Safety Verification of Nonlinear Hybrid Systems Based on Invariant Clusters

Hui Kong
IST Austria
Klosterneuburg, Austria
hui.kong@ist.ac.at

Sergiy Bogomolov
Australian National University
Canberra, Australia
sergiy.bogomolov@anu.edu.au

Christian Schilling
University of Freiburg
Freiburg, Germany
schillic@informatik.uni-freiburg.de

Yu Jiang
Tsinghua University, Beijing
Henan University, Kaifeng
China
jy1989@mail.tsinghua.edu.cn

Thomas A. Henzinger
IST Austria
Klosterneuburg, Austria
tah@ist.ac.at

ABSTRACT

In this paper, we propose an approach to automatically compute invariant clusters for nonlinear semialgebraic hybrid systems. An invariant cluster for an ordinary differential equation (ODE) is a multivariate polynomial invariant $g(\vec{u},\vec{x})=0$, parametric in $\vec{u}$, which can yield an infinite number of concrete invariants by assigning different values to $\vec{u}$ so that every trajectory of the system can be overapproximated precisely by the intersection of a group of concrete invariants. For semialgebraic systems, which involve ODEs with multivariate polynomial right-hand sides, given a template multivariate polynomial $g(\vec{u},\vec{x})$, an invariant cluster can be obtained by first computing the remainder of the Lie derivative of $g(\vec{u},\vec{x})$ divided by $g(\vec{u},\vec{x})$ and then solving the system of polynomial equations obtained from the coefficients of the remainder. Based on invariant clusters and sum-of-squares (SOS) programming, we present a new method for the safety verification of hybrid systems. Experiments on nonlinear benchmark systems from biology and control theory show that our approach is efficient.

CCS Concepts

•General and reference → Verification; •Theory of computation → Timed and hybrid models; •Software and its engineering → Formal methods; Model checking;

Keywords

hybrid system; nonlinear system; semialgebraic system; invariant; safety verification; SOS programming

HSCC'17, April 18 - 20, 2017, Pittsburgh, PA, USA

ISBN 978-1-4503-4590-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3049797.3054966

1. INTRODUCTION

A hybrid system [17] is a dynamical system that exhibits both discrete and continuous behaviors. In this paper, we consider the safety verification problem for hybrid systems. In other words, we want to automatically check whether a set of bad states can be reached from a set of initial states. For systems described by nonlinear differential equations this task is particularly complicated as computing the exact reachable set is usually infeasible. Existing approaches are mainly based on approximate reachable set computation [2, 9, 4, 6] and abstraction [33, 1, 11, 13, 5, 8, 7].

An invariant is a special kind of overapproximation for the reachable set of a system. Since invariants do not involve direct computation of the reachable set, they are especially suitable for dealing with nonlinear hybrid systems. However, automatically and efficiently generating sufficiently strong invariants is challenging [26, 12, 28, 24, 20, 27, 16, 29, 18].

In this work, we propose an approach to automatically compute *invariant clusters* for a class of nonlinear semialgebraic systems whose trajectories are algebraic, i.e., every trajectory of the system is essentially a subset of the intersection of zero level sets of a group of multivariate polynomials. An invariant cluster for a semialgebraic system is a parameterized multivariate polynomial invariant $g(\vec{u},\vec{x})=0$, with parameter $\vec{u}$, which can yield an infinite number of concrete invariants by assigning different values to $\vec{u}$ so that every trajectory of the system can be overapproximated precisely by the intersection of a group of concrete invariants.

We roughly describe the idea of computing invariant clusters. A sufficient condition for a trajectory of a semialgebraic system to start from and to always stay in the zero level set of a multivariate polynomial $g(\vec{x})$ (i.e., $\{\vec{x}\in\mathbb{R}^n \mid g(\vec{x})=0\}$) is that the Lie derivative $\mathcal{L}_{\vec{f}}g$ of $g(\vec{x})$ on the vector flow $\vec{f}$ can be divided exactly by $g(\vec{x})$ (i.e., the remainder of $\mathcal{L}_{\vec{f}}g$ w.r.t. $g(\vec{x})$ must be identical to 0). Therefore, if some $g(\vec{x})$ satisfies this condition, $g(\vec{x})=0$ is an invariant of the system. Given a template polynomial $g(\vec{u},\vec{x})$ with $\vec{u}$ as its coefficients, we can compute the remainder $r(\vec{u},\vec{x})$ of $\mathcal{L}_{\vec{f}}g$ w.r.t. $g(\vec{u},\vec{x})$ symbolically. Moreover, $r(\vec{u},\vec{x})\equiv 0$ implies that all the coefficients $a_i(\vec{u})$ of the monomials in $\vec{x}$ in $r(\vec{u},\vec{x})$ are equal to 0; thus we can set up a system P of polynomial equations in $\vec{u}$ from the coefficients $a_i(\vec{u})$. By solving P we get a set $\mathcal{C}$ of constraints on $\vec{u}$. For those elements in $\mathcal{C}$ that are linear

in $\vec{u}$, the corresponding parameterized polynomial equations $g(\vec{u},\vec{x})=0$ form an infinite set of invariants, which we call invariant clusters. Based on invariant clusters and sum-of-squares (SOS) programming, we propose a new method for the safety verification of hybrid systems.

The main contributions of this paper are as follows: 1) We propose to generate invariant clusters for nonlinear semialgebraic systems based on computing the remainder of the Lie derivative of a template polynomial $g(\vec{u},\vec{x})$ w.r.t. $g(\vec{u},\vec{x})$ and solving the system of polynomial equations obtained from the coefficients of the remainder. 2) We present a method to overapproximate trajectories precisely by using invariant clusters. 3) We apply invariant clusters to the safety verification of semialgebraic hybrid systems based on SOS programming. 4) We implemented a prototype tool to perform the aforementioned steps automatically. Experiments show that our approach is effective and efficient.

The paper is organized as follows. Section 2 is devoted to the preliminaries. In Section 3, we introduce the approach to computing invariant clusters and using them to characterize trajectories. In Section 4, we present a method to verify safety properties for semialgebraic continuous and hybrid systems based on invariant clusters. In Section 5, we present our experimental results. In Section 6, we introduce some related works. Finally, we conclude our paper in Section 7.

2. PRELIMINARIES

In this section, we recall some concepts used throughout the paper. We first clarify some notation conventions. If not specified otherwise, we decorate vectors $\vec{\cdot}$, we use the symbol $\mathbb{K}$ for a field, $\mathbb{R}$ for the real number field and $\mathbb{N}$ for the set of natural numbers, and we consider multivariate polynomials, e.g., elements of the ring $\mathbb{K}[\vec{x}]$, where the components of $\vec{x}$ act as indeterminates. In addition, for all the polynomials $g(\vec{u},\vec{x})$, we denote by $\vec{u}$ the vector composed of all the u_i and denote by $\vec{x}$ the vector composed of all the remaining variables that occur in the polynomial.

Definition 1 (Ideal). *[10] A subset I of $\mathbb{K}[\vec{x}]$, is called an **ideal** if 1) $0\in I$; 2) if $p,q\in I$, then $p+q\in I$; and 3) if $p\in I$ and $q\in\mathbb{K}[\vec{x}]$, then $pq\in I$.*

Definition 2 (Generated ideal). *[10] Let $g_1,\ldots,g_s$ be polynomials in $\mathbb{K}[\vec{x}]$. The **ideal generated by** $\{g_1,\ldots,g_s\}$ is*

$$\langle g_1,\ldots,g_s\rangle \stackrel{def}{=} \left\{\sum_{i=1}^{s} h_i g_i \mid h_1,\ldots,h_s\in\mathbb{K}[\vec{x}]\right\}.$$

Definition 3 (Algebraic variety). *Let $\mathbb{K}$ be an algebraically closed field and $I\subset\mathbb{K}[\vec{x}]$ be an ideal. We define the **algebraic variety** of I as*

$$\mathbb{V}(I)\stackrel{def}{=}\{\vec{x}\in\mathbb{K}^n \mid f(\vec{x})=0 \text{ for } f\in I\}.$$

Next, we present the notation of the Lie derivative, which is widely used in the discipline of differential geometry. Let $\vec{f}:\mathbb{R}^n\to\mathbb{R}^n$ be a continuous vector field such that $\dot{x}_i=f_i(\vec{x})$ where $\dot{x}_i$ is the time derivative of $x_i(t)$.

Definition 4 (Lie derivative). *For a given polynomial $p\in\mathbb{K}[\vec{x}]$ over $\vec{x}=(x_1,\ldots,x_n)$ and a continuous system $\dot{\vec{x}}=\vec{f}$, where $\vec{f}=(f_1,\ldots,f_n)$, the **Lie derivative** of $p\in\mathbb{K}[\vec{x}]$ along f of order k is defined as follows.*

$$\mathcal{L}_{\vec{f}}^{k}p\stackrel{def}{=}\begin{cases} p, & k=0\\ \sum_{i=1}^{n}\frac{\partial\mathcal{L}_{\vec{f}}^{k-1}p}{\partial x_i}\cdot f_i, & k\geq 1\end{cases}$$

Essentially, the k-th order Lie derivative of p is the k-th derivative of p w.r.t. time, i.e., reflects the change of p over time. We write $\mathcal{L}_{\vec{f}}p$ for $\mathcal{L}_{\vec{f}}^{1}p$.

We also use the following theorem for deciding the existence of a real solution of a system of polynomial constraints.

Theorem 1 (Real Nullstellensatz). *[32] The system of multivariate polynomial equations and inequalities $p_1(\vec{x})=0,\ldots,p_{m_1}(\vec{x})=0,\quad q_1(\vec{x})\geq 0,\ldots,q_{m_2}(\vec{x})\geq 0,$ $r_1(\vec{x})>0,\ldots,r_{m_3}(\vec{x})>0$ either has a solution in $\mathbb{R}^n$, or there exists the following polynomial identity*

$$\sum_{i=1}^{m_1}\beta_i p_i+\sum_{v\in\{0,1\}^{m_2}}\delta_v\prod_{j=1}^{m_2}q_j^{v_j}+\prod_{k=1}^{m_3}r_k^{d_k}+\sum_{v\in\{0,1\}^{m_3}}\eta_v\prod_{k=1}^{m_3}r_k^{v_k}+s=0 \tag{1}$$

where $d_k\in\mathbb{N}$ and p_i,q_j,r_k,β_i are polynomials and δ_v,η_v,s are SOS (sum-of-squares) polynomials in $\mathbb{R}[\vec{x}]$.

Remark 1. *Theorem 1 enables efficient decision if a system of polynomial constraints has a real solution. By moving the term s in equation (1) to the right-hand side and denoting the remaining terms by $R(\vec{x},\vec{y})$, we have $-R(\vec{x},\vec{y})=s$, which means that $-R(\vec{x},\vec{y})$ is SOS. Hence finding a set of polynomials $\beta_j,r_k,\delta_v,\eta_v,s$ and some d_k's that make $-R(\vec{x},\vec{y})$ SOS is sufficient to prove that the system has no real solution, which can be done efficiently by SOS programming [25].*

In this paper, we focus on semialgebraic continuous and hybrid systems; we define them in the following.

Definition 5 (Semialgebraic system). *A **semialgebraic system** is a triple $M\stackrel{def}{=}\langle X,\vec{f},X_0\rangle$, where*

1. *$X\subseteq\mathbb{R}^n$ is the state space of the system M,*
2. *$\vec{f}\in\mathbb{R}[\vec{x}]^n$ is a locally Lipschitz continuous polynomial vector field function, and*
3. *$X_0\subseteq X$ is the initial set, which is semialgebraic [32].*

The local Lipschitz continuity guarantees the existence and uniqueness of the differential equation $\dot{\vec{x}}=\vec{f}$ locally. A trajectory of a semialgebraic system is defined as follows.

Definition 6 (Trajectory). *Given a semialgebraic system M, a **trajectory** originating from a point $\vec{x}_0\in X_0$ to time $T>0$ is a continuous and differentiable function $\vec{x}(t):[0,T)\to\mathbb{R}^n$ such that 1) $\vec{x}(0)=\vec{x}_0$, and 2) $\forall\tau\in[0,T)$: $\frac{d\vec{x}}{dt}\big|_{t=\tau}=\vec{f}(\vec{x}(\tau))$. T is assumed to be within the maximal interval of existence of the solution from $\vec{x}_0$.*

Definition 7 (Safety). *Given an unsafe set $X_U\subseteq X$, a semialgebraic system $M=\langle X,\vec{f},X_0\rangle$ is said to be **safe** if no trajectory $\vec{x}(t)$ of M satisfies both $\vec{x}(0)\in X_0$ and $\exists\tau\in\mathbb{R}_{\geq 0}:\vec{x}(\tau)\in X_U$.*

Definition 8 (Hybrid System). *A **hybrid system** is described by a tuple $\mathcal{H}\stackrel{def}{=}\langle L,X,E,G,R,I,F\rangle$, where*

- *L is a finite set of locations (or modes),*
- *$X\subseteq\mathbb{R}^n$ is the continuous state space. The hybrid state space of the system is denoted by $\mathcal{X}=L\times X$ and a state is denoted by $(l,\vec{x})\in\mathcal{X}$,*

- $E \subseteq L \times L$ *is a set of discrete transitions, together with a guard mapping* $G : E \to 2^X$ *and a reset mapping* $R : E \times X \to 2^X$,
- $I : L \to 2^X$ *is an invariant mapping, and*
- $F : L \times X \to \mathbb{R}^n$ *is a vector field mapping that assigns to each location* l *a vector field* $\vec{f}$.

The transition and dynamic structure of the hybrid system defines a set of trajectories. A trajectory is a sequence originating from a state $(l_0, \vec{x}_0) \in \mathcal{X}_0$, where $\mathcal{X}_0 \subseteq \mathcal{X}$ is an initial set, and consisting of a series of interleaved continuous flows and discrete transitions. During the continuous flows, the system evolves following the vector field $F(l)$ at some location $l \in L$ as long as the invariant condition $I(l)$ is not violated. At some state $(l, \vec{x})$, if there is a discrete transition $(l, l') \in E$ such that $(l, \vec{x}) \in G(l, l')$ (we write $G(l, l')$ for $G((l, l'))$), the discrete transition can be taken and the system state can be reset to $R((l, l'), \vec{x})$. The problem of safety verification of a hybrid system is to prove that an unsafe set X_U cannot be reached from an initial set X_0.

3. COMPUTING INVARIANT CLUSTERS

In this section, we introduce the *invariant cluster* and show how to compute a set of invariant clusters and use it to overapproximate all trajectories of a semialgebraic system.

3.1 Invariants and invariant clusters

Given a semialgebraic system M, if we can find a multivariate polynomial $g(\vec{x}) \in \mathbb{R}[\vec{x}]$ such that for any trajectory $\vec{x}(t)$ of M, $g(\vec{x}(0)) \sim 0$ implies $g(\vec{x}(t)) \sim 0$ for all $t > 0$, where $\sim \in \{<, \leq, =, \geq, >\}$, then $g(\vec{x}) \sim 0$ is an *invariant* of the system. We call $g(\vec{x})$ an *invariant polynomial* of M. A trajectory $\vec{x}(t)$ is said to be *algebraic* if there exists a nonzero polynomial invariant $g(\vec{x}) = 0$ for $\vec{x}(t)$. Next we present a sufficient condition for $g(\vec{x})$ to be an invariant polynomial.

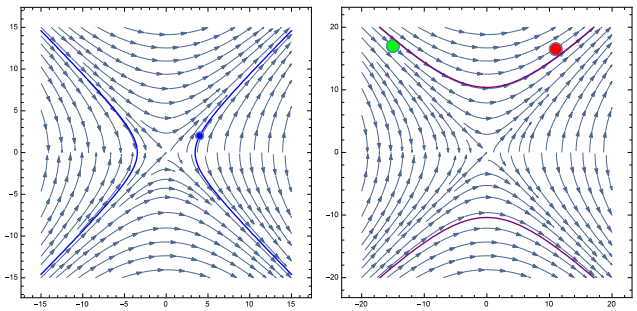
PROPOSITION 1. *Let* $M = \langle X, \vec{f}, X_0 \rangle$ *be a semialgebraic system and* $g(\vec{x}) \in \mathbb{R}[\vec{x}]$. *Then* $g \sim 0$ *is an invariant of* M *for every* $\sim \in \{<, \leq, =, \geq, >\}$ *if* $g(\vec{x})$ *satisfies*

$$\mathcal{L}_{\vec{f}} g \in \langle g \rangle \tag{2}$$

Proposition 1 states that all the polynomial equations and inequalities $g \sim 0$ are invariants of M if the Lie derivative of g belongs to the ideal $\langle g \rangle$. Note that every invariant satisfying condition (2) defines an enclosed region for trajectories, that is, no trajectory can enter or leave the region.

For a semialgebraic system whose trajectories are algebraic, the trajectories can usually be divided into several groups, and in each group all trajectories show similar curves. Essentially, these similar curves can be described identically by a unique parameterized polynomial equation that we characterize as an invariant cluster. The computation method of invariant clusters is presented in Subsection 3.2.

DEFINITION 9 (INVARIANT CLUSTER). *An* ***invariant cluster*** C *of a semialgebraic system is a set of invariants that can be uniformly described as* $C = \{g(\vec{u}, \vec{x}) = 0 \mid \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$, *where* $g(\vec{u}, \vec{x}) = \sum_{i=1}^{M} c_i(\vec{u}) X^i$ *satisfies* $\mathcal{L}_{\vec{f}} g \in \langle g \rangle$ *and* $c_i(\vec{u}) \in \mathbb{R}[\vec{u}]$ *are fixed linear polynomials in* $\vec{u} = (u_1, \cdots, u_K)$, X^i *are monomials in* $\vec{x} = (x_1, \cdots, x_n)$, *and* $M, K \in \mathbb{N}$.



(a) **(b)**

Figure 1: (a) Example 2. Curve defined by invariant cluster of Class$(C^*, \vec{x}_0)$ for $\vec{x}_0 = (4, 2)$. (b) Example 6. X_0: $(x+15)^2+(y-17)^2 \leq 1$, X_U: $(x_1-11)^2+(y_1-16.5)^2 \leq 1$.

Note that by requiring $\vec{u} \neq \vec{0}$ in Definition 9 as well as other related definitions, we exclude the trivial invariant $0 = 0$. Given an invariant cluster, by varying the parameter $\vec{u}$ we may obtain an infinite set of concrete invariants for the system. To be intuitive, we present a running example to demonstrate the related concepts throughout the paper.

EXAMPLE 1 (RUNNING EXAMPLE). *Consider the semialgebraic system* M_1 *described by* $[\dot{x}, \dot{y}] = [y^2, xy]$. *The set* $C^* = \{u_1 - u_3(x^2 - y^2) = 0 \mid (u_1, u_3) \in \mathbb{R}^2 \setminus \{\vec{0}\}\}$ *is an invariant cluster. It is easy to verify that the polynomial* $u_1 - u_3(x^2 - y^2)$ *satisfies condition* (2) *for all* $(u_1, u_3) \in \mathbb{R}^2$.

DEFINITION 10 (INVARIANT CLASS). *Given a semialgebraic system* M *with an initial point* $\vec{x}_0$ *and an invariant cluster* $C = \{g(\vec{u}, \vec{x}) = 0 \mid \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$ *of* M, *where* $K \in \mathbb{N}$, *an* ***invariant class*** *of* C *at* $\vec{x}_0$, *denoted by* **Class**$(C, \vec{x}_0)$, *is the set* $\{g(\vec{u}, \vec{x}) = 0 \mid g(\vec{u}, \vec{x}_0) = 0, \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$.

Given an invariant cluster C, by substituting a specific point $\vec{x}_0$ for $\vec{x}$ in $g(\vec{u}, \vec{x}) = 0$ we get a constraint $g(\vec{u}, \vec{x}_0) = 0$ on $\vec{u}$, which yields a subset **Class**$(C, \vec{x}_0)$ of C. Apparently, every member of **Class**$(C, \vec{x}_0)$ is an invariant for the trajectory originating from $\vec{x}_0$.

EXAMPLE 2 (RUNNING EXAMPLE). *For the given invariant cluster* C^* *in Example 1 and a given initial point* $\vec{x}_0 = (4, 2)$, *we get the invariant class* **Class**$(C^*, \vec{x}_0) = \{u_1 - u_3(x^2 - y^2) = 0 \mid u_1 - 12u_3 = 0, (u_1, u_3) \in \mathbb{R}^2 \setminus \{\vec{0}\}\}$. *Every member of* **Class**$(C^*, \vec{x}_0)$ *is an invariant for the trajectory of* M_1 *originating from* $\vec{x}_0$. *The algebraic variety defined by* **Class**$(C^*, \vec{x}_0)$ *is shown in Figure 1(a).*

An invariant class has the following important properties.

THEOREM 2. *Given an* n*-dimensional semialgebraic system* M *and an invariant class* $D = \{g(\vec{u}, \vec{x}) = 0 \mid g(\vec{u}, \vec{x}_0) = 0, \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$ *of* M *at a specified point* $\vec{x}_0$, *let* D_g *be the set of all invariant polynomials occurring in* D *and* $\pi_{\vec{x}_0}$ *be the trajectory of* M *originating from* $\vec{x}_0$. *Then,*

1. $\pi_{\vec{x}_0} \subseteq \mathbb{V}(D_g)$;
2. *there exists a subset* B *of* D_g *consisting of* m *members such that* $\langle D_g \rangle = \langle B \rangle$, *where* m *is the dimension of the hyperplane* $g(\vec{u}, \vec{x}_0) = 0$ *in terms of* $\vec{u}$ *in* $\mathbb{R}^K$ *and* $\langle D_g \rangle$ *and* $\langle B \rangle$ *denote the ideal generated by the members of* D_g *and* B, *respectively.*

Algorithm 1: Computation of invariant clusters

input : f: n-dimensional polynomial vector field;
N: upper bound for invariant polynomial degree
output: CFamily: a set of invariant clusters

```
1 CFamily ← ∅;
2 for i ← 1 to N do
3     g_{u⃗,x⃗} ← generate parameterized polynomial over x⃗ of
        degree i;
4     Lfg ← compute the Lie derivative of g_{u⃗,x⃗};
5     foreach monomial order O of x⃗ do
6         R_{u⃗,x⃗} ← compute remainder of Lfg w.r.t. g by O;
7         Coeffs ← collect coefficients of x⃗ in R_{u⃗,x⃗};
8         Solution ← solve system Coeffs on u⃗;
9         CFamily ← CFamily ∪ Solution;
```

REMARK 2. *The first property in Theorem 2 reveals that a trajectory $\pi_{\vec{x}_0}$ is always contained in the intersection of all the invariants in the invariant class D of $\vec{x}_0$. The second property asserts that the invariant class can be generated by a finite subset B of D if it consists of an infinite number of invariants. The algebraic variety $\mathbb{V}(B)$ (which is equivalent to $\mathbb{V}(D_g)$) forms an overapproximation for $\pi_{\vec{x}_0}$ and the quality of the overapproximation depends largely on the dimension m of $\mathbb{V}(B)$ (the lower the better). In the ideal case $m = 1$, $\mathbb{V}(B)$ shrinks to an algebraic curve and hence some part matches the trajectory precisely. In the case of $m > 1$, $\mathbb{V}(B)$ is usually a hypersurface. To make the overapproximation less conservative, we may take the union of multiple invariant classes from different invariant clusters (if they exist) to reduce the dimension of the algebraic variety.*

3.2 Invariant cluster computation

According to Proposition 1, if we can find a polynomial $g(\vec{x})$ such that $\mathcal{L}_{\vec{f}}g \in \langle g \rangle$, which is equivalent to that the remainder of $\mathcal{L}_{\vec{f}}g$ w.r.t. $g(\vec{x})$ is identical to 0, then $g(\vec{x}) = 0$ is an invariant of M. The idea is as follows. We first establish a template $g(\vec{u}, \vec{x})$ for $g(\vec{x})$ with parameter $\vec{u}$ and then compute the remainder $r(\vec{u}, \vec{x})$ of $\mathcal{L}_{\vec{f}}g$ w.r.t. $g(\vec{u}, \vec{x})$. According to the procedure of polynomial division [10], $r(\vec{u}, \vec{x})$ must be of the form $\sum_{i=1}^{K} \frac{b_i(\vec{u})}{u_j^d} X^i$, where $d \in \mathbb{N}$, $b_i(\vec{u})$ are homogeneous polynomials of degree $d+1$ over $\vec{u}$, u_j is the coefficient of the leading term of $g(\vec{u}, \vec{x})$ by some specified monomial order of $\vec{x}$, and X^i are monomials in $\vec{x}$. Since $r(\vec{u}, \vec{x}) \equiv 0$ implies $u_j \neq 0$ and $b_i(\vec{u}) = 0$ for all $i = 1, \ldots, K$, we obtain a system $\mathcal{C}$ of homogeneous polynomial equations in $\vec{u}$ plus $u_j \neq 0$ from the coefficients of $r(\vec{u}, \vec{x})$. Solving $\mathcal{C}$ may yield a set of invariant clusters of M if it exists. Note that all the aforementioned steps can be performed automatically in mathematical software such as *Maple*. Pseudocode for computing invariant clusters is shown in Algorithm 1. The motivation for the loop in line 5 is that the remainder may vary from the monomial order of $\vec{x}$ and produce different solutions. Using multiple orders helps to get more solutions.

REMARK 3. *In Algorithm 1, the key steps are computing the remainder in line 4 and solving the system of equations on $\vec{u}$ in line 8. The former takes only linear time and hence is very efficient. The latter involves solving a system of homogeneous polynomial equations, which is NP-complete [3]. In our implementation in* Maple*, we use the command* `solve`*, a sophisticated solver that combines a number of algorithms, including Gröbner basis and the elimination method based on resultants, and selects the best algorithm on the fly. In our experiments on nonlinear (parametric) systems of dimensions ranging from 2 to 8 the solver quickly determines whether a solution exists in most cases.*

Like most invariant generation approaches our approach is limited. Essentially, an invariant cluster could be an infinite set of Darboux polynomials and first integrals, which means that our approach applies only to integrable systems [14].

EXAMPLE 3 (RUNNING EXAMPLE). *According to Algorithm 1, the steps for computing the invariant clusters of degree 2 are as follows:*

1. *Generate the template polynomial of degree 2:*
$$g_2(\vec{u}, \vec{x}) = u_6 x^2 + u_5 xy + u_4 x + u_3 y^2 + u_2 y + u_1$$
2. *Compute the Lie derivative $\mathcal{L}_{\vec{f}}g_2$ using Definition 4:*
$$\begin{aligned}\mathcal{L}_{\vec{f}}g_2 &= \frac{\partial g_2}{\partial x}\dot{x} + \frac{\partial g_2}{\partial y}\dot{y} \\ &= u_5 x^2 y + (2\,u_3 + 2\,u_6)\,xy^2 + u_2 xy + y^3 u_5 + u_4 y^2\end{aligned}$$
3. *Compute the remainder of $\mathcal{L}_{\vec{f}}g_2$ w.r.t. g_2 by graded reverse lexicographic (*grevlex*) order of (x, y). Using this order, the leading term of $\mathcal{L}_{\vec{f}}g_2$ and g_2 is $u_5 x^2 y$ and $u_6 x^2$, respectively. Then:*
$$\begin{aligned}r(\vec{u}, \vec{x}) = \mathcal{L}_{\vec{f}}g_2 - \frac{u_5 y}{u_6} g_2 = \Big(&\left(2\,u_3 u_6 - u_5^2 + 2\,u_6^2\right) xy^2 \\ &+ \left(u_2 u_6 - u_4 u_5\right) xy + \left(-u_3 u_5 + u_5 u_6\right) y^3 \\ &+ \left(-u_2 u_5 + u_4 u_6\right) y^2 - u_1 u_5 y \Big) \frac{1}{u_6}\end{aligned}$$
4. *Collect the coefficients of $r(\vec{u}, \vec{x})$:*
$$S := \left\{ \frac{u_2 u_6 - u_4 u_5}{u_6}, \frac{-u_3 u_5 + u_5 u_6}{u_6}, \frac{-u_2 u_5 + u_4 u_6}{u_6}, \frac{2\,u_3 u_6 - u_5^2 + 2\,u_6^2}{u_6}, -\frac{u_1 u_5}{u_6} \right\}$$
5. *Solve the system formed by S. To save space, we just present one of the six solutions we obtained:*
$$C_6 = \{u_6 = -u_3, u_2 = u_4 = u_5 = 0, u_3 = u_3, u_1 = u_1\}$$
6. *Substitute the above solution C_6 for $\vec{u}$ in $g_2(\vec{u}, \vec{x})$. We get the following parameterized invariant polynomial:*
$$g_2(\vec{u}, \vec{x}) = -u_3 x^2 + u_3 y^2 + u_1$$

The other five solutions obtained in step 5. are in fact the products of the invariant polynomials $\{u_2 y, u_1(x+y), u_1(x-y)\}$ that were obtained when initially computing the invariants of degree 1. Hence they cannot increase the expressive power of the set of invariant clusters and should be dropped. The above solution is the one we have given in Example 1.

3.3 Overapproximating trajectories by invariant classes

In this section, we address how to overapproximate trajectories precisely by using invariant classes.

Algorithm 2: Computation of invariant classes

input : CFamily: set of invariant clusters;
$\vec{x}_0$: an initial point
output: ICls: list of invariant classes

```
ICls ← ∅;
foreach C ∈ CFamily do
    D ← Class(C, x⃗0);
    if D ≠ ∅ then
        m ← dimension of the hyperplane g(u⃗, x⃗0) = 0
          defining D;
        if m ≥ 1 then
            Basis ← basis {u1, ..., um} of g(u⃗, x⃗0) = 0;
            D ← {g(u⃗1, x⃗), ..., g(u⃗m, x⃗)}, ui ∈ Basis;
        ICls ← ICls ∪ D;
```

Invariant clusters can be divided into two categories according to the number of invariant classes that they can yield by varying the parameter $\vec{u}$. 1) **finite invariant cluster**: This kind of invariant cluster can yield only one invariant class no matter how $\vec{u}$ changes. For example, $\{u_1(x-y)=0 \mid u_1 \in \mathbb{R}\setminus\{0\}\}$ is such an invariant cluster for the running example. The trajectories covered by this invariant class are very limited. Moreover, the overapproximation is conservative due to the high dimension of the algebraic variety defined by the invariant class. 2) **infinite invariant cluster**: One such invariant cluster C can yield an infinite number of invariant classes $\mathbf{Class}(C,\vec{x}_0)$ as the initial point $\vec{x}_0$ varies, e.g., the invariant cluster C^* in Example 1. For the trajectory $\pi_{\vec{x}_0}$, the overapproximating precision of $\mathbf{Class}(C,\vec{x}_0)$ depends largely on the dimension m of the algebraic variety defined by $\mathbf{Class}(C,\vec{x}_0)$. In the best case $m=1$ we have a curve-to-curve match in part for the trajectory.

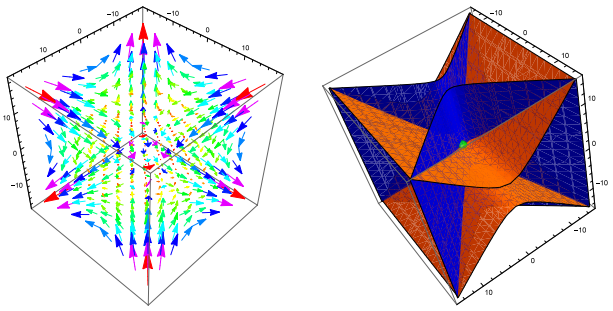
Next we introduce how to identify the invariant classes for a given point $\vec{x}_0$ from a set of invariant clusters and how to get a finite representation. To be intuitive, we first present a 3-dimensional system and a set of invariant clusters for it.

Example 4. *Consider the following semialgebraic system M_2: $[\dot{x},\dot{y},\dot{z}]=[yz,xz,xy]$. We obtain a set of invariant clusters consisting of 7 elements. Here we only present the infinite invariant cluster due to the page limit.*

$$C_7=\{g_7(\vec{u},\vec{x})=(-u_5-u_6)x^2+u_5y^2+u_6z^2+u_0 = 0 \mid \vec{u}\in\mathbb{R}^3\setminus\{\vec{0}\}\}$$

The invariant clusters are capable of overapproximating all the trajectories of the system M_2. For any given initial state, how can we identify the invariant classes from the set of invariant clusters? Suppose we want to find the invariant classes that can overapproximate the trajectory from the state $\vec{x}_0=(1,2,3)$. According to Theorem 2, we have Algorithm 2 for this purpose.

Remark 4. *In Algorithm 2, we enumerate the invariant clusters to find out which one can provide a non-empty invariant class* $\mathbf{Class}(C,\vec{x}_0)$ *for* $\vec{x}_0$. *For a* $\mathbf{Class}(C,\vec{x}_0)$ *to be nonempty, the corresponding hyperplane* $g(\vec{u},\vec{x}_0)=0$ *must have at least one solution to* $\vec{u}\in\mathbb{R}^K\setminus\{\vec{0}\}$, *which is equivalent to that its dimension must be at least* 1. *For a hyperplane in* $\mathbb{R}^K$, *its dimension is equal to* $K-1$. *Therefore,* $\mathbf{Class}(C,\vec{x}_0)$ *must be nonempty if* $K>1$ *and the basis of the hyperplane can be obtained through a basic linear algebraic computation (which will be illustrated in what follows). However, in case* $g(\vec{u},\vec{x}_0)$ *evaluates to* 0, *the hyperplane degenerates to the space* $\mathbb{R}^K\setminus\{\vec{0}\}$ *and the dimension will be* K. *Therefore, an invariant class with* $K=1$ *is nonempty iff* $g(\vec{u},\vec{x}_0)$ *evaluates to* 0. *For example, given an invariant cluster* $C^0=\{u_1(x-y)=0 \mid u_1\in\mathbb{R}\setminus\{0\}\}$ *and a point* $\vec{x}_0=(x_0,y_0)$, $\mathbf{Class}(C^0,\vec{x}_0)$ *is equal to* C^0 *if* $x_0=y_0$, *and otherwise* $\mathbf{Class}(C^0,\vec{x}_0)$ *is empty.*



(a) **(b)**

Figure 2: (a) 3D vector field of Example 4. (b) The intersection of the invariants $y^2-x^2-3=0$ (blue) and $z^2-x^2-8=0$ (orange) overapproximates the trajectory originating from $\vec{x}_0=(1,2,3)$ (green ball).

Example 5. *We continue from Example 4. For the given point* $\vec{x}_0=(1,2,3)$, *according to Algorithm 2, we find that only* $\mathbf{Class}(C_7,\vec{x}_0)=\{g_7(\vec{u},x,y,z)=0 \mid 3u_5+8u_6+u_0=0,\vec{u}\in\mathbb{R}^3\setminus\{\vec{0}\}\}$ *is nonempty. The dimension of the hyperplane* $H: 3u_5+8u_6+u_0=0$ *is* 2. *Since* $u_0=-3u_5-8u_6$, *to get the basis of* H, *we can write* $(u_0,u_5,u_6)=(-3u_5-8u_6,u_5,u_6)=u_5(-3,1,0)+u_6(-8,0,1)$. *Hence we have the basis* $\{(-3,1,0),(-8,0,1)\}$ *for* H. *As a result, we get a finite representation* $B=\{y^2-x^2-3=0,\ z^2-x^2-8=0\}$ *for* $\mathbf{Class}(C_7,\vec{x}_0)$. *It is easy to check by the* Maple *function* `HilbertDimension` *that* $dim(B)=1$. *Thus we obtain an algebraic variety* $\mathbb{V}(B)$ *that provides in part a curve-to-curve match to the trajectory* $\pi_{\vec{x}_0}$. *The 3-D vector field and the algebraic curve* $\mathbb{V}(B)$ *are shown in Figure 2.*

4. SAFETY VERIFICATION

4.1 Safety Verification of Continuous Systems

In this subsection, we show how to verify a safety property for a nonlinear system based on invariant clusters.

In Section 3, we have seen that a trajectory can be overapproximated by an invariant class. Since an invariant class is determined uniquely by a single hyperplane $g(\vec{u},\vec{x}_0)=0$ in $\mathbb{R}^K$ for an initial point $\vec{x}_0$, and a hyperplane without constant term (which holds for $g(\vec{u},\vec{x}_0)=0$) is uniquely determined by its normal vector, we can verify that two states do not lie on the same trajectory using the following theorem.

Theorem 3. *Given a semialgebraic system* $M=\langle X,\vec{f},X_0\rangle$ *and an invariant cluster* $C=\{g(\vec{u},\vec{x})=0 \mid \vec{u}=(u_1,\cdots,u_K)\in\mathbb{R}^K\setminus\{\vec{0}\}\}$ *of* M *with* $K>1$, *where* $g(\vec{u},\vec{x})=\sum_{i=1}^{K}\psi_i(\vec{x})u_i$ *and* $\psi_i(\vec{x})\in\mathbb{R}[\vec{x}]$, *an initial set* X_0, *and an unsafe set* X_U,

if there exists a pair of states $(\vec{x}_1, \vec{x}_2) \in X_0 \times X_U$ such that $\vec{x}_1$ and $\vec{x}_2$ lie on the same trajectory, one of the following two formulae must hold:

$$(i) \quad \exists k \in \mathbb{R}\backslash\{0\} : \ k\psi_i(\vec{x}_1) = \psi_i(\vec{x}_2), i = 1, \ldots, K \quad (3)$$

$$(ii) \quad \psi_i(\vec{x}_1) = \psi_i(\vec{x}_2) = 0, i = 1, \ldots, K \quad (4)$$

Moreover, if some $\psi_i(\vec{x}) \equiv 1$, i.e., $g(\vec{u}, \vec{x})$ contains a constant term u_i, then formula (3) simplifies to

$$\psi_i(\vec{x}_1) = \psi_i(\vec{x}_2), i = 1, \ldots, K \quad (5)$$

REMARK 5. *Instead of computing the invariants explicitly, Theorem 3 provides an alternative way to verify that two states $\vec{x}_1, \vec{x}_2$ do not lie on the same trajectory by checking the difference between the normal vectors of $g(\vec{u}, \vec{x}_1) = 0$ and $g(\vec{u}, \vec{x}_2) = 0$. Let us take Example 4 for illustration. We think of C_7 as a hyperplane over $\vec{u} \in \mathbb{R}^3$: $u_0 + (y^2 - x^2)u_5 + (z^2 - x^2)u_6 = 0$, hence the corresponding normal vector is $\vec{\mathcal{N}}(\vec{x}) = (1, y^2 - x^2, z^2 - x^2)$. Given two random points $\vec{x}_1 = (1, 2, 3)$ and $\vec{x}_2 = (5, \sqrt{27}, \sqrt{34})$, it is easy to verify that $\vec{\mathcal{N}}(\vec{x}_1) \neq \vec{\mathcal{N}}(\vec{x}_2)$, which means that $\vec{x}_1$ and $\vec{x}_2$ are not on the same trajectory. Alternatively, we can argue that the invariant class of $\vec{x}_1$ is $\{y^2 - x^2 - 3 = 0, z^2 - x^2 - 8 = 0\}$ and $\vec{x}_2$ does not belong to its solution set.*

Now we demonstrate how to verify a safety property of semialgebraic systems. Assume X_0 and X_U can be written as semialgebraic sets, i.e., $X_0 = \{\vec{x}_1 \in \mathbb{R}^n \mid p_{i_1}(\vec{x}_1) = 0, q_{j_1}(\vec{x}_1) \geq 0, r_{k_1}(\vec{x}_1) > 0, i_1 = 1, \ldots, l_1, j_1 = 1 \ldots m_1, k_1 = 1, \ldots, n_1\}$ and $X_U = \{\vec{x}_2 \in \mathbb{R}^n \mid p_{i_2}(\vec{x}_2) = 0, q_{j_2}(\vec{x}_2) \geq 0, r_{k_2}(\vec{x}_2) > 0, i_2 = l_1+1, \ldots, l_1+l_2, j_2 = m_1+1, \ldots, m_1+m_2, k_2 = n_1 + 1, \ldots, n_1 + n_2\}$. Then we have the following theorem for deciding the safety of a semialgebraic system.

THEOREM 4. *Given a semialgebraic system $M = \langle X, \vec{f}, X_0 \rangle$ and invariant cluster $C = \{g(\vec{u}, \vec{x}) = 0 \mid \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$ of M with $K \geq 2$, suppose the normal vector of the hyperplane $g(\vec{u}, \vec{x}) = 0$ over $\vec{u}$ is $(1, \psi_1(\vec{x}), \ldots, \psi_K(\vec{x}))$. Then the system M is safe if there exists the following polynomial identity*

$$\sum_{k=1}^{K} \gamma_k(\psi_k(\vec{x}_1) - \psi_k(\vec{x}_2)) + \sum_{i=1}^{l_1+l_2} \beta_i p_i + \sum_{v \in \{0,1\}^{m_1+m_2}} \delta_v \prod_{j=1}^{m_1+m_2} q_j^{v_j} + \sum_{v \in \{0,1\}^{n_1+n_2}} \eta_v \prod_{k=1}^{n_1+n_2} r_k^{v_k} + \prod_{k=1}^{n_1+n_2} r_k^{d_k} + s = 0 \quad (6)$$

where $d_k \in \mathbb{N}$ and β_i, γ_k are polynomials and δ_v, η_v, s are SOS polynomials in $\mathbb{R}[\vec{x}_1, \vec{x}_2]$.

REMARK 6. *Theorem 4 transforms the safety verification problem into a decision problem about the existence of a real solution of a system of polynomial equations and inequalities. As noted in Remark 1, this decision problem can be solved by SOS programming. Our implementation uses the efficient tool* SOSTOOLS *[25].*

In Theorem 4, we deal with a general semialgebraic system where the initial set and the unsafe set are represented by a set of polynomial equations and inequalities. However, if the system is described by much simpler set representations such as a single polynomial equation or inequality, the programming problem can be simplified correspondingly. If, e.g., both sets can be represented or overapproximated by a single polynomial equation $I(\vec{x}_1) = 0$ and $U(\vec{x}_2) = 0$, respectively, then the programming problem simplifies to (see [32])

$$\sum_{j=1}^{K} \alpha_j(\psi_j(\vec{x}_1) - \psi_j(\vec{x}_2)) + \beta I + \theta U - 1 \quad \text{is an } SOS \quad (7)$$

where $(\psi_1(\vec{x}), \ldots, \psi_K(\vec{x}))$ is the same as in Theorem 4 and $\alpha_j, \beta, \theta \in \mathbb{R}[\vec{x}_1, \vec{x}_2]$. Algorithm 3 summarizes safety verification based on the condition (7).

Algorithm 3: Safety verification

input : $\vec{\psi}$: the K-dimensional normal vector of an invariant cluster; $\mathrm{I}(\vec{x}_1)$: the initial set; $\mathrm{U}(\vec{x}_2)$: the unsafe set; N: the maximum degree of programming polynomials $\vec{\alpha}, \beta, \theta$

output : IsSafe: whether the system is safe

```
IsSafe ← False;
for i ← 1 to N do
    α⃗ ← generate a vector of polynomials of degree i;
    β ← generate a polynomial of degree i for I(x⃗1);
    θ ← generate a polynomial of degree i for U(x⃗2);
    P ← Σ_{j=1}^{K} α_j(ψ⃗_j(x⃗1) − ψ⃗_j(x⃗2)) + βI+θU −1;
    Solution ← perform SOS programming on P;
    if Solution is found then
        IsSafe ← True;
        break;
```

EXAMPLE 6 (RUNNING EXAMPLE 2). *Given the semialgebraic system M_3 by $[\dot{x}, \dot{y}] = [y^2, xy]$ and the initial set $X_0 = \{(x, y) \in \mathbb{R}^2 \mid I(x, y) = (x+15)^2 + (y-17)^2 - 1 \leq 0\}$, verify that the unsafe set $X_U = \{(x, y) \in \mathbb{R}^2 \mid U(x, y) = (x-11)^2 + (y-16.5)^2 - 1 \leq 0\}$ cannot be reached. The parameter space of $C^* = \{g(\vec{u}, \vec{x}) = u_1 - u_3(x^2 - y^2) = 0 \mid (u_1, u_3) \in \mathbb{R}^2 \setminus \{\vec{0}\}\}$ has dimension 1 and hence can provide an invariant class for every state in X_0 and X_U. The normal vector of the hyperplane $g(\vec{u}, \vec{x}) = 0$ is $(1, \psi_1(x, y)) = (1, y^2 - x^2)$. Let $\varphi(x_1, y_1, x_2, y_2) = \psi_1(x_1, y_1) - \psi_1(x_2, y_2)$. By Theorem 4 we only need to verify whether the following system of equations has no real solution.*

$$I(x_1, y_1) = (x_1 + 15)^2 + (y_1 - 17)^2 - 1 = 0$$
$$U(x_2, y_2) = (x_2 - 11)^2 + (y_2 - 16.5)^2 - 1 = 0$$
$$\varphi(x_1, y_1, x_2, y_2) = y_1^2 - x_1^2 - (y_2^2 - x_2^2) = 0$$

Note that we substitute (x_1, y_1), (x_2, y_2) for (x, y) in $I(x, y)$ and $U(x, y)$, respectively, to denote the different points in X_0 and X_U. To prove that the system is safe, we need to find $\alpha_i \in \mathbb{R}[x_1, y_1, x_2, y_2], i = 1, 2, 3$ such that $Prog = \alpha_1 I + \alpha_2 U + \alpha_3 \varphi - 1$ is SOS. We find three polynomials of degree 2 for α_i, respectively, hence the system is safe. Observe that the relative position of X_U is very close to the reachable set from X_0 (see Figure 1(b)). We failed to find a barrier certificate for this system using the methods in [20, 24].

In Theorem 4, we presented a sufficient condition for deciding if a semialgebraic system is safe. The theory originated from the fact that the system is safe if there is no invariant class intersecting both the initial and the unsafe set, which is equivalent to that the formula (6) holds. To

verify the latter, we need to find a set of witness polynomials by *SOS* programming. However, as the dimension of the system increases, the number of parametric polynomials involved increases correspondingly, which also leads to an increase in computational complexity. In what follows, we present a new method for safety verification that avoids this problem. The new method is based on Proposition 1, i.e., for any polynomial $g(\vec{x})$ satisfying $\mathcal{L}_{\vec{f}}g \in \langle g \rangle$, $g(x) \sim 0$ is an invariant for any $\sim \in \{<, \leq, =, \geq, >\}$.

PROPOSITION 2. *Given a semialgebraic system* $M = \langle X, \vec{f}, X_0 \rangle$ *and an invariant cluster* $C = \{g(\vec{u}, \vec{x}) = 0 \mid \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$ *of* M*, let* X_0 *and* X_U *be the initial set and the unsafe set, respectively. Then, the system is safe if there exists a* $\vec{u}^* \in \mathbb{R}^K \setminus \{\vec{0}\}$ *such that*

$$\forall \vec{x} \in X_0 : g(\vec{u}^*, \vec{x}) \geq 0 \tag{8}$$

$$\forall \vec{x} \in X_U : g(\vec{u}^*, \vec{x}) < 0 \tag{9}$$

According to Proposition 2, to verify the safety property, it suffices to find a $\vec{u}^* \in \mathbb{R}^K \setminus \{\vec{0}\}$ which satisfies the constraints (8) and (9). There are some constraint solving methods available, e.g, *SMT* solvers. However, the high complexity of *SMT* theories limits the applicability. In the following, we transform the above constraint-solving problem into an *SOS* programming problem, which can be solved efficiently. We write $\vec{P}(\vec{x}) \succeq \vec{0}$ to denote $p_i(\vec{x}) \geq 0, i = 1, \ldots, m$ for a polynomial vector $\vec{P}(\vec{x}) = (p_1(\vec{x}), \ldots, p_m(\vec{x}))$.

PROPOSITION 3. *Given a semialgebraic system* $M = \langle X, \vec{f}, X_0 \rangle$ *and an invariant cluster* $C = \{g(\vec{u}, \vec{x}) = 0 \mid \vec{u} \in \mathbb{R}^K \setminus \{\vec{0}\}\}$ *of* M *and a constant* $\epsilon \in \mathbb{R}_{>0}$*, let* $X_0 = \{\vec{x} \in \mathbb{R}^n \mid \vec{I} \succeq \vec{0}, \vec{I} \in \mathbb{R}[\vec{x}]^{m_1}\}$ *and* $X_U = \{\vec{x} \in \mathbb{R}^n \mid \vec{U} \succeq \vec{0}, \vec{U} \in \mathbb{R}[\vec{x}]^{m_2}\}$*. Then, the system is safe if there exist a* $\vec{u}^* \in \mathbb{R}^K \setminus \{\vec{0}\}$ *and two SOS polynomial vectors* $\vec{\mu}_1 \in \mathbb{R}[\vec{x}]^{m_1}$, $\vec{\mu}_2 \in \mathbb{R}[\vec{x}]^{m_2}$ *such that the following are SOS polynomials.*

$$g(\vec{u}^*, \vec{x}) - \vec{\mu}_1 \cdot \vec{I} \tag{10}$$

$$-g(\vec{u}^*, \vec{x}) - \vec{\mu}_2 \cdot \vec{U} - \epsilon \tag{11}$$

Similar to Theorem 4, Proposition 3 also reduces to an *SOS* programming problem. However, the ideas behind the theories are different. By Theorem 4 we attempt to prove no invariant class overapproximating the trajectory can intersect both X_0 and X_U, while by Proposition 3 we mean to find a hypersurface that can separate the reachable set from X_U. Apparently, there must exist no invariant class intersecting both X_0 and X_U if there exists such a hypersurface, but not vice versa. Hence the latter is more conservative than the former, but it is also more efficient in theory because it usually involves less unknown polynomials. For example, for an n-dimensional system with X_0 and X_U defined by a single polynomial inequality, respectively, we usually need $n + 1$ unknown polynomials for the former method, however, we need only 2 for the latter. We omit the algorithm based on Proposition 3, which is similar to Algorithm 3.

4.2 Safety Verification of Hybrid Systems

A hybrid system consists of a set of locations and a set of discrete transitions between locations. In general, different locations have different continuous dynamics and hence correspond to different invariant clusters. An invariant for the hybrid system can be synthesized from the set of invariant clusters of all locations. The idea is to pick a polynomial $g_l(\vec{u}_l^*, \vec{x})$ from the respective invariant cluster C_l for each location l such that $g_l(\vec{u}_l^*, \vec{x}) \geq 0$ is an invariant for the location l and all the invariants coupled together through the constraints at the discrete transitions form a hybrid invariant for the hybrid system.

PROPOSITION 4. *Given an* n*-dimensional hybrid system* $\mathcal{H} = \langle L, X, E, G, R, I, F \rangle$ *and a set of invariant clusters* $\{C_l, l = 1, \ldots, n\}$*, where* $C_l = \{g_l(\vec{u}_l, \vec{x}) = 0 \mid \vec{u}_l \in \mathbb{R}^{K_l} \setminus \{\vec{0}\}\}$ *with* $K_l > 1$ *is an invariant cluster for location* l*, the system is safe if there exists a set* $S_{\vec{u}} = \{\vec{u}_l^* \in \mathbb{R}^{K_l} \setminus \{\vec{0}\}, l = 1, \ldots, n\}$ *such that, for all* $l \in L$ *and* $(l, l') \in E$*, the following formulae hold:*

$$\forall \vec{x} \in \mathrm{Init}(l) : g_l(\vec{u}_l^*, x) \geq 0 \tag{12}$$

$$\begin{aligned} &\forall \vec{x} \in G(l, l'), \forall \vec{x}' \in R((l, l'), \vec{x}) : \\ &g_l(\vec{u}_l^*, \vec{x}) \geq 0 \implies g_{l'}(\vec{u}_{l'}^*, \vec{x}') \geq 0 \end{aligned} \tag{13}$$

$$\forall \vec{x} \in I(l) \cap \mathrm{Uns}(l) : g_l(\vec{u}_l^*, \vec{x}) < 0 \tag{14}$$

where $\mathrm{Init}(l)$ *and* $\mathrm{Uns}(l)$ *denote respectively the initial set and the unsafe set at location* l*.*

Similar to Proposition 2, we further transform the problem into an *SOS* programming problem. Consider a semialgebraic hybrid system $\mathcal{H} = \langle L, X, E, G, R, I, F \rangle$, where the mappings G, R, and I are defined in terms of polynomial inequalities as follows:

$$G : (l, l') \mapsto \{\vec{x} \in \mathbb{R}^n \mid \vec{G}_{ll'} \succeq 0, \vec{G}_{ll'} \in \mathbb{R}[\vec{x}]^{m_{ll'}}\}$$

$$R : ((l, l'), \vec{x}) \mapsto \{\vec{x} \in \mathbb{R}^n \mid \vec{R}_{ll'\vec{x}} \succeq 0, \vec{R}_{ll'\vec{x}} \in \mathbb{R}[\vec{x}]^{n_{ll'}}\}$$

$$I : l \mapsto \{\vec{x} \in \mathbb{R}^n \mid \vec{I}_l \succeq 0, \vec{I}_l \in \mathbb{R}[\vec{x}]^{p_l}\}$$

and the mappings of the initial and the unsafe set are defined as follows:

$$\mathrm{Init} : l \mapsto \{\vec{x} \in \mathbb{R}^n \mid \vec{\mathrm{Init}}_l \succeq 0, \vec{\mathrm{Init}}_l \in \mathbb{R}[\vec{x}]^{r_l}\}$$

$$\mathrm{Uns} : l \mapsto \{\vec{x} \in \mathbb{R}^n \mid \vec{\mathrm{Uns}}_l \succeq 0, \vec{\mathrm{Uns}}_l \in \mathbb{R}[\vec{x}]^{s_l}\}$$

where $m_{ll'}$, $n_{ll'}$, r_l, p_l and s_l are the dimensions of the polynomial vector spaces. Then we have the following proposition for safety verification of $\mathcal{H}$.

PROPOSITION 5. *Let the hybrid system* $\mathcal{H}$*, the initial set mapping* Init*, and the unsafe set mapping* Uns *be defined as above. Given a set of invariant clusters* $\{C_l, l = 1, \ldots, n\}$ *of* $\mathcal{H}$ *where* $C_l = \{g_l(\vec{u}_l, \vec{x}) = 0 \mid \vec{u}_l \in \mathbb{R}^{K_l} \setminus \{\vec{0}\}\}$ *with* $K_l > 1$ *is an invariant cluster for location* l*, a set* $S_\gamma = \{\gamma_{ll'} \in \mathbb{R}_{\geq 0}, (l, l') \in E\}$ *of constants, and a constant vector* $\vec{\epsilon} \in \mathbb{R}^n_{>0}$*, the system is safe if there exists a set* $S_u = \{\vec{u}_l^* \in \mathbb{R}^{K_l} \setminus \{\vec{0}\}, l = 1, \ldots, n\}$ *and five sets of SOS polynomial vectors* $\{\vec{\theta}_l \in \mathbb{R}[\vec{x}]^{s_l}, l \in L\}$, $\{\vec{\kappa}_{ll'} \in \mathbb{R}[\vec{x}]^{p_{ll'}}, (l, l') \in E\}$, $\{\vec{\sigma}_{ll'} \in \mathbb{R}[\vec{x}]^{q_{ll'}}, (l, l') \in E\}$, $\{\vec{\eta}_l \in \mathbb{R}[\vec{x}]^{t_l}, l \in L\}$, *and* $\{\vec{\nu}_l \in \mathbb{R}[\vec{x}]^{w_l}, l \in L\}$ *such that the following polynomials are SOS for all* $l \in L$ *and* $(l, l') \in E$*:*

$$g_l(\vec{u}_l^*, \vec{x}) - \vec{\theta}_l \cdot \vec{\mathrm{Init}}_l \tag{15}$$

$$g_{l'}(\vec{u}_{l'}^*, \vec{x}') - \gamma_{ll'} g_l(\vec{u}_l^*, \vec{x}) - \vec{\kappa}_{ll'} \cdot G_{ll'} - \vec{\sigma}_{ll'} \cdot R_{ll'\vec{x}} \tag{16}$$

$$-\vec{\nu}_l \cdot I_l - \vec{\eta}_l \cdot \vec{\mathrm{Uns}}_l - g_l(\vec{u}_l^*, \vec{x}) - \epsilon_l \tag{17}$$

The algorithm for computing invariants for semialgebraic hybrid systems based on Proposition 5 is very similar to Algorithm 3 for semialgebraic continuous systems except that it involves more *SOS* constraints on discrete transitions.

5. IMPLEMENTATION & EXPERIMENTS

Based on the approach presented in this paper, we implemented a prototype tool in *Maple* and *Matlab*, respectively. In *Maple*, we implemented the tool for computing invariant clusters and identifying invariant classes based on remainder computation. In *Matlab*, we implemented the tool for safety verification based on the *SOS* programming tool package *SOSTOOLS*. Currently, we manually transfer the invariant clusters computed in *Maple* to *Matlab* for safety verification.

Now we present the experimental results on nonlinear benchmark systems, run on a laptop with a 3.1GHz *Intel Core i7* CPU and 8 GB memory.

5.1 Longitudinal Motion of an Airplane

In this experiment, we study the 6th order longitudinal equations of motion that capture the vertical motion (climbing, descending) of an airplane [31, Chapter 5]. Let g denote the gravity acceleration, m the total mass of an airplane, M the aerodynamic and thrust moment w.r.t. the y axis, (X, Z) the aerodynamics and thrust forces w.r.t. axis x and z, and I_{yy} the second diagonal element of its inertia matrix. Then the motion of the airplane is described as follows.

$$\begin{aligned}
\dot{v} &= \frac{X}{m} - g\sin(\theta) - qw, \qquad & \dot{w} &= \frac{Z}{m} + g\cos(\theta) + qv,\\
\dot{x} &= w\sin(\theta) + v\cos(\theta), & \dot{z} &= -v\sin(\theta) + w\cos(\theta),\\
\dot{\theta} &= q, & \dot{q} &= \frac{M}{I_{yy}},
\end{aligned}$$

where the meanings of the variables are as follows: v: axial velocity, w: vertical velocity, x: range, z: altitude, q: pitch rate, θ: pitch angle.

To transform the above system into a semialgebraic system, we first introduce two additional variables d_1, d_2 such that $d_1 = \sin(\theta)$, $d_2 = \cos(\theta)$ and then substitute d_1 and d_2 respectively for $\sin(\theta)$ and $\cos(\theta)$ in the model. In addition, we get two more constraints $\dot{d_1} = qd_2$ and $\dot{d_2} = -qd_1$. As a result, the dimension of the system rises to 8. For this system, using the method in [12], Ghorbal et al. spent **1** hour finding three invariant polynomials of degree 3 on a laptop with a 1.7GHz *Intel Core i5* CPU and 4 GB memory. Using our method, we spent only **0.484** seconds obtaining an invariant cluster $g_9(\vec{u}, \vec{x}) = 0$ of degree 3. By applying the constraint $d_1^2 + d_2^2 = 1$, we reduce the normal vector of the hyperplane $g_9(\vec{u}, \vec{x}) = 0$ in $\vec{u}$ to $(1, \psi_1, \psi_2, \psi_3)$, where ψ_1, ψ_2, ψ_3 are defined as follows.

$$\begin{aligned}
\psi_1 &= \frac{Mmz}{I_{yy}Z} + \frac{gm\theta}{Z} + \left(\frac{mqv}{Z} + 1\right)\sin(\theta)\\
&\quad + \left(\frac{X}{Z} - \frac{mqw}{Z}\right)\cos(\theta)\\
\psi_2 &= -\frac{Xz}{Z} + x - \frac{gI_{yy}X\theta}{ZM} - I_{yy}\left(\frac{Xqv}{ZM} + \frac{qw}{M}\right)\sin(\theta)\\
&\quad + I_{yy}\left(\frac{Xqw}{ZM} - \frac{qv}{M} - \frac{X^2+Z^2}{ZMm}\right)\cos(\theta)\\
\psi_3 &= q^2 - 2\frac{M\theta}{I_{yy}}
\end{aligned}$$

Given a symbolic initial point $\vec{x}_0 = (v_0, w_0, x_0, z_0, \theta_0, q_0, d_1^0, d_2^0)$, we have verified that our invariant cluster defines the same algebraic variety as defined by the invariants in [12] by comparing their Gröbner bases. However, our method is much more efficient. Moreover, we also obtained the invariant clusters of higher degrees $(4-6)$ quickly. The experimental result is shown in Table 1. The first column is the degree of the invariants, the second column is the variables to be decided, the third column is the computing time in seconds, and the last column is the number of invariant clusters generated. As can be seen, in the most complicated case, where the number of the indeterminates reaches up to 3003, we spent only 200.9 seconds to discover an invariant cluster of degree 6. However, we found that these higher order invariant clusters have the same expressive power as the invariant cluster of degree 3 in terms of algebraic variety.

5.2 Looping particle

Consider a heavy particle on a circular path of radius r whose motion is described by the following differential equation

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}\dot{r\cos(\theta)}\\ \dot{r\sin(\theta)}\\ -\frac{g\cos(\theta)}{r}\end{bmatrix} = \begin{bmatrix}-r\sin(\theta)\dot{\theta}\\ r\cos(\theta)\dot{\theta}\\ -\frac{g(r\cos(\theta))}{r^2}\end{bmatrix} = \begin{bmatrix}-y\omega\\ x\omega\\ -\frac{gx}{r^2}\end{bmatrix}$$

Note that the above is a parameterized system with gravity acceleration g and radius r as parameters. Our tool finds the following invariant cluster consisting of a parametric polynomial of degree 2: $\{g(\vec{u}, \vec{x}) = 0 \mid g(\vec{u}, \vec{x}) = u_5x^2 + u_5y^2 + u_2\omega^2 + \frac{2u_2g}{r^2}y + u_0, \vec{u} \in \mathbb{R}^3 \setminus \{\vec{0}\}\}$. Given an arbitrary point $(x_0, y_0, \omega_0) = (2, 0, \omega_0)$, we get the invariant class $\{g(\vec{u}, \vec{x}) = 0 \mid (x_0^2 + y_0^2)u_5 + (\omega_0^2 + \frac{2g}{r^2}y_0)u_2 + u_0 = 0, \vec{u} \in \mathbb{R}^3 \setminus \{\vec{0}\}\}$. According to Algorithm 2, the algebraic variety representing the trajectory originating from (x_0, y_0, ω_0) is $\{(x, y, \omega) \in \mathbb{R}^3 \mid x^2 + y^2 - x_0^2 - y_0^2 = 0, \omega^2 + \frac{2g}{r^2}y - \omega_0^2 - \frac{2g}{r^2}y_0 = 0\}$. The results in [26] and [29] are special cases of our result when setting $(r, g) = (2, 10)$ and $(r, g, x_0, y_0) = (2, 10, 2, 0)$, respectively. Therefore, our method is more powerful in finding parameterized invariants for parameterized systems. See Table 1 for detailed experimental results.

5.3 Coupled spring-mass system

Consider a system with two springs of weights w_1, w_2.

$$\begin{bmatrix}\dot{x_1}\\ \dot{v_1}\\ \dot{x_2}\\ \dot{v_2}\end{bmatrix} = \begin{bmatrix}v_1\\ -\frac{k_1}{m_1}x_1 - \frac{k_2}{m_1}(x_1 - x_2)\\ v_2\\ -\frac{k_2}{m_2}(x_2 - x_1)\end{bmatrix}$$

One spring, having spring constant k_1, is attached to the ceiling, and the weight w_1 of mass m_1 is attached to the lower end of this spring. Attached to weight w_1 is a second spring with spring constant k_2, and the weight w_2 of mass m_2 is attached to the lower end of this spring. x_1 and x_2 denote the displacements of the center of masses of the weights w_1 and w_2 from equilibrium, respectively.

In this experiment, we first consider an instantiated version of the system by using the same parameters as in [28]: $\frac{k_1}{m_1} = \frac{k_2}{m_2} = k$ and $m_1 = 5m_2$. The experimental result is presented in Table 1. We found that the expressive power of the invariant clusters does not increases any more as the degree of the invariant clusters is greater than 3 and it took only 0.25 seconds to compute the invariant cluster of degree 3. Finally, we perform the computation directly on the fully parameterized system and we get the following parameterized invariant cluster that enables us to analyze the

Table 1: Benchmark results for the Longitudinal Motion of an Airplane (B1), the Looping Particle system (B2), and the Coupled Spring-Mass system (B3).

Degree of invariants	No. of variables			Running time (sec)			No. of invariant clusters		
	B1	B2	B3	B1	B2	B3	B1	B2	B3
1	9	4	6	0.016	0.015	0.047	0	0	0
2	45	10	21	0.031	0.047	0.078	1	1	0
3	165	20	56	0.484	0.049	0.250	1	0	1
4	495	35	126	3.844	0.156	1.109	1	1	1
5	1287	56	252	25.172	0.703	6.641	1	0	1
6	3003	84	462	200.903	3.000	32.109	1	1	1

system properties under different parameter settings.

$$\begin{aligned} g(\vec{u},\vec{x}) = {} & u_8 v_1 v_2 + \frac{k_2 x_1 x_2 (m_1 u_8 - 2m_2 u_{10})}{m_1 m_2} + u_{10} v_1^2 + u_1 \\ & + \frac{1}{2}\frac{v_2^2 (k_1 m_2 u_8 - k_2 m_1 u_8 + k_2 m_2 u_8 + 2k_2 m_2 u_{10})}{k_2 m_1} \\ & + \frac{1}{2}\frac{(2k_1 m_2 u_{10} - k_2 m_1 u_8 + 2k_2 m_2 u_{10}) x_1^2}{m_1 m_2} \\ & + \frac{1}{2}\frac{(k_1 m_2 u_8 - k_2 m_1 u_8 + 2k_2 m_2 u_{10}) x_2^2}{m_1 m_2} \end{aligned}$$

5.4 Hybrid controller

Consider a hybrid controller consisting of two control modes. The discrete transition diagram of the system is shown in Figure 3(a) and the vector fields describing the continuous behaviors are given as follows:

$$f_1(\vec{x}) = \begin{bmatrix} y^2 + 10y + 25 \\ 2xy + 10x - 40y - 200 \end{bmatrix},$$
$$f_2(\vec{x}) = \begin{bmatrix} -y^2 - 10y - 25 \\ 8xy + 40x - 160y - 800 \end{bmatrix}$$

The system starts from some point in $X_0 = \{(x,y) \in \mathbb{R}^2 \mid (x-9)^2+(y-20)^2 \leq 4\}$ and then evolves following the vector field $f_1(\vec{x})$ at location l_1 (Switch-On). The value of x keeps increasing until it reaches 35. Then the system switches to location l_2 (Switch-Off) without performing any reset operation. At location l_2, the system operates following the vector field $f_2(\vec{x})$ and the value of x keeps decreasing. As the value of x drops to 5, the system switches immediately back to location l_1 again. Our objective is to verify that the value of y will never exceed 48 in both locations.

For the convenience of *SOS* programming, we define the unsafe set as $\text{Uns}(l_1) = \text{Uns}(l_2) = \{(x,y) \in \mathbb{R}^2 \mid 48 < y < 60\}$, which is sufficient to prove $y \leq 48$ in locations l_1 and l_2. According to the theory proposed in Section 4.2, we first find an invariant cluster for each location, which consists of a parameterized polynomial, respectively: $g_1(\vec{u}_1,\vec{x}) = -\frac{1}{5}u_{12}x^2 + \frac{1}{10}u_{12}y^2 + 8u_{12}x + u_{12}y + u_{11}$ and $g_2(\vec{u}_2,\vec{x}) = \frac{4}{5}u_{22}x^2 + \frac{1}{10}u_{22}y^2 - 32u_{22}x + u_{22}y + u_{21}$. In the second phase, we make use of the constraint condition in Proposition 5 to compute a pair of vectors $\vec{u}_1^*$ and $\vec{u}_2^*$. By setting $\gamma_{12} = \gamma_{21} = 1$, our tool found a pair of $\vec{u}_1^* = (u_{11}, u_{12}) = (2.9747, 382.14)$ and $\vec{u}_2^* = (u_{21}, u_{22}) = (2.9747, 138.44)$. As shown in Figure 3(b), the curves of $g_1(\vec{u}_1^*,\vec{x}) = 0$ and $g_2(\vec{u}_2^*,\vec{x}) = 0$ form an upper bound for the reachable set in location l_1 and l_2, respectively, which lie below the unsafe region $y \geq 48$. Therefore, the system is safe.

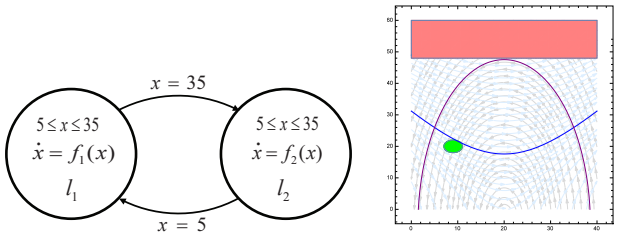


Figure 3: Hybrid controller from Subsection 5.4. (a) Hybrid automaton. (b) Hybrid invariant. Solid patch in green: initial set. Curve in blue: invariant for l_1. Curve in purple: invariant for l_2. Red shadow region on the top: unsafe region.

6. RELATED WORK

Many recent efforts have been made toward generating invariants for hybrid systems. Matringe et al. reduce the invariant generation problem to the computation of the associated eigenspaces by encoding the invariant constraints as symbolic matrices [26]. Ghorbal et al. use the invariant algebraic set formed by a polynomial and a finite set of its successive Lie derivatives to overapproximate vector flows [12]. Both of the aforementioned methods involve minimizing the rank of a symbolic matrix. Although in theory the problem of minimizing the rank of a symbolic matrix lies in the same complexity class as that of our problem, experiments show that our approach is more powerful in practice. Sankaranarayanan discovers invariants based on invariant ideal and pseudo ideal iteration [28], but this method is limited to algebraic systems. Moreover, none of the aforementioned methods involve verifying safety properties based on the invariants obtained. Tiwari et al. compute invariants for some special types of linear and nonlinear systems based on *Syzygy* computation and Gröbner basis theory as well as linear constraint solving [34]. Platzer et al. use quantifier elimination to find differential invariants [23]. Another approach considers barrier certificates based on different inductive conditions [24, 20, 21] which can be solved by *SOS* programming efficiently but is limited by the conservative inductive condition. Carbonell et al. generate invariants for linear systems [27]. Some other approaches focusing on different features of systems have also been proposed for constructing inductive invariants [19, 16, 29, 30, 22, 15].

7. CONCLUSION

In this paper, we proposed an approach to automatically generate invariant clusters for semialgebraic hybrid systems. Invariant clusters can overapproximate trajectories of the

system precisely. They can be obtained efficiently by computing the remainder of the Lie derivative of a template polynomial $g(\vec{u},\vec{x})$ w.r.t. $g(\vec{u},\vec{x})$ and then solving a system of homogeneous polynomial equations obtained from the remainder. Based on invariant clusters and *SOS* programming, we proposed a new method for safety verification of hybrid systems. Experiments show that our approach is efficient for a large class of biological and control systems.

Acknowledgment

This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), and by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems).

8. REFERENCES

[1] R. Alur, T. Dang, and F. Ivančić. Progress on reachability analysis of hybrid systems using predicate abstraction. In *HSCC*. 2003.

[2] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *HSCC*. 2003.

[3] A. Ayad. A survey on the complexity of solving algebraic systems. In *International Mathematical Forum*, volume 5, 2010.

[4] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. Guided search for hybrid systems based on coarse-grained space abstractions. *STTT*, 2015.

[5] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump. Assume-guarantee abstraction refinement meets hybrid systems. In *HVC*, 2014.

[6] S. Bogomolov, G. Frehse, R. Grosu, H. Ladan, A. Podelski, and M. Wehrle. A box-based distance between regions for guiding the reachability analysis of SpaceEx. In *CAV*, 2012.

[7] S. Bogomolov, C. Herrera, M. Muñiz, B. Westphal, and A. Podelski. Quasi-dependent variables in hybrid automata. In *HSCC*, 2014.

[8] S. Bogomolov, C. Schilling, E. Bartocci, G. Batt, H. Kong, and R. Grosu. Abstraction-based parameter synthesis for multiaffine systems. In *HVC*, 2015.

[9] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *RTSS*, 2012.

[10] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2007.

[11] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *HSCC*, 2010.

[12] K. Ghorbal and A. Platzer. Characterizing algebraic invariants by differential radical invariants. In *TACAS*. 2014.

[13] A. Girard and S. Martin. Synthesis for constrained nonlinear systems using hybridization and robust controllers on simplices. *IEEE Trans. Automat. Contr.*, 57(4), 2012.

[14] A. Goriely. *Integrability and nonintegrability of dynamical systems*, volume 19. World Scientific, 2001.

[15] E. Goubault, J. Jourdan, S. Putot, and S. Sankaranarayanan. Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials. In *ACC*, 2014.

[16] S. Gulwani and A. Tiwari. Constraint-based approach for analysis of hybrid systems. In *CAV*, 2008.

[17] T. A. Henzinger. The theory of hybrid automata. In *LICS*, 1996.

[18] Y. Jiang, H. Liu, H. Kong, R. Wang, M. Hosseini, J. Sun, and L. Sha. Use runtime verification to improve the quality of medical care practice. In *ICSE*, 2016.

[19] T. T. Johnson and S. Mitra. Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems. In *AIAA Infotech at Aerospace Conference*, 2013.

[20] H. Kong, F. He, X. Song, W. N. Hung, and M. Gu. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In *CAV*, 2013.

[21] H. Kong, X. Song, D. Han, M. Gu, and J. Sun. A new barrier certificate for safety verification of hybrid systems. *The Computer Journal*, 57(7):1033–1045, 2014.

[22] J. Liu, N. Zhan, and H. Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT*, 2011.

[23] A. Platzer and E. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In *CAV*, 2008.

[24] S. Prajna and A. Jadbabaie. Safety verification of hybrid systems using barrier certificates. *HSCC*, 2004.

[25] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS and its control applications*. 2005.

[26] R. Rebiha, A. V. Moura, and N. Matringe. Generating invariants for non-linear hybrid systems. *TCS*, 594, 2015.

[27] E. Rodríguez-Carbonell and A. Tiwari. Generating polynomial invariants for hybrid systems. *HSCC*, 2005.

[28] S. Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. In *HSCC*, 2010.

[29] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constructing invariants for hybrid systems. *HSCC*, 2004.

[30] B. Sassi, M. Amin, A. Girard, and S. Sankaranarayanan. Iterative computation of polyhedral invariants sets for polynomial dynamical systems. In *CDC*, 2014.

[31] R. F. Stengel. Flight dynamics. *Fluid Dynamics*, 1, 2004.

[32] G. Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2), 1974.

[33] A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1), 2008.

[34] A. Tiwari and G. Khanna. Nonlinear systems: Approximating reach sets. *HSCC*, 2004.