f = \{XY + 1, Y\}$$

leads to

$XY + 1$	+	+	+	0	0	-	-
Y	-	0	+	-	+	-	+

then, for example, $\forall x, y \in \mathbb{R}, xy + 1 < 0 \Rightarrow y \neq 0$.

Proof for Coq.

Proof steps in the algorithm:

- derivative of polynomials
- equality of polynomials (Euclidean divisions)
- intermediate value theorems:
 $a < b, P(a)P(b) < 0 \Rightarrow \exists c, a < c < b, P(c) = 0.$
 $a < b, P(a) > 0, P(b) > 0, \forall x \in]a; b[P'(x) < 0 \Rightarrow \forall x \in]a; b[P(x) > 0$
+ 44 other versions....
- axioms and basic lemmas on \mathbb{R} (as a real closed field).

Then, only elementary theory of \mathbb{R} is needed: present in the library of Coq.
Polynomials are axiomatized:

```

Parameter Pol : Type.
Parameter P1,P0,PX : Pol.
Parameters plusP, multP : Pol -> Pol -> Pol.
Parameter oppP : Pol -> Pol.
...
Variable Pol_is_a_ring : (Ring_Theory plusP multP P1 P0 oppP eqP).
Parameter Pc : R -> Pol.
Axiom plusR_plusP : (x, y : R) (Pc (Rplus x y)) == (plusP (Pc x) (Pc y)).
...
Parameter evalP : Pol -> R -> R.
Axiom evalP_X : (x : R) (evalP PX x) == x.
Axiom evalP_Pc : (x,c: R) (evalP (Pc c) x) == c.
...
Axiom
  evalP_plusP :
    (x : R)
    (p, q : Pol) (evalP (plusP p q) x) == (Rplus (evalP p x) (evalP q x)).
...

```

Algorithm: one variable.

$$f = \{f_1, \dots, f_s\} \subset \mathbb{Q}[X]$$

a **sign array** of f is given by a subdivision x_1, \dots, x_n of \mathbb{R} and the signs of f_1, \dots, f_s in the intervals:

Example: $f_1 = X^2 - 1, f_2 = 2X,$

$$x_1 = -1, x_2 = 0, x_3 = 1$$

sign $_f$	$] - \infty; -1[$	-1	$] -1; 0[$	0	$]0; 1[$	1	$]1; + \infty[$
$X^2 - 1$	+	0	-	-	-	0	+
$2X$	-	-	+	0	+	+	+

We will compute a new family f' ,

$f' < f$ ($<$ well-founded),

and deduce the sign array of f from the sign array of f' .

Theorem 1. Suppose that the f_i 's are non zero,
and $\deg(f_s) \geq \max \{1, \deg(f_1), \dots, \deg(f_{s-1})\}$

Let f'_s be the derivative of f_s .

Let g_1, \dots, g_s be the remainders of the Euclidean divisions:

$$f_s = q_s f'_s + g_s$$

$$\forall i, 1 \leq i < s, f_s = q_i f_i + g_i$$

Then we can compute the sign array of $\{f_1, \dots, f_s\}$ from the sign array of $\{f'_s, f_1, \dots, f_{s-1}, g_1, \dots, g_s\}$.

Proof. if α is a root of f'_s then $\text{sign}(f_s, \alpha) = \text{sign}(g_s, \alpha)$,
if α is a root of f_i then $\text{sign}(f_s, \alpha) = \text{sign}(g_i, \alpha)$,
and $\text{sign}(f_s, -\infty) = -\text{sign}(f'_s, -\infty)$, $\text{sign}(f_s, +\infty) = \text{sign}(f'_s, +\infty)$.

Between the roots of f'_s and the f_i , $-\infty$ and $+\infty$

if f_s changes its sign, it has a unique root (f'_s has a constant sign)

otherwise f_s keeps its sign (f'_s has a constant sign)

□

Example: $f_1 = X + 1, f_2 = X^2 - X - 1$

$$f_2' = 2X - 1 \quad f_2 = q_2 f_2' + g_2 = \frac{X}{2}(2X - 1) - \frac{X}{2} - 1 \quad g_2 = -\frac{X}{2} - 1$$

$$f_2 = q_1 f_1 + g_1 = X(X + 1) - 2X - 1 \quad g_1 = -2X - 1$$

sign array of $\{f_2', f_1, g_1, g_2\}$:

	$] - \infty; x_1[$	x_1	$]x_1; x_2[$	x_2	$]x_2; x_3[$	x_3	$]x_3; x_4[$	x_4	$]x_4; + \infty[$
$f_2' = 2X - 1$	-	-	-	-	-	-	-	0	+
$f_1 = X + 1$	-	-	-	0	+	+	+	+	+
$g_1 = -2X - 1$	+	+	+	+	+	0	-	-	-
$g_2 = -\frac{X}{2} - 1$	+	0	-	-	-	-	-	-	-

$x_1 = -2, x_2 = -1, x_3 = -\frac{1}{2}, x_4 = \frac{1}{2}$, but we don't need to know that...

sign array of $\{f_1, f_2\}$:

	x_2	x_4
$f_2 = X^2 - X - 1$		
$f_1 = X + 1$	0	+

	$] -\infty; x_2[$	x_2	$]x_2; y_1[$	y_1	$]y_1; x_4[$	x_4	$]x_4; y_2[$	y_2	$]y_2; +\infty[$
$f_2 = X^2 - X - 1$	+	+	+	0	-	-	-	0	+
$f_1 = X + 1$	-	0	+	+	+	+	+	+	+

Algorithm: several variables.

$$f_1, \dots, f_s \in \mathbb{Q}[X_1, \dots, X_{n-1}][X_n]$$

Adapt the method of one variable, and apply it recursively:

- i. leading coefficients must be non zero: maintain a list \mathcal{N} of non zero polynomials of $\mathbb{Q}[X_1, \dots, X_{n-1}]$, $\mathcal{N} = \emptyset$ at the beginning.
- ii. if every f_i is constant in X_n , then compute the sign arrays in the next variable: X_{n-1} . Return the columns where the f_i that divide a polynomial in \mathcal{N} are non zero.
- iii. else, if there is $f_{i_0} = c X_n^d + r$ s.t. c does not divide a polynomial of \mathcal{N} , branch:
 - $\{f_1, \dots, f_s\}$ with $\mathcal{N} := \mathcal{N} \cup \{c\}$
 - $\{f_1, \dots, f_{i_0-1}, r, f_{i_0+1}, f_s, c\}$ and return the sign arrays where c is identically zero.
- iv. else, apply the method for one variable, with pseudo-divisions:
 $c_i f_s = q_i f_i + g_i$, with $c_i \in \mathbb{Q}[X_1, \dots, X_{n-1}]$, $c_i > 0$.

Example: $f = \{XY + 1, Y\} \subset \mathbb{R}[X][Y]$

Leading coefficients:

$X \neq 0$: $\mathcal{N} := \{X\}$, now all leading coefficients are non zero, we apply the method for one variable:

$f_1 = Y$, $f_2 = XY + 1$, $f_2' = X$, $g_1 = 0$, $g_2 = 1$. Dropping the constant polynomials, this leads to:

$$f = \{Y, X\}.$$

Leading coefficients are non zero: we derivate and divide, getting:

$f = \{X, 1, 0, 0\}$ which is constant in Y . We treat now the variable X . By derivation and division, we get $f = \{1, 0\}$, with one sign array:

1	+
0	0

Then the sign array of $\{X\}$: (the subdivision is implicit)

X	-	0	+
---	---	---	---

Then for $\{X\}$ as polynomial in Y , there is 3 sign arrays:

X	-	X	0	X	+
-----	---	-----	---	-----	---

X divides a polynomial in \mathcal{N} , then we keep only the cases where X is non zero:

X	-	X	+
-----	---	-----	---

Back to $\{Y, X\}$, we have:

Y	-	0	+	Y	-	0	+
X	-	-	-	X	+	+	+

Then back to $\{XY + 1, Y\}$:

$XY + 1$	+	+	+	0	-	$XY + 1$	-	0	+	+	+
Y	-	0	+	+	+	Y	-	-	-	0	+

which ends the case $X \neq 0$.

$X = 0 : \mathcal{N} = \{ \}, f$ becomes $\{1, Y\}$, then $\{Y\}$

.....

which has one sign array:

Y	$-$	0	$+$
-----	-----	-----	-----

, valid, because its polynomials do not divide polynomials of \mathcal{N} .

Then the sign array for $\{XY + 1, Y\}$ is:

$XY + 1$	$+$	$+$	$+$
Y	$-$	0	$+$

Finally there are 3 sign arrays for $\{XY + 1, Y\}$:

$XY + 1$	$+$	$+$	$+$	0	$-$
Y	$-$	0	$+$	$+$	$+$

$XY + 1$	$-$	0	$+$	$+$	$+$
Y	$-$	$-$	$-$	0	$+$

$XY + 1$	$+$	$+$	$+$
Y	$-$	0	$+$

Improvements:

- square free factorization with subresultant gcd.
- compute a small positive factor for pseudo-division.
- use memo functions for factorization and positivity.
- adapt the algorithm to do effective quantifier elimination.
- future: use subresultants in chains of pseudo-division.

Complexity:  :- (

but we don't study robotic examples...

Best example solved: eliminate a, b, c, d in $a x^3 + b x^2 + c x + d$

Building the Coq proof: one variable.

1. Execute the algorithm (Ocaml), keeping a trace of its computations:
intermediate polynomials,
sign arrays,
coefficients of Euclidean divisions,
square free factorizations
2. Fill in the sign arrays with proof-terms:
one for each sign, root, and interval.
Proof-terms are in an *ad-hoc* representation:
application of lemmas and axioms (intermediate value theorems, etc)
to parameters (polynomials and integers)

3. Build a big formula describing the sign array of the original polynomials:

$$\phi = \exists \alpha_1, \dots, \alpha_r, \alpha_1 < \dots < \alpha_r,$$

$$\text{sign}(f_1,] - \infty; \alpha_1[) = w_{11} \wedge \dots \wedge \text{sign}(f_s,]\alpha_r; + \infty[) = w_{s2r+1}$$

4. Glue the proof-terms in sign arrays to obtain a proof-term of ϕ .

Unfortunately, the proof-term has holes, even inside types: intermediate polynomials in factorization equalities, proofs of equalities of polynomials and derivatives.

The tactic **Refine** fails in this case...

5. \Rightarrow Build the proof-term of ϕ by combination of atomic tactics: **much more technical**, but holes can be filled in by: the Ring tactic (polynomial equalities, Euclidean divisions), and a tactic using reflexion (derivatives).

The tactic Tarski (in progress).

Proves formulas of the kind:

$$\forall x_1, \dots, x_n: \mathbb{R}, I_1(x_1, \dots, x_n) \rightarrow \dots \rightarrow I_r(x_1, \dots, x_n)$$

where $I_k(x_1, \dots, x_n)$ is a polynomial inequality or equality.

1. Transform the problem in showing contradictions:

$$\forall x_1, \dots, x_n: \mathbb{R}, I_1(x_1, \dots, x_n) \rightarrow \dots \rightarrow I_r(x_1, \dots, x_n) \rightarrow \text{False}$$

with strict inequalities.

2. Compute the sign arrays of the polynomials and the associated formula ϕ .
3. Build the proof of contradiction from the proofs of signs in the arrays, if possible: each column of sign arrays must be different from the sign conditions given in the inequalities.
4. In case of failure, add to the context the products of real intervals where inequalities are satisfied.

These slides have been written with $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$,
a GNU software developed by Joris Van Der Hoeven (CNRS, labo. de
math., Paris-Sud Orsay).

$\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ works on linux, Windows, etc.