A Reflexive Tactic for Polynomial Positivity
using Numerical Solvers and Floating-Point Computations

Érik Martin-Dorel\textsuperscript{1}  Pierre Roux\textsuperscript{2}

\textsuperscript{1}IRIT, Université Paul Sabatier, Toulouse, France

\textsuperscript{2}ONERA, Toulouse, France

January 16th, 2017
CPP 2017
Motivation

- Polynomial inequalities in the real field are decidable (Tarski)
- But exact algo. expensive
Motivation

- Polynomial inequalities in the real field are decidable (Tarski)
- But exact algo. expensive
  ⇒ Use incomplete numerical methods
    - off-the-shelf optimization solvers
    - a posteriori validation with exact rational arithmetic: state of the art (simple but costly)
Motivation

- Polynomial inequalities in the real field are decidable (Tarski).
- But exact algo. expensive
  ⇒ Use incomplete numerical methods
    - off-the-shelf optimization solvers
    - a posteriori validation with exact rational arithmetic: state of the art (simple but costly)
    - a posteriori validation with floating-point arithmetic (more efficient but non trivial)
  ⇒ We’d like formal proofs
Motivation

- Polynomial inequalities in the real field are decidable (Tarski)
- But exact algo. expensive

⇒ Use incomplete numerical methods
  - off-the-shelf optimization solvers
  - a posteriori validation with exact rational arithmetic: state of the art (simple but costly)
  - a posteriori validation with floating-point arithmetic (more efficient but non trivial)

⇒ We’d like formal proofs

demo.v
# Agenda

1. **Sum of Squares (SOS) Polynomials**
2. **Numerical Verification**
3. **Formalization & Reflexive Tactic**
4. **Benchmarks**
5. **Conclusion**
Agenda

1. Sum of Squares (SOS) Polynomials
2. Numerical Verification
3. Formalization & Reflexive Tactic
4. Benchmarks
5. Conclusion
Sum of Squares (SOS) Polynomials

Definition (SOS Polynomial)

A polynomial $p$ is SOS if there are polynomials $q_1, \ldots, q_m$ s.t.

$$p = \sum_i q_i^2.$$

- If $p$ SOS then $p \geq 0$
Definition (SOS Polynomial)

A polynomial $p$ is SOS if there are polynomials $q_1, \ldots, q_m$ s.t.

$$p = \sum_i q_i^2.$$ 

- If $p$ SOS then $p \geq 0$
- $p$ SOS iff there exist $z := [1, x_0, x_1, x_0x_1, \ldots, x_n]$ and $Q \succeq 0$ (i.e., for all $x, x^T Q x \geq 0$) s.t.

$$p = z^T Q z.$$ 

$\Rightarrow$ SOS can be encoded as semi-definite programming (SDP).
SOS: Example

Is \( p(x, y) := 2x^4 + 2x^3y - x^2y^2 + 5y^4 \) SOS?

\[
p(x, y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}
\]

that is \( p(x, y) = q_{11}x^4 + 2q_{13}x^3y + 2q_{23}xy^3 + (2q_{12} + q_{33})x^2y^2 + q_{22}y^4 \)
Example

Is \( p(x, y) := 2x^4 + 2x^3 y - x^2 y^2 + 5y^4 \) SOS?

\[
p(x, y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}
\]

that is \( p(x, y) = q_{11}x^4 + 2q_{13}x^3 y + 2q_{23}xy^3 + (2q_{12} + q_{33})x^2 y^2 + q_{22}y^4 \)

hence \( q_{11} = 2, \ 2q_{13} = 2, \ 2q_{23} = 0, \ 2q_{12} + q_{33} = -1, \ q_{22} = 5. \)
**SOS: Example**

Is \( p(x, y) := 2x^4 + 2x^3y - x^2y^2 + 5y^4 \) SOS?

\[
p(x, y) = \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}^T \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \end{bmatrix}
\]

that is \( p(x, y) = q_{11}x^4 + 2q_{13}x^3y + 2q_{23}xy^3 + (2q_{12} + q_{33})x^2y^2 + q_{22}y^4 \)

hence \( q_{11} = 2, 2q_{13} = 2, 2q_{23} = 0, 2q_{12} + q_{33} = -1, q_{22} = 5. \)

For instance

\[
Q = \begin{bmatrix} 2 & -3 & 1 \\ -3 & 5 & 0 \\ 1 & 0 & 5 \end{bmatrix} = L^T L \quad L = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & -3 & 1 \\ 0 & 1 & 3 \end{bmatrix}
\]

hence \( p(x, y) = \frac{1}{2} \left(2x^2 - 3y^2 + xy\right)^2 + \frac{1}{2} \left(y^2 + 3xy\right)^2. \)
Agenda

1. Sum of Squares (SOS) Polynomials
2. Numerical Verification
3. Formalization & Reflexive Tactic
4. Benchmarks
5. Conclusion
SOS: Using approximate SDP solvers

Result $Q$ from SDP solver will only satisfy equality constraints up to some error $\delta$

$$p = z^{T}Qz + z^{T}Ez, \quad \forall i,j, |E_{i,j}| \leq \delta.$$
SOS: Using approximate SDP solvers

Result $Q$ from SDP solver will only satisfy equality constraints up to some error $\delta$

$$p = z^T Q z + z^T E z, \quad \forall i, j, |E_{i,j}| \leq \delta.$$ 

If $Q + E \succeq 0$ then $p = z^T (Q + E) z$ is SOS.
SOS: Using approximate SDP solvers

Result $Q$ from SDP solver will only satisfy equality constraints up to some error $\delta$

$$p = z^T Q z + z^T E z, \quad \forall i, j, |E_{i,j}| \leq \delta.$$  

If $Q + E \succeq 0$ then $p = z^T (Q + E) z$ is SOS.

- Hence the validation method: given $p \simeq z^T Q z$
  1. Check that all monomials of $p$ are in $z z^T$.
  2. Bound difference $\delta$ between coefficients of $p$ and $z^T Q z$.
  3. If $Q - s \delta I \succeq 0$ ($s :=$ size of $Q$), then $p$ is proved SOS.

- 2 can be done with interval arithmetic and 3 with a Cholesky decomposition ($\Theta(s^3)$ flops).

$\Rightarrow$ Efficient validation method using just floats.
Intuitively

\[ \{ X \mid X \succeq 0 \} \]

equality constraints
Intuitively

\{ X \mid X \geq 0 \}

equality constraints

\mathbb{Q}

+ Q
Intuitively

\[ \{ X \mid X \geq 0 \} \]

\[ p \text{ SOS} \]

equality constraints

\[ \{ Q + E \} \]
Intuitively

\[ \{ X \mid X \geq 0 \} \]

\[ \{ Q + E \} \]

cannot conclude

equality constraints
Intuitively

\[ \{ X \mid X \succeq 0 \} \]

\[ + \]

\[ \{ Q + E \} \]

cannot conclude

equality constraints
Cholesky Decomposition

To prove that $a \in \mathbb{R}$ is non negative, we can exhibit $r$ such that $a = r^2$ (typically $r = \sqrt{a}$).
To prove that $a \in \mathbb{R}$ is non-negative, we can exhibit $r$ such that $a = r^2$ (typically $r = \sqrt{a}$).

To prove that a matrix $A \in \mathbb{R}^{n \times n}$ is positive semi-definite, we can similarly expose $R$ such that $A = R^T R$

(since $x^T \left( R^T R \right) x = (Rx)^T (Rx) = \|Rx\|_2^2 \geq 0$).

Cholesky Decomposition
To prove that $a \in \mathbb{R}$ is non negative, we can exhibit $r$ such that $a = r^2$ (typically $r = \sqrt{a}$).

To prove that a matrix $A \in \mathbb{R}^{n \times n}$ is positive semi-definite we can similarly expose $R$ such that $A = R^T R$

(since $x^T (R^T R) x = (Rx)^T (Rx) = \|Rx\|_2^2 \geq 0$).

The Cholesky decomposition computes such a matrix $R$:

\[
R := 0;
\]

\[
\text{for } j \text{ from } 1 \text{ to } n \text{ do }
\]

\[
\text{for } i \text{ from } 1 \text{ to } j - 1 \text{ do }
\]

\[
R_{i,j} := \left( A_{i,j} - \sum_{k=1}^{i-1} R_{k,i} R_{k,j} \right) / R_{i,i};
\]

\[
\text{od}
\]

\[
R_{j,j} := \sqrt{ M_{j,j} - \sum_{k=1}^{j-1} R_{k,j}^2 };
\]

\[
\text{od}
\]

If it succeeds (no \(\sqrt{\text{of negative or div. by 0}}\)) then $A \succeq 0$. 

Érik Martin-Dorel, Pierre Roux

A Reflexive Tactic for Polynomial Positivity
With rounding errors $A \neq R^T R$, Cholesky can succeed while $A \not\succeq 0$. 
With rounding errors $A \neq R^TR$, Cholesky can succeed while $A \ngeqslant 0$.

But error is bounded and for some (tiny) $c \in \mathbb{R}$:
if Cholesky succeeds on $A$ then $A + cI \geq 0$.

Hence:

**Theorem**

If Cholesky succeeds on $A - cI$ then $A \geq 0$

holds for any $c \geq \frac{(s + 1) \varepsilon}{1 - (2s + 2) \varepsilon} \text{tr}(A) + 4(s + 1) \left(2(s + 2) + \max_i(A_{i,i})\right) \eta$
($\varepsilon$ and $\eta$ relative and absolute precision of floating-point format).

Proved in Coq (paper proof: 6 pages, Coq: 5.1 kloc)
Agenda

1. Sum of Squares (SOS) Polynomials
2. Numerical Verification
3. Formalization & Reflexive Tactic
4. Benchmarks
5. Conclusion
Outline of the formalization
Outline of the formalization

1. Effective multivariate polynomials
   - CoqEAL [Cano, Cohen, Dénès, Mörtberg, Rouhling, Siles]
     → uses SSReflect and MathComp [Gonthier et al.]
   - proof: SsrMultinomials [Strub]
   - implem.: FMapAVL from Coq stdlib
   - coefficients: $\mathbb{Q}$ as bigQ from Coq stdlib
Outline of the formalization

1. Effective multivariate polynomials
   - CoqEAL [Cano, Cohen, Dénès, Mörtberg, Rouhling, Siles]
   - \( \Rightarrow \) uses SSReflect and MathComp [Gonthier et al.]
   - proof: SsrMultinomials [Strub]
   - implem.: FMapAVL from Coq stdlib
   - coefficients: \( \mathbb{Q} \) as \texttt{bigQ} from Coq stdlib

2. Effective check for positive definite matrices
   - CoqEAL
   - proof: previous work
   - implem.: lists of lists, CoqEAL
   - coefficients: floating-point numbers from CoqInterval [Melquiond]
Outline of the formalization

1. Effective multivariate polynomials
   - CoqEAL [Cano, Cohen, Dénès, Mörtberg, Rouhling, Siles]
   - \( \leadsto \) uses SSReflect and MathComp [Gonthier et al.]
   - proof: SsrMultinomials [Strub]
   - implem.: FMapAVL from Coq stdlib
   - coefficients: \( \mathbb{Q} \) as \texttt{bigQ} from Coq stdlib

2. Effective check for positive definite matrices
   - CoqEAL
   - proof: previous work
   - implem.: lists of lists, CoqEAL
   - coefficients: floating-point numbers from CoqInterval [Melquiond]

3. Reflexive tactic
   - OCaml code as a wrapper for SDP solvers
   - Some Ltac code
Refinement proofs: overview of CoqEAL’s methodology

1. Implement algorithms in a general way (polymorphic functions; type classes)
Refinement proofs: overview of CoqEAL’s methodology

1. Implement algorithms in a general way (polymorphic functions; type classes)
2. Specialize the algorithms with proof-oriented datatypes; correctness proof w.r.t a specification

\[
x : A \\
g_A \\
g_A x
\]
Refinement proofs: overview of CoqEAL’s methodology

1. Implement algorithms in a general way
   (polymorphic functions; type classes)

2. Specialize the algorithms with proof-oriented datatypes;
correctness proof w.r.t a specification
(or w.r.t another algorithm \(\rightsquigarrow\) program refinement)

\[
\begin{align*}
x & = x : A \\
f & \downarrow \\
f \, x & = g_A \, x \\
\end{align*}
\]
Refinement proofs: overview of CoqEAL’s methodology

1. Implement algorithms in a general way (polymorphic functions; type classes)
2. Specialize the algorithms with proof-oriented datatypes; correctness proof w.r.t a specification (or w.r.t another algorithm \(\rightsquigarrow\) program refinement)
3. Specialize the algorithms with effective datatypes;

\[
\begin{align*}
  x &= x : A \\
  f \quad &\downarrow \quad g_A \\
  f \ x &= g_A \ x \\
  c &= c : C \\
  \quad &\downarrow \quad g_C \\
\end{align*}
\]
Refinement proofs: overview of CoqEAL’s methodology

1. Implement algorithms in a general way (polymorphic functions; type classes)
2. Specialize the algorithms with proof-oriented datatypes; correctness proof w.r.t a specification (or w.r.t another algorithm $\rightsquigarrow$ program refinement)
3. Specialize the algorithms with effective datatypes; correctness proof w.r.t proof-oriented version ($\rightsquigarrow$ data refinement)

\[
x : A \quad \text{refines} \quad \quad x = c : C
\]

\[
f x \quad \text{refines} \quad \quad gA x = gC c
\]
Effective multivariate polynomials

- Implemented in a modular way:

  ```lean
  Definition seqmultinom := list N.
  Module MultinomOrd <: OrderedType.
    Definition t := seqmultinom. (*...*) End MultinomOrd.
  Module FMapMultipoly (M : Sfun MultinomOrd).
    Definition effmpoly := M.t. (*...*) End FMapMultipoly.
  Module M := FMapAVL.Make MultinomOrd.
  Module PolyAVL := FMapMultipoly M.
  ```
Effective multivariate polynomials

- Implemented in a modular way:

Definition seqmultinom := list N.
Module MultinomOrd <: OrderedType.
  Definition t := seqmultinom. (*...*) End MultinomOrd.
Module FMapMultipoly (M : Sfun MultinomOrd).
  Definition effmpoly := M.t. (*...*) End FMapMultipoly.
Module M := FMapAVL.Make MultinomOrd.
Module PolyAVL := FMapMultipoly M.

- Main refinement predicates:

Rseqmultinom : ∀(n : nat), multinom n → seqmultinom → Type
Reffmpoly : ∀(T : ringType) (n : nat), mpoly n T → effmpoly T T → Type
ReffmpolyC : ∀(A : ringType) (C : Type), (A → C → Type) →
  ∀(n : nat), mpoly n A → effmpoly C C → Type
Effective multivariate polynomials

- Implemented in a modular way:

```
Definition seqmultinom := list N.
Module MultinomOrd <: OrderedType.
    Definition t := seqmultinom. (*...*) End MultinomOrd.
Module FMapMultipoly (M : Sfun MultinomOrd).
    Definition effmpoly := M.t. (*...*) End FMapMultipoly.
Module M := FMapAVL.Make MultinomOrd.
Module PolyAVL := FMapMultipoly M.
```

- Main refinement predicates:

```
Rseqmultinom : ∀(n : nat), multinom n → seqmultinom → Type
Reffmpoly : ∀(T : ringType) (n : nat), mpoly n T → effmpoly T → Type
ReffmpolyC : ∀(A : ringType) (C : Type), (A → C → Type) →
    ∀(n : nat), mpoly n A → effmpoly C → Type
```

- Proof-oriented type for coefficients: needs a ringType structure; instantiated with MathComp’s rat. Effective counterpart: bigQ.
Positive definiteness check for floating-point matrices

**Definition** `posdef` $(n : \text{nat}) (A : \mathbf{M}[\mathbb{R}]_n) :=$

\[ \forall (x : \mathbf{cV}[\mathbb{R}]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. (* \text{pointwise} < *) \]
Positive definiteness check for floating-point matrices

\textbf{Definition}  \texttt{posdef} \ (n : \text{nat}) \ (A : \ 'M[R]_n) \ := \\
\forall (x : \ 'cV[R]_n), \ x \neq 0 \ \rightarrow \ 0 < x^T \times A \times x. \ (* \ \text{pointwise} < *)

\texttt{posdef\_check} : \ \forall (mx : \ \text{Type} \ \rightarrow \ \text{nat} \ \rightarrow \ \text{nat} \ \rightarrow \ \text{Type}) \\
(T : \ \text{Type}) \ (n : \ \text{nat}), \\
(*...\text{type classes...} \rightarrow *) \\
mx \ T \ n \ n \ \rightarrow \ \text{bool}
Positive definiteness check for floating-point matrices

**Definition** `posdef` (n : nat) (A : 'M[R]_n) :=
\( \forall (x : 'cV[R]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. \) (* pointwise < *

`posdef_check` : \( \forall (mx : Type \rightarrow nat \rightarrow nat \rightarrow Type)

(T : Type) (n : nat),

(*...type classes...\rightarrow *)

mx T n n \rightarrow bool

- Correctness: use formal proof of Cholesky algo over \( \mathbb{R} \) (previous work)
Positive definiteness check for floating-point matrices

Definition **posdef** \((n : \text{nat}) (A : \text{'}M[R]_n) :=\)
\(\forall (x : \text{'cV[R]_n}), x \neq 0 \rightarrow 0 < x^T \times A \times x.\) (* pointwise \(<\)*)

**posdef\_check** : \(\forall (mx : \text{Type} \rightarrow \text{nat} \rightarrow \text{nat} \rightarrow \text{Type})\)
\((T : \text{Type}) (n : \text{nat}),\)
(*...type classes...\(\rightarrow\) *)
\(\text{mx} \ T \ n \ n \rightarrow \text{bool}\)

- Correctness: use formal proof of Cholesky algo over \(\mathbb{R}\) (previous work)
- Refinement 1: refine dependently-typed matrices with list-based ones
Positive definiteness check for floating-point matrices

Definition **posdef** \((n : \text{nat}) (A : 'M[R]_n) :=\)
\[\forall (x : 'cV[R]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. (* \text{pointwise } < *)\]

**posdef_check** : \(\forall (mx : \text{Type} \rightarrow \text{nat} \rightarrow \text{nat} \rightarrow \text{Type})\)
\((T : \text{Type}) (n : \text{nat}), (* \ldots \text{type classes} \ldots \rightarrow *)\)
\(mx T n n \rightarrow \text{bool}\)

- Correctness: use formal proof of Cholesky algo over \(\mathbb{R}\) (previous work)
- Refinement 1: refine dependently-typed matrices with list-based ones
- Refinement 2: refine real coefficients with floating-point ones:
Positive definiteness check for floating-point matrices

Definition **posdef** \((n : \text{nat}) (A : 'M[R]_n) := \)
\[\forall (x : 'cV[R]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. (* \text{pointwise} < *)\]

**posdef_check** : \(\forall (mx : \text{Type} \rightarrow \text{nat} \rightarrow \text{nat} \rightarrow \text{Type})\)
\[\ (T : \text{Type}) (n : \text{nat}), \ (*\ldots\text{type classes}\ldots\rightarrow *)\]
\[\ mx \ T \ n \ n \rightarrow \text{bool}\]

- Correctness: use formal proof of Cholesky algo over \(\mathbb{R}\) (previous work)
- Refinement 1: refine dependently-typed matrices with list-based ones
- Refinement 2: refine real coefficients with floating-point ones:
  - Rely on Float_infnan_spec
Positive definiteness check for floating-point matrices

Definition `posdef` (n : nat) (A : 'M[R]_n) :=
\( \forall (x : 'cV[R]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. \) (* pointwise < * )

`posdef_check` : \( \forall (mx : Type \rightarrow nat \rightarrow nat \rightarrow Type) \)
\( (T : Type) (n : nat), \)
(*...type classes...\( \rightarrow \) *)

mx T n n \rightarrow bool

- Correctness: use formal proof of Cholesky algo over \( \mathbb{R} \) (previous work)
- Refinement 1: refine dependently-typed matrices with list-based ones
- Refinement 2: refine real coefficients with floating-point ones:
  - Rely on `Float_infnan_spec`
    = formalization of the floating-point “standard model” (previous work)
Positive definiteness check for floating-point matrices

Definition `posdef` (n : nat) (A : 'M[R]_n) :=
\( \forall (x : 'cV[R]_n), x \neq 0 \rightarrow 0 < x^T \times A \times x. \) (* pointwise < *)

`posdef_check` : \( \forall (mx : Type \rightarrow nat \rightarrow nat \rightarrow Type) \)
\( (T : Type) (n : nat), \)
(*...type classes...\rightarrow *)
\( mx T n n \rightarrow \text{bool} \)

- Correctness: use formal proof of Cholesky algo over \( \mathbb{R} \) (previous work)
- Refinement 1: refine dependently-typed matrices with list-based ones
- Refinement 2: refine real coefficients with floating-point ones:
  - Rely on `Float_infnan_spec`
  - formalization of the floating-point “standard model” (previous work)
  - instantiated with CoqInterval’s floating-point implementation, restricted to 53 bits.
The validsdp tactic (1/3) – the big picture

\[ \text{Goal} \]

\[ x_i : \mathbb{R} \vdash 0 \leq r \]

\[ \text{Ltac} \quad \text{reification (Ltac)} \]

\[ (x, p) : \text{list}(\mathbb{R}) \times \text{AST} \]

\[ \text{transform to effective datatypes} \]

\[ P : \text{list}(\text{list}(\mathbb{N}) \times \mathbb{Q}) \]

\[ (z, Q) : \text{list}(\text{list}(\mathbb{N})) \times \text{list}(\text{list}(\mathbb{F})) \]

\[ \text{soswitness (OCaml)} \]

\[ \text{computation} \]

\[ \text{check}(x, p, (z, Q)) = \text{true} \]

\[ \text{correctness theorem} \]

\[ 0 \leq \text{interp}(x, p) \]

\[ \text{convertibility rule} \]
The validsdp tactic (2/3) – OCaml code

- Rely on the OSDP lib. (OCaml interface for off-the-shelf SDP solvers)
- Implement a Coq plugin (the ValidSDP.soswitness OCaml module provides a soswitness tactic that consists of a wrapper for OSDP)

OSDP library: 6.2 kloc of OCaml code + 1.2 kloc of C code.
ValidSDP.soswitness plugin: 0.3 kloc of OCaml code.
The validsdp tactic (3/3) – correctness theorem

Theorem soscheck_eff_wrapup_correct :
\( \forall (x : \text{list } R) \ (p : p_{\text{abstr\_poly}}) \)
\[ (zQ : \text{list } (\text{list } N) \ast \text{list } (\text{list } (s_{\text{float}} \text{bigZ} \text{bigZ}))), \]
soscheck_eff_wrapup x p zQ = true \rightarrow
\( (0 \leq \text{interp}_p p_{\text{abstr\_poly}} x p)\%R. \)

Coq: 2.0 kloc for the main tactic and proofs + 6.5 kloc of refinement proofs
(Cholesky: 3.0 kloc; FP arith: 1.3 kloc; multipoly: 2.2 kloc)
Agenda

1. Sum of Squares (SOS) Polynomials
2. Numerical Verification
3. Formalization & Reflexive Tactic
4. Benchmarks
5. Conclusion
Benchmarks (1/3)

- In the paper: OSDP 0.4.5 & ValidSDP 0.3
- In this presentation: OSDP 0.5.2 & ValidSDP version d60c663

- In both configurations: Coq 8.5.2, MathComp 1.6, Flocq 2.5.1, Coquelicot 2.1.1, CoqInterval 3.1.0, and dev. version of other libs

- Setup:
  - A desktop PC under Debian GNU/Linux Jessie
  - Core i5-4460S CPU clocked at 2.9 GHz
  - All timings are total elapsed time (in seconds)
  - Timeout of 900s
## Benchmarks (2/3)

| Problem      | n | d | \(0.75\) | \(1.12\) | \(5.16\) | \(14.93\) | \(2.61\) | \(12.31\) | \(1.05\) | \(9.40\) | \(48.44\) | \(8.36\) | \(15.62\) | \(3.18\) | \(268.75\) | \(16.67\) | \(131.13\) | \(26.15\) | \(1.11\) | \(2.04\) | \(1.64\) | \(5.18\) | \(245.52\) | \(14.50\) | \(16.07\) | \(0.41\) | \(1.82\) | \(0.88\) | \(5.19\) | \(25.89\) | \(2.63\) | \(17.68\) | \(0.75\) | \(1.58\) | \(—\) | \(1.05\) | \(9.40\) | \(48.44\) | \(8.36\) | \(15.62\) | \(0.75\) | \(1.58\) | \(—\) | \(1.05\) | \(9.40\) | \(48.44\) | \(8.36\) | \(15.62\) | \(1.25\) | \(10.53\) | \(1.11\) | \(6.08\) | \(73.65\) | \(7.34\) | \(14.28\) | \(1.21\) | \(10.53\) | \(1.11\) | \(6.08\) | \(73.65\) | \(7.34\) | \(14.28\) | \(1.27\) | \(79.25\) | \(1.25\) | \(5.37\) | \(73.54\) | \(7.58\) | \(14.14\) | \(0.94\) | \(1.50\) | \(—\) | \(3.85\) | \(—\) | \(—\) | \(13.85\) | \(0.56\) | \(2.05\) | \(—\) | \(4.05\) | \(—\) | \(—\) | \(13.28\) | \(0.76\) | \(2.11\) | \(—\) | \(3.68\) | \(—\) | \(—\) | \(13.76\) | \(0.21\) | \(2.09\) | \(1.74\) | \(4.22\) | \(—\) | \(8.04\) | \(—\) |
## Benchmarks (3/3)

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>d</th>
<th>SOSD (not verified)</th>
<th>MonniauxC11 (not verified)</th>
<th>NLCertify (not verified)</th>
<th>ValidSDP</th>
<th>PVS/Bernstein</th>
<th>NLCertify</th>
<th>HOL Light/Taylor</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs868</td>
<td>6</td>
<td>4</td>
<td>0.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fs884</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fs890</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td>7.78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex4_d4</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex4_d6</td>
<td>2</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex4_d8</td>
<td>2</td>
<td>24</td>
<td>16.99</td>
<td></td>
<td></td>
<td></td>
<td>82.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex4_d10</td>
<td>2</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex5_d4</td>
<td>3</td>
<td>8</td>
<td>1.67</td>
<td></td>
<td></td>
<td></td>
<td>13.63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex5_d6</td>
<td>3</td>
<td>12</td>
<td>16.10</td>
<td></td>
<td></td>
<td></td>
<td>66.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex5_d8</td>
<td>3</td>
<td>16</td>
<td>203.06</td>
<td></td>
<td></td>
<td></td>
<td>353.70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex5_d10</td>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex6_d4</td>
<td>4</td>
<td>8</td>
<td>16.82</td>
<td></td>
<td></td>
<td></td>
<td>44.99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex6_d6</td>
<td>4</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex7_d4</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex7_d6</td>
<td>2</td>
<td>18</td>
<td>1.50</td>
<td></td>
<td></td>
<td></td>
<td>26.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex7_d8</td>
<td>2</td>
<td>24</td>
<td>15.38</td>
<td></td>
<td></td>
<td></td>
<td>83.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex7_d10</td>
<td>2</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex8_d4</td>
<td>2</td>
<td>8</td>
<td>0.87</td>
<td>15.72</td>
<td></td>
<td></td>
<td>7.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex8_d6</td>
<td>2</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex8_d8</td>
<td>2</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ex8_d10</td>
<td>2</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agenda

1. Sum of Squares (SOS) Polynomials
2. Numerical Verification
3. Formalization & Reflexive Tactic
4. Benchmarks
5. Conclusion
Conclusion

- Context: formal proof of multivariate polynomial positivity
- A Coq reflexive tactic
  - Input: polynomial ineq. goals with real variables and rational constants
  - Use off-the-shelf SDP solvers as untrusted oracles
  - Numerical approach with formal floating-point arithmetic
  - Algorithm involving matrices (Cholesky)
Discussion and open questions

- Bottleneck 1: emulate floating-point arithmetic in Coq (1000x overhead)
- Bottleneck 2: we currently use \texttt{vm\_compute}.
  native\_compute issue with large terms (witnesses for large problems)
- Coq 8.6 comes with LtacProf \(\leadsto\) feasibility of a ‘\texttt{vm\_compute} profiler’?
- Possible extension of ValidSDP: add support of elementary functions
Questions

Thanks for your attention!

https://sourcesup.renater.fr/validsdp/
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to

- inexact termination
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to
  - inexact termination
  - failure of strict feasibility
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to
- inexact termination
- failure of strict feasibility
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to
- inexact termination
- failure of strict feasibility
- ill conditioning
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to
- inexact termination
- failure of strict feasibility
- ill conditioning
- floating-point rounding errors
Inaccuracy in Solving SDPs

SDP solvers only yield approximate solutions due to
- inexact termination
- failure of strict feasibility
- ill conditioning
- floating-point rounding errors

State of the art [Harrison, Peyrl and Parrilo, Monniaux and Corbineau, Kaltofen et al., Magron et al.]
- round to exact rational solution (heuristic)
- proofs in rational arithmetic (expensive).
We want to prove that
\[ p_1(x_1, \ldots, x_n) \geq 0 \land \ldots \land p_m(x_1, \ldots, x_n) \geq 0 \]
is not satisfiable.
Positivstellensatz

We want to prove that

\[ p_1(x_1, \ldots, x_n) \geq 0 \land \ldots \land p_m(x_1, \ldots, x_n) \geq 0 \]

is not satisfiable.

Sufficient condition: there exist \( r_i \in \mathbb{R}[x] \) s.t.

\[ -\sum_i r_i p_i > 0 \quad \text{and} \quad \forall i, r_i \geq 0 \]
Positivstellensatz

We want to prove that

\[ p_1(x_1, \ldots, x_n) \geq 0 \land \ldots \land p_m(x_1, \ldots, x_n) \geq 0 \]

is not satisfiable.

Sufficient condition: there exist \( r_i \in \mathbb{R}[x] \) s.t.

\[ -\sum_i r_i p_i > 0 \quad \text{and} \quad \forall i, r_i \geq 0 \]

- equivalence under hypotheses (Putinar’s Positivstellensatz)
- no practical bound on degrees of \( r_i \) \( \Rightarrow \) will be arbitrarily fixed