An algorithmic approach to stability verification of polyhedral switched systems

Pavithra Prabhakar[1] and Miriam García Soto[2]

Abstract— **We present an algorithmic approach for analyzing Lyapunov and asymptotic stability of polyhedral switched systems. A polyhedral switched system is a hybrid system in which the continuous dynamics is specified by polyhedral differential inclusions, the invariants and guards are specified by polyhedral sets and the switching between the modes do not involve reset of variables. The analysis consists of first constructing a finite weighted graph from the switched system and a finite partition of the state space, which represents a conservative approximation of the switched system. Then, the weighted graph is analyzed for certain structural properties, satisfaction of which implies stability. However, in the event that the weighted graph does not satisfy the properties, one cannot, in general, conclude that the system is not stable due to the conservativeness of the graph. Nevertheless, when the structural properties do not hold in the graph, a counter-example indicating a potential reason for the failure is returned. Further, a more precise approximation of the switched system can be constructed by considering a finer partition of the state-space in the construction of the finite weighted graph. We present experimental results on analyzing stability of switched systems using the above method.**

I. INTRODUCTION

In this paper, we focus on the problem of automatic stability verification of polyhedral switched systems. Switched systems [1] are a special class of hybrid systems [2] - systems exhibiting mixed discrete continuous behaviors - in which the continuous state of the system does not change during a mode switch. Switched systems are a natural model in supervisory control, wherein the plant consists of a finite number of operational modes, and the supervisor continuously observes the state of the system and takes decisions regarding the mode switches. Stability has been extensively investigated in the context of switched systems, and several sufficient conditions on the system and the switching behavior which ensure stability have been proposed (see [1], [3] and references therein).

One of the widely used approaches to stability verification of switched system is based on the notions of common and multiple Lyapunov functions [4], [5], [6]. In the former, a common function which acts as a Lyapunov function for every mode is sought, and in the latter, a set of Lyapunov functions one for each mode is sought such that together they satisfy some consistency conditions on the switching. Automated verification of stability based on Lyapunov function can be characterized as deductive verification in the formal methods terminology. It encompasses a search for a Lyapunov function based on a template, such as a polynomial with coefficients as parameters, which serves as a candidate function. The requirements of Lyapunov function are encoded as a sum-of-squares programming problem over the template, which can be efficient solved using tools such as SOSTOOLS [7], [8], [9]. One of the major limiting factors of this approach is the ingenuity required in providing the right templates; and automatically learning the templates is a challenge which has not been adequately addressed. Moreover, if a template fails to satisfy the conditions of Lyapunov function, then it does not provide insights into the potential reasons for instability or towards the choice of a better template. To overcome these limitations, we propose an algorithmic approach.

Our approach consists of constructing a finite weighted graph which represents a conservative approximation of the switched systems, and inferring stability by analyzing certain properties of the graph. We focus on the class of polyhedral switched systems (*PSS*) and present a detailed algorithm for analyzing both Lyapunov and asymptotic stability. These are systems in which the invariants for the modes and the guards on the switching are convex polyhedral sets; further, the dynamics in each mode is specified as a polyhedral differential inclusion $\dot{x} \in P$, where P is a compact convex polyhedral set.

The algorithm takes as input a *PSS* $\mathcal{H}$ and a finite partition of the state-space into convex polyhedral sets $\mathcal{P}$, and outputs a finite weighted graph $\mathcal{G}(\mathcal{H}, \mathcal{P})$. The vertices of the graph correspond to pairs consisting of a mode of the system and an element of the partition. An edge between two mode-element pairs indicates the existence of an execution starting from the first mode and a point on the first element to the second mode and a point on the second element such that it remains in a single element at all the intermediate time instances. And the weight on the edge corresponds to the maximum *scaling* - ratio of the final state to the initial state - over all such executions. Hence, corresponding to every execution of the system, there exists a path in the graph which tracks the scalings associated with various time points in the execution.

We present efficiently verifiable conditions on the graph, such that the satisfaction of the conditions implies Lyapunov and asymptotic stability. For instance, if the graph has no edges with weight $+\infty$ and no cycles in the graph with the product of weights on the edges greater than 1, then the system is Lyapunov stable. However, if the graph does not satisfy the properties, then, in general, one cannot

[1]Pavithra Prabhakar is with the Faculty of IMDEA Software Institute, Madrid, Spain. Email: pavithra.prabhakar@imdea.org

[2]Miriam García Soto is a PhD student with the IMDEA Software Institute and ETSIINF, Universidad Politécnica de Madrid, Madrid, Spain Email: miriam.garcia@imdea.org

conclude that the system is not Lyapunov stable, due to the conservativeness of the graph construction. However, an interesting feature of our analysis is a potential reason for instability in the form of a counter-example. For instance, the graph returns a cycle with the product of weights on the edges greater than 1 indicating the possibility of a diverging execution obtained by traversing the cycle infinitely many times. Another interesting feature is the ability to construct a less conservative abstraction, by starting with a finer partition, which can in turn be obtained by analyzing the counter-example.

Construction of the finite weighted graph involves computing a non-trivial reachability predicate which captures all pairs of states of the system for which there is an execution from the first state to second while remaining within a single element of the partition. Existence of an edge then corresponds to satisfiability of the predicate and the weight corresponds to solving an optimization problem over the predicate. We show that we can construct a formula which is a boolean combination of linear constraints which is equivalent to the reachability predicate and hence compute the weight by solving a bunch of linear programming problems. The construction of the formula is involved owing to the fact that the number of mode switches that can occur during an execution within an element of the partition is unbounded due to the presence of cycles in the underlying switching graph. We reduce the analysis to that of an acyclic graph using the notion of strongly connected components, and hence, bound the number of switches for the purpose of analysis.

The algorithm has been implemented in a tool called AVERIST (Algorithmic VERIfier for STability). We illustrate the algorithmic approach on an example using the tool, including counter-examples and refinement. Currently, the choice of the appropriate predicates to add for the refinement is carried out through manual examination of the counter-example. Future work will focus on automating this process through a counter-example guided abstraction refinement approach [10].

Preliminary ideas of the algorithm appear in [11] for piecewise constant derivative system - systems with constant dynamics, non-overlapping invariants and guards, and in [12] for two-dimensional rectangular switched systems. In contrast to the former, we consider a much larger class of systems, and to the latter, we extend the algorithm to polyhedral dynamics, higher dimensions and provide experimental results. The algorithm presented can be extended to richer dynamics by providing constructions for the reachability predicates over the new dynamics.

II. Preliminaries

Let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{N}$ denote the set of reals, non-negative reals and natural numbers, respectively. Given a function F, we use $Dom(F)$ to denote the domain of F. Given a set $A \subseteq Dom(F)$, we denote by $F\lfloor_A$, the restriction of F to the domain A.

a) Intervals, time domain and interval domain: An *interval* is a closed convex subset of $\mathbb{R}$. Given an interval I, *first*(I) denotes the greatest lower bound of I and *last*(I) denotes the least upper bound of I. A *time domain* is an interval I such that *first*$(I) = 0$. An *interval domain* is a finite or infinite sequence of intervals $\iota = I_0 I_1 \cdots$ such that *first*$(I_0) = 0$, *last*$(I_i) =$ *first*(I_{i+1}) for all i, and if $Dom(\iota)$ is infinite, then for every $n \in \mathbb{N}$, there exists $m \in \mathbb{N}$ such that $n \in I_m$. We denote by $[\![\iota]\!]$ the interval $\cup_{i \in Dom(\iota)} \iota(i)$.

b) Continuous state-space: We use $|x|$ to denote the infinity norm of $x \in \mathbb{R}^n$ and $x \cdot y$ to denote the dot product of $x, y \in \mathbb{R}^n$. Given $\epsilon \in \mathbb{R}_{\geq 0}$, we use $B_\epsilon(x)$ to denote an open ball around x of radius ϵ, that is, $B_\epsilon(x) = \{y \,|\, |x - y| < \epsilon\}$.

c) Convex polyhedral sets: Let *PolySets*(n) denote the set of all convex polyhedral sets contained in $\mathbb{R}^n$, and *CPolySets*(n) denote the set of closed sets in *PolySets*(n). A convex polyhedral set p is *elementary* if p has an empty intersection with the boundary of the closure of p. We will use $[\![C]\!]$ to denote the convex polyhedral set defined by a set of linear constraints C. Given a set $p \subseteq \mathbb{R}^n$, *Plane*(p, n) is the intersection of all hyper-planes of dimension n, denoted *Hyper*(n), which contain p. *Plane*$(p, n) = \{x \in \mathbb{R}^n \,|\, \forall h \in$ *Hyper*$(n), (p \subseteq h \Rightarrow x \in h)\}$.

A *partition* $\mathcal{P}$ of $S \subseteq \mathbb{R}^n$ into convex polyhedral sets is a finite set of convex polyhedral sets $\{P_1, \cdots, P_k\}$ such that $\cup_{i=1}^k P_i = S$ and for each $i \neq j$, $P_i \cap P_j = \emptyset$. An *elementary partition* is a partition in which all the convex polyhedral sets are elementary; the sets are referred to as elements.

III. Polyhedral Switched Systems

A switched system [1] models supervisory control in which the supervisor observes the state of the system and switches between a finite number of operational modes of the system. In each mode, the continuous state evolves according to a pre-assigned continuous dynamics and satisfies certain invariant conditions; mode switch occurs when certain guards are satisfied, and in particular, the continuous state remains the same during the switch. We focus on the class of switched systems in which the continuous dynamics is specified using polyhedral differential inclusions, and the invariants and guards are specified using convex polyhedral sets.

Definition. A *polyhedral switched system* (*PSS*) is a tuple $\mathcal{H} = (Loc, Edges, Cont, Flow, Inv, Guard)$, where:

- *Loc* is a finite set of control modes or locations;
- *Edges* $\subseteq$ *Loc* $\times$ *Loc* is a finite set of edges;
- *Cont* $= \mathbb{R}^n$, for some n, is the continuous state-space;
- *Flow* : *Loc* $\to$ *CPolySets*(n) is the flow function;
- *Inv* : *Loc* $\to$ *PolySets*(n) is the invariant function; and
- *Guard* : *Edges* $\to$ *PolySets*(n) is the guard function.

We call n, the dimension of $\mathcal{H}$.

The switched system starts in a location q and a continuous state x. In a mode q, the continuous state evolves inside *Inv*(q) such that the differential of the evolution at anytime lies within *Flow*(q). The mode can switch from q_1 to q_2 if (q_1, q_2) is an edge of the system and the continuous state at the switching satisfies the guard associated with the edge.

Switched systems differ from a hybrid system in that the continuous state does not change in a switched system during a mode switch.

A. Semantics

Next, we present the semantics of a polyhedral switched system as a set of executions of the system.

Definition. An *execution* σ of a *PSS* $\mathcal{H} = (\mathit{Loc}, \mathit{Edges}, \mathit{Cont}, \mathit{Flow}, \mathit{Inv}, \mathit{Guard})$ of dimension n is a triple (ι, η, γ), such that:

- ι is an interval domain;
- $\eta : [\![\iota]\!] \to \mathbb{R}^n$ such that for each $i \in \mathit{Dom}(\iota)$, $\eta|_{\iota(i)}$ is a differentiable function;
- $\gamma : \mathit{Dom}(\iota) \to \mathit{Loc}$ such that:
  - for all $i \in \mathit{Dom}(\iota)$, for all $t \in \iota(i)$, $\eta(t) \in \mathit{Inv}(\gamma(i))$ and $\dot{\eta}(t) \in \mathit{Flow}(\gamma(i))$;
  - for all $0 \leq i < |\mathit{Dom}(\iota)|$, $(\gamma(i), \gamma(i+1)) \in \mathit{Edges}$ and $\eta(\mathit{last}(\iota(i))) \in \mathit{Guard}((\gamma(i), \gamma(i+1)))$.

We denote the set of all executions of $\mathcal{H}$ by $\mathit{Exec}(\mathcal{H})$.

Definition. An execution $\sigma = (\iota, \eta, \gamma)$ of $\mathcal{H}$ is said to be *complete* if $\mathit{Dom}(\eta) = [0, \infty)$; otherwise, it is called *finite*.

Definition. An execution $\sigma = (\iota, \eta, \gamma)$ is said to be *piecewise-linear* if η is piecewise linear. We denote the set of all piecewise-linear executions of $\mathcal{H}$ by $\mathit{PWExec}(\mathcal{H})$.

IV. Stability: Lyapunov and Asymptotic

We define Lyapunov and asymptotic stability. We will assume without loss of generality that the origin $\bar{0}$, is the equilibrium point.

Definition. A *PSS* $\mathcal{H}$ is said to be *Lyapunov stable* with respect to $\bar{0}$, if for every real $\epsilon > 0$, there exists a real $\delta > 0$ such that for every execution $\sigma = (\iota, \eta, \gamma) \in \mathit{Exec}(\mathcal{H})$ with $\eta(0) \in B_\delta(\bar{0})$, $\eta(t) \in B_\epsilon(\bar{0})$ for every $t \in \mathit{Dom}(\eta)$.

Given an execution σ, it is said to converge to x, denoted $\mathit{Conv}(\sigma, x)$, if for every real $\epsilon > 0$, there exists a $T \in \mathit{Dom}(\eta)$, such that $\eta(t) \in B_\epsilon(x)$ for every $t \geq T$.

Definition. A *PSS* $\mathcal{H}$ is said to be *asymptotically stable* with respect to $\bar{0}$, if it is Lyapunov stable with respect to $\bar{0}$ and there exists a $\delta > 0$ such that for every complete execution $\sigma = (\iota, \eta, \gamma) \in \Sigma$, $\eta(0) \in B_\delta(\bar{0})$ implies $\mathit{Conv}(\sigma, \bar{0})$.

We make the following observations the first two of which extend similar observations in [12], [11]:

- Firstly, we notice that in the definition of Lyapunov stability, we only need to consider $\epsilon \in [0, r]$ for some positive real number r. Hence, Lyapunov and asymptotic stability depend only on the behavior of $\mathcal{H}$ in a small neighborhood around the origin.
- For the purpose of stability analysis, the only constraints on the invariants and guards of interest are homogeneous linear constraints (constraints of the form $a \cdot x \sim 0$, $\sim \in \{<, \leq\}$). We assume from now on that $\mathcal{H}$ is in a normal form, that is, the invariants and guards of *PSS* $\mathcal{H}$ are defined by homogeneous linear constraints.
- In the definition of Lyapunov and asymptotic stability, we can replace $\mathit{Exec}(\mathcal{H})$ by $\mathit{PWExec}(\mathcal{H})$, that is, we only need to consider piecewise linear executions of $\mathcal{H}$. This follows from the fact that given any execution $\sigma \in \mathit{Exec}(\mathcal{H})$, there exists an execution $\sigma' \in \mathit{PWExec}(\mathcal{H})$ where the time spent in each of the pieces is bounded by T and the states at the end-points of the pieces coincide with the states at the corresponding times in σ. The above fact is a consequence of the property that every trajectory with finite time domain $[0, T]$ and derivative in a convex polyhedral set P at all time points, can be replaced by a linear trajectory with the same time domain $[0, T]$, same values for states at the end-points, and a constant derivate in P. Hence, from now on we take the semantics of $\mathcal{H}$ to be $\mathit{PWExec}(\mathcal{H})$.

V. Stability verification procedure

In this section, we present an algorithmic approach for verifying stability of polyhedral switched systems. This is an extension of the algorithms in [11] for piecewise constant derivative systems and in [12] for two dimensional rectangular switched systems. The verification procedure consists of two parts:

1) Extracting a finite weighted graph from the polyhedral switched system using an elementary partition of the state-space.
2) Analyzing the graph for deducing stability.

We discuss the two parts in detail in the following. However, we will defer the computational aspects to the next section.

A. Formal definition of the graph

The graph construction takes as input an elementary partition of the state-space and a polyhedral switched system, and outputs a finite weighted graph. The graph captures the sequence of elements the executions of the system traverse using the notion of an almost-inside element execution.

Definition. Let $\mathcal{H}$ be a polyhedral switched system and $\mathcal{P}$ an elementary partition of its state-space. Given an element p of $\mathcal{P}$, a *p-execution* is an execution σ such that $\eta(t) \in p$ for every $0 < t < \mathit{last}(\mathit{Dom}(\eta))$. An *element execution* is an execution which is a p-execution for some element p of $\mathcal{P}$.

The weights in the graph correspond to scalings of executions which measure the relative distance of the end-points of the executions to the origin. Given a finite execution σ, its scaling, denoted $\mathit{Scaling}(\sigma)$ is given by:

$$\mathit{Scaling}(\sigma) = \frac{|\eta(\mathit{last}(\mathit{Dom}(\eta)))|}{|\eta(0)|}.$$

The vertices of the graph correspond to location-element pairs; edges between two location-element pairs correspond to the presence of an element execution from the first location-element pair to the second, and weights on edges correspond to an upper-bound on the scaling of the executions corresponding to the edge.

Let us fix a polyhedral switched system $\mathcal{H} = (\mathit{Loc}, \mathit{Edges}, \mathit{Cont}, \mathit{Flow}, \mathit{Inv}, \mathit{Guard})$ in normal form, and an element partition $\mathcal{P}$ of the state-space. We will assume that every element p of the partition $\mathcal{P}$ is such that for each invariant or guard r, p is either contained in r or is disjoint from r. Given $q_1, q_2 \in \mathit{Loc}$ and $p_1, p_2 \in \mathcal{P}$, let $\mathit{AI}(\sigma, (q_1, p_1), (q_2, p_2))$ denote the fact that σ is an element execution from a state in (q_1, p_1) to a state in (q_2, p_2).

The weighted graph $\mathcal{G}(\mathcal{H}, \mathcal{P}) = (V, E, W)$ is defined as follows.

1) The set of vertices V is given by $\mathit{Loc} \times \mathcal{P}$;
2) The set of edges $E \subseteq V \times V$ is given by $\{((q_1, p_1), (q_2, p_2)) \mid \exists \sigma, \mathit{AI}(\sigma, (q_1, p_1), (q_2, p_2))\}$;
3) The weight function W is given by $W(((q_1, p_1), (q_2, p_2))) = \sup\{\mathit{Scaling}(\sigma) \mid \exists \sigma, \mathit{AI}(\sigma, (q_1, p_1), (q_2, p_2))\}$.

B. Stability Criteria

The weighted graph $\mathcal{G}(\mathcal{H}, \mathcal{P}) = (V, E, W)$ is analysed for the satisfaction of the following properties:

P1 For every edge $e \in E$, $W(e) \in \mathbb{R}_{\geq 0}$.

P2 For every simple cycle (cycles with no repeating vertices) of $\mathcal{G}(\mathcal{H}, \mathcal{P})$, the weight of the cycle (product of the weights associated with its edges) is less than or equal to 1.

P3 Every simple cycle of $\mathcal{G}(\mathcal{H}, \mathcal{P})$ has weight less than 1.

Intuitive, Condition [P1] implies that there might exist an execution which diverges from the origin in an element, Condition [P2] implies that there might exist executions starting arbitrarily close to the origin which switch between the elements infinitely often and diverge, and Condition [P3] ensure that there might exist executions starting arbitrarily close to the origin which switch between the elements infinitely often and do not converge to the origin. In addition, we need the following property, which captures convergence within an element.

P4 For every element $p \in \mathcal{P}$, there exists no complete execution σ such that $\eta(t) \in p$, for all $t \in [0, \infty)$ and $\mathit{Conv}(\sigma, \bar{0})$ does not hold.

The following theorems characterize the criteria for Lyapunov stability and asymptotic stability in terms of the above properties.

Theorem *1:* (Lyapunov) If Conditions [P1] and [P2] hold for $\mathcal{G}(\mathcal{H}, \mathcal{P})$, then $\mathcal{H}$ is Lyapunov stable.

Theorem *2:* (Asymptotic) If Conditions [P1], [P3] and [P4] hold for $\mathcal{G}(\mathcal{H}, \mathcal{P})$, then $\mathcal{H}$ is asymptotically stable.

VI. Computational Aspects

A. Preliminaries

Let us fix a polyhedral switched system $\mathcal{H} = (\mathit{Loc}, \mathit{Edges}, \mathit{Cont}, \mathit{Flow}, \mathit{Inv}, \mathit{Guard})$ of dimension n. The underlying graph of a hybrid system $\mathcal{H}$, denoted $\mathcal{UG}(\mathcal{H})$, is the pair of the set of locations and the set of edges $(\mathit{Loc}, \mathit{Edges})$.

Given a set $p \subseteq \mathbb{R}^n$, define $\mathcal{H} \cap p$ to be the *PSS* $(\mathit{Loc}_p, \mathit{Edges}_p, \mathit{Cont}_p, \mathit{Flow}_p, \mathit{Inv}_p, \mathit{Guard}_p)$, where:

- $\mathit{Loc}_p = \{q \in \mathit{Loc} \mid p \subseteq \mathit{Inv}(q)\}$,
- $\mathit{Edges}_p = \{(q_1, q_2) \in (\mathit{Loc}_p \times \mathit{Loc}_p) \cap \mathit{Edges} \mid p \subseteq \mathit{Guard}((q_1, q_2))\}$,
- $\mathit{Flow}_p(q) = \mathit{Flow}(q) \cap \mathit{Plane}(p, n)$ for each $q \in \mathit{Loc}_p$,
- $\mathit{Inv}_p(q) = p$ for each $q \in \mathit{Loc}_p$, and
- $\mathit{Guard}_p(e) = p$, for each $e \in \mathit{Edges}_p$.

Proposition *1:* σ is a p-execution of $\mathcal{H}$ if and only if σ is an execution of $\mathcal{H} \cap p$.

Proof: If σ is an execution of $\mathcal{H} \cap p$, then it is an execution of $\mathcal{H}$ which always remains within p except possibly at end times. Hence, it is a p-execution of $\mathcal{H}$.

Suppose σ is a p-execution of $\mathcal{H}$. Then it always remains within p except possibly at end times. We need to show that the derivative at all times belongs to $\mathit{Plane}(p, n)$. We observe that each of the hyper-planes containing p are defined by homogenous linear constraints, since, p itself is defined by homogeneous linear constraints. Suppose that the derivative of σ does not belong to $\mathit{Plane}(p, n)$ at some point t, then σ necessarily goes out of $\mathit{Plane}(p, n)$ at t for sometime. Since $p \subseteq \mathit{Plane}(p, n)$, this implies that σ is not a p-execution. Hence, the derivative belongs to $\mathit{Plane}(p, n)$. ■

Let us fix an element $p \in \mathcal{P}$. Consider a path $\pi = q_1, q_2, \ldots, q_k$ in $\mathcal{UG}(\mathcal{H} \cap p)$. Given elements p_1 and p_2 which are subsets of the boundary of p, the following formula represents points $x \in p_1$ and $y \in p_2$, such that there exists a p-execution from x reaching y which follows π, that is, spends some time t_1 in q_1, then t_2 in q_2, etc. The z_is represent the continuous state just before exiting location q_i.

$$\psi_\pi(x, y) \equiv \exists t_1, \ldots, t_k, z_0, z_1, \ldots, z_k : x \in p_1 \wedge y \in p_2 \wedge \quad (1)$$

$$z_0 = x \wedge z_k = y \wedge \bigwedge_{i=1}^{k} \frac{z_{i+1} - z_i}{t_i} \in \mathit{Flow}_p(q_i) \wedge \bigwedge_{i=1}^{k-1} z_i \in \mathit{Inv}_p(q_i)$$

We note that $\frac{z_{i+1} - z_i}{t_i} \in \mathit{Flow}(q_i)$ can be expressed as a conjunction of linear constraints by substituting $\frac{z_{i+1} - z_i}{t_i}$ in the linear constraints defining $\mathit{Flow}_p(q_i)$ and multiplying thoughout by t_i. For instance, if $a \cdot x \sim c$ is a constraint of $\mathit{Flow}(q_i)$, then the corresponding constraints on z_i, z_{i+1}, t_i is $a \cdot (z_{i+1} - z_i) \sim c t_i$.

Next, we present a result about a strongly connected *PSS*. If there is a path from every vertex of $\mathcal{UG}(\mathcal{H})$ to every other vertex, then we say that $\mathcal{H}$ is *strongly connected*. Next, we state a property of strongly connected *PSS*.

Lemma *1:* Suppose $\mathcal{H} \cap p$ is a strongly connected *PSS* with locations $\{q_1, \ldots, q_k\}$. Let q_x, q_y be locations of $\mathcal{H} \cap p$ and $x, y \in p$. There is an execution from (q_x, x) to (q_y, y) in $\mathcal{H} \cap p$ if and only if $\psi(x, y)$ is satisfiable, where:

$$\hat{\psi}(x, y) \equiv \exists t_1, \ldots, t_k, z_0, z_1, \ldots, z_k : x \in p \wedge y \in p \wedge \quad (2)$$

$$z_0 = x \wedge z_k = y \wedge \bigwedge_{i=1}^{k} \frac{z_{i+1} - z_i}{t_i} \in \mathit{Flow}_p(v_i)$$

Note that unlike in Equation 1, in Equation 2, we do not check for satisfaction of invariants at the intermediate points

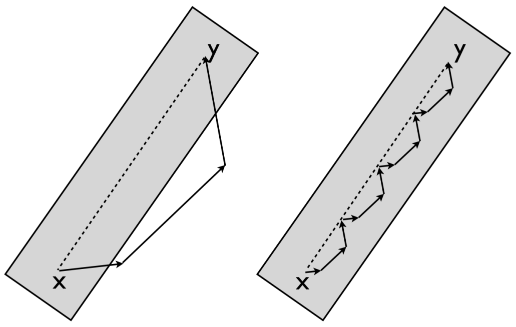

Fig. 1. Scale-and-repeat technique

z_i. Also, the formula is independent of q_x and q_y. Below we sketch a proof of the lemma.

(Only if) Suppose σ is an execution of $\mathcal{H} \cap p$ from (q_x, x) to (q_y, y). Let t_i be the total time spent in location q_i in σ. Let x_i be the sum of the distances traveled while in q_i. Set $z_0 = x$, $z_i = \sum_{j=1}^{i} x_i$. These values satisfy Equation 2. The condition corresponding to flow is satisfied due to the convexity of the sets $\mathit{Flow}_p(q_i)$, that is, if different flows in $\mathit{Flow}_p(q_i)$ were followed during different visits to q_i, then the same distance can be travel in the same time with a single flow in $\mathit{Flow}_p(q_i)$.

(If) Suppose there exists values for z_i, t_i such that $\hat{\psi}(x, y)$ is satisfied. Let us construct a potential execution which starts at z_0 follows the flow $\frac{z_1 - z_0}{t_1}$ in q_1 for time t_1, then follows $\frac{z_2 - z_1}{t_2}$ in q_2 for time t_2, and so on. An example of such a potential execution is shown in Figure 1. This potential execution might violate two conditions of an execution of $\mathcal{H} \cap p$: (1) there might not be an edge from q_i to q_{i+1}, also the initial and finial locations may not be q_x and q_y, respectively, and (2) the invariant p might not be satisfied at the intermediate points. We can fix (1) by replacing the edge from q_i to q_{i+1} by an execution corresponding to instantaneous jumps along a path from q_i to q_{i+1}, which we know exist due to the strong connectedness of $\mathcal{H} \cap p$. To fix (2), we use a scale and repeat technique as illustrated on the right in Figure 1. First, we choose an $\epsilon > 0$, such that $B_\epsilon(xy) \cap \mathit{Plane}(p, n)$ is contained in p, where xy is the line joining xy. Such an ϵ can always be found since p is elementary and hence x and y are not on it boundary. Next, we scale down the execution by a factor of α, wherein we spend time αt_i in each location. α is chosen such that the resulting execution is within an ϵ-tube around the line joining x and y. Then multiple such executions are stitched together by adding instantaneous jumps if required between the last location of one copy and the first location of the next copy.

B. Weighted Graph Construction

In this section, we sketch the algorithm for determining if there exists an edge between nodes (q_1, p_1) and (q_2, p_2), and compute the weight in case the edge exists. Our broad approach is to define an SMT formula $\varphi_{(q_1,p_1),(q_2,p_2),p}(x, y)$ such that the formula evaluates to true for values of $x \in p_1$ and $y \in p_2$ such that there is a p-execution from (q_1, x) to (q_2, y). Hence, the existence of an edge between (q_1, p_2) and (q_2, p_2) is equivalent to the satisfiability of the formula $\bigvee_{p \in \mathcal{P}} \varphi_{(q_1,p_1),(q_1,p_2),p}(x, y)$. And the weight is equivalent to $\sup |y|/|x|$ such that (x, y) satisfy $\bigvee_{p \in \mathcal{P}} \varphi_{(q_1,p_1),(q_1,p_2),p}(x, y)$. The above optimization problem can be converted to $4|\mathcal{P}|n^2$ linear programming problems using the same transformation as in [11].

To define $\varphi_{(q_1,p_1),(q_1,p_2),p}(x, y)$, let us analyse a p-execution σ from (q_1, p_1) to (q_2, p_2). It can be split into a sequence of 5 subexecutions $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5$, where σ_1 represents the instantaneous mode changes on p_1, σ_2 is a linear p-execution from p_1 to p with no mode change, σ_3 is a p-execution from p to p (with possibly several mode changes), σ_4 is a linear p-execution from p to p_2 with no mode change, and σ_5 represents the instantaneous mode changes on p_2. The crux of the formula is in constructing a formula capturing the end-points of executions of the form σ_3, in which there is no a priori bound on the number of switching.

To compute the relational reachability predicate for executions of type σ_3, first, we decompose $\mathcal{UG}(\mathcal{H} \cap p)$ into maximal strongly connected components $\{C_1, \ldots, C_m\}$. The quotient graph of $\mathcal{UG}(\mathcal{H} \cap p)$ with vertices $\{C_1, \ldots, C_m\}$ is acyclic. Since the number of paths in the quotient graph are finite, it suffices to compute the reachability predicate for each path in this acyclic graph. Further, it suffices to compute the reachability predicate for a single component, since, they can be composed to obtain the predicate for the whole path. The reachability predicate for a single component can by computed using Lemma 1.

C. Verifying Conditions [P1-P4]

Given the graph $\mathcal{G}(\mathcal{H}, \mathcal{P})$, conditions [P1-P3] can be efficiently verified with time complexity $O(|V||E|)$, where $|V|$ and $|E|$ are the number of vertices and edges in $\mathcal{G}(\mathcal{H}, \mathcal{P})$, respectively. Verifying Condition [P1] consists of iterating over the edges and checking if the values of the weights is $< +\infty$. Conditions [P2] and [P3] can be check by a modification of Bellman-Ford algorithm to detect negative cycles (the addition operation is replaced by multiplication).

Next, we present a verifiable condition which is equivalent to Condition [P4].

Note that any complete execution which remains within p will eventually enter a strongly connected component of $\mathcal{H} \cap p$ and stay there. It will not converge if $\bar{0}$ is in the convex hull of the flows associated with the locations in that component or there exists a vector in the convex hull which points into p from the origin (and hence diverges). Hence, Condition [P4] is equivalent to checking:

P4' For each $C \in \mathit{SCC}(\mathcal{H} \cap p)$, $\bar{0} \notin \mathit{CHull}(\bigcup_{q \in C} \mathit{Flow}(q))$ and $\mathit{CHull}(\bigcup_{q \in C} \mathit{Flow}(q)) \cap p = \emptyset$.

VII. Implementation

The stability verification procedure has been implement in Python 2.7.3 in a tool called AVERIST (Algorithmic VERIfier for STability). It uses the Z3 SMT solver [13]

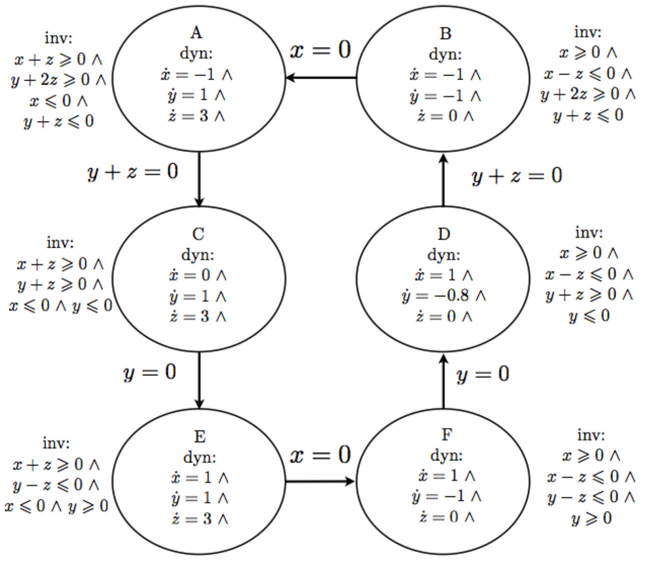

Fig. 2. Switched System Example

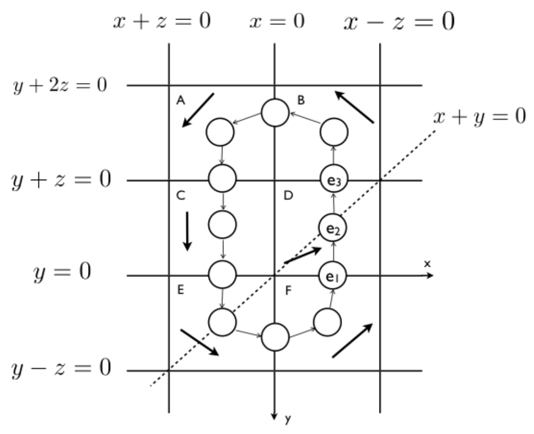

Fig. 3. Illustration of Abstraction-Refinement

to check satisfiability of the φ formulas to determine the existence of edges in the graph $\mathcal{G}(\mathcal{H}, \mathcal{P})$. It uses GLPK, the GNU linear programming kit, for solving the linear optimization problems over φ to compute the weights on the edges. The NetworkX Python package is used for graph algorithms required in computing the formula and in analysing of graphs for stability criteria.

We illustrate the algorithm on an example which has been verified using AVERIST. The switched system is shown in Figure 2 and pictorial illustration of the system when the value of the variable $z = 1$ is shown in Figure 3. The system does not blow up while remaining in any single cell, and as we will see from the proof of stability given by the tool, the system does not have any infinite executions. Hence, the scaling associated with any execution is bounded and the system is both Lyapunov and asymptotically stable.

Part of the graph constructed by the algorithm is shown in Figure 3. It has a cycle with weight > 1. The same is returned by the tool. However, observe that the path e_1 to e_2 to e_3 is infeasible even if both the edges e_1 to e_2 and e_2 to e_3 are feasible. We add the constraint $x + y = 0$ shown by the dotted line in Figure 3 to the partition to create a finer partition. This breaks the cycle and AVERIST returns that the system is stable.

VIII. Conclusions

We presented an algorithmic approach to stability verification of polyhedral switched systems based on a finite weighted graph construction. An interesting feature of our technique is that it returns a counter-example in the event that the graph fails to infer stability. One future direction will focus on the use of the counter-example to automatically construct a finer abstraction [10]. Another direction will focus on the extension of the approach to linear dynamics, for which computing the reachability predicate for determining the edges and the weights is more complicated.

IX. Acknowledgement

The research leading to the results in the paper has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n 631622. The first author holds a *venia docendi* from UPM.

References

[1] D. Liberzon, *Switching in Systems and Control*. Boston : Birkhäuser, 2003.

[2] T. A. Henzinger, "The Theory of Hybrid Automata," in *Logic in Computer Science*, 1996, pp. 278–292.

[3] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: A survey of recent results," *IEEE Trans. Automat. Contr.*, vol. 54, no. 2, pp. 308–322, 2009.

[4] J. C. Geromel and P. Colaneri, "Stability and stabilization of continuous-time switched linear systems," *SIAM J. Control Optim.*, vol. 45, no. 5, pp. 1915–1930, Dec. 2006.

[5] J. P. Hespanha, "Uniform stability of switched linear systems: extensions of lasalle's invariance principle." *IEEE Trans. Automat. Contr.*, vol. 49, no. 4, pp. 470–482, 2004.

[6] P. Mason, U. V. Boscain, and Y. Chitour, "Common polynomial lyapunov functions for linear switched systems." *SIAM J. Control and Optimization*, vol. 45, no. 1, pp. 226–245, 2006.

[7] P. A. Parrilo, *Structure Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, CA, 2000.

[8] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Conference on Decision and Control*, 2002.

[9] E. Möhlmann and O. E. Theel, "Stabhyli: a tool for automatic stability verification of non-linear hybrid systems," in *Hybrid Systems: Computation and Control*. ACM, 2013, pp. 107–112.

[10] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-Guided Abstraction Refinement," in *Conference on Computer Aided Verification*, 2000, pp. 154–169.

[11] P. Prabhakar and M. G. Soto, "Abstraction based model-checking of stability of hybrid systems," in *Conference on Computer Aided Verification*, 2013, pp. 280–295.

[12] P. Prabhakar and M. Viswanathan, "On the decidability of stability of hybrid systems," *Hybrid Systems: Computation and Control*, 2013.

[13] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, vol. 4963. Springer, 2008, pp. 337–340.