Conic Abstractions for Hybrid Systems

Sergiy Bogomolov[1,2], Mirco Giacobbe[2], Thomas A. Henzinger[2], and Hui Kong[2]

[1] Australian National University, Canberra, Australia
[2] IST Austria, Klosterneuburg, Austria

Abstract. Despite researchers' efforts in the last couple of decades, reachability analysis is still a challenging problem even for linear hybrid systems. Among the existing approaches, the most practical ones are mainly based on bounded-time reachable set over-approximations. For the purpose of unbounded-time analysis, one important strategy is to abstract the original system and find an invariant for the abstraction. In this paper, we propose an approach to constructing a new kind of abstraction called conic abstraction for affine hybrid systems, and to computing reachable sets based on this abstraction. The essential feature of a conic abstraction is that it partitions the state space of a system into a set of convex polyhedral cones which is derived from a uniform conic partition of the derivative space. Such a set of polyhedral cones is able to cut all trajectories of the system into almost straight segments so that every segment of a reach pipe in a polyhedral cone tends to be straight as well, and hence can be over-approximated tightly by polyhedra using similar techniques as *HyTech* or *PHAVer*. In particular, for diagonalizable affine systems, our approach can guarantee to find an invariant for unbounded reachable sets, which is beyond the capability of bounded-time reachability analysis tools. We implemented the approach in a tool and experiments on benchmarks show that our approach is more powerful than *SpaceEx* and *PHAVer* in dealing with diagonalizable systems.

Keywords: affine system, hybrid system, reachability analysis, conic abstraction, discrete abstraction

1 Introduction

Hybrid systems [1,2] are systems that admit interacting discrete and continuous dynamics. Reachability analysis of hybrid systems has been a major research issue over the past couple of decades [3,4,5,6,7,8]. An important part of the effort has been devoted to hybrid systems where the continuous dynamics is described by linear or affine differential equations or inclusions. For the purpose of efficient computation, a number of representations of convex set have been proposed, including polyhedrons [9,10], ellipsoids [11,12], hyperrectangles [13], zonotopes [14,15], and support functions [16]. A common feature of these approaches is that all of them apply only to reachability analysis with bounded continuous time although sometimes a fixed point could be found.

For the purpose of unbounded-time analysis, a very useful strategy is to use lightweight runtime technique for continuous online verification[17,18], and another important strategy is to abstract the original system and find an invariant for the abstraction [19]. However, obtaining a high-quality abstraction automatically for the original system is challenging by itself and this is why *PHAVer* chooses to leave this important work to users, who have some domain expertise available for this purpose [20]. Roughly speaking, the ultimate goal of abstraction is to use a partition of the state space which is as coarse as possible, to derive an over-approximation of the original system which is as accurate as possible and allows a computation of the reachable state set which is as efficient as possible. Depending on the set representation that is used, the schemes that have been proposed for state space partition vary significantly [21,5,22,23,24,25,26,27,28]. When polyhedra are used for the set representation of states, a guiding principle for state space partitioning is that the partition should result in a set of regions that are as "straight" as possible. By "straight region", we mean that the maximal angle between the derivative vectors in that region (which we define as the *twisting* of the region) is small, so that every trajectory tends to be straight in the region. The benefit of straight regions is that they can be over-approximated accurately by polyhedra. However, for a given system, obtaining the least number of straight regions under a given threshold of twisting is by no means trivial.

With this principle in mind, we propose a new abstraction called *conic abstraction* for affine hybrid systems and we compute reachable state sets based on the abstraction. Given an n-D linear system defined by $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$, assume that $\boldsymbol{A}$ is an invertible matrix (note that any affine system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ can be transformed into a linear system under this assumption). The basic idea behind conic abstraction is as follows. First, the derivative space of the system is partitioned uniformly into a set $\boldsymbol{D}$ of convex polyhedral cones. Then, $\boldsymbol{D}$ is mapped back from the derivative space to the state space to obtain a conic partition $\boldsymbol{C}$ of state space, i.e., $\forall C_i \in \boldsymbol{C} : \exists D_i \in \boldsymbol{D} : C_i = \{\boldsymbol{A}^{-1}\boldsymbol{y} \mid \boldsymbol{y} \in D_i\}$. Finally, every state region C_i is treated as a discrete location ("mode") and the discrete transitions between these modes are decided on-the-fly according to whether there exists a trajectory between them. By doing so, we can easily obtain the differential inclusion D_i for each polyhedral cone C_i. Therefore, for any subset I_i of C_i, the reachable set of I_i in C_i can be overapproximated by $(I_i \oplus D_i) \cap C_i$, where $\oplus$ denotes the Minkowski sum. More importantly, since the twisting of C_i is determined by the maximal angle of D_i, the partition can be refined easily to any desired precision, by shrinking the maximal angle of the conic partition of the derivative space. Note that an important feature of C_i is that it is an unbounded set, however, with a bounded twisting, which means that each C_i captures infinitely long trajectories only if they are straight enough. *Diagonalizable* affine systems, for which the matrix $\boldsymbol{A}$ is diagonalizable, form such a class of systems, because for diagonalizable systems all trajectories eventually evolve into approximately straight lines.

Using properties of diagonalizable affine systems, we develop an algorithm that constructs a conic abstraction as a directed acyclic graph (DAG) for which

an invariant (i.e., an over-approximation of the reachable state set) exists and the computation of the invariant is guaranteed to terminate. The algorithm is implemented in a tool and experiments on randomly generated examples as well as published benchmarks show that our approach is more powerful than *PHAVer* in finding unbounded invariants. Note that computing an unbounded invariant for diagonalizable affine systems lies beyond the capability of tools for time-bounded reachability analysis, such as *SpaceEx* [29].

The main contributions of this paper are as follows. First, we propose conic abstractions and a method for constructing them for affine hybrid systems. The core idea lies in deriving a state space partition from a uniform partition of the derivative space. Second, we develop an algorithm for building conic abstractions as DAGs for diagonalizable affine systems and for computing invariants on these abstractions. Finally, we implement and evaluate our approach in a tool.

The paper is organized as follows. Section 2 is devoted to preliminary definitions. In Section 3, we introduce conic abstractions for affine systems. In Section 4, we show how to construct conic abstractions as DAGs for diagonalizable systems. Section 5 describes how we compute invariants for continuous systems and affine hybrid systems. In Section 6, we present our experimental results. Finally, we conclude with Section 7.

2 Preliminaries

In this section, we recall some concepts used throughout the paper. We first clarify some notation conventions. We use bold uppercase letters such as $\boldsymbol{A}$ to denote matrices and bold lowercase letters such as $\boldsymbol{b}$ to denote vectors and $diag(\lambda_1, \cdots, \lambda_n)$ to denote a diagonal matrix with $\lambda_1, \cdots, \lambda_n$ as its diagonal elements. We call a dynamical system defined as $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ an affine system and we use a superscript T for the transpose of a matrix.

Definition 1 (Affine System). *An n-dimensional affine system consists of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$, which define the vector flow $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$, and an initial region $X_0 \subseteq \mathbb{R}^n$ defined by a polyhedron.*

Whenever the initial set is immaterial, we refer to an affine system just as to $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$. We next introduce the concept of Lie derivative.

Definition 2 (Lie derivative). *For a given polynomial $p \in \mathbb{K}[\boldsymbol{x}]$ and a continuous system $\dot{\boldsymbol{x}} = \boldsymbol{f}$, the **Lie derivative** of $p \in \mathbb{K}[\boldsymbol{x}]$ along $\boldsymbol{f}$ is defined as $\mathcal{L}_{\boldsymbol{f}} p \stackrel{def}{=} \langle \nabla p, \boldsymbol{f}^T \rangle$.*

For an affine system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, we can simply write the Lie derivative as $\mathcal{L}_{\boldsymbol{A}}\langle \boldsymbol{a}, \boldsymbol{x}\rangle = \langle \boldsymbol{a}\boldsymbol{A}, \boldsymbol{x}^T\rangle + \langle \boldsymbol{a}, \boldsymbol{b}^T\rangle$. We call a polyhedral cone C an intersection of linear inequalities of the form $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \leq 0$, and we denote its boundary as ∂C. For $X, Y \subseteq \mathbb{R}^n$, $X \oplus Y$ denotes their Minkowski sum $\{\boldsymbol{x} + \boldsymbol{y} \colon \boldsymbol{x} \in X \text{ and } \boldsymbol{y} \in Y\}$, and for $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $X \subseteq \mathbb{R}^n$, $\boldsymbol{A}X$ denotes the linear transformation $\{\boldsymbol{A}\boldsymbol{x} \colon \boldsymbol{x} \in X\}$.

3 Conic Abstractions of Affine Systems

Discrete abstraction is a basic strategy for verifying continuous and hybrid systems. There are many abstraction approaches proposed for this purpose. Rectangular abstraction [19,5,30] and nonlinear abstraction [22,23,24,26] are widely used. However, even for linear systems, the existing abstraction approaches are still inefficient. In this section, focusing on linear systems, we propose a new abstraction approach called *conic abstraction*. However, since every affine system can be transformed into an equivalent linear system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$, as we discuss in Sec. 4, our discussion applies to affine systems too.

The idea is that we partition the state space of a linear system into a set of convex polyhedral cones. We call this set a *conic partition*.

Definition 3 (Conic Partition). *A conic partition is a set of polyhedral cones Δ such that $\cup_{C_i \in \Delta} C_i = \mathbb{R}^n$ and every two cones $C_1, C_2 \in \Delta$ have disjoint interiors, i.e., $(C_1 \backslash \partial C_1) \cap (C_2 \backslash \partial C_2) = \emptyset$.*

We call an element of the partition $C \in \Delta$ a region. Then we construct a graph whose vertices correspond to partition regions and edges indicate possible flow between them. We call such a graph a *conic abstraction*.

Definition 4 (Conic Abstraction). *The conic abstraction of the linear system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ derived from the conic partition Δ consists of the finite directed graph (L, E) as follows. Every vertex $l_C \in L$ corresponds to one and only one cone $C \in \Delta$. There exists an edge $(l_{C_1}, l_{C_2}) \in E$ if and only if there exists a plane $F_1 = \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x}^T \rangle = 0\}$ such that 1) $\partial C_1 \cap \partial C_2 \subseteq F_1$, 2) $C_1 \subseteq \{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x}^T \rangle \leq 0\}$, 3) the Lie derivative of $\langle \boldsymbol{a}, \boldsymbol{x}^T \rangle$ is non-negative at some common point, i.e., $\mathcal{L}_{\boldsymbol{A}} \langle \boldsymbol{a}, \boldsymbol{x}^T \rangle \geq 0$ for some $\boldsymbol{x} \in \partial C_1 \cap \partial C_2$.*

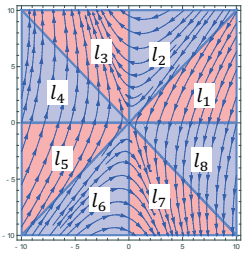
We elaborate on how to construct a conic abstraction for diagonalizable systems in Sec. 4. A conic abstraction can be seen as a Linear Hybrid Automaton (LHA, [1]), whose locations l_C are such that its invariant is given by C, its flow is given by a differential inclusion defined as $\dot{\boldsymbol{x}} \in \boldsymbol{A}C$, and whose switch guards consist of the common facet of the respective adjacent cones.

Example 1 (running example). Consider the linear system described by $\dot{x} = -2x - 2y, \dot{y} = -5x + y$. A conic partition of the state space, the corresponding differential inclusion and the conic abstraction of the system is shown in Fig. 1a, Fig. 1b and Fig. 1c, respectively. As you can see, both the invariant and the differential inclusion of each location are polyhedral cones. □

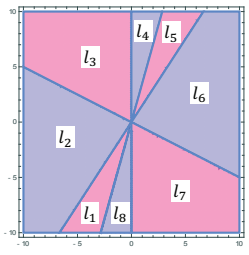
Similarly as for the symbolic reachability analysis of LHA [2], the set of states that are reachable from an initial set $X \subseteq \mathbb{R}^n$ through the continuous flow at location $l_C \in L$ corresponding to $C \in \Delta$ is given by

$$(X \oplus \boldsymbol{A}C) \cap C. \tag{1}$$

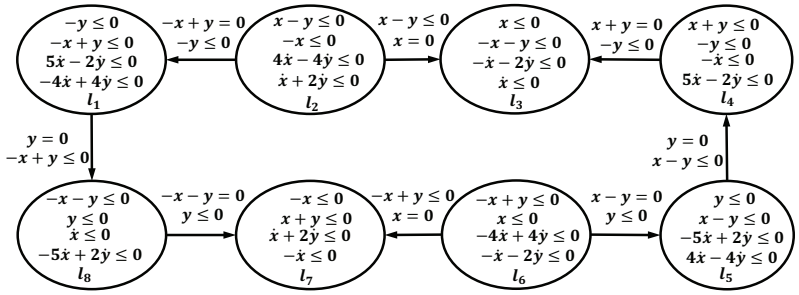
A conic abstraction represents an overapproximation of the system, whose tightness depends on the maximum angle between any two points in the cone $\boldsymbol{A}C$



(a)



(b)



(c)

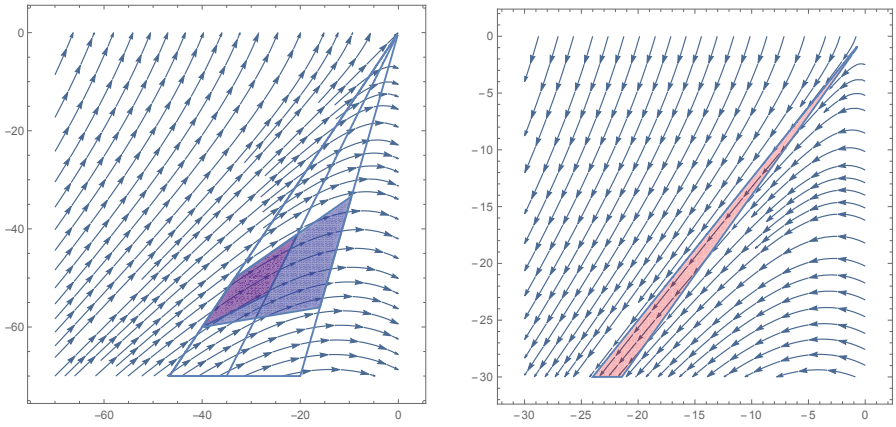
Fig. 1: Example 1. (a) Conic partition of state space. (b) Conic differential inclusion. (c) Conic abstraction of the system.

in derivative space. Roughly speaking, the more acute the cone $\boldsymbol{AC}$ in derivative space, the more accurate the overapproximation. Fig. 2a shows a comparison between conic partitions with different accuracies (depicted in two different shades) for the same initial region. We encapsulate the accuracy given by a partition with the notion of twisting.

Definition 5 (Twisting of a state region). *Let $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ be a linear system and $P \subseteq \mathbb{R}^n$ be a (not necessarily conic) region of the state space. Then P is said to have a twisting of θ (or to be θ twisted) if it satisfies that*

$$\sup_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in P} \arccos\left(\frac{\langle \dot{\boldsymbol{x}}_1, \dot{\boldsymbol{x}}_2 \rangle}{\|\dot{\boldsymbol{x}}_1\| \|\dot{\boldsymbol{x}}_2\|}\right) = \theta. \tag{2}$$

Intuitively, a cone with smaller twisting allows only trajectory segments that are almost straight, inducing a more accurate overapproximation. In the con-



(a) **(b)**

Fig. 2: (a) Overapproximation inside different cones. The smaller the cone, the more precise the overapproximation. (b) A cone capable of offering accurate overapproximation for unbounded reach pipe.

text of conic abstraction, properly inducing smaller and smaller twistings induce refinements of the abstraction, providing a better overapproximation.

Definition 6 (Conic abstraction refinement). *Given two conic abstractions* (L_1, E_1) *and* (L_2, E_2) *for a linear system* $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$, (L_2, E_2) *refines* (L_1, E_1) *if* $|L_2| > |L_1|$ *and for all* $l_1 \in L_1$ *with cone* C_1 *there does always exist* $l_2^1, \ldots, l_2^m \in L_2$ *with cones* $C_2^1, \ldots, C_2^m$ *such that* $C_1 = C_2^1 \cup \cdots \cup C_2^m$.

It is subject of Sec. 4 how to generate abstraction refinements by tuning the value of twisting.

The property we desire is that the twisting of every state partition is bounded by a small angle θ. A common strategy to achieve this goal is to split the state space into small rectangles iteratively until the twisting of each rectangle falls below θ [19,31,30]. However, such strategy is inefficient, as the twisting may not change uniformly in a rectangular partition. On the contrary, a conic partition naturally enjoys bounded twisting using unbounded regions. This allows a conic partition to accurately overapproximate both bounded and unbounded reach pipes, if in the latter case the trajectories are straight enough. Figure 2b shows such an example, where the tiny cone overapproximates all trajectories entering it, as they tend to be parallel to its left boundary.

3.1 Conic abstractions derived from derivative space partitions

In existing work on discrete abstraction of continuous systems, to obtain a high-quality state space partition, the focus is mostly placed on state space. However,

what really matters here is the derivative space. Therefore, our state space partition should be derived from a derivative space partition. Given a continuous system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, every convex cone D in the derivative space with a maximal angle θ corresponds to a set C of states which has a twisting of θ. Moreover, C can be obtained through simple substitution. However, for nonlinear systems, C is nonlinear and is hard to handle, so we leave it for future work.

We assume that the systems under consideration are linear. To derive a conic abstraction for an n-dimensional linear system, we first partition the whole derivative space into a set Ω of convex polyhedral cones which satisfies that

1. $\bigcup_{D_i \in \Omega} D_i = \mathbb{R}^n$;
2. $\forall D_i, D_j \in \Omega : (D_i \backslash \partial D_i) \bigcap (D_i \backslash \partial D_j) = \emptyset$;
3. $\forall D_i \in \Omega : \sphericalangle D_i \leq \theta$, where $\sphericalangle D_i$ denotes the maximal angle of D_i (i.e. the maximal angle between the vectors in D_i) and θ is a given bound.

By mapping Ω back to the state space, we can obtain another set Δ of state regions. The property of Δ is formalized in the following theorem.

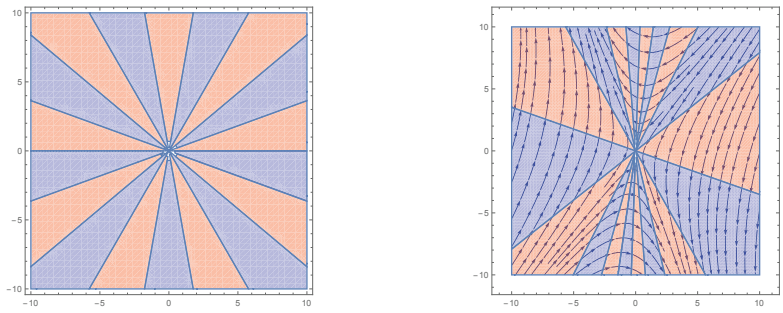
Theorem 1. *Given a linear system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ let Ω be a set of convex polyhedral cones defined as above and $\Delta = \{\boldsymbol{A}^{-1} D \mid D \in \Omega\}$. Then,*

1. *every $C_i \in \Delta$ is a convex polyhedral cone and the twisting of C_i is θ-bounded;*
2. *$\bigcup_{C_i \in \Delta} C_i = \mathbb{R}^n$;*
3. *$\forall C_i, C_j \in \Delta : (C_i \backslash \partial C_i) \bigcap (C_i \backslash \partial C_j) = \emptyset$;*

Remark 1. According to Theorem 1, we know that, given any linear system $\mathcal{H}$ with an invertible matrix $\boldsymbol{A}$ and a θ-bounded conic partition Ω of the derivative space, a conic partition Δ for the state space with θ-bounded twisting can be obtained by a linear transformation. Note that the twisting of C_i is θ-bounded does not mean that C_i is θ-bounded. Conversely, the maximal angle of each cone C_i varies significantly depending on how straight the trajectories are in that cone. Roughly speaking, the straighter the trajectories are, the larger the maximal angle of C_i is, provided that the twisting is the same. □

Now, let us get back to the issue of generating a conic partition of the derivative space. Our approach borrows the idea of slicing watermelons. Concretely, given an n-dimensional derivative space, we first choose a group of seed planes passing through the origin and then generate a cluster of planes by rotating each seed plane counterclockwise around an independent axis by a fixed angle θ_1, step by step until no further θ_1 rotation is possible. Finally, the whole vector space can be sliced into a set of convex polyhedral cones by the generated planes and each of them is θ_2-bounded for some θ_2. By mapping these cones into the state space, we can achieve a conic partition of θ_2-bounded twisting for the state space. The following example shows how a conic state space partition derived from a uniform derivative space partition looks like.

Example 2 (running example). Consider the following linear system $\mathcal{H}$ described by $\dot{x} = -2x - 2y, \dot{y} = -5x + y$. As shown in Fig. 3a, the derivative space is first



(a) (b)

Fig. 3: Example 2 (a) Uniformly conic partition of the the derivative space. (b) Conic partition of state space derived from the derivative space partition.

uniformly partitioned into 18 cones. Then, these cones are mapped into the state space. As can be seen in Fig. 3b, in every cone, the straighter the trajectories are, the larger the maximal angle of the cone is. □

The reachable set computation of a conic abstraction is a basic operation of linear hybrid automata. As usual, due to the undecidable nature of the issue, the reachable set computation of a conic abstraction cannot guarantee to terminate for a general linear system. However, for the conic abstraction of a specific class of systems, the reachable set computation can be guaranteed to terminate, which is shown in the next section.

4 Diagonalizable Systems

In this section, we focus on a class of affine systems for which the matrix used to describe the system dynamics is diagonalizable in $\mathbb{R}$, called diagonalizable systems. The reason why diagonalizable systems are interesting is that, given a conic abstraction, the reachable set computation is guaranteed to terminate. Formally, a diagonalizable system is defined as follows.

Definition 7 (Diagonalizable system). *An affine system* $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ *is diagonalizable if there exist a real matrix* Q *such that* $Q^{-1}\boldsymbol{A}Q = \boldsymbol{diag}(\lambda_1, \cdots, \lambda_n)$*, where* $\lambda_i \in \mathbb{R}, \lambda_i \neq 0, i = 1, \cdots, n$*.*

In the following, we introduce how to derive a conic abstraction for a diagonalizable system and how to overapproximate their reachable sets by the conic abstraction. We also extend the theory to hybrid affine systems.

4.1 Properties of diagonalizable systems

The most important feature of diagonalizable system is that all of their eigenvalues are real numbers. Given a diagonalizable affine system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$ with initial region X_0, by doing a translation on the coordinate system with $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{A}^{-1}\boldsymbol{b}$, we can always transform the system into a linear system $\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y}$ with initial region $Y_0 = X_0 \oplus \{\boldsymbol{A}^{-1}\boldsymbol{b}\}$. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_j$ be the corresponding eigenvectors respectively, then the general solution of the linear system can be written as (refer to [32])

$$\boldsymbol{x}(t) = c_1 e^{\lambda_1 t}\boldsymbol{u}_1 + \cdots + c_n e^{\lambda_n t}\boldsymbol{u}_n \tag{3}$$

where $c_1, \ldots, c_n$ depends on the initial value $\boldsymbol{x}_0$ of the system of differential equations and can be obtained by solving $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Let $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ and $\boldsymbol{c} = (c_1, \ldots, c_n)$, $Cone(\boldsymbol{c}, \boldsymbol{U}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \sum_{i=1}^{n} t_i c_i \boldsymbol{u}_i, t_i \geq 0\}$ denote the convex polyhedral cone generated by the vectors $c_1\boldsymbol{u}_1, \ldots, c_n\boldsymbol{u}_n$. Then, we have the following theorem.

Theorem 2. *Given a diagonalizable system $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, let $\boldsymbol{U}$ be defined as above and $\Xi = \{-1, 1\}^n$. Then, for every $\boldsymbol{\xi} \in \Xi$, $Cone(\boldsymbol{\xi}, \boldsymbol{U})$ is an invariant and the twisting of $Cone(\boldsymbol{\xi}, \boldsymbol{U})$ is bounded by radian π.*

Remark 2. According to Theorem 2, the state space of a diagonalizable system can always be partitioned into a set of invariant cones and the twisting of every invariant cone is bounded by radian π. Therefore, given a diagonalizable system, to overapproximate the reachable set, we do not have to construct a conic abstraction for the whole state space. Instead, we only need to figure out which invariant cones the initial set spans and then construct a conic abstraction for each of them respectively. As mentioned previously, we would start from partitioning the derivative space. Based on the property of diagonalizable system, we develop a partitioning scheme which can construct a conic abstraction as a directed acyclic graph.

4.2 Diagonalization and conic partition

The first step of constructing a conic partition consists of diagonalizing the original system. Given a diagonalizable system $\dot{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{y}$ with initial region Y_0, a diagonalization of it is a linear system $\dot{\boldsymbol{z}} = \boldsymbol{A}_\lambda \boldsymbol{z}$ with initial region Z_0 where $\boldsymbol{A}_\lambda = \boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}$ is a diagonal matrix and $Z_0 = \boldsymbol{Q}^{-1}Y_0$ for some $\boldsymbol{Q}$. In theory, the diagonalized system is equivalent to the original system in terms of safety verification. However, by doing diagonalization, we manage to transform every invariant cone and its derivative cone into an independent orthant respectively. Since an orthant as a cone has some good properties such as having a fixed maximal angle of $\frac{\pi}{2}$ and all the generating vectors of the invariant cones are orthogonal to each other, we propose a special conic partition scheme, called *radial partition*, which can result in a directed acyclic graph for the conic abstraction.

Given a diagonalized n-dimensional system $\dot{\boldsymbol{z}} = \boldsymbol{A}_{\boldsymbol{\lambda}}\boldsymbol{z}$ and an orthant $\boldsymbol{O} = \{\boldsymbol{z} \in \mathbb{R}^n \mid \boldsymbol{B}\boldsymbol{z} \leq \boldsymbol{0}\}$ in derivative space, where $\boldsymbol{B} = diag(b_{11}, \ldots, b_{nn})$ with

$b_{ii} = 1$ or -1. Let $\boldsymbol{B}_i, \boldsymbol{B}_j$ be the i'th and j'th row vectors of $\boldsymbol{B}$ respectively, where $i \neq j$. The basic idea of *radial partition* is as follows. For every pair of $(\boldsymbol{B}_i, \boldsymbol{B}_j)$, we generate a sequence of vectors $S_{ij} : \boldsymbol{v}_{ij1}, \ldots, \boldsymbol{v}_{ij(K_{ij}+1)}$ by rotating the vector $\boldsymbol{v}_{ij1} = \boldsymbol{B}_j$ from $\boldsymbol{B}_i$ to $\boldsymbol{B}_j$ step by step with an rotating amplitude $\frac{\pi}{2K_{ij}}$. Then, S_{ij} is used as the sequence of normal vectors of partitioning planes. Thus, each pair of adjacent vectors $\boldsymbol{v}_{ijk}, \boldsymbol{v}_{ijk+1}$ forms a slice $\{\boldsymbol{z} \in \mathbb{R}^n \mid \langle \boldsymbol{v}_{ijk}, \boldsymbol{z}^T \rangle \leq 0, \langle -\boldsymbol{v}_{ijk+1}, \boldsymbol{z}^T \rangle \leq 0\}$ of the orthant O and O will be partitioned into K_{ij} slices by all the planes formed by S_{ij}. Hence, we can get $\frac{n(n-1)}{2}$ ordered sequences of planes at most totally. These planes intersecting each other yield a conic partition D for the orthant O. However, we do not really need so many sequences of partitioning planes. Actually, $n-1$ sequences of planes suffices to construct a partition with an arbitrarily small maximal angle.

For the conic abstraction derived from radial partition, we have the following theorem.

Theorem 3. *Every conic abstraction derived from a radial partition of the derivative space is a directed acyclic graph.*

Remark 3. By Theorem 3, the reachable set exploration of the conic abstraction derived from a radial partition is guaranteed to terminate. Moreover, as indicated in the proof, the direction of the discrete transition between locations can be easily determined by the sign of the Lie derivatives of the partitioning planes at the beginning [33]. □

5 Time-unbounded Reachability Analysis

In this section, we present how to compute the overapproximation of reach pipe of a given affine system based on the conic abstraction.

We first diagonalize the system (as in Sec. 4.2) and we identify the regions that hit the initial region. Then we iteratively explore the adjacent regions, while computing and storing the reach pipe. In particular, we build the control graph of the conic abstraction incrementally and only for those locations that are indeed reachable. We outline our procedure in Algorithm 1.

- The first two lines aim to translate the equilibrium point to the origin and further diagonalize the system. The initial set X_0 undergoes a similar transformation.
- In line 3–6, we split the initial set into multiple regions. For each split, we compute the overapproximation of the reach pipe inside the respective region, as defined in Eq. 1. We store the result in R and we push the region to H for further exploration.
- In line 7–13, we compute the overapproximation of following reach pipes inside the adjacent regions. The result is joined to what previously computed in the same region. The join consists of taking a convex hull between template polyhedra. Each such successor region is pushed to H.

Algorithm 1: Reach pipe overapproximation of affine systems

input : System $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ and initial set X_0;
local : Heap of partition regions H; /*stores unique elements only*/
output: Map from partition region to polyhedron R; /*by default maps to $\emptyset$*/

1 $\boldsymbol{A_\lambda} \leftarrow \boldsymbol{Q}^{-1}\boldsymbol{A}\boldsymbol{Q}$; /*diagonalize*/
2 $Z_0 \leftarrow \boldsymbol{Q}^{-1}(X_0 \oplus \{\boldsymbol{A}^{-1}\boldsymbol{b}\})$; /*transform into linear system and diagonalize*/
3 **foreach** *C partition region in state space such that $Z_0 \cap C \neq \emptyset$* **do**
4 | insert into R(C) the template polyhedron of $[(Z_0 \cap C) \oplus \boldsymbol{A_\lambda} C] \cap C$;
5 | push C into H;
6 **end**
7 **while** *H is not empty* **do**
8 | $C \leftarrow$ pop the top of H;
9 | **foreach** *D successor partition region of C such that $R(C) \cap D \neq \emptyset$* **do**
10 | | join R(D) with the template polyhedron of $[(R(C) \cap D) \oplus \boldsymbol{A_\lambda} D] \cap D$;
11 | | push D into H;
12 | **end**
13 **end**

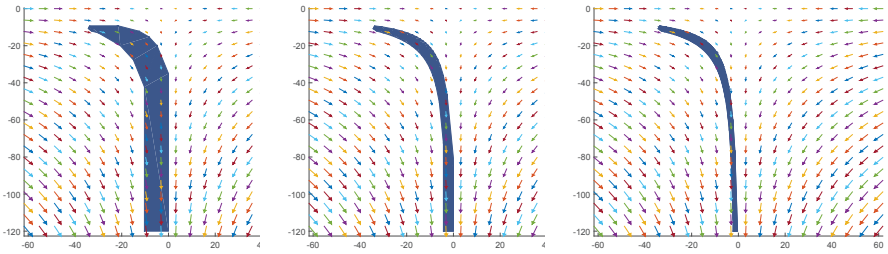
We optimize the exploration order so to explore the successors of a specific region at most once, namely we want the heap H to never pop a region twice at line 8. To this aim, we instruct H to maintain a topological order between regions given by the graph of the conic abstraction (see Def. 4). Such order always exists, as a radial partition always induces an acyclic one (see Thm. 3). Similarly, on the enumeration of line 9, each region D must satisfy the same order w.r.t. C. Concretely, the order between regions is the closure of the order given by the Lie derivative of their common facets (as in Def. 4).

We produce a map from partition regions to template polyhedra, where each template polyhedron overapproximates the reach pipe at the respective region. Precisely, the template polyhedron of a convex set $X \subseteq \mathbb{R}^n$ w.r.t. the finite set of directions $D \subseteq \mathbb{R}^n$, which we call the template, is the tightest polyhedron enclosing X whose facets are normal to all and only the directions in D. We efficiently compute the template polyhedra at lines 4 and 10 using linear programming [34] and the convex hull at line 10 by simply taking for each direction the facet that is the loosest between the two. The choice of template is critical for the quality of the abstraction and the efficiency of the procedure. In each region we use the octagonal template, augmented with the normals of the facets of both the derivative and the state space cones.

In the following, we exemplify the result of the procedure on our running example under different granularities of the partition.

Example 3 (running example). Consider the system in Example 1, let the initial set be $X_0 = \{(x, y) \in \mathbb{R}^2 \mid -30 \leq x \leq -28, -45 \leq y \leq -43\}$ and the invariant be $\mathbb{R}^2$. We diagonalize and transform the system dynamics into $\dot{x} = -4x, \dot{y} = 3y$ with initial state $Z_0 = \{(x, y) \in \mathbb{R}^2 \mid -x + \frac{2}{5}y \leq 30, x - \frac{2}{5}y \leq -28, -x - y \leq 45, x + y \leq -43\}$. By partitioning the orthant into 5, 20 and 60 cones respec-

tively, we got 3 overapproximations of different accuracies for the unbounded reachable set, which is shown in Fig.4. As can be seen, the precision of the over-approximation increases rapidly with the number of cones. □



(a) $K = 5$ **(b)** $K = 20$ **(c)** $K = 60$

Fig. 4: Unbounded invariants obtained for Example 3 under different numbers of slices of partition.

5.1 Mode switching

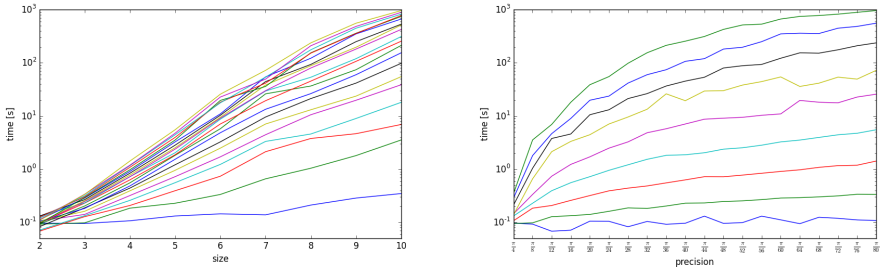
The theory presented in the previous sections can be easily extended to deal with hybrid systems. Given a hybrid system, the conic abstraction of each discrete location can be done as presented. However, due to the transformation of the system dynamics in each location, the same transformation also needs to be applied to the guards and reset operations of the discrete transitions between locations.

Concretely, let $\dot{\boldsymbol{y}} = \boldsymbol{A}_i\boldsymbol{y} + \boldsymbol{b}_i$ and $\dot{\boldsymbol{y}} = \boldsymbol{A}_j\boldsymbol{y} + \boldsymbol{b}_j$ be the dynamics of two discrete locations l_i, l_j of a hybrid system, $G_{ij} = \{\boldsymbol{y} \in \mathbb{R}^n \mid \boldsymbol{J}_{ij}\boldsymbol{y} \leq \boldsymbol{h}_{ij}\}$ be the guard of the transition (l_i, l_j) and $T_{ij} : \boldsymbol{y}' \mapsto \boldsymbol{M}_{ij}\boldsymbol{y} + \boldsymbol{e}_{ij}$ be the reset operation. Suppose the diagonalization of $\boldsymbol{A}_i, \boldsymbol{A}_j$ are $\boldsymbol{A}_{\boldsymbol{\lambda}_i} = \boldsymbol{Q}_i^{-1}\boldsymbol{A}_i\boldsymbol{Q}_i$, $\boldsymbol{A}_{\boldsymbol{\lambda}_j} = \boldsymbol{Q}_j^{-1}\boldsymbol{A}_j\boldsymbol{Q}_j$, respectively. Let $\boldsymbol{x}$ be the variable name after transformation, then we have $l_i : \boldsymbol{y} = \boldsymbol{Q}_i\boldsymbol{x} + \boldsymbol{A}_i^{-1}\boldsymbol{b}_i$ and $l_j : \boldsymbol{y} = \boldsymbol{Q}_j\boldsymbol{x} + \boldsymbol{A}_j^{-1}\boldsymbol{b}_j$. Thus, the guard and the reset operation are transformed into the following.

$$G_{ij}^* = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{J}_{ij}\boldsymbol{Q}_i\boldsymbol{x} \leq \boldsymbol{h}_{ij} - \boldsymbol{J}_{ij}\boldsymbol{A}_i^{-1}\boldsymbol{b}_i\} \tag{4}$$

$$T_{ij}^* : \boldsymbol{x} \mapsto \boldsymbol{Q}_j^{-1}(\boldsymbol{M}_{ij}\boldsymbol{Q}_i\boldsymbol{x} + \boldsymbol{M}_{ij}\boldsymbol{A}_i^{-1}\boldsymbol{b}_i + \boldsymbol{e}_{ij} - \boldsymbol{A}_j^{-1}\boldsymbol{b}_j) \tag{5}$$

Location invariants I_j are transformed as well using the formula $I_j' = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \boldsymbol{Q}^{-1}(\boldsymbol{y} + \boldsymbol{A}^{-1}\boldsymbol{b}), \boldsymbol{y} \in I_j\}$. By applying the above transformations to the whole hybrid system and then performing the conic abstraction, we obtain an LHA, whose reachability analysis can be done as usual. However, unlike for pure continuous systems, termination is not guaranteed.



(a) (b)

Fig. 5: Scalability of our method in computing the abstraction of purely continuous systems. The abscissa of (a) refer to the number of variables and each curve refers to a precision (maximum angle), while the abscissa of (b) refers to precisions and each curve refers to a system size (# variables). Both ordinates show the average runtime for 50 randomly generated systems for each system size and precision.

6 Experiments

We have implemented the procedure presented above in C++ using the GLPK library for linear programming [35], and we have performed two experiments. In the first, we have performed a scalability test using purely continuous systems given by random matrices of increasing size and for increasing precision of the analysis. In the second, we have considered the room heating benchmark and compared against *SpaceEx* under scenarios `supp` and `stc` and *PHAVer* [16,36,20].

We generated random diagonal matrices with non-zero distinct integer values between -10 and 10 on the diagonal. Then we measured the runtime of our procedure for the maximum angles of $\frac{\pi}{2k}$ for increasing k (two by two) and the initial state being a unit box centered in $(50, \ldots, 50)$. Figure 5a shows that the runtime increases exponentially with the number of variables, while the more the precision increases the less (for fixed system size) the difference in runtime is affected. The latter is also confirmed by Fig. 5b, which shows that the runtime increases polynomially with the increase in precision and that the number of variables affects the degree of the polynomial as, in fact, the number of partitions is worst case k^n.

The room heating benchmark describes a protocol for heating a number of rooms with a limited number of shared heaters [37]. We consider houses with 2 to 6 ordered rooms, each room is only adjacent to the previous and the following room, and all but one room have a heater. The temperature of room i is governed by a linear ODE of the form

$$\dot{x}_i = ch + b_i(u - x_i) + \sum_{j \neq i} a_{ij}(x_j - x_i) \tag{6}$$

	Time part.		Conic part.							
	SpaceEx		*PHAVer*				Our method			
	`supp`	`stc`	$\pi/4$	$\pi/20$	$\pi/40$	$\pi/80$	$\pi/4$	$\pi/20$	$\pi/40$	$\pi/80$
`heat-2`	err	oot	0.17	2.20	9.86	50.86	0.24	0.25	0.31	0.41
`heat-3`	err	oot	oot	oot	oot	-	147	2.41	5.18	12.32
`heat-4`	err	oot	oot	-	-	-	14155	278	190	1217
`heat-5`	err	oot	oot	-	-	-	oot	oot	27467	56671
`heat-6`	err	oot	-	-	-	-	oot	oot	oot	oot

Table 1: Runtimes for the abstraction of the room heating benchmark with 2 to 6 rooms. *SpaceEx* has been run with scenarios `supp` and `stc`, template `oct`, and time horizon of 1. *PHAVer* has been run on explicit conic partitions for the given precisions whose generation time is excluded here. We used a 2.6 GHz CPU with 4Gb RAM. The key err indicates error, oot out of time (24 hours), and - experiment not executed, i.e., the explicit partitioning run out of 24 hours time.

where c is the heater efficiency, h indicates whether the heater is present, b_i is the room dispersion, u is external temperature, and a_{ij} is the heat exchange between rooms ($a_{ij} = 0$ for non-adjacent rooms). The switching logic moves a heater from a room to an adjacent room if the temperature difference exceeds a threshold and the latter is colder. In addition, we augmented every mode with a dummy self switch, so to force *SpaceEx* to perform time-unbounded reachability.

We have verified the room heating benchmark using *SpaceEx* with both scenarios `supp` and `stc` and in both cases it either crashes or timeouts. Conversely, using *PHAVer* the procedure terminated, but for small models only. Similarly to our method, *PHAVer* abstracts affine systems into LHA, but it requires the user to provide an explicit partition of the state space (rather than the derivative space). We have generated equivalent conic abstractions in the form of explicit LHA and verified them with *PHAVer*. Note that if such LHA is not provided, *PHAVer* computes trivial abstractions by using the whole mode invariants as partitions. *PHAVer* uses quantifier elimination for forward reachability, while we compute template polyhedra.

The time results are shown in Tab. 1. First, PHAver always times out for systems with more than 2 variables and even for 2-dimensional it scales poorly in precision compared to our method. Second, beyond three dimensional systems our method is even faster than generating the explicit LHA. The scalability in dimensionality indicated the advantage of using template polyhedra rather quantifier elimination while the scalability in precision demonstrates the advantage of using our incremental construction of the conic partition. Figure 6 depicts the abstractions for the 2 and 3 rooms systems and for precisions $\pi/20$, $\pi/80$, and additionally $\pi/400$, computed using our method. Predictably, one can see how the quality of the abstraction increases as the precision increases.

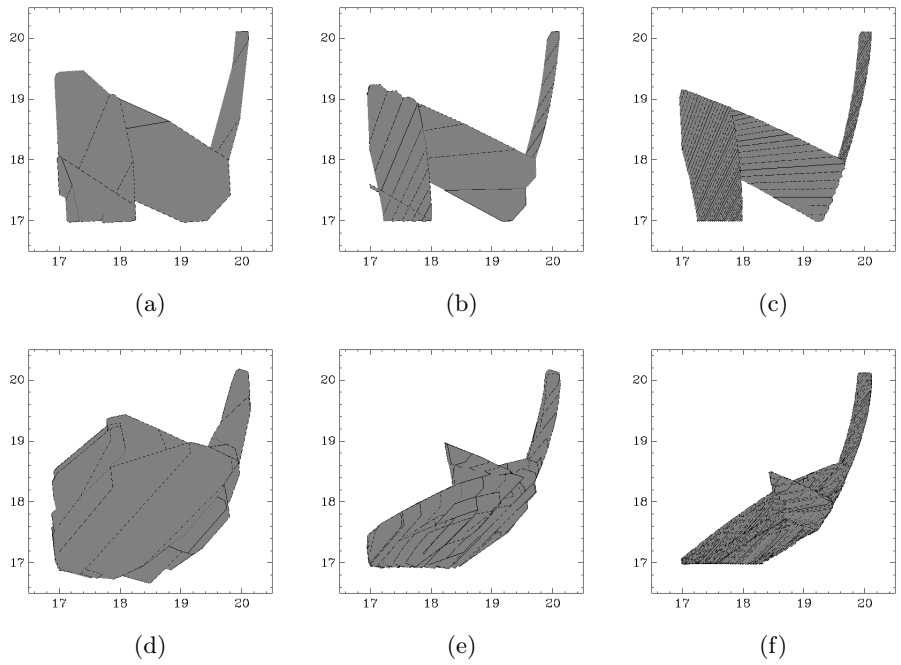


Fig. 6: Conic abstractions of the heating benchmark for 2 rooms (a, b, and c) and 2-dimensional projection for 3 rooms (d, e, and f) for resp. precisions $\pi/20$ (a and d), $\pi/80$ (b and e), and $\pi/400$ (c and f).

7 Conclusion

Deriving a high-quality abstraction for hybrid systems for the purpose of reachability analysis remains a challenging issue to this day. To attack this issue, we propose conic abstractions and a method for constructing them for affine hybrid systems. The core idea lies in deriving a state space partition from a uniform partition of the derivative space. In particular, for diagonalizable affine systems, we develop an algorithm for building conic abstractions as DAGs and for computing invariants on these abstractions. We implement the approach in a tool and experiments on benchmarks show that our approach is more powerful than *SpaceEx* and *PHAVer* for the time-unbounded reachability analysis of diagonalizable systems.

Acknowledgments

This work was partly supported by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award) and by the ARC project DP140104219 (Robust AI Planning for Hybrid Systems).

References

1. T. Henzinger, "The theory of hybrid automata," in *Proc. IEEE Symp. Logic in Computer Science*, 1996, pp. 278–292.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.
3. X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for nonlinear hybrid systems," in *CAV*. Springer, 2013, pp. 258–263.
4. T. Dang and O. Maler, "Reachability analysis via face lifting," *HSCC*, pp. 96–109, 1998.
5. M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *HSCC*. Springer, 2006, pp. 348–362.
6. P. Prabhakar and M. Viswanathan, "A dynamic algorithm for approximate flow computations," in *HSCC*, 2011, pp. 133–142.
7. R. Lal and P. Prabhakar, "Bounded error flowpipe computation of parameterized linear systems," in *2015 International Conference on Embedded Software, EMSOFT 2015, Amsterdam, Netherlands, October 4-9, 2015*, 2015, pp. 237–246.
8. H. Kong, S. Bogomolov, C. Schilling, Y. Jiang, and T. A. Henzinger, "Safety verification of nonlinear hybrid systems based on invariant clusters," in *HSCC*, ser. HSCC '17. New York, NY, USA: ACM, 2017, pp. 163–172.
9. A. Chutinan and B. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," *HSCC*, pp. 76–90, 1999.
10. E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," *HSCC*, pp. 20–31, 2000.
11. A. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis: internal approximation," *Systems & control letters*, vol. 41, no. 3, pp. 201–211, 2000.
12. O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," *HSCC*, pp. 73–88, 2000.
13. O. Stursberg and B. H. Krogh, "Efficient representation and computation of reachable sets for hybrid systems," in *HSCC*. Springer, 2003, pp. 482–497.
14. A. Girard, "Reachability of uncertain linear systems using zonotopes," *HSCC*, pp. 291–305, 2005.
15. A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *HSCC*. Springer, 2006, pp. 257–271.
16. C. Le Guernic and A. Girard, "Reachability analysis of hybrid systems using support functions," in *International Conference on Computer Aided Verification*. Springer, 2009, pp. 540–554.
17. Y. Jiang, H. Song, R. Wang, M. Gu, J. Sun, and L. Sha, "Data-centered runtime verification of wireless medical cyber-physical system," *IEEE Transactions on Industrial Informatics*, 2016.
18. Y. Jiang, H. Zhang, Z. Li, Y. Deng, X. Song, M. Gu, and J. Sun, "Design and optimization of multiclocked embedded systems using formal techniques," *IEEE transactions on industrial electronics*, vol. 62, no. 2, pp. 1270–1278, 2015.
19. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 1997, pp. 460–463.
20. G. Frehse, "Phaver: algorithmic verification of hybrid systems past hytech," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 3, pp. 263–279, 2008.

21. G. Batt, C. Belta, and R. Weiss, "Temporal logic analysis of gene networks under parameter uncertainty," *Transactions on Automatic Control*, vol. 53, no. Special Issue, pp. 215–229, 2008.
22. R. Alur, T. Dang, and F. Ivančić, "Progress on reachability analysis of hybrid systems using predicate abstraction," in *HSCC*. Springer, 2003, pp. 4–19.
23. A. Tiwari and G. Khanna, "Series of abstractions for hybrid automata," in *HSCC*. Springer, 2002, pp. 465–478.
24. A. Tiwari, "Abstractions for hybrid systems," *Formal Methods in System Design*, vol. 32, no. 1, pp. 57–83, 2008.
25. N. Roohi, P. Prabhakar, and M. Viswanathan, "Hybridization based cegar for hybrid automata with affine dynamics," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2016, pp. 752–769.
26. A. Sogokon, K. Ghorbal, P. B. Jackson, and A. Platzer, "A method for invariant generation for polynomial continuous systems," in *TACAS*. Springer, 2016, pp. 268–288.
27. E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Informatica*, vol. 43, no. 7, pp. 451–476, 2007.
28. T. Henzinger and H. Wong-Toi, "Linear phase-portrait approximations for nonlinear hybrid systems," *Hybrid Systems III*, pp. 377–388, 1996.
29. G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2011, pp. 379–395.
30. G. Frehse, "Phaver: Algorithmic verification of hybrid systems past hytech," in *International workshop on hybrid systems: computation and control*. Springer, 2005, pp. 258–273.
31. L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Automatic rectangular refinement of affine hybrid systems," in *Proceedings of the Third International Conference on Formal Modeling and Analysis of Timed Systems*, ser. FORMATS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 144–161.
32. M. W. Hirsch, S. Smale, and R. L. Devaney, *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2012.
33. H. Kong, E. Bartocci, S. Bogomolov, R. Grosu, T. A. Henzinger, Y. Jiang, and C. Schilling, "Discrete abstraction of multiaffine systems," in *International Workshop on Hybrid Systems Biology*. Springer, 2016, pp. 128–144.
34. S. Sankaranarayanan, H. B. Sipma, and Z. Manna, "Scalable analysis of linear systems using mathematical programming," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2005, pp. 25–41.
35. "GLPK (GNU linear programming kit)." [Online]. Available: www.gnu.org/software/glpk
36. G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," in *Proceedings of the 16th international conference on Hybrid systems: computation and control*. ACM, 2013, pp. 203–212.
37. A. Fehnker and F. Ivančić, "Benchmarks for hybrid systems verification," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2004, pp. 326–341.