Piecewise Robust Barrier Tubes for Nonlinear Hybrid Systems with Uncertainty

Hui Kong[1(✉)], Ezio Bartocci[3], Yu Jiang[4], and Thomas A. Henzinger[2]

[1] Max-Planck-Institute for Software Systems, Kaiserslautern, Germany
hkong@mpi-sws.org
[2] IST Austria, Klosterneuburg, Austria
[3] TU Wien, Vienna, Austria
[4] Tsinghua University, Beijing, China

Abstract. Piecewise Barrier Tubes (PBT) is a new technique for flowpipe overapproximation for nonlinear systems with polynomial dynamics, which leverages a combination of barrier certificates. PBT has advantages over traditional time-step based methods in dealing with those nonlinear dynamical systems in which there is a large difference in speed between trajectories, producing an overapproximation that is time independent. However, the existing approach for PBT is not efficient due to the application of interval methods for enclosure-box computation, and it can only deal with continuous dynamical systems without uncertainty. In this paper, we extend the approach with the ability to handle both continuous and hybrid dynamical systems with uncertainty that can reside in parameters and/or noise. We also improve the efficiency of the method significantly, by avoiding the use of interval-based methods for the enclosure-box computation without loosing soundness. We have developed a C++ prototype implementing the proposed approach and we evaluate it on several benchmarks. The experiments show that our approach is more efficient and precise than other methods in the literature.

1 Introduction

Hybrid systems (HS) [21] are a suitable mathematical framework to model dynamical systems with both discrete and continuous dynamics. This formalism has been successfully adopted to design cyber-physical systems (CPS) whose behavior is characterized by an embedded software monitoring and/or controlling a physical substratum. Formal verification of HS has indeed a practical impact in engineering by assuring important safety-critical requirements at design-time.

This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23, S11405-N23 (RiSE/SHiNE), ADynNet (P28182), and Z211-N23 (Wittgenstein Award) and the Deutsche Forschungsgemeinschaft project 389792660-TRR 248.

É. André and M. Stoelinga (Eds.): FORMATS 2019, LNCS 11750, pp. 123–141, 2019.
https://doi.org/10.1007/978-3-030-29662-9_8

Despite the great effort to advance the state-of-the-art, reachability analysis of HS remains one of the most challenging verification tasks. Although the problem of reachability analysis is in general undecidable [21] for HS, in the last decade several efficient and scalable semidecidable approaches have been proposed to analyse HS with linear dynamics [14,15,19,22,23,36,41].

HS with nonlinear ordinary differential equations (ODEs) remains still very challenging to solve because these ODEs do not have a closed form solution in general. One common strategy to tackle this problem is to compute an over-approximation (also called flowpipe) that contains all the possible trajectories originating from an initial set of states within a bounded-time horizon [1,6,9–11]. If the overapproximation does not intersect with the unsafe set of states, then the system is safe. However, if the overapproximation is too coarse, it may intersect the unsafe set of states only due to the approximation errors and then the verdict about safety may be inconclusive. Thus, one of the main problem to address is *how to efficiently compute tight over-approximations of the reachable set of states for nonlinear continuous and hybrid systems.*

To overcome this problem, in a recent paper [24], we have introduced the notion of Piecewise Barrier Tubes (PBT), a new flowpipe overapproximation for nonlinear systems with polynomial dynamics. The main idea of this approach is that for each segment of a flowpipe, it constructs a coarse box that is big enough to contain the segment and then it computes in the box a set of barrier functions [26,34] which work together to form a tube surrounding the flowpipe.

PBT has advantages over traditional time-step based methods in dealing with those nonlinear dynamical systems in which there is a large difference in speed between trajectories, producing a tight over-approximation that is time independent. However, the approach in [24] cannot handle uncertainty and hybrid systems. In addition, the use of interval method for enclosure-box computation reduces its efficiency.

In this paper, we extend the approach with the ability to handle both continuous and hybrid dynamical systems with uncertainty which can reside in parameters and/or noise. We improve the efficiency of the method significantly, by avoiding the use of interval method for enclosure-box computation without loosing soundness. We have developed a C++ prototype implementing the proposed approach and we evaluate it on several benchmarks. The experiments show that our approach is more efficient and precise than other methods proposed in the literature.

The other existing techniques used to compute a bounded flowpipe are mainly based on interval method [32] or Taylor model [4]. Interval method is quite efficient even for high dimensional systems [32], but it suffers from the *wrapping effect* that arises due to an uncontrollable growth of the interval enclosure that accumulates overapproximation errors. The use of Taylor model is more precise because it uses a vector of polynomials plus a vector of small intervals to symbolically represent the flowpipe. However, checking the intersection with the unsafe region requires generally the use of interval method that brings back the wrapping effect. In particular, the wrapping effect can explode easily when the flowpipe segment over a time interval is stretched drastically due to a large difference in speed between individual trajectories.

Only recently, tools such as CLRT [9,10], Flow* [6], MathSAT SMT solver [7,8], HySAT/iSAT [12], dReach [27], C2E2 [11] and CORA [1], have made some progresses in verifying nonlinear continuous and hybrid models. Some of these tools [7,12,27] are based on decision procedures that overcome the theoretical limits in nonlinear theories over the reals. The main idea is to encode the reachability problem for nonlinear systems as first-order logic formulas over the real numbers. A satisfiability modulo theories (SMT) solver implementing such procedures can return either a verdict of unsatisfiability when the unsafe region is not reached or an inconclusive verdict [12,27] such as δ-sat if the problem is satisfiable given a certain precision δ (the same problem may result unsatisfiable by increasing the precision). However, in the case of unsatisfiability these tools generally do not provide a reachable set representation that explains the verdict. Other techniques for reachability analysis of nonlinear systems include invariant generation [25,31,38,39,42], abstraction and hybridization [2,5,16,28,33,37].

The paper is organized as follows. Section 2 presents the necessary preliminaries. Section 3 shows how to compute robust barrier certificates using linear programming, while in Sect. 4 we present our approach to address the reachability analysis problem of nonlinear continuous and hybrid systems with uncertainty. Section 5 provides our experimental results.

2 Preliminaries

In this section, we recall some concepts used throughout the paper. We first clarify some notation conventions. If not specified otherwise, we use boldface lower case letters to denote vectors, we use $\mathbb{R}$ for the real numbers field and $\mathbb{N}$ for the set of natural numbers, and we consider multivariate polynomials in $\mathbb{R}[\boldsymbol{x}]$, where the components of $\boldsymbol{x}$ act as indeterminates. In addition, for all the polynomials $B(\boldsymbol{c}, \boldsymbol{x})$, we denote by $\boldsymbol{c}$ the vector composed of all the c_i and denote by $\boldsymbol{x}$ the vector composed of all the remaining variables x_i that occur in the polynomial. We use $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ to denote the domain of nonnegative real number and positive real number respectively. With an abuse of notation, we sometimes use $B(\boldsymbol{x}) = 0$ for the semialgebraic set it defines. ∂S denotes the boundary of compact set S.

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

Definition 1 (Lie derivative). *For a given polynomial $p \in \mathbb{R}[\boldsymbol{x}]$ over $\boldsymbol{x} = (x_1, \ldots, x_n)$ and a continuous system $\dot{\boldsymbol{x}} = \boldsymbol{f}$, where $\boldsymbol{f} = (f_1, \ldots, f_n)$, the **Lie derivative** of $p \in \mathbb{R}[\boldsymbol{x}]$ along $\boldsymbol{f}$ is defined as $\mathcal{L}_{\boldsymbol{f}} p = \sum_{i=1}^{n} \frac{\partial p}{\partial x_i} \cdot f_i$.*

Essentially, the Lie derivative of p is the time derivative of p, i.e., reflects the change of p over time.

In this paper, we focus on semialgebraic systems with uncertainty, which is described by the following ODE.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1}$$

where $\boldsymbol{f}$ is a vector of polynomial functions, $\boldsymbol{x}(t)$ is a solution of the system, $\boldsymbol{u}(t)$ is the vector of uncertain parameters and/or perturbation and $\boldsymbol{u}(t)$ is Lipschitz continuous. Note that we do not make a distinction between uncertain parameters and perturbation since we deal with them uniformly. Formally, semialgebraic system with uncertainty is defined as follows.

Definition 2 (Semialgebraic system with uncertainty). *A **semialgebraic system with uncertainty** is a 5-tuple $\mathcal{M} \stackrel{def}{=} \langle X, \boldsymbol{f}, X_0, \mathcal{I}, \mathcal{U} \rangle$, where*

1. *$X \subseteq \mathbb{R}^n$ is the state space of the system $\mathcal{M}$,*
2. *$\boldsymbol{f} \in \mathbb{R}[\boldsymbol{x}, \boldsymbol{u}]^n$ is locally Lipschitz continuous vector function defining the vector flow as in ODE (1),*
3. *$X_0 \subseteq X$ is the initial set, which is semialgebraic [43],*
4. *$\mathcal{I}$ is the invariant or domain of the system,*
5. *$\mathcal{U}$ is a domain for the uncertain parameters and perturbation, i.e., $\boldsymbol{u}(t) \in \mathcal{U}$*

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

Definition 3 (Trajectory). *Given a semialgebraic system with uncertainty $\mathcal{M}$, a **trajectory** originating from a point $\boldsymbol{x}_0 \in X_0$ to time $T > 0$ is a continuous and differentiable function $\boldsymbol{\zeta}(\boldsymbol{x}_0, t) : [0, T) \to \mathbb{R}^n$ such that (1) $\boldsymbol{\zeta}(x_0, 0) = \boldsymbol{x}_0$, and (2) $\exists \boldsymbol{u}(\cdot) : \forall \tau \in [0, T)$: $\frac{d\boldsymbol{\zeta}}{dt}\big|_{t=\tau} = \boldsymbol{f}(\boldsymbol{\zeta}(\boldsymbol{x}_0, \tau), \boldsymbol{u}(\tau))$, where $\boldsymbol{u}(\cdot) : [0, T) \to \mathcal{U}$. T is assumed to be within the maximal interval of existence of the solution from $\boldsymbol{x}_0$.*

For ease of readability, we also use $\zeta(t)$ for $\zeta(\boldsymbol{x}_0, t)$ if it is clear from the context.

3 Robust Barrier Certificate by Linear Programming

A barrier certificate for a continuous dynamics system is a real-valued function $B(\boldsymbol{x})$ such that (1) the initial set and the unsafe set are located on different sides of the hyper-surface $\mathcal{H} = \{\boldsymbol{x} \in \mathbb{R}^n \mid B(\boldsymbol{x}) = 0\}$ respectively, and (2) no trajectory originating from the same side of $\mathcal{H}$ as the initial set can cross through $\mathcal{H}$ to reach the other side. Therefore, the existence of such a function $B(\boldsymbol{x})$ can guarantee the safety of the system. The above condition can be formalized using an infinite sequence of higher order Lie derivatives [44]. Unfortunately, this formalization cannot be applied directly to barrier certificate computation. Therefore, a couple of sufficient conditions for the above condition have been proposed [17,26,30]. Most recently, based on the sufficient condition in [34], a new approach was proposed to overapproximate the flowpipe of nonlinear continuous dynamical systems using combination of barrier certificates [24].

However, the approach is limited to continuous dynamical systems without uncertainty. To tackle this problem, we extend the approach to deal with continuous and hybrid systems with uncertainty. Similarly, we adopt the same barrier certificate condition as [24], but we introduce the uncertainty in the barrier certificate condition. Note that in order to distinguish it from barrier certificate for dynamical system without uncertainty, we call a barrier certificate satisfying the following condition *robust barrier certificate*.

Theorem 1. *Given an uncertain semialgebraic system* $\mathcal{M} = \langle X, \boldsymbol{f}, X_0, \mathcal{I}, \mathcal{U} \rangle$, *let* X_{us} *be the unsafe set, the system is guaranteed to be safe if there exists a real-valued function* $B(\boldsymbol{x})$ *such that*

$$\forall \boldsymbol{x} \in X_0 : B(\boldsymbol{x}) > 0 \tag{2}$$

$$\forall (\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{I} \times \mathcal{U} : \mathcal{L}_{\boldsymbol{f}} B > 0 \tag{3}$$

$$\forall \boldsymbol{x} \in X_{us} : B(\boldsymbol{x}) < 0 \tag{4}$$

The most common approach to barrier certificate computation is by SOS programming [26,34]. The idea of this kind of approach is to first relax the original constraints like (2)–(4) into a set of positive semidefinite (PSD) polynomials by applying Putinar representation [35], which is further relaxed by requiring every PSD polynomial has a sum-of-squares decomposition, which can be solved by SOS programming in polynomial time. However, constructing automatically a set of consistent templates for the barrier certificate as well as the auxiliary polynomials is not trivial. In addition, SOS programming method can yield fake solution sometimes due to numerical error.

An alternative to SOS programming based approaches is to use linear programming based approaches. This class of approaches relies on an LP-relaxation to the original constraint. In [3,40], to compute Lyapunov function, an LP-relaxation was obtained by applying Handelman representation to the original constraint. Recently, this kind of LP-relaxation was adopted in [24] to compute piecewise barrier tubes. In [45], an extended version of Handelman representation, called Krivine representation [29], was employed for barrier certificate computation. Compared to Handelman representation, which can only deal with convex polytopes, Krivine representation can deal with more general compact semialgebraic sets. However, Krivine representation requires normalizing the polynomials involved, which is expensive.

In this paper, we adopt the same representation as in [24], i.e., Handelman representation as our LP-relaxation scheme for Theorem 1. We assume that the initial set X_0, the unsafe set X_{us}, the invariant $\mathcal{I}$, the parameter and/or perturbance space are all convex and compact polyhedra, i.e., $X_0 = \{\boldsymbol{x} \in \mathbb{R}^n \mid p_1(\boldsymbol{x}) \geq 0, \cdots, p_{m_1}(\boldsymbol{x}) \geq 0\}$, $\mathcal{I} = \{\boldsymbol{x} \in \mathbb{R}^n \mid q_1(\boldsymbol{x}) \geq 0, \cdots, q_{m_2}(\boldsymbol{x}) \geq 0\}$, $\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^l \mid w_1(\boldsymbol{u}) \geq 0, \cdots, w_{m_3}(\boldsymbol{u}) \geq 0\}$ and $X_{us} = \{\boldsymbol{x} \in \mathbb{R}^n \mid r_1(\boldsymbol{x}) \geq 0, \cdots, r_{m_4}(\boldsymbol{x}) \geq 0\}$ where $p_i(\boldsymbol{x})$, $q_i(\boldsymbol{x})$, $r_k(\boldsymbol{x})$ and $w_i(\boldsymbol{u})$, are all linear polynomials. Then, Theorem 1 can be relaxed as follows.

Theorem 2. *Given a semialgebraic system with uncertainty* $\mathcal{M} = \langle X, \boldsymbol{f}, X_0, \mathcal{I}, \mathcal{U} \rangle$*, let* X_0*,* X_{us}*,* $\mathcal{I}$ *and* $\mathcal{U}$ *be defined as above, the system is guaranteed to be safe if there exists a real-valued polynomial function* $B(\boldsymbol{x})$ *such that*

$$B(\boldsymbol{x}) \equiv \sum_{|\boldsymbol{\alpha}| \leq M_1} \lambda_{\boldsymbol{\alpha}} \prod_{i=1}^{m_1} p_i^{\alpha_i} + \epsilon_1 \tag{5}$$

$$\mathcal{L}_{\boldsymbol{f}} B \equiv \sum_{|\boldsymbol{\beta}| \leq M_2} \lambda_{\boldsymbol{\beta}} \prod_{i=1}^{m_2} q_i^{\beta_i} \prod_{j=1}^{m_3} w_j^{\beta_{m_2+j}} + \epsilon_2 \tag{6}$$

$$-B(\boldsymbol{x}) \equiv \sum_{|\boldsymbol{\gamma}| \leq M_3} \lambda_{\boldsymbol{\gamma}} \prod_{i=1}^{m_4} r_i^{\gamma_i} + \epsilon_3 \tag{7}$$

where $\boldsymbol{\alpha} = (\alpha_k), \boldsymbol{\beta} = (\beta_k), \boldsymbol{\gamma} = (\gamma_k)$*,* $\lambda_{\boldsymbol{\alpha}}, \lambda_{\boldsymbol{\beta}}, \lambda_{\boldsymbol{\gamma}} \in \mathbb{R}_{\geq 0}$*,* $\epsilon_i \in \mathbb{R}_{>0}$ *and* $M_i \in \mathbb{N}, i = 1, \cdots, 3$*.*

Remark 1. Theorem 2 implies that the system $\mathcal{M}$ can be proved to be safe as long as we can find a real-valued polynomial function $B(\boldsymbol{x})$ such that $B(x)$, $-B(\boldsymbol{x})$ and $\mathcal{L}_{\boldsymbol{f}} B$ can be written as a nonnegative combination of the products of the powers of the polynomials defining X_0, X_{us} and $\mathcal{I} \times \mathcal{U}$ respectively. This theorem provides us with a solution to solve barrier certificate by linear programming. Given a polynomial template $B(\boldsymbol{c}, \boldsymbol{x})$ for $B(\boldsymbol{x})$, where $\boldsymbol{c}$ is the coefficients of the monomials to be decided in $B(\boldsymbol{c}, \boldsymbol{x})$, we substitute $B(\boldsymbol{c}, \boldsymbol{x})$ for $B(\boldsymbol{x})$ occurring in the conditions (5)–(7) to obtain three polynomial identities in $\mathbb{R}[\boldsymbol{x}]$ with linear polynomials in $\mathbb{R}[\boldsymbol{c}, \boldsymbol{\lambda}]$ as their coefficients, where $\boldsymbol{\lambda}$ is a vector composed of all the $\lambda_{\boldsymbol{\alpha}}, \lambda_{\boldsymbol{\beta}}, \lambda_{\boldsymbol{\gamma}}$ occurring in (5)–(7). Since (5)–(7) are identities, then all the coefficients of the corresponding monomials on both sides of the identities must be identical. By collecting the corresponding coefficients of the monomials on both sides of the identities and let them equal respectively, we obtain a system S of linear equations and inequalities on $\boldsymbol{c}, \boldsymbol{\lambda}$. Now, finding a robust barrier certificate is converted to finding a feasible solution for S, which can be solved by linear programming efficiently. Since the degree of $B(\boldsymbol{c}, \boldsymbol{x})$ is key to the expressive power of $B(\boldsymbol{c}, \boldsymbol{x})$, in our implementation, we attempt to solve a barrier certificate from a group of templates with different degrees.

Due to the page limit, we do not elaborate on our algorithm for barrier certificate computation, but we demonstrate how it works in the following example.

Example 1. Given a 2D system defined by $\dot{x} = 2x + 3y + u_1, \dot{y} = -4x + 2y + u_2$, let $X_0 = \{(x, y) \in \mathbb{R}^2 \mid p_1 = x + 100 \geq 0, p_2 = -90 - x \geq 0, p_3 = y + 45 \geq 0, p_4 = -40 - y \geq 0\}$, $\mathcal{I} = \{(x, y) \in \mathbb{R}^2 \mid q_1 = x + 110 \geq 0, q_2 = -80 - x \geq 0, q_3 = y + 45 \geq 0, q_4 = -20 - y \geq 0\}$, $\mathcal{U} = \{(u_1, u_2) \in \mathbb{R}^2 \mid w_1 = u_1 + 50.0 \geq 0, w_2 = 50.0 - u_1 \geq 0, w_3 = u_2 + 50.0 \geq 0, w_4 = 50.0 - u_2 \geq 0\}$ and $X_{us} = \{(x, y) \in \mathbb{R}^2 \mid r_1 = x + 98 \geq 0, r_2 = -90 - x \geq 0, r_3 = y + 24 \geq 0, r_4 = -20 - y \geq 0\}$. Assume $B(\boldsymbol{c}, \boldsymbol{x}) = c_0 + c_1 x + c_2 y$, $M_i = \epsilon_i = 1$ for $i = 1, \cdots, 3$, then we obtain the following polynomial identities according to Theorem 2

$$c_1 + c_2x + c_3y - \sum_{i=1}^{4} \lambda_{1i}p_i - \epsilon_1 \equiv 0$$

$$c_2(2x + 3y + u_1) + c_3(-4x + 2y + u_2) - \sum_{j=1}^{4} \lambda_{2j}q_j - \sum_{j=1}^{4} \lambda_{3j}w_j - \epsilon_2 \equiv 0$$

$$-\ (c_1 + c_2x + c_3y) - \sum_{k=1}^{4} \lambda_{4k}r_k - \epsilon_3 \equiv 0$$

where $\lambda_{ij} \geq 0$ for $i, j = 1, \cdots, 4$. If we collect the coefficients of x, y, u_1, u_2 in the above polynomials and let them be 0, we obtain a system S of linear polynomial equations and inequalities over c_i, λ_{ij}. By solving S using linear programming, we obtain a feasible solution with $c_1 = -1263.5, c_2 = -11.5, c_3 = -5.85$.

4 Piecewise Robust Barrier Tubes

The idea of piecewise robust barrier tubes (PRBTs) is to use robust barrier tubes (RBTs) to piecewise overapproximate the flowpipe segments of nonlinear hybrid systems with uncertainty, where each RBT is essentially a cluster of robust barrier certificates which are situated around the flowpipe to form a tight tube enclosing the flowpipe. The basic idea of PRBT computation is shown in Algorithm 1.

4.1 Construction of the Enclosure-Box

A key step in PRBT computation is the construction of enclosure-box for a given compact initial set. Note that here *an enclosure-box is a hyperrectangle that entirely contains a flowpipe segment*. In principle, the smaller the enclosure-box, the easier it is to compute a barrier tube. However, to make full use of the power of nonlinear overapproximation, it is desirable to have as big enclosure-box as possible so that fewer barrier tubes are needed to cover a flowpipe.

In [24], interval method was adopted to build an enclosure-box. However, the main problem with interval method is that the enclosure-box thus computed is usually very small which will result in a big number of barrier tubes for a fixed length of flowpipe. On the one hand, this will lead to an increasing burden on barrier tube computation. On the other hand, the capability of barrier tube in overapproximating complex flowpipe can not be fully released. For these reasons, we choose to use a purely simulation-based approach without losing soundness.

A key concept involved in our simulation-based enclosure-box construction is *twisting of trajectory*, which is a measure of maximal bending of trajectories in a box. For the convenience of presentation, we present the formal definition of twisting of trajectory as follows.

Algorithm 1. PRBT computation

input : $\boldsymbol{f}$: dynamics of the system; X_0: Initial set; $\mathcal{U}$: set of uncertainty; N: number of robust barrier tubes (RBT) in PRBT; (θ_{min}, d_{min}): parameters for simulation

output: PRBT: piecewise robust barrier tube

```
1  PRBT ← empty queue;
2  while Length(PRBT) < N do
3      [Found, θ, d] ← [false, θ_0, d_0] ;
4      while θ > θ_min do
5          E ← construct a coarse enclsoure-box for X_0 by (θ, d)-simulation;
6          [Found, RBT, X_0'] ←compute RBT inside E and obtain a set
           X_0' ⊇ (RBT ∩ ∂E) ;
7          if not Found then
8              (θ, d) ← 1/2 * (θ, d); // to shrink E
9              continue;
10         else
11             PRBT ← Push(PRBT, RBT); // add RBT to the queue of PRBT
12             X_0 ← X_0' ; // update X_0 for computing next RBT
13             break;
14     if not Found then break ;
15 return PRBT;
```

Definition 4 (Twisting of trajectory). *Let $\mathcal{M}$ be a continuous system and $\zeta(t)$ be a trajectory of $\mathcal{M}$. Then, $\zeta(t)$ is said to have a twisting of θ on the time interval $I = [T_1, T_2]$, written as $\xi_I(\zeta)$, if it satisfies that $\xi_I(\zeta) = \theta$, where* $\xi_I(\zeta) \stackrel{def}{=} \sup_{t_1,t_2 \in I} \arccos\left(\frac{\langle \dot{\zeta}(t_1), \dot{\zeta}(t_2)\rangle}{\|\dot{\zeta}(t_1)\|\|\dot{\zeta}(t_2)\|}\right)$.

Then, we have Algorithm 2 to compute enclosure-box.

Remark 2. In this paper, we assume that both X_0 and $\mathcal{U}$ are defined by hyper-rectangles. The basic idea of enclosure-box construction is that, given a continuous dynamical system with uncertainty, we first remove the uncertainty by taking the center point $\boldsymbol{u}_c$ of $\mathcal{U}$ for the dynamics (line 1–2). Then, we sample a set S_0 of points from X_0 for simulation (line 3). Prior to doing simulation for S_0, we first select a point $\boldsymbol{x}_0$ (usually the center point of X_0) to do (θ, d)-simulation to obtain the end point $\boldsymbol{x}_e$ of the simulation (line 7). *A (θ, d)-simulation is a simulation that stops either when the twisting of the simulation reaches θ or when the Euclidean distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}_e$ reaches d.* The motivation to get the end point $\boldsymbol{x}_e$ is that, there are n planes of the form $x_i = \boldsymbol{x}_e^i$ (the i'th element of $\boldsymbol{x}_e$) intersecting at $\boldsymbol{x}_e$, so we want to check if one of the n planes, say P, was hit by all the simulations that start from S_0, and if yes, it is very likely that P cut through the entire flowpipe. Then, we take P as one of the facets of the desired enclosure-box E. In addition, during the simulations, we simultaneously

Algorithm 2. Construct enclosure-box

input : $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$: system dynamics; X_0: initial set; $\mathcal{U}$: uncertain parameters; θ: twisting of simulation; θ_{min}: minimal theta for simulation; d: maximum distance of simulation;

output: E: an enclosure-box containing X_0; P: plane where flowpipe exits ; G: range of intersection of $Flow_f(X_0)$ with plane P by simulation

```
1  u_c ← center point of U;
2  f_c(x) ← the center dynamic f(x, u_c);
3  S_0 ← sample a set of points from X_0;
4  select a point x_0 ∈ S_0;
5  succ ← false;
6  while θ ≥ θ_min do
7      x_e ← end point of (θ, d)-simulation of f_c(x) for x_0;
8      foreach x_e^i: plane in the i'th dimension of x_e do
9          do simulation for all the points in S_0, update G and E;
10         if all the simulations hit x_e^i then
11             P ← x_e^i;
12             succ ← true;
13     if succ then
14         bloat E s.t Flow_f(X_0) exits from E only through the facet in P;
15         return [E, P, G];
16     else
17         [θ, d] ← 1/2 * [θ, d];
```

keep updating (1) the boundary where the simulations can reach and use that range as our candidate enclosure-box E, and (2) the boundary range G where the simulations intersect with the plane $P : x_i = \boldsymbol{x}_e^i$. If we end up finding such a plane P, we will push the other facets of E outwards to make the flowpipe exit only from this specific facet of E. Of course, this objective cannot be guaranteed only by simulation and pushing, we need to further check if the flowpipe does not intersect the other facets of E, which can be done according to Theorem 3.

Theorem 3. *Given an uncertain semialgebraic system* $\mathcal{M} = \langle X, \boldsymbol{f}, X_0, \mathcal{I}, \mathcal{U} \rangle$, *assume* $E \subset \mathcal{I}$ *is an enclosure-box of* X_0 *and* F_i *is a facet of* E. *The flowpipe of* $\mathcal{M}$ *from* X_0 *does not intersect* F_i, *i.e,* $(Flow_f(X_0) \cap F_i) \cap E = \emptyset$ *if there exists a barrier certificate* $B_i(\boldsymbol{x})$ *for* F_i *inside* E.

Remark 3. Theorem 3 can be easily proved by the definition of barrier certificate, which is ignored here. In order to make sure that the flowpipe evades a facet F_i of E, according to Theorem 1, we only need to find a barrier certificate for F_i. In the case of no barrier certificate being found, further bloating to the facet of E will be performed. If bloating facet still end up with failure, we keep shrinking E by setting (θ, d) to $(\theta/2, d/2)$ until barrier certificates are found for all the facet of E or θ gets less than some threshold θ_{min}.

4.2 Computation of Robust Barrier Tube

An ideal application scenario of barrier certificate is when we can prove the safety property using a single barrier certificate. Unfortunately, this is usually not true because the flowpipe can be very complicated so that no polynomial function of a specified degree satisfies the constraint. In the previous subsection, we introduce how to obtain for an initial set X_0 an enclosure-box E in which the system dynamics is simple enough so that a robust barrier certificate $B(\boldsymbol{x})$ can be easily computed. Therefore, we can compute a set of robust barrier certificates, which we call Robust Barrier Tube (RBT), to create a tight overapproximation for the flowpipe provided that there is a set of auxiliary sets serving as unsafe sets. Formally, we define RBT as follows.

Definition 5 (Robust Barrier Tube (RBT)). *Given a semialgebraic system* $\mathcal{M} = \langle X, \boldsymbol{f}, X_0, \mathcal{I}, \mathcal{U} \rangle$*, let* E *be an enclosure-box of* X_0 *and* $X_{AS} = \{X_{AS}^i : X_{AS}^i \subseteq E\}$ *be a set of auxiliary sets (AS), an RBT is a set of real-valued functions* $\Phi = \{B_i(\boldsymbol{x}), i = 1, \cdots, m\}$ *such that for all* $B_i(\boldsymbol{x}) \in \Phi$*: (i)* $\forall \boldsymbol{x} \in X_0 : B_i(\boldsymbol{x}) > 0$*, (ii)* $\forall (\boldsymbol{x}, \boldsymbol{u}) \in E \times \mathcal{U} : \mathcal{L}_{\boldsymbol{f}} B_i > 0$*, and (iii)* $\forall \boldsymbol{x} \in X_{AS}^i : B_i(\boldsymbol{x}) < 0$*.*

The precision of RBT depends closely on the set X_{AS} of ASs. Therefore, to derive a good barrier tube, we need to first construct a set of high quality ASs. The factors that could affect the quality of the set X_{AS} of ASs include (1) the number of ASs, and (2) the position, size and shape of AS. Roughly speaking, the more ASs we have, if positioned properly, the more precise the RBT would be. Regarding the position, size and shape of AS, a desirable AS should (1) be as close to the flowpipe as possible, (2) spread widely around the flowpipe, and (3) be shaped like a shell for the flowpipe. Intuitively, a high quality set of ASs could be shaped like a ring around a human finger so that the barrier tube is tightly confined in the narrow space between the ring and the finger. With the key factors aforementioned in mind, we developed Algorithm 3 for RBT computation.

Remark 4. In principle, the more barrier certificates we use, the better overapproximation we may achieve. However, using more barrier certificates also means more computation time. Therefore, we have to make a trade-off between precision and efficiency. In Algorithm 3, we choose to use RBT consisting of $2(n-1)$ barrier certificates for n dimensional dynamical systems, which means we need to construct $2(n-1)$ ASs. We use the same scheme as in [24] to construct ASs. Recall that we get a coarse region `G` where the flowpipe intersects with one of the facets of `E` during the construction of the enclosure-box `E`. Since `G` is an $n-1$ dimensional box, the RBT must contain `G`. Therefore, we choose to construct $2(n-1)$ ASs which are able to form a tight hollow hyper-rectangle around `G`. The idea is that for each facet $\mathtt{G}_{ij}$ of `G`, we construct an $n-1$ dimensional hyper-rectangle between $\mathtt{G}_{ij}$ and E_{ij} as an `AS` (line 2), where E_{ij} is the $n-1$ dimensional face of `E` that corresponds to `G`. Then, we use Algorithm 3 to compute an RBT(line 4). In the **while** loop 3, we try to find the best barrier certificate by adjusting the width of `AS` (line 5 and 6) iteratively until the difference in width between two consecutive `AS`s is less than the specified threshold ϵ. To be intuitive, we provide Fig. 1 to demonstrate the process.

Algorithm 3. Compute robust barrier tube

input : $\boldsymbol{f}$: system dynamics; X_0: Initial set; E: enclosure-box of X_0; $\mathcal{U}$: set of uncertainty; P: plane where flowpipe exits from enclosure-box E; G: box approx. of $(\mathsf{P} \cap Flow_{\boldsymbol{f}}(X_0))$ by simulation; ϵ: difference between AS's (auxiliary set)

output: RBT: barrier tube; X_0': box over-approx. of $(\mathsf{RBT} \cap E)$

foreach G_{ij}: *a facet of* G **do**
  AS $\longleftarrow$ `CreateAS`(G, P, G_{ij});
  while *true* **do**
    [*found*, B_{ij}] $\longleftarrow$ `ComputeRBC`($\boldsymbol{f}$, X_0, E, AS, $\mathcal{U}$);
    if *found* **then** AS' $\longleftarrow$ `Expand` (AS) ;
    else AS' $\longleftarrow$ `Contract` (AS) ;
    if `Diff`(AS', AS) $\leq \epsilon$ **then**
      break;
    AS $\longleftarrow$ AS';
  if *found* **then**
    RBT $\longleftarrow$ `Push`(RBT, B_{ij});
    break;
  else
    return FAIL
return SUCCEED;

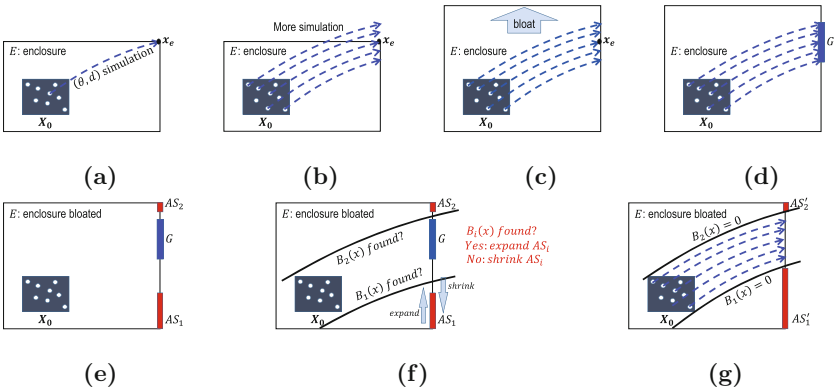


Fig. 1. (a)→(g): demonstration of RBT computation

4.3 PRBT for Continuous Dynamics

The idea of computing PRBT is straightforward. Given an initial set X_0, we first construct a coarse enclosure-box E containing X_0 and then we further compute an RBT inside E to get a much more precise overapproximation for the flowpipe. Meanwhile, we obtain a hyper-rectangle R formed by ASs with a hollow $X_0{}'$ in the middle. Since the intersection of the RBT and the facet of E is contained

entirely in the hollow ${X_0}'$ of R, we use ${X_0}'$ as a new initial set and repeat the entire process to compute a PRBT step by step. Since our approach is time independent, the length of a PRBT cannot be measured by the length of time horizon. Hence, in our implementation, we try to compute a specified number of RBTs.

4.4 PRBT for Hybrid Dynamics

To extend our approach with the ability to deal with hybrid systems, we need to handle two problems (i) compute the intersection of RBT and guard set, and (ii) compute the image of the intersection after discrete jump. In general, these two issues can be very hard depending on what kind of guard sets and transitions are defined for the hybrid systems.

In this paper, we make some assumptions on the hybrid systems under consideration. Let a discrete transition τ be defined as follows.

$$\tau_{ll'} = \langle \texttt{Guard}_{ll'}, \texttt{Trans}_{ll'} \rangle \tag{8}$$

where l and l' are the locations of the dynamics before and after a discrete transition respectively, $\texttt{Guard}_{ll'} = \{\boldsymbol{x} \in \mathbb{R}^n \mid x_i \sim b_i, \sim \in \{\leq, \geq\}\}$ and $\texttt{Trans}_{ll'} : \boldsymbol{x'} = A\boldsymbol{x}$, where A is an n-dimensional matrix. Based on this assumption, the problem of computing the intersection of RBT and guard set is reduced to computing the intersection of RBT with a plane of $x_i = b_i$, which can be handled using a similar strategy to computing the intersection of RBT with the facet of enclosure-box. Hence, we have Algorithm 4 to deal with discrete transition of a hybrid system.

Remark 5. The strategy to deal with discrete transitions of hybrid systems is that every time we obtain an enclosure-box $\texttt{E}$, we first detect whether $\texttt{E}$ intersects with some guard set $^{\texttt{Guard}_{11'}}$. If no, we proceed with the normal process of PRBT computation. Otherwise, we switch to the procedure of Algorithm 4 in which the input X_0^l is the last state set whose enclosure-box intersects with $^{\texttt{Guard}_{11'}}$. Since the flowpipe may not cross through the guard plane entirely, we use the while loop in line 2 to compute an overapproximation for the intersection. The basic idea of the while loop is that, given a state set X_0^l, we first construct an enclosure-box $\texttt{E}$ by simulation (line 4), if $\texttt{E}$ intersects with $^{\texttt{Guard}_{11'}}$, we shrink $\texttt{E}$ by cutting off the part of $\texttt{E}$ that lies in the guard set (line 7). As a result of this operation, the flowpipe could exit from $\texttt{E}$ not only through the guard plane but also through other facets of $\texttt{E}$. For each of those facets, we compute an overapproximation for its intersection with the flowpipe using simulation and barrier certificate computation (line 8 and 11). In addition, since those intersections X_0^{ij} that do not lie in the guard plane could still reach the guard plane later, we therefore push them into a queue for further exploration.

Algorithm 4. handle discrete transition of hybrid system

input : X_0^l: intermediate initial set at location l; $\mathsf{Guard}_{ll'}$: guard set of transition $\tau_{ll'}$; $\mathsf{Trans}_{ll'}$: image mapping of transition $\tau_{ll'}$
output: $X_0^{l'}$: image of transition $\tau_{ll'}$

$\mathsf{InitQ} \leftarrow \mathtt{Push}(\mathsf{InitQ}, X_0^l)$;
while InitQ *not empty* **do**
  $X_0^l \leftarrow \mathtt{Pop}(\mathsf{InitQ})$;
  $\mathsf{E} \leftarrow$ construct enclosure-box for X_0^l ;
  if $\mathsf{E} \cap \mathsf{Guard}_{ll'} == \emptyset$ **then**
    continue;
  $\mathsf{E} \leftarrow \mathsf{E} \cap \overline{\mathsf{Guard}}_{ll'}$;
  $X_{\Phi \cap \mathsf{E}} \leftarrow$ do simulation and barrier certificate computation to find an overapproximation for the region where the flowpipe Φ intersects with the guard plane $x_i = b_i$;
  $Q_{\Phi \cap \mathsf{E}} \leftarrow \mathtt{Push}(Q_{\Phi \cap \mathsf{E}}, X_{\Phi \cap \mathsf{E}})$;
  foreach E_{ij}*: facet of* E *except guard plane* **do**
    $X_0^{ij} \leftarrow$ do simulation and barrier certificate computation to an overapproximation for the region where the barrier tube intersects with E_{ij}; $\mathsf{InitQ} \leftarrow \mathtt{Push}(\mathsf{InitQ}, X_0^{ij})$;
$X_{\Phi \cap \mathsf{E}} \leftarrow$ box overapprox. $Q_{\Phi \cap E}$;
$X_0^{l'} \leftarrow \mathsf{Trans}_{ll'} X_{\Phi \cap \mathsf{E}}$;

5 Implementation and Experiments

We have developed PRBT, a software prototype written in C++ that implements the concepts and the algorithms presented in this paper. PRBT computes piecewise robust barrier tubes for nonlinear continuous and hybrid systems with polynomial dynamics. We compare our approach in efficiency and precision with the state-of-the-art tools Flow* and CORA using several benchmarks of nonlinear continuous and hybrid systems. Note that since C2E2 does not support uncertainty, so we cannot compare with it. The experiments were carried out on a desktop computer with a 3.6 GHz *Intel 8 Core i7-7700* CPU and 32 GB memory.

5.1 Nonlinear Continuous Systems

We consider six nonlinear benchmark systems with polynomial dynamics for which their models and settings are provided in Table 1.

The experimental results are reported in Table 2. Since our approach is time independent, which is different from Flow* and CORA, to make the comparison fair enough, we choose to compute a slightly longer flowpipe than the other two tools. Note that there are two columns for time for Flow*. The reason why we have an extra time column for Flow* is that it can be very fast and precise to compute the Taylor model for a given system. However, Taylor models cannot be in general applied directly to solve the safety verification problem. Checking

Table 1. Continuous dynamical model definitions

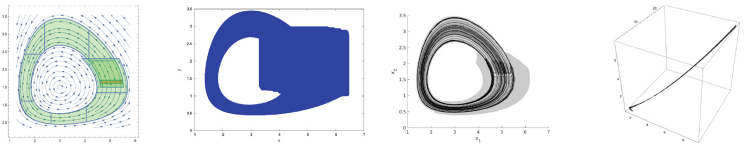
Model	Dynamics	Uncertainty	X_0
Controller 2D	$\dot{x} = d_1 xy + y^3 + 2$ $\dot{y} = d_2 x^2 + 2x - 3y$	$d_1 \in [0.95, 1.05]$ $d_2 \in [0.95, 1.05]$	$x \in [29.9, 30.1]$ $y \in [-38, -36]$
Van der Pol Oscillator	$\dot{x} = y + d_1$ $\dot{y} = y - x - x^2 y + d_2$	$d_1 \in [-0.01, 0.01]$ $d_2 \in [-0.01, 0.01]$	$x \in [1, 1.5]$ $y \in [2.40, 2.45]$
Lotka-Volterra	$\dot{x} = x(1.5 - y) + d_1$ $\dot{y} = -y(3 - x) - d_2$	$d_1 \in [-0.01, 0.01]$ $d_2 \in [-0.01, 0.01]$	$x \in [4.6, 5.5]$ $y \in [1.6, 1.7]$
Buckling Column	$\dot{x} = y + d_1$ $\dot{y} = 2x - x^3 - 0.2y + 0.1 + d_2$	$d_1 \in [-0.01, 0.01]$ $d_1 \in [-0.01, 0.01]$	$x \in [-0.5, -0.4]$ $y \in [-0.5, -0.4]$
Jet Engine	$\dot{x} = -y - 1.5x^2 - 0.5x^3 - 0.5 + d_1$ $\dot{y} = 3x - y + d_2$	$d_1 \in [-0.005, 0.005]$ $d_2 \in [-0.005, 0.005]$	$x \in [1.19, 1.21]$ $y \in [0.8, 1.0]$
Controller 3D	$\dot{x} = 10(y - x) + d_1$ $\dot{y} = x^3 + d_2$ $\dot{z} = xy - 2.667z$	$d_1 \in [-0.001, 0.001]$ $d_2 \in [-0.001, 0.001]$	$x \in [1.79, 1.81]$ $y \in [1.0, 1.1]$ $z \in [0.5, 0.6]$

their intersection with the unsafe set requires their transformation into simpler geometric form (e.g. box), which has an exponential complexity in the number of the dimensions and it needs to be considered in the overall running time.

Remark 6. Table 2 shows us how brutal the reality of reachability analysis of nonlinear systems is and this gets even worse in the presence of uncertainty and large initial set. As can be seen in Table 2, both Flow* and CORA failed to give a solution for half of the benchmarks either due to timeout or due to exception. This phenomenon may be alleviated if smaller initial sets are provided or uncertainty is removed. In terms of computing time T, PRBT does not always outperform the other two tools. Actually, Flow* or CORA can be much faster in

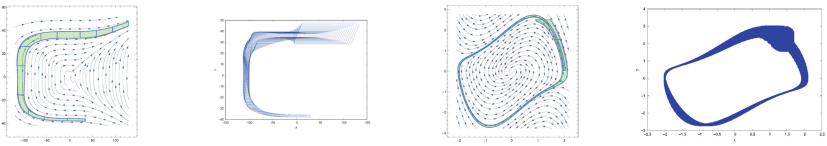
Table 2. Experimental results for benchmark systems. #var: number of variables; T: computing time for flowpipe; TT: computing time including box transformation; N: number of flowpipe segments; D: candidate degrees for template polynomial (for PRBT only); TH: time horizon for flowpipe (for Flow* and CORA only). F/E: failed to terminate under 30 min or exception happened.

Model	#var	PRBT			TH	Flow*			CORA	
		T	N	D		T	TT	N	T	N
Controller 2D	2	35.73	12	3	0.012	48.8	417.64	240	F	1200
Van der Pol	2	221.62	17	4	6.74	23.88	1111.05	135	E	–
Lotka-Volterra	2	30.10	9	4	3.20	6.06	405.32	320	40.30	160
Buckling Column	2	74.02	35	3	14.00	F	–	–	734.81	1400
Jet Engine	2	240.98	18	4	9.50	F	–	–	1.69	190
Controller 3D	3	774.58	20	3	0.55	F	–	–	F	–



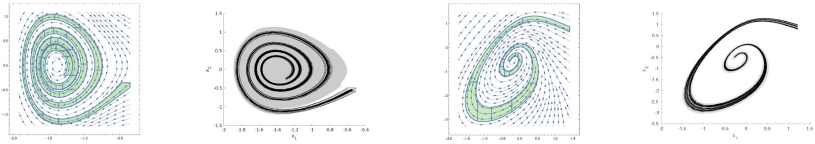
(a) PRBT (b) Flow* (c) CORA (d) PRBT

Fig. 2. Lotka-Volterra: (a), (b), (c); controller 3D: (d)



(a) PRBT (b) Flow* (c) PRBT (d) Flow*

Fig. 3. Controller 2d: (a), (b); Van der Pol Oscillator: (c), (d)



(a) PRBT (b) CORA (c) PRBT (d) CORA

Fig. 4. Buckling Column: (a), (b); Jet Engine: (c), (d)

some cases. However, when the box transformation time for Taylor model was taken into account, the total computing time TT of Flow* increased significantly. One point to note here is that, PRBT, in general, produces a much smaller number N of flowpipe segments than the other two, which means that the time used to check the intersection of flowpipe with the unsafe set can be reduced considerably. In addition, as shown in Figs. 2, 3 and 4, PRBT is more precise than the other two on average.

5.2 Nonlinear Hybrid System

We use the tunnel diode oscillator (TDO) circuit (with different setting) introduced in [20] to illustrate the application of our approach to hybrid system. The two state space variables are the voltage $x_1 = V_C$ across capacitor and the current $x_2 = I_L$ through the inductor. The system dynamics is described as follows,

$$\dot{x}_1 = \frac{1}{C}(-h(x_1) + x_2) \qquad \dot{x}_2 = \frac{1}{L}(-x_1 - \frac{x_2}{G} + V_{in})$$

where $h(x_1)$ describes the tunnel diode characteristic and $V_{in} = 0.3\,\mathrm{V}$, $G = 5\,\mathrm{m}\,\Omega^{-1}$, $L = 0.5\,\mu\mathrm{H}$ and $C = 2\,\mathrm{pF}$.

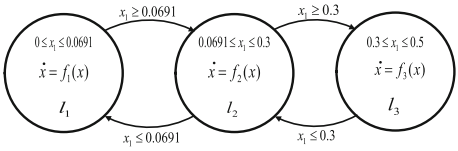
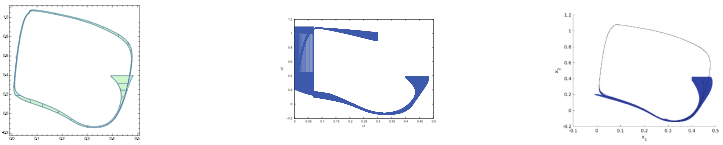


Fig. 5. Hybridized model of TDO



(a) PRBT **(b) Flow*** **(c) CORA**

Fig. 6. Flowpipe of hybridized TDO

For this model, we want to define an initial state region X_0 which can guarantee the oscillating behaviour for the system. Due to the highly nonlinear behaviour of the system, a common strategy to deal with this model is to use a hybridized model to approximate the dynamics system and then apply formal verification to the hybrid model [13,18]. In our experiment, we use three cubic equations to approximate the curve of $h(x_1)$.

$$h(x_1) = \begin{cases} 0.000847012 + 35.2297x_1 - 395.261x_1^2 + 1372.29x_1^3, \ 0 \leq x_1 \leq 0.0691 \\ 1.242 - 0.033x_1 - 47.4311x_1^2 + 116.48x_1^3, \ 0.0691 \leq x_1 \leq 0.3 \\ -16.544 + 139.64x_1 - 389.245x_1^2 + 359.948x_1^3, \ 0.3 \leq x_1 \leq 0.50 \end{cases}$$

From the piecewise function $h(x_1)$, we can derive a 3-mode hybrid system which is shown in Fig. 5. The system switches between the locations as the value of x_1 changes.

Let the initial set be $X_0 = \{(x_1, x_2) \in \mathbb{R}^2 \mid 0.40 \leq x_1 \leq 0.48, 0.38 \leq x_2 \leq 0.39\}$ on location l_3, we compute an overapproximation for the flowpipe using PRBT, Flow* and CORA respectively. As illustrated in Fig. 6, both PRBT and CORA found an invariant with roughly the same precision, which indicates the model oscillates for the initial set, while Flow* ran into an error.

References

1. Althoff, M., Grebenyuk, D.: Implementation of interval arithmetic in CORA 2016. In: Proceedings of ARCH. EPiC Series in Computing, vol. 43, pp. 91–105. EasyChair (2017)
2. Asarin, E., Dang, T., Girard, A.: Hybridization methods for the analysis of nonlinear systems. Acta Inform. **43**(7), 451–476 (2007)
3. Ben Sassi, M.A., Sankaranarayanan, S., Chen, X., Ábrahám, E.: Linear relaxations of polynomial positivity for polynomial lyapunovfunction synthesis. IMA J. Math. Control. Inf. **33**(3), 723–756 (2015)

4. Berz, M., Makino, K.: Verified integration of odes and flows using differential algebraic methods on high-order taylor models. Reliab. Comput. **4**(4), 361–369 (1998)
5. Bogomolov, S., Schilling, C., Bartocci, E., Batt, G., Kong, H., Grosu, R.: Abstraction-based parameter synthesis for multiaffine systems. In: Piterman, N. (ed.) HVC 2015. LNCS, vol. 9434, pp. 19–35. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26287-1_2
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, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18
7. Cimatti, A., Griggio, A., Irfan, A., Roveri, M., Sebastiani, R.: Experimenting on solving nonlinear integer arithmetic with incremental linearization. In: Beyersdorff, O., Wintersteiger, C.M. (eds.) SAT 2018. LNCS, vol. 10929, pp. 383–398. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94144-8_23
8. Cimatti, A., Griggio, A., Irfan, A., Roveri, M., Sebastiani, R.: Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions. ACM Trans. Comput. Log. **19**(3), 19:1–19:52 (2018)
9. Cyranka, J., Islam, M.A., Byrne, G., Jones, P., Smolka, S.A., Grosu, R.: Lagrangian reachabililty. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 379–400. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_19
10. Cyranka, J., Islam, Md.A., Smolka, S.A., Gao, S., Grosu, R.: Tight continuous-time reachtubes for lagrangian reachability. In: Proceedings of CDC 2018: 57th IEEE Conference on Decision and Control. IEEE (2018, to appear)
11. Duggirala, P.S., Mitra, S., Viswanathan, M., Potok, M.: C2E2: a verification tool for stateflow models. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 68–82. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_5
12. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. JSAT **1**(3–4), 209–236 (2007)
13. Frehse, G., Krogh, B.H., Rutenbar, R.A.: Verification of hybrid systems using iterative refinement. In: Proceedings of SRC TECHCON 2005, Portland, USA, 24–26 October 2005
14. Frehse, G., et al.: SpaceEx: scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_30
15. Girard, A., Le Guernic, C.: Efficient reachability analysis for linear systems using support functions. Proc. IFAC World Congr. **41**(2), 8966–8971 (2008)
16. Grosu, R., et al.: From cardiac cells to genetic regulatory networks. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 396–411. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_31
17. 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, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70545-1_18
18. Gupta, S., Krogh, B.H., Rutenbar, R.A.: Towards formal verification of analog and mixed-signal designs. In: TECHCON (2003)
19. Gurung, A., Ray, R., Bartocci, E., Bogomolov, S., Grosu, R.: Parallel reachability analysis of hybrid systems in xspeed. Int. J. Softw. Tools Technol. Transf., 1–23 (2018, to appear)

20. Hartong, W., Hedrich, L., Barke, E.: Model checking algorithms for analog verification. In: Proceedings of the 39th annual Design Automation Conference, pp. 542–547. ACM (2002)
21. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings of IEEE Symposium on Logic in Computer Science, pp. 278–292 (1996)
22. Jiang, Y., Song, H., Wang, R., Gu, M., Sun, J., Sha, L.: Data-centered runtime verification of wireless medical cyber-physical system. IEEE Trans. Ind. Inform. **13**(4), 1900–1909 (2017)
23. Jiang, Y., Wang, M., Liu, H., Hosseini, M., Sun, J.: Dependable integrated clinical system architecture with runtime verification. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 951–956, November 2017
24. Kong, H., Bartocci, E., Henzinger, T.A.: Reachable set over-approximation for nonlinear systems using piecewise barrier tubes. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 449–467. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_24
25. Kong, H., Bogomolov, S., Schilling, C., Jiang, Y., Henzinger, T.A.: Safety verification of nonlinear hybrid systems based on invariant clusters. In: Proceedings of HSCC 2017: the 20th International Conference on Hybrid Systems: Computation and Control, pp. 163–172. ACM (2017)
26. Kong, H., He, F., Song, X., Hung, W.N.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, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_17
27. Kong, S., Gao, S., Chen, W., Clarke, E.: dReach: δ-reachability analysis for hybrid systems. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 200–205. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_15
28. Krilavicius, T.: Hybrid techniques for hybrid systems. Ph.D. thesis, University of Twente, Enschede, Netherlands (2006)
29. Lasserre, J.B.: Polynomial programming: LP-relaxations also converge. SIAM J. Optim. **15**(2), 383–393 (2005)
30. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: Proceedings of EMSOFT 2011: the 11th International Conference on Embedded Software, pp. 97–106. ACM (2011)
31. Matringe, N., Moura, A.V., Rebiha, R.: Generating invariants for non-linear hybrid systems by linear algebraic methods. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 373–389. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15769-1_23
32. Nedialkov, N.S.: Interval tools for ODEs and DAEs. In: Proceedings of SCAN 2006: the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, pp. 4–4. IEEE (2006)
33. Prabhakar, P., García Soto, M.: Hybridization for stability analysis of switched linear systems. In: Proceedings of HSCC 2016: of the 19th International Conference on Hybrid Systems: Computation and Control, pp. 71–80. ACM (2016)
34. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 477–492. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24743-2_32
35. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana Univ. Math. J. **42**(3), 969–984 (1993)

36. Ray, R., Gurung, A., Das, B., Bartocci, E., Bogomolov, S., Grosu, R.: XSpeed: accelerating reachability analysis on multi-core processors. In: Piterman, N. (ed.) HVC 2015. LNCS, vol. 9434, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26287-1_1
37. Roohi, N., Prabhakar, P., Viswanathan, M.: Hybridization based CEGAR for hybrid automata with affine dynamics. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 752–769. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_48
38. Sankaranarayanan, S.: Automatic invariant generation for hybrid systems using ideal fixed points. In: Proceedings of HSCC 2010: the 13th ACM International Conference on Hybrid Systems: Computation and Control, pp. 221–230. ACM (2010)
39. 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, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24743-2_36
40. Sankaranarayanan, S., Chen, X., et al.: Lyapunov function synthesis using handelman representations. IFAC Proc. Vol. **46**(23), 576–581 (2013)
41. Schupp, S., Ábrahám, E., Makhlouf, I.B., Kowalewski, S.: HyPro: A C++ library of state set representations for hybrid systems reachability analysis. In: Barrett, C., Davies, M., Kahsai, T. (eds.) NFM 2017. LNCS, vol. 10227, pp. 288–294. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57288-8_20
42. Sogokon, A., Ghorbal, K., Jackson, P.B., Platzer, A.: A method for invariant generation for polynomial continuous systems. In: Jobstmann, B., Leino, K.R.M. (eds.) VMCAI 2016. LNCS, vol. 9583, pp. 268–288. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49122-5_13
43. Stengle, G.: A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. Mathematische Annalen **207**(2), 87–97 (1974)
44. Taly, A., Tiwari, A.: Deductive verification of continuous dynamical systems. In: FSTTCS, vol. 4, pp. 383–394 (2009)
45. Yang, Z., Huang, C., Chen, X., Lin, W., Liu, Z.: A linear programming relaxation based approach for generating barrier certificates of hybrid systems. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 721–738. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48989-6_44