Verification and Synthesis

Using Real Quantifier Elimination

Thomas Sturm

Max-Planck-Institute for Informatik

Saarbrucken, Germany

sturm@mpi-inf.mpg.de

Ashish Tiwari

SRI International

Menlo Park, USA

tiwari@csl.sri.com

Formal Methods

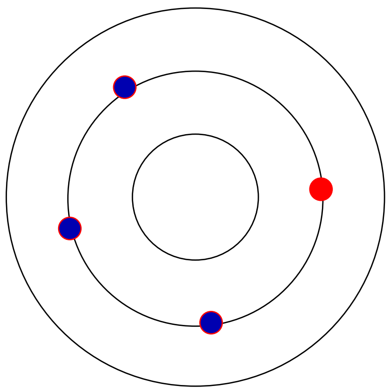
Model and analyze systems formally

Two aspects:

- Formal model of dynamical system M
- Formal property specification ϕ

Example:

$$\begin{aligned} M &:= \{\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x\} \\ \phi &:= (x = 1 \wedge y = 0 \Rightarrow \mathbb{G}(x \leq 1)) \end{aligned}$$



Verification Problem: Prove $M \models \phi$

Certificate-Based Verification

A certificate for $M \models \phi$ is Φ such that

1. $\models \Phi \Rightarrow \phi$
2. $M \models \Phi$ is locally checkable
   $M \models \Phi$ reduces to a formula in the (underlying FO) logic

Examples:

Property ϕ	Certificate Φ
safety	inductive invariant
stability	Lyapunov function
termination	ranking function
controlled safety	controlled inductive invariant

Certificate-Based Verification

Certificate-based verification reduces the verification problem to an $\exists\forall$ formula.

$$M \models \phi$$

$$\Uparrow$$

$$\exists \Phi : ((M \models \Phi) \ \wedge \ (\Phi \Rightarrow \phi))$$

$$\Uparrow$$

$$\exists \Phi : \forall \vec{x} : \text{ quantifier-free FO formula}$$

$$\Uparrow$$

$$\exists \vec{a} : \forall \vec{x} : \text{ quantifier-free FO formula}$$

The last step performed by choosing a template for Φ

Example: Certificate-Based Safety

Example: $$\frac{dx_1}{dt} = x_2 \qquad \frac{dx_2}{dt} = -x_1$$

Problem: If $x_1 = 1$ and $x_2 = 0$ initially, prove $G(x_1 \le 1)$

Let us find a certificate of the form $p \le 0$ where $p := ax_1^2 + bx_2^2 + c$

We need to solve

$$\exists a, b, c : \forall x_1, x_2 : \quad \begin{array}{l} (p = 0 \Rightarrow \frac{dp}{dt} \le 0) \ \wedge \\ (x_1 = 1 \wedge x_2 = 0 \Rightarrow p \le 0) \ \wedge \\ (p \le 0 \Rightarrow x_1 \le 1) \end{array}$$

We get $p := x_1^2 + x_2^2 - 1$. Proved.

Certificate-Based Verification: Observations

A generic approach for verification based on symbolic constraint solving

- Observation 1: Verification = searching for right witness
- Observation 2: Bounded search for witnesses of a specific form
- Net result: Verification problem $\mapsto$ $\exists\forall$ problem

$\exists\forall$ formula depends on the property ϕ and certificate Φ

Can also handle uncontrollable inputs/noise

Example: Certificate-based Verification

Consider the system M:

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

Initially: $x_1 = 0, x_2 = 1$

Property: $|x_1| \leq 1$ always

Guess

- Template for witness $\Phi := W \leq 0$, where $W := ax_1^2 + bx_2^2 + c$
- Template for assumption $A := |x_d| < d$

Example Continued

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

$$\begin{aligned} x_1 = 0 \ \wedge \ x_2 = 1 &\Rightarrow W \leq 0 \\ A \ \wedge \ W = 0 &\Rightarrow \frac{dW}{dt} < 0 \\ W \leq 0 &\Rightarrow |x_1| \leq 1 \end{aligned}$$

Ask contraint solver for satisfiability of above formula

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

$$\begin{aligned} x_1 = 0 \ \wedge \ x_2 = 1 &\Rightarrow x_1^2 + x_2^2 - 1 \leq 0 \\ |x_d| < 1 \ \wedge \ x_1^2 + x_2^2 - 1 = 0 &\Rightarrow 2x_1(-x_1 - x_2) + 2x_2(x_1 - x_2 + x_d) < 0 \\ x_1^2 + x_2^2 - 1 \leq 0 &\Rightarrow |x_1| \leq 1 \end{aligned}$$

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

Solving ∃∀ Formulas

Two symbolic approaches:

- Virtual Substitution: scalable, but limited applicability
- Cylindrical Algebraic Decomposition: general, but unscalable

Combination Approach for QE

Solve quantified formula ϕ:

- ϕ_1 := apply virtual substitution (`redlog`) on ϕ as long as possible
- ϕ_2 := apply simplifier (`slfq`) to simplify ϕ_1
- if ϕ_2 is $\exists\vec{x} : \bigvee_i \phi_{2i}$

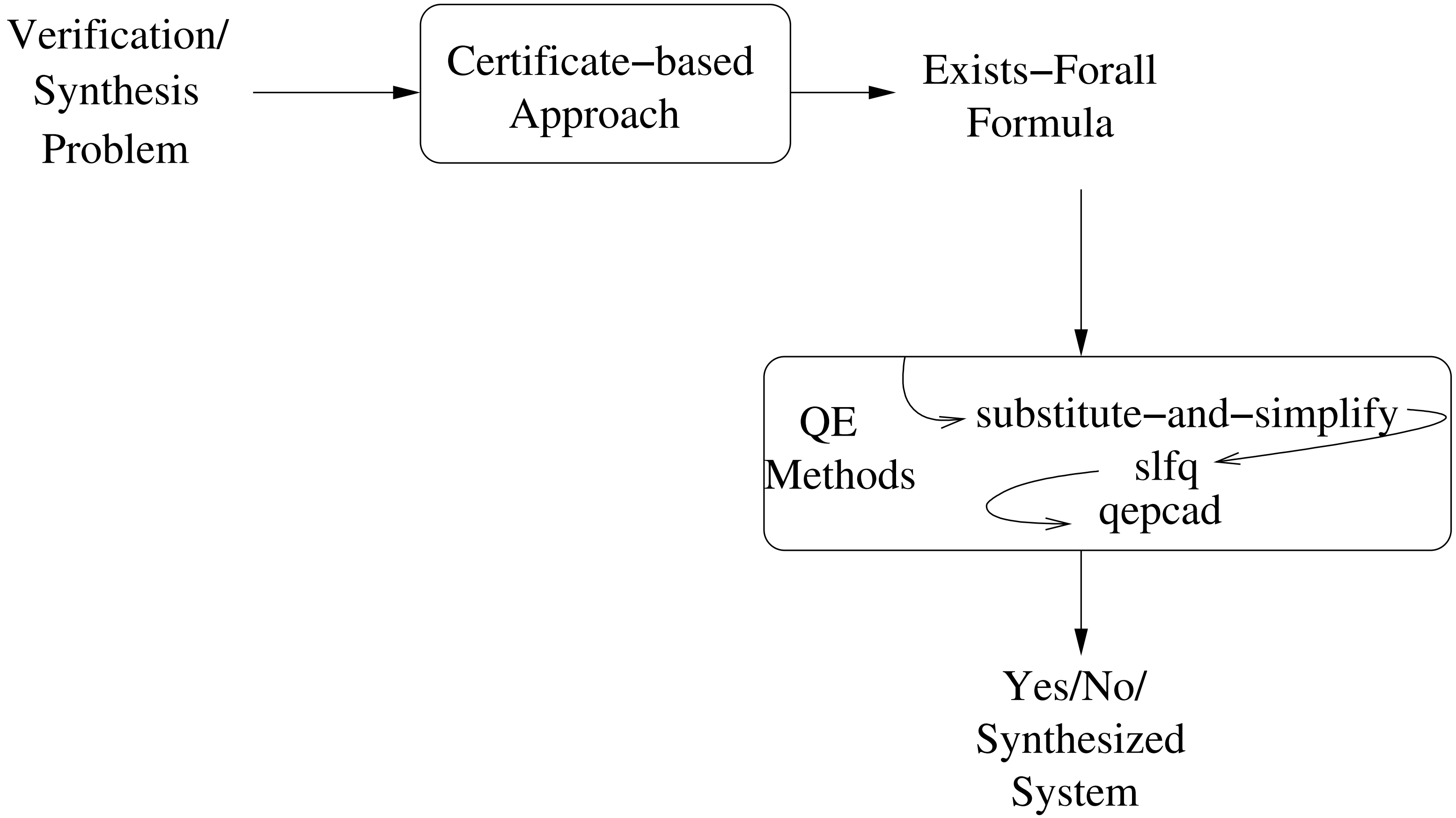
  $\phi_3 := \bigvee_i$ `qepcad`(ϕ_{2i}) // Can be limited to a subset of i's

  else $\phi_3 :=$ `qepcad`(ϕ_2)
- return ϕ_3

The tool `qepcad` used with `Singular`

All components interfaced via Reduce

Overall Approach



Key Observation: Need sufficient formula ψ on $\vec{a}$ s.t. $\psi(\vec{a}) \Rightarrow \forall x : \Psi(\vec{a}, \vec{x})$

Examples

Benchmark examples:

- Adaptive cruise control: verify that cars do not collide
- Robot motion: synthesize safe switching logic
- Adaptive flight control: verify stability
- Inverted pendulum: synthesize stable switching controller

Other examples:

- Navigation benchmarks: Safety verification of hybrid systems
- PID controllers: Stability verification of open controllers
- Train gate controller synthesis
- Others: LCR circuit, thermostat, insulin infusion pump controller

Adaptive Cruise Control

Consider a cruise control:

$$
\begin{aligned}
\dot{v} &= a \\
\dot{gap} &= -v + v_f \\
\dot{v_f} &= a_f \\
\dot{a} &= -4v + 3v_f - 3a + gap \qquad \text{Controller}
\end{aligned}
$$

where v, a is velocity and acceleration of this car, v_f, a_f is the same for car in front, and gap is the distance between the two cars.

Physical limits puts constraints on v, v_f, a, a_f.

Adaptive Cruise Control

Goal: Find initial states such that, if ACC mode is initiated in those states, then cars will not collide.

Solution: Pick a linear template for the initial states $\texttt{Init}(\vec{a})$ and for the inductive invariant $\texttt{Inv}(\vec{b})$ and solve the resulting $\exists\forall$ formula.

The formula states that there exists $\vec{a}$ and $\vec{b}$ such that

(1) all initial states in $\texttt{Init}(\vec{a})$ are also in $\texttt{Inv}(\vec{b})$, and

(2) all states in $\texttt{Inv}(\vec{b})$ are in Safe, and

(3) the system dynamics cannot force the system to go out of the set $\texttt{Inv}(\vec{b})$

Formulas encoding (1),(2),(3) are $\forall$ formulas

Adaptive Cruise Control: Analysis

Complexity of the generated $\exists\vec{a} : \forall\vec{x} : \phi$ formula:

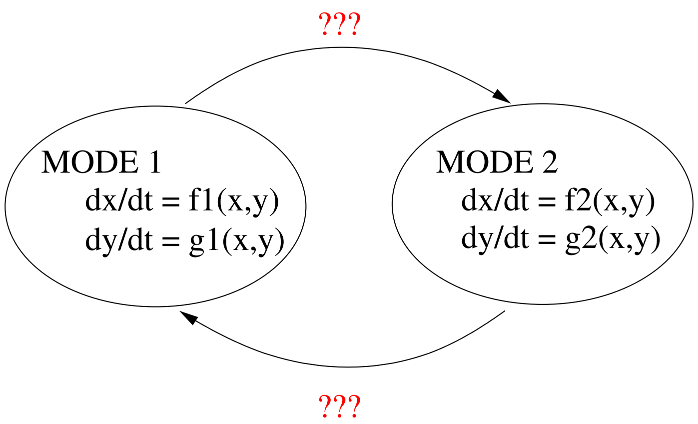
- $|\vec{a}| = 4$
- $|\vec{x}| = 5$
- $\texttt{degree}(\phi) = 2$

Results:

- Virtual substitution eliminates all but one variable
- Returns a disjunction of 584 subformulas containing 33365 atomic formulas (nested to depth 13)
- Simplifier `slfq` fails
- But succeeds on part of the formula
- That is sufficient to give a useful answer

Switching Logic Synthesis

Do not verify, synthesize correct systems



Problem: Under what conditions to switch between the components so that final system is safe.

Solution: Find a set of states (Φ) within which the two modes can keep the system

Examples: robot motion, thermostat, inverted pendulum

Adaptive Flight Control: Model

Goal: Verify an adaptive flight controller

Flight controller: Keeps the plane stable in flight

Adaptive: Learn and compensate for damages, aging and so on

The dynamics of the aircraft are given by

$$\dot{\vec{x}} = A\vec{x} + B\vec{u} + G\vec{z} + f(\vec{x}, \vec{u}, \vec{z}) \quad (1)$$

where
$\vec{x}$: 3×1 vector of roll, pitch, and yaw rates of the aircraft
$\vec{u}$: 3×1 vector of aileron, elevator, and rudder inputs
$\vec{z}$: 3×1 trim state vector of angle of attack, angle of sideslip, and engine throttle
A, B, G are known matrices in $\Re^{3\times 3}$
f represent the unknown term (uncertainty or damage)

Adaptive Flight Control: Modeling

We built a continuous dynamical system model

State space: $x_m, intx_e, x, L, \beta, f$

$$\begin{aligned}
\dot{x_m} &= A_m(x_m - r) \\
i\dot{n}tx_e &= x_m - x \\
\dot{x} &= A_m(x_m - r) + K_p(x_m - x) + K_i intx_e - L'\beta + f \\
\dot{L} &= -\Gamma\beta(intx_e^T K_i^{-1} + (x_m - x)^T K_p^{-1}(I + K_i^{-1})) \\
\dot{\beta} &= \ldots \\
\dot{f} &= \ldots
\end{aligned}$$

Constants	:	Γ, K_p, K_i, A_m,
Unknown/Symbolic Parameters	:	r, f, $\dot{f}$

Adaptive Flight Control: Analysis

Goal: Show that the error eventually falls below a certain threshold

Assume boundedness of certain expression

The $\exists \vec{a} : \forall \vec{x} : \phi$ formula says that there exists a Lyapunov function (of a given form)

- $|\vec{a}| = 5$
- $|\vec{x}| = 5$
- degree = 4

Output of virtual substitution not simplified by slfq

If certain $\exists$ variables are instantiated, then `slfq` succesfully simplifies output of virtual substution (48 subformulas, depth 10, 1081 atomic formulas) in 27s using 1897 `qepcad` calls to the required answer

Inverted Pendulum

Maintain an inverted pendulum around its unstable equilibrium by controlling the force on the cart on which the pendulum is mounted

$$\begin{aligned}
\frac{dx}{dt} &= v \\
\frac{dv}{dt} &= \frac{(F - ml\omega^2 \sin(\theta) + mg\cos(\theta)\sin(\theta))}{(M + m - m\cos(\theta)\cos(\theta))} \\
\frac{d\theta}{dt} &= \omega \\
\frac{d\omega}{dt} &= (g\sin(\theta) + \cos(\theta)\frac{dv}{dt})/l
\end{aligned}$$

where $F \in \{2, -2, 0\}$

Goal: Synthesize switching controller to maintain safety

Inverted Pendulum: Analysis

Replace trigonometric functions by Taylor approximations

Formula statistics:

- $|\vec{a}| = 2$
- $|\vec{x}| = 2$
- degree = 7

virtual substitution + slfq simplification + partial instantiation + qepcad generates a controlled invariant:

$$-\theta^2 - (300/4801)\omega^2 + (1/100) \geq 0$$

PI Controller

PI controller: A generic controller for driving an unknown plant to some setpoint

Controller:

$$\frac{d\texttt{interr}}{dt} = \begin{cases} \texttt{err} & \text{if } \texttt{interr}^2 = 1 \ \wedge \\ & \quad \texttt{err} * \texttt{interr} < 0 \\ \texttt{err} & \text{if } \texttt{interr}^2 < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$u = K_p * \texttt{err} + K_i * \texttt{interr}$$

Plant:

$$\frac{dx}{dt} = \beta - \alpha * u$$

$$\alpha \in [a, b]$$

$$\beta \in [a_1, b_1]$$

What plants can the PI controller successfully control?

PI Controller: Analysis

Formula:

- $|\vec{a}| = 6$
- $|\vec{x}| = 4$
- degree = 2

Virtual substitution is usually fast `slfq` takes about 200 seconds, 9000 qepcad calls

Theorem: Suppose the controller gains satisfy:

$$K_p \geq 500 \;\wedge\; K_p \geq K_i \;\wedge\; K_p + K_i \geq 500$$

and suppose $a > 0$, $b = +\infty$, $a_1 = -500 * a$ and $b_1 = 500 * a$. Then, the PI feedback control system always eventually reaches a state where $\mathtt{err}^2 \leq 1$.

Conclusion

QE procedure:

- Virtual substitution + slfq + qepcad is a potent combination of tools for solving hard QE problems
- Virtual substitution often takes negligible time
- But it generates huge formulas
- `slfq` is crucial for simplifying the large formulas

Verification + benchmarks:

- Verification + synthesis of hybrid systems can be reduced to to $\exists\forall$ formulas
- Maintaining an active webpage of benchmarks
- Apart from Certificate-based methods, constructing relational abstraction also generates $\exists\forall$ formulas

Future work: numeric methods, combining with SMT solvers