

Verification using Constraint Solving

Ashish Tiwari

SRI International

333 Ravenswood Ave

Menlo Park, CA 94025

Supported in part by NASA IRAC NRA grant number **NNX08AB95A** and
NSF grants **CNS-0720721** and **CSR-0917398**

Formal Verification

Formal verification gives **correctness guarantees** – for **all possible behaviors**

1. **Model** the **system**
 - (a) Can include **plant, controller** – abstract or refined
 - (b) Can include **disturbances, unknown parameters**
2. **Specify** the **property** – safety, stability
3. Formally **verify** if **property** is ever violated in the system **model**

Why Formal Verification?

Why use formal verification?

1. Alternative to doing **simulation** and **testing**
2. Equivalent to doing an **analytic proof**
3. Do a **new proof**, or **machine check/validate** a hand proof
4. Verify **different** safety and stability **properties**
5. Redo proofs if design is **changed**
6. Applies to both **design** and **implementation**
7. **Carry proofs** with design/code... help in **certification**

Classical Verification Approaches

Broadly classified into two groups–

- **Static**
 - Iterative over-approximation of the reachable set
 - Abstraction
- **Dynamic or Run-Time**
 - Monitoring
 - Smart simulations

Bounded Verification

Bounded Verification is a **different static technique** for **Safety and Stability verification** of **Discrete, Continuous** and **Hybrid** dynamical **systems**

- Reduce **verification problem** to **constraint solving**
- Use **modern constraint solvers** to solve the constraint

Outline/Summary

Part I: Bounded Verification

Part II: Solving $\exists\forall$ formulas

Part III: Analyzing adaptive flight control

Part I: Bounded Verification

Bounded Verification

A generic static approach for verification based on symbolic constraint solving

Key Observation 1: Verification = searching for right witness

Property	Witness
Stability	Lyapunov function
Safety	Inductive Invariant
Liveness	Ranking function
Controllability	Controlled Invariant

How to find the right witness?

Finding the Witness

Key idea 2: Bounded search for witnesses of a **specific form**

High-level outline of the procedure:

1. Fix a form (**template**) for the witness function

Quadratic template: $ax^2 + by^2$

2. Existence of a witness (of the chosen form) is encoded as a **constraint**

$$\exists a, b : \forall x, y : ax^2 + by^2 \geq c \Rightarrow \frac{d}{dt}(ax^2 + by^2) < 0$$

3. Solve the constraint

Overview of Bounded Verification

Given a system over $S(\vec{x}, \vec{x}_d)$, and (optionally) property P :

- Guess template for the witness $W(\vec{a}, \vec{x})$
- Guess template for the assumption $A(\vec{b}, \vec{x}_d)$ (if any)
- Generate the $\exists\forall$ **verification condition**: $\exists\vec{a}, \vec{b} : \forall\vec{x}, \vec{x}_d : A(\vec{b}, \vec{x}_d) \wedge \dots \Rightarrow \phi$
 - Formula ϕ states that W is a witness for S and P
- **Solve** the formula to get values for \vec{a} and \vec{b}

Example of Bounded Verification

Consider the **system**:

$$\begin{aligned}\frac{dx_1}{dt} &= -x_1 - x_2 \\ \frac{dx_2}{dt} &= x_1 - x_2 + x_d\end{aligned}$$

Initially: $x_1 = 0, x_2 = 1$

Property: $|x_1| \leq 1$ always

Guess

- Template for **witness** $W := ax_1^2 + bx_2^2 + c$
- Template for **assumption** $A := |x_d| < d$

Example Continued

Verification Condition: $\exists a, b, c, d : \forall x_1, x_2, x_d :$

$$x_1 = 0 \wedge x_2 = 1 \Rightarrow W \leq 0$$

$$A \wedge W = 0 \Rightarrow \frac{dW}{dt} < 0$$

$$W \leq 0 \Rightarrow |x_1| \leq 1$$

Ask constraint solver for satisfiability of above formula

Solver says: $a = 1, b = 1, c = -1, d = 1$

$$x_1 = 0 \wedge x_2 = 1 \Rightarrow x_1^2 + x_2^2 - 1 \leq 0$$

$$|x_d| < 1 \wedge x_1^2 + x_2^2 - 1 = 0 \Rightarrow 2x_1(-x_1 - x_2) + 2x_2(x_1 - x_2 + x_d) < 0$$

$$x_1^2 + x_2^2 - 1 \leq 0 \Rightarrow |x_1| \leq 1$$

This **proves** that $|x_1| \leq 1$ always.

Related Work

The bounded verification approach encompasses

- Template-based invariant generation (SankaranarayananSM04, T04, Kapur05)
- Barrier certificates (PrajnaJ04)
- Constraint-based approach for verification (GulwaniT08)

Bounded verification is the dual of **bounded falsification**
(or, bounded model checking)

The **real** problem is **deciding $\exists \forall$ formulas over the reals**

Part II: Solving $\exists\forall$ formulas

Solving $\exists\forall$ formulas

Bounded verification: verification of hybrid systems \mapsto checking validity of

$$\exists \vec{u} : \forall \vec{x} : \phi$$

When ϕ contains only polynomials, this is decidable (e.g. QEPCAD)

We are developing more **practical** procedures to decide $\exists \vec{u} : \forall \vec{x} : \phi$

By adapting our procedure for solving $\forall \vec{x} : \phi$

Procedure for Nonlinear Reals

The approach is **generalization** of **Simplex**

- Introduce **slack variables** s.t. all inequality constraints are of the form $v > 0$, or $w \geq 0$

$$\begin{array}{l} P = 0, \quad Q > 0, \quad R \geq 0 \quad \mapsto \\ \underline{P = 0}, \quad \underline{Q - \vec{v} = 0}, \quad \underline{R - \vec{w} = 0}, \quad \vec{v} > 0, \quad \vec{w} \geq 0 \end{array}$$

- **Search** for a polynomial p s.t.

$$\begin{array}{l} \underline{P = 0} \quad \Rightarrow \quad p = 0 \\ \vec{v} > 0, \quad \vec{w} \geq 0 \quad \Rightarrow \quad p > 0 \end{array}$$

- To search for p , compute the **Gröbner basis** for P using different possible orderings (pivot)

Example

$$|x_d| < 1 \wedge x_1^2 + x_2^2 - 1 = 0 \Rightarrow 2x_1(-x_1 - x_2) + 2x_2(x_1 - x_2 + x_d) < 0$$

$$-1 < x_d, \quad x_d < 1, \quad x_1^2 + x_2^2 = 1, \quad -2x_1^2 - 2x_2^2 + 2x_2x_d \geq 0$$

$$x_d = v_1 - 1, \quad x_d = -v_2 + 1, \quad x_1^2 = -x_2^2 + 1, \quad -2x_1^2 = 2x_2^2 - 2x_2x_d + w$$

$$x_d = v_1 - 1, \quad v_1 = -v_2 + 2, \quad x_1^2 = -x_2^2 + 1, \quad 2x_2x_d = w + 2$$

$$4x_2^2x_d^2 = (w + 2)^2$$

$$4x_d^2 = 4x_1^2x_d^2 + (w + 2)^2$$

$$0 = 4v_1v_2 + 4x_1^2x_d^2 + w^2 + 4w$$

Positivstellensatz

What guarantees the existence of such a witness?

The constraint

$$\{p = 0 : p \in P\} \cup \{q \geq 0 : q \in Q\} \cup \{r \neq 0 : r \in R\}$$

is unsatisfiable (over the reals) iff

there exist polynomials p , q , and r such that

$$p \in \text{Ideal}(P)$$

$$\{\sum_i p_i q_i : p_i \in P\}$$

$$q \in \text{Cone}[Q]$$

$$\{\sum_i s_i^2 q_1 q_2 \dots q_k : q_j \in Q\}$$

$$r \in [R]$$

$$\{r_1 r_2 \dots r_k : r_i \in R\}$$

$$p + q + r^2 \equiv 0$$

We are searching for this “ $p + q + r^2$ ” by pivoting

Discussion and Conclusion

- The template+constraint-solving approach is **different** from the **usual** verification approaches
 - **reachability, abstraction**
- **Bounded Falsification (BMC)** vs. **Bounded Verification**
- This is the **classical** approach – only **slightly modified** to
 - generate more **precise non-convex** constraints
 - solved using modern solvers
- Systems have **several** invariants/Lyapunov functions – that can be searched using **few** templates
- **Correct systems have simple witnesses**

Future Challenges

- **Proof rules** for verification of hybrid systems
- Solver for **nonlinear constraints**
 - Solver for \exists formulas
 - Solver for $\exists\forall$ formulas
 - Need to balance **completeness** and **efficiency**
 - Domain-specific **heuristics**
- **Automation for template generation** for specific domains
- **Runtime Analysis: monitor** the invariants
- **Cheaper invariants using cheaper analysis**
- **Compose assumptions and guarantees**