Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants

Citation for published version:
Sogokon, A, Jackson, P & Johnson, TT 2017, Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants. in NASA Formal Methods Symposium: NFM 2017: NASA Formal Methods. Lecture Notes in Computer Science, vol. 10227, Springer, Cham, pp. 194-211, 9th NASA Formal Methods Symposium , Moffett Field, United States, 16/05/17. DOI: 10.1007/978-3-319-57288-8_14

Digital Object Identifier (DOI):
10.1007/978-3-319-57288-8_14

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
 NASA Formal Methods Symposium

Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants *

Andrew Sogokon (0000-0002-5849-7991)[1], Paul B. Jackson (0000-0003-3863-8336)[2], and Taylor T. Johnson (0000-0001-8021-9923)[1]

[1] Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA
{andrew.sogokon|taylor.johnson}@vanderbilt.edu

[2] Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, UK
Paul.Jackson@ed.ac.uk

Abstract We propose a method for verifying persistence of nonlinear hybrid systems. Given some system and an initial set of states, the method can guarantee that system trajectories always eventually evolve into some specified target subset of the states of one of the discrete modes of the system, and always remain within this target region. The method also computes a time-bound within which the target region is always reached. The approach combines flow-pipe computation with deductive reasoning about invariants and is more general than each technique alone. We illustrate the method with a case study concerning showing that potentially destructive stick-slip oscillations of an oil-well drill eventually die away for a certain choice of drill control parameters. The case study demonstrates how just using flow-pipes or just reasoning about invariants alone can be insufficient. The case study also nicely shows the richness of systems that the method can handle: the case study features a mode with non-polynomial (nonlinear) ODEs and we manage to prove the persistence property with the aid of an automatic prover specifically designed for handling transcendental functions.

1 Introduction

Hybrid systems combine discrete and continuous behaviour and provide a very general framework for modelling and analyzing the behaviour of systems such as those implemented in modern embedded control software. Although a number of tools and methods have been developed for verifying properties of hybrid systems, most are geared towards proving bounded-time safety properties, often employing set reachability computations based on constructing over-approximating enclosures of the reachable states of ordinary differential equations (e.g. [7,14,13,21]). Methods capable of proving unbounded-time safety properties often rely (explicitly or otherwise) on constructing *continuous invariants* (e.g. [42,25], and referred to in short as *invariants*). Such invariants may be thought of as a generalization of *positively invariant sets* (see e.g. [5]) and

* This material is based upon work supported by the UK Engineering and Physical Sciences Research Council under grants EPSRC EP/I010335/1 and EP/J001058/1, the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258.

which are analogous to inductive invariants used in computer science to reason about the correctness of discrete programs using Hoare logic.

We argue in this paper that a combined approach employing bounded time reachability analysis and reasoning about invariants can be effective in proving *persistence* and *safety* properties in non-polynomial (nonlinear) hybrid systems. We illustrate the combined approach using a detailed case study with non-polynomial ODEs for which neither approach individually was sufficient to establish the desired safety and persistence properties.

Methods for bounded time safety verification cannot in general be applied to prove safety for all time and their accuracy tends to degrade for large time bounds, especially for nonlinear systems. Verification using invariants, while a powerful technique that can prove strong properties about nonlinear systems, relies on the ability to find invariants that are sufficient for proving the unbounded time safety property. In practice, many invariants for the system can be found which fall short of this requirement, often for the simple reason that they do not include all the initial states of the system. We show how a combined approach employing both verification methods can, in some cases, address these limitations.

Contributions.

In this paper we (**I**) show that bounded time safety verification based on flowpipe construction can be naturally combined with invariants to verify persistence and unbounded time safety properties, addressing some of the limitations of each verification method when considered in isolation. (**II**) To illustrate the approach, we consider a simplified torsional model of a conventional oil well drill string that has been the subject of numerous studies by Navarro-López et al. [34]. (**III**) We discuss some of the challenges that currently stand in the way of fully automatic verification using this approach. Additionally, we provide a readable overview of the methods employed in the verification process and the obstacles that present themselves when these methods are applied in practice.

2 Safety and Persistence for Hybrid Automata

2.1 Preliminaries

A number of formalisms exist for specifying hybrid systems. The most popular framework at present is that of hybrid automata [3,19], which are essentially discrete transition systems in which each discrete state represents an operating mode inside which the system evolves continuously according to an ODE under some evolution constraint. Additionally, transition guards and reset maps are used to specify the discrete transition behaviour (i.e. switching) between the operating modes. A sketch of the syntax and semantics of hybrid automata is as follows.

Definition 1 (Hybrid automaton [26]). *Formally, a hybrid automaton is given by* $(Q, Var, \boldsymbol{f}, Init, Inv, T, G, R)$, *where*

- $Q = \{q_0, q_1, \ldots, q_k\}$ *is a* finite *set of discrete states (modes),*
- $Var = \{x_1, x_2, \ldots, x_n\}$ *is a* finite *set of continuous variables,*
- $f : Q \times \mathbb{R}^n \to \mathbb{R}^n$ *gives the vector field defining continuous evolution inside each mode,*

- $\mathit{Init} \subset Q \times \mathbb{R}^n$ *is the set of initial states,*
- $\mathit{Inv} : Q \to 2^{\mathbb{R}^n}$ *gives the* mode invariants *constraining evolution for every discrete state,*
- $T \subseteq Q \times Q$ *is the transition relation,*
- $G : T \to 2^{\mathbb{R}^n}$ *gives the guard conditions for enabling transitions,*
- $R : T \to 2^{\mathbb{R}^n \times \mathbb{R}^n}$ *gives the reset map.*

A *hybrid state* of the automaton is of the form $(q, \boldsymbol{x}) \in Q \times \mathbb{R}^n$. A *hybrid time trajectory* is a sequence (which may be finite or infinite) of intervals $\tau = \{I_i\}_{i=0}^N$, for which $I_i = [\tau_i, \tau_i']$ for all $i < N$ and $\tau_i \leq \tau_i' = \tau_{i+1}$ for all i. If the sequence is finite, then either $I_N = [\tau_N, \tau_N']$ or $I_N = [\tau_N, \tau_N')$. Intuitively, one may think of τ_i as the times at which discrete transitions occur. An *execution* (or a *run* or *trajectory*) of a hybrid automaton defined to be $(\tau, q, \varphi_t^i(\boldsymbol{x}))$, where τ is a hybrid time trajectory, $q : \langle\tau\rangle \to Q$ (where $\langle\tau\rangle$ is defined to be the set $\{0, 1, \ldots, N\}$ if τ is finite and $\{0, 1, \ldots\}$ otherwise) and $\varphi_t^i(\boldsymbol{x})$ is a collection of diffeomorphisms $\varphi_t^i(\boldsymbol{x}) : I_i \to \mathbb{R}^n$ such that $(q(0), \varphi_0^0(\boldsymbol{x})) \in \mathit{Init}$, for all $t \in [\tau_i, \tau_i')$ $\dot{\boldsymbol{x}} = f(q(i), \varphi_t^i(\boldsymbol{x}))$ and $\varphi_t^i(\boldsymbol{x}) \in \mathit{Inv}(i)$. For all $i \in \langle\tau\rangle \setminus \{N\}$ it is also required that transitions respect the guards and reset maps, i.e. $e = (q(i), q(i+1)) \in T$, $\varphi_{\tau_i'}^i(\boldsymbol{x}) \in G(e)$ and $(\varphi_{\tau_i'}^i(\boldsymbol{x}), \varphi_{\tau_{i+1}}^{i+1}(\boldsymbol{x})) \in R(e)$.

We consider MTL[3] formulas satisfied by trajectories. The satisfaction relation is of form $\rho \models^p \phi$, read as "*trajectory ρ at position p satisfies temporal logic formula ϕ*", where positions on a trajectory are identified by pairs of form (i, t) where $i \leq N$ and time $t \in I_t$. We use the MTL modality $\square_I \phi$ which states that formula ϕ always holds in time interval I in the future. Formally, this can be defined as $\rho \models^p \square_I \phi \;\equiv\; \forall p' \geq p$ s.t. $(p'.2 - p.2) \in I.\, \rho \models^{p'} \phi$, where $(i', t') \geq (i, t) \equiv i' > i \vee (i' = i \wedge t' \geq t)$. Similarly we can define the modality $\Diamond_I \phi$ which states that formula ϕ eventually holds at some time in the time interval I in the future. An MTL formula is valid for a given hybrid automaton if it is satisfied by all trajectories of that automaton starting at position $(0, 0)$. For clarity when writing MTL formulas, we assume trajectories are not restricted to start in *Init* states and instead introduce *Init* predicates into the formulas when we want restrictions.

Alternative formalisms for hybrid systems, such as *hybrid programs* [41], enjoy the property of having a compositional semantics and can be used to verify properties of systems by verifying properties of their parts in a theorem prover [44,15]. Other formal modelling frameworks for hybrid systems, such as *Hybrid CSP* [24], have also found application in theorem provers [60,62].

2.2 Bounded Time Safety and Eventuality

The *bounded-time safety verification problem* (with some finite time bound $t > 0$) is concerned with establishing that given an initial set of states $\mathrm{Init} \subseteq Q \times \mathbb{R}^n$ and a set of safe states $\mathrm{Safe} \subseteq Q \times \mathbb{R}^n$, the state of the system may not leave Safe within time t along any valid trajectory τ of the system. In the absence of closed-form solutions to the ODEs, this property may be established by verified integration, i.e. by computing successive over-approximating enclosures (known as *flowpipes*) of the reachable states in discrete time steps. Bounded-time reachability analysis can be extended to full hy-

[3] Metric Temporal Logic; see e.g. [22].

brid systems by also computing/over-approximating the discrete reachable states (up to some finite bound on the number of discrete transitions).

A number of bounded-time verification tools for hybrid systems have been developed based on verified integration using interval enclosures. For instance, *iSAT-ODE*, a verification tool for hybrid systems developed by Eggers et al. [13] relies on the verified integration tool *VNODE-LP* by Nedialkov [37] for computing the enclosures. Other examples include *dReach*, a reachability analysis tool for hybrid systems developed by Kong et al. [21], which uses the *CAPD* library [1]. Over-approximating enclosures can in practice be very precise for small time horizons, but tend to become conservative when the time bound is large (due to the so-called *wrapping effect*, which is a problem caused by the successive build-up of over-approximation errors that arises in interval-based methods; see e.g. [38].) An alternative verified integration method using *Taylor models* was introduced by Makino and Berz (see [4,38]) and can address some of these drawbacks, often providing tighter enclosures of the reachable set. Implementations of the method have been reported in *COSY INFINITY*, a scientific computing tool by Makino and Berz [29]; *VSPODE*, a tool for computing validated solutions to parametric ODEs by Lin and Stadtherr [23]; and in *Flow**, a bounded-time verification for hybrid systems developed by Chen et al. [7].

Because flowpipes provide an over-approximation of the reachable states at a given time, verified integration using flowpipes can also be used to reason about *liveness* properties such as *eventuality*, i.e. when a system is guaranteed to eventually enter some *target set* having started off at some point in an initial set. The bounded-time safety and eventuality properties may be more concisely expressed by using MTL notation, i.e. by writing $\mathrm{Init} \rightarrow \Box_{[0,t]}\,\mathrm{Safe}$, and $\mathrm{Init} \rightarrow \Diamond_{[0,t]}\,\mathrm{Target}$, where Init describes the initial set of states, $\mathrm{Safe} \subseteq Q \times \mathbb{R}^n$ is the set of safe states and $\mathrm{Target} \subseteq Q \times \mathbb{R}^n$ is the target region which is to be eventually attained.

Remark 2. The bounded time eventuality properties we consider in this paper are more restrictive than the general (unbounded time) case. For instance, consider a continuous 2-dimensional system governed by $\dot{x}_1 = x_2, \dot{x}_2 = 0$ and confined to evolve in the region where $x_2 > 0$. If one starts this system inside a state where $x_1 = 0$, it will eventually evolve into a state where $x_1 = 1$ by following the solution, however one may not put a finite bound on the time for this to happen. Thus, while $x_1 = 0 \rightarrow \Diamond_{[0,\infty)}\, x_1 = 1$ is true for this system the bounded time eventuality property $x_1 = 0 \rightarrow \Diamond_{[0,t]}\, x_1 = 1$, will not hold for any finite $t > 0$.

2.3 Unbounded Time Safety

A safety property for unbounded time may be more concisely expressed using an MTL formula:

$$\mathrm{Init} \rightarrow \Box_{[0,\infty)}\,\mathrm{Safe}.$$

A proof of such a safety assertion is most commonly achieved by finding an appropriate *invariant*, $I \subseteq Q \times \mathbb{R}^n$, which contains no unsafe states (i.e. $I \subseteq \mathrm{Safe}$) and such that the state of the system may not escape from I into an unsafe state along any valid trajectory of the system. Invariance is a special kind of safety assertion and may be written as $\mathrm{I} \rightarrow \Box_{[0,\infty)}\,\mathrm{I}$. A number of techniques have been developed for proving invariance properties for continuous systems without the need to compute solutions to the ODEs [49,41,58,25,17,53].

2.4 Combining Unbounded Time Safety with Eventuality to Prove Persistence

In linear temporal logic, a *persistence* property states that a formula is 'eventually always' true. For instance, using persistence one may express the property that a system starting in any initial state always eventually reaches some target set and then always stays within this set. Using MTL notation, we can write this as:

$$\mathrm{Init} \rightarrow \Diamond_{[0,\infty)} \Box_{[0,\infty)} \mathrm{Target}.$$

Persistence properties generalize the concept of stability. With stability one is concerned with showing that the state of a system always converges to some particular equilibrium point. With persistence, one only requires that the system state eventually becomes always trapped within some set of states.

In this paper we are concerned with a slightly stronger form of persistence, where one ensures that the target set is always reached within some specified time t:

$$\mathrm{Init} \rightarrow \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Target}.$$

We observe that a way of proving this is to find a set $I \subseteq \mathrm{Target}$ such that:

1. $\mathrm{Init} \rightarrow \Diamond_{[0,t]} I$ holds, and
2. I is an invariant for the system.

This fact can be stated more formally as a rule of inference:

$$(\mathrm{Persistence})\ \frac{\mathrm{Init} \rightarrow \Diamond_{[0,t]} I \qquad I \rightarrow \Box_{[0,\infty)} I \qquad I \rightarrow \mathrm{Target}}{\mathrm{Init} \rightarrow \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Target}}\ .$$

Previous Sections 2.2 and 2.3 respectively surveyed how the eventuality premise $\mathrm{Init} \rightarrow \Diamond_{[0,t]} I$ and invariant premise $I \rightarrow \Box_{[0,\infty)} I$ can be established by a variety of automated techniques. In Section 5 we explore automation challenges further and remark on ongoing work addressing how to automatically generate suitable invariants I.

2.5 Using Persistence to Prove Safety

Finding appropriate invariants to prove unbounded time safety as explained above in Section 2.3 can in practice be very difficult. It might be the case that invariants $I \subseteq \mathrm{Safe}$ for the system can be found, but also ensuring that $\mathrm{Init} \subseteq I$ is infeasible. Nevertheless it might be the case that one of these invariants I is always eventually reached by trajectories starting in Init and all those trajectories are contained within Safe. In such cases, Safe is indeed a safety property of the system when starting from any point in Init. More precisely, if one can find an invariant I as explained above in Section 2.4 to show the persistence property: $\mathrm{Init} \rightarrow \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Safe}$, and further one can show for the same time bound t that: $\mathrm{Init} \rightarrow \Box_{[0,t]} \mathrm{Safe}$, then one has: $\mathrm{Init} \rightarrow \Box_{[0,\infty)} \mathrm{Safe}$. As a result, one may potentially utilize invariants that were by themselves insufficient for proving the safety property.

Remark 3. The problem of showing that a state satisfying $\Box_{[0,\infty)} \mathrm{Safe}$ is reached in finite time t, while ensuring that the formula $\Box_{[0,t]} \mathrm{Safe}$ also holds (i.e. states satisfying $\neg\mathrm{Safe}$ are avoided up to time t) is sometimes called a *reach-avoid problem* [61].

Even if one's goal is to establish bounded-time rather than unbounded-time safety properties, this inference scheme could still be of use, as it could significantly reduce the time bound t needed for bounded time reachability analysis. In practice, successive over-approximation of the reachable states using flowpipes tends to become conservative for large values of t. In highly non-linear systems one can realistically expect to compute flowpipes only for very modest time bounds (e.g. in chaotic systems flowpipes are guaranteed to 'blow up', but invariants may still sometimes be found). Instead, it may in some cases be possible to prove the safety property by computing flowpipes up to some small time bound, after which the system can be shown to be inside an invariant that implies the safety property for all times thereafter.

3 An example persistence verification problem

Stick-slip oscillations are commonly encountered in mechanical engineering in the context of modelling the effects of dynamic friction. Informally, the phenomenon manifests itself in the system becoming "stuck" and "unstuck" repeatedly, which results in unsteady "jerky" motions. In engineering practice, stick-slip oscillations can often degrade performance and cause failures when operating expensive machinery [36]. Although the problem of demonstrating absence of stick-slip oscillations in a system is primarily motivated by safety considerations, it would be misleading to call this a *safety verification problem*. Instead, the problem may broadly be described as that of demonstrating that the system (in finite time) enters a state in which no stick-slip motion is possible and remains there indefinitely. Using MTL one may write:

$$\mathrm{Init} \rightarrow \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Steady},$$

where Steady describes the states in which harmful oscillations cannot occur. The formula may informally be read as saying that "from any initial configuration, the system will eventually evolve within time t into a state region where it is always steady".

As an example of a system in which eventual absence of stick-slip oscillations is important, we consider a well-studied [34] model of a simplified conventional oil well drill string. The system can be characterized in terms of the following variables: φ_r, the angular displacement of the top rotary system; φ_b, the angular displacement of the drilling bit; $\dot{\varphi}_r$, the angular velocity of the top rotary system; and $\dot{\varphi}_b$, the angular velocity of the drilling bit. The continuous state of the system $\boldsymbol{x}(t) \in \mathbb{R}^3$ can be described in terms of these variables, i.e. $\boldsymbol{x}(t) = (\dot{\varphi}_r,\ \varphi_r - \varphi_b,\ \dot{\varphi}_b)^T$. The system has two control parameters: W_{ob} giving the weight applied on the drilling bit, and $u = T_m$ giving the surface motor torque. The dynamics is governed a non-linear system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, given by:

$$\dot{x}_1 = \frac{1}{J_r}\Big(-(c_t + c_r)x_1 - k_t x_2 + c_t x_3 + u\Big), \tag{1}$$

$$\dot{x}_2 = x_1 - x_3, \tag{2}$$

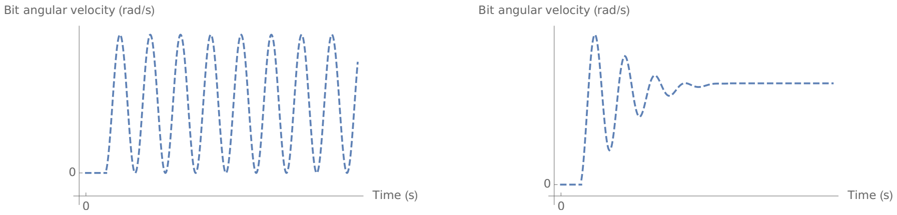
$$\dot{x}_3 = \frac{1}{J_b}\Big(c_t x_1 + k_t x_2 - (c_t + c_b)x_3 - T_{f_b}(x_3)\Big). \tag{3}$$

The term $T_{f_b}(x_3)$ denotes the friction modelling the bit-rock contact and is responsible for the non-polynomial non-linearity. It is given by

$$W_{ob} R_b \Big(\mu_{c_b} + (\mu_{s_b} - \mu_{c_b}) e^{-\frac{\gamma_b}{\nu_f}|x_3|} \Big) \mathrm{sgn}(x_3),$$

where $\mathrm{sgn}(x_3) = \frac{x_3}{|x_3|}$ if $x_3 \neq 0$ and $\mathrm{sgn}(x_3) \in [-1, 1]$ if $x_3 = 0$. Constants used in the model [34] are as follows: $c_b = 50\,\mathrm{Nms/rad}$, $k_t = 861.5336$ Nm/rad, $J_r = 2212\,\mathrm{kg\,m}^2$, $J_b = 471.9698\ \mathrm{kg\,m}^2$, $R_b = 0.155575\,\mathrm{m}$, $c_t = 172.3067$ Nms/rad, $c_r = 425\,\mathrm{Nms/rad}$, $\mu_{cb} = 0.5$, $\mu_{sb} = 0.8$, $\gamma_b = 0.9$, $\nu_f = 1$ rad/s. Even though at first glance the system looks like a plain continuous system with a single set of differential equations, it is effectively a hybrid system with at least 3 modes, where the drilling bit is: "rotating forward" ($x_3 > 0$), "stopped" ($x_3 = 0$), and "rotating backward" ($x_3 < 0$). A sub-mode of the stopped mode models when the drill bit is stuck. In this sub-mode, the torque components on the drill bit due to c_t, c_b and k_t are insufficient to overcome the static friction $W_{ob} R_b \mu_{c_b}$, and $\mathrm{sgn}(x_3)$ is further constrained so as to ensure $\dot{x}_3 = 0$.

Once the drill is in operation, so-called *stick-slip oscillations* can cause damage when the bit repeatedly becomes stuck and unstuck due to friction in the bottom hole assembly. In the model this behaviour would correspond to the system entering a state where $x_3 = 0$ repeatedly. The objective is to verify the eventual absence of stick-slip oscillations in the system initialised at the origin (i.e. at rest) for some given choice of the control parameters W_{ob} and u. Previous work by Navarro-López and Carter [34] explored modelling the simplified model of the drill as a hybrid automaton and simulated the resulting models in Stateflow and Modelica.



(a) Stick-slip motion (undesirable)

(b) Stabilization (desired behaviour)

Figure 1: Simulations can exhibit stabilization with positive bit angular velocity and stick-slip bit motion.

Simulations, such as those obtained in [34], using different models and control parameters for the drill can suggest stick-slip oscillations or their absence (illustrated in Fig. 1) in a particular model, however the task of verifying their eventual absence cannot be adequately addressed with simulation alone. In practice however, simulation is incredibly useful in providing some degree of confidence in the overall result, which is very important to know before attempting verification.

A simulation of the system with a concrete choice for the control parameters $W_{ob} = 50,000$ N and $u = 6,000$ Nm, shown as a trajectory in the 3-dimensional state space in Fig 3a, suggests that the system does not exhibit stick-slip oscillations, because the trajectory is observed to start at the origin, escape the surface ($x_3 = 0$)[4] and stabilize around a point where the angular velocity of the drilling bit is positive ($x_3 > 0$).

4 Verifying Persistence

The property of interest, i.e. the eventual absence of stick-slip oscillation that we observe in the simulation, may be phrased as the following formula in metric temporal logic: $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \rightarrow \Diamond_{[0,t]} \, \Box_{[0,\infty)} \, x_3 > 0$, which informally asserts that the system initialised at the origin will *eventually* (diamond modality) enter a state where it is *always* (box modality) the case that $x_3 > 0$. In the following sections we describe a method for proving this assertion. Following our approach, we break the problem down into the following two sub-problems:

1. Finding an appropriate invariant I in which the property $\Box_{[0,t]} \, x_3 > 0$ holds. For this we employ continuous/positive invariants, discussed in the next section.
2. Proving that the system reaches a state in the set I in finite time when initialised at the origin, i.e. $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \rightarrow \Diamond_{[0,t]} \, I$. [5]

4.1 Continuous Invariant

Finding continuous invariants that are sufficient to guarantee a given property is in practice remarkably difficult. Methods for automatic continuous invariant generation have been reported by numerous authors [49,59,18,53,52,25,63,16,30,54], but in practice often result in “coarse” invariants that cannot be used to prove the property of interest, or require an unreasonable amount of time due to their reliance on expensive real quantifier elimination algorithms.

Stability analysis (involving a linearisation; see [56] for details) can be used to suggest a polynomial function $V : \mathbb{R}^n \to \mathbb{R}$, given by

$$\begin{aligned} V(\boldsymbol{x}) = {} & 50599.6 - 14235.7x_1 + 1234.22x_1^2 - 4351.43x_2 + 342.329x_1x_2 \\ & + 288.032x_2^2 - 3865.81x_3 + 367.657x_1x_3 + 18.2594x_2x_3 + 241.37x_3^2, \end{aligned}$$

for which we can reasonably conjecture that $V(\boldsymbol{x}) \leq 1400$ defines a positively invariant set under the flow of our non-linear system. Geometrically, this represents an ellipsoid that lies above the surface defined by $x_3 = 0$ in the state space (see Fig. 3b). In order to prove the invariance property, it is sufficient to show that the following holds:[6]

$$\forall \, \boldsymbol{x} \in \mathbb{R}^3. \; V(\boldsymbol{x}) = 1400 \rightarrow \nabla V \cdot f(\boldsymbol{x}) < 0. \tag{4}$$

Unfortunately, in the presence of non-polynomial terms[7] a first order sentence will in general not belong to a decidable theory [51], although there has recently been progress in broadening the scope of the popular CAD algorithm [9] for real quantifier elimination to work with restricted classes of non-polynomial problems [57].

[4] The system exhibits *sliding behaviour* on a portion of this surface known as the *sliding set*. See [34].

[5] Files for the case study are available online. `http://www.verivital.com/nfm2017`

[6] Here ∇ denotes the *gradient* of V, i.e. the vector of partial derivatives $(\frac{\partial V}{\partial x_1}, \ldots, \frac{\partial V}{\partial x_n})$.

[7] E.g. those featured in the right-hand side of the ODE, i.e. $f(\boldsymbol{x})$.

In practice, this conjecture is easily proved in under 5 seconds using MetiTarski, an automatic theorem prover, developed by L.C. Paulson and co-workers at the University of Cambridge, designed specifically for proving universally quantified first order conjectures featuring transcendental functions (such as $\sin$,$\cos$, $\ln$, $\exp$, etc.) The interested reader may find more details about the MetiTarski system in [2,40].

Remark 4. Although Wolfram's *Mathematica* 10 computer algebra system also provides some functionality for proving first-order conjectures featuring non-polynomial expressions using its `Reduce[]` function, we were unable (on our system[8]) to prove conjecture (4) this way after over an hour of computation, after which the Mathematica kernel crashed.

The automatic proof of conjecture (4) obtained using MetiTarski (provided we trust the system) establishes that $V(\boldsymbol{x}) \leq 1400$ defines a positively invariant set, and thus we are guaranteed that solutions initialised inside this set remain there at all future times. In order to be certain that no outgoing discrete transitions of the hybrid system are possible when the system is evolving inside $V(\boldsymbol{x}) \leq 1400$, we further require a proof of the following conjecture featuring only polynomial terms:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^3.\ V(\boldsymbol{x}) \leq 1400 \rightarrow x_3 > 0. \tag{5}$$

An automatic proof of this conjecture may be obtained using an implementation of a decision procedure for first-order real arithmetic.

4.2 Verified Integration

In order to show that the system does indeed enter the positively invariant ellipsoid $V(\boldsymbol{x}) \leq 1400$ in finite time, it is not sufficient to observe this in a simulation (as in Fig. 3b), which is why we use a tool employing *verified integration* based on Taylor models. *Flow** (implemented by Chen et al. [7]) is a bounded-time safety verification tool for hybrid systems that computes Taylor models to analyze continuous reachability. The tool works by computing successive over-approximations (flowpipes) of the reachable set of the system, which are internally represented using Taylor models (but which may in turn be over-approximated by a bounding hyper-box and easily rendered).

Fig. 2a shows the bounding boxes of solution enclosures computed from the point initial condition at the origin using Flow* with adaptive time steps and Taylor models of order 13, a time bound of 12.7 and the same control parameters used in the simulation (i.e. $u = 6,000$ Nm, $W_{ob} = 50,000$ N). We observe that once solutions escape to the region where $x_3 > 0$, they maintain a positive x_3 component for the duration of the time bound.

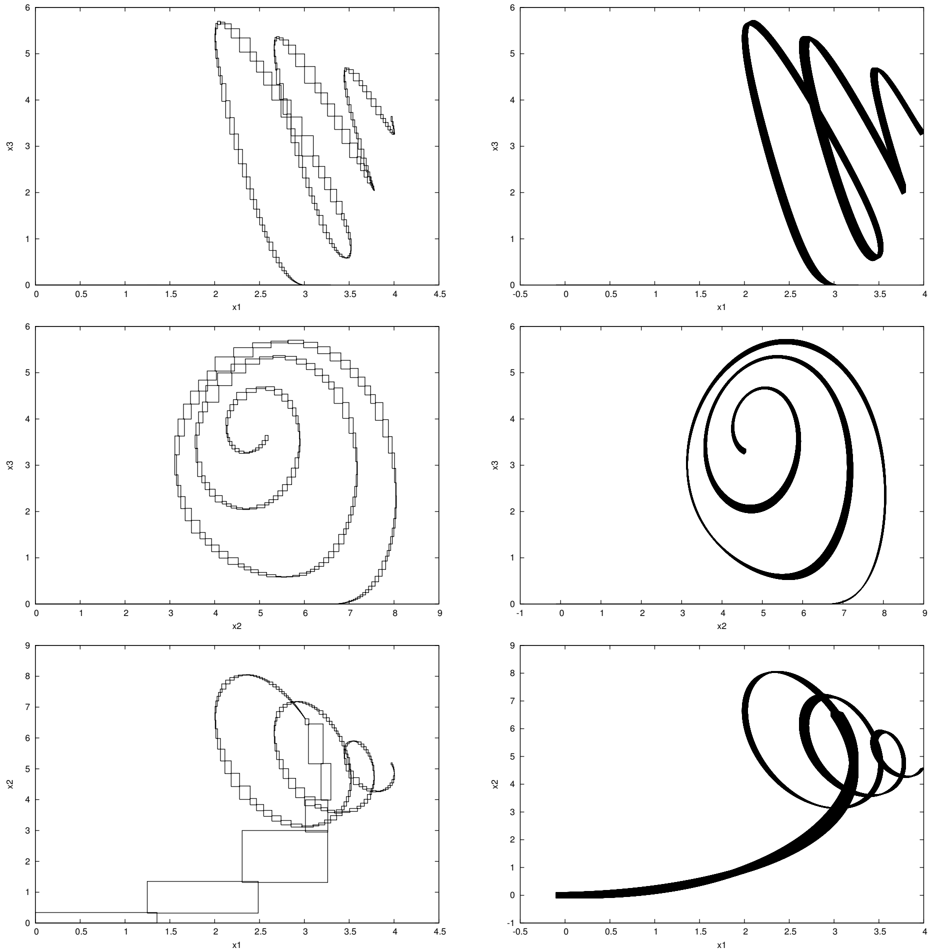
The last flowpipe computed by Flow* for this problem can be bounded inside the hyper-rectangle $\mathrm{BoundBox}$ characterized by the formula

$$\mathrm{BoundBox} \equiv \frac{39}{10} \leq x_1 \leq 4 \wedge \frac{51}{10} \leq x_2 \leq \frac{26}{5} \wedge \frac{7}{2} \leq x_3 \leq \frac{37}{10}.$$

Once more, using a decision procedure for real arithmetic, we can check that the following sentence is true:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^3.\ \mathrm{BoundBox} \rightarrow V(\boldsymbol{x}) \leq 1400.$$

[8] Intel i5-2520M CPU @ 2.50GHz, 4GB RAM, running Arch Linux kernel 4.2.5-1.



(a) Verified integration up to time $t = 12.7$ from a point initial condition at the origin.

(b) Verified integration up to time $t = 12.2$ from an interval initial condition.

Figure 2: Verified integration using Flow*.

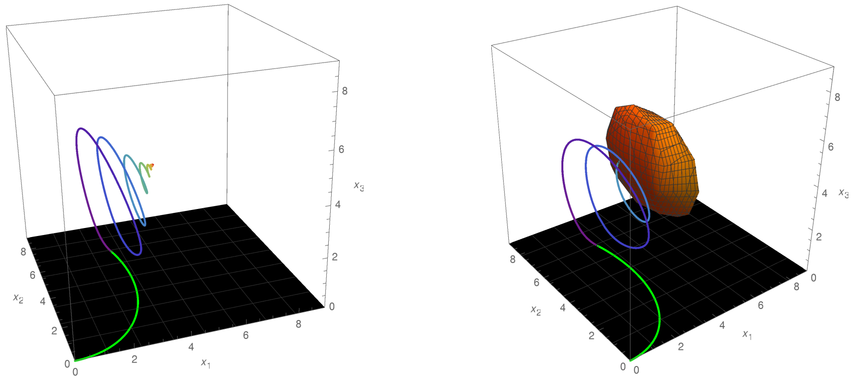
If we are able to establish the following facts:

1. $I \rightarrow \square_{[0,\infty)} I$ (I is a continuous invariant),
2. $I \rightarrow \mathrm{Steady}$ (inside I, there are no harmful oscillations), and
3. $\mathrm{Init} \rightarrow \Diamond_{[0,t]} I$ (the system enters the region I in finite time),

then we can conclude that $\mathrm{Init} \rightarrow \Diamond_{[0,t]} \square_{[0,\infty)} \mathrm{Steady}$ is also true and the system does not exhibit harmful stick-slip oscillations when started inside Init. By taking Init to be the origin $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0$, I to be the positively invariant sub-level set $V(\boldsymbol{x}) \leq 1400$ and Steady to be $x_3 > 0$, we are able to conclude the temporal property:

$$x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \rightarrow \Diamond_{[0,t]} \square_{[t,\infty)} x_3 > 0.$$

Verified integration using Taylor models also allows us to consider *sets* of possible initial conditions, rather than initial points (illustrated in Fig. 2b). This is useful when there is uncertainty about the system's initial configuration; however, in practice this comes with a significant performance overhead for verified integration.



(a) Simulation showing stabilization with positive bit angular velocity.

(b) Simulation showing eventual entry into an ellipsoidal invariant.

Figure 3: Simulation of the hybrid system initialised at the origin with $W_{ob} = 50,000$ N and $u = 6000$ Nm. The trajectory is contained by the flowpipes shown in Fig. 2a and is observed to enter the positively invariant ellipsoid $V(\boldsymbol{x}) \leq 1400$, illustrating the persistence property of eventual absence of stick-slip oscillations.

5 Outlook and Challenges to Automation

Correctness of reachability analysis tools based on verified integration is a soundness critical to the overall verification approach, which makes for a strong case in favour of using formally verified implementations. At present few are available, e.g. see recent work by Immler [20] which presented a formally verified continuous reachability algorithm based on adaptive Runge-Kutta methods. Verified implementations of Taylor model-based reachability analysis algorithms for continuous and hybrid systems would clearly be very valuable. One alternative to over-approximating reachable sets of continuous systems using flowpipes is based on simulating the system using a finite set of sampling trajectories and employs *sensitivity analysis* to address the coverage problem. This technique was explored by Donzé and Maler in [10]. A similar approach employing *matrix measures* has more recently been studied by Maidens and Arcak [28,27].

As an alternative to using verified integration, a number of deductive methods are available for proving eventuality properties in continuous and hybrid systems (e.g. [42,55]). These approaches can be much more powerful since they allow one to work with more general classes of initial and target regions that are necessarily out of scope for methods based on verified integration (e.g. they can work with initial sets that are unbounded, disconnected, etc.) Making effective use of the deductive verification tools currently in existence typically requires significant input and expertise on part of the user (finding the right invariants being one of the major stumbling blocks in practice), in stark contrast to the near-complete level of automation offered by tools based on verified integration. Methods for automatic continuous invariant generation are crucial to the mechanization of the overall verification approach. Progress on this problem would be hugely enabling for non-experts and specialists alike, as it would relieve them from the task of manually constructing appropriate invariants, which often requires intuition and expertise. Work in this area is ongoing (see e.g. [43,25,54]). Indeed, progress on this problem is also crucial to providing a greater level of automation in deductive verification tools.

6 Related Work

Combining elements of qualitative and quantitative reasoning[9] to study the behaviour of dynamical systems has previously been explored in the case of planar systems by Nishida et al. [39]. The idea of combining bounded-time reachability analysis with qualitative analysis in the form of discrete abstraction was investigated by Clarke et al. in [8]. Similar ideas are employed by Carter [6] and Navarro-López in [35], where the concept of *deadness* is introduced and used as a way of disproving liveness properties. Intuitively, deadness is a formalization of an idea that inside certain regions the system cannot be live, i.e. some desired property may never become true as the system evolves inside a "deadness region". These ideas were used in a case study [6, Chapter 5] also featuring the drill system studied in [34], but with a different set of control parameters and in which the verification objective was to prove the existence of a *single trajectory* for which the drill eventually gets "stuck", which is sufficient to disprove the liveness (oscillation) property.

Region stability is similar to our notion of persistence [45], which requires all trajectories to eventually reach some region of the state space. Sound and complete proof rules for establishing region stability have been explored and automated [47], as have more efficient encodings of the proof rule that scale better in dimensionality [31]. However, all algorithms we are aware of for checking region stability require linear or simpler (timed or rectangular) ODEs [45,47,46,31,11,48]. Strong attractors are basins of attraction where every state in the state space eventually reaches a region of the state space [45]. Some algorithms do not check region stability, but actually check stronger properties such as strong attraction, that imply region stability [45]. In contrast to these works, our method checks the weaker notion of persistence for nonlinear ODEs.

She and Ratschan studied methods of proving set eventuality in continuous systems under constraints using Lyapunov-like functions [50]. Duggirala and Mitra also employed Lyapunov-like function concepts to prove inevitability properties in hybrid systems [12]. Möhlmann et al. developed Stabhyil [33], which can be applied to nonlinear hybrid systems and checks classical notions of Lyapunov stability, which is a strictly stronger property than persistence. In [32] Möhlmann et al. extended their work and applied similar ideas, using information about (necessarily invariant) sub-level sets of Lyapunov functions to terminate reachability analysis used for safety verification. Prabhakar and Soto have explored abstractions that enable proving stability properties without having to search for Lyapunov functions, albeit these are not currently applicable to nonlinear systems [48]. In summary, in contrast to other works listed above, our approach enables proving persistence properties in conjunction with safety properties

[9] e.g numerical solution computation with "qualitative" features, such as invariance of certain regions.

for nonlinear, non-polynomial hybrid systems and does not put restrictions on the form or the type of the invariant used in conjunction with bounded time reachability analysis.

7 Conclusion

This paper explored a combined technique for safety and persistence verification employing continuous invariants and reachable set computation based on constructing flowpipes. The approach was illustrated on a model of a simplified oil well drill string system studied by Navarro-López et al., where the verification objective is to prove absence of damaging stick-slip oscillations. The system was useful in highlighting many of the existing practical challenges to applying and automating the proposed verification method. Many competing approaches already exist for verifying safety in hybrid systems, but these rarely combine different methods for reachability analysis and deductive verification, which our approach combines. We demonstrate that a combination of different approaches can be more practically useful than each constituent approach taken in isolation.

Acknowledgements The authors wish to thank to the anonymous reviewers for their careful reading and valuable suggestions for improving this paper.

References

1. CAPD library. Online `http://capd.ii.uj.edu.pl/`
2. Akbarpour, B., Paulson, L.C.: MetiTarski: An automatic theorem prover for real-valued special functions. Journal of Automated Reasoning 44(3), 175–205 (2010)
3. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. pp. 209–229 (1992)
4. Berz, M., Makino, K.: Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. Reliable Computing 4(4), 361–369 (1998)
5. Blanchini, F.: Set invariance in control. Automatica 35(11), 1747–1767 (1999)
6. Carter, R.A.: Verification of liveness properties on hybrid dynamical systems. Ph.D. thesis, University of Manchester, School of Computer Science (2013)
7. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: CAV. pp. 258–263 (2013)
8. Clarke, E.M., Fehnker, A., Han, Z., Krogh, B.H., Ouaknine, J., Stursberg, O., Theobald, M.: Abstraction and counterexample-guided refinement in model checking of hybrid systems. International Journal of Foundations of Computer Science 14(4), 583–604 (2003)
9. Collins, G.E.: Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages. pp. 134–183 (1975)
10. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: HSCC. pp. 174–189 (2007)
11. Duggirala, P.S., Mitra, S.: Abstraction refinement for stability. In: 2011 IEEE/ACM International Conference on Cyber-Physical Systems, ICCPS. Proceedings, pp. 22–31 (Apr 2011)
12. Duggirala, P.S., Mitra, S.: Lyapunov abstractions for inevitability of hybrid systems. In: HSCC. pp. 115–124. ACM, New York, NY, USA (2012)
13. Eggers, A., Ramdani, N., Nedialkov, N.S., Fränzle, M.: Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. Software and System Modeling 14(1), 121–148 (2015)
14. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable Verification of Hybrid Systems. In: CAV (2011)
15. Fulton, N., Mitsch, S., Quesel, J.D., Völp, M., Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In: CADE (2015)

16. Ghorbal, K., Platzer, A.: Characterizing algebraic invariants by differential radical invariants. In: TACAS. pp. 279–294 (2014)
17. Ghorbal, K., Sogokon, A., Platzer, A.: A hierarchy of proof rules for checking differential invariance of algebraic sets. In: VMCAI. pp. 431–448 (2015)
18. Gulwani, S., Tiwari, A.: Constraint-based approach for analysis of hybrid systems. In: Gupta, A., Malik, S. (eds.) CAV, LNCS, vol. 5123, pp. 190–203. Springer (2008)
19. Henzinger, T.A.: The theory of hybrid automata. pp. 278–292. IEEE Comp. Soc. Press (1996)
20. Immler, F.: Verified reachability analysis of continuous systems. In: TACAS (2015)
21. Kong, S., Gao, S., Chen, W., Clarke, E.M.: dReach: δ-reachability analysis for hybrid systems. In: TACAS 2015. pp. 200–205 (2015)
22. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems 2(4), 255–299 (1990)
23. Lin, Y., Stadtherr, M.A.: Validated solutions of initial value problems for parametric ODEs. Applied Numerical Mathematics 57(10), 1145–1162 (2007)
24. Liu, J., Lv, J., Quan, Z., Zhan, N., Zhao, H., Zhou, C., Zou, L.: A calculus for hybrid CSP. In: Programming Languages and Systems, pp. 1–15 (2010)
25. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT. pp. 97–106. ACM (2011)
26. Lygeros, J., Johansson, K.H., Simić, S.N., Zhang, J., Sastry, S.S.: Dynamical properties of hybrid automata. IEEE Transactions on Automatic Control 48(1), 2–17 (2003)
27. Maidens, J.N., Arcak, M.: Reachability analysis of nonlinear systems using matrix measures. IEEE Transactions on Automatic Control 60(1), 265–270 (Jan 2015)
28. Maidens, J.N., Arcak, M.: Trajectory-based reachability analysis of switched nonlinear systems using matrix measures. In: CDC. pp. 6358–6364 (Dec 2014)
29. Makino, K., Berz, M.: Cosy infinity version 9. Nuclear Instruments and Methods in Physics Research Section A 558(1), 346–350 (2006)
30. Matringe, N., Moura, A.V., Rebiha, R.: Generating invariants for non-linear hybrid systems by linear algebraic methods. In: SAS. pp. 373–389 (2010)
31. Mitrohin, C., Podelski, A.: Composing stability proofs for hybrid systems. In: Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS. Proceedings, pp. 286–300 (2011)
32. Möhlmann, E., Hagemann, W., Theel, O.E.: Hybrid tools for hybrid systems - proving stability and safety at once. In: FORMATS. pp. 222–239 (2015)
33. Möhlmann, E., Theel, O.: Stabhyli: A tool for automatic stability verification of non-linear hybrid systems. In: HSCC. pp. 107–112. ACM (2013)
34. Navarro-López, E.M., Carter, R.: Hybrid automata: an insight into the discrete abstraction of discontinuous systems. International Journal of Systems Science 42(11), 1883–1898 (2011)
35. Navarro-López, E.M., Carter, R.: Deadness and how to disprove liveness in hybrid dynamical systems. Theor. Comput. Sci. 642(C), 1–23 (Aug 2016)
36. Navarro-López, E.M., Suárez, R.: Practical approach to modelling and controlling stick-slip oscillations in oilwell drillstrings. In: Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on. vol. 2, pp. 1454–1460. IEEE (2004)
37. Nedialkov, N.S.: Interval Tools for ODEs and DAEs. In: SCAN (2006)
38. Neher, M., Jackson, K.R., Nedialkov, N.S.: On Taylor model based integration of ODEs. SIAM Journal on Numerical Analysis 45(1), 236–262 (2007)
39. Nishida, T., Mizutani, K., Kubota, A., Doshita, S.: Automated phase portrait analysis by integrating qualitative and quantitative analysis. In: Proceedings of the 9th National Conference on Artificial Intelligence. pp. 811–816 (1991)
40. Paulson, L.C.: MetiTarski: Past and Future. In: Beringer, L., Felty, A. (eds.) Interactive Theorem Proving, LNCS, vol. 7406, pp. 1–10. Springer Berlin Heidelberg (2012)

41. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reasoning 41(2), 143–189 (2008)
42. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. 20(1), 309–352 (2010)
43. Platzer, A., Clarke, E.M.: Computing differential invariants of hybrid systems as fixedpoints. In: CAV. pp. 176–189 (2008)
44. Platzer, A., Quesel, J.D.: KeYmaera: A hybrid theorem prover for hybrid systems. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR. pp. 171–178 (2008)
45. Podelski, A., Wagner, S.: Model checking of hybrid systems: From reachability towards stability. In: HSCC. Proceedings, pp. 507–521 (2006)
46. Podelski, A., Wagner, S.: Region stability proofs for hybrid systems. In: FORMATS. Proceedings, pp. 320–335 (2007)
47. Podelski, A., Wagner, S.: A sound and complete proof rule for region stability of hybrid systems. In: HSCC. Proceedings, pp. 750–753. Springer (2007)
48. Prabhakar, P., Garcia Soto, M.: Abstraction based model-checking of stability of hybrid systems. In: Computer Aided Verification - 25th International Conference, CAV. Proceedings. pp. 280–295 (2013)
49. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: HSCC. pp. 477–492. Springer (2004)
50. Ratschan, S., She, Z.: Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions. SIAM J. Control and Optimization 48(7), 4377–4394 (Jul 2010)
51. Richardson, D.: Some undecidable problems involving elementary functions of a real variable. Journal of Symbolic Logic 33(4), 514–520 (12 1968)
52. Sankaranarayanan, S.: Automatic invariant generation for hybrid systems using ideal fixed points. In: HSCC. pp. 221–230 (2010)
53. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constructing invariants for hybrid systems. FMSD 32(1), 25–55 (2008)
54. Sogokon, A., Ghorbal, K., Jackson, P.B., Platzer, A.: A method for invariant generation for polynomial continuous systems. In: VMCAI 2016. pp. 268–288 (2016)
55. Sogokon, A., Jackson, P.B.: Direct formal verification of liveness properties in continuous and hybrid dynamical systems. In: FM 2015. pp. 514–531 (2015)
56. Sogokon, A., Jackson, P.B., Johnson, T.T.: Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants. Tech. rep., Vanderbilt University (2017)
57. Strzeboński, A.W.: Cylindrical decomposition for systems transcendental in the first variable. J. Symb. Comput. 46(11), 1284–1290 (2011)
58. Taly, A., Tiwari, A.: Deductive verification of continuous dynamical systems. In: Kannan, R., Kumar, K.N. (eds.) FSTTCS. LIPIcs, vol. 4, pp. 383–394. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2009)
59. Tiwari, A.: Generating box invariants. In: Egerstedt, M., Mishra, B. (eds.) HSCC, LNCS, vol. 4981, pp. 658–661. Springer (2008)
60. Wang, S., Zhan, N., Zou, L.: An Improved HHL Prover: An Interactive Theorem Prover for Hybrid Systems. In: ICFEM. pp. 382–399 (2015)
61. Xue, B., Easwaran, A., Cho, N.J., Fränzle, M.: Reach-avoid verification for nonlinear systems based on boundary analysis. IEEE Transactions on Automatic Control (2016)
62. 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: FM. pp. 733–748 (2014)
63. Zhao, H., Zhan, N., Kapur, D.: Synthesizing switching controllers for hybrid systems by generating invariants. In: Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday. pp. 354–373 (2013)