ISCAS-SKLCS-14-03 **February, 2014**

中 国 科 学 院 软 件 研 究 所
计 算 机 科 学 国 家 重 点 实 验 室
技 术 报 告

Abstraction of Elementary Hybrid Systems by Variable Transformation

by

Jiang Liu, Naijun Zhan, Hengjun Zhao
Liang Zou

State key Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
Beijing 100190. China

Abstraction of Elementary Hybrid Systems by Variable Transformation

Jiang Liu[1], Naijun Zhan[2], Hengjun Zhao[2], and Liang Zou[2]

[1] Chongqing Key Lab. of Automated Reasoning and Cognition, CIGIT, CAS
[2] State Key Lab. of Comput. Sci., Institute of Software, CAS

Abstract. Elementary hybrid systems (EHSs) are those hybrid systems (HSs) whose expressions may contain non-polynomial functions such as $\frac{1}{x}$, e^x, $\ln(x)$, $\sin(x)$ and $\cos(x)$, and their compositions. EHSs are very common in practice, and in particular, many of them are safety-critical. However, verification of EHSs is very hard, even intractable, because of those non-polynomial expressions, specifically, the transcendental expressions among them. Most of the existing approaches are based on explicit computation of approximate reachable sets using numeric computation, but these approaches suffer from the inflation of numerical errors, and with them only reachable sets within a bounded time interval can be approximated. In this paper, we first give a general solution to reducing verification of elementary dynamical systems (EDSs) to that of polynomial dynamical systems (PDSs). Then, we discuss how to extend the approach to EHSs. The benefit of our approach is to open a wider window to verification of EHSs, as all the well-established verification techniques for polynomial hybrid systems (PHSs) are applicable to EHSs after such abstraction. In this way, it is possible to avoid the inflation of numerical errors in the verification of EHSs. Particularly, verification of unbounded safety properties and stability analysis of EHSs becomes possible. Finally, we show how to apply our approach to some complicated real world examples.

Keywords: verification, elementary and transcendental functions, abstraction, variable transformation, elementary hybrid systems

1 Introduction

Complex Embedded Systems (CESs) consist of software and hardware components that operate autonomous devices interacting with the physical environment. They are now part of our daily life and are used in many industrial sectors including automotive, aerospace, consumer electronics, communications, medical, manufacturing and so on. CESs are used to carry out highly complex and often critical functions. They are used to monitor and control industrial plants, complex transportation equipment, communication infrastructure, etc. The development process of CESs is widely recognized as a highly complex and challenging task. A thorough validation and verification activity is necessary to enhance the quality of the CESs and, in particular, to fulfill the quality criteria mandated by the relevant standards. Hybrid systems (HSs) are mathematical models with precise mathematical semantics for CESs, wherein continuous physical

dynamics are combined with discrete transitions. Based on HSs, rigorous analysis and verification of CESs become feasible, so that errors can be detected and corrected in the very early stage of the design of CESs.

In practice, it is very often to model complex physical environments by ordinary differential equations (ODEs) with non-polynomial functions such as reciprocal function $\frac{1}{x}$, exponential function e^x, logarithm function $\ln(x)$, trigonometric functions $\sin(x)$ and $\cos(x)$, and their compositions. We call those HSs containing elementary functions elementary HSs (EHSs). Because of those non-polynomial expressions, in particular, the transcendental expressions among them, verification of EHSs becomes very hard, even intractable. Most of the existing approaches are based on explicit computation of the approximate reachable sets using numerical computation [4, 3, 5]. However these approaches suffer from the inflation of numerical errors, also called wrapping effects; besides, only reachable sets within a bounded time interval can be approximated. As an alternative, abstracting some specific subclasses of elementary dynamical and hybrid systems to polynomial ones [16–18] by variable transformation [9, 24] has been proposed, so that the resulted systems can be verified by well-established verification techniques for polynomial systems, which are based on either symbolic computation or numeric computation or their combination. Moreover, in [21] the author considered how to further abstract some polynomial dynamical and hybrid systems to linear ones for improving verification efficiency; while in [23, 15], the authors investigated how to abstract linear hybrid systems to infinite state transition systems for further improving verification efficiency.

In this paper, we first give a general solution to reducing the verification of an elementary dynamical system (EDS) to that of a polynomial dynamical system (PDS) by variable transformation. This is achieved by the following two steps: first, inspired by the work of [9, 24], we present an algorithm to reducing an EDS to a PDS; second, we prove the EDS is simulated by the PDS in the sense that for any trajectory of the EDS, there always exists a trajectory of the PDS corresponding to it, so that the PDS can be seen as an abstraction of the EDS. Therefore, safety verification and stability analysis of the EDS is naturally reduced to the counterparts of the PDS. Then, we discuss how to extend the approach to abstract an EHS by a polynomial HS (PHS). Finally, we discuss how to combine the abstraction technique presented here with invariant generation techniques for PHSs to verify some complicated real-world examples.

The benefits of our approach includes at least the following folds: 1) it opens a wider window to the verification of EHSs, as all the well-established verification techniques for PHSs are applicable to EHSs; 2) it provides the possibility to verify unbounded safety property, and to analyze stability of EHSs; 3) it provides the possibility to avoid error inflation in approximating the reachable sets. As a result, one can flexibly verify an EHS either exactly by symbolic computation, or approximately by numeric computation, or by their combination, depending on the tradeoff between efficiency and precision; 4) using the abstraction, one can verify more complicated properties of a considered EHS, as invariants with transcendental functions that imply the property to be verified may be synthesized.

1.1 A Motivating Example

We use a real-world example of the guidance and control of a lunar lander , as illustrated by Figure 1, to explain the motivation of this work. The dynamics of the lunar lander is

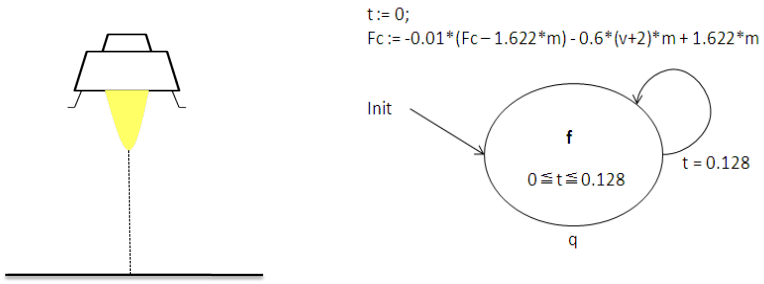

Fig. 1. The lunar lander and its guidance-control system.

given by

$$\mathbf{f} \mathrel{\widehat{=}} \begin{cases} \dot{v} &= \frac{F_c}{m} - 1.622 \\ \dot{m} &= -\frac{F_c}{2500} \\ \dot{F}_c &= 0 \\ \dot{t} &= 1 \end{cases}, \tag{1}$$

where v and m denote the vertical velocity and mass of the lunar lander; F_c denotes the thrust imposed on the lander, which is kept constant during one sampling cycle of length 0.128 seconds; at each sampling point, F_c is updated according to the guidance law shown in the right part of Figure 1. Note that the derivative of v involves non-polynomial expression $\frac{1}{m}$.

We want to verify that with the initial condition $v = -2$m/s, $m = 1250$kg, $F_c = 2027.5$N, the vertical velocity of the lunar lander will be kept around the target velocity -2m/s, i.e. $|v - (-2)| \leqslant \varepsilon$, where $\varepsilon = 0.05$ is the specified bound for fluctuation of v.

Paper organization. The rest of the paper is organized as: We briefly review some basic notions in Section 2; Section 3 is devoted to the transformation from an EDS to a PDS, and then we prove the PDS is an abstraction of the EDS by showing there is a simulation map between them in Section 4; Section 5 discusses how to extend the results to EHSs; Section 6 gives some case studies; Section 7 concludes this paper.

2 Preliminaries

In this section, we briefly introduce some basic notions that will be used later.

Elementary Functions. The class of elementary functions considered in this paper with variables $X = \{x_1, \ldots, x_n\}$ over $\mathbb{R}^n$, denoted by $\mathscr{E}(X; \mathbb{R}^n)$, ranged over $f, g, \ldots$, possibly with subscripts or superscripts, is constructed by the following grammar:

$$f, g ::= c \mid x_1, \ldots, x_n \mid f \pm g \mid f \times g \mid \frac{f}{g} \mid f^a \mid e^f \mid \ln(f) \mid \sin(f) \mid \cos(f) \mid f \circ g$$

Herein, $c \in \mathbb{R}$ is a real constant; $a \in \mathbb{Q}$ is a rational constant; $\frac{f}{g}$ is a *rational* function with $f \neq 0$; $e^f, \ln(f), \sin(f)$ and $\cos(f)$ are *transcendental* functions; $f \circ g$ denotes the composition of f and g. Note that the limitation to the above grammar is not essential. For example, tangent and cotangent functions $\tan(f), \cot(f)$ can be easily defined. Besides, the presented approach in this paper can also be extended to deal with other elementary functions such as inverse trigonometric functions $\arcsin(f), \arccos(f)$ etc.

Example 1 *$f(x, y) = \frac{1}{1+y^2} - x$ is an elementary function, wherein $\frac{1}{1+y^2}$ is a rational one; $h(x, y) = 2\sin(x)\cos(x) - 3\sin(x)$ is another elementary function, wherein $\sin(x)$ and $\cos(x)$ are* transcendental.

Continuous Dynamical Systems. A continuous dynamical system (CDS) is modeled by first-order autonomous ordinary differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{2}$$

where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a vector function, called a *vector field* in $\mathbb{R}^n$. If $\mathbf{f}$ is *elementary* (resp. *polynomial*), the corresponding CDS is called *elementary* (resp. *polynomial*). In many contexts, a CDS $\mathcal{S}$ may be equipped with an initial set Ξ and a domain $\mathcal{D}$, represented as a triple $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$. In what follows, all CDSs will refer to the triple form unless otherwise stated.

Hybrid Systems. Hybrid systems (HSs) are those systems that exhibit both continuous evolutions and discrete transitions between different modes. A widely adopted model of HSs is *hybrid automata* [1, 6].

Definition 1 (Hybrid Automaton). *A hybrid automaton (HA) is a system $\mathcal{H} \widehat{=} (Q, X, f, D, E, G, R, \Xi)$, where*

- *$Q = \{q_1, \ldots, q_m\}$ is a finite set of discrete states (or modes);*
- *$X = \{x_1, \ldots, x_n\}$ is a finite set of continuous state variables, with $\mathbf{x} = (x_1, \ldots, x_n)$ ranging over $\mathbb{R}^n$;*
- *$f : Q \to (\mathbb{R}^n \to \mathbb{R}^n)$ assigns to each mode $q \in Q$ a locally Lipschitz continuous vector field $\mathbf{f}_q$;*
- *D assigns to each mode $q \in Q$ a mode domain $\mathcal{D}_q \subseteq \mathbb{R}^n$;*
- *$E \subseteq Q \times Q$ is a finite set of discrete transitions;*
- *G assigns to each transition $e \in E$ a switching guard $G_e \subseteq \mathbb{R}^n$;*
- *R assigns to each transition $e \in E$ a reset function R_e: $\mathbb{R}^n \to 2^{\mathbb{R}^n}$;*
- *Ξ assigns to each $q \in Q$ a set of initial states $\Xi_q \subseteq \mathbb{R}^n$.*

An HA is called elementary *(resp.* polynomial*) if all its expressions are* elementary *(resp.* polynomial*).*

The state space of $\mathcal{H}$ is $\mathbb{H} \widehat{=} Q \times \mathbb{R}^n$, the domain of $\mathcal{H}$ is $D_{\mathcal{H}} \widehat{=} \bigcup_{q \in Q}(\{q\} \times \mathcal{D}_q)$, and the set of all initial states is denoted by $\Xi_{\mathcal{H}} \widehat{=} \bigcup_{q \in Q}(\{q\} \times \Xi_q)$. The semantics of $\mathcal{H}$ can be characterized by the set of *hybrid trajectories* accepted by $\mathcal{H}$ or the *reachable set* of states of $\mathcal{H}$, please refer to [1, 6] for the detail.

In what follows, in order to distinguish different CDSs (HSs), we may annotate a CDS $\mathcal{S}$ (an HS $\mathcal{H}$) with the vector of its continuous state variables $\mathbf{x}$ as $\mathcal{S}_{\mathbf{x}}$ ($\mathcal{H}_{\mathbf{x}}$).

3 Polynomializing

In this section, we present an algorithm to transform an elementary CDS $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$ (denoted by EDS for short) to a polynomial one (PDS). The basic idea is similar to the one given in [9, 24]. We introduce a fresh variable v for each non-polynomial term t_{np} in $\Xi, \mathbf{f}$ and $\mathcal{D}$, and then substitute v for t_{np}; meanwhile differentiate the two sides of the replacement equation $v = t_{np}$ w.r.t. time and obtain a new ODE $\dot{v} = \dot{t}_{np}$, which is then, together with the original $\mathbf{f}$, simplified to a polynomial ODE using the replacement equations $v = t_{np}$.

We will first demonstrate the idea on basic elementary functions: $\frac{1}{x}$, e^x, $\ln(x)$, $\sin(x)$, $\cos(x)$. In what follows, let $p(x, y), q(x, y) \in \mathbb{R}[x, y]$ be two polynomials. Consider the following ODE

$$\begin{cases} \dot{x} = p(x, y) + f \\ \dot{y} = q(x, y) \end{cases}, \tag{3}$$

where $f = \frac{1}{x}, e^x, \ln(x), \sin(x)$, or $\cos(x)$.

- When $f = \frac{1}{x}$ in (3), let $u = \frac{1}{x}$. Then $\dot{u} = -x^{-2} \cdot \dot{x} = -u^2 \cdot \dot{x}$. Thus, we obtain a new PDS
$$\begin{cases} \dot{x} = p(x, y) + u \\ \dot{y} = q(x, y) \\ \dot{u} = -u^2 \cdot (p(x, y) + u) \end{cases}.$$
- When $f = e^x$ in (3), set $u = e^x$. Then $\dot{u} = e^x \cdot \dot{x} = u \cdot \dot{x}$. Thus we derives a new PDS
$$\begin{cases} \dot{x} = p(x, y) + u \\ \dot{y} = q(x, y) \\ \dot{u} = u \cdot (p(x, y) + u) \end{cases}.$$
- When $x = \ln(x)$ in (3), it is a little complicated. First, take $u = \ln(x)$, so $\dot{u} = \frac{1}{x} \cdot \dot{x}$. Thus, we obtain a new system as
$$\begin{cases} \dot{x} = p(x, y) + u \\ \dot{y} = q(x, y) \\ \dot{u} = \frac{1}{x} \cdot (p(x, y) + u) \end{cases},$$
which is not polynomial yet. To eliminate $\frac{1}{x}$ in the new system, further set $v = \frac{1}{x}$. Such a variable v is called an *adjoint variable* of u. Thus, $\dot{v} = -x^{-2} \cdot \dot{x} = -v^2 \cdot \dot{x}$.

So (3) is eventually polynomialized to

$$\begin{cases} \dot{x} = p(x,y) + u \\ \dot{y} = q(x,y) \\ \dot{u} = v \cdot (p(x,y) + u) \\ \dot{v} = -v^2 \cdot (p(x,y) + u) \end{cases}.$$

– When $f = \sin(x)$ in (3), set $u = \sin(x)$. Then $\dot{u} = \cos(x) \cdot \dot{x}$. In this case, a new non-polynomial expression $\cos(x)$ appears. To eliminate it, further set $v = \cos(x)$, accordingly, $\dot{v} = -\sin(x) \cdot \dot{x} = -u \cdot \dot{x}$. Herein, the variable v is also called an *adjoint variable* of u. Eventually, the system is polynomialized to

$$\begin{cases} \dot{x} = p(x,y) + u \\ \dot{y} = q(x,y) \\ \dot{u} = v \cdot (p(x,y) + u) \\ \dot{v} = -u \cdot (p(x,y) + u) \end{cases}.$$

– When $f = \cos(x)$ in (3), it can be handled similarly as the case of $\sin(x)$.

Now, let's consider the cases of composition of elementary functions. Obviously, the outmost form of any composition of elementary functions must be one of $f \pm g, f \times g, \frac{f}{g}, f^a, e^f, \ln(f), \sin(f), \cos(f)$. Therefore we can iterate the above procedure discussed on basic cases until all the sub-expressions have been polynomialized. We illustrate the idea by the following example:

$$\begin{cases} \dot{x} = p(x,y) + \ln(\sin(y) + 1) \\ \dot{y} = q(x,y) \end{cases}. \tag{4}$$

To eliminate the non-polynomial expressions in (4), set $u = \sin(y)$, $v = \cos(y)$, $w = \ln(u+1)$, $z = \frac{1}{u+1}$. Then, (4) is polynomialized as

$$\begin{cases} \dot{x} = p(x,y) + w \\ \dot{y} = q(x,y) \\ \dot{u} = v \cdot q(x,y) \\ \dot{v} = -u \cdot q(x,y) \\ \dot{w} = z \cdot v \cdot q(x,y) \\ \dot{z} = -z^2 \cdot v \cdot q(x,y) \end{cases}.$$

The non-polynomial expressions occurring in the initial and domain constraints of the EDS can be coped with similarly.

Now, we implement the above idea by the following two steps (See Algorithm 4 in Appendix D[3]): in the first step, we reduce an EDS to a PDS by variable transformation. To this end, we implement Algorithm 1 (see Appendix D) that transforms an elementary expression to a polynomial one by variable transformation, and meanwhile records all replacement equations produced during the course in the set *eqs*, which is a global variable initialized to *empty* when starting the algorithm. We first call Algorithm 1 to each of the outmost non-polynomial expressions occurring in the initial and

[3] Here, we see an EDS as a specific EHS only with one mode, without discrete jumps.

domain constraints of the EDS, and then to each of the RHSs (right-hand-sides) of the ODEs, and finally replace all non-polynomial expressions occurring in the considered EDS with the corresponding introduced variable in *eqs*. In the second step, in order to guarantee the resulted PDS to be an abstraction of the EDS in the sense of simulation that will be discussed later, we have to differentiate the equations eqs obtained in Algorithm 1 and put them as part of the resulted PDS. This is implemented by Algorithm 2 (see Appendix D).

Given an EDS $\mathcal{S}_{\mathbf{x}} = \langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, we use $\mathbf{v} = (v_1, \ldots, v_m)$ to denote the new variables introduced in step 1. According to Algorithm 1, each v_i corresponds to a replacement in eqs. We will denote by $\mathbf{v} = \Gamma(\mathbf{x})$ all the replacements recorded in eqs. Then the ODEs of the resulted PDS can be expressed as $(\dot{\mathbf{x}}, \dot{\mathbf{v}}) = (\mathbf{f}[\mathbf{v}/\Gamma(\mathbf{x})], \dot{\Gamma(\mathbf{x})}[\mathbf{v}/\Gamma(\mathbf{x})])$, where $expr[\mathbf{v}/\Gamma(\mathbf{x})]$ means replacing any occurrence of the non-polynomial expression $\Gamma(\mathbf{x})_i$ (the i-th component of $\Gamma(\mathbf{x})$) in the expression $expr$ by the corresponding variable v_i. Thus the resulted PDS can be expressed as

$$\langle\, \Xi[\mathbf{v}/\Gamma(\mathbf{x})], (\mathbf{f}[\mathbf{v}/\Gamma(\mathbf{x})], \dot{\Gamma(\mathbf{x})}[\mathbf{v}/\Gamma(\mathbf{x})]), \mathcal{D}[\mathbf{v}/\Gamma(\mathbf{x})] \,\rangle,$$

denoted by $\Gamma(\mathcal{S}_{\mathbf{x}})$ for short.

Example 2 *The EDS* $\langle \Xi \mathrel{\widehat{=}} x = 1, \mathbf{f} \mathrel{\widehat{=}} (\dot{x} = e^x - 1), \mathcal{D} \mathrel{\widehat{=}} x^5 < e^x \rangle$ *can be abstracted to the PDS* $\langle x = 1, (\dot{x} = v - 1; \dot{v} = v(v-1)), x^5 < v \rangle$ *by the map* $\Theta : \mathbb{R} \to \mathbb{R}^2$ *such that* $\Theta(x) = (x, v) = (x, e^x)$.

4 Abstraction of Elementary Dynamical Systems

In this section, we will show that given an EDS, the resulted PDS obtained using the transformation in the previous section is an abstraction of the EDS in the sense of simulation defined in [21].

Definition 3 (Simulation [21]) *Given two CDSs* $\mathcal{S}_{\mathbf{x}} = \langle \Xi_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}, \mathcal{D}_{\mathbf{x}} \rangle$ *and* $\mathcal{S}_{\mathbf{y}} = \langle \Xi_{\mathbf{y}}, \mathbf{f}_{\mathbf{y}}, \mathcal{D}_{\mathbf{y}} \rangle$*, we say* $\mathcal{S}_{\mathbf{y}}$ *simulates* $\mathcal{S}_{\mathbf{x}}$ *or* $\mathcal{S}_{\mathbf{x}}$ *is simulated by* $\mathcal{S}_{\mathbf{y}}$ *via* $\Theta : \mathbb{R}^{|\mathbf{x}|} \to \mathbb{R}^{|\mathbf{y}|}$*, if* Θ *satisfies*

1. $\Theta(\Xi_{\mathbf{x}}) \subseteq \Xi_{\mathbf{y}}$ *and* $\Theta(\mathcal{D}_{\mathbf{x}}) \subseteq \mathcal{D}_{\mathbf{y}}$*, and*
2. *for any trajectory* $\tau : [0, T) \to \mathcal{D}_{\mathbf{x}}$ *of* $\mathcal{S}_{\mathbf{x}}$*,* $\Theta \circ \tau$ *is a trajectory of* $\mathcal{S}_{\mathbf{y}}$.

Such Θ *is called a* simulation map.

In essence, a simulation is just a map Θ that transfers each trajectory of $\mathcal{S}_{\mathbf{x}}$ to a trajectory of $\mathcal{S}_{\mathbf{y}}$. The inverse of Θ, denoted by Θ^{-1}, is defined by $\Theta^{-1}(Y) = \{\mathbf{x} \mid \Theta(\mathbf{x}) \in Y\}$, i.e., $\Theta^{-1}(Y)$ denotes the preimage of Y under Θ. Obviously, Θ^{-1} does not satisfy the above conditions of a simulation map in general, although Θ does, which means a simulation is not symmetric in general.

The following theorem indicates that there does exist a simulation map between a given EDS and the corresponding transformed PDS, which can be easily constructed from the polynomializing.

Theorem 4. *Given an EDS $\mathcal{S}_{\mathbf{x}} = \langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, define $\Theta : \mathbb{R}^{|\mathbf{x}|} \to \mathbb{R}^{|\mathbf{x}|+|\mathbf{v}|}$ as $\Theta(\mathbf{x}) = (\mathbf{x}, \mathbf{v}) = (\mathbf{x}, \Gamma(\mathbf{x}))$. Then Θ is a simulation map between $\mathcal{S}_{\mathbf{x}}$ and $\Gamma(\mathcal{S}_{\mathbf{x}})$.*

Regarding the simulation map Θ considered in Theorem 4, its inverse is a projection map, with the following nice property.

Theorem 5 (Projection Trajectory). *Given an EDS $\mathcal{S}_{\mathbf{x}}$, suppose Θ is the simulation map from $\mathcal{S}_{\mathbf{x}}$ to $\Gamma(\mathcal{S}_{\mathbf{x}})$ defined as in Theorem 4. If $\tau_{(\mathbf{x},\mathbf{v})}(t)$ is the trajectory of $\Gamma(\mathcal{S}_{\mathbf{x}})$ starting from $(\mathbf{x}_0, \Gamma(\mathbf{x}_0))$, then $\Theta^{-1}(\tau_{(\mathbf{x},\mathbf{v})}(t)) = \tau_{(\mathbf{x},\mathbf{v})}(t) \restriction_{\mathbf{x}}$ is the trajectory of $\mathcal{S}_{\mathbf{x}}$ starting from $\mathbf{x}_0$.*

The notion of *continuous invariant* [18, 11] plays a central role in the verification of CDSs and HSs in a deductive way.

Definition 2 (Continuous Invariant [18, 11]). *A subset $I \subseteq \mathbb{R}^n$ is called a* continuous invariant *(CI) of a CDS $\mathcal{S} := (\Xi, \mathbf{f}, \mathcal{D})$ iff $\Xi \subseteq I$ and*

$$\forall \mathbf{x}_0 \in I \forall T \geqslant 0 (\forall t \in [0, T].\, \tau(\mathbf{x}_0; t) \in \mathcal{D} \Longrightarrow \forall t \in [0, T].\, \tau(\mathbf{x}_0; t) \in I) \tag{5}$$

where $\tau(\mathbf{x}_0; t)$ stands for the trajectory of $\mathcal{S}$ starting from $\mathbf{x}_0$.

As in [21], we will prove the the following theorem which indicates the inverse of the simulation map considered in Theorem 4 preserves invariants so that synthesizing invariants of an EDS is reduced to that of the resulted PDS, and the verification of the EDS is therefore reduced to that of the PDS.

Theorem 6 (Simulation Invariant of EDSs). *Given an EDS $\mathcal{S}_{\mathbf{x}} = \langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, suppose Θ is the simulation map from $\mathcal{S}_{\mathbf{x}}$ to $\Gamma(\mathcal{S}_{\mathbf{x}})$ defined as in Theorem 4. Then, if $I \subseteq \mathbb{R}^{|\mathbf{x}|+|\mathbf{v}|}$ is a CI of $\Gamma(\mathcal{S}_{\mathbf{x}})$, then $\Theta^{-1}(I) \cap \mathcal{D}$ is a CI of $\mathcal{S}_{\mathbf{x}}$.*

Because of the decidability of elementary algebra and geometry [25], the notion of semi-algebraic set plays very important role in the verification of PDSs or PHSs. A semi-algebraic set I is represented by

$$I := \{(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^{n+m} \mid \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}, \mathbf{v}) \triangleright 0\},$$

where $\triangleright \in \{<, \leqslant\}$ and p_{kj} are polynomials in $\mathbb{R}[\mathbf{x}, \mathbf{v}]$. Using Theorem 6, we can first synthesize a semi-algebraic invariant for the polynomialized PDS using existing approaches like in [20, 18, 11, 10] for synthesizing semi-algebraic invariants for the PDS. Then, using the replacement map Γ, we can obtain invariants containing non-polynomial expressions for the considered EDS.

Corollary 7 *Given an EDS $\mathcal{S}_{\mathbf{x}} = \langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, if $I = \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}, \mathbf{v}) \triangleright 0$ is a semi-algebraic invariant of $\Gamma(\mathcal{S}_{\mathbf{x}})$, then*

$$I^* \mathrel{\widehat{=}} \{\mathbf{x} \in \mathbb{R}^{|\mathbf{x}|} \mid I[\Gamma(\mathbf{x})/\mathbf{v}] = \bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(\mathbf{x}, \Gamma(\mathbf{x})) \triangleright 0\},$$

is a continuous invariant of $\mathcal{S}_{\mathbf{x}}$.

Stability is another important property of CDSs. The author of [21] observed that the stability may not be kept w.r.t. the inverse of a simulation map. However, the inverse of the simulation map defined in Theorem 4 preserves stability faithfully. Thus, the stability analysis of a considered EDS is reduced to that of the resulted PDS, which can be achieved by the well-established stability analysis approaches for PDSs [12].

Theorem 8 (Simulation Stability). *Given an EDS $\mathcal{S}_{\mathbf{x}} = \langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, suppose Θ is the simulation map from $\mathcal{S}_{\mathbf{x}}$ to $\Gamma(\mathcal{S}_{\mathbf{x}})$ defined as in Theorem 4. Then, if an equilibrium point $\mathbf{y}_e = (\mathbf{x}_e, \mathbf{v}_e) \in \Theta(\mathcal{D})$ of $\Gamma(\mathcal{S}_{\mathbf{x}})$ is stable (either Lyapunov or asymptotic), then $\Theta^{-1}(\mathbf{y}_e) = \mathbf{x}_e$ is also an equilibrium point of $\mathcal{S}_{\mathbf{x}}$, which is stable (correspondingly, Lyapunov or asymptotic).*

5 Abstraction of Elementary Hybrid Systems

In this section, we investigate how to abstract an elementary hybrid system (EHS) to a polynomial hybrid system (PHS). To this end, we first define a simulation map between two HSs, then we show how to polynomialize an EHS to a PHS such that there is a simulation map between them.

By Definition 1, an HS $\mathcal{H} \widehat{=} (Q, X, f, D, E, G, R, \Xi)$ consists of a set of CDSs $\{\langle \Xi_q, \mathbf{f}_q, \mathcal{D}_q \rangle\}_{q \in Q}$ with discrete jumps among them. Therefore we will generalize the notion of simulation in Definition 3 to HSs by unifying the simulations of $\langle \Xi_q, \mathbf{f}_q, \mathcal{D}_q \rangle$ together with the discrete jumps E along the lines of [8, 19, 22] as follows:

Definition 9 (HA-Simulation) *An HS $\mathcal{H}_{\mathbf{y}} \widehat{=} (Q, Y, f_{\mathbf{y}}, D_{\mathbf{y}}, E, G_{\mathbf{y}}, R_{\mathbf{y}}, \Xi_{\mathbf{y}})$ simulates another HS $\mathcal{H}_{\mathbf{x}} \widehat{=} (Q, X, f_{\mathbf{x}}, D_{\mathbf{x}}, E, G_{\mathbf{x}}, R_{\mathbf{x}}, \Xi_{\mathbf{x}})$ via a map $\Theta : \mathbb{R}^{|\mathbf{x}|} \to \mathbb{R}^{|\mathbf{y}|}$, if it satisfies the following conditions:*

1. *$\Theta(G_{\mathbf{x},e}) \subseteq G_{\mathbf{y},e}$ for every $e \in E$;*
2. *$\Theta(R_{\mathbf{x},e}(\mathbf{x}_0)) \subseteq R_{\mathbf{y},e}(\Theta(\mathbf{x}_0))$ for all $\mathbf{x}_0 \in \mathrm{Dom}(R_{\mathbf{x},e})$ and every $e \in E$;*
3. *$\langle \Xi_{\mathbf{y},q}, \mathbf{f}_{\mathbf{y},q}, \mathcal{D}_{\mathbf{y},q} \rangle$ simulates $\langle \Xi_{\mathbf{x},q}, \mathbf{f}_{\mathbf{x},q}, \mathcal{D}_{\mathbf{x},q} \rangle$ via Θ for each $q \in Q$.*

Note that in Definition 9, we have assumed that the modes and jumps of $\mathcal{H}_{\mathbf{y}}$ are the same as $\mathcal{H}_{\mathbf{x}}$. Intuitively, a simulation relation Θ between two hybrid systems $\mathcal{H}_{\mathbf{x}}$ and $\mathcal{H}_{\mathbf{y}}$ is a map such that if $\tau : [0, T) \to \mathbb{H}_{\mathbf{x}}$ is a trajectory of $\mathcal{H}_{\mathbf{x}}$ then $\Theta(\tau)$ is a trajectory of $\mathcal{H}_{\mathbf{y}}$ over time interval $[0, T)$. Such simulation relation is interesting as its inverse preserves invariants similar to the case of CDSs, so that we can reduce the verification of an EHS to that of the polynomialized PHS. Namely,

Theorem 10 (Simulation Invariant of HSs). *If $\mathcal{H}_{\mathbf{y}}$ simulates $\mathcal{H}_{\mathbf{x}}$ via map Θ as definition 9 and I is an invariant of $\mathcal{H}_{\mathbf{y}}$, then $\Theta^{-1}(I)$ is an invariant of $\mathcal{H}_{\mathbf{x}}$.*

In the previous sections, we have presented a method to abstract an EDS to a PDS such that the PDS is a simulation of the EDS. Now we show, given an HA $\mathcal{H}_{\mathbf{x}}$, how to construct a simulation map Θ and a corresponding simulation HA $\mathcal{H}_{\mathbf{y}}$, as defined in Definition 9.

Actually, this can be easily done by just extending the previous abstraction approach a bit to take into account guard constraints and reset functions. That is, for each mode q of $\mathcal{H}_{\mathbf{x}}$, we introduce a new variable v for each non-polynomial term t_{np} appearing in

$\Xi_q, \mathbf{f}_q, \mathcal{D}_q$, and G_e, R_e for each $e \in E$ starting from q, and meanwhile add $v = t_{np}$ to the set of replacement equations. In this way, for each mode q we will obtain a set of new variables. For ease of presentation, we take the union of all new variables at each mode and denote it by $\mathbf{v}$. Accordingly, the set of replacement equations is denoted by $\mathbf{v} = \Gamma(\mathbf{x})$. Let $\Theta(\mathbf{x}) = \mathbf{y} = (\mathbf{x}, \Gamma(\mathbf{x}))$. Then as in Theorem 4, we can use Θ to build a simulation $\mathcal{H}_\mathbf{y}$ of $\mathcal{H}_\mathbf{x}$ such that

- for each $q \in Q$
  - $\Xi_{\mathbf{y},q} \widehat{=} \Xi_{\mathbf{x},q}[\mathbf{v}/\Gamma(\mathbf{x})]$;
  - $\mathbf{f}_{\mathbf{y},q} \widehat{=} (\mathbf{f}_{\mathbf{x},q}[\mathbf{v}/\Gamma(\mathbf{x})], \dot{\Gamma(\mathbf{x})}[\mathbf{v}/\Gamma(\mathbf{x})])$;
  - $\mathcal{D}_{\mathbf{y},q} \widehat{=} \mathcal{D}_{\mathbf{x},q}[\mathbf{v}/\Gamma(\mathbf{x})]$; and
- for each $e \in E$,
  - $G_{\mathbf{y},e} \widehat{=} G_{\mathbf{x},e}[\mathbf{v}/\Gamma(\mathbf{x})]$;
  - $R_{\mathbf{y},e}(\mathbf{x}, \mathbf{v}) \widehat{=} \{(\mathbf{x}', \mathbf{v}')\}$ such that $\mathbf{x}' \widehat{=} R_{\mathbf{x},e}(\mathbf{x})[\mathbf{v}/\Gamma(\mathbf{x})]$ and $\mathbf{v}' \in \mathbb{R}^{|\mathbf{v}|}$.

It is easy to check according to Definition 9 that Θ is a simulation relation between $\mathcal{H}_\mathbf{y}$ and $\mathcal{H}_\mathbf{x}$; furthermore, $\mathcal{H}_\mathbf{y}$ is a PHS. The implementation is given in Algorithm 4.

Example 11 *Consider the HA shown in the left part of Figure 2.*

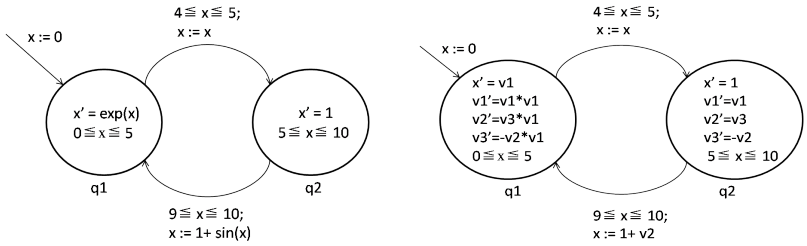

Fig. 2. The original and transformed hybrid system for Example 11.

Using our algorithm, we can get a simulation map $\Theta : \mathbb{R} \to \mathbb{R}^4$ such that $\Theta(x) = (x, v_1, v_2, v_3) = (x, e^x, \sin(x), \cos(x))$. The transformed HA is shown in the right part of Figure 2. Note that the omitting $\mathbf{v}$ in the initial set, domains, guards and reset functions means that we assume $\mathbf{v}$ can be, or reset to, any value in our abstraction.

Usually the abstraction $\mathcal{H}_\mathbf{y}$ of $\mathcal{H}_\mathbf{x}$ is very coarse because we abstract the replacement relation $\mathbf{v} = \Gamma(\mathbf{x})$ away. In practice, to generate sufficiently strong invariants for safety verification of hybrid systems, we need to refine $\mathcal{H}_\mathbf{y}$ using $\mathbf{v} = \Gamma(\mathbf{x})$. Some details will be discussed in the following section on experiments.

6 Implementation and Experimental Results

We have implemented a tool based on the algorithms proposed above (see Appendix D). Given any input EHS, it automatically abstracts the EHS to a PHS, and meanwhile generate a set of replacement equations used in the abstraction. The abstraction of EDSs to PDSs can be dealt with as a special case. Next, we use several examples to show how to do safety verification for EDSs or EHSs using our abstraction techniques and tool.

6.1 An Example with Transcendental ODEs

Consider the EDS $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$ with $\Xi \mathrel{\widehat{=}} (x+0.5)^2 + (y-0.5)^2 - 0.16 \leqslant 0$, $\mathcal{D} \mathrel{\widehat{=}} -2 \leqslant x \leqslant 2 \wedge -2 \leqslant y \leqslant 2$ and

$$\mathbf{f} \mathrel{\widehat{=}} \begin{cases} \dot{x} = e^{-x} + y - 1 \\ \dot{y} = -\sin^2(x) \end{cases} . \tag{6}$$

We want to verify that the unsafe region $\mathcal{U} \mathrel{\widehat{=}} (x-0.7)^2 + (y+0.7)^2 - 0.09 \leqslant 0$ will never be reached.

By our algorithm, we get a simulation map $\Theta : \mathbb{R}^2 \to \mathbb{R}^5$ such that $\Theta(x, y) = (x, y, v_1, v_2, v_3) = (x, y, \sin(x), e^{-x}, \cos(x))$, and a corresponding PDS that simulates $(\Xi, \mathbf{f}, \mathcal{D})$, denoted by $(\Theta(\Xi), \Theta(\mathbf{f}), \Theta(\mathcal{D}))$, such that $\Theta(\Xi) \mathrel{\widehat{=}} \Xi$, $\Theta(\mathcal{D}) \mathrel{\widehat{=}} \mathcal{D}$, and

$$\Theta(\mathbf{f}) \mathrel{\widehat{=}} \begin{cases} \dot{x} &= v_2 + y - 1 \\ \dot{y} &= -v_1^2 \\ \dot{v}_1 &= v_3(v_2 + y - 1) \\ \dot{v}_2 &= -v_2(v_2 + y - 1) \\ \dot{v}_3 &= -v_1(v_2 + y - 1) \end{cases} . \tag{7}$$

If the template-based method [20, 11] is used for invariant generation for the transformed PDS, and if we use a template with variables x, y, then the derivatives of x, y over $\Theta(\mathcal{D})$ need to be considered. According to (7), this means the values of v_1, v_2 over $\Theta(\mathcal{D})$ are relevant to invariant generation. Therefore we refine $\Theta(\mathcal{D})$ by $\tilde{\Theta}(\mathcal{D}) \mathrel{\widehat{=}} \mathcal{D} \wedge v_1 = \sin(x) \wedge v_2 = e^{-x}$. Similar strategies will be applied in the following investigation of other examples.

The part $v_1 = \sin(x) \wedge v_2 = e^{-x}$ in $\tilde{\Theta}(\mathcal{D})$ can be approximated using Taylor model representations [14]. The basic idea is to represent $\sin(x)$ (or e^{-x}) over the interval $[-2, 2]$ as $p(x) + I$, where $p(x)$ is the polynomial expansion of $\sin(x)$ (or e^{-x}) up to a certain degree, and $I = [l, u]$ is an over-approximation of the truncation error. Here we use the tool COSY INFINITY[4] to compute the Taylor polynomials of $\sin(x)$ and e^{-x} up to degree 6 and the corresponding truncation errors. Thus $\tilde{\Theta}(\mathcal{D})$ is abstracted into

$$TM(\tilde{\Theta}(\mathcal{D})) \mathrel{\widehat{=}} \mathcal{D} \wedge p_1(x) + l_1 \leqslant v_1 \leqslant p_1(x) + u_1 \wedge p_2(x) + l_2 \leqslant v_2 \leqslant p_2(x) + u_2 \,.$$

For the details please refer to Appendix B.

Finally we get a polynomial abstraction $\langle \Xi, \Theta(\mathbf{f}), TM(\tilde{\Theta}(\mathcal{D})) \rangle$ of $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$. Then by applying the SOS-relaxation approach to invariant generation for polynomial systems [20, 10], we successfully generate a continuous invariant $p(x, y) \leqslant 0$ of degree 5 (see Appendix C) using the Matlab-based tool YALMIP [13] and SDPT3 [26]. By Theorem 6, $p(x, y) \leqslant 0$ is a continuous invariant of $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$ and the safety property $\neg\mathcal{U}$ is verified. Please see Figure 3 for an illustration of $\mathbf{f}$ (the black arrows), $\mathcal{D}$ (the outer white box), the synthesized invariant $p(x, y) \leqslant 0$ (the shaded light blue area with curved boundary), Ξ (the blue circle inside the invariant) and $\mathcal{U}$ (the red circle outside the invariant).

[4] `http://bt.pa.msu.edu/index_cosy.htm`

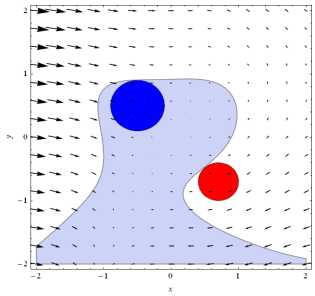

Fig. 3. Synthesized invariant for the system in Section 6.1.

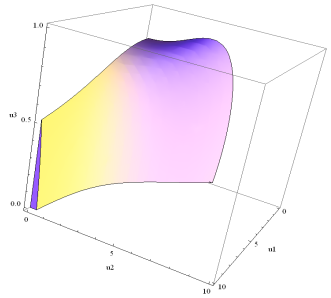

Fig. 4. Synthesized invariant for the HIV transmission model.

6.2 HIV Transmission

The following continuous dynamics, with the assumption that there is no recruitment of population, has been developed to model HIV transmission [2]

$$\mathbf{f} \mathrel{\widehat{=}} \begin{cases} \dot{u}_1 = -\frac{\beta c u_1 u_2}{u_1+u_2+u_3} - \mu u_1 \\ \dot{u}_2 = \frac{\beta c u_1 u_2}{u_1+u_2+u_3} - (\mu+\nu) u_2 \\ \dot{u}_3 = \nu u_2 - \alpha u_3 \end{cases} , \tag{8}$$

where $u_1(t), u_2(t), u_3(t)$ denote the part of population that is HIV susceptible, HIV infected, and that has AIDS respectively, β is the possibility of infection per partner contact, c is the rate of partner change, μ is the death rate of non-AIDS population, α is the death rate of AIDS patients, and ν is the rate at which HIV infected people develop AIDS. Note that the dynamics involves non-polynomial term $\frac{1}{u_1+u_2+u_3}$. In this paper, the parameters are chosen to be $\beta = 0.2, c = 10, \mu = 0.008, \alpha = 0.95, \nu = 0.1$.

We want to verify that with the initial set

$$\Xi \mathrel{\widehat{=}} u_1 \in [9.985, 9.995] \wedge u_2 \in [0.005, 0.015] \wedge u_3 \in [0, 0.003],$$

the population of AIDS patients alive will always be below 1 (the population is measured in thousands). That is, the system $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$ satisfies $\mathcal{S} \mathrel{\widehat{=}} u_3 \leqslant 1$, where

$$\mathcal{D} \mathrel{\widehat{=}} u_1 \geqslant 0 \wedge u_2 \geqslant 0 \wedge u_3 \geqslant 0 \wedge 0 < u_1 + u_2 + u_3 \leqslant 10.013\,.^{5}$$

By introducing a new variable v_1 to represent $\frac{1}{u_1+u_2+u_3}$, we obtain a simulation map Θ and a corresponding simulation $\langle \Theta(\Xi), \Theta(\mathbf{f}), \Theta(\mathcal{D}) \rangle$ of $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$, where $\Theta(\Xi) \mathrel{\widehat{=}} \Xi$, $\Theta(\mathcal{D}) \mathrel{\widehat{=}} \mathcal{D} \wedge v_1(u_1 + u_2 + u_3) = 1$ after refinement using the replacement equation, and

$$\Theta(\mathbf{f}) \mathrel{\widehat{=}} \begin{cases} \dot{u}_1 = -\beta c u_1 u_2 v_1 - \mu u_1 \\ \dot{u}_2 = \beta c u_1 u_2 v_1 - (\mu+\nu) u_2 \\ \dot{u}_3 = \nu u_2 - \alpha u_3 \end{cases} . \tag{9}$$

Note that only those derivatives relevant to invariant generation are presented in $\Theta(\mathbf{f})$.

[5] According to dynamics (8), the entire population is non-increasing, so $u_1 + u_2 + u_3$ has an upper bound.

By assuming an invariant template $p(u_1, u_2, u_3) \leqslant 0$ of degree 4 and applying the SOS-relaxation approach, we generate an invariant (see Appendix C) which verifies $\mathcal{S}$. Figure 4 is an illustration of the synthesized invariant.

As a comparison, using the tool Flow* [3] with a fixed time step of 0.001, adaptive orders of 1 to 8, and remainder estimation of 1e-6, we can only compute the reachable set of $\langle \Xi, \mathbf{f}, \mathcal{D} \rangle$ up to time 3.319. Figure 5 shows that the computed $u_3(t)$ is likely to be diverging.

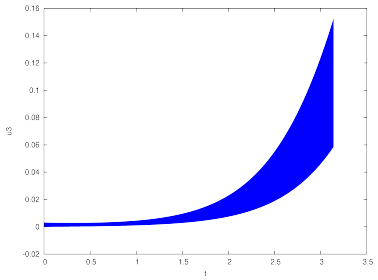

Fig. 5. Computed reachable set for the HIV transmission model.

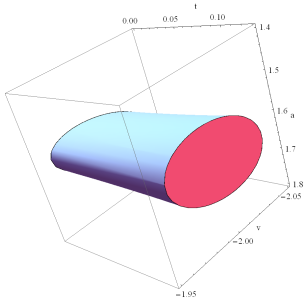

Fig. 6. Synthesized invariant for the lunar lander system.

6.3 The Lunar Lander

The motivating example of the lunar lander as illustrated by the right part of Figure 1 can be represented by the HA $\mathcal{H} \mathrel{\widehat{=}} (Q, X, f, D, E, G, R, \Xi)$, where

- $Q \mathrel{\widehat{=}} \{q\}$; $X \mathrel{\widehat{=}} \{v, m, F_c, t\}$;
- $\mathbf{f}_q$ is given by (1);
- $\mathcal{D}_q \mathrel{\widehat{=}} 1 \leqslant t \leqslant 0.128$;
- $E \mathrel{\widehat{=}} \{e\} = \{(q, q)\}$;
- $G_e \mathrel{\widehat{=}} t = 0.128$;
- $R_e(v, m, F_t, t) = \{(v,\ m,\ -0.01(F_c - 1.622m) - 0.6(v + 2)m + 1.622m,\ 0)\}$;
- $\Xi_q \mathrel{\widehat{=}} v = -2 \wedge m = 1250 \wedge F_c = 2027.5 \wedge t = 0$.

We will verify the safety property $\mathcal{S} \mathrel{\widehat{=}} -2.05 \leqslant v \leqslant -1.95$ for this system.

The way of constructing a simulation for $\mathcal{H}$ is slightly different from what has been proposed above: instead of representing $\frac{1}{m}$ as a new variable, we will replace $\frac{F_c}{m}$ by a new variable a and thus get a simulation map $\Theta : \mathbb{R}^4 \to \mathbb{R}^3$ such that $\Theta(v, m, F_c, t) = (v, a, t) = (v, \frac{F_c}{m}, t)$.

Using Θ we construct a HA $\mathcal{H}' \mathrel{\widehat{=}} \big(Q, \Theta(X), \Theta(f), \Theta(D), E, \Theta(G), \Theta(R), \Theta(\Xi)\big)$ of $\mathcal{H}$ such that

- continuous state variables $\Theta(X) \mathrel{\widehat{=}} \{v, a, t\}$;

– continuous dynamics

$$\Theta(\mathbf{f}) \mathrel{\widehat{=}} \begin{cases} \dot{v} = a - 1.622 \\ \dot{a} = \frac{a^2}{2500} \\ \dot{t} = 1 \end{cases} ; \tag{10}$$

– initial set $\Theta(\Xi_q) \mathrel{\widehat{=}} v = -2 \wedge a = 1.622 \wedge t = 0$;
– reset function $\Theta(R_e)(v, a, t) = \{(v,\ -0.01(a - 1.622) - 0.6(v + 2) + 1.622,\ 0)\}$;
– and the other components are the same as $\mathcal{H}$.

It is easy to check that $\mathcal{H}'$ is a simulation of $\mathcal{H}$ according to Definition 9.

For $\mathcal{H}'$, using the SOS-relaxation approach with a template $p(v, a, t) \leqslant 0$ of degree 6, we generate an invariant (see Appendix C) that can verify $\mathcal{S}$. By Theorem 10, a substitution of $\frac{F_c}{m}$ for a in $p(v, a, t) \leqslant 0$ gives an invariant of $\mathcal{H}$, and thus the safety property of the original system is verified. Figure 6 is an illustration of the synthesized invariant for the lunar lander system.

6.4 Two-Tank System

The two-tank system shown in the left part of Figure 7 has been studied in [4, 7] as a benchmark for bounded model checking (BMC) of hybrid systems. It models two connected tanks, the liquid levels of which are denoted by x_1 and x_2 respectively. The system switches from mode q_1 (or q_2) to q_2 (or q_1) when x_2 reaches 1 at q_1 (or q_2). The system's dynamics involve non-polynomial terms such as $\sqrt{x_1}$ or $\sqrt{x_1 - x_2 + 1}$. The verification objective is to show that starting from mode q_1 with the initial set $\Xi \mathrel{\widehat{=}} 5.25 \leqslant x_1 \leqslant 5.75 \wedge 0 \leqslant x_2 \leqslant 0.5$, the system will never reach the unsafe set $\mathcal{U} \mathrel{\widehat{=}} (x_1 - 4.25)^2 + (x_2 - 0.25)^2 - 0.0625 \leqslant 0$ when staying at mode q_1.

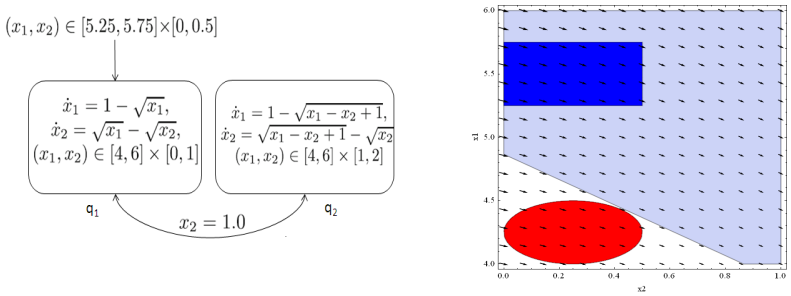

Fig. 7. The HA model and the synthesized invariant for the two-tank system.

Let $\mathcal{D}_{q_1} \mathrel{\widehat{=}} 4 \leqslant x_1 \leqslant 6 \wedge 0 \leqslant x_2 \leqslant 1$ and

$$\mathbf{f}_{q_1} \mathrel{\widehat{=}} \begin{cases} \dot{x}_1 = 1 - \sqrt{x_1} \\ \dot{x}_2 = \sqrt{x_1} - \sqrt{x_2} \end{cases} \tag{11}$$

be the domain and vector field of mode q_1. By introducing two new variables v_1 and v_2 to represent $\sqrt{x_1}$ and $\sqrt{x_2}$ respectively (and some other new variables), we obtain a simulation map Θ and a corresponding PHS such that

– the vector field of mode q_1 is

$$\Theta(\mathbf{f}_{q_1}) \widehat{=} \begin{cases} \dot{x}_1 = 1 - v_1 \\ \dot{x}_2 = v_1 - v_2 \end{cases}, \tag{12}$$

where only those derivatives relevant to invariant generation are presented; the domain of mode q_1 is $\Theta(\mathcal{D}_{q_1}) \widehat{=} \mathcal{D}_{q_1} \wedge v_1 = \sqrt{x_1} \wedge v_2 = \sqrt{x_2}$, or equivalently, $\Theta(\mathcal{D}_{q_1}) \widehat{=} \mathcal{D}_{q_1} \wedge v_1^2 = x_1 \wedge v_2^2 = x_2 \wedge v_1 \geqslant 0 \wedge v_2 \geqslant 0$, after refinement by the replacement equations;

– the initial set, jumps, transition guards, reset functions, and the domain of mode q_2 are kept the same as the original system.

We assume a simple linear invariant template $c_0 + c_1 x_1 + c_2 x_2 \leqslant 0$ for mode q_1 and adopt the domain, i.e. $\mathcal{D}_{q_2} \widehat{=} x_1 \in [4, 6] \wedge x_2 \in [1, 2]$, as a trivial invariant for mode q_2. Using the SOS-relaxation method, we get an invariant $8.133827714 - 1.6714x_1 - 1.6729x_2 \leqslant 0$ that verifies the safety property $\neg\mathcal{U}$. Please see the right part of Figure 7 for an illustration of the initial set (the inner blue rectangle), domain (the outer white rectangle), vector field (the black arrows), unsafe region (the red ellipse), and the synthesized invariant (the light blue trapezoid) for mode q_1.

As a comparison, the BMC tool hydlogic can prove the safety property of the two-tank system for two steps [7], and the tool iSAT-ODE can do so up to a depth of 40 with a slightly smaller initial set [4] .

7 Conclusion

In this paper, we first presented an approach to reducing an EDS (EHS) to a PDS (resp. PHSs) by variable transformation, and then established a simulation relation between them, so that verification and stability analysis of the EDS (EHS) can be reduced to the counterparts of the corresponding PDS (PHS). Thus our work opens a wider window to verification and stability analysis of EDSs and EHSs, as all the well-established techniques for PDSs and PHSs are applicable to them, which are based on either numeric computation, or symbolic computation or their combination. We apply the approach to the safety verification of several real-world examples. By comparing with the existing approaches, the experimental results indicated the effectiveness of our approach. In future, we will further study how to get more accurate abstractions from the non-polynomial equations of variable replacement.

References

1. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid Systems'93, LNCS, vol. 736, pp. 209–229 (1993)
2. Anderson, R.M.: The role of mathematical models in the study of HIV transmission and the epidemiology of AIDS. Journal of Acquired Immune Deficiency Syndromes 3(1), 241–256 (1988)
3. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: CAV'13. pp. 258–263 (2013)

4. Eggers, A., Ramdani, N., Nedialkov, N., Fränzle, M.: Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. Software & Systems Modeling pp. 1–28 (2012)
5. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: CADE'13. pp. 208–214 (2013)
6. Henzinger, T.: The theory of hybrid automata. In: LICS'96. pp. 278–292 (1996)
7. Ishii, D., Ueda, K., Hosobe, H.: An interval-based SAT modulo ODE solver for model checking nonlinear hybrid systems. International Journal on Software Tools for Technology Transfer 13(5), 449–461 (2011)
8. Kaynar, D.K., Lynch, N., Segala, R., Vaandrager, F.: Timed I/O automata: A mathematical framework for modeling and analyzing real-time systems. In: RTSS'03. pp. 166– (2003)
9. Kerner, E.H.: Universal formats for nonlinear ordinary differential systems. Journal of Mathematical Physics 22(7), 1366–1371 (1981)
10. Kong, H., He, F., Song, X., Hung, W., Gu, M.: Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In: CAV'13. pp. 242–257 (2013)
11. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT'11. pp. 97–106 (2011)
12. Liu, J., Zhan, N., Zhao, H.: Automatically discovering relaxed lyapunov functions for polynomial dynamical systems. Mathematics in Computer Science 6(4), 395–408 (2012)
13. Löfberg, J.: YALMIP : A toolbox for modeling and optimization in MATLAB. In: Proc. of the CACSD Conference (2004)
14. Makino, K., Berz, M.: Taylor models and other validated functional inclusion methods. International Journal of Pure and Applied Mathematics 4(4), 379–456 (2003)
15. Mover, S., Cimatti, A., Tiwari, A., Tonetta, S.: Time-aware relational abstractions for hybrid systems. In: EMSOFT'13. pp. 1–10 (2013)
16. Papachristodoulou, A., Prajna, S.: On the construction of Lyapunov functions using the sum of squares decomposition. In: CDC'02. vol. 3, pp. 3482–3487 (2002)
17. Papachristodoulou, A., Prajna, S.: Analysis of non-polynomial systems using the sum of squares decomposition. In: Positive Polynomials in Control, Lecture Notes in Control and Information Science, vol. 312, pp. 23–43 (2005)
18. Platzer, A., Clarke, E.M.: Computing differential invariants of hybrid systems as fixedpoints. In: CAV'08. LNCS, vol. 5123, pp. 176–189 (2008)
19. Prabhakar, P., Dullerud, G., Viswanathan, M.: Pre-orders for reasoning about stability. In: HSCC'12. pp. 197–206 (2012)
20. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: HSCC'04. LNCS, vol. 2993, pp. 477–492. Springer (2004)
21. Sankaranarayanan, S.: Automatic abstraction of non-linear systems using change of variables transformations. In: HSCC'11 (2011)
22. Sankaranarayanan, S.: Change-of-bases abstractions for non-linear systems. CoRR abs/1204.4347 (2012)
23. Sankaranarayanan, S., Tiwari, A.: Relational abstractions for continuous and hybrid systems. In: CAV'11. pp. 686–702 (2011)
24. Savageau, M.A., Voit, E.O.: Recasting nonlinear differential equations as s-systems: a canonical nonlinear form. Mathematical Biosciences 87(1), 83 – 115 (1987)
25. Tarski, A.: A Decision Method for Elementary Algebra and Geometry. University of California Press, Berkeley (May 1951)
26. Toh, K.C., Todd, M., Tütüncü, R.H.: SDPT3 – a MATLAB software package for semidefinite programming. Optimization Methods and Software 11, 545–581 (1999)

A Proofs

Proof for Theorem 4 is as follows:

Proof. Obviously, $\Theta(\Xi) \widehat{=} \Xi \wedge \mathbf{v} = \Gamma(\mathbf{x})$ is a subset of $\Xi[\mathbf{v}/\Gamma(x)]$, and $\Theta(\mathcal{D}) \widehat{=} \mathcal{D} \wedge \mathbf{v} = \Gamma(\mathbf{x})$ is a subset of $\mathcal{D}[\mathbf{v}/\Gamma(x)]$. Therefore the first condition of Definition 3 is satisfied.

Now, we only need to prove that the second condition in Definition 3 holds. Suppose $\mathbf{x}_0 \in \Xi_{\mathbf{x}}$, and $T > 0$ such that there is a unique trajectory $\tau_{\mathbf{x}}(\mathbf{x}_0, t)$ of $\mathcal{S}_{\mathbf{x}}$ (denoted by $\tau_{\mathbf{x}}(t)$ for short) over $[0, T)$, starting from $\mathbf{x}_0$. Let $\mathbf{v}_0 = \Gamma(\mathbf{x}_0)$, and thus $\Theta(\mathbf{x}_0) = (\mathbf{x}_0, \mathbf{v}_0) \in \Xi[\mathbf{v}/\Gamma(\mathbf{x})]$. By definition of Θ, we have $\Theta(\tau_{\mathbf{x}}(t)) = (\tau_{\mathbf{x}}(t), \Gamma(\tau_{\mathbf{x}}(t)))$ for $t \in [0, T)$, denoted by $\tau_{(\mathbf{x},\mathbf{v})}(t)$. Then we have to prove that $\tau_{(\mathbf{x},\mathbf{v})}(t)$ is a trajectory of $\Gamma(\mathcal{S}_{\mathbf{x}})$ on $[0, T)$, starting from $(\mathbf{x}_0, \mathbf{v}_0) \in \Xi[\mathbf{v}/\Gamma(\mathbf{x})]$. To this end, it suffices to prove $\frac{d\tau_{(\mathbf{x},\mathbf{v})}(t)}{dt} = (\mathbf{f}[\mathbf{v}/\Gamma(\mathbf{x})], \dot{\Gamma(\mathbf{x})}[\mathbf{v}/\Gamma(\mathbf{x})])$ for $t \in [0, T)$, because it is assumed the EDS satisfies the local Lipschitz condition. It is obvious according to the chain rule and the equalities in Γ. □

Proof for Theorem 5 is as follows:

Proof. By Theorem 4, it is straightforward. □

Proof for Theorem 6 is as follows:

Proof. For any $\mathbf{x}_0 \in \Theta^{-1}(I)$, obviously, $\mathbf{y}_0 := (\mathbf{x}_0, \Gamma(\mathbf{x}_0)) \in I$. Consider a trajectory $\tau(\mathbf{x}_0, t)$ of $\mathcal{S}_{\mathbf{x}}$, by Theorem 4, $\tau(\mathbf{y}_0, t) = \Theta(\tau(\mathbf{x}_0, t)) = (\tau(\mathbf{x}_0, t), \Gamma(\tau(\mathbf{x}_0, t)))$ is a trajectory of $\Gamma(\mathcal{S}_{\mathbf{x}})$. As I is a continuous invariant of $\Gamma(\mathcal{S}_{\mathbf{x}})$, for any $T \geqslant 0$, it follows that $\tau(\mathbf{y}_0, t) = (\tau(\mathbf{x}_0, t), \Gamma(\tau(\mathbf{x}_0, t))) \in I$ for all $t \in T$ if $\tau(\mathbf{y}_0, t) \in \Theta(\mathcal{D})$ for all $t \in T$. Therefore, $\tau(\mathbf{x}_0, t) = \Theta^{-1}(\tau(\mathbf{y}_0, t)) \in \Theta^{-1}(I)$ for all $t \in T$ if $\tau(\mathbf{x}_0, t) \in \mathcal{D}$ for all $t \in T$. This implies $\Theta^{-1}(I)$ is a CI of $\mathcal{S}_{\mathbf{x}}$ by Definition 2. □

Proof for Corollary 7 is as follows:

Proof. By Theorem 6, $\Theta^{-1}(I) = \{\mathbf{x} \mid (\mathbf{x}, \Gamma(\mathbf{x})) \in I\}$ is a CI of $\mathcal{S}_{\mathbf{x}}$. Now, we only need to prove $I^* = \Theta^{-1}(I)$. For the forward direction, for any $\mathbf{x} \in I^*$, we have

$$\bigvee_{k=1}^{K} \bigwedge_{j=1}^{J_k} p_{kj}(X, \Gamma(X)) \triangleright 0. \tag{13}$$

So, $(\mathbf{x}, \Gamma(\mathbf{x})) \in I$, which implies $\mathbf{x} \in \Theta^{-1}(I)$. For the backward direction, for any $\mathbf{x} \in \Theta^{-1}(I)$, obviously $(\mathbf{x}, \Gamma(\mathbf{x})) \in I$, i.e., (13) holds. Hence $\mathbf{x} \in I^*$. This completes the proof. □

Proof for Theorem 8 is as follows:

Proof. Let $\tau(\mathbf{x}_0, t)$ be a trajectory of $\mathcal{S}_{\mathbf{x}}$ starting from $\mathbf{x}_0$ and $\tau(\mathbf{y}_0, t)$ be a trajectory of $\Gamma(\mathcal{S}_{\mathbf{x}})$ starting from $\mathbf{y}_0$. By Theorem 4, if $\mathbf{y}_0 = \Theta(\mathbf{x}_0)$ then $\tau(\mathbf{y}_0, t) = \Theta(\tau(\mathbf{x}_0, t)) = (\tau(\mathbf{x}_0, t), \Gamma(\tau(\mathbf{x}_0, t)))$ for all t because of the assumption of the local Lipschhitz condition. W.l.o.g., assume that $\mathbf{y}_e$ is an asymptotic stable point of $\Gamma(\mathcal{S}_{\mathbf{x}})$. Thus, there is some $\delta > 0$ such that for $\forall \mathbf{y}_0$

$$\|\mathbf{y}_0 - \mathbf{y}_e\| < \delta \implies \lim_{t \to \infty} \|\tau(\mathbf{y}_0, t) - \mathbf{y}_e\| = 0,$$

where, $\|\cdot\|$ is the Euclidean norm (L^2-norm), given by $\|\mathbf{u}\| := \sqrt{u_1^2 + \cdots + u_n^2}$ for $\mathbf{u} \in \mathbb{R}^n$. Let $D_\delta = \{\mathbf{y} \mid \|\mathbf{y} - \mathbf{y}_e\| < \delta\}$. Since Θ is continuous, $\Theta^{-1}(D_\delta)$ is open. As $\mathbf{x}_e \in \Theta^{-1}(D_\delta)$, there must be some $\delta_\mathbf{x} > 0$ such that

$$\{(\mathbf{x}, \Gamma(\mathbf{x})) \mid \|\mathbf{x} - \mathbf{x}_e\| < \delta_\mathbf{x}\} \subset \{\mathbf{y} \mid \|\mathbf{y} - \mathbf{y}_e\| < \delta\}.$$

Therefore, for such $\delta_\mathbf{x}$, it follows $\|\mathbf{x}_0 - \mathbf{x}_e\| < \delta_\mathbf{x} \implies \|\mathbf{y}_0 - \mathbf{y}_e\| < \delta$. Furthermore,

$$\|\mathbf{y}_0 - \mathbf{y}_e\| < \delta \implies \lim_{t\to\infty} \|\varsigma(\mathbf{x}_0, t) - \mathbf{x}_e\| \leqslant \lim_{t\to\infty} \|\tau(\mathbf{y}_0, t) - \mathbf{y}_e\| = 0.$$

So, $\|\mathbf{x}_0 - \mathbf{x}_e\| < \delta_\mathbf{x} \implies \lim_{t\to\infty} \|\varsigma(\mathbf{x}_0, t) - \mathbf{x}_e\| = 0$. □

Proof for Theorem 10 is as follows:

Proof. We prove this theorem by contradiction. Suppose I is an invariant of $\mathcal{H}_\mathbf{y}$, but $\Theta^{-1}(I)$ is not an invariant of $\mathcal{H}_\mathbf{x}$. Thus, there is some $(q_0, \mathbf{x}_0) \in \Theta^{-1}(I)$ and $t^* > 0$ such that the trajectory $\tau((q_0, \mathbf{x}_0), t)$ of $\mathcal{H}_\mathbf{x}$ will go out of $\Theta^{-1}(I)$ at t^*, namely, $\tau((q_0, \mathbf{x}_0), t^*) \notin \Theta^{-1}(I)$. On the other hand, $\Theta(\tau((q_0, \mathbf{x}_0), t))$ is a trajectory of $\mathcal{H}_\mathbf{y}$ starting with $\Theta(q_0, \mathbf{x}_0)$, say $\sigma(\Theta(q_0, \mathbf{x}_0), t)$. As $\Theta(q_0, \mathbf{x}_0) \in I$, $\sigma(\Theta(q_0, \mathbf{x}_0), t) \in I$ for all t and thus $\Theta(\tau((q_0, \mathbf{x}_0), t)) \in I$. Therefore, $\Theta(\tau((q_0, \mathbf{x}_0), t^*)) \in I$, which contradicts to $\tau((q_0, \mathbf{x}_0), t^*) \notin \Theta^{-1}(I)$. □

B Taylor Approximation

The following picture illustrates how $v_1 = \sin(x)$ and $v_2 = e^{-x}$ are approximated over $[-2, 2]$.

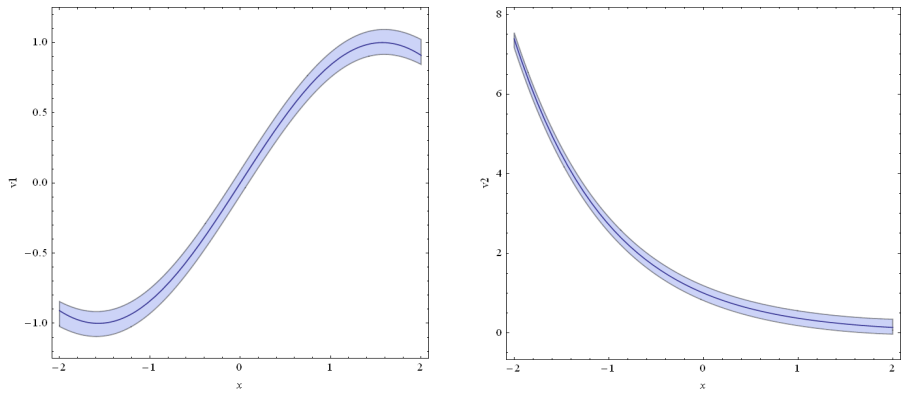

Fig. 8. Taylor model approximations of domain in Section 6.1.

The left part of Figure 8 shows that $p_1(x) + l_1 \leqslant v_1 \leqslant p_1(x) + u_1$ with $l_1 = -0.08888888888890931$, $u_1 = 0.08888888888890931$ and

$$p_1(x) = 2(0.5x) - 1.333333333333333(0.5x)^3 + 0.2666666666666667(0.5x)^5 \ .$$

The right part of Figure 8 shows that $p_2(x) + l_2 \leqslant v_2 \leqslant p_2(x) + u_2$ with $l_2 = -0.1876585675919477$, $u_2 = 0.1876585675919477$ and

$$p_2(x) = 1 - 2(0.5x) + 2(0.5x)^2 - 1.333333333333333(0.5x)^3 + 0.6666666666666666(0.5x)^4 \\ -0.2666666666666667(0.5x)^5 + 0.08888888888888889(0.5x)^6 \ .$$

C Invariants

Synthesized Invariant for Section 6.1

$$\begin{aligned} Inv := & -4.339475565 + 0.2502x - 2.1577y + 1.1323x^2 - 3.0180xy + 3.6947y^2 + \\ & 2.2921x^3 - 2.9991x^2y + 6.3034xy^2 + 0.6624y^3 + 3.7675x^4 + 0.6817x^3y + \\ & 0.4972x^2y^2 - 2.9627xy^3 + 0.3242y^4 - 1.4501x^5 + 0.1213x^4y + 0.2364x^3y^2 \\ & +1.4077x^2y^3 - 1.6868xy^4 + 3.8198y^5 \leqslant 0 \end{aligned}$$

Synthesized Invariant for Section 6.2

$$\begin{aligned} Inv := & -2.028911595 - 0.0400u_1 - 0.0400u_2 + 0.1263u_3 - 0.0043u_1^2 - 0.0246u_1u_2 \\ & -0.0202u_2^2 + 0.0168u_1u_3 + 2.8240u_3^2 - 0.0023u_1^3 - 0.0134u_1^2u_2 - 0.0347u_1u_2^2 \\ & -0.0237u_2^3 + 0.0078u_1^2u_3 + 0.0098u_1u_2u_3 - 0.0129u_2^2u_3 + 0.0209u_1u_3^2 + \\ & 0.0215u_2u_3^2 + 0.0340u_3^3 + 0.0124u_1^3u_2 + 0.0238u_1^2u_2^2 + 0.0180u_1u_2^3 + \\ & 0.0063u_2^4 + 0.0028u_1^3u_3 + 0.0029u_1^2u_2u_3 - 0.0157u_1u_2^2u_3 - 0.0094u_2^3u_3 + \\ & 0.0429u_1^2u_3^2 + 0.0754u_1u_2u_3^2 + 0.0311u_2^2u_3^2 + 0.0221u_1u_3^3 + 0.0285u_2u_3^3 \\ & +0.0988u_3^4 \leqslant 0 \end{aligned}$$

Synthesized Invariant for Section 6.3

$$\begin{aligned} Inv := & 2.923220751 + 1.8001v - 0.3617a + 0.3125t + 0.1253v^2 + 0.4761va + 0.6071a^2 - \\ & 0.0215vt + 0.0411at + 0.2134t^2 - 0.0139v^3 + 0.1290v^2a + 0.1632va^2 - 0.0794a^3 \\ & -0.1704v^2t - 0.5940vat - 0.8830a^2t - 0.0601vt^2 - 0.1362at^2 - 0.1348t^3 + \\ & 0.0682v^4 + 0.0507v^3a + 0.0241v^2a^2 - 0.0111va^3 + 0.0253a^4 + 0.2461v^3t + \\ & 0.2375v^2at - 0.1857va^2t + 0.0081a^3t + 0.2070v^2t^2 + 0.1207vat^2 + 0.6008a^2t^2 \\ & -0.0652vt^3 + 0.0524at^3 + 0.0862t^4 - 0.0901v^4t - 0.2519v^3at - 0.1448v^2a^2t \\ & -0.0286va^3t - 0.0686a^4t + 0.2320v^3t^2 - 0.5085v^2at^2 - 0.1506va^2t^2 - 0.0668a^3t^2 \\ & +0.0234v^2t^3 - 0.1743vat^3 - 0.0975a^2t^3 + 0.2561vt^4 - 0.2055at^4 - 0.4458t^5 + \\ & 0.1576v^4t^2 + 0.0656v^3t^3 + 0.2481v^2t^4 + 0.1100vt^5 + 0.3695t^6 \leqslant 0 \end{aligned}$$

D Algorithms

Algorithm 1: Reducing an elementary expression to a polynomial one (**VT**(*expr*, *eqs*))

Require: An elementary expression *expr* and a set of equations *eqs* as input
Ensure: The returned expression is polynomial, and equals to the input expression in the context of equations *eqs*

1: **if** $expr = c$ or $expr = x$ **then**
2: **return** $(expr, eqs)$;
3: **else if** $expr = \frac{expr_1}{expr_2}$ **then**
4: $(expr_2, eqs) = \mathbf{VT}(expr_2, eqs)$; **return** $\mathbf{VT}(expr_1 * newVar, eqs.\mathbf{add}(newVar, \frac{1}{expr_2}))$;
5: **else if** $expr = expr_1^{\frac{n_1}{n_2}}$ **then**
6: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; **return** $(newVar^{n_1}, eqs.\mathbf{add}(newVar, expr_1^{\frac{1}{n_2}}))$;
7: **else if** $expr = e^{expr_1}$ **then**
8: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; **return** $(newVar, eqs.\mathbf{add}(newVar, e^{expr_1}))$;
9: **else if** $expr = \ln(expr_1)$ **then**
10: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; **return** $(newVar, eqs.\mathbf{add}(newVar, \ln(expr_1)))$;
11: **else if** $expr = \sin(expr_1)$ **then**
12: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; **return** $(newVar, eqs.\mathbf{add}(newVar, \sin(expr_1)))$;
13: **else if** $expr = \cos(expr_1)$ **then**
14: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; **return** $(newVar, eqs.\mathbf{add}(newVar, \cos(expr_1)))$;
15: **else if** $expr = expr_1 + expr_1$ **then**
16: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; $(expr_2, eqs) = \mathbf{VT}(expr_2, eqs)$;
17: **return** $(expr_1 + expr_2, eqs)$;
18: **else**
19: $(expr_1, eqs) = \mathbf{VT}(expr_1, eqs)$; $(expr_2, eqs) = \mathbf{VT}(expr_2, eqs)$;
20: **return** $(expr_1 \times expr_2, eqs)$;
21: **end if**

In Algorithm 1, *newVar* denotes a fresh variable, and *eqs* records the replacements during the variable transformation.

In Algorithm 2, **op**, **left**, and **right** returns the outermost operation, and its left and right operands for a given expression, respectively. **left** and **right** return the operand in case the outmost operation is one ary; *newVar* denotes a fresh variable. Algorithm 2 must terminate, because the number of elements of *eqs* can only increase finite times, obviously, no more than the number of the subexpressions of the EDS.

In algorithm 3, $\mathbf{omExp}(ode)$ returns the set of the outmost expressions of ode, and **VT** and **U** call Algorithm 1 and 2, respectively.

In Algorithm 4, $\mathbf{omExp}(form)$ returns the set of the outmost expressions of formula *form*, and **VT** and **TransODEs** call Algorithm 1 and 3, respectively.

Algorithm 2: Updating the dynamical system according to the replacement equations *eqs* (**U**(*odes*, *eqs*))

Require: Polynomial differential equations $odes$ and a set of equations *eqs* as input (where all expressions in *eqs* are polynomial except the outermost operator)
Ensure: The resulting polynomial differential equations simulate the initial $odes$ and *eqs*

for $(var, expr)$ **in** *eqs* **do**
  if $expr = \frac{1}{expr_2}$ **then**
    $odes$.**add**$(var, -var^2 * \dot{expr}_2)$;
  else if $expr = expr_1^{\frac{1}{n_2}}$ **then**
    eqs.**add**$(newVar, 1/expr)$; $odes$.**add**$(var, \frac{1}{n_2} * newVar^{n_2-1} * \dot{expr}_1)$;
  else if $expr = e^{expr_1}$ **then**
    $odes$.**add**$(var, var * \dot{expr}_1)$;
  else if $expr = \ln(expr_1)$ **then**
    eqs.**add**$(newVar, \frac{1}{expr_1})$; $odes$.**add**$(var, newVar * \dot{expr}_1)$;
  else if $expr = \sin(expr_1)$ **then**
    eqs.**add**$(newVar, \cos(expr_1))$; $odes$.**add**$(var, newVar * \dot{expr}_1)$;
  else if $expr = \cos(expr_1)$ **then**
    eqs.**add**$(newVar, \sin(expr_1))$; $odes$.**add**$(var, -newVar * \dot{expr}_1)$;
  else
    The algorithm should not run this breach;
  end if
end for
return $odes$;

Algorithm 3: Transforming elementary ODEs to polynomial ODEs (**TransEODEs**(*odes*, *eqs*))

Require: ODEs *odes* and a list of replacement equations *eqs* as input
Ensure: The resulting ODEs are polynomial

for ode **in** $odes$ **do**
  for exp **in omExp**(ode) **do**
    (*exp*, *eqs*) = **VT**(*exp*, *eqs*);
  end for
end for
return U(*odes*, *eqs*);

Algorithm 4: Transforming elementary hybrid systems (**TransEHS**(*hs*))

Require: An elementary hybrid system *hs* as input
Ensure: The resulting hybrid system is a PHS which simulates the input EHS

Set eqs to empty;
for $mode$ **in** hs **do**
  for exp **in omExp**$(mode.init)$ **do**
    (exp, eqs) = **VT**(exp, eqs);
  end for
  for exp **in omExp**$(mode.domain)$ **do**
    (exp, eqs) = **VT**(exp, eqs);
  end for
  for exp **in omExp**$(mode.guard)$ **do**
    (exp, eqs) = **VT**(exp, eqs);
  end for
  for exp **in omExp**$(mode.reset)$ **do**
    (exp, eqs) = **VT**$(exp, eqs.expr)$;
  end for
  mode.odes = **TransEODEs**(*odes*, *eqs*);
end for
return *hs*;