Verification of Real-valued Programs

Rameez Wajid

Formal Methods Seminar, Nov 25, 2025

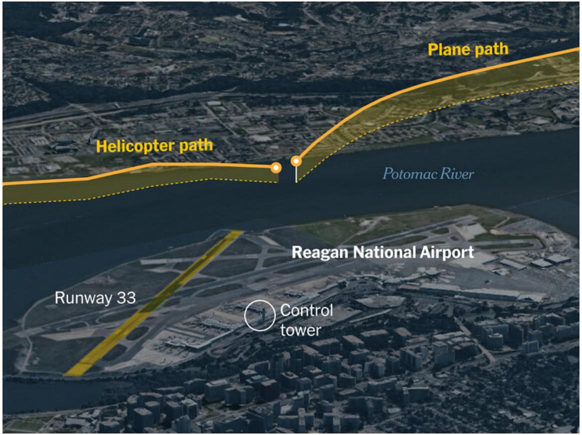
Programming Languages and Verification
University of Colorado Boulder

Outline

1. Motivation
2. Symbolic Methods
3. Sum of Squares and SDP
4. Positivstellensatz and Certification
5. Barrier Certificates and Safety
6. Toolchain and Practice
7. Conclusion

Motivation


Washington plane crash
BBC NEWS


Plane path
Helicopter path
Potomac River
Reagan National Airport
Runway 33
Control
tower

Verification Over Reals is Hard

Verification Over Reals is Hard

- Real-valued programs have **contninous** state spaces.

Verification Over Reals is Hard

- Real-valued programs have **contninous** state spaces.
- Nonlinear arithmetic leads to undecidable theories.

Verification Over Reals is Hard

- Real-valued programs have **contninous** state spaces.
- Nonlinear arithmetic leads to undecidable theories.
- First-order theory of reals is decidable but **doubly-exponential**.

Verification Over Reals is Hard

- Real-valued programs have **contninous** state spaces.
- Nonlinear arithmetic leads to undecidable theories.
- First-order theory of reals is decidable but **doubly-exponential**.
- Quantifier elimination and SMT solving struggle with **scaling**.

CUPLV

Verification Over Reals is Hard

- Real-valued programs have **continous** state spaces.
- Nonlinear arithmetic leads to undecidable theories.
- First-order theory of reals is decidable but **doubly-exponential**.
- Quantifier elimination and SMT solving struggle with **scaling**.

Need to exploit some (algebraic) structure (or accept conservative approximations)

Symbolic Methods: Quantifier Elimination and Gröbner Bases

- **Quantifier Elimination**:
- Eliminate quantifiers from logical formulas over the reals.
- Cylindrical Algebraic Decomposition (CAD) [Tarski/Collins].
- High complexity: 2^{2^n} in general.
- Effective on low-dimensional systems.
- **Gröbner Bases**: Solve polynomial equations by computing canonical basis.
- Useful for loop invariants and ideal membership.
- Still **doubly-exponential** in worst case.

Motivating SOS: From Nonnegativity to Certificates

- Instead of brute-force search over the reals, we can attempt to prove a required inequality by exhibiting a **certificate**.
- If we need to show a polynomial $f(x)$ is always nonnegative, it suffices to express f in a form that makes nonnegativity obvious.
- One powerful certificate is a sum of squares (SOS) representation: If $f(x)$ can be written as $f(x) = \sum_i h_i^2(x)$ for some polynomials h_i, then clearly $f(x) \geq 0$ **for all** x.
- SOS is a sufficient condition for polynomial nonnegativity (every SOS is globally ≥ 0 by construction)
- SOS decomposition constitutes a proof / certificate of f's nonnegativity

Hilbert and the Limits of SOS

- **Hilbert's Seventeenth Problem**: Can any positive semi-definite polynomial be written as a sum of squares of rational functions: $p = \sum_{j=1}^{k} \frac{\sigma_j^2}{\xi_j^2}$?
- Hilbert showed in 1888 that not all positive polynomials are SOS.
- **Motzkin Polynomial**: $x^4y^2 + x^2y^4 + 1 - 3x^2y^2$ is ≥ 0, not SOS.
- But every positive polynomial is a sum of squares of rational functions (Artin 1927).

CUPLV

SOS and Semidefinite Programming (SDP)

- Checking SOS reduces to SDP:

$$f(x) = Z(x)^T Q Z(x), \quad Q \succeq 0$$

- SDP is convex & solvable efficiently (e.g., MOSEK, SDPT3).
- Tools: SOSTOOLS, YALMIP.

CUPLV

SOS to SDP: How It Works

- Express $f(x) = Z(x)^T Q Z(x)$.
- $Z(x)$ is a monomial basis vector.
- $Q \succeq 0$ ensures positivity.
- SDP solvers can find Q.

Positivestellensatz

Let $S = \{\vec{x} \in \mathbb{R}^n \mid p_1(\vec{x}) \geq 0 \ \wedge \ \cdots \ \wedge \ p_m(\vec{x}) \geq 0\}$.

- We wish to show that $p \geq 0$ on S for given p.
- Important: We will need S to be compact.

Putinar's Positivstellensatz

Let $M = \{\sum_{j=1}^{m} \sigma_j p_j + \sigma_0 \mid \sigma_0, \ldots, \sigma_m \text{ SOS}\}$.

- Archimedean Property: There exists a K such that

$$K - (x_1^2 + \cdots + x_n^2) \in M$$

- Theorem (Putinar'1993):
- If $p \in M$ then $p_1 \geq 0 \ \wedge \ \cdots \wedge \ p_m \geq 0 \ \models \ p \geq 0$.
- If S compact and M is Archimedean, then $p_1 \geq 0 \ \wedge \ \cdots \ \wedge \ p_m \geq 0 \models p > 0$ then $p \in M$.

CUPLV

Positivstellensatz to Semi-Definite Programming

- Problem:prove the following entailment.

$$p_1 \geq 0 \ \wedge \ \cdots \ \wedge \ p_m \geq 0 \ \models \ p \geq 0$$

- Strategy:Find, $\sigma_0, \ldots, \sigma_m$ such that

$$p = \sigma_0 + \sum_{j=1}^{m} \sigma_j p_j, \text{ and } \sigma_j \text{ SOS}$$

- Bound the degrees of $\sigma_0, \ldots, \sigma_m \in \mathbb{R}_{2d}[\vec{x}]$

CUPLV

Reduction to SDP

- Fix a basis of monomials $\mu(\vec{x})$
- $\sigma_i = \mu^t X_i \mu$
- $p = \sigma_0 + \sum_{j=1}^{m} \sigma_j p_j$
- Equate monomials on LHS and RHS.
- $\sum_{j=0}^{m} (P_{i,j}, X_j) = c_i$
- Place $X_1, \ldots, X_n$ in a block diagonal form.

$$X = \begin{bmatrix} X_1 & 0 & \cdots & 0 \\ 0 & X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & X_n \end{bmatrix}$$

CUPLV

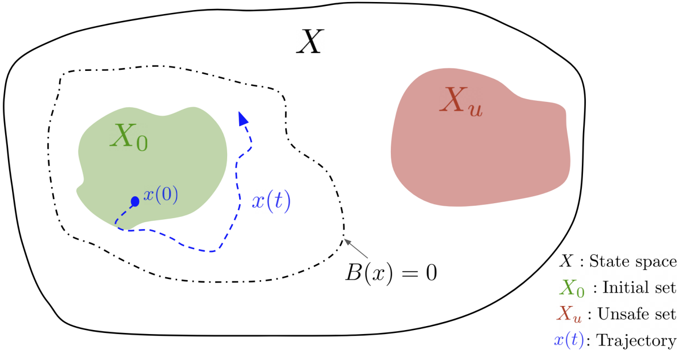
Certificates

Degree \ Field	Complex	Real
Linear	*Range/Kernel* Linear Algebra	*Farkas Lemma* Linear Programming
Polynomial	*Nullstellensatz* Bounded degree: LP Groebner bases	*Positivstellensatz* Bounded degree: SDP

[1]

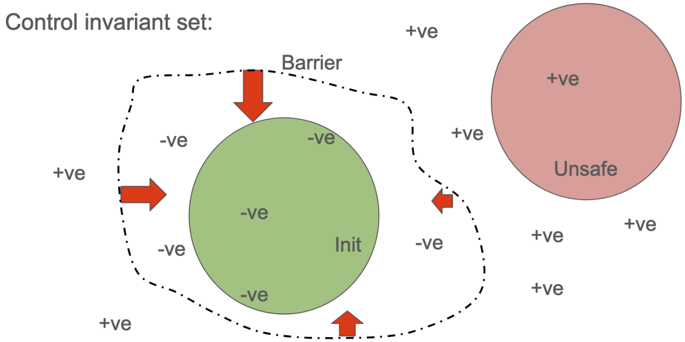
[1] P. Parrilo and S. Lall, ECC 2003

Barrier Functions - [Prajna et al.]



- $B(\vec{x}) > 0$ for all $\vec{x} \in X_u$ (B is **positive** when unsafe)
- $B(\vec{x}) < 0$ for all $\vec{x} \in X_i$ (B is **negative** when init)
- $B(\vec{x}) = 0$ implies $\nabla B(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq 0$

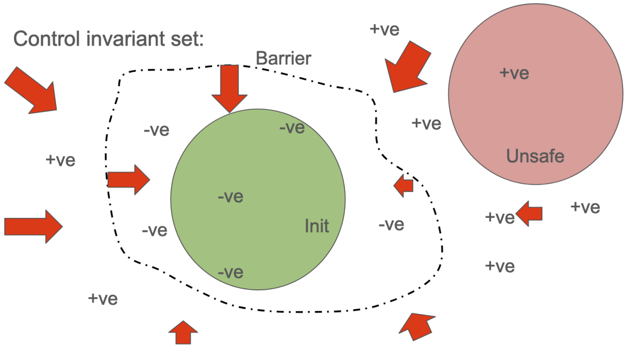
Control Barrier Functions - [Ames et al.]



- $B(\vec{x}) > 0$ for all $\vec{x} \in X_u$ (B is **positive** when unsafe)
- $B(\vec{x}) < 0$ for all $\vec{x} \in X_i$ (B is **negative** when init)
- $B(\vec{x}) = 0$ implies there **exists a control input** $\vec{u} \in U$ such that $\nabla B(\vec{x}) \cdot f(\vec{x}, \vec{u}) < 0$

Control Barrier Functions - Exponential [Kong et al.]

- State: $\vec{x} \in \mathbb{R}^n$
- Control inputs: $\vec{u} \in \mathbb{R}^m$
- $\dot{\vec{x}} = f(\vec{x}, \vec{u}),\ X \subseteq \mathbb{R}^n,$



- $B(\vec{x}) > 0$ for all $\vec{x} \in X_u$ (B is **positive** when unsafe)
- $B(\vec{x}) < 0$ for all $\vec{x} \in X_i$ (B is **negative** when init)
- for all $\vec{x} \in \mathbb{R}^n$ there **exists a control input** $\vec{u} \in U$ s.t. $\nabla B(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq -\lambda B(\vec{x})$

Control Barrier Functions - Exponential [Kong et al.]

It's a hard problem:

- Trying to synthesize *Barrier* and *Control* simultaneously
- Bilinearity

- State: $\vec{x} \in \mathbb{R}^n$
- Control inputs: $\vec{u} \in \mathbb{R}^m$
- $\dot{\vec{x}} = f(\vec{x}, \vec{u})$, $X \subseteq \mathbb{R}^n$,

- $B(\vec{x}) > 0$ for all $\vec{x} \in X_u$ (B is **positive** when unsafe)
- $B(\vec{x}) \leq 0$ for all $\vec{x} \in X_i$ (B is **negative** when init)
- for all $\vec{x} \in \mathbb{R}^n$ there **exists a control input** $\vec{u} \in U$ s.t. $\nabla B(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq -\lambda B(\vec{x})$

Barrier Synthesis using SOS

$$\text{Find} \quad B(\vec{x})\ s.t.$$

$$\left.\begin{aligned} \forall \vec{x} \in X_u, B(\vec{x}) > 0 \\ \forall \vec{x} \in X_o, B(\vec{x}) < 0 \\ \forall \vec{x}, \nabla B(\vec{x}) \cdot f(\vec{x}) \leq -\lambda B(\vec{x}) \end{aligned}\right\}$$

Enforced using SOS

$+$

Putinar's Positivstellensatz

[Parillo et al.]

Certifying SOS Programs

Verify that numerical issues do not invalidate the SOS programming results.

- Each barrier has multiple entailment relations:

$$p_1(\vec{x}) \geq 0, \ldots, p_m(\vec{x}) \geq 0 \ \models \ p \geq 0\,,$$

- Certify via a Putinar positivstellensatz proof that states that

$$\exists\ \sigma_1, \ldots, \sigma_m \in \mathrm{SOS}_d[\vec{x}]\ p - \sigma_1 p_1 - \cdots - \sigma_m p_m \in \mathrm{SOS}_d[\vec{x}]\,,$$

($\mathrm{SOS}_d[\vec{x}]$ represents the set of all SOS polynomials over $\vec{x}$ of degree at most d)

Certifying SOS Programs

$$\begin{cases} B_i(\vec{x}) > 0; \forall \vec{x} \in X_u \\ B_i(\vec{x}) \leq 0; \forall \vec{x} \in X_i \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq \lambda B_i(\vec{x}) \end{cases}$$

$$\begin{aligned} B_i(\vec{x}) &\equiv \Sigma \alpha_i p_i + \alpha_0 \\ -B_i(\vec{x}) &\equiv \Sigma \beta_i q_i + \beta_0 \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) - \lambda B_i(\vec{x}) &\equiv \Sigma \sigma_i r_i + \sigma_0 \end{aligned}$$

$$\alpha_i, \beta_i, \sigma_i, \ldots \implies m(\vec{x})^\top Q_i m(\vec{x})$$

Q_i should be **positive semi-definite**

Certifying SOS Programs

How to certify:

- output the polynomials $\sigma_1, \ldots, \sigma_m$
- compute the "residue" $p - \sigma_1 p_1 - \cdots - \sigma_m p_m$
- obtain a representation $\sigma_i = m(\vec{x})^\top Q_i m(\vec{x})$
- verify that Q_i is positive semi-definite by computing its Cholesky decomposition

The C++ library *Eigen* was used to carry out the Cholesky decomposition using 512 bit floating point representation

Robust Sum of Squares

$$\begin{cases} B_i(\vec{x}) > 0; \forall \vec{x} \in X_u \\ B_i(\vec{x}) \leq 0; \forall \vec{x} \in X_i \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq \lambda B_i(\vec{x}) \end{cases}$$

$$\begin{aligned} B_i(\vec{x}) &\equiv \Sigma \alpha_i p_i + \alpha_0 \\ -B_i(\vec{x}) &\equiv \Sigma \beta_i q_i + \beta_0 \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) - \lambda B_i(\vec{x}) &\equiv \Sigma \sigma_i r_i + \sigma_0 \end{aligned}$$

$$\alpha_i, \beta_i, \sigma_i, \ldots \implies m(\vec{x})^\top Q_i m(\vec{x})$$

Q_i should be **positive semi-definite**

Robust Sum of Squares

$$\begin{cases} B_i(\vec{x}) > 0; \forall \vec{x} \in X_u \\ B_i(\vec{x}) \leq 0; \forall \vec{x} \in X_i \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq \lambda B_i(\vec{x}) \end{cases}$$

$$\begin{aligned} B_i(\vec{x}) &\equiv \Sigma \alpha_i p_i + \alpha_0 + \alpha_{DSOS} \\ -B_i(\vec{x}) &\equiv \Sigma \beta_i q_i + \beta_0 + \beta_{DSOS} \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) - \lambda B_i(\vec{x}) &\equiv \Sigma \sigma_i r_i + \sigma_0 + \sigma_{DSOS} \end{aligned}$$

$$\alpha_i, \beta_i, \sigma_i, \ldots \implies m(\vec{x})^\top Q_i m(\vec{x})$$

Q_i should be **positive semi-definite**

Robust Sum of Squares

$$\begin{cases} B_i(\vec{x}) > 0; \forall \vec{x} \in X_u \\ B_i(\vec{x}) \leq 0; \forall \vec{x} \in X_i \\ \nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) \leq \lambda B_i(\vec{x}) \end{cases}$$

$$B_i(\vec{x}) \equiv \Sigma \alpha_i p_i + \alpha_0 + \alpha_{DSOS}$$
$$-B_i(\vec{x}) \equiv \Sigma \beta_i q_i + \beta_0 + \beta_{DSOS}$$
$$\nabla B_i(\vec{x}) \cdot f(\vec{x}, \vec{u}) - \lambda B_i(\vec{x}) \equiv \Sigma \sigma_i r_i + \sigma_0 + \sigma_{DSOS}$$

$$\alpha_i, \beta_i, \sigma_i, \ldots \implies m(\vec{x})^\top Q_i m(\vec{x})$$

Q_i should be **positive semi-definite**

Can we automate these proofs in LEAN?

Example: Robot Avoiding Obstacle

- Dynamical system with circular unsafe region.
- Synthesize $B(\vec{x})$ to separate safe and unsafe sets.
- Use SOS programming to certify $\dot{B}(\vec{x}) \leq 0$.

Tools and Workflow

- Modeling: MATLAB, Python, **Julia**
- SOS Programming: SOSTOOLS, YALMIP, **SumOfSquares.jl**
- SDP Solvers: SeDuMi, SDPT3, MOSEK, **CSDP**

Example Workflow

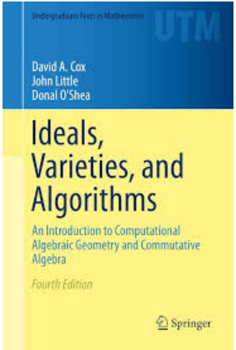
- Define variables and constraints.
- Encode certificate (invariant, barrier, etc).
- Run SOS optimization.
- Export certificate and verify.

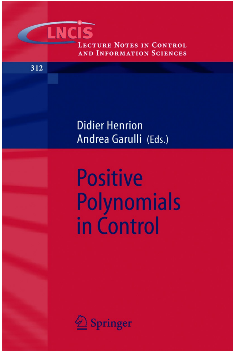
To jupyter notebook ...

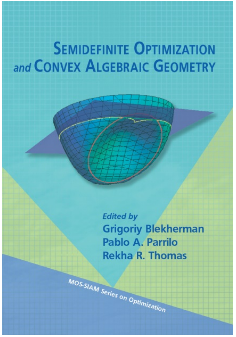
Takeaways

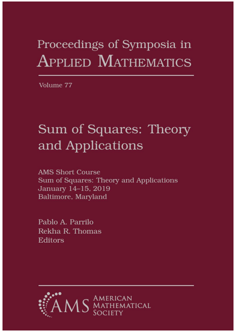
- SOS provides efficient method for real-valued verification.
- Positivstellensatz connects constraints to proof.
- SOS + SDP scales better than symbolic QE.
- Useful in hybrid systems, control, optimization.

Further Reading









www.sumofsquares.org

CUPLV