Parsimonious, Simulation Based Verification of Linear Systems

Parasara Sridhar Duggirala[1(✉)] and Mahesh Viswanathan[2]

[1] University of Connecticut, Mansfield, USA
psd@uconn.edu
[2] University of Illinois, Urbana-Champaign, Champaign, USA
vmahesh@illinois.edu

Abstract. We present a technique to verify safety properties of linear systems (possibly time varying) using very few simulations. For a linear system of dimension n, our technique needs $n+1$ simulation runs. This is in contrast to current simulation based approaches, where the number of simulations either depends upon the number of vertices in the convex polyhedral initial set, or on the proximity of the unsafe set to the set of reachable states. At its core, our algorithm exploits the superposition principle of linear systems. Our algorithm computes both an over and an under approximation of the set of reachable states.

1 Introduction

Cyberphysical systems, that involve the close interaction of a computing device with a physical process, are most faithfully modeled as a hybrid system that exhibits both discrete and continuous changes to system state. The mathematical model of a hybrid system consists of a finite collection of *control modes* where the system state evolves continuously with time. Transitions between control modes are governed by constraints on the system state.

A commonly occurring special class of hybrid systems is one where the continuous dynamics in each control mode is mathematically described using a *time-varying linear differential equation* of the form

$$\dot{x} = A(t)x + B(t), \tag{1}$$

where $A(t)$ and $B(t)$ are matrices which may themselves be changing with time. While verifying invariant properties for such systems is known to be undecidable in general, the set of states reachable within bounded time (and bounded number of discrete steps) can be approximated with arbitrary precision. One of the core challenges in computing such bounded-time reachable sets is to compute the set of all states reachable within a time bound for a *single control mode* with no mode switches (often referred as continuous post).

There are two main approaches to computing the continuous post for a mode within time bound T. The first approach [7,12,18] exploits the linearity of the system dynamics. For continuous dynamics given by Eq. (1), let us denote by

S. Chaudhuri and A. Farzan (Eds.): CAV 2016, Part I, LNCS 9779, pp. 477–494, 2016.
DOI: 10.1007/978-3-319-41528-4_26

$\xi(x,t)$ the state at time t starting from x. It is well known that the state reached at time t when starting from $\alpha x_1 + (1-\alpha)x_2$ $(0 \leq \alpha \leq 1)$, a convex combination of states x_1 and x_2, is given by $\alpha\xi(x_1,t) + (1-\alpha)\xi(x_2,t)$. Hence, if the initial set of states is a *convex, bounded polytope*, then the set of states reached at time t is the convex hull of the states reached from each vertex of the initial polytope. Further, the set of states reached *within* time t is over-approximated by bloating the convex hull of the vertices of the initial polytope and the vertices of the polytope of states reached at time t. The bloating factor, determined by a careful error analysis, depends on the length of time t. Thus to get a good approximation of the reach set within a time bound T, the interval $[0,T]$ is broken up into small steps adaptively [13,22]. The cost of computing the reach set in this approach, therefore, depends on two things (1) the number of vertices in the initial polytope (which is exponential in the dimension of the system), and (2) the number of smaller intervals the time interval $[0,T]$ is divided into. The efficiency of this approach also depends on the data structure used to store the set of reachable states. Ellipsoids [17], convex polyhedra [12], zonotopes [14], support functions [18], polynomial zonotopes [2], and Taylor models [5], are some of the popular data structures used. Each of these data structures requires developing new algorithms for computing the reachable set for a given class of systems.

The second approach is a *simulation-based* approach [9,10,15]. Here, the initial set is partitioned into smaller neighborhoods, and the system is simulated from the center of each neighborhood. Based on the norms of matrices A and B, one can compute an envelope around each simulation trace that guarantees the containment of the trajectory starting from any point in a given initial partition. The reachable set is therefore over-approximated by a collection of simulation tubes. The quality of this set can be improved by computing a finer partitioning of the initial set. Thus, for a safe system, the number of simulations needed, depends on how far the unsafe set is from the reachable set; if it is far, a coarse initial partition suffices, and if it is close then we need a fine initial partition, which means many simulations. Though this approach may require significantly more simulations, it enjoys a couple of advantages over the previous approach. First, since this approach does not rely on convexity properties of linear systems, it can be used to analyze non-convex initial sets and time varying linear systems (where $A(t)$ and $B(t)$ change with time). Second, not only can it be used to prove safety, but also to find counterexamples.

Apart from these two approaches, a few theorem proving approaches have also been proposed [16,20,21,23,24,26]. In these approaches one does not compute the set of reachable states, but rather prove that a certain safety property is satisfied. Therefore, this technique can be used for proving safety of non-convex and unbounded initial sets, but also requires additional manual effort.

Inspired by the simulation-based approach, we present a new approach for computing the reachable set for linear systems. Our approach combines the advantages of each of the above approaches. First, like the simulation-based approach, it can be used to analyze non-convex initial states, time-varying

linear systems, and it can prove unsafety of systems in addition to safety. Second, and more importantly, it uses significantly fewer simulations — to compute the reachable set of an n-dimensional system, we need to simulate the system from only $n+1$ initial states. This is in contrast to the potentially exponentially many vertices to be propagated in the non-simulation approach, and potentially much larger than exponentially-many simulations in the simulation-based approach. Third, our approach does not require any additional computation if the initial set changes, as long as the "center" of the set remains the same; what this means precisely will be become clearer later in this introduction as we describe our approach. Fourth, the previous two approaches only work for *bounded initial sets*. Our new approach, on the other hand, can handle unbounded initial sets. Finally, since our method only relies on simulations, it does not require a formal model, and can be used to analyze *black-box* systems.

The main idea behind our approach is to exploit what is sometimes called the *superposition principle*. Let us consider an n-dimensional system (i.e., continuous state is in $\mathbb{R}^n$) described by Eq. (1). For vectors $v_1, v_2, \ldots v_n$, initial "center" x_0, and constants $\alpha_1, \alpha_2, \ldots \alpha_n$, the superposition principle says that

$$\xi(x_0 + \sum_{i=1}^{n} \alpha_i v_i, t) = \xi(x_0, t) + \sum_{i=1}^{n} \alpha_i (\xi(x_0 + v_i, t) - \xi(x_0, t)) \tag{2}$$

Thus, if the initial set is of the form $x_0 + \sum_{i=1}^{n} \alpha_i v_i$ where the coefficients $\bar{\alpha}$ belong to some set Δ, then the set of states reached at time t is given by $\xi(x_0, t) + \sum_{i=1}^{n} \alpha_i v_i'$ with $\bar{\alpha} \in \Delta$, where $v_i' = \xi(x_0 + v_i, t) - \xi(x_0, t)$.

Notice, that this representation of the states at time t, only requires us to find $\xi(x_0, t), \xi(x_0 + v_1, t), \ldots \xi(x_0 + v_n, t)$, which can be obtained by only $n + 1$ simulations. We call this representation of sets of states as the linear span of a center x_0 and basis vectors $\{v_i\}_{i=1}^{n}$ with coefficients $\bar{\alpha} \in \Delta$ *generalized star sets*. Such generalized star sets naturally generalize standard shapes like polytopes, ellipsoids, and non-convex sets. Using generalized star sets makes reachable set computation simple. Moreover, if the initial set changes because of a change in Δ, the superposition principle tells us that we don't need to do any additional simulations in order to represent the reachable set at time t. We show how this basic idea can be adapted to account for simulation errors, to construct both under and over approximations of the reachable set of states, efficiently.

Our experimental results substantiate our belief that this new approach can serve as the founding principle that underlies the next advance in the scalable analysis of time varying linear systems. Our method scales to high dimensional systems and beats all current verification technologies by at least an order of magnitude. This is not surprising given the obvious theoretical advantages it enjoys over past methods due to the reduced number of simulations it needs.

2 Preliminaries

We refer to states and vectors as elements in $\mathbb{R}^n$. We denote the ℓ^∞ norm of the vectors and states by $||\cdot||$. To avoid confusion we denote states by x_i and vectors

by v_i. Given two states x_1 and x_2, the difference vector is defined as $v = x_2 - x_1$. Given a set $S \subseteq \mathbb{R}^n$, $\mathsf{diameter}(S) \triangleq \sup\{||x-y|| \mid x, y \in S\}$. For a set $S \subseteq \mathbb{R}^n$, a point $x \in S$ is said to be a center if $\forall y \in S.\ ||x-y|| \leq \mathsf{diameter}(S)/2$. A set S may or may not have a center; convex sets do have a center. When a set S has a center there maybe many; we will abuse notation and use $\mathsf{center}(S)$ to denote one picked by the axiom of choice. A predicate $P : \mathbb{R}^n \to \{\top, \bot\}$ denotes a set of vectors denoted by $[\![P]\!] = \{v | P(v) = \top\}$. We abuse notation and denote both the predicate P and the set $[\![P]\!]$ as P. The ball of radius δ around a state x is defined as $B_\delta(x) = \{y \mid ||x-y|| \leq \delta\}$; similarly, for a set $S \subseteq \mathbb{R}^n$, $B_\delta(S) = \cup_{x \in S} B_\delta(x)$. Given two vectors $p, q \in \mathbb{R}^k$ where $p = [p_1, p_2, \ldots, p_k]^T$ and $q = [q_1, q_2, \ldots, q_k]^T$, we say that $p \leq q$ if and only if $\forall i.\, p_i \leq q_i$. Given $x \in \mathbb{R}^n$ and $S \subseteq \mathbb{R}^n$, the set of difference vectors from x to S, is defined as $\mathsf{diff}(S, x) \triangleq \{\, v \mid \exists x' \in S, v = x' - x\}$.

We will find it convenient to represent subsets of states using a representation that we call *generalized star sets*, which we define next.

Definition 1. *A* generalized star set *is a tuple $\Theta = \langle x_0, V, P\rangle$ where $x_0 \in \mathbb{R}^n$ is called the* center*, $V = \{v_1, v_2, \ldots v_m\}$ is a set of m ($\leq n$) vectors in $\mathbb{R}^n$ called the* basis*, and $P : \mathbb{R}^n \to \{\top, \bot\}$ is a predicate.*

A generalized star set Θ defines a subset of $\mathbb{R}^n$ as follows.

$$[\![\Theta]\!] = \{x \mid \exists \bar{\alpha} = [\alpha_1, \ldots, \alpha_m]^T \text{ such that } x = x_0 + \Sigma_{i=1}^n \alpha_i v_i \text{ and } P(\bar{\alpha}) = \top\}$$

Sometimes we will refer to both Θ and $[\![\Theta]\!]$ as Θ.

In the above definition of generalized star sets, the size of the vector set V will often be determined by the dimension of the set $[\![\Theta]\!]$ being defined, and the vectors will be linearly independent. However, we do not require this. Generalized star sets are a generalization of many natural sets of states. Depending on the predicate P, generalized star representation can define a variety of sets including non-convex sets and convex sets like polyhedra and ellipsoids. We provide some examples of such sets.

Example 1. Consider the 2-dimensional plane $\mathbb{R}^2$. Take $V = \{[1,0]^T, [0,1]^T\}$ the set of unit vectors along the two axes, and $x_0 = (3,3)$.

Consider $g = [1,1,1,1]^T$, and $P(\bar{\alpha}) = C\bar{\alpha} \leq g$ where $C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T$

The generalized star set $\Theta = \langle x_0, V, P\rangle$ defines the rectangular set

$$[\![\Theta]\!] = B_1(3,3) = \{(x,y) \mid 2 \leq x \leq 4 \wedge 2 \leq y \leq 4\}$$

On the other hand, defining $P(\bar{\alpha}) = (\alpha_1 - 3)^2 + (\alpha_2 - 3)^2 \leq 1$ defines the disc of radius 1 with center $(3,3)$.

Consider a system described by the linear ODE

$$\dot{x} = A(t)x + B(t). \tag{3}$$

The solution of the above ODE with initial state x_0 is denoted as $\xi(x_0, t)$. For this solution $\frac{d}{dt}(\xi(x_0,t)) = A(t)\xi(x_0,t) + B(t)$ and $\xi(x_0, 0) = x_0$. For well defined linear time varying systems, the state at time t is given using the state transformation matrix $\Phi : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^{n \times n}$ such that the trajectory at time t is given as

$$\xi(x_0, t) = \Phi(t, 0)x_0 + \int_0^t \Phi(t, s)B(s)ds. \tag{4}$$

Notice that for linear time invariant systems, the expression for $\Phi(t_2, t_1) = e^{A(t_2 - t_1)}$.

For performing simulation based verification, instead of using a numerical simulation which returns a sequence of states, we use *validated simulations* which returns a sequence of sets of states with the following guarantees.

Definition 2. *For a system described by Eq. (3), with closed form $\xi(x_0, t)$ given by Eq. (4), an (x_0, T, ϵ, h)-validated simulation of $\xi(x_0, t)$ is $\psi = (R_1, [t_0, t_1])$, $(R_2, [t_1, t_2])$, $\ldots$, $(R_k, [t_{m-1}, t_m])$ where $R_i \subseteq \mathbb{R}^n$ such that*

1. $\forall 1 \leq i \leq m, t_i - t_{i-1} \leq h$, $t_0 = 0$, $t_m = T$.
2. $\forall 1 \leq i \leq m, \forall t \in [t_{i-1}, t_i], \xi(x_0, t) \in R_i$.
3. $\forall 1 \leq i \leq m, \mathsf{diameter}(R_i) \leq \epsilon$.

The first condition enforces that the time step for each of these regions is bounded by h. The second condition enforces that for each interval $[t_{i-1}, t_i]$ the trajectory is contained within the region R_i. The third condition enforces that the diameter of each region is bounded by ϵ. Existing numerical solvers such as CAPD, and VNODE-LP can compute validated simulations which contain the trajectory. For these tools, the sets R_i are convex, polyhedral sets. Therefore, we assume that the subroutine $\mathsf{valSim}(x_0, T, h)$ returns $\langle \psi, \epsilon \rangle$ such that ψ is an (x_0, T, ϵ, h)-validated simulation (with R_i being convex). In addition, as $h \rightarrow 0$, $\epsilon \rightarrow 0$.

Definition 3. *For a system in Eq. (3), and initial set Θ, the set of states reachable within time bound T is $ReachSet_{\langle A,B \rangle}(\Theta, T) = \{\xi(x_0, t) | x_0 \in \Theta, 0 \leq t \leq T\}$. We drop A and B from the ReachSet when it is clear from the context.*

A set R_O is said to be an over-approximation *of the reachable states within time T if $ReachSet(\Theta, T) \subseteq R_O$. Analogously, R_U is said to be an* under-approximation *of the set of reachable states within time bound T, if $R_U \subseteq ReachSet(\Theta, T)$.*

Definition 4. *The system given in Eq. (3) is said to be safe for bounded time T from the initial state Θ and unsafe set U if $ReachSet(\Theta, T) \cap U = \emptyset$.*

3 Computing Reachable Sets from Simulations

In this section we outline how to compute reachable sets of n-dimensional linear systems, using at most $n + 1$ simulations. We begin (Sect. 3.1) by making an observation that is often called the *superposition principle*. This principle enables

us to express the set of states reached at time t as a generalized star set, if the initial states is given as a generalized star set. In Sect. 3.2, we show how the superposition principle can be used to compute the set of reachable states, under the assumption that the exact trajectory from each initial state can be computed. Finally, in Sect. 3.3, we show how all of these ideas can used when we only have access to validated simulation engines.

3.1 Superposition Principle for Linear Systems

In order to explain the superposition principle, let us fix a system described by Eq. (3). Recall from Eq. (4), the solution is for the system is given as

$$\xi(x_0, t) = \Phi(t, 0)x_0 + \int_0^t \Phi(t, s)B(s)ds.$$

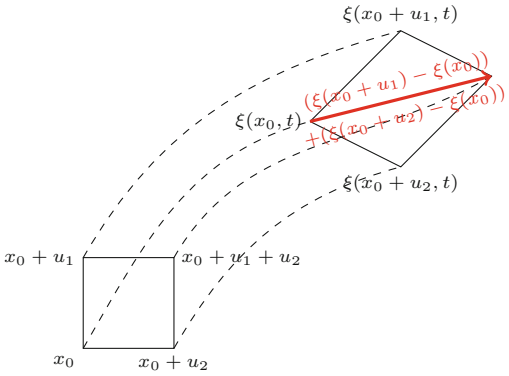

Fig. 1. Observe that the state reached at time t from $x_0 + u_1 + u_2$ is identical to $\xi(x_0, t) + (\xi(x_0 + u_1, t) - \xi(x_0, t)) + (\xi(x_0 + u_2, t) - \xi(x_0, t))$.

Consider two initial states x_0 and $x_0 + u_1$, for some vector u_1. From the solution given in Eq. (4), we have

$$\xi(x_0 + u_1, t) - \xi(x_0, t) = \Phi(t, 0)u_1 \tag{5}$$

For two vectors u_1 and u_2, and state x_0, from Eq. (4), we have

$$\begin{aligned}
&\xi(x_0 + \alpha_1 u_1 + \alpha_2 u_2) \\
&= \Phi(t, 0)(x_0 + \alpha_1 u_1 + \alpha_2 u_2) + \int_0^t \Phi(t, s)B(s)ds \\
&= [\Phi(t, 0)x_0 + \int_0^t \Phi(t, s)B(s)ds] + \alpha_1\Phi(t, 0)u_1 + \alpha_2\Phi(t, 0)u_2 \\
&= \xi(x_0, t) + \alpha_1\Phi(t, 0)u_1 + \alpha_2\Phi(t, 0)u_2 \\
&= \xi(x_0, t) + \alpha_1[\xi(x_0 + u_1, t) - \xi(x_0, t)] + \alpha_2[\xi(x_0 + u_2, t) - \xi(x_0, t)]
\end{aligned}$$

The above equation suggests that linear combinations of $\xi(x_0 + u_1, t) - \xi(x_0, t)$ and $\xi(x_0 + u_2, t) - \xi(x_0, t)$ gives us the difference between trajectories starting from initial state x_0 and $x_0 + \alpha_1 u_1 + \alpha_2 u_2$. This is illustrated in Fig. 1. Extending this observation to n vectors we have

$$\xi(x_0 + \Sigma_{i=1}^{n} \alpha_i u_i, t) = \xi(x_0, t) + \Sigma_{i=1}^{n} \alpha_i (\xi(x_0 + u_i, t) - \xi(x_0, t)). \quad (6)$$

3.2 Reach Sets from Exact Trajectories

In this section, we will outline how the superposition principle can be used construct the reachable states at a given time t. Let us fix an initial set given as a generalized star set $\Theta = \langle x_0, V, P \rangle$, where $V = \{v_1, v_2, \ldots v_m\}$. We begin by showing how to compute $Reach_t(\Theta)$, the set of states reached at time t; $Reach_t(\Theta)$ is defined precisely as follows.

$$Reach_t(\Theta) = \{\xi(x, t) \mid x \in [\![\Theta]\!]\}.$$

The reachable states at time t is computed by Algorithm 1 as a generalized star set.

input : Initial Set: $\Theta = \langle x_0, V, P \rangle$, Time instance: t
output: $Reach_t(\Theta)$
1 $x_0' \leftarrow \xi(x_0, t)$;
2 **for** *each* $v_i \in V$ **do**
3 $\quad x_i' \leftarrow \xi(x_0 + v_i, t)$;
4 $\quad v_i' \leftarrow x_i' - x_0'$;
5 **end**
6 $V' \leftarrow \{v_1', \ldots, v_m'\}$;
7 $Reach_t(\Theta) \leftarrow \langle x_0', V', P \rangle$;
8 **return** $Reach_t(\Theta)$;

Algorithm 1. Algorithm that computes the reachable set at time t from $n+1$ simulations.

The algorithm in line 1 computes the state of trajectory starting from the initial state x_0 at time t as x_0'. The loop in lines 2 to 4 computes x_i', the state of the trajectory starting from $x_0 + v_i$ at time t. The reachable set at time t is given as as generalized star set $\langle x_0', V', P \rangle$, where $V' = \{v_1', \ldots, v_n'\}$ with $v_i' = x_i' - x_0'$. Theorem 1 proves that the set returned is indeed the reachable set.

Theorem 1. *The set $Reach_t(\Theta)$ is the reachable set for Θ at time t.*

Proof. Let us consider the set of vectors $V' = \{v_1', \ldots, v_m'\}$. Observe from Eq. (5) that $v_i' = \Phi(t, 0) v_i$.

A state y is reachable at time t, if y is the state reached at time t when starting from some initial state $x' \in [\![\Theta]\!]$. More formally, a state $y \in Reach_t(\Theta)$

if and only if $\exists \bar{\alpha} = [\alpha_1, \ldots, \alpha_m]^T$ such that $P(\bar{\alpha}) = \top$ and $y = \xi(x', t)$ where $x' = x_0 + \Sigma_{i=1}^{n} \alpha_i v_i$. From lines 1, 3, and 4, we have that

$$y = \xi(x_0, t) + \Sigma_{i=1}^{n} \alpha_i (\xi(x_0 + v_i, t) - \xi(x_0, t)).$$

Thus, $y \in \langle x'_0, V', P \rangle$ establishing the correctness of the algorithm.

We conclude this proof by observing that since $\Phi(t, 0)$ is an invertible matrix, V' is linearly independent set of vectors, if V is linearly independent.

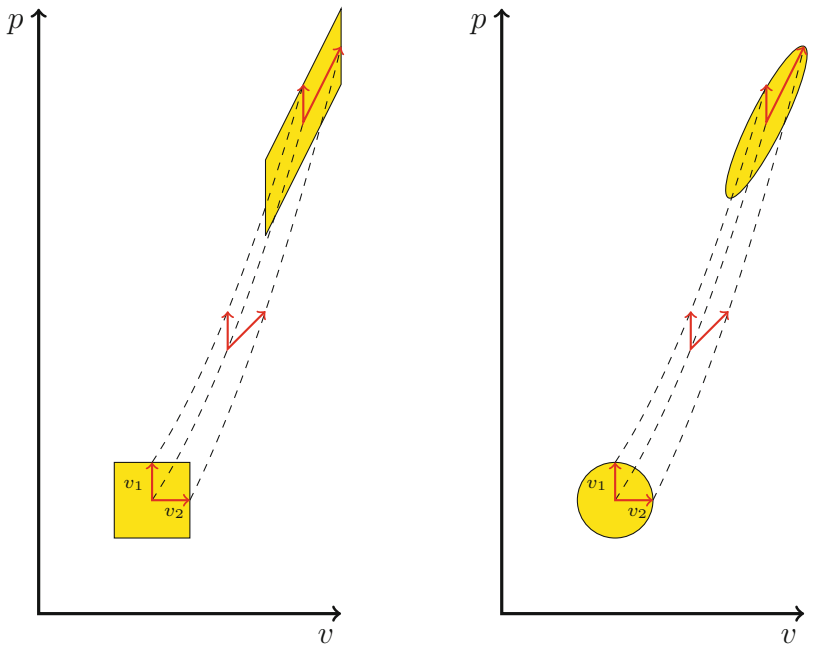

Fig. 2. Reachable set of car moving in 1-dimension with constant acceleration. In both graphs, car velocity v is plotted on the x-axis and position p is on the y-axis. The set of initial states and the set of reachable states at time 2 are shown in yellow. The vectors defining the sets is shown in red at time $0, 1$, and 2. On the left, the initial set is the ball of radius 1 with center $(3, 3)$ with respect to ℓ^∞-norm. On the right the initial set is the same except that the ball is defined with respect to the ℓ^2-norm. Notice that the evolution of the vectors that define the generalized star set is the same in both the left and the right. (Color figure online)

Example 2. Consider the simple example of a car moving in 1-dimension with constant acceleration of 2 units. Taking the state of the system to be the car position (p) and velocity (v), the dynamics can be described as

$$\dot{p} = v \qquad \dot{v} = 2$$

Consider the polyhedral initial set given as a generalized star set. In other words, $\Theta = \langle x_0, V, P\rangle$, where $x_0 = (3,3)$, $V = \{[1,0]^T, [0,1]^T\}$, $g = [1,1,1,1]^T$ and $P(\bar{\alpha}) = C\bar{\alpha} \leq g$ where $C = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}^T$ The evolution of the reachable set is shown in the left graph in Fig. 2. The reachable set at time 2 is given by the generalized star set $\Delta = \langle x_0', V', P\rangle$, where $x_0 = (7,13)$, and $V' = \{[0,1]^T, [1,2]^T\}$. The only part that changes in the generalized star representation of the reachable set at time 2 is the center and the set of vectors.

Suppose we consider the initial set to be the disc of radius 1 with center $(3,3)$ as in Example 1. That is the initial set if given as $\langle (3,3), \{[1,0]^T, [0,1]^T\}, P\rangle$ where $P(\bar{\alpha}) = (\alpha_1 - 3)^2 + (\alpha_2 - 3)^2 \leq 1$. The evolution of the reachable set over time is shown on the right graph in Fig. 2. The reachable set at time 2 is described as $\langle (7,13), \{[0,1]^T, [1,2]^T\}, P\rangle$. Thus the way the center and the set of vectors change is the same for both the box and the disc initial sets. The evolution of the center and vectors is independent of the "shape" of the initial set. The difference in the reachable sets only arises because of the difference in the predicate used to describe the initial sets.

3.3 Computing Reachable Set from Validated Simulations

Algorithm 1 computes the reachable set at time t when the *exact* state of the $n+1$ trajectories starting from x_0, $x_0 + v_1$, ..., $x_0 + v_n$ at time t is known. However, computing the exact state requires computing the closed form expression for Φ. This expression Φ in the simplest case where $A(t)$ and $B(t)$ are time invariant matrices requires computing matrix exponentials and so the exact expression can only be computed for very special matrices. We now present a new technique in Algorithm 2 (based on Algorithm 1) for computing a formula with existential quantifiers that represents the overapproximation and the underapproximation of the reachable set of states.

Informally, instead of computing the exact trajectories starting from x_0, $x_0 + v_1$, ..., $x_0 + v_n$, we compute their validated simulations. We assume that all these validated simulations are *synchronized*, i.e., the number of intervals in all the validated simulations are the same. Although according to Definition 2, each of these validated simulations can have different time intervals and regions, we can split the required intervals further and generate new validated simulations such that all the $n + 1$ validated simulations have the same number of regions. We assume that there are m such regions in each validated simulation, i.e., the simulation from x_0, denoted as $\langle \psi^0, \epsilon\rangle \leftarrow \mathsf{valSim}(x_0, h, T)$ is such that $\psi^0 = (R_1^0, [t_0, t_1])$, $(R_2^0, [t_1, t_2])$, ..., $(R_m^0, [t_{m-1}, t_m])$. The validated simulation $\langle \psi^i, \epsilon\rangle \leftarrow \mathsf{valSim}(x_0 + v_i, h, T)$ is such that $\psi^i = (R_1^i, [t_0, t_1]), \ldots, (R_m^i, [t_{m-1}, t_m])$.

input : Initial Set: $\Theta = \langle x_0, V, P \rangle$, Time bound: T
output: Overapproximation and underapproximation of the reachable set of states.

1 $\langle \psi^0, \epsilon^0 \rangle \leftarrow \mathsf{valSim}(x_0, h, T)$;
2 **for** *each* $v_i \in V$ **do**
3 $\quad \langle \psi^i, \epsilon^i \rangle \leftarrow \mathsf{valSim}(x_0 + v_i, h, T)$;
4 **end**
5 **for** $j = 1 \dots m$ **do**
6 $\quad OverReach[j] \leftarrow OA(R_j^0, R_j^1, \dots, R_j^n, P)$;
7 $\quad UnderReach[j] \leftarrow UA(R_j^0, R_j^1, \dots, R_j^n, P)$;
8 **end**
9 **return** $(OverReach, UnderReach)$;

Algorithm 2. Algorithm that computes the overapproximation and underapproximation of reachable set for each time interval.

Given $R_0, R_1, \dots, R_n \subseteq \mathbb{R}^n$, $OA(R_0, R_1, \dots, R_n, P)$ is a formula with quantifiers that represents an overapproximation of the reachable set is defined as:

$$OA \triangleq \{\, x \mid \exists x_0 \in R_0, \exists v_i \in \mathsf{diff}(R_i, x_0), \exists \bar{\alpha}, \\ x = x_0 + \Sigma_{i=1}^n \alpha_i v_i \wedge P(\bar{\alpha}) = \top \,\} \tag{7}$$

Similarly $UA(R_0, R_1, \dots, R_n, P)$ is a formula with quantifiers that represents an underapproximation of the reachable set is defined as:

$$UA \triangleq \{\, x \mid \forall x_0 \in R_0, \forall v_i \in \mathsf{diff}(R_i, x_0), \exists \bar{\alpha}, \\ x = x_0 + \Sigma_{i=1}^n \alpha_i v_i, P(\bar{\alpha}) = \top \,\} \tag{8}$$

Theorem 2. *$OverReach[j]$ and $UnderReach[j]$ computed in line 6 and 7 give an overapproximation and underapproximation of the reachable set of states for the time interval $[t_{j-1}, t_j]$ respectively.*

Proof. The proof follows from the proof of Theorem 1.

Overapproximation: Consider the $Reach_t(\Theta)$ for some $t \in [t_{j-1}, t_j]$. A state $x \in Reach_t(\Theta)$ if and only if $\exists \bar{\alpha}$ such that $x = x_0' + \Sigma_{i=1}^n \alpha_i v_i', P(\bar{\alpha}) = \top$. From Definition 2, it follows that $x_0' \in R_j^0$ and $v_i' \in \mathsf{diff}(R_j^i, x_0')$. As the formula is existentially quantified, if follows that $x \in OverReach[j]$. Therefore, $\cup_{t \in [t_{j-1}, t_j]} Reach_t(\Theta) \subseteq OverReach[j]$.

Underapproximation: Consider a state $x \in UnderReach[j]$. Therefore, $\forall x_0' \in R_j^0, \forall v_i' \in \mathsf{diff}(R_j^i, x_0'), \exists \bar{\alpha}$, such that $x = x_0' + \Sigma_{i=1}^n \alpha_i v_i', P(\bar{\alpha}) = \top$. Now consider $Reach_t(\Theta)$ for some time instance $t \in [t_{j-1}, t_j]$. As x_0' and v_i' is universally quantified, it follows that $x \in Reach_t(\Theta)$. Therefore $UnderReach[j] \subseteq \cap_{t \in [t_{j-1}, t_j]} Reach_t(\Theta)$.

Remark 1. Algorithm 2 can be used for safety verification. Given an unsafe set of states U, one can check whether the overapproximation ($OverReach[i]$) and underapproximation ($UnderReach[i]$) computed in lines 6 and 7 has any state in

unsafe set using SMT solvers like Z3. Moreover, this technique can prove that the system is unsafe and provide counterexamples from the model for SMT formula if satisfied.

Algorithm 2 has several advantages compared to the existing techniques for reachable set computation. First, the algorithm uses only $m+1$ numerical simulations, where m is the number of vectors in the set V. Second, it can compute reachable set not just for convex sets, but also for non-convex sets. Third, the initial set can be unbounded. Finally, the algorithm can compute underapproximation of the reachable set as well. Typical reachable set computation techniques require that the initial set is bounded and convex and specified in a special form like convex polyhedra, zonotopes, or ellipsoids. Moreover, techniques for computing underapproximation require special computation techniques and bounding the error for underapproximation is a challenging problem.

Notice that the formulas for computing overapproximation OA in line 6 and underapproximation UA in line 7 contain product terms of α_i and v_i. Hence, even for special initial sets like convex polyhedra, checking system safety using OA or UA involves reasoning about bilinear constraints, which is NP-hard. Moreover, our representation of UA has alternating quantifiers which adds to the challenges. To overcome these issues, we present a new overapproximation of the reachable set with a quantifiable bounded error, for initial sets that have special geometric properties like bounded convex polyhedra or ellipsoids. While we will not present a new underapproximation that avoids quantifier alternation, we will present a technique that can efficiently detect unsafety.

4 Faster Reachable Set Computation for Special Initial Sets

In this section, we present an algorithm for computing the reachable set when the initial set is given as a bounded convex polyhedron or an ellipsoid. For the presentation used in this paper, the set considered will be a polyhedra if the predicate P is given by linear inequalities $Cx \leq d$. Consider a bounded polyhedral initial set represented as $\Theta \triangleq \langle x_0, V, C, d\rangle$ where $C \in \mathbb{R}^{k\times n}$ is a $k \times n$ matrix, $d \in \mathbb{R}^k$. Recall that the set it represents is $[\![\Theta]\!] = \{x | x = x_0 + \Sigma_{i=1}^{n} \alpha_i v_i, C\bar{\alpha} \leq d\}$. For a bounded polyhedral set $[\![\Theta]\!]$, one can pick a state x_0 in the set Θ and an orthonormal basis V such that $max\{||\alpha_i||\} \leq \frac{1}{n}$. We assume that such a representation of the initial set Θ is provided. We now present a technique to compute a polyhedral representation of the overapproximation of the reachable set represented by the formula OA with quantifiers.

For a given time interval, assume that $R_0, R_1, \ldots, R_n$ are the regions returned by the $n+1$ validated simulations ($dia(R_i) \leq \epsilon$) and $OA(R_0, R_1, \ldots, R_n, C, d)$ gives the overapproximation predicate, defined in Eq. (7). For polyhedral initial set, the only nonlinear term in Eq. (7) for OA is the product term $\alpha_i v_i$. To eliminate this product term, we pick a fixed v_i (defined below), estimate the error in the resulting set given this fixed basis, and *bloat* the polyhedron based on this error analysis.

Theorem 3. *Given regions $R_0, R_1, \ldots, R_n$, and the set $OA(R_0, R_1, \ldots, R_n, C, d)$ defined according to Eq. (7), we have $OA \subseteq B_\delta(R)$ where $R \triangleq \langle x_0, V, C, d\rangle$ where $x_0 = center(R_0)$, $V = \{v_1, \ldots, v_n\}$ where $v_i = center(R_i) - center(R_0)$ and $\delta = 3\epsilon$ and $\epsilon = max_{i=0}^{n}\{dia(R_i)\}$.*

Proof. Consider a state $x' \in OA(R_0, R_1, \ldots, R_n, C, d)$, then there exists $x'_0 \in R_0$, $x_1 \in R_1, \ldots, x_n \in R_n$ where $v'_i = x'_i - x'_0$ such that $x' \in R' \triangleq \langle x'_0, \mathcal{U}_2, C, d\rangle$, $\mathcal{U}_2 = \{v'_1, \ldots, v'_n\}$.

Since $x' \in R'$, $\exists \alpha_1, \ldots, \alpha_n$ such that $x' = x'_0 + \alpha_1 v'_1 + \ldots + \alpha_n v'_n$. Consider the corresponding state $x \in R$ such that $x = x_0 + \alpha_1 v_1 + \ldots + \alpha_n v_n$. The distance between x and x' is given as:

$$
\begin{aligned}
||x - x'|| &= ||x_0 + \alpha_1 v_1 + \ldots + \alpha_n v_n - (x'_0 + \alpha_1 v'_1 + \ldots + \alpha_n v'_n)|| \\
&= ||(x_0 - x'_0) + \alpha_1(v_1 - v'_1) + \ldots + \alpha_n(v_n - v'_n)|| \\
&\leq ||x_0 - x'_0|| + \Sigma_{i=1}^{n}||\alpha_i|| \cdot ||v_i - v'_i|| \\
&\leq dia(R_0) + \Sigma_{i=1}^{n}||\alpha_i|| \cdot ||x_i - x'_i + (x'_0 - x_0)|| \\
&\leq dia(R_0) + \Sigma_{i=1}^{n}||\alpha_i|| \cdot (dia(R_i) + dia(R_0)) \\
&\leq dia(R_0) + \Sigma_{i=1}^{n}||\alpha_i|| \cdot 2\epsilon \\
&\leq dia(R_0) + \Sigma_{i=1}^{n} max\{\ ||\alpha_i||\ \} \cdot 2\epsilon \\
&\leq \epsilon + n \cdot \frac{1}{n} \cdot 2\epsilon \\
&\leq 3\epsilon
\end{aligned}
$$

Hence, the maximum distance between any two states x and x' is bounded by δ where $\delta = 3\epsilon$ and $x \in R$. Therefore $x' \in B_\delta(R)$.

Therefore for checking safety, instead of performing quantifier elimination, one can perform the following computations: (1) Compute the polyhedron R with $center(R_0)$ as the center, $center(R_i) - center(R_0)$ as the basis vectors, and predicate given as linear inequalities $C\bar{\alpha} \leq d$. Bloat the polyhedron R by the amount δ. Check for common states between unsafe region U and $B_\delta(R)$. Theorem 3 proves that if $B_\delta(R) \cap U = \emptyset$ then the reachable set does not have any unsafe state and hence the result is guaranteed to be sound.

Notice that the proof also gives a technique for checking when the system is unsafe. If $\exists x \in R$ such that $B_\delta(x) \subseteq U$, then it follows that for any choice of x'_0 and the basis vectors $\mathcal{U}_2$, the corresponding state x' is in the unsafe set. Therefore, the system is unsafe.

The proof for Theorem 3 can be extended to any general bounded sets and not necessarily for polyhedra. However, checking the safety with respect to general sets is computationally harder than checking for polyhedra. We consider one special case where the initial set is an ellipsoid. An ellipsoid can be defined as $E \triangleq \langle x_0, V, C, 1\rangle$ where $[\![E]\!] = \{x | x = x_0 + \alpha_1 v_1 + \ldots + \alpha_n v_n, \bar{\alpha}^T C \bar{\alpha} \leq 1\}$.

Corollary 1. *For initial set defined as $\Theta = \langle x_0, V, C, 1\rangle$ and given regions $R_0, R_1, \ldots, R_n$ for computing $OA(R_0, R_1, \ldots, R_n, C, 1)$ in Eq. (7), $OA \subseteq B_\delta(R)$*

where $R \triangleq \langle x'_0, V', C, 1\rangle$ *where* $x'_0 = center(R_0)$, $V' = \{v'_1, \ldots, v'_n\}$ *where* $v'_i = center(R_i) - center(R_0)$ *and* $\delta = 3\epsilon$ *and* $\epsilon = max_{i=0}^{n}\{dia(R_i)\}$.

5 Extension to Hybrid Systems

In this section, we outline the extension of the algorithm to hybrid systems. In principle, the Algorithm 2 computes the set of reachable states for a given continuous linear system for a given time interval. Therefore, one can essentially apply the algorithm used in tools like Phaver and SpaceEx for computing the reachable set of states for a hybrid system. For simplicity, we assume that all the invariants for the modes and guards for discrete transitions to be convex polyhedra and all the reset mappings to be linear functions. Under this assumptions, we present the algorithm for reachable set computation for hybrid system.

The algorithm performs the following three steps iteratively until the time horizon for verification. First, for the given mode and a given initial set, the algorithm computes the reachable set for that mode from that initial set for the bounded time specified using Algorithm 2. Second, the reachable set is pruned by removing all the states that violate the invariant. Third and lastly, the reachable set is checked to satisfy any guards for discrete transitions, and if so, the initial states for the next mode are computed by applying the reset map of the states that satisfy the guard predicate. As the reachable set of states for a hybrid system at a given time might belong to two different modes, we track the discrete transitions using a queue of set and location pairs.

input : Hybrid System: $\mathcal{A}$, Initial Set: Θ, Initial mode: m_0, Time bound: T
output: Bounded time reachable set: $ReachSet_{\mathcal{A}}(\Theta, T)$
1 regionQueue $\leftarrow \langle \Theta, m_0 \rangle$;
2 **for** *each* $\langle \Theta, m \rangle$ *in* regionQueue **do**
3 reachMode $\leftarrow Alg2(\Theta, m)$;
4 reachMode $\leftarrow$ reachMode $\cap$ Invariant$_m$;
5 nextRegions $\leftarrow$ discreteTransitions(reachMode);
6 $ReachSet_{\mathcal{A}}(\Theta, T) \leftarrow$ reachMode $\cup$ $ReachSet_{\mathcal{A}}(\Theta, T)$;
7 regionQueue.*append*(nextRegions);
8 **end**
9 **return** $ReachSet_{\mathcal{A}}(\Theta, T)$;

Algorithm 3. Algorithm that computes the reachable set for hybrid systems.

Algorithm 3 computes the reachable set for a hybrid system. As the problem in general is undecidable, the loop need not terminate. The main loop that performs the three key steps iteratively happens from line 2 to line 8. Line 3 computes the reachable set of states from Θ for the corresponding mode using Algorithm 2. Line 4 checks the invariant for the reachable set and line 5 computes the states reached after discrete transitions. Although we present the algorithm here, in this paper, we perform experiments on purely continuous system to demonstrate the efficiency of Algorithm 2.

6 Experiments

To demonstrate the applicability of the proposed approach, we have implemented this algorithm as an extension of the tool C2E2 [11]. C2E2 is a dynamic analysis tool that implements a simulation based verification algorithm for nonlinear hybrid systems where the model is annotated with discrepancy functions. Unlike C2E2, this approach would not require the linear systems model to be provided with a discrepancy function. For generating the validated simulations, C2E2 uses a validated numerical integration engine called CAPD [1]. As the systems considered in this paper are restricted to linear systems, instead of using CAPD, we use the numerical integration engine ODEINT[1], which is a part of BOOST libraries. Unlike CAPD, ODEINT does not provide validated simulations, therefore, for computing rigorous bounds on the numerical simulation, we use error analysis provided in [4] for the 4th order Runge-Kutta method that is used in our experiments.

The experimental section is divided into 3 parts. First, we verify the safety property of several high dimensional linear time invariant systems with polyhedral initial sets and polyhedral unsafe sets. For checking the intersection of the reachable set computed with the unsafe states, we use GLPK library[2]. Second, we consider several linear time varying systems. Finally, we verify safety property of linear time invariant systems with non-convex initial and unsafe sets and also for unbounded initial and unsafe sets. All experiments were performed on a system with i7 Quad-core processor with 8GB memory running Ubuntu 11.10.

6.1 High Dimensional Linear Time Invariant Systems

We compare the performance of our approach with the state-of-the-art tool for linear systems verification SpaceEx on several high dimensional linear systems. Though the reachability computation can be extended to hybrid systems (Sect. 5 in Appendix), our experiments here are restricted to continuous systems; we believe our main contribution is the algorithm for reachability for continuous systems with the extension to hybrid systems being standard. The experimental results are provided in Table 1. The tank system considered in Table 1 is one of the examples provided with SpaceEx. In this example, the water level in tank i is model as a continuous variable x_i. The tank i leaks into tank $i+1$ and the rate of leakage is proportional to the water level in tank i, making the system a linear system. The 28 dimensional helicopter system and the 9 dimensional insulin system are also part of the examples provided by SpaceEx. The platoon system is a controller for stabilizing a platoon of vehicles and is obtained from [19].

The experiments show that our approach outperforms SpaceEx by at least an order of magnitude. This is mainly because as the number of dimensions increases, the complexity of representing the reachable set of states as a support function increases exponentially. Whereas in our approach, the number of

[1] http://headmyshoulder.github.io/odeint-v2/.
[2] https://www.gnu.org/software/glpk/.

simulations performed only increases linearly with the number of dimensions and the representation of the reachable set is just a basis transformation of the representation of the initial set of states considered. Also, notice that the time taken for computing the validated simulations using the approach in [4] takes the majority of the verification time as opposed to checking the safety of reachable set. An advantage our approach enjoys over SpaceEx is that we can compute underapproximations and hence conclude that the system is unsafe and provide counterexamples. We however note that for the experiments in Table 1, the results reported by SpaceEx were indeed consistent with the results reported by our approach.

Table 1. Experimental results for verification of high dimensional linear time invariant systems. Vars: number of variables, TH: time horizon for verification, Sims: total number of simulations, Simu. time: time taken for simulations, Verif. result: result of verification. TO: time out for 5 min.

Benchmark	Vars.	TH	Sims.	Simu. time.	Verif. time	C2E2	SpaceEx	Verif. result
Insulin	8	10	9	0.157 s	0.049 s	0.206 s	8.07 s	Safe
Insulin	8	10	9	0.166 s	0.034 s	0.2 s	7.89 s	Unsafe
Platoon	10	25	11	0.337 s	0.019 s	0.356 s	TO	Safe
Platoon	10	25	11	0.323 s	0.019 s	0.342 s	TO	Unsafe
Tank-10	10	20	11	0.745 s	0.206 s	0.951 s	4.886 s	Safe
Tank-10	10	20	11	0.721 s	0.19 s	0.911 s	4.992 s	Unsafe
Tank-15	15	20	16	1.325 s	0.363 s	1.688 s	8.176 s	Safe
Tank-18	18	20	19	1.705 s	0.569 s	2.274 s	10.466 s	Safe
Helicopter	28	20	29	3.192 s	1.634 s	4.826 s	2m1.66s	Safe

6.2 Verifying Linear Time Varying Systems

Typical approaches for computing reachable set for linear time varying systems differ considerably from that of linear time invariant systems. Therefore, there is a lack of tools that are geared towards verifying linear time varying systems. For the experimental evaluation, we model the linear time varying system as a nonlinear system with *time* as a variable t and compare the results of our approach with the tool Flow* [6] that can verify nonlinear systems. The experimental results are provided in Fig. 3(a). The tank system in Fig. 3(a) is similar to the linear time invariant system, except that water is being pumped into Tank 1 at a rate that decreasing with time. Therefore, the differential equation governing the dynamics contains t^{-1} term which makes it non polynomial. The second example is a modified version of the uncertain linear system from [3].

The experiments show that our approach outperforms Flow* by at least an order of magnitude. Also, similar to SpaceEx, the time taken by Flow* increases exponentially as the number of dimensions in the system increases, whereas in

Benchmark.	Vars.	Flow*.	C2E2
Tank	2	1.56 s	0.132 s
Tank	4	4.28 s	0.198 s
Tank	6	9.41 s	0.287 s
Tank	8	18.73 s	0.356 s
Tank	10	33.67 s	0.484 s
LTV [3]	5	7.51 s	0.24 s
LTV	7	12.09 s	0.31 s
LTV	9	18.18 s	0.4 s

(a) Verifying linear time varying systems

Benchmark.	Dim.	TH.	Init. Set.	Res.	Time
ACC [25]	3	2	NC	Safe	2.185
ACC	3	2	UB	Safe	1.774
ACC	3	2	NC	Unsafe	1.11
ACC	3	2	UB	Unsafe	1.01
Tank	5	1	NC	Safe	2.717
Tank	5	1	UB	Safe	2.145
Tank	5	1	NC	Unsafe	1.722
Tank	5	1	UB	Unsafe	1.519

(b) Verifying nonconvex and unbounded initial sets.

Fig. 3. Verification of linear time varying systems and non-convex and unbounded initial sets. Res.: verification result. NC: nonconvex initial set, UB: unbounded initial set

our approach, the number of simulations required increases only linearly with the number of dimensions.

6.3 Non-convex and Unbounded Initial Sets

An advantage of our approach is that we can compute the reachable set when the initial set of states is non-convex and also when the initial set is unbounded. To demonstrate this, we compute reachable set of states for several benchmark examples given in Fig. 3(b). In these experiments, we consider non-convex and unbounded initial sets symbolically represented as conjunctions of polynomial inequalities. We use Z3 [8] SMT solver for performing quantifier elimination and inferring whether the system is safe or unsafe. As the complexity of quantifier elimination over reals is exponentially more than linear real arithmetic, the time taken for verification is more than for polyhedral initial sets even for low dimensional systems. Unlike the existing theorem proving based approaches which can verify non-convex or unbounded initial set that requires some manual effort, our approach is completely automatic.

6.4 Discussion

It is evident from Table 1 and Fig. 3(a), (b) that our approach outperforms the existing approaches. Furthermore, our technique works for computing both over-approximation and underapproximation for linear time invariant and linear time varying systems. In case of polyhedral initial and unsafe sets, notice from Table 1 that the time taken for verification is only a fraction of the time taken from simulations. Given two bounded polyhedral initial sets $\Theta_1 \triangleq \langle x_0, V, C_1, d_1 \rangle$ and $\Theta_2 \triangleq \langle x_0, V, C_2, d_2 \rangle$, with the same center x_0 and the same set of basis vectors V,

the reachable set computation technique *need not* generate $n+1$ simulations Θ_1 and $n+1$ simulations for Θ_2. Instead, it can *reuse* the same set of simulations runs used for Θ_1 and compute the reachable set for Θ_2 thus reducing the number of simulations *per* verification. This would also bring down the total time for verification as computing simulations is computationally more expensive than verifying safety. Furthermore, given k bounded polyhedral initial sets, $\Theta_1, \ldots, \Theta_k$, by performing k coordinate transformations one can represent these sets with a common center and basis vectors and the amortized number of simulations for verification would be $\frac{n+1}{k}$ where n is the number of dimensions of the system. This is a significant advantage of our approach as opposed to the reachable set computation performed by SpaceEx, where, a change in the initial set would require discarding the reachable set computed and recomputing the new reachable set from scratch.

References

1. Computer assisted proofs in dynamic groups (capd). http://capd.ii.uj.edu.pl/index.php
2. Althoff, M.: Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control, pp. 173–182. ACM (2013)
3. Althoff, M., Le Guernic, C., Krogh, B.H.: Reachable set computation for uncertain time-varying linear systems. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control, pp. 93–102. ACM (2011)
4. Bouissou, O., Martel, M.: Grklib: a guaranteed runge kutta library. In: 12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics, SCAN 2006, p. 8. IEEE (2006)
5. Chen, X., Abraham, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: RTSS (2012)
6. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013)
7. Chutinan, A., Krogh, B.H.: Computational techniques for hybrid system verification. IEEE Trans. Autom. Control **48**, 64–75 (2003)
8. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
9. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 174–189. Springer, Heidelberg (2007)
10. Duggirala, P.S., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: Proceedings of the 13th International Conference on Embedded Software (EMSOFT 2013), Montreal, Canada (2013)
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)
12. Frehse, G.: PHAVer: algorithmic verification of hybrid systems past HyTech. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005)

13. 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: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011)
14. Girard, A.: Reachability of uncertain linear systems using zonotopes. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 291–305. Springer, Heidelberg (2005)
15. Julius, A.A., Fainekos, G.E., Anand, M., Lee, I., Pappas, G.J.: Robust test generation and coverage for hybrid systems. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 329–342. Springer, Heidelberg (2007)
16. 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)
17. Kurzhanski, A.B., Varaiya, P.: Ellipsoidal techniques for reachability analysis: internal approximation. Syst. Control Lett. **41**(3), 201–211 (2000)
18. Le Guernic, C., Girard, A.: Reachability analysis of hybrid systems using support functions. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 540–554. Springer, Heidelberg (2009)
19. Makhlouf, I.B., Kowalewski, S.: Networked cooperative platoon of vehicles for testing methods and verification tools. In: Applied Verification for Continuous and Hybrid Systems. CPS-VO (2014)
20. Mitra, S., Archer, M.: PVS strategies for proving abstraction properties of automata. Electron. Notes Theor. Comput. Sci. **125**(2), 45–65 (2005)
21. Platzer, A., Quesel, J.-D.: KeYmaera: a hybrid theorem prover for hybrid systems (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 171–178. Springer, Heidelberg (2008)
22. Prabhakar, P., Viswanathan, M.: A dynamic algorithm for approximate flow computations. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control, pp. 133–142. ACM (2011)
23. 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)
24. Taly, A., Tiwari, A.: Deductive verification of continuous dynamical systems. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, 15–17 December 2009, pp. 383–394. IIT Kanpur, India (2009)
25. Tiwari, A.: Approximate reachability for linear systems. In: Maler, O., Pnueli, A. (eds.) HSCC 2003. LNCS, vol. 2623, pp. 514–525. Springer, Heidelberg (2003)
26. Tiwari, A.: HybridSAL relational abstracter. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 725–731. Springer, Heidelberg (2012)