Abstraction based Model-Checking of Stability of Hybrid Systems

Pavithra Prabhakar and Miriam Garcia Soto

IMDEA Software Institute

Abstract. In this paper, we present a novel abstraction technique and a model-checking algorithm for verifying Lyapunov and asymptotic stability of a class of hybrid systems called *piecewise constant derivatives*. We propose a new abstract data structure, namely, *finite weighted graphs*, and a modification of the predicate abstraction based on the *faces* in the system description. The weights on the edges trace the distance of the executions from the origin, and are computed by using linear programming. Model-checking consists of analyzing the finite weighted graph for the absence of certain kinds of cycles which can be solved by dynamic programming. We show that the abstraction is sound in that a positive result on the analysis of the graph implies that the original system is stable. Finally, we present our experiments with a prototype implementation of the abstraction and verification procedures which demonstrate the feasibility of the approach.

1 Introduction

With the ubiquitous use of embedded control systems, particularly in safety critical application areas such as avionics and automotive, there is an ever increasing need for scalable automated verification of embedded programs. A unique feature of these programs is their interaction with a physical world which is typically continuous, and hence the behavior of the system as a whole consists of mixed discrete-continuous components. The verification of hybrid systems has remained a stubbornly challenging problem due to the complex behavior exhibited by these systems. It is a well known fact that the problem of verifying even simple properties such as safety is undecidable for a class of hybrid systems with relatively simple dynamics [13]. Hence, much effort has been invested in investigating approximate analysis methods, which can be broadly grouped into the following categories.

One direction involves methods for *approximate post analysis*. Computing the post of a set of states is a crucial step in state-space exploration based model-checking algorithms. The exact post computation is feasible for only a small subclass of hybrid systems; and hence the focus has been on computing approximations of the post sets. Various efficient methods and representations for the post set have been proposed including zonotopes, polytopes, ellipsoids and support functions [10, 11, 24]. Bounded error approximations of the entire hybrid

systems have also been explored, and include methods such as hybridization and polynomial approximations [27, 2, 21, 5].

The other set of results concern *abstraction based analysis*, where a simplified model of the system is computed and analysed to infer correctness properties about the original model. Various methods for computing abstract models include predicate abstraction, hybrid abstractions [8, 22, 1, 6]. These methods in general do not provide any bound on the error of the abstraction; hence, they are usually accompanied by an abstraction-refinement loop based on, for example, counter-example analysis. Most of the above techniques can be classified as model-checking algorithms and mainly focus on safety verification. There have also been deductive verification techniques for safety verification [26, 19].

In this paper, we focus on a different class of properties, namely, stability. Stability properties capture the notion that small perturbations to the initial state or input to the system result in only small changes to the future behaviors of the system. Stability is an important property expected out of every control system so much so that a system which is not stable is deemed useless. The study of stability analysis of purely continuous systems is a mature field and there exist characterizations and automatically verifiable methods for their analysis [15]. Most of these methods can be classified under deductive verification, where the proof of stability is established by exhibiting a Lyapunov function [18, 17], which represents an "energy" function which decreases over the trajectories of the system. There also exist extensions of Lyapunov function based analysis methods for hybrid systems [16, 7, 4, 12], which either exhibit a common Lyapunov function for the entire hybrid system or combine the analysis of a Lyapunov function for each mode of the hybrid system. However, constructing Lyapunov functions in an extremely arduous task; and often requires the ingenuity of the designer. Hence, the state-of-the-art technique for proving stability of hybrid systems relies on automating the search for Lyapunov functions. Complete results in this direction only exist for certain subclasses of systems such as purely continuous linear dynamical systems, where it is known that a quadratic function always exists and can be found by solving linear matrix inequalities.

In contrast to the traditional approach to stability analysis, the focus of this paper is in exploring model-checking as an alternate method for stability analysis. The standard methods of abstraction such as predicate abstraction [1, 6] do not suffice for this purpose, since as pointed out in [23], the notion of simulation does not suffice to preserve stability and instead needs a stronger notion of simulation strengthened with continuity requirements.

The main contribution of this paper is an abstraction procedure and a model-checking algorithm for verifying stability of a class of hybrid systems called *piecewise constant derivatives* studied in [3]. Our abstraction procedure can be interpreted as a modified predicate abstraction procedure. The abstract model is a finite weighted graph whose nodes correspond to the faces of the regions defined by the predicates. The edges correspond to the existence of an execution from one face to the other, similar to the predicate abstraction; however, in addition, it is also annotated with a weight which measures the closeness of the

execution to the origin between its end-points. The finite weighted graph is then analysed for the absence of cycles in which the product of the weights on the edges exceeds 1, and the absence of such cycles implies stability of the system. We consider two classical notions of stability, namely, Lyapunov and asymptotic stability. Our analysis procedure is conservative in general, however, for the two dimensional case, it coincides with the decidability algorithm in [25]. The paper [25] explores the decidability boundary for Lyapunov and asymptotic stability and shows the decidability in 2 dimensions for a class of hybrid systems slightly more general than *PCD*s and undecidability for *PCD*s with 5 dimensions.

There has been some work on predicate abstraction based analysis for certain weaker notions of stability, such as, region stability, which require that all trajectories of the system reach a region in a finite time [20, 9].

Though we focus on a simple class of systems, we believe that the technique of face based abstraction into finite weighted graph and its analysis can be carried over to more complex classes of hybrid systems. Below we summarize the main features and contributions of the paper.

1. A face based abstraction procedure to reduce the verification of Lyapunov and asymptotic stability in a sound manner to the analysis on a finite weighted graph.
2. A model-checking algorithm for analyzing the abstract finite weighted graph.
3. The model-checking algorithm returns an "abstract counter-example" which is crucial in developing an abstraction refinement framework.
4. A prototype implementation of the abstraction and verification procedure, which demonstrates the feasibility of the technique.

In contrast to the previous works on deductive verification based analysis for Lyapunov and asymptotic stability, our method has the flavor of an algorithmic verification technique based on state-space exploration. Additional details can be found at the following url:
http://software.imdea.org/people/pavithra.prabhakar/Details/home.html

2 Overview of the abstraction and model-checking algorithm

In this section, we will illustrate with examples the main ideas behind the abstraction based model-checking algorithm.

A piecewise constant derivative system (PCD) [3] is a hybrid system which partitions the state-space into some finite number of polyhedral regions and associates a flow vector with each region specifying the direction in which the state evolves while in the region. For example, shown in Figure 2 is a two-dimensional PCD which has four regions corresponding to the four quadrants, r_1 with flow $(-1, 1)$, r_2 with flow $(-1, -1)$, r_3 with flow $(1, -1)$ and r_4 with flow $(1, 1)$. A sample run of the PCD is the trajectory σ which starts as $(1, 0)$ and moves along a straight line to $(0, 1)$, then to $(-1, 0)$, $(0, -1)$ and back to $(1, 0)$.

This is a finite execution of the system. Repeating σ an infinite number of times gives a complete execution, an execution in which the time diverges to infinity.

Intuitively, Lyapunov stability captures the notion that every execution starting close to the origin remains close to the origin. More precisely, it says that given any $\epsilon > 0$, we can find a $\delta > 0$, such that all trajectories starting within distance δ from the origin remain with in distance ϵ of the origin at all times. Note that starting at any point in any δ-unit square around the origin, the executions of the system remain with in a $\sqrt{2}\delta$ unit square around the origin. Hence, the system is stable. On the other hand, if the flow associated with r_4 were $(2, 1)$ instead of $(1, 1)$, then for every δ, there exist trajectories which start within distance δ but diverge from the origin.

The other canonical notion of stability is *asymptotic stability* which requires along with Lyapunov stability that all complete executions converge to the origin. The 2 dimensional system in Figure 2 is not asymptotically stable because the complete execution corresponding to repeating the execution σ, which moves along the rhombus, infinitely many times does not converge to the origin.

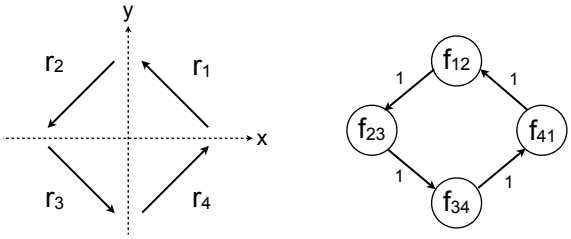

Fig. 1. 2 dimensional system and its abstraction

The main idea behind the analysis is to construct a finite weighted graph which contains information about the distance from origin of the executions of the system. More precisely, the nodes of the graph correspond to the faces of the regions. And there is an edge between two faces if there exists an execution which starts at some point on the first face and reaches some point on the other face while remaining in the common region. This part is similar in some sense to predicate abstraction except that we consider the faces instead of the regions. The additional element is the weights on the edges of the graph. The weight corresponds to the maximum scaling in the distance to the origin between any point in the first face and the points reachable from it on the second face. Abstract graph for the 2 dimensional example in shown on the right in Figure 2. It consists of four faces f_{12}, f_{23}, f_{34} and f_{41}, where f_{ij} corresponds to the common set of points between region i and region j. The weight on all the edges is 1. For example, starting at distance δ on f_{41}, that is, $(\delta, 0)$, the unique point

on f_{12} reachable is $(0, \delta)$. Hence the ratio of the distance of $(0, \delta)$ from the origin to the distance of $(\delta, 0)$ from the origin is $\delta/\delta = 1$.

The abstract graph has the property that corresponding to every execution of the original system, there is a path in the graph where each edge corresponds to the part of the execution which remains within a particular region. More over, the weight on the edge is an upper bound on the weight of the scaling associated with executions corresponding to the edge. The main theorem of the paper states that if the abstract graph does not contain a cycle such that the product of the weights is > 1 and the flow associated with all the regions is such that there exist no trajectories which diverge while remaining within the region, then the system is Lyapunov stable. Further, it is asymptotically stable if there are no cycles such that the product of the weights is 1. The theorem essentially states that the abstraction is sound with respect to stability analysis. It turns out that the theorem is in fact a necessary and sufficient condition for the 2 dimensional case as shown in [25]. However, the same is not true in higher dimensions as shown below.

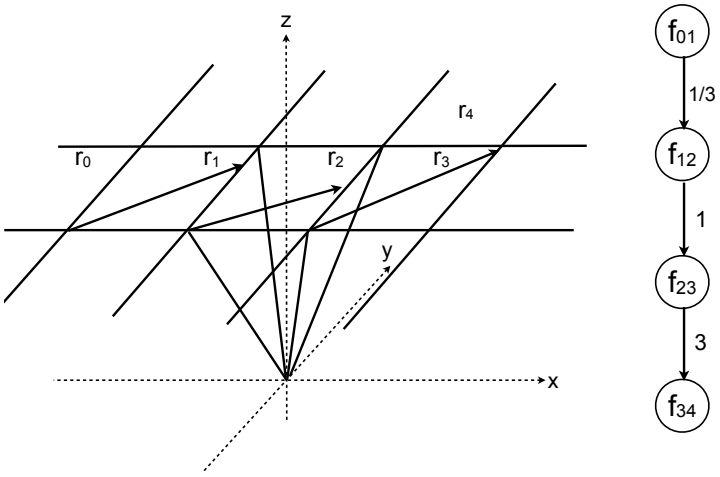

Fig. 2. 3 dimensional system and its abstraction

In the Figure 2, r_2 is the region given by $-z \leq x \leq z, -z \leq y \leq z, z \geq 0$. Similarly, r_0, r_1, r_3 are obtained by replacing the constraint on x by $x \leq -3z$, $-3z \leq x \leq z$ and $z \leq x \leq 3z$, respectively. r_4 is given by $z \leq x \leq 3z, z \leq y, z \geq 0$. The flow vectors of r_1, r_2 and r_3 are given by $(2, 1.5, 0), (2, 1, 0)$ and $(2, 2, 0)$, respectively. A part of the abstract graph corresponding to the faces f_{01}, f_{12}, f_{23} and f_{34} are shown in Figure 2. We use the infinity norm for the distance, that is, distance between two points (x_1, y_1, z_1) and (x_2, y_2, z_3) is given

by $max\{|x_1 - x_2|, |y_1 - y_2|, |z_1 - z_3|\}$. The ratios on the edges are computed as before. For example consider the edge from f_{01} to f_{12}. Since the flow vectors are parallel to the xy-plane, it suffices to consider any particular value of z to compute the ratios. Let us fix $z = 1$. All the points on f_{01} are at distance 3. The points reached on f_{12} from points on f_{01} are at distance at most 1. In fact, the point $(-1, 0.75, 1)$ at distance 1 from the origin is reachable from the point $(-3, 0.25, 1)$ on f_{01}, hence the ratio is $1/3$.

Note that there does not exist an execution corresponding to the path $f_{01}f_{12}f_{23}$, that is, a point on f_{01} such that there exists an execution from it which reaches f_{12} and continues to reach f_{23}. Though there exists an execution corresponding to the path $f_{12}f_{23}f_{34}$, there does not exist one in which the part of the execution corresponding to $f_{12}f_{23}$ has a scaling of 1 and the part corresponding to $f_{23}f_{34}$ has value 3. Hence, in general, the abstract graph is conservative.

3 Preliminaries

Notations. Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, $\mathbb{Q}$ and $\mathbb{N}$ denote the set of reals, non-negative reals, rationals and natural numbers, respectively. We use $[n]$ to denote the set of natural numbers $\{1, \cdots, n\}$. Given a function F, we use $Dom(F)$ to denote the domain of F. Given a function $F : A \to B$ and a set $A' \subseteq A$, $F(A')$ will denote the set $\{F(a) \mid a \in A'\}$. Given a finite set A, $|A|$, denotes the number of elements of A.

Euclidean space. Let X denote the n-dimensional Euclidean space $\mathbb{R}^n$, for some n. Given $\mathbf{x} = (x_1, \cdots, x_n) \in X$ and $1 \leq i \leq n$, we use $\langle \mathbf{x} \rangle_i$ to denote the i-th component of x, namely, x_i. Given $\mathbf{x}, \mathbf{y} \in X$, we use the standard notation $\mathbf{x} \cdot \mathbf{y}$ to denote the dot product of the vectors $\mathbf{x}$ and $\mathbf{y}$ and $|\mathbf{x}|$ to denote the norm of $\mathbf{x}$. Most of our results are independent of the particular norm used, however, in some parts we will focus on the infinity norm, which is defined as $|\mathbf{x}| = max_{1 \leq i \leq n} \langle \mathbf{x} \rangle_i$. We use $B_\epsilon(\mathbf{x})$ to denote an open ball of radius ϵ around $\mathbf{x}$, that is, the set $\{\mathbf{y} \mid |\mathbf{x} - \mathbf{y}| < \epsilon\}$.

Sequences. Let *SeqDom* denote the set of all subsets of $\mathbb{N}$ which are prefix closed, where a set $S \subseteq \mathbb{N}$ is prefix closed if for every $m, n \in \mathbb{N}$ such that $n \in S$ and $m < n$, $m \in S$. A *sequence* over a set A is a mapping from an element of *SeqDom* to A. Given a sequence $\pi : S \to A$, we also denote the sequence by enumerating its elements in the order, that is, $\pi(0), \pi(1), \cdots$.

Linear Constraints and Convex Polyhedral Sets. A *linear expression* is an expression of the form $\mathbf{a} \cdot \mathbf{x} + \mathbf{b}$, where $\mathbf{x}$ is a vector of n variables and $\mathbf{a}, \mathbf{b} \in X$. A *linear constraint* or *predicate* is a term $e \sim 0$, where e is a linear expression and $\sim$ is a relational operator in $\{<, \leq, =\}$. A linear constraint $\mathbf{a} \cdot \mathbf{x} + \mathbf{b}$ is called *homogenous* if $\mathbf{b} = \mathbf{0}$. Given a linear constraint C given by $\mathbf{a} \cdot \mathbf{x} + b \sim \mathbf{0}$, we use $[\![C]\!]$ to denote the set of all values $\mathbf{v} \in \mathbb{R}^n$ such that $\mathbf{a} \cdot \mathbf{v} + \mathbf{b} \sim \mathbf{0}$. Given a set of linear constraints $\mathcal{C}$, we use $[\![\mathcal{C}]\!]$ to denote the convex set defined by $\mathcal{C}$, namely, $\bigcap_{C \in \mathcal{C}} [\![C]\!]$.

A *half-space* in X is a set which can be expressed as the set of all points $\mathbf{x} \in X$ satisfying a linear constraint, $\mathbf{a}\cdot\mathbf{x}+\mathbf{b} \sim \mathbf{0}$, where $\sim \in \{<, \leq\}$. A *convex polyhedral set* is an intersection of finitely many half-spaces. We will use *ConvPolyhed*(X) to denote the set of all convex polyhedral subsets of X.

Given a convex polyhedral set P defined by the constraints $\{e_1 \sim_1 \mathbf{0}, \cdots, e_k \sim_k \mathbf{0}\}$, a *face* of P is a non-empty polyhedral set defined by constraints of the form $\{e_1 \sim'_1 \mathbf{0}, \cdots, e_k \sim'_k \mathbf{0}\}$, where each $\sim'_i$ is either $\sim_i$ or $=$, and at least one of the $\sim'_i$ is $=$. We denote the set of faces of a convex polyhedral set P by *Faces*(P).

Given a convex polyhedral set P defined by the constraints $\{e_1 \sim_1 \mathbf{0}, \cdots, e_k \sim_k \mathbf{0}\}$, the closure of P, denoted *Closure*(P), is the set defined by the constraints $\{e_1 \leq \mathbf{0}, \cdots, e_k \leq \mathbf{0}\}$. Essentially, every $e_i < 0$ is replaced by $e_i \leq 0$. Note that *Closure*(P) contains all the limit points of P.

Given a convex polyhedral set S and a point $\mathbf{s} \in S$, we call the *cone* of S at $\mathbf{s}$, the set of all vectors $\mathbf{v} \neq \mathbf{0}$ such that there exists a $t > 0$, for which $\mathbf{s} + \mathbf{v}t \in S$. We denote this set by *Cone*$(S, \mathbf{s})$.

A *partition* $\mathcal{P}$ of $\mathbb{R}^n$ into convex polyhedral sets is a finite set of convex polyhedral sets $\{P_1, \cdots, P_k\}$ such that $\cup_{i=1}^{k} P_i = \mathbb{R}^n$ and for each $i \neq j$, $P_i \cap P_j = \emptyset$. We will call the P_is as regions.

Graphs. A *graph* G is a pair (V, E), where V is a finite set of nodes and $E \subseteq V \times V$ is a finite set of edges. A *path* π of a graph $G = (V, E)$ is a finite sequence of nodes $v_0, \cdots, v_k$ such that $(v_i, v_{i+1}) \in E$ for $0 \leq i < k$. The length of a path π, denoted $|\pi|$, is the number of edges occurring in it. A path π is *simple* if all the nodes occurring in the path are distinct. A *cycle* in a graph $G = (V, E)$ is a path $\pi = v_0, \cdots, v_k$ such that the first and the last nodes are the same, that is, $v_0 = v_k$. A cycle is *simple* if all the nodes except the last one are distinct.

A node v is *reachable* from a node u if there exists a path whose first element is u and the last element is v, that is, there exists a π such that $\pi(0) = u$ and $\pi(|Dom(\pi)| - 1) = v$.

We associate weights with the edges of a graph using weighting functions. A *weighted graph* is a triple $G = (V, E, W)$, where (V, E) is a graph and $W : E \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is a weighting function. We extend the weight function to a path as follows. Weight of a path π of G, denoted $W(\pi)$, is $\Pi_{0 \leq i < |Dom(\pi)|-1} w((\pi(i), \pi(i+1)))$.

4 Piecewise Constant Derivative System (*PCD*)

In this section, we introduce a formal model for a subclass of hybrid systems called piecewise constant derivative systems [3]. Hybrid automata [14] are a popular formal model for systems with mixed discrete-continuous behaviors. A hybrid automaton combines the finite automaton model for discrete systems and the differential equation formalism for continuous dynamics. Here we consider a class of systems in which the statespace is partitioned into a finite number of regions with each region corresponding to a discrete mode; and the continuous

dynamics is provided by a differential equation of the form $\dot{\mathbf{x}} = \mathbf{a}$, where $\mathbf{x}$ is a vector of variables and $\mathbf{a}$ is a constant vector. Hence, in each region the variables evolve at a constant rate given by the vector associated with the region and is allowed to switch to a neighboring region at the common boundary. Next, we define the system formally.

Definition 1. *An n-dimensional PCD $\mathcal{H} = (\mathcal{P}, \varphi)$, where $\mathcal{P}$ is a partition of $\mathbb{R}^n$ into convex polyhedral sets and $\varphi : \mathcal{P} \to \mathbb{R}^n$ is the flow function.*

The semantics of a *PCD* $\mathcal{H}$ is given by the set of executions exhibited by the system. An execution of a *PCD* starting from a point $\mathbf{x} \in \mathbb{R}^n$ follows the flow associated with the region in which $\mathbf{x}$ lies until it reaches the boundary of a neighboring region upon which it switches to the flow of the new region.

We need some definitions. An *execution interval* is either $[0, T]$ representing the set $\{t \in \mathbb{R} \mid 0 \leq t \leq T\}$, $[0, T)$ representing the set $\{t \in \mathbb{R} \mid 0 \leq t < T\}$ or $[0, \infty)$ representing the set $\{t \in \mathbb{R} \mid 0 \leq t\}$. We denote the set of execution intervals by *IntExec*. Given an execution interval $I \in$ *IntExec*, we define its *size*, denoted *Size*(I), to be T if $I = [0, T]$ or $[0, T)$ and ∞ otherwise.

Definition 2. *An* execution σ *of* $\mathcal{H} = (\mathcal{P}, \varphi)$ *is a function* $\sigma : I \to \mathbb{R}^n$, *where* $I \in$ *IntExec, such that there exists a finite or infinite sequence* $\eta = (P_1, \delta_1)(P_2, \delta_2) \cdots$ *satisfying for every* $1 \leq i \leq$ *Dom*(η)*:*

- $P_i \in \mathcal{P}$ *and if* $i <$ *Dom*(η), $P_i \neq P_{i+1}$,
- δ_i *in* $\mathbb{R}_{\geq 0}/0$ *for* $i <$ *Dom*(η) *and if Dom*(η) *is finite, then* $\delta_{|Dom(\eta)|}$ *in* $(\mathbb{R}_{\geq 0}/0) \cup \{\infty\}$.
- $I = [0, t_{Dom(\eta)}]$ *if Dom*(η) *is finite and* $I = [0, t_{Dom(\eta)})$ *otherwise, where* $t_0 = 0$ *and for* $1 \leq j \leq$ *Dom*(η), $\sum_{l=1}^{j} \delta_l = t_j$, *and*
- *for every* $t \in I$, $\sigma(t) = \sigma(t_i) + \varphi(P_i)(t - t_i)$, *where* i *is the smallest integer such that* $t_i \leq t \leq t_{i+1}$.

We denote the set of all executions of $\mathcal{H}$ by *Exec*$(\mathcal{H})$. We call an execution σ with domain $[0, T]$ for some $T \in \mathbb{R}_{\geq 0}$ a *finite* execution and one with domain $[0, \infty)$ a *complete* execution.

Definition 3. *Given a finite execution σ, we define the* scaling *of σ, denoted Scaling*(σ)*, to be* $|\sigma(Size(Dom(\sigma)))|/|\sigma(0)|$.

Representation of a PCD. We represent the *PCD* by specifying a common set of linear predicates such that each of the regions is the conjunction of the predicates in either the positive or the negative form. More precisely, an n-dimensional *PCD* is represented by $\mathcal{H} = (\mathcal{C}, F)$, where $\mathcal{C}$ is a set of linear predicates and $F : 2^{\mathcal{C}} \to \mathbb{R}^n$ is the flow function. The regions of $\mathcal{H}$, denoted *Regions*$(\mathcal{C})$ or *Regions*$(\mathcal{H})$, are the non-empty sets $R \subseteq \mathbb{R}^n$ such that $R = \cap_{C \in A} [\![C]\!] \bigcap \cap_{C \notin A} \mathbb{R}^n \backslash [\![C]\!]$ for some $A \subseteq \mathcal{C}$. We call R the region *generated by* A. $\mathcal{H}$ consists of at most $2^{|\mathcal{C}|}$ polyhedral sets. The flow associated with a region R generated by $A \subseteq \mathcal{C}$ is $F(A)$. From now on, we use the both the notations for representing a *PCD*, and the particular notation will be clear from the context.

Notation. We will assume that the constants in the linear predicates and flows are all rational numbers.

5 Stability: Lyapunov and Asymptotic

In this section, we define two classical notions of stability in control theory, and state some general results about stability of *PCD*. We consider stability of the system with respect to an equilibrium point, which in our setting will be the origin.

Definition 4. $\mathbf{0}$ *is an* equilibrium point *of a PCD* $\mathcal{H}$ *if every execution of* $\mathcal{H}$ *starting at* $\mathbf{0}$ *remains at* $\mathbf{0}$*, that is, every execution* $\sigma \in Exec(\mathcal{H})$ *with* $\sigma(0) = \mathbf{0}$ *satisfies* $\sigma(t) = \mathbf{0}$ *for every* $t \in Dom(\sigma)$*.*

Intuitively, Lyapunov stability captures the notion that an execution starting close to the equilibrium point remains close to it, and asymptotic stability, in addition, requires that executions starting in a small neighborhood around the equilibrium point converge to it.

Definition 5. *A PCD* $\mathcal{H}$ *is said to be* Lyapunov stable*, if for every* $\epsilon > 0$*, there exists a* $\delta > 0$ *such that for every execution* $\sigma \in Exec(\mathcal{H})$ *with* $\sigma(0) \in B_\delta(\mathbf{0})$*,* $\sigma(t) \in B_\epsilon(\mathbf{0})$ *for every* $t \in Dom(\sigma)$*.*

We use $Lyap(\mathcal{H}, \epsilon, \delta)$ to denote the fact that for every execution $\sigma \in Exec(\mathcal{H})$ with $\sigma(0) \in B_\delta(\mathbf{0})$, $\sigma(t) \in B_\epsilon(\mathbf{0})$ for every $t \in Dom(\sigma)$. In fact, we do not need to consider all possible values for ϵ in the definition of Lyapunov stability but only values in a small neighborhood around $\mathbf{0}$.

Proposition 1 *A PCD* $\mathcal{H}$ *is Lyapunov stable if and only if there exists an* $\epsilon' > 0$ *such that for every* $0 < \epsilon < \epsilon'$*, there exists a* $\delta > 0$ *for which* $Lyap(\mathcal{H}, \epsilon, \delta)$ *holds.*

Definition 6. *A PCD* $\mathcal{H}$ *is said to be* asymptotically stable*, if it is Lyapunov stable and there exists a* $\delta > 0$ *such that every complete execution* $\sigma \in Exec(\mathcal{H})$ *with* $\sigma(0) \in B_\delta(\mathbf{0})$ *converges to* $\mathbf{0}$*, that is, for every* $\epsilon > 0$*, there exists a* $T \in Dom(\sigma)$*, such that* $\sigma(t) \in B_\epsilon(\mathbf{0})$ *for every* $t \geq T$*.*

We use $Asymp(\mathcal{H}, \delta)$ to denote the fact that every complete execution $\sigma \in Exec(\mathcal{H})$ with $\sigma(0) \in B_\delta(\mathbf{0})$ converges to $\mathbf{0}$.

6 Abstraction based Model-Checking

In this section, we describe the abstraction based model-checking procedure for Lyapunov and asymptotic stability. The main steps in the procedure are as follows:

1. *Preprocessing*: We convert a *PCD* to a normal form in which all the constraints are homogenous.
2. *Abstraction*: We construct an abstract finite weighted graph using face based abstraction which is a sound abstraction for analyzing Lyapunov and asymptotic stability. The construction of the graph involves solving satisfiability of a set of linear constraints and linear programming problems.

3. *Model-Checking*: This involves analyzing the abstract graph for the absence of cycles with weight greater than 1 (or greater than or equal to 1) and analyzing the regions of the *PCD* for "non-explosion".

Next, we explain each of the steps in detail.

6.1 Normal Form for *PCD*

In this section, we present a reduction of a general *PCD* to one in a normal form. We say that a *PCD* is in normal form if each of the predicates defining the regions is homogeneous.

Definition 7. *A PCD $\mathcal{H} = (\mathcal{C}, F)$ is said to be in* normal form *if each of the predicates in $\mathcal{C}$ is homogenous.*

Next, we define a transformation of a *PCD* $\mathcal{H}$ to a normal form *NF*($\mathcal{H}$) such that $\mathcal{H}$ and *NF*($\mathcal{H}$) are equivalent with respect to Lyapunov and asymptotic stability.

Definition 8. *Given $\mathcal{H} = (\mathcal{C}, F)$, NF($\mathcal{H}$) $= (\mathcal{C}', F')$, where $\mathcal{C}' \subseteq \mathcal{C}$ is the set of homogenous predicates in $\mathcal{C}$, and $F'(A) = F(A')$, where $A' = A \cup \{C \,|\, C \in \mathcal{C}\backslash\mathcal{C}', \mathbf{0} \in [\![C]\!]\}$.*

NF($\mathcal{H}$) is in normal form by definition. Note that all the region of *NF*($\mathcal{H}$) contain $\mathbf{0}$ on their boundary.

Proposition 1. *Let $\mathcal{H} = (\mathcal{C}, F)$ be a PCD. Then NF($\mathcal{H}$) $= (\mathcal{C}', F')$ is a normal PCD such that:*

1. *$\mathcal{H}$ is Lyapunov stable if and only if $\hat{\mathcal{H}}$ is Lyapunov stable.*
2. *$\mathcal{H}$ is asymptotically stable if and only if $\hat{\mathcal{H}}$ is asymptotically stable.*

6.2 Abstraction: Construction of the Weighted Graph

In this section, we present the abstraction of a *PCD* to a finite weighted graph. The abstraction consists of the faces of the regions of the *PCD* as the vertices, and an edge corresponds to the fact that there exists an execution from one face to the other. In addition, we also track how much "closer" the execution moves towards the origin between the starting and ending points of executions from one face to the other. This "scaling" in the distance to the origin appears as a weight on the corresponding edge. The finite graph is then be analysed for deducing the stability of the original *PCD*.

First, we need the definition of a region execution.

Definition 9. *Given a region R of $\mathcal{H}$, an* R-execution *of $\mathcal{H}$ is an execution σ of $\mathcal{H}$ such that $\sigma(t) \in R$ for every $t \in (0, Size(Dom(\sigma)))$. An R-execution σ of $\mathcal{H}$ is said to be* extreme *if it is finite and $\sigma(0) \in f_1$ and $\sigma(Size(Dom(\sigma))) \in f_2$ for some $f_1, f_2 \in Faces(R)$. A* region execution *of $\mathcal{H}$ is an* R-execution *of $\mathcal{H}$ for some region R of $\mathcal{H}$.*

Remark 1. Note that every region execution is either a finite execution or a complete execution.

Abstract Graph $G(\mathcal{H})$ and its correctness Let $\mathcal{H} = (\mathcal{C}, F)$ be an n-dimensional *PCD*. The weighted graph $G(\mathcal{H}) = (V, E, W)$ is defined as follows.

1. The set of vertices V is the set of faces of the regions of $\mathcal{H}$, that is, $f \in \mathit{Faces}(R)$, where $R \in \mathit{Regions}(\mathcal{C})$.
2. The set of edges E consists of pairs (f_1, f_2) such that $f_1, f_2 \in \mathit{Faces}(R)$ for some $R \in \mathit{Regions}(\mathcal{C})$ and there exists a finite R-execution $\sigma \in \mathit{Exec}(\mathcal{H})$ with $\sigma(0) \in f_1$, $\sigma(\mathit{Size}(\mathit{Dom}(\sigma))) \in f_2$.
3. The weight associated with the edge $e = (f_1, f_2) \in E$ is given by $\sup\{|\sigma(\mathit{Size}(\mathit{Dom}(\sigma)))|/|\sigma(0)| : \sigma$ is a finite region execution of $\mathcal{H}, \sigma(0) \in f_1, \sigma(0) \neq \mathbf{0}, \sigma(\mathit{Size}(\ \mathit{Dom}(\sigma))) \in f_2\}$.

Note that the weight on all the edges is greater than or equal to 0. The next proposition states that the weight on an edge is an upper bound on the scaling of all the executions corresponding to it.

Proposition 2. *A finite R-execution σ of $\mathcal{H}$ with $\sigma(0) \in f_1$ and $\sigma(\mathit{Size}(\mathit{Dom}(\sigma)))$ $\in f_2$ for some $f_1, f_2 \in \mathit{Faces}(R)$ is such that $\mathit{Scaling}(\sigma) \leq W((f_1, f_2))$.*

The next theorem states a sufficient condition on the graph $G(\mathcal{H})$ for the *PCD* $\mathcal{H}$ to be Lyapunov stable. We need the definition of an exploding region.

Definition 10. *An execution σ of $\mathcal{H}$ is said to be* diverging *if for every $\epsilon > 0$, there exists a $t \in \mathit{Dom}(\sigma)$ such that $\sigma(t) \notin B_\epsilon(\mathbf{0})$.*

Definition 11. *A region R of a PCD $\mathcal{H}$ is said to be* exploding *if there exists an R-execution σ such that σ diverges.*

Proposition 3. *A region R of a PCD $\mathcal{H}$ is exploding if and only if $\varphi(R) \in \mathit{Cone}(R, \mathbf{0})$.*

Theorem 1. *A PCD $\mathcal{H}$ is Lyapunov stable if the following hold:*

1. *Every simple cycle π of $G(\mathcal{H})$ satisfies $W(\pi) \leq 1$.*
2. *No region of $\mathcal{H}$ is exploding.*

The above theorem says that the abstraction using the finite weighted graph is sound. In fact, for the two dimensional case the theorem provides a necessary and sufficient condition (see [25]). However, the same is not true for higher dimensions.

Theorem 2. *A PCD $\mathcal{H}$ is asymptotically stable if the following hold:*

1. *Every simple cycle π of $G(\mathcal{H})$ satisfies $W(\pi) < 1$.*
2. *No region of $\mathcal{H}$ is exploding.*

Computing the Graph Next, we explain the algorithm for computing the edges and the weights associated with them.

Given a region $R \in \mathit{Regions}(\mathcal{H})$ and faces $f_1, f_2 \in \mathit{Faces}(R)$, let $W_{\mathit{Reg}}(f_1, f_2, R)$ denote $\sup\{|\sigma(\mathit{Size}(\mathit{Dom}(\sigma)))|/|\sigma(0)| \mid \sigma$ is a finite R-execution of $\mathcal{H}, \sigma(0) \in f_1, \sigma(0) \neq \mathbf{0}, \sigma(\mathit{Size}(\mathit{Dom}(\sigma))) \in f_2\}$. Note that $W((f_1, f_2)) = \mathit{max}\{ W_{\mathit{Reg}}(f_1, f_2, R) \mid R \in \mathit{Regions}(\mathcal{H}), f_1, f_2 \in \mathit{Faces}(R)\}$. Next we show how to compute $W_{\mathit{Reg}}(f_1, f_2, R)$ by using linear programming.

Let us fix a region R and faces $f_1, f_2 \in \mathit{Faces}(R)$. First, observe that $W_{\mathit{Reg}}(f_1, f_2, R)$ is equal to

$$\sup\{|\mathbf{v}_2|/|\mathbf{v}_1| \mid \exists t \geq 0, \mathbf{v}_1 \in f_1, \mathbf{v}_1 \neq \mathbf{0}, \mathbf{v}_2 \in f_2, \mathbf{v}_2 = \mathbf{v}_1 + \varphi(R)t\} \tag{1}$$

Let $\psi(f_1, f_2, R) = \exists t \geq 0, \mathbf{v}_1 \in f_1, \mathbf{v}_1 \neq \mathbf{0}, \mathbf{v}_2 \in f_2, \mathbf{v}_2 = \mathbf{v}_1 + \varphi(R)t$. If $\psi(f_1, f_2, R)$ is feasible, then Equation 1 is equivalent to the following:

$$\sup |\mathbf{v}_2|/|\mathbf{v}_1| s.t. \tag{2}$$
$$t \geq 0, \mathbf{v}_1 \in \mathit{Closure}(f_1), \mathbf{v}_1 \neq \mathbf{0}, \mathbf{v}_2 \in \mathit{Closure}(f_2), \mathbf{v}_2 = \mathbf{v}_1 + \varphi(R)t$$

Further, we observe that it suffices to consider $\mathbf{v}_1$ such that $|\mathbf{v}_1| = 1$, due to the fact that the constraints corresponding to the faces and the regions are homogenous linear constraints. Hence Equation 2 is equivalent to:

$$\sup |\mathbf{v}_2| s.t. \tag{3}$$
$$t \geq 0, \mathbf{v}_1 \in \mathit{Closure}(f_1), \mathbf{v}_2 \in \mathit{Closure}(f_2), \mathbf{v}_2 = \mathbf{v}_1 + \varphi(R)t, |\mathbf{v}_1| = 1$$

Note that Equation 3 is not a linear program in general. However, observe that the results in the paper do not depend on the norm. Hence, we can choose the infinity norm for our analysis, which is defined as, $|\mathbf{x}| = \mathit{max}_i \langle \mathbf{x} \rangle_i$. Then we can computing Equation 3 by solving $O(n^2)$ linear programs, where n is the dimension of $\mathcal{H}$, as follows. Define a linear program $P(i, j, \alpha, \beta)$, where $1 \leq i, j \leq n$ and $\alpha, \beta \in \{-1, +1\}$ as follows:

$$\mathit{max}\ \alpha \langle \mathbf{y} \rangle_i s.t. \tag{4}$$
$$t \geq 0, \mathbf{x} \in \mathit{Closure}(f_1), \mathbf{y} \in \mathit{Closure}(f_2), \mathbf{y} = \mathbf{x} + \varphi(R)t$$
$$\alpha \langle \mathbf{y} \rangle_i \geq 0, \langle \mathbf{x} \rangle_j = \beta,$$

and for every $1 \leq k \leq n, k \neq j$

$$-1 \leq \mathbf{x}_k \leq 1$$

Note that for a fixed i, j, α and β, $P(i, j, \alpha, \beta)$ is a linear programming problem. Further, Equation 3 is equivalent to $\mathit{max}_{i,j,\alpha,\beta} P(i, j, \alpha, \beta)$.

The following lemma summarizes the computation of the weights on the edges, and show that solving Equation 3 is equivalent to solving $4n^2$ linear programming problems.

Lemma 1. *If $\psi(f_1, f_2, R)$ holds, then*

$$W_{\mathit{Reg}}(f_1, f_2, R) = \mathit{max}_{1 \leq i,j \leq n, \alpha, \beta \in \{-1,+1\}} P(i, j, \alpha, \beta).$$

6.3 Model-Checking

In this section, we discuss the algorithm for verifying the conditions in Theorem 1 and Theorem 2.

First, note that in order to check if a region R is exploding, using Proposition 3, it suffices to check if $\varphi(R) \in \mathit{Cone}(R, \mathbf{0})$. This condition is equivalent to checking if $t > 0 \wedge \varphi(R)t \in R$ is satisfiable, which can be verified using an SMT solver for linear real arithmetic.

Next, we describe a dynamic programming solution for analyzing the graph for cycles with weight greater than or equal to 1. It is similar to the the dynamic programming solution to the shortest path algorithm. However, the operation of addition is replaced by multiplication here. We compute iteratively $g^k(f_1, f_2)$ for each $1 \leq k \leq s$, where s is number of vertices in the graph and $g^k(f_1, f_2)$ is the maximum weighted path among all paths from f_1 to f_2 in $G(\mathcal{H}) = (V, E, W)$ of length at most k. The iterative computation is as follows:

1. $g^1(f_1, f_2) = W(f_1, f_2)$ if $(f_1, f_2) \in E$ and -1 otherwise.
2. $g^k(f_1, f_2) = max\{g^{k-1}(f_1, f_2), max_{(f_2', f_2) \in E}(g^{k-1}(f_1, f_2') \times W(f_2', f_2))\}$, for $1 < k \leq s$.

The next lemma states that the maximum weight of the paths in the graph can be used to analyse for existence of cycles of weight greater than or equal to 1.

Lemma 2. *Let g be the function computed above, and s be the number of vertices in $G(\mathcal{H})$. Then:*

1. *$g^s(f, f) \leq 1$ for every $f \in V$ if and only if $G(\mathcal{H})$ does not contain simple cycles with weight > 1.*
2. *$g^s(f, f) < 1$ for every $f \in V$ if and only if $G(\mathcal{H})$ does not contain simple cycles with weight ≥ 1.*

The above lemma relies on the fact that if there exists a cycle of weight > 1, then at least one of the cycles in its simple cycle decomposition has weight > 1. The above algorithm takes time polynomial in the number of vertices or equivalently faces. One can in fact trace the cycles which violates the condition of Theorem 1 or Theorem 2. Such a cycle is a *counter-example* in the abstract weighted graph which shows a potential violation of stability.

7 Experiments

Our abstraction and refinement algorithms have been implement in Python 2.7 and the experiments are run on Mac OS X 10.5.8 with a 2 GHz Inter Core 2 Duo processor and a 4GB 1067 MHz DDR3 memory. We use the SMT solver Z3 version 4.3.2 to solve satisfiability of linear real arithmetic formulas which are required for determining the existence of edges and checking if a region is exploding. We use the linear programming package GLPK version 4.8 for solving the linear optimization problem required in constructing the weights on the edges. There are no standard benchmarks for evaluating the stability of hybrid systems, especially, for the class that we consider. Hence, most of the experiments are performed on hand constructed examples. Many of them are generalizations of the three dimensional example considered in Section 2 by starting with an n-dimensional grid and transforming it systematically into an example in a normal

form of $n+1$-dimensions. We believe that by extending the approach to more complex hybrid systems, we can experiment on examples from real applications.

Our results are tabulated in Table 1. In the table, the column with heading *Var* corresponds to the number of variables in the *PCD*, *Pred* corresponds to the number of predicates, *Reg* to the number of regions, *Faces* to the number of faces, *FT*, *GT*, *VT* and *Total* are the time in seconds for constructing the faces, the weighted graph, the verification (graph analysis for cycles) and the total time, *Lyap* states whether the system is Lyapunov stable or not, *Asymp* states whether the system is asymptotically stable or not, and *GL* and *GA* report the result of the abstraction based analysis stating whether the abstract system is Lyapunov stable and asymptotically stable respectively.

Var	*Pred*	*Reg*	*Faces*	*FT(sec)*	*GT* (sec)	*VT* (sec)	*Total* (sec)	*Lyap*	*GL*	*Asymp*	*GA*
2	1	8	9	2.3	1.8	< 1	5	*Y*	*Y*	*Y*	*Y*
2	2	12	13	5.7	2.6	< 1	9	*Y*	*Y*	*Y*	*Y*
2	3	16	17	10.2	3.8	< 1	15	*Y*	*Y*	*Y*	*Y*
2	4	20	21	15.2	4.5	< 1	21	*Y*	*Y*	*Y*	*Y*
2	5	24	25	21.7	5.6	< 1	28	*Y*	*Y*	*Y*	*Y*
3	1	32	75	36.7	62.4	< 1	100	*Y*	*N*	*N*	*N*
3	2	72	179	104.1	156.9	38	299	*Y*	*N*	*N*	*N*
3	3	128	331	130.7	306.7	575	1013	*Y*	*N*	*N*	*N*

Table 1. Experiments

We consider a two dimensional and a three dimensional example, and experiment with an increasing set of predicates. The face construction and the graph construction time increase almost linearly in the number of faces. The graph analysis time is at most cubic in the size of the graph. However, the number of faces can increase quickly with the increase in the number of predicates. Hence, it is crucial to choose the predicates carefully. Also, observe that there are cases where the system is stable, but the graph construction is too coarse and hence concludes that the system is unstable. Adding more predicates might prove that the system is stable. In this paper, we do not focus on the problem of choosing the right predicates. The future work will focus on this problem by incorporating an abstraction refinement loop to the algorithm.

8 Conclusions

In this paper, we explored model-checking as an alternate approach to stability verification. We proposed a modification of predicate abstraction based on faces in the system to abstract into a finite weighted graph, where the weights track the increase or decrease in the distance of the state along the executions, which can be efficiently constructed using linear programming. The graph was then

verified for the absence of simple cycles whose weight is less than or equal to 1 for Lyapunov stability and less than 1 for asymptotic stability, which can be efficiently solved using dynamic programming. Our experimental results show that the method has the potential to scale.

Our model-checking algorithm returns an abstract counter-example which is a simple cycle with weight greater than 1 for Lyapunov stability and weight greater than or equal to 1 for asymptotic stability. One interesting future direction is to develop a counter-example guided abstraction refinement framework using the abstract counter-example returned in the model-checking phase. Another research direction is to extend the results to more complex dynamics. The extension of our results to the case of hybrid systems with polyhedral constraints (not necessarily non-overlapping invariants) and constant or rectangular flows should be straightforward. More complex classes might require additional work given that the exact post computation is not feasible in general.

9 Acknowledgements

We would like to thank Mahesh Viswanathan for discussions on the topic.

References

1. R. Alur, T. Dang, and F. Ivancic. Counter-Example Guided Predicate Abstraction of Hybrid Systems. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 208–223, 2003.
2. E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.
3. Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138(1):35–65, 1995.
4. M.S. Branicky. Stability of hybrid systems. In H. Unbehauen, editor, *Encylopedia of Life Support Systems*, volume Theme 6.43:Control Sytems, Robotics and Automation, chapter Article 6.43.28.3. UNESCO Publishing, 2004.
5. Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, 2012.
6. E.M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems. *International Journal on Foundations of Computer Science*, 14(4):583–604, 2003.
7. G. Davrazos and N.T. Koussoulas. A review of stability results for switched and hybrid systems. In *Proceedings of the Mediterranean Conference on Control*, 2001.
8. H. Dierks, S. Kupferschmid, and K.G. Larsen. Automatic Abstraction Refinement for Timed Automata. In *Proceedings of Formal Modeling and Analysis of Timed Systems*, pages 114–129, 2007.
9. Parasara Sridhar Duggirala and Sayan Mitra. Lyapunov abstractions for inevitability of hybrid systems. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 115–124, 2012.

10. Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *Hybrid Systems*, pages 258–273, 2005.
11. A. Girard. Reachability of uncertain linear systems using zonotopes. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 291–305, 2005.
12. R. Goebel, R. Sanfelice, and A. Teel. Hybrid dynamical systems. *IEEE Control Systems, Control Systems Magazine*, 29:28–93, 2009.
13. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the ACM Symposium on Theory of Computation*, pages 373–382, 1995.
14. Thomas A. Henzinger. The Theory of Hybrid Automata. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.
15. H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, 1996.
16. D. Liberzon. *Switching in Systems and Control*. Boston : Birkhuser, 2003.
17. Jens Oehlerking, Henning Burchardt, and Oliver E. Theel. Fully automated stability verification for piecewise affine systems. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 741–745, 2007.
18. P. A. Parrilo. *Structure Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, CA, May 2000., 2000.
19. André Platzer and Jan-David Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *International Joint Conference on Automated Reasoning*, pages 171–178, 2008.
20. Andreas Podelski and Silke Wagner. Model checking of hybrid systems: From reachability towards stability. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. Springer, 2006.
21. P. Prabhakar, V. Vladimerou, M. Viswanathan, and G.E. Dullerud. Verifying tolerant systems using polynomial approximations. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 181–190, 2009.
22. Pavithra Prabhakar, Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Hybrid automata-based cegar for rectangular hybrid automata. In *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation*, 2013.
23. Pavithra Prabhakar, Geir E. Dullerud, and Mahesh Viswanathan. Pre-orders for reasoning about stability. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 197–206, 2012.
24. Pavithra Prabhakar and Mahesh Viswanathan. A dynamic algorithm for approximate flow computations. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 133–143, 2010.
25. Pavithra Prabhakar and Mahesh Viswanathan. On the decidability of stability of hybrid systems. *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, 2013.
26. Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 477–492. Springer, 2004.
27. A. Puri, V.S. Borkar, and P. Varaiya. ϵ-approximation of differential inclusions. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 362–376, 1995.