Abstraction of Elementary Hybrid Systems by Variable Transformation*

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

[1] Chongqing Key Lab. of Automated Reasoning and Cognition, Chongqing Institute of Green and Intelligent Technology, CAS
{liujiang, zhaohengjun}@cigit.ac.cn
[2] State Key Lab. of Computer Science, Institute of Software, CAS
{znj,zoul}@ios.ac.cn

Abstract. Elementary hybrid systems (EHSs) are those hybrid systems (HSs) containing elementary functions such as exp, ln, sin, cos, etc. EHSs are very common in practice, especially in safety-critical domains. Due to the non-polynomial expressions which lead to undecidable arithmetic, verification of EHSs is very hard. Existing approaches based on partition of the state space or overapproximation of reachable sets suffer from state space explosion or inflation of numerical errors. In this paper, we propose a symbolic abstraction approach that reduces EHSs to polynomial hybrid systems (PHSs), by replacing all non-polynomial terms with newly introduced variables. Thus the verification of EHSs is reduced to the one of PHSs, enabling us to apply all the well-established verification techniques and tools for PHSs to EHSs. In this way, it is possible to avoid the limitations of many existing methods. We illustrate the abstraction approach and its application in safety verification of EHSs by several real world examples.

Keywords: hybrid system, abstraction, elementary function, invariant, verification

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 to carry out highly complex and often critical functions. 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 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 design.

* This work has been partially supported by "973 Program" under grant No. 2014CB340701, NSFC grants (Nos. 91118007, 91418204 and 61202131), the CAS/SAFEA International Partnership Program for Creative Research Teams, CAS Western Light Program and CAS Youth Innovation Promotion Association (No. 2015315), Chongqing Science and Technology Commission projects cstc2012ggB40004, cstc2014jcsfglyjs0005 and cstc2014zktjccxyyB0031.

In practice, it is very common to model complex physical environments by ordinary differential equations (ODEs) with elementary 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 such HSs elementary HSs (EHSs). As elementary expressions usually lead to undecidable arithmetic, the verification of EHSs becomes very hard, even intractable. Existing methods that deal with EHS verification include the level-set method [25], the hybridization method [4, 15], the gridding-based abstraction refinement method [31], the interval SMT solver-based method [9, 8], the Taylor model-based flowpipe approximation method [5], and so on. These methods rely either on iterative partition of the state space or on iterative computation of approximate reachable sets, which can quickly lead to explosion of state numbers or inflation of numerical errors. Moreover, most of the above mentioned methods can only do bounded model checking (BMC).

As an alternative, the constraint-based approach verifies the safety property of an HS by solving corresponding constraints symbolically or numerically, to discover a barrier (inductive invariant) that separates the reachable set from the unsafe region, which avoids exhaustive gridding or brute-force computation, and can thus overcome the limitations of the above mentioned methods. However, this method has mainly been applied to verification of polynomial hybrid systems (PHSs) [35, 29, 28, 12, 20]. Although ideas about generating invariants for EHSs appeared in [28, 10], they were talked about in an ad hoc way. In [34], the author proposed a change-of-bases method to transform EHSs to PHSs, even to linear systems, but the success depends on the choice of the set of basis functions, and therefore does not apply to general EHSs.

In this paper, we investigate symbolic abstraction of general EHSs to PHSs, by extending [34] with early works on polynomialization of elementary ODEs [16, 36]. Herein the abstraction is accomplished by introducing new variables to replace the non-polynomial terms. With the substitution, flows, guards and other components of the EHSs are transformed according to the chain rule of differentiation, or by the over-approximation methods proposed in the paper, so that for any trajectory of the EHSs, there always exists a corresponding trajectory of the reduced PHSs. Besides, such abstraction preserves (inductive) invariant sets, that is, any (inductive) invariant of the over-approximating PHS corresponds to an (inductive) invariant of the original EHS. Therefore, verification of the EHSs is naturally reduced to the one of the reduced PHSs. This will be shown by several real world verification problems.

The proposed abstraction applies to general EHSs. The benefit of the proposed abstraction is that it enables all the well-established verification techniques and tools for PHSs, especially the constraint-based approaches such as DAL [27] and SOS [29, 18], to be applied to EHSs, and thus provides the possibility of avoiding such limitations as error inflation, state space explosion and boundedness for existing EHS verification methods. A by-product is that it also provides the possibility of generating invariants with elementary functions for PHSs, thus enhancing the power of existing PHS verification methods. In short, the proposed abstraction method can be a good alternative or complement to existing approaches.

Related Work. This work is most closely related to [34] and [16]. The abstraction in this paper is performed by systematic augmentation of the original system rather than change-of-bases, thus essentially different from [34] and generally applicable. Compared to [16], this paper gives a clearer reduction procedure for elementary ODEs and discusses the extension to hybrid systems. It was proved in [30] that safety verification of nonlinear hybrid systems is quasi-semidecidable, but to find efficient verification algorithms remains an open problem. An approximation technique for abstracting nonlinear hybrid systems to PHSs based on Taylor polynomial was proposed in [19], which requires the ODEs to have closed-form solutions to abstract the continuous flow transitions. In the recent work [7], following the line of [39], the author proposed predicate-based abstraction of general nonlinear hybrid systems by using the automated theorem prover MetiTarski [1]. In [26], the authors adopted similar recasting techniques to ours for stability analysis of non-polynomial systems. Regarding non-polynomial invariants for polynomial continuous or hybrid systems, [32] presented the first method for generating transcendental invariants using formal power series, while the more recent work [11] proposed a Darboux Polynomial-based method. Both [32] and [11] can only find non-polynomial invariants of limited forms.

Paper Organization. The rest of the paper is organized as follows. We briefly review some basic notions about hybrid systems and the theory of abstraction for hybrid systems in Section 2. Section 3 is devoted to the transformation from EDSs to PDSs, and from EHSs to PHSs. Section 4 discusses how to use the proposed abstraction approach for safety verification of EHSs. Section 5 concludes this paper.

2 Preliminary

In this section, we briefly introduce the basic knowledge of hybrid systems and define what we call *elementary hybrid systems*. Besides, we also recall the basic theory of abstraction for hybrid systems originally developed in [33, 34].

Throughout this paper, we use $\mathbb{N}, \mathbb{Q}, \mathbb{R}$ to denote the set of *natural*, *rational* and *real* numbers respectively. Given a set A, the power set of A is denoted by 2^A, and the Cartesian product of n duplicates of A is denoted by A^n; for instance, $\mathbb{R}^n$ stands for the n-dimensional Euclidean space. A vector element $(a_1, a_2, \ldots, a_n) \in A^n$ is usually abbreviated by a boldface letter $\mathbf{a}$ when its dimension is clear from the context.

2.1 Elementary Continuous and Hybrid Systems

A continuous dynamical system (CDS) is modeled by first-order autonomous ordinary differential equations (ODEs)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ and $\mathbf{f} : U \to \mathbb{R}^n$ is a vector function, called a vector field, defined on an open set $U \subseteq \mathbb{R}^n$. If $\mathbf{f}$ satisfies the *local Lipschitz condition* [17], then for any $\mathbf{x}_0 \in U$, there exists a unique differentiable vector function $\mathbf{x}(t) : (a, b) \to U$, where (a, b) is an open interval containing 0, such that $\mathbf{x}(0) = \mathbf{x}_0$ and the derivative of $\mathbf{x}(t)$ w.r.t. t satisfies $\forall t \in (a, b). \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}(t))$. Such $\mathbf{x}(t)$ is called the *solution* to (1) with initial value $\mathbf{x}_0$, or the *trajectory* of (1) starting from $\mathbf{x}_0$.

In many contexts, a CDS $\mathcal{C}$ may be equipped with an initial set $\varXi$ and a domain D, represented as a triple $\mathcal{C} \mathrel{\widehat{=}} (\varXi, \mathbf{f}, D)$.[3] If $\mathbf{f}$ is defined on $U \subseteq \mathbb{R}^n$, then $\varXi$ and D should

[3] In this paper, the symbol $\widehat{=}$ is interpreted as "defined as".

satisfy $\varXi \subseteq D \subseteq U$. In what follows, all CDSs will refer to the triple form unless otherwise stated. Hybrid systems (HSs) are those systems that exhibit both continuous evolutions and discrete transitions. A popular model of HSs is *hybrid automata* [2, 13].

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

- *$Q = \{q_1, \ldots, q_m\}$ is a finite set of 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 $\mathbb{R}^n$;*
- *$f : Q \to (U_q \to \mathbb{R}^n)$ assigns to each mode $q \in Q$ a locally Lipschitz continuous vector field $\mathbf{f}_q$ defined on an open set $U_q \subseteq \mathbb{R}^n$;*
- *D assigns to each mode $q \in Q$ a domain $D_q \subseteq U_q$;*
- *$E \subseteq Q \times Q$ is a finite set of discrete transitions;*
- *G assigns to each transition $e \in E$ a guard $G_e \subseteq \mathbb{R}^n$;*
- *R assigns to each transition $e \in E$ a set-valued reset function R_e: $G_e \to 2^{\mathbb{R}^n}$;*
- *$\varXi$ assigns to each $q \in Q$ a set of initial states $\varXi_q \subseteq D_q$.*

Actually an HA can be regarded as a composition of a finite set of CDSs $\mathcal{C}_q \widehat{=} (\varXi_q, \mathbf{f}_q, D_q)$ for $q \in Q$, together with the set of transition relations specified by (G_e, R_e) for $e \in E$. Conversely, any CDS can be regarded as a special HA with a single mode and without discrete transitions.

In this paper, we consider the class of HSs that can be defined by multivariate *elementary* functions given by the following grammar:

$$f, g ::= c \mid x \mid f + g \mid f - g \mid f \times g \mid \frac{f}{g} \mid f^a \mid e^f \mid \ln(f) \mid \sin(f) \mid \cos(f) \qquad (2)$$

where $c \in \mathbb{R}$ is any real constant, $a \in \mathbb{Q}$ is any rational constant, and x can be any variable from the set of real-valued variables $\{x_1, \ldots, x_n\}$. In particular, the set of functions constructed only by the first 5 constructs are multivariate *polynomials* in $x_1, x_2, \ldots, x_n$.

The limitation of elementary functions to grammar (2) is not essential. For example, tangent and cotangent functions $\tan(f), \cot(f)$ can be easily defined. Besides, the presented approach in this paper is also applicable to other elementary functions not mentioned above, such as inverse trigonometric functions $\arcsin(f), \arccos(f)$, etc.

Definition 2 (Elementary and Polynomial HSs). *An HS or a CDS is called* elementary *(resp.* polynomial*) if it can be expressed by* elementary *(resp.* polynomial*) functions together with relational symbols $\geqslant, >, \leqslant, <, =, \neq$ and Boolean connectives $\wedge, \vee, \neg, \longrightarrow, \longleftrightarrow$, etc.*

Note that in Definition 2, the presented symbol set is complete but not minimal. Elementary (resp. polynomial) HSs or CDSs will be denoted by EHSs or EDSs (resp. PHSs or PDSs) for short.

Example 1 (Bouncing Ball). Figure 1 depicts the HA model of a bouncing ball on a sine-waved surface, adapted from a similar one in [14]. The motion of the ball stays in

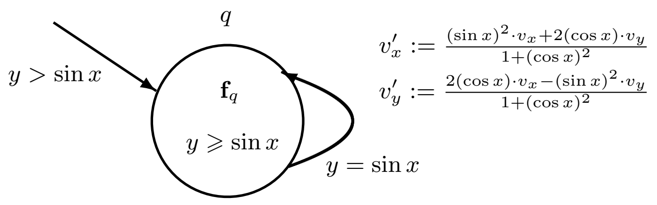

q
$y > \sin x$
$\mathbf{f}_q$
$y \geqslant \sin x$
$y = \sin x$
$v_x' := \frac{(\sin x)^2 \cdot v_x + 2(\cos x)\cdot v_y}{1+(\cos x)^2}$
$v_y' := \frac{2(\cos x)\cdot v_x - (\sin x)^2 \cdot v_y}{1+(\cos x)^2}$

Fig. 1. The HA model of a bouncing ball on a sine-waved surface

the two-dimensional x-y plane, with x denoting the horizontal position, y denoting the height, and v_x and v_y denoting the velocity along the two directions respectively. When the ball hits the surface given by the sine wave $y = \sin x$, its velocity changes instantaneously according to the law of perfectly elastic collision. Here, using the notation of Definition 1, we have $Q = \{q\}$, $X = \{x, y, v_x, v_y\}$, $\mathbf{f}_q$ defines the ODE

$$\begin{cases} \dot{x} &= v_x \\ \dot{y} &= v_y \\ \dot{v}_x &= 0 \\ \dot{v}_y &= -9.8 \end{cases},$$

$D_q \widehat{=} y \geqslant \sin x$, $E = \{e\}$ with $e = (q, q)$, $G_e \widehat{=} y = \sin x$, $\Xi_q \widehat{=} y > \sin x$, and $R_e(x, y, v_x, v_y) \widehat{=} \{(x, y, v_x', v_y')\}$ with v_x', v_y' shown in the figure.

2.2 Semantics of Hybrid Systems

Given an HA $\mathcal{H}$, denote the state space of $\mathcal{H}$ by $\mathbb{H} \widehat{=} Q \times \mathbb{R}^n$, the domain of $\mathcal{H}$ by $D_{\mathcal{H}} \widehat{=} \bigcup_{q\in Q}(\{q\} \times D_q)$, and the set of all initial states 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 *reachable* states of $\mathcal{H}$.

Definition 3 (Reachable Set). *Given an HA $\mathcal{H}$, the* reachable set *of $\mathcal{H}$, denoted by $\mathcal{R}_{\mathcal{H}}$, consists of such $(q, \mathbf{x}) \in \mathbb{H}$ for which there exists a finite sequence*

$$(q_0, \mathbf{x}_0), (q_1, \mathbf{x}_1), \ldots, (q_l, \mathbf{x}_l)$$

such that $(q_0, \mathbf{x}_0) \in \Xi_{\mathcal{H}}$, $(q_l, \mathbf{x}_l) = (q, \mathbf{x})$, and for any $0 \leqslant i \leqslant l-1$, one of the following two conditions holds:

– *(Discrete Jump): $e = (q_i, q_{i+1}) \in E$, $\mathbf{x}_i \in G_e$ and $\mathbf{x}_{i+1} \in R_e(\mathbf{x}_i)$; or*
– *(Continuous Evolution): $q_i = q_{i+1}$, and there exists a $\delta \geqslant 0$ s.t. the trajectory $\mathbf{x}(t)$ of $\dot{\mathbf{x}} = \mathbf{f}_{q_i}$ starting from $\mathbf{x}_i$ satisfies*
  - *$\mathbf{x}(t) \in D_{q_i}$ for all $t \in [0, \delta]$; and*
  - *$\mathbf{x}(\delta) = \mathbf{x}_{i+1}$.*

Exact computation of reachable sets of hybrid systems is generally an intractable problem. For verification of safety properties, appropriate over-approximations of reachable sets will suffice.

Definition 4 (Invariant). *Given an HA $\mathcal{H}$, a set $\mathcal{I} \widehat{=} \bigcup_{q\in Q}(\{q\} \times I_q) \subseteq \mathbb{H}$ is called an* invariant *of $\mathcal{H}$, if $\mathcal{I}$ is a superset of the reachable set $\mathcal{R}_{\mathcal{H}}$, i.e. $\mathcal{R}_{\mathcal{H}} \subseteq \mathcal{I}$.*

Definition 5 (Inductive Invariant). *Given an HA $\mathcal{H}$, a set $\mathcal{I} \mathrel{\widehat{=}} \bigcup_{q\in Q}(\{q\} \times I_q) \subseteq \mathbb{H}$ is called an* inductive invariant *of $\mathcal{H}$, if $\mathcal{I}$ satisfies the following conditions:*

- *$\Xi_q \subseteq I_q$ for all $q \in Q$;*
- *for any $e = (q, q') \in E$, if $\mathbf{x} \in I_q \cap G_e$, then $R_e(\mathbf{x}) \subseteq I_{q'}$;*
- *for any $q \in Q$ and any $\mathbf{x}_0 \in I_q$, if $\mathbf{x}(t)$ is the trajectory of $\dot{\mathbf{x}} = \mathbf{f}_q$ starting from $\mathbf{x}_0$, and there exists $T \geqslant 0$ s.t. $\mathbf{x}(t) \in D_q$ for all $t \in [0, T]$, then $\mathbf{x}(T) \in I_q$.*

It is easy to check that any inductive invariant is also an invariant.

2.3 Abstraction of Hybrid Systems

We next briefly introduce the kind of abstraction for HSs proposed in [33, 34] and the significant properties about such an abstraction.

In what follows, to distinguish between the dimensions of an HS and its abstraction, we will annotate an HS $\mathcal{H}$ (a CDS $\mathcal{C}$) with the vector of its continuous state variables $\mathbf{x}$ as $\mathcal{H}_{\mathbf{x}}$ ($\mathcal{C}_{\mathbf{x}}$). We use $|\mathbf{x}|$ to denote the dimension of $\mathbf{x}$. Given a vector function Θ that maps from $D \subseteq \mathbb{R}^{|\mathbf{x}|}$ to $\mathbb{R}^{|\mathbf{y}|}$, let $\Theta(A) \mathrel{\widehat{=}} \{\Theta(\mathbf{x}) \mid \mathbf{x} \in A\}$ for any $A \subseteq D$, and $\Theta^{-1}(B) \mathrel{\widehat{=}} \{\mathbf{x} \in D \mid \Theta(\mathbf{x}) \in B\}$ for any $B \subseteq \mathbb{R}^{|\mathbf{y}|}$.

Definition 6 (Simulation [33]). *Given two CDSs $\mathcal{C}_{\mathbf{x}} \mathrel{\widehat{=}} (\Xi_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}, D_{\mathbf{x}})$ and $\mathcal{C}_{\mathbf{y}} \mathrel{\widehat{=}} (\Xi_{\mathbf{y}}, \mathbf{f}_{\mathbf{y}}, D_{\mathbf{y}})$, we say $\mathcal{C}_{\mathbf{y}}$ simulates $\mathcal{C}_{\mathbf{x}}$ or $\mathcal{C}_{\mathbf{x}}$ is simulated by $\mathcal{C}_{\mathbf{y}}$ via a continuously differentiable mapping $\Theta : D_{\mathbf{x}} \to \mathbb{R}^{|\mathbf{y}|}$, if Θ satisfies*

- *$\Theta(\Xi_{\mathbf{x}}) \subseteq \Xi_{\mathbf{y}}$, $\Theta(D_{\mathbf{x}}) \subseteq D_{\mathbf{y}}$; and*
- *for any trajectory $\mathbf{x}(t)$ of $\mathcal{C}_{\mathbf{x}}$ (i.e. a trajectory of $\dot{\mathbf{x}} = \mathbf{f}_{\mathbf{x}}(\mathbf{x})$ that starts from $\Xi_{\mathbf{x}}$ and stays in $D_{\mathbf{x}}$), $\Theta \circ \mathbf{x}(t)$ is a trajectory of $\mathcal{C}_{\mathbf{y}}$, where $\circ$ denotes composition of functions.*

We call $\mathcal{C}_{\mathbf{y}}$ an abstraction *of $\mathcal{C}_{\mathbf{x}}$ under the* simulation map *Θ.*

Abstraction of an HS can be obtained by abstracting the CDS corresponding to each mode using an individual simulation map. As argued in [34], it can be assumed without loss of generality that the collection of simulation maps for each mode all map to an Euclidean space of the same dimension, say $\mathbb{R}^{|\mathbf{y}|}$.

Definition 7 (Simulation [34]). *Given two HSs $\mathcal{H}_{\mathbf{x}} \mathrel{\widehat{=}} (Q, X, f_{\mathbf{x}}, D_{\mathbf{x}}, E, G_{\mathbf{x}}, R_{\mathbf{x}}, \Xi_{\mathbf{x}})$ and $\mathcal{H}_{\mathbf{y}} \mathrel{\widehat{=}} (Q, Y, f_{\mathbf{y}}, D_{\mathbf{y}}, E, G_{\mathbf{y}}, R_{\mathbf{y}}, \Xi_{\mathbf{y}})$, we say $\mathcal{H}_{\mathbf{y}}$ simulates $\mathcal{H}_{\mathbf{x}}$ via the set of maps $\{\Theta_q : D_{\mathbf{x},q} \to \mathbb{R}^{|\mathbf{y}|} \mid q \in Q\}$, if the following hold:*

- *$(\Xi_{\mathbf{y},q}, \mathbf{f}_{\mathbf{y},q}, D_{\mathbf{y},q})$ simulates $(\Xi_{\mathbf{x},q}, \mathbf{f}_{\mathbf{x},q}, D_{\mathbf{x},q})$ via Θ_q, for each $q \in Q$;*
- *$\Theta_q(G_{\mathbf{x},e}) \subseteq G_{\mathbf{y},e}$, for any $e = (q, q') \in E$;*
- *$\Theta_{q'}(R_{\mathbf{x},e}(\mathbf{x})) \subseteq R_{\mathbf{y},e}(\Theta_q(\mathbf{x}))$, for any $e = (q, q') \in E$ and any $\mathbf{x} \in G_{\mathbf{x},e}$.*

We call $\mathcal{H}_{\mathbf{y}}$ an abstraction *of $\mathcal{H}_{\mathbf{x}}$ under the set of* simulation maps *$\{\Theta_q \mid q \in Q\}$.*

Intuitively, if $\mathcal{H}_{\mathbf{y}}$ is an abstraction of $\mathcal{H}_{\mathbf{x}}$, then for any $(q, \mathbf{x})$ reachable by $\mathcal{H}_{\mathbf{x}}$, $(q, \Theta_q(\mathbf{x}))$ is a state reachable by $\mathcal{H}_{\mathbf{y}}$. Actually, we can prove the following nice property about such abstractions.

Theorem 1 (Invariant Preserving Property). *If $\mathcal{H}_{\mathbf{y}}$ is an abstraction of $\mathcal{H}_{\mathbf{x}}$ under simulation maps $\{\Theta_q \mid q \in Q\}$, and $\mathcal{I}_{\mathbf{y}} \mathrel{\widehat{=}} \bigcup_{q\in Q}(\{q\} \times I_{\mathbf{y},q})$ is an invariant (resp. inductive invariant) of $\mathcal{H}_{\mathbf{y}}$, then $\mathcal{I}_{\mathbf{x}} \mathrel{\widehat{=}} \bigcup_{q\in Q}(\{q\} \times I_{\mathbf{x},q})$ with $I_{\mathbf{x},q} \mathrel{\widehat{=}} \Theta_q^{-1}(I_{\mathbf{y},q})$ is an invariant (resp. inductive invariant) of $\mathcal{H}_{\mathbf{x}}$.*

Theorem 1 extends Theorem 3.2 of [33] in two aspects: firstly, it deals with HSs, and secondly, it applies to both invariants and inductive invariants; nevertheless, the proof of Theorem 1 can be given in a similar way and so is omitted here. The significance of Theorem 1 lies in the possibility of analyzing a complex HS by analyzing certain abstractions of it, which may be of simpler forms and thus allow the use of any available techniques and tools.

The following theorem proposed in [33] is very useful for checking or constructing simulation maps.

Theorem 2 (Simulation Checking [33]). *Let $\mathcal{C}_{\mathbf{x}}$, $\mathcal{C}_{\mathbf{y}}$, Θ be specified as in Definition 6. Suppose $|\mathbf{x}| = n, |\mathbf{y}| = \ell$, and $\Theta \mathrel{\widehat{=}} (\theta_1, \theta_2, \ldots, \theta_\ell)$. Then $\mathcal{C}_{\mathbf{y}}$ simulates $\mathcal{C}_{\mathbf{x}}$ if*

- *$\Theta(\Xi_{\mathbf{x}}) \subseteq \Xi_{\mathbf{y}}$, $\Theta(D_{\mathbf{x}}) \subseteq D_{\mathbf{y}}$; and*
- *$\mathbf{f}_{\mathbf{y}}(\Theta(\mathbf{x})) = \mathcal{J}_\Theta(\mathbf{x}) \cdot \mathbf{f}_{\mathbf{x}}(\mathbf{x})$, for any $\mathbf{x} \in D_{\mathbf{x}}$, where $\mathbf{f}_{\mathbf{x}}(\mathbf{x})$ is seen as a column vector, and $\mathcal{J}_\Theta(\mathbf{x})$ represents the* Jacobian matrix *of Θ at point $\mathbf{x}$.*

Example 2. Let $\mathcal{C}_{\mathbf{x}} \mathrel{\widehat{=}} \big(x \in [0,1], \dot{x} = e^x, x \in \mathbb{R}\big)$ with $\mathbf{x} = x$, $\mathcal{C}_{\mathbf{y}} \mathrel{\widehat{=}} \big(y_1 \in [0,1] \wedge y_2 \in [1,3], (\dot{y}_1, \dot{y}_2) = (y_2, y_2^2), y_1 \in \mathbb{R} \wedge y_2 > 0\big)$ with $\mathbf{y} = (y_1, y_2)$, and $\Theta \mathrel{\widehat{=}} (x, e^x)$. Then it can be checked according to Theorem 2 that $\mathcal{C}_{\mathbf{y}}$ simulates $\mathcal{C}_{\mathbf{x}}$.

We will employ Theorem 2 to prove the correctness of our abstraction of EHSs in the following section.

3 Polynomial Abstraction of EHSs

In this section, given any EHS as defined in Definition 2, we will construct a PHS that simulates the EHS in the sense of Definition 7. We will first show how elementary ODEs can be transformed into polynomial ones, and then discuss how to deal with initial sets, domains, guards, and reset maps.

3.1 Polynomialization of Elementary ODEs

In this part, we illustrate how to transform an elementary ODE equivalently into a polynomial one by introducing new variables to replace non-polynomial terms. The basic idea here is similar to [16], but we give a clearer statement of the transformation procedure and extend it from ODEs to hybrid systems.

Univariate Basic Elementary Functions. For

$$\dot{x} = f(x) \tag{3}$$

- if $f(x) = \frac{1}{x}$, then let $v = \frac{1}{x}$, and thus $\dot{v} = -\frac{\dot{x}}{x^2}$. Therefore (3) is transformed to[4]

$$\begin{cases} \dot{x} = v \\ \dot{v} = -v^3 \end{cases}$$

[4] By $v = \frac{1}{x}$, the set $\{(x, v) \mid x = 0, v \in \mathbb{R}\}$ is excluded from the domain of the transformed polynomial ODE. Such a consequence will not be explicitly mentioned in the rest of this paper.

– if $f(x) = \sqrt{x}$, then let $v = \sqrt{x}$, and thus $\dot{v} = \frac{\dot{x}}{2\sqrt{x}}$. Therefore (3) is transformed to

$$\begin{cases} \dot{x} = v \\ \dot{v} = \frac{1}{2} \end{cases}$$

– if $f(x) = e^x$, then let $v = e^x$, and thus $\dot{v} = e^x \cdot \dot{x}$. Therefore (3) is transformed to

$$\begin{cases} \dot{x} = v \\ \dot{v} = v^2 \end{cases}$$

– if $f(x) = \ln x$, then let $v = \ln x$, and thus $\dot{v} = \frac{\dot{x}}{x}$; then further let $u = \frac{1}{x}$, and thus $\dot{u} = -\frac{\dot{x}}{x^2}$. Therefore (3) is transformed to

$$\begin{cases} \dot{x} = v \\ \dot{v} = uv \\ \dot{u} = -u^2 v \end{cases}$$

– if $f(x) = \sin x$, then let $v = \sin x$, and thus $\dot{v} = \dot{x} \cdot \cos x$; then further let $u = \cos x$, and thus $\dot{u} = -\sin x \cdot \dot{x}$. Therefore (3) is transformed to

$$\begin{cases} \dot{x} = v \\ \dot{v} = uv \\ \dot{u} = -v^2 \end{cases}$$

– if $f(x) = \cos x$, then the transformation is analogous to the case of $f(x) = \sin x$.

Compositional and Multivariate Functions. Obviously, the outmost form of any compositional elementary function must be one of $f \pm g$, $f \times g$, $\frac{f}{g}$, f^a, e^f, $\ln(f)$, $\sin(f)$, $\cos(f)$. Therefore given a compositional function, we can iterate the above procedure discussed on basic cases from the innermost non-polynomial sub-term to the outside, until all the sub-expressions have been transformed into polynomials. For example,

– if $f(x) = \ln(2 + \sin x)$, we can let

$$\begin{cases} v &= \sin x \\ u &= \cos x \\ w &= \ln(2+v) = \ln(2+\sin x) \\ z &= \frac{1}{2+v} = \frac{1}{2+\sin x} \end{cases}, \text{ and then (3) is transformed to } \begin{cases} \dot{x} &= w \\ \dot{v} &= uw \\ \dot{u} &= -vw \\ \dot{w} &= zuw \\ \dot{z} &= -z^2 uw \end{cases}.$$

Handling multivariate functions is straightforward.

In summary, we give the following assertion on polynomializing elementary ODES, the correctness of which can be given based on the formal transformation algorithms presented in the full version of this paper [22].

Proposition 1 (Polynomial Recasting). *Given an ODE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ with $\mathbf{f}(\mathbf{x})$ an elementary vector function defined on an open set $U \subseteq \mathbb{R}^n$, there exists a collection of variable replacement equations $\mathbf{v} = \Gamma(\mathbf{x})$, where $\mathbf{v} = (v_1, v_2, \ldots, v_m)$ is a vector of* *new variables and* $\Gamma(\mathbf{x}) = (\gamma_1(\mathbf{x}), \gamma_2(\mathbf{x}), \ldots, \gamma_m(\mathbf{x})) : U \to \mathbb{R}^m$ *is an elementary vector function, such that*

$$\begin{pmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{x}) \\ \mathcal{J}_\Gamma(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{x}) \\ \mathcal{J}_\Gamma(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) \end{pmatrix} \Big[\!\!\Big[\mathbf{v}/\Gamma(\mathbf{x}) \Big]\!\!\Big] \mathrel{\widehat{=}} \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{v}) \tag{4}$$

becomes a polynomial ODE, that is, $\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{v})$ *is a polynomial vector function in variables* $\mathbf{x}$ *and* $\mathbf{v}$. *Here* $\mathsf{expr}[\![\mathbf{v}/\Gamma(\mathbf{x})]\!]$ *means replacing any occurrence of the non-polynomial term* $\gamma_i(\mathbf{x})$ *in the expression* expr *by the corresponding variable* v_i, *for all* $1 \leqslant i \leqslant m$.

It can be proved that the number of variables $\mathbf{v}$ is at most twice the number of non-polynomial terms in the original ODE, which can be a small number in practice. The transformed polynomial ODE as specified in Proposition 1 is *equivalent* to the original one in the following sense.

Theorem 3 (Trajectory Equivalence). *Let* $\mathbf{f}(\mathbf{x})$, $\Gamma(\mathbf{x})$ *and* $\tilde{\mathbf{f}}(\mathbf{x}, \mathbf{v})$ *be as specified in Proposition 1. Then for any trajectory* $\mathbf{x}(t)$ *of* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *starting from* $\mathbf{x}_0 \in U \subseteq \mathbb{R}^n$, $\big(\mathbf{x}(t), \Gamma(\mathbf{x}(t))\big)$ *is the trajectory of* $(\dot{\mathbf{x}}, \dot{\mathbf{v}}) = \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{v})$ *starting from* $(\mathbf{x}_0, \Gamma(\mathbf{x}_0))$*; conversely, for any trajectory* $(\mathbf{x}(t), \mathbf{v}(t))$ *of* $(\dot{\mathbf{x}}, \dot{\mathbf{v}}) = \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{v})$ *starting from* $(\mathbf{x}_0, \mathbf{v}_0) \in \mathbb{R}^{n+m}$, *if* $\mathbf{x}_0 \in U$ *and* $\mathbf{v}_0 = \Gamma(\mathbf{x}_0)$, *then* $\mathbf{x}(t)$ *is the trajectory of* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ *starting from* $\mathbf{x}_0$.

Proof. The result can be deduced directly from (4). □

3.2 Abstracting EDSs by PDSs

In this part, given an EDS $\mathcal{C}_\mathbf{x} \mathrel{\widehat{=}} (\Xi_\mathbf{x}, \mathbf{f}_\mathbf{x}, D_\mathbf{x})$ we will construct a PDS $C_\mathbf{y} \mathrel{\widehat{=}} (\Xi_\mathbf{y}, \mathbf{f}_\mathbf{y}, D_\mathbf{y})$ that simulates $\mathcal{C}_\mathbf{x}$. The construction is based on the procedure introduced in Section 3.1 on polynomial transformation of elementary ODEs. The basic idea is to construct a simulation map using the replacement equations. The difference here is that when abstracting an EDS, we need to replace non-polynomial terms occurring in not only the vector field, but also the initial set and domain. Suppose we have obtained the replacement equation $\mathbf{v} = \Gamma(\mathbf{x})$ and $\mathbf{f}_\mathbf{y}$ as defined in (4). Then we define the simulation map $\Theta : D_\mathbf{x} \to \mathbb{R}^{|\mathbf{y}|}$ as[5]

$$\Theta(\mathbf{x}) = (\mathbf{x}, \Gamma(\mathbf{x})) \ . \tag{5}$$

Now $\Xi_\mathbf{y}$ and $D_\mathbf{y}$ can be constructed using Θ, as illustrated in detail next.

To construct $\Xi_\mathbf{y}$, consider the image of $\Xi_\mathbf{x}$ under the simulation map Θ

$$\Theta(\Xi_\mathbf{x}) = \{(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^{|\mathbf{y}|} \mid \mathbf{x} \in \Xi_\mathbf{x} \wedge \mathbf{v} = \Gamma(\mathbf{x})\} \,,$$

briefly denoted by $\Theta(\Xi_\mathbf{x}) \mathrel{\widehat{=}} \Xi_\mathbf{x} \wedge \mathbf{v} = \Gamma(\mathbf{x})$, or alternatively

$$\Theta(\Xi_\mathbf{x}) \mathrel{\widehat{=}} \Xi_\mathbf{x}[\![\mathbf{v}/\Gamma(\mathbf{x})]\!] \wedge \mathbf{v} = \Gamma(\mathbf{x}) \,. \tag{6}$$

The first conjunct in (6) is of polynomial form, but the second conjunct contains elementary functions. By Definition 6, we need to get a polynomial over-approximation $\Xi_\mathbf{y}$ of $\Theta(\Xi_\mathbf{x})$, which means we need to abstract $\mathbf{v} = \Gamma(\mathbf{x})$ in (6) by polynomial expressions. We propose the following four ways to do so.

[5] Here we assume that all elementary functions in $\Xi_\mathbf{x}$, $\mathbf{f}_\mathbf{x}$ and $D_\mathbf{x}$ are defined on $D_\mathbf{x}$.

(W1) When $\mathit{\Gamma}(\mathbf{x})$ are some special kinds of elementary functions, $\mathbf{v} = \mathit{\Gamma}(\mathbf{x})$ can be equivalently transformed to polynomial expressions, e.g.

$$\begin{cases} v = \dfrac{1}{x} \Longleftrightarrow vx = 1 \\ v = \sqrt{x} \Longleftrightarrow v^2 = x \,\wedge\, v \geqslant 0 \end{cases} . \tag{7}$$

(W2) If $\mathit{\Xi}_{\mathbf{x}}$ is a bounded region and the upper/lower bounds of each component x_i of $\mathbf{x}$ can be easily obtained, then we can compute the Taylor polynomial expansion $\mathbf{p}(\mathbf{x})$ of $\mathit{\Gamma}(\mathbf{x})$ over the bounded region up to a certain degree, as well as an interval over-approximation $\mathbf{I}$ of the corresponding truncation error, such that $\mathbf{v} = \mathit{\Gamma}(\mathbf{x})$ can be approximated by $\mathbf{v} \in (\mathbf{p}(\mathbf{x}) + \mathbf{I})$.

(W3) We can also compute the range of $\mathit{\Gamma}(\mathbf{x})$ (over $\mathit{\Xi}_{\mathbf{x}}$) as an over-approximation of $\mathbf{v}$, e.g.

$$\begin{cases} v = \sin x \Longrightarrow -1 \leqslant v \leqslant 1 \\ v = e^x \Longrightarrow v > 0 \end{cases} . \tag{8}$$

(W4) The simplest way is to remove the constraint $\mathbf{v} = \mathit{\Gamma}(\mathbf{x})$ entirely, which means $\mathbf{v}$ is allowed to take any value from $\mathbb{R}^{|\mathbf{v}|}$.

From (W1) to (W4), the over-approximation of $\mathbf{v} = \mathit{\Gamma}(\mathbf{x})$ becomes more and more coarse. Usually it takes more effort to obtain a more refined abstraction, but the result would be more useful for analysis of the original system.

The construction of $D_{\mathbf{y}}$ is similar to $\mathit{\Xi}_{\mathbf{y}}$. Then we can give the following conclusion, the proof of which can be found in [22].

Theorem 4 (Abstracting EDS by PDS). *Given an EDS $\mathcal{C}_{\mathbf{x}} \mathrel{\widehat{=}} (\mathit{\Xi}_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}, D_{\mathbf{x}})$, let $\mathcal{C}_{\mathbf{y}} \mathrel{\widehat{=}} (\mathit{\Xi}_{\mathbf{y}}, \mathbf{f}_{\mathbf{y}}, D_{\mathbf{y}})$, where $\mathbf{f}_{\mathbf{y}}$ is given by (4), and $\mathit{\Xi}_{\mathbf{y}}$, $D_{\mathbf{y}}$ are given by (6) together with (W1)-(W4). Then $\mathcal{C}_{\mathbf{y}}$ is a* polynomial *abstraction of $\mathcal{C}_{\mathbf{x}}$ in the sense of Definition 6, under simulation map $\mathit{\Theta}$ defined by (5).*

Example 3. Consider the EDS $\mathcal{C}_{\mathbf{x}} \mathrel{\widehat{=}} (\mathit{\Xi}_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}, D_{\mathbf{x}})$, where

– $\mathit{\Xi}_{\mathbf{x}} \mathrel{\widehat{=}} (x + 0.5)^2 + (y - 0.5)^2 - 0.16 \leqslant 0$;
– $D_{\mathbf{x}} \mathrel{\widehat{=}} -2 \leqslant x \leqslant 2 \wedge -2 \leqslant y \leqslant 2$; and
– $\mathbf{f}_{\mathbf{x}}$ defines the ODE

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} e^{-x} + y - 1 \\ -\sin^2(x) \end{pmatrix} . \tag{9}$$

A PDS $\mathcal{C}_{\mathbf{y}}$ that simulates $\mathcal{C}_{\mathbf{x}}$ can be constructed as follows.

1) Noticing that $\mathit{\Xi}_{\mathbf{x}}$ and $D_{\mathbf{x}}$ are both in polynomial forms, we only need to replace non-polynomial terms in $\mathbf{f}_{\mathbf{x}}$. We finally obtain the replacement relations $\mathbf{v} = \mathit{\Gamma}(\mathbf{x})$

$$(v_1, v_2, v_3) = (\sin x, e^{-x}, \cos x) \tag{10}$$

and the transformed polynomial ODE

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \end{pmatrix} = \begin{pmatrix} v_2 + y - 1 \\ -v_1^2 \\ v_3(v_2 + y - 1) \\ -v_2(v_2 + y - 1) \\ -v_1(v_2 + y - 1) \end{pmatrix} , \tag{11}$$

the right-hand-side of which is defined to be $\mathbf{f_y}$.

2) The simulation map Θ is given by

$$\Theta(x, y) = (x, y, \sin x, e^{-x}, \cos x)\ . \tag{12}$$

The images of $\Xi_{\mathbf{x}}$ and $D_{\mathbf{x}}$ under Θ are

$$\Theta(\Xi_{\mathbf{x}}) \mathrel{\widehat{=}} \Xi_{\mathbf{x}} \wedge v_1 = \sin x \wedge v_2 = e^{-x} \wedge v_3 = \cos x$$

and

$$\Theta(D_{\mathbf{x}}) \mathrel{\widehat{=}} D_{\mathbf{x}} \wedge v_1 = \sin x \wedge v_2 = e^{-x} \wedge v_3 = \cos x$$

respectively. For the above two formulas, (W1) is not applicable, whereas we can use any of (W2)-(W4) to abstract them. Here we give one possible way adopting (W2): first, using the tool COSY INFINITY[6] for *Taylor model* [24] computation, we expand $\sin x$, e^{-x} and $\cos x$ over $x \in [-2, 2]$ at $x = 0$ up to degree 6, and obtain

$$TM_{\mathbf{x},\mathbf{v}} \mathrel{\widehat{=}} \begin{array}{c} p_1(x) + l_1 \leqslant v_1 \leqslant p_1(x) + u_1 \\ \wedge\ p_2(x) + l_2 \leqslant v_2 \leqslant p_2(x) + u_2 \\ \wedge\ p_3(x) + l_3 \leqslant v_3 \leqslant p_3(x) + u_3 \end{array} ; \tag{13}$$

then we define $D_{\mathbf{y}} \mathrel{\widehat{=}} D_{\mathbf{x}} \wedge TM_{\mathbf{x},\mathbf{v}}$; second, from $\Xi_{\mathbf{x}}$ it can be deduced that $x \in [-0.9, -0.1]$ for any $(x, y) \in \Xi_{\mathbf{x}}$, and then we can compute the Taylor models of $\mathbf{v} = \Gamma(\mathbf{x})$ over $[-0.9, -0.1]$ and obtain $\Xi_{\mathbf{y}}$. Please see [22] for details of computed Taylor models. Thus we finally get a PDS $\mathcal{C}_{\mathbf{y}} \mathrel{\widehat{=}} (\Xi_{\mathbf{y}}, \mathbf{f_y}, D_{\mathbf{y}})$ that simulates $\mathcal{C}_{\mathbf{x}}$.

3.3 Abstracting EHSs by PHSs

In the previous sections, we have presented a method to abstract an EDS to a PDS such that the PDS simulates the EDS. Now we show, given an EHS $\mathcal{H}_{\mathbf{x}}$, how to construct a PHS $\mathcal{H}_{\mathbf{y}}$ that simulates $\mathcal{H}_{\mathbf{x}}$. Actually, this can be easily done by just extending the previous abstraction approach a bit to take into account guard constraints and reset functions. Another difference is that we need to treat each mode of an HA separately by constructing an individual simulation map for each of them. For the details of how guards and reset maps can be abstracted, the readers can refer to the full version [22].

Example 4. Consider the EHS in Example 1 except for that the initial set is replaced by $\Xi_q \mathrel{\widehat{=}} y \in [4.9, 5.1] \wedge x = 0 \wedge v_x = -1 \wedge v_y = 0$. Denote this EHS by $\mathcal{H}_{\mathbf{x}}$ with $\mathbf{x} = (x, y, v_x, v_y)$. By applying the proposed abstraction approach, we obtained the replacement equations $(u_1, u_2, u_3) = (\sin x, \cos x, \frac{1}{1+(\cos x)^2})$ and the corresponding polynomial abstraction $\mathcal{H}_{\mathbf{y}}$, with the polynomial guard $G_{\mathbf{y},e} \mathrel{\widehat{=}} y = u_1$ and polynomial reset mapping $R_{\mathbf{y},e}(\mathbf{y}) \mathrel{\widehat{=}} \{(x, y, v'_x, v'_y, u_1, u_2, u_3)\}$ where

$$\begin{cases} v'_x = u_3 \cdot (u_1^2 \cdot v_x + 2u_2 \cdot v_y) \\ v'_y = u_3 \cdot (2u_2 \cdot v_x - u_1^2 \cdot v_y) \end{cases} .$$

The complete model of $\mathcal{H}_{\mathbf{y}}$ can be found in [22].

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

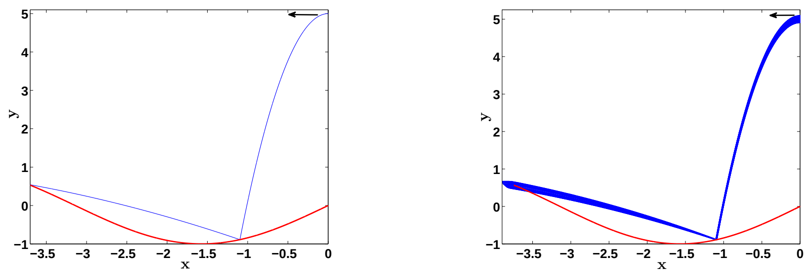


Fig. 2. Simulated trajectory with $y = 5$ and reachable set over-approximation with $y \in [4.9, 5.1]$

Then we can use existing tools for PHSs to analyze the behavior of the bouncing ball. Here we use the state-of-the-art nonlinear hybrid system analyzer Flow* [6]. The right picture in Figure 2 shows the computed reachable set over-approximation (projected to the x-y plane) of $\mathcal{H}_{\mathbf{y}}$ with the initial set $\Xi_{\mathbf{y},q}$ for the first two jumps, which is also the reachable set over-approximation of $\mathcal{H}_{\mathbf{x}}$ by Theorem 1. Note that such an analysis would NOT have been possible directly on $\mathcal{H}_{\mathbf{x}}$ in Flow* since its current version does not support elementary functions in domains, guards, or reset functions[7].

4 Application in Safety Verification of EHSs

One of the most important problems in the study of HSs is safety verification. Given an HS $\mathcal{H}$, a safety requirement for $\mathcal{H}$ can be specified as $\mathcal{S} \mathrel{\widehat{=}} \bigcup_{q\in Q}(\{q\} \times S_q)$ with $S_q \subseteq \mathbb{R}^n$ the safe region of mode q. Alternatively, a safety property can be given as a set of unsafe regions $\mathcal{US} \mathrel{\widehat{=}} \bigcup_{q\in Q}(\{q\} \times \bar{S}_q)$ with $\bar{S}_q$ the complement of S_q in $\mathbb{R}^n$. The safety verification problem asks whether $\mathcal{R}_{\mathcal{H}} \subseteq \mathcal{S}$, or equivalently, whether $\mathcal{R}_{\mathcal{H}} \cap \mathcal{US} = \emptyset$.

The following result relates the safety verification problem of an HS $\mathcal{H}_{\mathbf{x}}$ to that of $\mathcal{H}_{\mathbf{y}}$ which simulates $\mathcal{H}_{\mathbf{x}}$. The proof of it can be found in [22].

Theorem 5 (Safety Relation). *Let $\mathcal{US}_{\mathbf{x}} \mathrel{\widehat{=}} \bigcup_q(\{q\} \times \bar{S}_{\mathbf{x},q})$ be a safety requirement of the HS $\mathcal{H}_{\mathbf{x}}$. Suppose $\mathcal{H}_{\mathbf{y}}$ simulates $\mathcal{H}_{\mathbf{x}}$ via simulation maps $\{\Theta_q \mid q \in Q\}$ and $\mathcal{US}_{\mathbf{y}} \mathrel{\widehat{=}} \bigcup_q(\{q\} \times \bar{S}_{\mathbf{y},q})$ satisfies $\bar{S}_{\mathbf{y},q} \supseteq \Theta_q(\bar{S}_{\mathbf{x},q})$ for any $q \in Q$. Then $\mathcal{H}_{\mathbf{x}}$ is safe w.r.t. $\mathcal{US}_{\mathbf{x}}$, if $\mathcal{H}_{\mathbf{y}}$ is safe w.r.t. $\mathcal{US}_{\mathbf{y}}$.*

Note that if the safety properties of EHSs are not in polynomial forms but contain elementary functions, we can replace the non-polynomial terms by new variables when constructing the simulation map, as we do for the EHSs themselves.

Theorem 5 allows us to take advantage of constraint-based approaches for PHSs to verify safety properties of EHSs. In the rest of this section, we show how to perform safety verification for EHSs by combining the previous proposed polynomial abstraction method with constraint-based verification techniques for PHSs.

[7] Although Flow* does support nonlinear continuous dynamics with non-polynomial terms such as sine, cosine, square root, etc.

4.1 Generating Invariants for EHSs

In this and next subsections, for simplicity, we will use EDSs as special cases of EHSs to illustrate how to generate inductive invariants for safety verification of EHSs.

Given an EDS $\mathcal{C}_{\mathbf{x}} \widehat{=} (\varXi_{\mathbf{x}}, \mathbf{f}_{\mathbf{x}}, D_{\mathbf{x}})$ and an unsafe region $\bar{S}_{\mathbf{x}}$, we first construct a PDS $\mathcal{C}_{\mathbf{y}} \widehat{=} (\varXi_{\mathbf{y}}, \mathbf{f}_{\mathbf{y}}, D_{\mathbf{y}})$ that simulates $\mathcal{C}_{\mathbf{x}}$, as well as the polynomial abstraction $\bar{S}_{\mathbf{y}}$ of $\bar{S}_{\mathbf{x}}$. According to Theorem 1 and 5, if we can find a semi-algebraic[8] inductive invariant $P(\mathbf{y}) = P(\mathbf{x}, \mathbf{v})$ for $\mathcal{C}_{\mathbf{y}}$ with $\mathbf{v} = \varGamma(\mathbf{x})$ the replacement equations, such that $P(\mathbf{x}, \mathbf{v})$ is a certificate of the safety of $\mathcal{C}_{\mathbf{y}}$ w.r.t. $\bar{S}_{\mathbf{y}}$, then $P(\mathbf{x}, \varGamma(\mathbf{x}))$ is an inductive invariant certificate of the safety of $\mathcal{C}_{\mathbf{x}}$ w.r.t. $\bar{S}_{\mathbf{x}}$. If $P(\mathbf{x}, \mathbf{v})$ does contain variables $\mathbf{v}$, then $P(\mathbf{x}, \varGamma(\mathbf{x}))$ gives an elementary invariant of $\mathcal{C}_{\mathbf{x}}$; otherwise $P(\mathbf{x}, \varGamma(\mathbf{x}))$ is just a polynomial invariant.

Example 5. Consider the EDS $\mathcal{C}_{\mathbf{x}}$ in Example 3. We are going to verify the safety of $\mathcal{C}_{\mathbf{x}}$ w.r.t. an unsafe region $\bar{S}_{\mathbf{x}} \widehat{=} (x-0.7)^2+(y+0.7)^2-0.09 \leqslant 0$. To this end, we first verify the safety of the polynomial abstraction $\mathcal{C}_{\mathbf{y}}$ of $\mathcal{C}_{\mathbf{x}}$ obtained in Example 3, w.r.t. an unsafe region $\bar{S}_{\mathbf{y}}$ that abstracts $\bar{S}_{\mathbf{x}}$ in the way of (W2). By applying the SOS-relaxation-based invariant generation approach [29, 18] with a polynomial template $p_1(\mathbf{u}, \mathbf{x}, \mathbf{v}) \leqslant 0$ of degree 3 (in $\mathbf{x}, \mathbf{v}$), where $\mathbf{v} = \varGamma(\mathbf{x})$ is defined in (10) and $\mathbf{u}$ is a vector of undetermined parameters, and using the Matlab-based tool YALMIP [23] and SeDuMi [37] (or S-DPT3 [40]), we successfully generated an invariant $p_1(x, y, \sin x, e^{-x}, \cos x) \leqslant 0$ that verifies the safety of $\mathcal{C}_{\mathbf{x}}$. Please see the left part of Figure 3 for an illustration of $\mathbf{f}_{\mathbf{x}}$ (the black arrows), $D_{\mathbf{x}}$ (the outer white box), $\varXi_{\mathbf{x}}$ (the white circle), $\bar{S}_{\mathbf{x}}$ (the black circle), as well as the synthesized invariant (the grey area with curved boundary).

We next try to generate polynomial invariants for $\mathcal{C}_{\mathbf{x}}$ by constructing less accurate abstractions $\mathcal{C}_{\mathbf{y}}$ and $\bar{S}_{\mathbf{y}}$ of $\mathcal{C}_{\mathbf{x}}$ and $\bar{S}_{\mathbf{x}}$ respectively (see [22] for the details). With a polynomial template $p_2(\mathbf{u}, \mathbf{x}) \leqslant 0$ of degree 5 (in $\mathbf{x}$), we obtain an invariant $p_2(x, y) \leqslant 0$ that verifies safety of $\mathcal{C}_{\mathbf{x}}$, as shown in the right part of Figure 3. The explicit forms of both p_1 and p_2 can be found in [22].

We can see that the elementary invariant is sharper than the polynomial invariant and separates better from the unsafe region. This indicates that by allowing non-polynomial terms in templates, invariants of higher quality may be generated and thus increases the possibility of verifying safety properties of EHSs. Moreover, it also suggests that even for purely polynomial systems, one could assume any kind of elementary terms in a predefined template when generating invariants, which gives a more general method than [32, 11] for generating elementary invariants for PHSs. Of course, computing elementary invariants generally takes more efforts than polynomial invariants.

4.2 More Experiments

We have implemented the proposed abstraction approach (not including the part on abstraction of replacement equations) and experimented with it on the following EHS verification examples.[9]

[8] A set $A \subseteq \mathbb{R}^n$ is called *semi-algebraic* if it can be defined by Boolean combinations of polynomial equations or inequalities.

[9] The formal abstraction algorithms can be found in [22], and all the input files for the experiments can be obtained at http://lcs.ios.ac.cn/%7Ezoul/casestudies/fm2015.zip

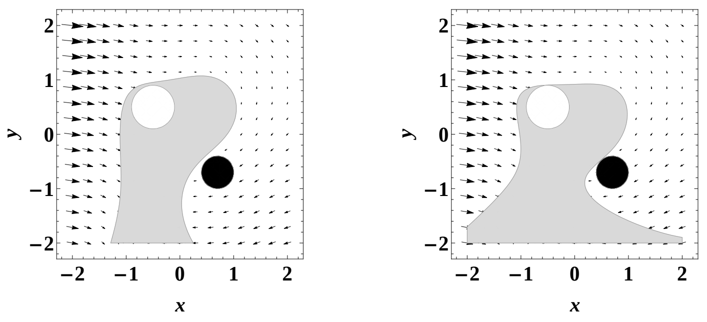


Fig. 3. Comparison of elementary and polynomial invariants generated in Example 5

Example 6 (HIV Transmission). The following continuous dynamics, with the assumption that there is no recruitment of population, has been developed to model HIV transmission [3]

$$\mathbf{f} \mathrel{\widehat{=}} \begin{cases} \dot{u}_1 = -\frac{\beta c u_1 u_2}{u_1+u_2+u_3} - \mu u_1 \\ \dot{u}_2 = \frac{\beta c u_1 u_2}{u_1+u_2+u_3} - (\mu+\nu) u_2 \\ \dot{u}_3 = \nu u_2 - \alpha u_3 \end{cases}, \tag{14}$$

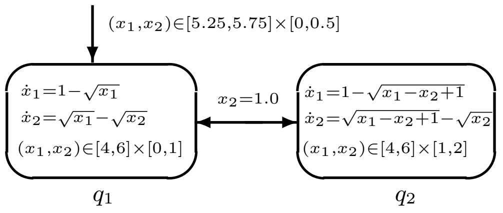
where $u_1(t), u_2(t), u_3(t)$ denote the part of population that is HIV susceptible, HIV infected, and that has AIDS respectively, β is the possibility of infection per partner contact, c is the rate of partner change, μ is the death rate of non-AIDS population, α is the death rate of AIDS patients, 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 $\beta = 0.2, c = 10, \mu = 0.008, \alpha = 0.95$, and $\nu = 0.1$. 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 $(\Xi, \mathbf{f}, D)$ satisfies $S \mathrel{\widehat{=}} u_3 \leqslant 1$, where $D \mathrel{\widehat{=}} u_1 \geqslant 0 \wedge u_2 \geqslant 0 \wedge u_3 \geqslant 0 \wedge 0 < u_1 + u_2 + u_3 \leqslant 10.013$.[10]

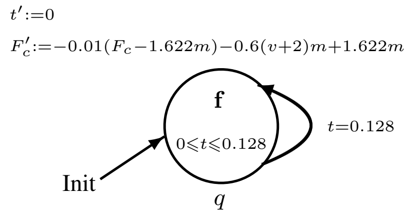
Example 7 (Two-Tanks). The two-tanks system shown in Figure 4 comes from [38] and has been studied in [31, 14, 8, 7] as a benchmark for safety verification of hybrid systems. It models two connected tanks, the liquid levels of which are denoted by x_1 and x_2 respectively. The system switches between mode q_1 and q_2 when x_2 reaches 1. 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 $\Xi_{q_1} \mathrel{\widehat{=}} 5.25 \leqslant x_1 \leqslant 5.75 \wedge 0 \leqslant x_2 \leqslant 0.5$, the system will never reach the unsafe set $\bar{S}_{q_1} \mathrel{\widehat{=}} (x_1 - 4.25)^2 + (x_2 - 0.25)^2 - 0.0625 \leqslant 0$ when staying at mode q_1.

[10] According to dynamics (14), the entire population is non-increasing, so $u_1 + u_2 + u_3$ has an upper bound.



$(x_1,x_2)\in[5.25,5.75]\times[0,0.5]$
$\dot{x}_1=1-\sqrt{x_1}$
$\dot{x}_2=\sqrt{x_1}-\sqrt{x_2}$
$(x_1,x_2)\in[4,6]\times[0,1]$
$x_2=1.0$
$\dot{x}_1=1-\sqrt{x_1-x_2+1}$
$\dot{x}_2=\sqrt{x_1-x_2+1}-\sqrt{x_2}$
$(x_1,x_2)\in[4,6]\times[1,2]$
q_1
q_2

Fig. 4. The two-tanks system



$t':=0$
$F_c':=-0.01(F_c-1.622m)-0.6(v+2)m+1.622m$
f
$0\leqslant t\leqslant 0.128$
$t=0.128$
Init
q

Fig. 5. The lunar lander system

Example 8 (Lunar Lander). Consider a real-world example of the guidance and control of a lunar lander [41], as illustrated by Figure 5. The dynamics of the lander defined by **f** is given by

$$\begin{cases} \dot{v} &= \frac{F_c}{m} - 1.622 \\ \dot{m} &= -\frac{F_c}{2500} \\ \dot{F}_c &= 0 \\ \dot{t} &= 1 \end{cases}, \qquad (15)$$

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 Figure 5. Note that the derivative of v involves non-polynomial expression $\frac{1}{m}$. We want to verify that with the initial condition $t = 0$s, $v = -2$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.

Using the proposed abstraction method and the SOS-relaxation-based invariant generation method [29, 18], we have successfully verified Examples 6-8. Besides, we have also compared with the performances of the EHS verification tools HSOLVER [31], Flow* [6], dReach [9] and iSAT-ODE [8] on these examples (including Example 5).[11] All the time costs are shown in Table 1. The results are obtained on the platform with Intel Core i5-3470 CPU and 4GB RAM running Windows 7 (for the proposed abstraction approach) or Ubuntu Linux 14.04 (for the other tools).

Table 1. Verification results of different methods

	EHS2PHS	HSOLVER	Flow*	dReach	iSAT-ODE
E.g. 5	1.324 or 7.994	0.739	⊖	–	⊖
E.g. 6	5.186	–	⊖	–	⊖
E.g. 7	0.977	0.477	76.742	20.351	0.949
E.g. 8	2.645	–	3.310	–	54.364

[11] Note that since Flow*, dReach and iSAT-ODE can only do BMC, we have assumed a time bound of 20s and 10s resp. for E.g. 5 and 6, and a jump bound of 40 steps and 100 steps resp. for E.g. 7 and 8.

For Table 1, we have the following remarks:

- time is measured in seconds;
- the second column (EHS2PHS) corresponds to the time costs of both abstraction and verification for the proposed approach in this paper; the time cost on E.g. 5 using the abstraction approach depends on whether we generate polynomial (1.324s) or elementary (7.994s) invariants;
- the other four tools are called on the original EHSs rather than their polynomial abstractions; $-$ means timeout ($>$ 1 hour); $\ominus$ means abnormal termination due to error inflation of continuous integration;
- all the details of inputs, experiment results, all well as options for using the tools can be obtained by investigating the aforementioned online files.

From Table 1 we can see that the time costs of the proposed abstraction approach are all acceptable, whereas there do exist examples that existing approaches cannot solve effectively.[12]

5 Conclusions

In this paper, we presented an approach to reducing an EHS to a PHS by variable transformation, and established the simulation relation between them, so that safety verification of the EHS can be reduced to that of the corresponding PHS. Thus our work enables all the well-established techniques for PHS verification to be applicable to EHSs. In particular, combined with invariant-based approach to safety verification for PHSs, it provides the possibility of overcoming the limitations of existing EHS verification approaches. Experimental results on real-world examples indicated the effectiveness of our approach.

In the future, it deserves to investigate how to reduce stability analysis of EHSs to that of PHSs using the technique developed in this paper, while stability analysis of PHSs can be well done by synthesizing (relaxed) polynomial Lyaponuv functions (e.g., cf. [21]).

Acknowledgements. We thank Prof. Martin Fränzle, Dr. Andreas Eggers, Dr. Sicun Gao and Dr. Xin Chen for their instructions on using the verification tools. We also thank the anonymous referees for their valuable comments on the earlier drafts.

References

1. Akbarpour, B., Paulson, L.: MetiTarski: An automatic theorem prover for real-valued special functions. Journal of Automated Reasoning 44(3), 175–205 (2010)
2. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems, LNCS, vol. 736, pp. 209–229. Springer Berlin Heidelberg (1993)
3. 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)

[12] One may notice that HSOLVER is faster on the examples that it can solve, the possible reason for which is that a coarse abstraction happens to be sufficient for proving the safety of studied systems, thus saving much time for refinement.

4. Asarin, E., Dang, T., Girard, A.: Hybridization methods for the analysis of nonlinear systems. Acta Informatica 43(7), 451–476 (2007)
5. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: RTSS 2012. pp. 183–192. IEEE Computer Society, Los Alamitos, CA, USA (2012)
6. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013, LNCS, vol. 8044, pp. 258–263. Springer Berlin Heidelberg (2013)
7. Denman, W.: Verifying nonpolynomial hybrid systems by qualitative abstraction and automated theorem proving. In: Badger, J.M., Rozier, K.Y. (eds.) NFM 2014, LNCS, vol. 8430, pp. 203–208. Springer International Publishing (2014)
8. 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)
9. Gao, S., Kong, S., Clarke, E.: dReach: Reachability analysis for nonlinear hybrid systems (tool paper). In: HSCC 2013 (2013), `http://dreal.cs.cmu.edu/#!dreach.md`
10. Ghorbal, K., Platzer, A.: Characterizing algebraic invariants by differential radical invariants. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014, LNCS, vol. 8413, pp. 279–294. Springer Berlin Heidelberg (2014)
11. Goubault, E., Jourdan, J.H., Putot, S., Sankaranarayanan, S.: Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials. pp. 3571–3578. ACC 2014 (2014)
12. Gulwani, S., Tiwari, A.: Constraint-based approach for analysis of hybrid systems. In: Gupta, A., Malik, S. (eds.) CAV 2008, LNCS, vol. 5123, pp. 190–203. Springer Berlin Heidelberg (2008)
13. Henzinger, T.A.: The theory of hybrid automata. In: LICS 1996. pp. 278–292. IEEE Computer Society (Jul 1996)
14. 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)
15. Johnson, T.T., Green, J., Mitra, S., Dudley, R., Erwin, R.S.: Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 252–266. Springer Berlin Heidelberg (2012)
16. Kerner, E.H.: Universal formats for nonlinear ordinary differential systems. Journal of Mathematical Physics 22(7), 1366–1371 (1981)
17. Khalil, H.K.: Nonlinear Systems. Prentice Hall, third edn. (Dec 2001)
18. Kong, H., He, F., Song, X., Hung, W.N., Gu, M.: Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 242–257. Springer Berlin Heidelberg (2013)
19. Lanotte, R., Tini, S.: Taylor approximation for hybrid systems. Information and Computation 205(11), 1575–1607 (Nov 2007)
20. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT 2011. pp. 97–106. ACM, New York, NY, USA (2011)
21. Liu, J., Zhan, N., Zhao, H.: Automatically discovering relaxed Lyapunov functions for polynomial dynamical systems. Mathematics in Computer Science 6(4), 395–408 (2012)
22. Liu, J., Zhan, N., Zhao, H., Zou, L.: Abstraction of elementary hybrid systems by variable transformation. CoRR abs/1403.7022 (2014), `http://arxiv.org/abs/1403.7022`
23. Löfberg, J.: YALMIP : A toolbox for modeling and optimization in MATLAB. In: Proc. of the CACSD Conference. Taipei, Taiwan (2004), `http://users.isy.liu.se/johanl/yalmip/`

24. Makino, K., Berz, M.: Taylor models and other validated functional inclusion methods. International Journal of Pure and Applied Mathematics 4(4), 379–456 (2003)
25. Mitchell, I., Tomlin, C.J.: Level set methods for computation in hybrid systems. In: Lynch, N., Krogh, B.H. (eds.) HSCC 2000, LNCS, vol. 1790, pp. 310–323. Springer Berlin Heidelberg (2000)
26. Papachristodoulou, A., Prajna, S.: Analysis of non-polynomial systems using the sum of squares decomposition. In: Henrion, D., Garulli, A. (eds.) Positive Polynomials in Control, Lecture Notes in Control and Information Science, vol. 312, pp. 23–43. Springer Berlin Heidelberg (2005)
27. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. and Comput. 20(1), 309–352 (Feb 2010)
28. Platzer, A., Clarke, E.M.: Computing differential invariants of hybrid systems as fixedpoints. In: Gupta, A., Malik, S. (eds.) CAV 2008, LNCS, vol. 5123, pp. 176–189. Springer Berlin Heidelberg (2008)
29. Prajna, S., Jadbabaie, A., Pappas, G.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Transactions on Automatic Control 52(8), 1415–1428 (2007)
30. Ratschan, S.: Safety verification of non-linear hybrid systems is quasi-decidable. Formal Methods in System Design 44(1), 71–90 (2014)
31. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation-based abstraction refinement. ACM Trans. Embed. Comput. Syst. 6(1) (Feb 2007)
32. Rebiha, R., Matringe, N., Moura, A.V.: Transcendental inductive invariants generation for non-linear differential and hybrid systems. In: HSCC 2012. pp. 25–34. ACM, New York, NY, USA (2012)
33. Sankaranarayanan, S.: Automatic abstraction of non-linear systems using change of bases transformations. In: HSCC 2011. pp. 143–152. ACM, New York, NY, USA (2011)
34. Sankaranarayanan, S.: Change-of-bases abstractions for non-linear systems. CoRR abs/1204.4347 (2012), `http://arxiv.org/abs/1204.4347`
35. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constructing invariants for hybrid systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004, LNCS, vol. 2993, pp. 539–554. Springer Berlin Heidelberg (2004)
36. Savageau, M.A., Voit, E.O.: Recasting nonlinear differential equations as S-systems: a canonical nonlinear form. Mathematical Biosciences 87(1), 83–115 (1987)
37. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software 11-12, 625–653 (1999)
38. Stursberg, O., Kowalewski, S., Hoffmann, I., Preußig, J.: Comparing timed and hybrid automata as approximations of continuous systems. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S. (eds.) Hybrid Systems IV, LNCS, vol. 1273, pp. 361–377. Springer Berlin Heidelberg (1997)
39. Tiwari, A.: Abstractions for hybrid systems. Formal Methods in System Design 32(1), 57–83 (2008)
40. Toh, K.C., Todd, M., Tütüncü, R.H.: SDPT3 – a MATLAB software package for semidefinite programming. Optimization Methods and Software 11, 545–581 (1999)
41. Zhao, H., Yang, M., Zhan, N., Gu, B., Zou, L., Chen, Y.: Formal verification of a descent guidance control program of a lunar lander. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014, LNCS, vol. 8442, pp. 733–748. Springer International Publishing Switzerland (2014)