Verification and Synthesis Using Real Quantifier Elimination

Thomas Sturm
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
sturm@mpi-inf.mpg.de

Ashish Tiwari*
SRI International
Menlo Park, CA 94025
tiwari@csl.sri.com

ABSTRACT

We present the application of real quantifier elimination to formal verification and synthesis of continuous and switched dynamical systems. Through a series of case studies, we show how first-order formulas over the reals arise when formally analyzing models of complex control systems. Existing off-the-shelf quantifier elimination procedures are not successful in eliminating quantifiers from many of our benchmarks. We therefore automatically combine three established software components: virtual subtitution based quantifier elimination in Reduce/Redlog, cylindrical algebraic decomposition implemented in Qepcad, and the simplifier Slfq implemented on top of Qepcad. We use this combination to successfully analyze various models of systems including adaptive cruise control in automobiles, adaptive flight control system, and the classical inverted pendulum problem studied in control theory.

Categories and Subject Descriptors: I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; I.1.4 [Symbolic and Algebraic Manipulation]: Applications; I.6.4 [Simulation and Modeling]: Model Analysis

General Terms: Algorithms, Experimentation, Verification

Keywords: Formal verification, Safety, Stability, Lyapunov functions, Inductive invariants, Controller synthesis

1. INTRODUCTION

Physical processes in the world around us are often modeled using the real numbers. The temperature of a room, the speed of a car, the angle of descent of an airplane, relative population of a species, protein concentration in a cell, blood glucose concentration in a human, and the amount of a chemical in a tank are a few of the countless quantities that arise in science and engineering and that are modeled using real-valued variables. Many of these physical quantities are being controlled by computer software implementing sophisticated control algorithms. The resulting system – consisting of a physical plant and a software controller – is called a *cyber-physical system*. These systems are often employed in safety-critical applications. Before a newly designed complex system, such as a new flight-control system or a modern automobile cruise control module, can be certified and deployed, it is imperative to guarantee its correctness. One of the most pressing needs today is to develop tools that can formally analyze, and possibly also correctly synthesize, such systems.

The current practice in verification of cyber-physical systems is based on performing extensive (numerical) simulation and testing. However, simulation-based methods are incomplete – how do we know that the system has been tested enough? In the past few years, there has been an extensive push for extending formal verification approaches to also verify physical and cyber-physical systems. Broadly speaking, these techniques can be classified as follows:
(a) *reach-set methods* that compute the set of all reachable states of the system, either exactly [21], or approximately [34]
(b) *abstraction-based methods* that first abstract the system and then analyze the abstraction [33]
(c) *certificate-based methods* that directly search for certificates of correctness (such as inductive invariants and Lyapunov functions) of systems [30, 23, 26, 22, 18]

While all these techniques have had some success, the certificate-based methods are turning out to be particularly effective in proving deep properties of complex systems. At the core of all certificate-based methods, and also many of the other verification methods above, lie reasoning engines that perform inferences in the theory of reals.

During the past 40 years significant advances have been made in tools and techniques for performing quantifier elimination in the first-order theory of reals (QE). For practical purposes, substitution methods [35, 37] and their implementation in `redlog` [12] play a prominent role for formulas where the degrees of the quantified variables are small. For general formulas, partial cylindrical algebraic decomposition (CAD) and its implementation in the `qepcad` package is an important tool [8, 2, 9, 6]. Another efficient implementation of CAD exists in the commercial computer algebra system Mathematica, and a less mature one in `redlog`. For both, substitution methods and CAD, simplification of quantifier-free results is important [13, 7]. This is particularly true for substitution methods. While these methods typically have

*Research partially funded by DARPA under contract FA8650-10-C-7078, NSF grants CSR-0917398 and SHF:CSR-1017483, and NASA grant NNX08AB95A.

ISSAC'11, June 8–11, 2011, San Jose, California, USA.

been referred to as *virtual substitution*, recently the term *substitute-and-simplify*[1] has been coined, which adequately describes the equal relevance of the core elimination method and the simplification aspect.

In this paper we reduce a class of verification and synthesis problems that arise when analyzing cyber-physical systems to quantifer elimination problems over the reals. It turns out that the corresponding formulas are computationally too hard to be solved by the established software. Our approach here is to automatically combine several systems to a fully automatic procedure suitable for our purposes.

The goal of this paper is threefold:
(1) Present the application of quantifier elimination to the verification and synthesis of switched and hybrid dynamical systems
(2) Present some benchmarks and details on how they were generated so that the larger community can benefit from the availability of real benchmarks and improvements in the technology for performing quantifier elimination can be calibrated[2]
(3) Present a novel combination of tools and techniques for performing quantifier elimination that is more scalable and promising than any of its components

In Section 2 we describe the design and the architecture of our software. In Sections 3 we present several verification and synthesis case studies that are challenging for most of the existing analysis tools, but that we were able to successfully solve using the certificate-based approach implemented using quantifier elimination. In Section 4 we summarize and evaluate our results.

2. A META-QE PROCEDURE FOR CONTROL THEORY

To start with, let us recall some basic asymptotic complexity results for real quantifier elimination, the substitute-and-simplify method, and CAD. For simplicity, we restrict ourselves to prenex formulas. In the worst case, the running time of real QE is asymptotically bounded from both above and below by a double exponential function in the length of the input formula [35, 10]. With CAD the relevant complexity parameter is the number of variables, where it makes no difference whether or not they are quantified. With substitution methods, in contrast, the quantification significantly contributes to the complexity: The procedure is doubly exponential only in the number of quantifier alternations. For a fixed number of quantifer alternations the number of quantifiers contributes only singly exponential to the complexity, and unquantified variables contribute only polynomially [35, 37].

While the substitution method is appealing from the complexity standpoint, it is limited in its applicability to formulas of low degree in the quantified variables. More precisely, assuming w.l.o.g that all right-hand sides of real constraints are 0 and that all left-hand sides are expanded to distributive polynomials, the maximal total degree in the quantified variables must not exceed 2. Moreover, with the successive elimination of the quantifiers from the inside to the outside, the degrees of outer quantifiers possibly increase so that before elimination it is not even predictable whether or not the elimination procedure will succeed. In principle, substitution methods can be generalized to arbitrary degrees [37, 36], but these extensions have not been implemented so far, and it is unclear whether or not they are practical.

Our first idea is to apply substitute-and-simplify as long as there occur no degree violations, and then to use CAD for the remaining problem. Obviously, we benefit from the advantages of the substitution method wrt. complexity at the beginning, and later on we possibly enter CAD with a formula where several variables have already been eliminated. In addition, the quantifier-free formula establishing the final result will be generated by CAD, yielding another advantage: It is well-known that elimination results of CAD, in contrast to most other real QE methods, are very concise, nonredundant, and intuitively interpretable. In this raw form, our approach has been available within `redlog` and mentioned in several presentations by the first author since 2005. The CAD used then was always that of `redlog`.

Our present work refines this idea in several ways. First, we use `qepcad` instead of `redlog`'s own CAD. This improves performance of the overall method. Second, `qepcad` is not used directly after a degree violation. Before invoking `qepcad`, we apply `slfq` to the quantifier-free part of the intermediate result. `slfq` is a simplifier for quantifier-free formulas over the reals that uses multiple `qepcad` calls on subformulas in a divide-and-conquer approach [7]. Simplification using `slfq` significantly reduces the size of the intermediate formula generated by the substitution method and makes `qepcad` run much faster. Third, when using `qepcad` to eliminate quantifiers, we distribute existential quantifiers over top-level disjunctions and solve several smaller QE problems, $\exists \vec{x} : \phi_i$, in place of one large QE problem $\exists \vec{x} : \vee_i \phi_i$. This optimization is frequently usable in our application since all our input formulas are of the form "exists-forall", and substitution methods turn out to successfully eliminate all the universal variables and some of the existential variables. Recall that the successive elimination of existential quantifiers via substitution methods systematically yields comprehensive disjunctions. Hence, the result produced by the substitution method is often of the form $\exists \vec{x} : \vee_i \phi_i$, enabling the optimization above. Note also that `slfq` preserves this form unless it is particularly successful. A more important benefit of this optimization is the following: If `qepcad` fails due to limited time or space on some of the subproblems, the disjunction of the elimination results for the successful subproblems still yields a sufficient (although not necessary) quantifier-free condition for the original question. In our application, this partial result is still enough to successfully complete the analysis task; see the example in Section 3.1.

To complete the picture, we can also automatically use the CAD of Mathematica instead of `qepcad`. However, we did not observe any significant advantage of Mathematica on our benchmarks and, hence, we do not discuss it further. Finally, `qepcad` itself can optionally spawn `Singular` [11] processes to speed up its Gröbner basis computations. We systematically made use of this feature and observed that this considerably improved our computation times.

All relevant software components, i.e., Reduce/`redlog`, `slfq`, `qepcad`, and `Singular`, as well as all the benchmarks discussed in this paper (and many others), are freely available on the Web [32]. Using the current head version of Reduce, the process communication described here should run out-of-the-box using default installations of `slfq`, `qep-`

[1] In an ISSAC 2010 presentation by C. Zengler
[2] The website [32] contains all benchmark described in this paper and many others, too.

`cad`, and `Singular`. The system can be used either in batch or interactively. With interactive use, Reduce serves as the interface, and all results obtained from external components are available there for further processing.

We also experimented with using *generic* quantifier elimination [14], which makes some nondegeneracy assumptions during the initial substitute-and-simplify elimination step. The assumptions made were found to not influence the final outcome, especially in those cases where there was a "robust" verification proof; see Section 3.2 for more discussion.

All computations described here have been carried out on an Intel Xeon E5630 2.53GHz single-core processor (x86_64 arch) with 4G RAM running Ubuntu Linux 2.6.32-26.

3. CERTIFICATE-BASED TECHNIQUES

Given a system, say modeled using differential equations, and given a property, the *verification problem* asks whether the system satisfies the property. Given an incomplete system and given a property, the *synthesis problem* asks whether the system can be completed so that it satisfies the property. A *certificate* for a verification problem (respectively, synthesis problem) is a (Boolean or real valued) function on the state space of the system such that existence of such a function is sufficient, and perhaps even necessary, condition for the verification problem (synthesis problem) to have a positive answer. For example, a Lyapunov function is a certificate for stability verification, and an inductive invariant (also known as barrier certificate) is a certificate for safety verification. A controlled Lyapunov function (respectively, controlled inductive invariant) is a certificate for a synthesis problem whose goal is stability (safety).

Certificate-based methods tranform the verification problem (respectively, the synthesis problem) into a search for an appropriate certificate. The space of certificates is unbounded, and hence, we bound the search for a certificate by fixing the *form* of the certificate. For example, for stability verification, we can restrict search to quadratic Lyapunov functions.

Formally defining certificates for different verification and synthesis problems is beyond the scope of this paper. In each subsequent section, we precisely define a different verification or synthesis problem and then define a certificate for that problem. The search for a certificate is cast as an $\exists\forall$ problem. The overall approach is shown in Figure 1.

3.1 Adaptive Cruise Control: Proving Collision Avoidance

Here, we prove collision avoidance for two cars under cruise control. The rear car uses a cruise control law that actively adjusts its acceleration based on its own velocity, acceleration, the relative velocity of the leading car, and the gap between the two cars. Proving collision avoidance means showing that the two cars will not collide assuming that the cruise control is activated in a safe initial configuration.

The cruise control law is taken from the leader control developed in [16] and also discussed in [24] and [30]; see also the Berkeley Path project [4]. Let gap, v_f, v, and a, respectively, represent the gap between the two cars, the velocity of the leading car, and the velocity and acceleration of the rear car. The dynamics of the velocity of the rear car and the gap are given by

$$\begin{aligned} \frac{dv}{dt} &= a, \quad a \in [-5, 2] \\ \frac{dgap}{dt} &= v_f - v \end{aligned} \tag{1}$$

The restriction of a to be in the range $[-5, 2]$ comes from the physical constraint on the braking and accelerating capability of the car. The dynamics of the acceleration a of the rear car are assumed to be determined by the following control law

$$\dot{a} = -3a - 3(v - v_f) + (gap - (v + 10)) \tag{2}$$

The dynamics of the leading car are given by

$$\frac{dv_f}{dt} = a_f, \quad a_f \in [-5, 2] \tag{3}$$

Note that we do not assume any particular acceleration profile for the leading car. Thus, the leading car can behave nondeterministically as long as its acceleration remains within the $[-5, 2]$ range.

We wish to find the initial states such that if we activate the above cruise control law in those states, then the control law will guarantee that there will be no collision. The initial states we are interested in are of the form

$$Init := (gap = 10 \wedge a = 0 \wedge v_f = c_1 \wedge v = c_2) \tag{4}$$

where c_1, c_2 are parameters. The set of *safe* (collision-free) states is defined by

$$\texttt{safe} := (gap > 0) \tag{5}$$

How to prove collision avoidance for the above (parametric) system? We use the *certificate-based approach* for verification [18, 23, 30, 28]. The idea behind proving safety is to discover an invariant set, *Inv*, such that
(1) all initial states in *Init* are also in *Inv*, and
(2) all states in *Inv* are in `safe`, and
(3) the system dynamics cannot force the system to go out of the set *Inv*
Such a set *Inv* is a *certificate* for safety. The problem is that we do not know *Inv*. We can discover *Inv* by fixing its form. Let us assume a linear form for *Inv*, that is,

$$p := c_3 v + c_4 v_f + c_5 a + gap + c_6, \; Inv := (p \geq 0) \tag{6}$$

Let Inv' denote the constraints imposed by physical reality,

$$Inv' := (a \in [-5, 2] \wedge a_f \in [-5, 2] \wedge v \geq 0 \wedge v_f \geq 0) \tag{7}$$

Now, the three conditions for *Inv* to be a certificate for safety can be encoded in the following formula

$$\begin{aligned} \phi_1 &:= (Init \wedge Inv' \Rightarrow Inv) \\ \phi_2 &:= (Inv \wedge Inv' \Rightarrow (gap > 0)) \\ \phi_3 &:= (p = 0 \wedge Inv' \Rightarrow \frac{dp}{dt} \geq 0) \end{aligned} \tag{8}$$

where dp/dt can be symbolically computed using the definition of p in Equation (6) and the dynamics in Equation (1) and Equation (2) as

$$\begin{aligned} \frac{dp}{dt} := {} & c_3 a + c_4 a_f + \\ & c_5(-3a - 4v + 3v_f + gap - 10) + (v_f - v) \end{aligned} \tag{9}$$

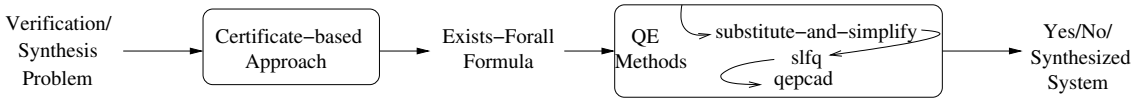

Figure 1: Overall approach for verification and synthesis using quantifier elimination.

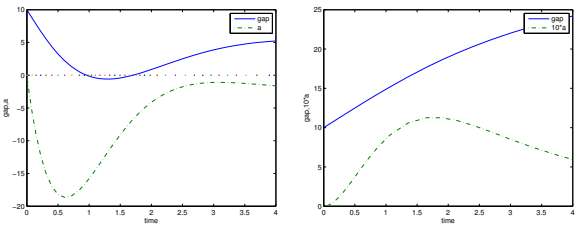

Figure 2: Simulation of the adaptive cruise control system. (Left) Plot of gap, a: The cars collide (gap becomes negative) even though the initial conditions $v = 30, vf = 15$ satisfy the precondition under which safety was established. This occurs because the assumption $a \geq -5$ is also violated. (Right) Plot of $gap, 10a$: The assumptions always hold, and cars do not collide. The initial conditions $v = 15, vf = 20$ satisfy the precondition for safety.

Thus, we can find the conditions on the initial velocities c_1, c_2 that will guarantee collision avoidance by eliminating quantifiers from the following formula (File acc7.red [32])

$$\exists c_3, c_4, c_5, c_6 : \forall v, v_f, gap, a, a_f : (\phi_1 \wedge \phi_2 \wedge \phi_3) \qquad (10)$$

where ϕ_1, ϕ_2, ϕ_3 are as defined in Equation (8) expanded using the definitions given in Equation (4), Equation (6), Equation (7), and Equation (9).

Analysis Results

We use `redlog`, `qepcad`, and `slfq` to eliminate quantifiers from Formula (10) and to then simplify it. Virtual substitution implemented in `redlog` is able to eliminate all but one quantified variable, namely c_5, from the above formula, but it returns a huge formula that is a disjunction of 584 subformulas and contains 33365 atomic formulas (nested to depth 13). The simplifier `slfq` fails to simplify this formula. Since `slfq` fails on the 34th disjunct, we simplify the first 33 disjuncts using `slfq` under the assumption that $c_1 > 0$ and $c_2 > 0$ and obtain $c_2^2 - 30c_2 - 75 \leq 0$, which is, surprisingly, independent of c_1 and c_5. Since we have discarded some disjuncts, the actual equivalent formula would have been $c_2^2 - 30c_2 - 75 \leq 0 \vee \psi$ for some ψ. This means that, if the initial velocity of the rear car is at most 32, then we are guaranteed to be safe. (There may be other cases, too, when we would be safe, since we have lost some solutions in the approximation.) We can verify that our results are correct by instantiating c_1 and c_2 by the respective conditions and proving the implications, which are immediately proved by `redlog` (File acc7-verify.red [32]).

Why is this result interesting? It is because we have established collision freedom without making any assumption on the dynamics of the leading car. So, the leading car is free to choose any driving profile, and the control law would guarantee safety. However, there is one caveat. We have assumed that the physical constraints (in Equation (7)) remain true always. We have not proved that those constraints would not be violated. In fact, let us consider a scenario in which the leading car starts at velocity of 15m/s (decelerating at -3m/s^2), and the rear car starts at velocity 30 m/s at a distance of 10m behind it. Since the velocity 30 is less than 32, we expect that the two cars will not collide. However, a simulation plot in Figure 2 (left) shows that the cars collide. This occurs because, even before the cars collide, the physical constraint ($a \geq -5$) is violated. In the simulation shown in Figure 2 (right), the physical constraints are not violated and there is no collision. So, what we have actually proved is the following fact.

PROPOSITION 1. *If the rear car follows the control law in Equation (2) and the dynamics of the leading and rear car are given by Equation (1) and Equation (3), and the cars start in any initial configuration where $gap = 10$, $a = 0$, $0 \leq v \leq 32$, $0 \leq v_f$, then either one of the physical constraints in Equation (7) is violated or the cars never collide.*

3.2 1-D Robot: Synthesizing Safe Switching Logic

Consider a robot moving in a 1-dimensional space. Let x denote the position of the robot and v denote its velocity. Suppose we can control the robot by controlling the force we apply on it: we can either force the robot to have an acceleration of $+1$ units or -1 units. Thus, the robot has two modes with the following dynamics:

$$\texttt{Mode}_1 : \frac{dx}{dt} = v,\ \frac{dv}{dt} = 1 \quad \texttt{Mode}_2 : \frac{dx}{dt} = v,\ \frac{dv}{dt} = -1 \qquad (11)$$

We wish to switch between these two modes so that the robot remains within a specified range, say $70 \leq x \leq 80$. We assume that initially the robot is in a state such that $74 \leq x \leq 76$ and $v = 0$.

$$\begin{aligned} \phi_{\texttt{safe}} &:= 70 \leq x \wedge x \leq 80 \\ \phi_{\texttt{init}} &:= 74 \leq x \wedge x \leq 76 \wedge v = 0 \end{aligned} \qquad (12)$$

It is clear that we have to switch to $\texttt{Mode}_1$ before x reaches 70 and we have to switch to $\texttt{Mode}_2$ before x reaches 80. A recent paper [27] proposed a certificate-based method to solve this "switching logic synthesis problem" by deciding formulas in the theory of reals. Specifically, quantifier elimination is used to find a *controlled inductive invariant*, which is any set, say defined by $V_1 \geq 0 \wedge V_2 \geq 0$, that satisfies the following formula:

$$\begin{aligned} (\phi_{\texttt{init}} &\Rightarrow V_1 \geq 0 \wedge V_2 \geq 0) \ \wedge \\ (V_1 \geq 0 \wedge V_2 \geq 0 &\Rightarrow \phi_{\texttt{safe}}) \ \wedge \\ (V_1 = 0 \wedge V_2 \geq 0 &\Rightarrow (\frac{dV_1}{dt}\Big|_{\texttt{Mode}_1} \geq 0 \vee \frac{dV_1}{dt}\Big|_{\texttt{Mode}_2} \geq 0)) \\ (V_2 = 0 \wedge V_1 \geq 0 &\Rightarrow (\frac{dV_2}{dt}\Big|_{\texttt{Mode}_1} \geq 0 \vee \frac{dV_2}{dt}\Big|_{\texttt{Mode}_2} \geq 0)) \end{aligned} \qquad (13)$$

where $dV_1/dt|_{\texttt{Mode}_1}$ denotes the derivative of V_1 in $\texttt{Mode}_1$.

Suppose that we are given the following forms for V_1, V_2:

$$\begin{aligned} V_1 &:= v^2 + c_2 v + c_3 x + c_4 \\ V_2 &:= -v^2 - c_2 v - c_3 x + c_5 \end{aligned} \tag{14}$$

where c_2, c_3, c_4, c_5 are some (unknown) constants. For these choices of V_1 and V_2, we can symbolically compute the derivatives dV_1/dt and dV_2/dt in the two modes using the dynamics in Equation (11).

We can determine if there exist constants $c_2, \ldots, c_5$ that will make Formula (13) true by deciding the formula

$$\exists c_2, c_3, c_4, c_5 : \forall x, v : \phi \tag{15}$$

where ϕ is the formula in Equation (13) with subexpressions replaced by their definitions given in Equation (12), Equation (14), and $dV_i/dt|_{\texttt{Mode}_j}$ replaced by the symbolic value of dV_i/dt in $\texttt{Mode}_j$ (File thermo-two-fail.red [32]).

Analysis Results

Using virtual substitution we can eliminate the inner universally quantified variables and get an equivalent constraint on the unknowns $c_2 \ldots, c_5$. If we existentially quantify these variables and ask `qepcad` if the formula is satisfiable, it fails. However, `slfq` is able to simplify the formula to `false` – in 1.03 seconds and using 604 `qepcad` calls – indicating that there is no controlled inductive invariant of this form.

We retry using a different template for V_1 and V_2:

$$\begin{aligned} V_1 &:= c_1 x - v^2 + c_2 p + c_3 \\ V_2 &:= -c_1 x - v^2 + c_2 p + c_4 \end{aligned} \tag{16}$$

Using these new forms for V_1 and V_2 we get a new $\exists\forall$ formula (File thermo-two-orig.red [32]). In this case, we succeed: `redlog` successfully eliminates the universal variables quickly, but the resulting constraint on c_1, c_2, c_3, c_4 is not easy to check for satisfiability. Even `slfq` fails to simplify the constraints. But looking at the constraints and using one of the disjuncts, $c_1 = 2$, as an assumption, `slfq` successfully simplifies the formula in 120 milliseconds using 296 `qepcad` calls, to

$$\begin{aligned} c_2 = 0 \ \wedge\ c_4 \geq 76c_1 \quad &\wedge \quad 4c_4 + c_2^2 - 320c_1 \leq 0 \ \wedge \\ c_3 + 74c_1 \geq 0 \quad &\wedge \quad 4c_3 + c_2^2 + 280c_1 \leq 0 \end{aligned}$$

Thus, we find that $c_1 = 2$, $c_2 = 0$, $c_4 = 156$, $c_3 = -144$ is a possible solution and, hence, we get the inductive controlled invariant,

$$2x - v^2 - 144 \geq 0 \ \wedge\ -2x - v^2 + 156 \geq 0$$

A simulation of the system that maintains this invariant is shown in Figure 3 (left).

We finally see if we can do better using more general quadratic templates. Let us use the following template

$$V_1 := -v^2 - c_1(x - 75)^2 + c_2, \tag{17}$$

which is motivated by the desire to bound the absolute values of v and $x - 75$. Using this template, we can expand the formula in Equation (13) to get another quantifier elimination problem (File thermo-two-quad.red [32]). This new problem is considerably simpler as `redlog` immediately returns after successfully eliminating the universal variables v and x. The condition on a, b is simplified by `slfq` in 60 milliseconds (using 37 `qepcad` calls) to

$$c_1 > 0 \wedge c_2 \leq 25c_1 \wedge c_2 \geq c_1 \wedge c_1 c_2 \leq 1$$

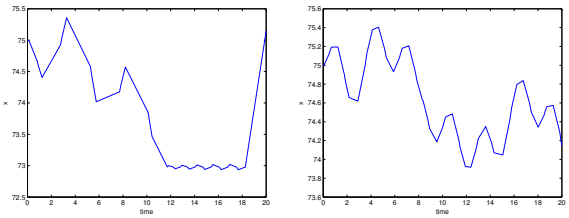

Figure 3: Simulation of the synthesized robot controller. (Left) Plot of x using the controller synthesized from a linear-quadratic controlled invariant. (Right) Plot of x using the controller synthesized from the quadratic controlled invariant. In both cases, the controller switches modes as soon as the boundary of the invariant is reached, which ensures that x remains in the safe interval $[70, 80]$.

Thus, by choosing $c_2 = 1$ and $c_1 = 1/25$, we get the following controlled inductive invariant

$$-v^2 - (x - 75)^2/25 + 1 \geq 0$$

A simulation of the controller generated using this controlled inductive invariant is shown in Figure 3 (right).

Using an ostensibly more "complex" template here leads to a "simpler" QE problem. One reason why the complex template is better is that it leads to a more "robust" proof – even if the model of the system were to change slightly (the coefficients were to be perturbed), there would still be invariants of the complex quadratic form, whereas there may not be an invariant of the simpler linear-quadratic form. While quantifier elimination is a hard problem, it could still be successful in verifying systems that are "well-designed" since such systems are more likely to be robust, and hence have small (easy) proofs [15].

3.3 Adaptive Flight Control: Proving Stability

We consider the problem of verifying the stability of a closed loop model of an adaptive flight control system.

The roll, pitch, and yaw rates of an airplane are controlled by setting the angles on the respective actuators – ailerons, elevators, and rudder. The task of the flight controller is to set the actuators so that the airplane reaches some desired roll, pitch, and yaw state. A typical proportional-integral (PI) controller is used for this purpose. However, recently there is renewed interest in evaluating *adaptive* controllers because of their potential to successfully control a damaged or aging aircraft.

Let $\texttt{w}_e$ denote the error between the desired state and the current state of the airplane. In general, $\texttt{w}_e$ (and other state variables that we will consider below) will be a vector (consisting of the error in roll, pitch, yaw); however, for simplicity (and also since the proof generalizes from a scalar to a vector), we let $\texttt{w}_e$ be a scalar. Let $\texttt{intw}_e$ denote the integral of $\texttt{w}_e$. The state variables $\texttt{w}_e$ and $\texttt{intw}_e$ are required to describe the standard PI controller. The adaptation term is modeled using three additional variables: β, L, and L^*, where β are the kernel functions, L are the weights learned by the adaptation procedure, and L^* are the optimal weights. Again we will assume that these are scalars in

our analysis here. The dynamics of the system are given by

$$\begin{aligned}\frac{d\mathtt{intw}_e}{dt} &= \mathtt{w}_e \\ \frac{d\mathtt{w}_e}{dt} &= -10\mathtt{intw}_e - 5\mathtt{w}_e + (L - L^*)\beta \\ \frac{dL}{dt} &= (-1000\mathtt{intw}_e - 2200\mathtt{w}_e)\beta \\ \frac{dL^*}{dt} &= 0 \end{aligned} \quad (18)$$

The dynamics of L describe the (neural) learning law used to update the current model of the plant damage. For further details on the actual model and the simplifications used to obtain the model above, we refer the reader to [31].

We are interested in showing that the error $\mathtt{w}_e$ eventually falls below a certain threshold. We prove this by showing that whenever the error is greater than that threshold, a certain positive semidefinite function will decrease. We assume that we are given a template for the positive semidefinite function, namely,

$$V := \mathtt{w}_e{}^2 + b\mathtt{intw}_e{}^2 + c(L - L^*)^2 \quad (19)$$

where b, c are unknown constants that we need to find. We cannot prove that the error $\mathtt{w}_e$ will eventually be bounded without making further assumptions. Specifically, we first assume that the absolute value of $\mathtt{intw}_e$ is bounded and also assume that the absolute value of the term $(L - L^*)\beta$ is bounded. Under these assumptions, the formula ϕ defined below says that V always decreases whenever $\mathtt{w}_e{}^2 \geq 1$:

$$\begin{aligned}\phi := \; & (\mathtt{intw}_e{}^2 \leq 1 \wedge (L - L^*)^2\beta^2 \leq 1 \wedge \mathtt{w}_e{}^2 \geq 1) \\ & \Rightarrow \frac{dV}{dt} < 0 \end{aligned} \quad (20)$$

where dV/dt can be computed using the definition of V and the dynamics given above as

$$\begin{aligned}\frac{dV}{dt} := \; & 2\mathtt{w}_e(-5\mathtt{w}_e - 10\mathtt{intw}_e + (L - L^*)\beta) + 2b\mathtt{intw}_e\mathtt{w}_e + \\ & 2c(L - L^*)(-1000\mathtt{intw}_e - 2200\mathtt{w}_e)\beta \end{aligned}$$

Thus, we get the quantified formula $\exists b, c : b > 0 \wedge c > 0 \wedge \forall \mathtt{w}_e, \mathtt{intw}_e, L, L^*, \beta : \phi$ in File adaptive-simpl.red [32].

Using virtual subtitution of `redlog`, we successfully eliminate all the universal quantifiers except $\mathtt{w}_e$ to get an equivalent formula on $b, c, \mathtt{w}_e$. This formula is a conjunction of 48 subformulas and contains 1081 atomic formulas, nested to a depth of 10. We simplify it using `slfq` under the assumption that $b > 0 \wedge c > 0$ and get the following simplified formula after 27.45 seconds of system time and 1897 `qepcad` calls:

$$b - 14 < 3200c < 16 - b \wedge 6 - b < 1200c < b - 4$$

The variable $\mathtt{w}_e$ is eliminated automatically. Thus, for example, we can choose $b = 10$ and $c = 1/600$. This proves that, under the assumptions made, the error $\mathtt{w}_e$ is always eventually bounded.

The choice of bounds on the assumptions may seem arbitrary. In fact, we can formulate the problem by replacing the concrete values for the bounds by symbolic constants and get the following formula.

$$\begin{aligned}\phi := \; & (\mathtt{intw}_e{}^2 \leq e \wedge (L - L^*)^2\beta^2 \leq d \wedge \mathtt{w}_e{}^2 \geq a) \\ & \Rightarrow \frac{dV}{dt} < 0 \end{aligned} \quad (21)$$

The $\exists\forall$ formula obtained from quantifying the above formula is contained in File adaptive-hard.red [32]. Eliminating quantifiers from this formula proceeds as before, and `redlog` successfully eliminates all universal quantifiers except $\mathtt{w}_e$. However, `slfq` is unable to simplify the resulting formula (in 30 minutes of real time).

Finally consider the case when only the kernel functions used by the learning module (β) are known to be bounded. Under this assumption, we wish to search for a nonincreasing positive semidefinite function of the form,

$$V := a\mathtt{intw}_e{}^2 + (\mathtt{w}_e + b\mathtt{intw}_e)^2 + c(L - L^*)^2 \quad (22)$$

where a, b, c are unknown constants to be determined such that the resulting function V is nonincreasing. This is stated in the following formula, under the assumption that β remains bounded.

$$\phi := (\beta^2 \leq 1 \Rightarrow \frac{dV}{dt} \leq 0); \quad (23)$$

where the derivative of V is easily calculated using the definition of V in Equation (22) and the dynamics of the system defined in Equation (18). Thus, we get the quantified formula $\forall \mathtt{w}_e, \mathtt{intw}_e, \beta, L, L^* : \phi$ in File adaptive-final.red [32]. Virtual substitution is successful in eliminating all the quantified variables. This quantifier-free formula is simplified by `slfq` in 60 milliseconds and using 29 `qepcad` calls to

$$\begin{aligned}11b = 5 \wedge 1000c = b \wedge \\ (a + 2b^2 - 10b - 20)a + (b^2 - 10b + 45)b^2 \leq 100(b - 1)\end{aligned}$$

which suggests $b = 5/11, c = 1/2200, a = 5$ as one possible solution. (In fact, a can be any number in the interval $[3, 21]$). This proves bounded stability of the adaptive PI controller assuming that the kernel functions remain bounded.

3.4 Inverted Pendulum: Synthesizing Stable Controller

A classic problem in control pertains to maintaining an inverted pendulum around its unstable equilibrium by controlling the force on the cart on which the pendulum is mounted. The state of the inverted pendulum can be described using four continuous variables, the position x of the cart, the velocity v of the cart, the angular deviation θ of the pendulum from its unstable equilibrium point $\theta = 0$, and the angular velocity ω of the pendulum. The dynamics of the inverted pendulum are obtained by balancing forces and can be rewritten in state space form as

$$\begin{aligned}\frac{dx}{dt} &= v \\ \frac{dv}{dt} &= \frac{(F - ml\omega^2\sin(\theta) + mg\cos(\theta)\sin(\theta))}{(M + m - m\cos(\theta)\cos(\theta))} \\ \frac{d\theta}{dt} &= \omega \\ \frac{d\omega}{dt} &= (g\sin(\theta) + \cos(\theta)\frac{dv}{dt})/l \end{aligned} \quad (24)$$

where $g = 10$ is the acceleration due to gravity, $m = 0.5$ is the mass of the pendulum, $M = 0.5$ is the mass of the cart, $l = 0.3$ is the length of the pendulum, and F is the force on the cart.

Since we cannot perform quantifier elimination on formulas containing trigonometric functions, and since we know that θ remains close to 0, we approximate the trigonometric

functions by the first few terms of their Taylor expansions. We assume that we have three modes available to us to control the pendulum depending on the force F we apply: we can choose $F = 2$, or $F = -2$, or $F = 0$. After substituting the values for the parameters and Taylor approximations of the trigonometric functions, we get the following equations for the dynamics of θ and ω.

$$\begin{aligned} \frac{d\theta}{dt} &= \omega \qquad (25) \\ \frac{d\omega}{dt} &= 50\theta - \omega^2\theta/2 - 100\theta^3/3 + 3\omega^2\theta^3/4 \\ &\quad + F(10/3 - 5\theta^2 + 5\theta^4/3) \end{aligned}$$

where $F \in \{+2, 0, -2\}$ in the three modes.

We wish to keep the pendulum inside a safe region, namely,

$$\texttt{safe} \;:=\; (20\theta^2 \leq 1)$$

The goal is to design a controller that will take the system to the region `safe` and keep it there. The controller takes the form of logical conditions for switching between the three modes. Assume that initially the pendulum is in a state that satisfies the constraint $-1 \leq 20\theta \leq 1 \wedge \omega = 0$.

We solve the problem by synthesizing an inductive controlled invariant for the problem. We assume that the user specifies the following template for searching for a controlled invariant $V \geq 0$ where

$$V \;:=\; -\theta^2 - b\omega^2 + c$$

Thus, we need to find b, c such that $V \geq 0$ becomes an inductive controlled invariant. This happens when the formula ϕ becomes valid:

$$\begin{aligned} \phi &:= \phi_1 \wedge \phi_2 \wedge \phi_3 \\ \phi_1 &:= ((-1 \leq 20\theta \leq 1 \wedge \omega = 0) \Rightarrow \\ &\qquad (-\theta^2 - \omega^2 b + c \geq 0)); \\ \phi_2 &:= (-\theta^2 - \omega^2 b + c \geq 0 \Rightarrow 1 \geq 20\theta^2); \\ \phi_3 &:= (-\theta^2 - \omega^2 b + c = 0 \Rightarrow \\ &\qquad (\frac{dV}{dt}\Big|_{\texttt{Mode}_1} \geq 0 \vee \frac{dV}{dt}\Big|_{\texttt{Mode}_2} \geq 0 \vee \frac{dV}{dt}\Big|_{\texttt{Mode}_3} \geq 0)) \end{aligned}$$

where dV/dt in the three modes is easily computed symbolically. This way we get the universally quantified formula $\forall\theta, \omega : \phi$ with free variables b, c (File inverted-pend.red [32]).

We use virtual substitution of `redlog` to eliminate the inner universal variables. It successfully eliminates θ, but it fails to eliminate ω. Neither `qepcad` succeeds in reasonable time (5 minutes user time) to eliminate ω, nor `slfq` is successful in simplifying the formula. However, the formula produced by `redlog` has a conjunct $c \geq 1/400$. We pick an arbitrary value, $1/100$, for c that is greater than $1/400$. Now the quantifier elimination problem is much simpler (File inverted-pend-easy.red [32]) and after virtual substitution eliminates θ, `qepcad` is able to eliminate ω in 190 milliseconds to give the constraint $4801b - 300 \geq 0$ on b. Thus, we get the following controlled inductive invariant for the inverted pendulum system:

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

Using this controlled invariant, we can now synthesize an algorithm for switching between the three modes of the system so that the system remains inside the set `safe`. A sample simulation of the synthesized system is shown in Figure 4.

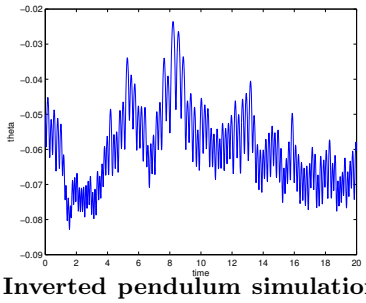

Figure 4: Inverted pendulum simulation. Plot of θ when a mode switching controller is synthesized using the generated controlled invariant. By switching mode as soon as the boundary of the controlled invariant is reached, we guarantee that θ remains inside the safe set.

4. RELATED WORK AND CONCLUSION

Quantifier elimination has been applied earlier to solve nonlinear control system design [20]. However, this early work focused only on continuous (and not switched) systems, and moreover only on simple properties. Properties like safety and stability were not considered – partly because they cannot be captured as semialgebraic sets in a sound and complete way. We give up completeness and generate an $\exists\forall$ formula that is only sufficient for the verification or synthesis problem. Consequently, whenever the $\exists\forall$ formula is valid, we successfully verify the system (or synthesize a correct system), but when the formula is not valid, it does not mean the system is faulty (or unsynthesizable).

The classical way of proving stability by finding Lyapunov functions is an instance of certificate-based verification. In this classical approach, the search for a Lyapunov function of a particular form is reduced to solving an $\exists\forall\phi$ formula, where ϕ is an *atomic* fact. Numerical techniques in the form of semidefinite programming exist for solving such problems [22, 5]. The work on barrier certificates [23] moves this overall approach from stability to safety.

There is plenty of work on certificate-based verification of hybrid systems [30, 26, 18], but none of it has used CAD or substitute-and-simplify methods and instead relied on approximate methods to eliminate quantifiers. In a recent paper [29], we used `qepcad` to solve $\exists\forall$ formulas arising from certificate-based synthesis. This paper builds upon [29] by proposing certificate-based analysis as a uniform approach for verification and synthesis, presenting several different benchmarks, and solving them using a combination of symbolic tools for quantifier elimination and simplification. Recently, Anai [1] used a combination of numerical methods (sum-of-squares) and symbolic quantifier elimination methods to solve problems arising in control, and such an integration is left for future work here.

It might be noteworthy that there has been considerable theoretical research on real quantifier elimination beyond the methods discussed here [17, 25, 3]. Most unfortunately, this appears not to have led to practically applicable software so far. An interesting step towards making alternative approaches practically useful has been made in [19] recently.

In summary, the *certificate-based approach* for verification and synthesis is a promising technique for formal analysis of complex cyber-physical systems. Its success is, however, crucially dependent on real quantifier elimination methods.

Certificate-based approach reduces verification and synthesis problems for continuous and switched systems to first-order formulas over the reals. An automatic combination of various software components for quantifier elimination and simplification was used to successfully process the first-order formulas. Our case studies described here provide an interesting set of benchmarks for real quantifier elimination.

Acknowledgments

We would like to thank Andreas Weber for encouraging and supporting the first author in visiting SRI.

5. REFERENCES

[1] H. Anai. A symbolic-numeric approach to nonlinear dynamical system analysis, 2010. SIAM/MSRI workshop on hybrid method. for symb.-numeric comp.

[2] D. S. Arnon, G. E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. Computing*, 13(4):865–877, 1984.

[3] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. of the ACM*, 43(6):1002–1045, 1996.

[4] California PATH: Partners for advanced transit and highways. `http://www.path.berkeley.edu/`.

[5] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[6] C. W. Brown. QEPCAD B: a program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37(4):97–108, 2003.

[7] C. W. Brown and C. Gross. Efficient preprocessing methods for quantifier elimination. In *CASC*, volume 4194 of *LNCS*, pages 89–100. Springer-Verlag, 2006.

[8] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition – preliminary report. *ACM SIGSAM Bulletin*, 8(3):80–90, Aug. 1974.

[9] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symbolic Computation*, 12(3):299–328, Sept. 1991.

[10] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. of Symbolic Computation*, 5(1–2):29–35, Feb.–Apr. 1988.

[11] W. Decker et al. SINGULAR 3-1-2 — A computer algebra system for polynomial computations, 2010. `http://www.singular.uni-kl.de`.

[12] A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[13] A. Dolzmann and T. Sturm. Simplification of quantifier-free formulae over ordered fields. *J. of Symbolic Computation*, 24(2):209–231, Aug. 1997.

[14] A. Dolzmann, T. Sturm, and V. Weispfenning. A new approach for automatic theorem proving in real geometry. *J. Automated Reasoning*, 21(3), 1998.

[15] D. Gayme, M. Fazel, and J. C. Doyle. Complexity in automation of SOS proofs: An illustrative example. In *45th IEEE Conf. on Decision and Control*, 2006.

[16] D. Godbole and J. Lygeros. Longitudinal control of the lead car of a platoon. *IEEE Transactions on Vehicular Technology*, 43(4):1125–35, 1994.

[17] D. Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1-2):65–108, 1988.

[18] S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In *Proc. 20th CAV*, volume 5123 of *LNCS*, pages 190–203. Springer, 2008.

[19] H. Hong and M. Safey El Din. Variant real quantifier elimination: algorithm and application. In *ISSAC*, pages 183–190. ACM, 2009.

[20] M. Jirstrand. Nonlinear control system design by quantifier elimination. *J. Symb. Comput.*, 24(2):137–152, 1997.

[21] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computations for families of linear vector fields. *J. Symbolic Computation*, 32(3):231–253, 2001.

[22] P. A. Parrilo. SOS methods for semi-algebraic games and optimization. In *HSCC 2005*, volume 3414 of *LNCS*, page 54. Springer, 2005.

[23] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Trans. on Automatic Control*, 52(8):1415–1428, 2007.

[24] A. Puri and P. Varaiya. Driving safely in smart cars. In *Proc. 1995 American Control Conference*, 1995.

[25] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13(3):255–352, Mar. 1992.

[26] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. In *HSCC*, volume 2993 of *LNCS*, pages 539–554. Springer, 2004.

[27] A. Taly, S. Gulwani, and A. Tiwari. Synthesizing switching logic using constraint solving. In *VMCAI*, volume 5403 of *LNCS*, pages 305–319. Springer, 2009.

[28] A. Taly and A. Tiwari. Deductive verification of continuous dynamical systems. In *FSTTCS*, volume 4 of *LIPIcs*, pages 383–394, 2009.

[29] A. Taly and A. Tiwari. Switching logic synthesis for reachability. In *EMSOFT*, 2010.

[30] A. Tiwari. Approximate reachability for linear systems. In *Proc. 6th HSCC*, volume 2623 of *LNCS*, pages 514–525. Springer, 2003.

[31] A. Tiwari. Bounded verification of adaptive flight control systems. In *Proc. AIAA Infotech@Aerospace*, 2010. AIAA-2010-3362.

[32] A. Tiwari. Certificate-based verification: Tools and benchmarks, 2011. `http://www.csl.sri.com/~tiwari/existsforall/`.

[33] A. Tiwari and G. Khanna. Series of abstractions for hybrid automata. In *HSCC*, volume 2289 of *LNCS*, pages 465–478. Springer, 2002.

[34] C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proc. of the IEEE*, 91(7), 2003.

[35] V. Weispfenning. The complexity of linear problems in fields. *J. of Symbolic Computation*, 5(1&2):3–27, 1988.

[36] V. Weispfenning. Quantifier elimination for real algebra—the cubic case. In *Proc. ISSAC*, pages 258–263. ACM Press, New York, 1994.

[37] V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Alg. in Eng. Comm. Comp.*, 8(2):85–101, 1997.