An Algorithmic Approach to Global Asymptotic Stability Verification of Hybrid Systems

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

In this paper, we present an algorithmic approach to global asymptotic stability (GAS) verification of hybrid systems. Our broad approach consists of reducing the GAS verification to the verification of a region stability (RS) analysis problem and an asymptotic stability (AS) analysis problem. We use a recently developed quantitative predicate abstraction technique for AS analysis and extract from it a stability zone with respect to which we perform RS analysis. We present a new algorithm for RS analysis based on abstractions. While we develop the theory for polyhedral hybrid systems, our broad approach of decomposing GAS analysis to RS and AS analysis can be applied to more general class of systems including linear hybrid systems. As a proof of concept, we apply the GAS verification algorithm to a linear hybrid system model of a cruise control for an automatic gearbox, and provide a semi-automated proof of GAS.

I. Introduction

Formal verification of embedded and cyber-physical systems has gained prominence in the recent years owing to the safety criticality of the application areas. Hybrid systems refer to systems exhibiting mixed discrete-continuous behaviors, and thus, are an apt formalism for modeling embedded software (that execute in discrete steps) interacting with the physical world (that evolve continuously). In this paper, we address the problem of verification of an important property in the context of hybrid control systems, namely, global asymptotic stability. It captures the property that a system behavior starting at any state of the system will converge to a desired state and can be maintained there under small perturbations. For instance, one expects a cruise control to drive the vehicle velocity to a set velocity and also to maintain the velocity at the set velocity in the presence of disturbances that arise due to uphills and downhills on the road.

Stability analysis [8], [9] is a well-studied problem in control theory, especially for continuous dynamical systems. Asymptotic stability is a classical notion that states that small perturbations in the equilibrium point (a desired state in which the system remains under no perturbations), lead to only small perturbations in the behaviors of the system; and all executions starting in a small neighborhood of the system converge to the equilibrium point. Asymptotic stability is a local property about a small neighborhood around the equilibrium. In contrast, GAS requires convergence to the equilibrium point starting from any initial state. Region stability is a practical notion of stability that relaxes the condition of convergence to the equilibrium point in GAS to reachability to a small region around the equilibrium.

Asymptotic and global asymptotic stability analysis methods in control theory rely on exhibiting a function called the Lyapunov function that ensures that the trajectories of the system converge to the equilibrium [9], [8], [6]. Computational methods to find Lyapunov functions rely on template based methods that fix a candidate Lyapunov function template such as a polynomial with coefficients as parameters, and use constraint solving and optimization techniques to find the parameters [13], [21], [11], [12]. For hybrid systems, this approach has been extended with the notions of common and multiple Lyapunov functions. The former considers a single function that serves as a Lyapunov function for all the modes of the system, while the latter considers a set of functions, each of which serves as a Lyapunov function for one or more modes, and in addition, satisfies certain conditions related to the mode switchings [10]. Region stability analysis using Lyapunov functions and similar notions such as ranking functions have been considered [4], [5], [15], [16].

Some of the computational difficulties with template based methods include the choice of the right templates [7], a steep increase in the complexity of constraint solving with the increase in the degree of template polynomials as well as numerical issues that arise while solving the constraints/ optimization problems, as illustrated in [19]. Algorithmic approaches based on abstractions that construct weighted graphs for asymptotic stability analysis have gained interest [17], [19]. One can draw an analogue between the abstractions and the templates. The techniques in [17], [18], [19] show that unlike template based methods abstraction based methods return counterexamples that indicate a potential reason for failure of the abstraction for the purpose of establishing stability, and these counterexamples can be used to guide the refinement process. Further, these methods consist of solving simpler feasibility/optimization problems such as linear programming problems as opposed to LMI or SOS

partially supported by EU FP7 Marie Curie Career Integration Grant no. 631622 and NSF CAREER award no. 1552668.

[1]Miriam García Soto is with IMDEA Software Institute, Madrid, Spain miriam.garcia@imdea.org

[2] Pavithra Prabhakar is with the Department of Computer Science, Kansas State University, Manhattan, KS, 66503, USA pprabhakar@ksu.edu

optimization that is required by template based methods. Hence, they provide numerical robustness as illustrated by the experiments in [19]. However, they suffer from the curse of dimensionality and are currently restricted to the class of linear hybrid systems, while template based methods are applicable to nonlinear hybrid systems [13], [7].

In this paper, we build up on the algorithmic approaches in [17], [18], [19], to obtain one for GAS verification. Our main result consists of a decomposition theorem that reduces the GAS verification problem into a AS verification problem and a RS verification problem. More precisely, the GAS verification of a polyhedral hybrid system $\mathcal{H}$, is reduced to checking (1) the asymptotic stability of a system $\mathcal{H}'$ that captures the dynamics of $\mathcal{H}$ close to the origin, and (2) the region stability of $\mathcal{H}$ with respect to a zone Z which ensures that the execution starting from Z remains within the center of $\mathcal{H}$. Step (1) is verified using a previously proposed quantitative predicate abstraction (QPA) method [17]. The weighted graph which exhibits the proof of AS of $\mathcal{H}'$ is used to compute the stability zone Z. A new algorithm for RS verification is provided. Except for the use of QPA for AS verification, all the other algorithms in the paper are new including the computation of Z and the integration of RS and AS verification procedures. As a proof of concept, we apply the GAS verification algorithm to a linear hybrid system model of a cruise control for a automatic gearbox to analyse the convergence of the vehicle velocity to a set velocity. To summarize, the main highlights of our method are:

- A novel algorithmic method for global asymptotic stability analysis that does not rely on Lyapunov function synthesis;
- A graph based proof that provides insights into the GAS, and counter-examples that provide insights into the potential violations in the case that GAS is not established.

II. Preliminaries

Given a function $F : A \to B$ and $C \subseteq A$, let $\mathrm{dom}(F)$ denote the domain of F and $F\lfloor_C$ denote the function with domain C, which maps $c \in C$ to $F(c)$.

a) Convex polyhedral sets: A point in the continuous space $\mathbb{R}^n$ is represented by x which is of the form $(x_1, \ldots, x_n)$. A linear expression over $\mathbb{R}^n$ is a function $p : \mathbb{R}^n \to \mathbb{R}$ where $p(\mathrm{x}) = a\mathrm{x} + b$ with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ being constant values. A linear constraint is an expression of the form $p(x) \sim 0$ where $\sim \in \{<, \leqslant, =\}$. A convex polyhedral set is a convex set P of the form $\{x \in \mathbb{R}^n : p_1(x) \sim 0, \ldots, p_k(x) \sim 0\}$, where $p_1, \ldots, p_k$ are linear expressions. A minimal set of linear constraints that define a convex polyhedral set P is denoted as $\mathcal{LE}(P)$. Given a polyhedral set P, let $\mathring{P}$ denote the interior of P, $\overline{P}$ denote the closure of P and $\partial(P)$ denote the boundary of P. A cone generated by a polyhedral set P is defined as $\mathrm{Cone(P)} = \{\alpha \mathrm{x} \in \mathbb{R}^n : \mathrm{x} \in P \text{ and } \alpha \in \mathbb{R}_{\geqslant 0}\}$. We will use $\mathrm{Poly}(X)$ to denote the set of all convex polyhedral subsets of X, and $\mathrm{CPoly}(X)$ to denote the set of compact convex polyhedral sets.

b) Partition: A polyhedral partition $\mathcal{P}$ of $X \subseteq \mathbb{R}^n$ is a finite set of closed convex polyhedral sets, $\{P_1, \ldots, P_k\}$, such that $X = \cup_{i=1}^{k} P_i$ and $\mathring{P}_i \cap \mathring{P}_j = \emptyset$, for $1 \leqslant i < j \leqslant k$.

A facet of an element P of such partition is a convex polyhedral set determined by the intersection of the set defined by some linear constraint in $\mathcal{LE}(P)$ and $\partial(P)$. The set of all facets of an element P is denoted as $\mathrm{Facets}(P)$.

c) Sequences: A sequence over a set A is a function $s : I \to A$, where I is either $\{1, \cdots, n\}$ for some natural number n or I is $\mathbb{N}$.

d) Graphs and weighted graphs: A graph $\mathcal{G}$ is a pair (V, E), where V is a finite set of vertices and $E \subseteq V \times V$ is a finite set of edges. A path of a graph is a finite or infinite sequence of vertices $\pi = v_0 v_1 \ldots$ such that (v_i, v_{i+1}) is an edge for each $i < \mathrm{len}(\pi)$, where $\mathrm{len}(\pi)$ is the number of nodes in path π. A cycle is a finite path where the first and the last vertices are the same; and it is simple if all the vertices are different except for the first and the last.

A weighted graph $\mathcal{G} = (V, E, \mathrm{W})$ where (V, E) is a graph and $\mathrm{W} : E \to \mathbb{R}_{\geq 0} \cup \{+\infty\}$ is a weighting function on the edges. The weight of a path π is $\Pi_{i<\mathrm{len}(\pi)} \mathrm{W}(v_i, v_{i+1})$.

III. Polyhedral Switched Systems

A well known formalism for modelling systems combining discrete and continuous behaviors is that of hybrid automata [1]. A hybrid automaton consists of a finite state automaton to model the discrete dynamics, and differential equations/inclusions to model the continuous dynamics. Here, we consider an important subclass of hybrid systems called polyhedral switched systems (PSS) that consist of polyhedral inclusions to model the continuous dynamics. These are convenient models for abstractions of complex hybrid systems including those with linear and non-linear continuous dynamics [20], [19].

Definition 1: An n-dimensional PSS is a tuple $\mathcal{H} = (\mathrm{Loc}, \mathrm{Edges}, \mathrm{X}, \mathrm{Flow}, \mathrm{Inv}, \mathrm{Guard})$, where:

- Loc is a finite set of locations;
- Edges $\subseteq$ Loc $\times$ Loc is a finite set of edges;
- $\mathrm{X} = \mathbb{R}^n$ is the continuous state space;
- Flow : Loc $\to$ CPoly(X) is the flow function;
- Inv : Loc $\to$ Poly(X) is the invariant function; and
- Guard : Edges $\to$ Poly(X) is the guard function.

Figure 1a, shows a system that evolves in the 2-dimensional space, that is divided into rectangular polyhedral sets labelled by $q_1, \ldots, q_{10}$. In a region q_i, the system evolves by following the vector F_i depicted inside the region. The system evolution changes the direction when it hits the boundary of the region. A polyhedral switched system capturing the behavior is provided in Figure 2. A regions q_i in Figure 1a correspond to a location q_i, whose invariant $\mathrm{Inv}(q_i)$ is the two

dimensional region it represents, and the flow function $\text{Flow}(q_i) = \{F_i\}$ maps each location with its polyhedral dynamics (represented here by a single vector). There is an edge between two locations if the corresponding regions are adjacent and the flow associated with them are such that they allows the evolution from one region to the other. In this example, we see that the interior of the invariants associated with the locations do not overlap. However, in general, the definition of PSS allows the (interiors of) the invariants to overlap.

An execution of a switched system starts in a mode $q \in \text{Loc}$ and a continuous state $x \in \text{X}$, and consists of a sequence of continuous and discrete evolutions. A continuous evolution corresponds to a trajectory inside $\text{Inv}(q)$ and is such that its derivative belongs to the set $\text{Flow}(q)$ at all times. A discrete evolution consists of a location switch along an edge from some q_1 to q_2, such that the continuous state (which remains unchanged during the switch) satisfies the guard. Formally, the semantics of a PSS $\mathcal{H}$ is given by the set of executions exhibited by the system.

Definition 2: An execution σ of a PSS of dimension n, $\mathcal{H} = (\text{Loc}, \text{Edges}, \text{X}, \text{Flow}, \text{Inv}, \text{Guard})$, is a triple (ι, η, γ) such that:

- ι is a finite or infinite sequence of intervals, such that, for each $i \in \text{dom}(\iota)$, $\iota(i) = [t_i, t_{i+1}]$, $t_0 = 0$ and $t_i \leq t_{i+1}$ and $\mathcal{I}(\iota) = \cup_i \iota(i)$;
- $\eta : \mathcal{I}(\iota) \to \text{X}$ such that for each $\iota(i)$, $\eta \lfloor_{\iota(i)}$ is a differentiable function;
- $\gamma : \text{dom}(\iota) \to \text{Loc}$ such that:
  - for all $i \in I$, for all $t \in \iota(i)$, $\eta(t) \in \text{Inv}(\gamma(i))$ and $\dot{\eta}(t) \in \text{Flow}(\gamma(i))$;
  - for all pairs $i, i+1 \in I$, $(\gamma(i), \gamma(i+1)) \in \text{Edges}$ and $\eta(t_i) \in \text{Guard}((\gamma(i), \gamma(i+1)))$.

A sample execution of the system in Figure 2 is shown in Figure 1a by the blue dotted trajectory. Note that $\text{dom}(\eta)$ may even be a finite interval of the form $[0, t)$. In this case, the interval $\text{dom}(\iota)$ is infinite, and the intervals $[t_i, t_{i+1}]$ do not cover $[0, \infty)$.

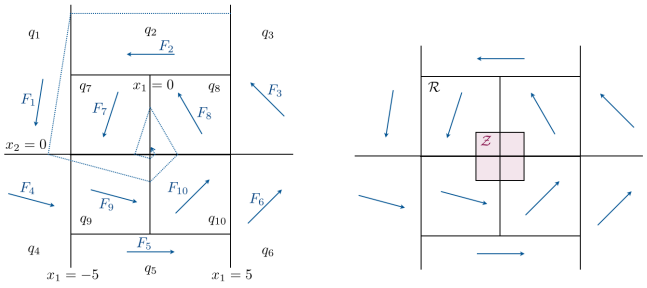
An execution $\sigma = (\iota, \eta, \gamma)$ of $\mathcal{H}$ is said to be complete if $\mathcal{I}(\iota)$ is $[0, \infty)$; otherwise, it is called finite. The set of all executions of $\mathcal{H}$ will be denoted by $\text{Exec}(\mathcal{H})$, and the set of all complete executions by $\text{CExec}(\mathcal{H})$.

An execution $\sigma' = (\iota', \eta', \gamma')$ is said to extend $\sigma = (\iota, \eta, \gamma)$ if $\text{dom}(\iota)$ is a strict subset of $\text{dom}(\iota')$, $\text{dom}(\eta)$ is a strict subset of $\text{dom}(\eta')$, for all $i \in \text{dom}(\iota)$, $\iota(i) = \iota'(i)$ and $\gamma(i) = \gamma'(i)$, and for all $t \in \text{dom}(\eta)$, $\eta(t) = \eta'(t)$. An execution σ is maximal if it cannot be extended. Note that complete executions are maximal.

We define the scaling of an execution by a positive real number.

Definition 3: Given an execution $\sigma = (\iota, \eta, \gamma)$ of $\mathcal{H}$ and a real number $\alpha > 0$, we define $\alpha\sigma$ to be the entity (ι', η', γ'), where:

- $\text{dom}(\iota') = \text{dom}(\iota)$ and if $\iota(i) = [t, t']$, then $\iota'(i) = [\alpha t, \alpha t']$;



(a) Sample execution (b) Stability zone

Fig. 1: Polyhedral switched system

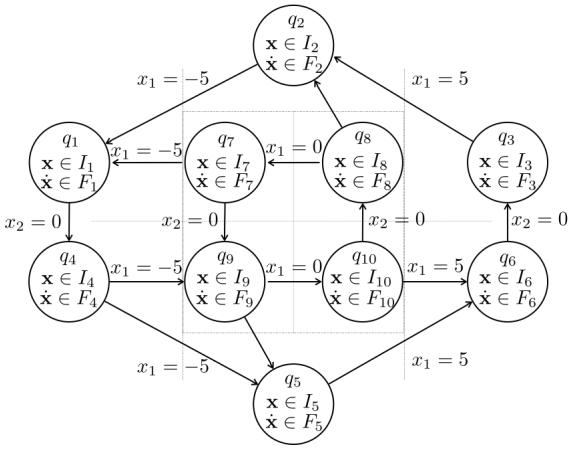


Fig. 2: Hybrid automaton

- $\eta'(t) = \alpha\eta(t/\alpha)$; and
- $\gamma' = \gamma$.

Note that $\alpha\sigma$ may not in general be an execution, since, it may not satisfy the invariant and guard conditions in the definition of an execution. Note, however, that it satisfies the flow conditions, since, $\frac{d}{dt}\eta'(t) = \frac{d}{dt}[\alpha\eta(t/\alpha)] = \alpha\frac{d}{dt}[\eta(t/\alpha)] = \alpha(\frac{d}{dt}\eta)(t/\alpha)\frac{1}{\alpha} = (\frac{d}{dt}\eta)(t/\alpha) \in \text{Flow}(\gamma(i))$, where $t/\alpha \in [t_i, t_{i+1}]$. Then $t \in [\alpha t_i, \alpha t_{i+1}]$ and $\frac{d}{dt}\eta'(t) \in \text{Flow}(\gamma(i))$.

IV. Stability notions

Global asymptotic stability is a desired property in control systems. It assures that all the executions of the system converge to a desired equilibrium state and any perturbations to this state do not drive the system too far. Our main objective in this paper is to analyse global asymptotic stability, however, we will reduce the problem to the verification of other weaker notions of stability. Here, we introduce all these notions.

First, we define a notion of local stability that ensures that small perturbation to an equilibrium state results in only small deviations in the behaviors of the system. We will assume that the equilibrium point is the origin, $\bar{0}$, without loss of generality. Let $B_\epsilon(\bar{0})$ be an open ball around $\bar{0}$ of radius ϵ, that is, $B_\epsilon(\bar{0}) = \{\text{x} : ||\text{x}|| < \epsilon\}$, where $||\text{x}||$ refers to a norm. (Here, we will assume infinity norm, unless otherwise stated).

Definition 4: A PSS $\mathcal{H}$ is said to be Lyapunov stable (LS), if for every $\epsilon > 0$, there exists a $\delta > 0$ such that for every execution $\sigma = (\iota, \eta, \gamma) \in \text{Exec}(\mathcal{H})$ with $\eta(0) \in B_\delta(\bar{0})$, $\eta(t) \in B_\epsilon(\bar{0})$ for every $t \in \mathcal{I}(\iota)$.

First, we remark that Lyapunov stability is a property about a small neighborhood around the origin. Suppose that we show that we can find a δ satisfying the conditions in Lyapunov stability for every ϵ in the interval $[0, \epsilon']$ for a small enough ϵ', then we can find a δ for any ϵ. For $\epsilon > \epsilon'$, we choose a δ that ensures that the trajectories remain within ϵ', which will imply that they remain within ϵ. Hence, we need to focus on a very small neighborhood of the origin to deduce Lyapunov stability.

Definition 5: An execution $\sigma = (\iota, \eta, \gamma) \in \text{Exec}(\mathcal{H})$ is said to be convergent to $\bar{0}$ if for every $\epsilon > 0$ there exists a value $T \geq 0$ such that for every $t \geq T$, $\eta(t) \in B_\epsilon(\bar{0})$.

Definition 6: A PSS $\mathcal{H}$ is said to be asymptotically stable (AS) if it is Lyapunov stable and there exists a value $\zeta > 0$ such that every execution $\sigma = (\iota, \eta, \gamma) \in \text{CExec}(\mathcal{H})$ with $\eta(0) \in B_\zeta(\bar{0})$ is convergent to $\bar{0}$.

We illustrate Lyapunov and asymptotic stability by considering the PSS shown in Figure 1a. First, let us focus on the region $\mathcal{R}$ given by the invariants of q_7, q_8, q_9 and q_{10}. The dynamics are given by $F_7 = (-1, -3), F_8 = (-1, 2), F_9 = (3, -1)$ and $F_{10} = (1, 1)$. Let $2M$ be the length/width of $\mathcal{R}$. Consider an execution which starts at distance $d < M/2$ on the positive x axis, that is, at $(d, 0)$. It will reach the positive y axis at $(0, 2d)$, then the negative x axis at $(-2d/3, 0)$, the negative y axis at $(0, -2d/9)$ and finally return to the negative x axis at $(2d/9, 0)$. Hence, the execution gets closer by a factor of $2/9$ with respect to where it started, and in the meantime never goes beyond a factor of 2 away from the origin. In particular, it never leaves the region $\mathcal{R}$. (A similar observation can be made about the executions starting on the other axes). Hence, in Figure 1a, if we start in the region $\mathcal{R}/2$, which is a box similar to $\mathcal{R}$ centered at the origin, but whose length and width are half that of $\mathcal{R}$, then the executions will remain within $\mathcal{R}$ and eventually get closer to the origin. Hence, given any ϵ' such that $B_{\epsilon'}(\bar{0})$ is contained in $\mathcal{R}$, we can choose $\delta = \epsilon'/2$ and ensure that all executions starting in $B_{\epsilon'}(\bar{0})$ will remain within $B_\delta(\bar{0})$. Therefore, the system is Lyapunov stable. Further, since, the executions get closer by a factor of $2/9$ in between two successive traversal through the positive x axis, we also obtain that they converge, and hence, the system is asymptotically stable.

Next we define global asymptotic stability which unlike asymptotic stability requires convergence to the origin starting from any state.

Definition 7: A PSS $\mathcal{H}$ is said to be globally asymptotically stable (GAS) with respect to $\bar{0}$ if it is Lyapunov stable and every execution $\sigma = (\iota, \eta, \gamma) \in \text{CExec}(\mathcal{H})$ is convergent to $\bar{0}$.

To show that the system in Figure 1a is GAS, we need to show that all the executions of the system will converge to the origin. Note that it does not suffice to show that all the executions that start outside $\mathcal{R}$, enter $\mathcal{R}$. This is because, in concluding asymptotic stability we only showed that the executions starting from a small neighborhood, namely, $\mathcal{R}/2$, around the origin converge to the origin. In fact, the executions that start closer to the boundary of $\mathcal{R}$ may behave very differently from those starting closer to the origin, in that, the former executions may leave the region $\mathcal{R}$. For instance, an execution starting at $(M, 0)$ will not reach the $\mathcal{R}$ on the positive y axis. However, if we show that all executions reach $\mathcal{R}/2$, then we know that they also converge to the origin. This will be the intuition behind our algorithm for GAS verification. Here, we formalize the notion that captures the fact that all executions reach a region.

Definition 8: A PSS $\mathcal{H}$ is said to be region stable (RS) with respect to a set $R \subseteq \text{X}$ if for every maximal execution $\sigma = (\iota, \eta, \gamma) \in \text{Exec}(\mathcal{H})$ there exists a value $T \geq 0$ such that $\eta(T) \in R$.

Often, region stability not only requires that the executions reach R, but also requires the executions to remain within R always after some time [16]. Note that we are considering a weaker notion than the standard region stability because it suffices for our purpose.

In Figure 1a, it can be shown that all executions that start outside the region $\mathcal{R}$ given by q_7, q_8, q_9 and q_{10} will eventually enter $\mathcal{R}$, and hence, the system is region stable with respect to $\mathcal{R}$. Let $F_1 = (-1, -5), F_4 = (2, -1), F_5 = (1, 0), F_6 = (1, 1), F_3 = (-1, 1)$ and $F_2 = (-1, 0)$. Note that any execution that starts at $(-M, -N)$, where $N > M$, will follow F_5, F_6, F_3, F_2 and reach $(-M, N)$. From $(-M, N)$, on following F_1 and F_4, the points $(-M, -N/10)$ is reached. Hence, any execution which starts in q_1, q_2, q_4 and q_5 will eventually enter q_9, and executions that start in F_6 and F_3 will either enter q_8 or reach q_2, and in the latter case reach q_9.

V. GAS verification

In this section, we present an algorithm for global asymptotic stability verification. Our main result is a decomposition theorem that states that global asymptotic stability verification can be reduced to an asymptotic stability verification problem and a region stability verification problem.

As pointed out before, we cannot conclude the GAS of the system in Figure 1a, by showing that it is asymptotically stable, and it is region stable with respect to the region $\mathcal{R}$. This is because asymptotic stability only ensures convergence of executions starting close to the origin, and the executions that enter the region $\mathcal{R}$ from outside may not reach the points close to the origin, and hence, again leave the region. For instance, in Figure 1a, if we start at $(3M/2, 0)$, we reach $(M, M/2)$ by following F_3, and further, reach $(3M/4, M)$ by following F_8, then potentially switch to q_2 and reach q_1 outside of $\mathcal{R}$.

However, if we are able to construct a zone $\mathcal{Z}$ within $\mathcal{R}$ from which all the executions are guaranteed to remain

within $\mathcal{R}$, and have previously established asymptotic stability, then we can conclude that the executions starting from $\mathcal{Z}$ also converge to the origin. Here, we use downward closedness of the executions of the polyhedral switched systems. The intuition is that every execution starting at point z in $\mathcal{Z}$ is similar to an execution that starts at z/α, for any $\alpha \geq 1$. Note that it is crucial that the execution from z remains within $\mathcal{Z}$. We call the zone $\mathcal{Z}$ a stability zone. Global asymptotic stability can then be deduced by showing region stability with respect to $\mathcal{Z}$. Theorem 1 captures this intuition. Below we formalize the concepts required in stating the theorem. Let us fix a polyhedral switched system $\mathcal{H} = (\text{Loc}, \text{Edges}, \text{X}, \text{Flow}, \text{Inv}, \text{Guard})$ for the rest of the section.

A centre zone consists of a set of states of the system adjacent to the origin which contain downward closed subsets of invariants and guards. A set $S \subseteq \text{Loc} \times \text{X}$ is downward closed if for every $(q, x) \in S$, $(q, \alpha x) \in S$ for every $0 < \alpha < 1$.

Definition 9: A centre zone of $\mathcal{H}$ is a downward closed set $\mathcal{R}$ such that

$$\mathcal{R} \subseteq \{(q, x) \in \text{Loc} \times \text{X} : x \in \text{Inv}(q), \bar{0} \in \overline{\text{Inv}(q)}\}$$

and for every $(q, x), (q', x') \in \mathcal{R}$ with $(q, q') \in \text{Edges}$, $\text{Guard}(q, q')$ is downward closed.
Note that if $\bar{0} \in \overline{\text{Inv}(q)}$, then $\text{Inv}(q)$ is necessarily a downward closed set. In Figure 1a, the region $\mathcal{R}$ given by q_7, q_8, q_9 and q_{10} is a centre zone. In the sequel, we use the symbol $\mathcal{R}$ to always refer to the centre zone.

Definition 10: Given a center zone $\mathcal{R}$, a set $\mathcal{Z} \subseteq \mathcal{R}$ is called a stability zone of $\mathcal{H}$ w.r.t $\mathcal{R}$ if for every $\sigma \in \text{CExec}(\mathcal{H})$, $\sigma(t) \in \mathcal{R}$ for every $t \geq 0$.

Theorem 1: Let $\mathcal{H}$ be a PSS, $\mathcal{R}$ be a center zone of $\mathcal{H}$ and $\mathcal{Z}$ be a corresponding stability zone of $\mathcal{H}$. If $\mathcal{H}$ is asymptotically stable with respect to $\bar{0}$ and $\mathcal{H}$ is region stable with respect to $\mathcal{Z}$, then $\mathcal{H}$ is globally asymptotically stable with respect to $\bar{0}$.

Proof: Since $\mathcal{H}$ is asymptotically stable, to show that $\mathcal{H}$ is GAS, we need to show that every complete execution of $\mathcal{H}$ converges to the origin. However, since, we have that all complete executions are maximal, by region stability with respect to $\mathcal{Z}$, they reach $\mathcal{Z}$. Hence, it remains to show that all complete executions that start from $\mathcal{Z}$ converge to the origin. From asymptotic stability of $\mathcal{H}$, we know that there is some $\delta > 0$, such that all executions that start in $B_\delta(\bar{0})$ converge to the origin. Consider an execution σ that starts in $\mathcal{Z}$. There is some $0 < \alpha < 1$ such that $\alpha\sigma$ starts in $B_\delta(\bar{0})$. We note that $\alpha\sigma$ is an execution of $\mathcal{H}$, since, it satisfies the corresponding invariants and guards, which follows from the fact that the invariants and guards restricted to $\mathcal{R}$ are downward closed sets. ■

The technique to analyse GAS of $\mathcal{H}$ with respect to the origin is described in Algorithm 1. Given a PSS $\mathcal{H}$ the GAS verification process returns either a 'yes', 'no' or 'non-established' answer. A 'yes' answer implies GAS of the system $\mathcal{H}$, a 'no' answer implies instability and a 'non-established' answer means that the system $\mathcal{H}$ could not be deduced to be either GAS or otherwise, and hence, may or may not be stable. This verification process consists of the following main steps.

Algorithm 1 Global asymptotic stability verification

Require: $\mathcal{H}$
Ensure: GAS
1: $\text{AS}, \mathcal{G} :=$ AS-verification($\mathcal{H}$)
2: if AS then
3: $\quad \mathcal{R} :=$ Centre($\mathcal{H}$)
4: $\quad \mathcal{Z} :=$ Stability-zone($\mathcal{G}, \mathcal{R}$)
5: $\quad$ return RS-verification($\mathcal{H}, \mathcal{Z}$)
6: else
7: $\quad$ return AS

First, the asymptotic stability of $\mathcal{H}$ is analysed by calling the function AS-verification($\mathcal{H}$). It returns a 'yes', 'no' or 'non-established' answer. A 'yes' answer means that AS was deduced, and in this case a weighted graph $\mathcal{G}$ that serves as the proof of AS is returned. A 'no' answer states that $\mathcal{H}$ is not asymptotically stable and 'non-established' answer implies that AS or otherwise could not be deduced for the system $\mathcal{H}$. These two last answers are accompanied by an empty graph $\mathcal{G}$. In the case that the asymptotic stability of the system $\mathcal{H}$ is deduced, the algorithm proceeds to construct the centre zone $\mathcal{R}$, a stability zone $\mathcal{Z}$ and to verify the region stability of $\mathcal{H}$ with respect to the stability zone $\mathcal{Z}$. A 'no' or 'non-established' answer to AS verification of $\mathcal{H}$ results in the same answer for GAS of $\mathcal{H}$, since global asymptotic stability requires asymptotic stability.

The centre zone of $\mathcal{H}$ is computed by the function Centre($\mathcal{H}$) based on Definition 9. Our stability zone computation algorithm takes as input the centre zone $\mathcal{R}$ as well as the graph $\mathcal{G}$ returned from the asymptotic stability analysis.

Region stability verification is performed by the function RS-verification($\mathcal{H}, \mathcal{Z}$) over the system $\mathcal{H}$ by considering the stability zone $\mathcal{Z}$. Again it returns either 'yes', 'no' or 'non-established'. If region stability of $\mathcal{H}$ is established, then $\mathcal{H}$ is concluded to be GAS in accordance with Theorem 1. If the output to region stability analysis is either 'no' or 'non-established', then in general we cannot conclude that GAS cannot hold. Firstly, we do not have a converse of Theorem 1, and secondly, the region stability could be violated by the existence of maximal finite executions that do not reach the zone, but which can be safely ignored for GAS deduction. Next, we explain the different procedures in detail.

VI. AS verification

In this section, we describe the asymptotic stability verification method for polyhedral switched systems based on abstractions [17]. It is important to understand this construction, since the implementation of our

integration algorithm (Theorem 1) relies crucially on the foundations of this construction.

Observe that since asymptotic stability is a local property, it suffices to restrict the system to a small neighborhood around the origin for the purpose of asymptotic stability analysis. Hence, we define a PSS extracted from $\mathcal{H}$ that uses the dynamics in a small neighborhood for the whole statespace.

Definition 11: Given a PSS $\mathcal{H}$, the extracted PSS from $\mathcal{H}$ is an hybrid automaton $\mathcal{H}^e = (\mathrm{Loc}^e, \mathrm{Edges}^e, \mathrm{X}, \mathrm{Flow}^e, \mathrm{Inv}^e, \mathrm{Guard}^e)$ such that

- $\mathrm{Loc}^e = \{q \in \mathrm{Loc} : \mathcal{R} \cap (q \times \mathrm{Inv}(q)) \neq \emptyset\}$,
- $\mathrm{Edges}^e = \{(q_1, q_2) \in \mathrm{Edges} : q_1, q_2 \in \mathrm{Loc}^e\}$,
- $\mathrm{Flow}^e = \mathrm{Flow}\lfloor_{\mathrm{Loc}^e}$,
- $\mathrm{Inv}^e : \mathrm{Loc}^e \rightarrow \mathrm{Poly}(X)$ with $\mathrm{Inv}^e(q) = \mathrm{Cone}(\mathrm{Inv(q)})$,
- $\mathrm{Guard}^e = \mathrm{Guard}\lfloor_{\mathrm{Edges}^e}$.

Proposition 1: Given a PSS $\mathcal{H}$ and the extracted PSS $\mathcal{H}^e$ from $\mathcal{H}$, $\mathcal{H}$ is asymptotically stable if and only if $\mathcal{H}^e$ is asymptotically stable.

Proposition 1 states that it is enough to analyse AS of the extracted PSS $\mathcal{H}^e$. The full process for AS analysis of system $\mathcal{H}$ is described in Algorithm 2. First, the extracted PSS from $\mathcal{H}$ based on Definition 11 is constructed by the function Extract($\mathcal{H}$). It requires the computation of Cone(P) for a polyhedral set P. It is computed as follows:

$$\mathrm{Cone(P)} = \begin{cases} \bigcap_{\substack{p_i(\mathrm{x}) \sim 0 \in \mathcal{LE}(P) \\ p_i(\bar{0}) = 0}} \{\mathrm{x} \in \mathbb{R}^n : p_i(\mathrm{x}) \sim 0\} & \text{if } \bar{0} \in \partial(P) \\ \mathbb{R}^n & \text{if } \bar{0} \in \mathring{P} \end{cases}$$

Essentially, Cone(P) is represented by the constraints in $\mathcal{LE}(P)$ that contain $\bar{0}$ in their closure. Note that every P corresponds to an invariant whose closure contains the origin by Definition 11.

Next, we need to analyse the asymptotic stability of PSS $\mathcal{H}'$. The broad approach is to construct an abstraction (a simplification) such that certain analysis on the abstract system yields stability of $\mathcal{H}'$. However, standard finite state abstractions do not preserve stability and hence, quantitative abstractions that construct finite weighted graphs have been proposed in [17]. The abstraction procedure is based on a polyhedral partiton $\mathcal{P}$. The nodes in the weighted abstract graph $\mathcal{G}$ correspond to the pairs (q, f), where $q \in \mathrm{Loc}_{\mathcal{H}}$ is a location of the hybrid automaton and $f \in \mathrm{Facets}(\mathcal{P})$ corresponds to the facets of the polyhedral partition. An edge between two nodes, (q_1, f_1) and (q_2, f_2), represents the existence of an execution starting from some state (q_1, x_1) where $\mathrm{x}_1 \in f_1$ that reaches some state (q_2, x_2) with $\mathrm{x}_2 \in f_2$. The weight associated with an edge is an upper bound on the scaling values for all the executions starting from (q_1, f_1) and finishing at (q_2, f_2), that is, it is the maximum ratio $\frac{||\mathrm{x}_2||}{||\mathrm{x}_1||}$ over all pairs $(\mathrm{x}_1, \mathrm{x}_2)$, such that there is an execution from (q_1, x_1) to (q_2, x_2), where $\mathrm{x}_1 \in f_1$ and $\mathrm{x}_2 \in f_2$. The system is deduced to be asymptotically stable, if

Algorithm 2 Asymptotic stability verification

Require: $\mathcal{H}$
Ensure: AS
1: $\mathcal{H}' :=$ Extract($\mathcal{H}$)
2: $\mathcal{P} :=$ Generate-partition($\mathcal{H}'$)
3: $\mathcal{G} :=$ Abstraction($\mathcal{H}', \mathcal{P}$)
4: return AS-satisfaction($\mathcal{G}$)

the following two properties hold of the weight graph $\mathcal{G}$ abstracting the PSS $\mathcal{H}^e$:

*A*1 $\mathrm{W}(e) < +\infty$ for every edge e in $\mathcal{G}$.
*A*2 Every simple cycle π of $\mathcal{G}$ satisfies $\mathrm{W}(\pi) < 1$.

Condition *A*2 that the executions that cross infinitely many facets converge to the origin, since, the scalings associated with such executions is provided by the corresponding paths in the graph. Condition *A*1 ensures that the executions do not diverge while remaining within a single region of the partition.

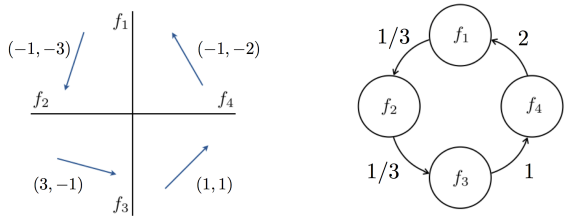
In particular, the product of the weights on a path provide an upper bound on the ratio of the norm of the end to that of the starting point of all executions that follow the sequence of facets specified by the path (see [17] for further details).

Hence, the algorithm proceeds with the generation of the partition using the function Generate-partition($\mathcal{H}'$). The function either automatically generates the partition from the PSS $\mathcal{H}'$ or seeks the user input in computing the partition. The user input can be taken in the form of a set of linear expressions that are then used to construct the partition. A useful set of linear expressions to add either automatically or manually are those that define the invariants and the guards in $\mathcal{H}'$. The PSS $\mathcal{H}'$ and the state space partition $\mathcal{P}$ are the input to the function Abstraction($\mathcal{H}', \mathcal{P}$), which outputs the weighted graph abstraction $\mathcal{G}$. The function AS-satisfaction($\mathcal{G}$) verifies that the Conditions *A*1 and *A*2 hold. The function AS-satisfaction($\mathcal{G}$) returns AS of $\mathcal{H}'$ if both *A*1 and *A*2 are satisfied. If *A*1 is violated, the algorithm will output instability, and if *A*1 is satisfied but *A*2 is not satisfied, the output will be a 'do not know' answer.

Figure 3 shows the quantitative predicate abstraction over the extracted PSS from the one in Figure 1a. It is a simplified version, just to give a flavour of the construction procedure. Observe that every execution starting from f_4 reaching f_1 by following the vector associated with q_8 has as maximum scaling 2/1, that is, 2. The other weights are computed in an analoguous manner.

VII. Stability zone

In this section we introduce the technique to construct a stability zone when given a weighted graph $\mathcal{G}$ abstracting the system $\mathcal{H}'$, and a center zone $\mathcal{R}$. Let $\mathcal{R}$ be a ball of radius M for some M (note that when we use the infinity norm, a ball gives us a rectangular set). The



(a) Extracted PSS (b) Weighted graph

Fig. 3: Quantitative predicate abstraction

detailed algorithm is given in Algorithm 3, in which the stability zone is defined as a scaled version of the centre zone $\mathcal{R}$.

Algorithm 3 Stability zone construction

Require: $\mathcal{G}, \mathcal{R}$
Ensure: $\mathcal{Z}$
1: s := Maximum-scaling$(\mathcal{G}, \mathcal{R})$
2: s′ := $2 \cdot$ s
3: return $\frac{1}{\text{s}'}\mathcal{R}$

First, the maximum weight of a path in $\mathcal{G}$ is computed with the function Maximum-scaling$(\mathcal{G}, \mathcal{R})$, which returns $\max\{1, \max\{\text{W}(\pi) : \pi \text{ is a path in } \mathcal{G}\}\}$. This value represents an upper bound on the variation between the distances to the origin of a pair of states belonging to the same execution of the system $\mathcal{H}'$ whose abstraction is represented by $\mathcal{G}$. Denote this bound as s and observe that an execution in $\mathcal{H}'$ starting at a distance d from $\bar{0}$ can only reach as far as sd from $\bar{0}$. The stability zone $\mathcal{Z}$ is taken to be $\frac{1}{\text{s}'}\mathcal{R}$ for some s′ > s. Algorithm 3 considers s′ to be 2s. Observe that in Figure 1 the stability zone $\mathcal{Z}$ coincides with a scaled version of $\mathcal{R}$.

VIII. RS verification

In this section, we introduce our abstraction based algorithm for region stability analysis. Given a PSS $\mathcal{H}$ and a stability zone $\mathcal{Z}$, region stability of $\mathcal{H}$ with respect to $\mathcal{Z}$ is verified. An algorithmic procedure for RS verification is shown in Algorithm 4. It consists of the following steps:

Initially, a state partition $\mathcal{P}$ is generated as explained in the previous section, by the function Generate-partition$(\mathcal{H})$. This partition $\mathcal{P}$ is refined into a partition Partition$(\mathcal{P}, \mathcal{Z})$ using the stability zone $\mathcal{Z}$. If $\mathcal{P} = \{P_1, \ldots, P_k\}$, then $\mathcal{P}'$= Partition$(\mathcal{P}, \mathcal{Z})$ is defined as $\{P_1 \setminus \mathcal{Z}, \ldots, P_k \setminus \mathcal{Z}, P_1 \cap \mathcal{Z}, \ldots, P_k \cap \mathcal{Z}\}$. The new partition ensures that the zone $\mathcal{Z}$ is a union of certain regions of the partition, and hence, the boundary of $\mathcal{Z}$ can be captured by the facets of the partition. Let us call these facets $F_{\mathcal{Z}}$.

The PSS $\mathcal{H}$ and the state space partition $\mathcal{P}'$ are input to Abstraction$(\mathcal{H}, \mathcal{P}')$, which constructs a weighted graph $\mathcal{G}1$ by applying the abstraction over the system

Algorithm 4 Region stability verification

Require: $\mathcal{H}, \mathcal{Z}$
Ensure: RS
1: $\mathcal{P}$:= Generate-partition$(\mathcal{H})$
2: $\mathcal{P}'$:= Partition$(\mathcal{P}, \mathcal{Z})$
3: $\mathcal{G}1$:= Abstraction$(\mathcal{H}, \mathcal{P}')$
4: $\mathcal{G}2$:= Delete-inner-nodes$(\mathcal{G}1)$
5: $\mathcal{G}3$:= Delete-non-reachable-nodes$(\mathcal{G}2)$
6: if edge in $\mathcal{G}3$ with W$(edge) = \infty$ then
7: return false
8: $\mathcal{C}$:= Cycles$(\mathcal{G}3)$
9: if $\mathcal{C}$ not empty then
10: return non-established
11: $\mathcal{N}2$:= Not-outgoing-edges$(\mathcal{G}3)$
12: $\mathcal{N}3$:= Nodes$(\mathcal{G}3, \text{Facets}(\mathcal{Z}))$
13: if node in $\mathcal{N}2$ and not in $\mathcal{N}3$ then
14: return non-established
15: return true

$\mathcal{H}$ explained in Section VI. This is slightly modified to capture those executions which remain within a region (may not necessarily diverge), such executions are captured using infinite weight edges as well. In order to establish region stability, we need to show that all executions that start outside the zone $\mathcal{Z}$ will eventually reach its boundary. Hence, we can ignore the subgraph that corresponds to the interior of $\mathcal{Z}$. This is achieved by the function Delete-inner-nodes$(\mathcal{G}1, \mathcal{Z})$ that deletes all nodes that correspond to the interior of $\mathcal{Z}$ and the edges out of them, and outputs $\mathcal{G}2$. Then we remove all nodes that are not reachable from the initial nodes which capture all the points from which we want to reach $\mathcal{Z}$. Delete-non-reachable-nodes$(\mathcal{G}2, \mathcal{N}1)$ removes all non-reachable nodes resulting in the graph $\mathcal{G}3$. From the modified graph construction, this also detects any executions that remain in a region and do not move to the next facet. Next, we ensure that there are no executions that diverge while remaining in a single region, this is ensured by checking that there are no edges with infinite weight in $\mathcal{G}3$. Note that a cycle can represent an infinite execution that follows the cycle infinitely many times and hence, does not reach $\mathcal{Z}$. The function Cycles$(\mathcal{G}3)$ returns the simple cycles in the graph; if it is empty, we proceed with the analysis. Finally, we need to ensure that none of the executions get stuck (terminate) at any of facets. The function Not-outgoing-edges$(\mathcal{G})$ computes all the nodes which have outgoing edges. The only nodes that are allowed not to be in this set are those corresponding to $\mathcal{Z}_F$.

IX. Extension to other dynamics

In this paper, we focused on the analysis of global asymptotic stability of polyhedral switched systems. Theorem 1 provides a decomposition theorem for GAS verification in terms of asymptotic stability and region stability verification. It uses certain properties about

the polyhedral inclusion dynamics, namely, that the solutions are scaling invariant in the sense that if σ is a solution of a polyhedral inclusion dynamics specified by the polyhedron P, then $\alpha\sigma$ is a solution as well. A similar property can be established for linear dynamics, however, here, in the definition of $\alpha\sigma$ we do not need to scale the time. More precisely, if $x(t)$ is a solution of $\dot{x} = Ax$, then $y(t)$ given by $\alpha x(t)$ is also its solution. Hence, the decomposition theorem can be extended to linear hybrid systems. However, such properties do not hold for general non-linear dynamics.

An alternate approach to deal with the general class of hybrid systems is to use hybridization to reduce them to polyhedral hybrid systems [20], [2], [3]. Hybridization constructs a partition of the statespace into a finite number of regions and over-approximates a given dynamics by a simpler one. Hybridization results in an over-approximate system and hence, it suffices to establish the GAS of the over-approximate system to deduce the same for the given system. We use the hybridization approach in the sequel to analyse a linear hybrid system modeling an automatic gearbox; we construct a polyhedral switched system that over-approximates the linear hybrid system and analyse the PSS for GAS.

X. Implementation

The purpose of this section is to show the details of the implementation for the GAS verification procedure, by enumerating the different functions, the software used, the implemented algorithms and the manually performed instructions. An overall flow of the process is shown in Figure 4. The input to the process is an hybrid automaton $\mathcal{H}$, which is analysed to deduce global asymptotic stability with respect to the origin. The output is a yes (True) or no (False) or unknown answer to the stability analysis. The input automaton can be a PSS or a linear hybrid automaton(LHA). In the case that the input $\mathcal{H}$ is a LHA, a polyhedral switched system $\mathcal{H}'$ is constructed by applying a hybridization technique over the linear switched systems, which is described in [19]. This hybridization is implemented in the tool Averist and requires the LHA $\mathcal{H}$ and a set of predicates for partitioning the state-space. Those predicates can be generated automatically by the tool or provided by the user. The hybridization construction ensures that if the system $\mathcal{H}'$ is GAS, then $\mathcal{H}$ is GAS as well. Then the PSS is analysed for asymptotic stability by constructing the abstract weighted graph $\mathcal{G}$ and model-checking it. This is again performed by Averist. In the case of 'yes' answer an abstract weighted graph $\mathcal{G}$ is returned. This weighted graph is the input to the stability zone construction. The stability zone construction comprises of three different computations — maximum weight computation over all the paths in the graph $\mathcal{G}$, the centre region construction and finally the stability zone construction. The maximum weight computation in $\mathcal{G}$ is implemented in Python 2.7. It transform the maximum product weight computation

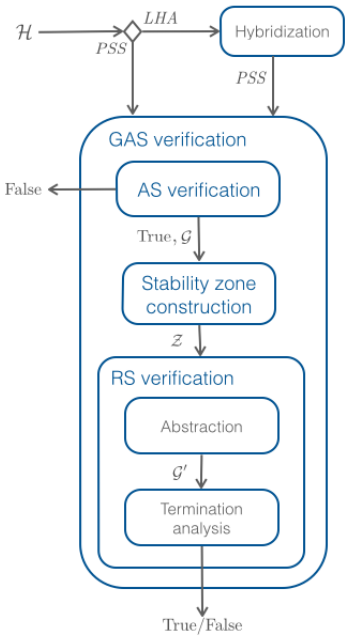


Fig. 4: GAS flow diagram

into a minimum sum weight computation by considering the $-log_{10}$ of all the weights in the graph, which is then solved by Dijkstra's algorithm included in the NetworkX Python package. The centre region construction at this point is performed manually. The values for the linear expressions delimiting the stability zone are computed automatically in Python 2.7, but the obtained linear expressions are added manually to perform the region stability verification.

This construction of the abstract graph in stability zone verification is done by calling Averist. The termination analysis is fully automated and uses different algorithms included in the NetworkX Python package for the analysis with graphs.

XI. An automatic gearbox

In this section we present an automatic gearbox example and analyse global asymptotic stability. The gearbox example is a linear hybrid automaton. We construct a polyhedral hybrid automaton and then apply our GAS verification algorithm on that. We illustrate the overall process through the automatic gearbox model presented in [14].

A. Model

An automatic gearbox is a device for controlling the gear changes in a vehicle. It consists of four different positions and at each position the dynamics is governed by a linear system. The dynamics describe the evolution of two variables, v and T_I, where v is the difference between the desired velocity v_d and the current velocity v_c and T_I corresponds to the integrator torque state $\Delta\mathrm{T}_I$. Every switching is performed when some constraint

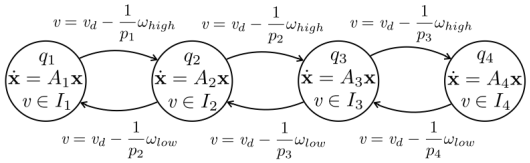


Fig. 5: Model of the gearbox

over the velocity is satisfied. The target is to achieve a desired velocity v_d, for which we need to verify global asymptotic stability of the system with respect to $\mathrm{v} = 0$ and $\mathrm{T}_I = 0$. The value T_I is set to 0 every time the driver inputs a new desired velocity.

The linear switched system modelling the system is shown in Figure 5, where the state is defined by $\mathrm{x} = (\mathrm{v}, \mathrm{T}_I)^\top$ and the closed-loop dynamics are determined at each gear by the matrix

$$A_q = \begin{pmatrix} -p_q k_q/M & -p_q/M \\ k_q/T & 0 \end{pmatrix}$$

with q referring to each of the modes q_1, q_2, q_3 and q_4. The invariant intervals are $I_1 = [\mathrm{v}_d - \frac{1}{p_1}\omega_{high}, \infty)$, $I_2 = [\mathrm{v}_d - \frac{1}{p_2}\omega_{low}, \mathrm{v}_d - \frac{1}{p_2}\omega_{high}]$, $I_3 = [\mathrm{v}_d - \frac{1}{p_3}\omega_{low}, \mathrm{v}_d - \frac{1}{p_3}\omega_{high}]$ and $I_4 = (-\infty, \mathrm{v}_d - \frac{1}{p_4}\omega_{low}]$. The values for each parameter in the linear switched system definition appear in Table I.

TABLE I: Parameter values

$M = 1500$		$T = 40$	
$\omega_{low} = 500$		$\omega_{high} = 230$	
$p_1 = 50$	$p_2 = 32$	$p_3 = 20$	$p_4 = 14$
$k_1 = 15/4$	$k_2 = 375/64$	$k_3 = 75/8$	$k_4 = 375/28$

Figure 6 shows some representative sample executions of the automatic gearbox for $\mathrm{v}_d = 30$. The horizontal axis represents the velocity difference v and the vertical axis represents the integrator torque T_I. Since the value T_I is set to 0 every time the driver inputs a new desired velocity, every execution starts at the horizontal axis. The initial state depends on the current velocity of the vehicle. The origin is the state such that $\mathrm{v}_d = \mathrm{v}_c$, that is, the state where the target velocity is achieved. An execution starting from the negative side of the axis, that is $\mathrm{v} < 0$, means that the vehicle needs to slow down, because the current velocity is higher than v_d. The gear is in the 4th position, which corresponds to mode q_4. Observe that the execution evolves continuously in location q_4 until reaching the origin. In case of starting from a positive value of v, the current velocity is lower than v_d and the gear position will depend on this value. We depicted two different executions, one starting from location q_1 and another one starting from location q_2. The execution starting from q_1 switches three times. The current velocity increases continuously in q_1 until reaching the value $\frac{1}{p_1}\omega_{high}$ (involved in the guard constraint from q_1 to q_2 in Figure 5), represented by the rightmost solid red line. Then, it switches from location q_1 to location q_2. The switching implies a dynamics change but no jump on the state. The execution evolves by following the new dynamics until reaching a new value $\frac{1}{p_2}\omega_{high}$ for the current velocity and it switches from location q_2 to location q_3. The next switching is from location q_3 to q_4, where the execution converges to the origin. An analogous behavior is observed in the case of the execution starting from location q_2.

B. Analysis

Let us consider the gear box hybrid automaton in Figure 5 where we set the desired velocity v_d to be 30. Let the corresponding linear switched system be $\mathcal{H}_{gb}$. Next, we explain the different GAS verification steps performed for the linear switched system $\mathcal{H}_{gb}$.

Step 1.- A new system description of $\mathcal{H}_{gb}$ in which all the constants are integers is performed. This is necessary since the different tools we are using require such property. Every polyhedral set in PPL needs to be defined by linear expressions with integer parameters. Therefore, all the constant values in $\mathcal{H}_{gb}$ are expressed as rational numbers and after multiplied by 1344000, which is the least common multiple of all the denominator values.

Step 2.- A polyhedral switched system $\mathcal{H}_1$ is constructed by applying a hybridization technique over the linear switched system $\mathcal{H}_{gb}$. The hybridization technique returns in addition to $\mathcal{H}_1$, the polyhedral state space partition used for construction. Let us denote the partition as $\mathcal{P}$. It is the one shown in Figure 6, where the state space is divided by the linear constraints which describe the invariants of the original gearbox, the linear constraint $\mathrm{T}_I = 0$ which defines the initial states and the linear equation $375\mathrm{v} + 28\mathrm{T}_I = 0$ which equalizes to zero the velocity coordinate of the flow vectors defining executions close to the origin.

Step 3.- From the PSS $\mathcal{H}_1$ we extract a PSS $\mathcal{H}_2$ which consists of only one location, where the invariant is the full planar continuous space and the flow is defined by the polyhedral approximation of the linear dynamics in the q_4 mode of the original system. Then asymptotic stability proof reduces to that of a single mode system.

Step 4.- Asymptotic stability of $\mathcal{H}_2$ is analysed by using the tool Averist. The hybrid system $\mathcal{H}_2$ and the state space partition $\mathcal{P}$ are the inputs to the software, which returns an answer and a weighted abstract graph $\mathcal{G}$. We obtain a positive answer about asymptotic stability. The abstraction process results in the weighted graph shown in Figure 7, where the weights appear truncated for simplicity.

Since the asymptotic stability proof consists of a proof for a single mode system, the asymptotic stability could have been verified by eigenvalue analysis over the linear dynamics in q_4, so over a linear version of the extracted $\mathcal{H}_2$ shown in the left side of Figure 7, but we require extra information for the stability zone construction which is

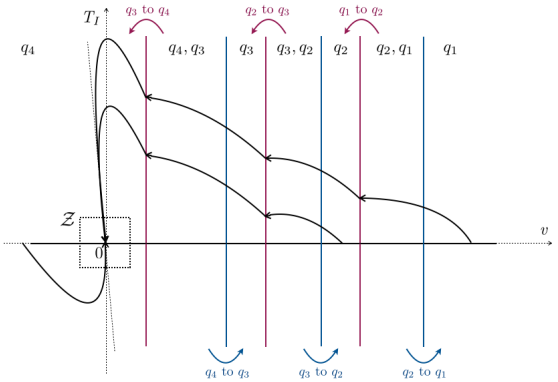


Fig. 6: Partition and sample executions

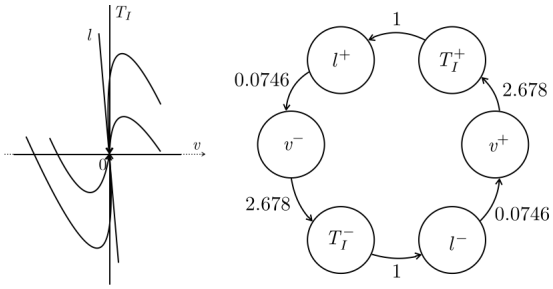


Fig. 7: Asymptotic stability verification

provided by the abstract weighted graph $\mathcal{G}$, shown on the right in Figure 7

Step 4.- The stability zone $\mathcal{Z}$ is computed. The center zone is $q_4 \times ((-\infty, 18240000] \times \mathbb{R})$ that is shown in Figure 6. The extracted system that is used in the asymptotic stability analysis and subsequently in the computation of $\mathcal{Z}$, is depicted on the left in Figure 7. The abstract weighted graph $\mathcal{G}$ is shown on the right in Figure 7. Observe that in Figure 7 the maximum value of any path in the graph is in fact the value of the edge with weight 2.678. We denote this weight as s. Then, the stability zone defined as in Algorithm 3 corresponds to $\alpha((-\infty, 18240000] \times \mathbb{R})$ where $\alpha \simeq 0.1867$. We are going to be more restrictive and instead of considering such stability region, we construct a smaller one, contained in the previous one. This new region is a ball centered on the origin with radius 1250000 (this is a not automatic choice), which is less than $\alpha \cdot 18240000 = 3405408$. Let us denote this ball as $\mathcal{Z}$. The predicates defining the ball $\mathcal{Z}$ are added manually in the implementation.

Step 5.- A weighted graph is automatically constructed to verify region stability. A new partition $\mathcal{P}'$ is automatically constructed by intersecting the polyhedral elements in $\mathcal{P}$ with $\mathcal{Z}$. Then, the new state space partition $\mathcal{P}'$ is input to perform a predicate abstraction over the $\mathcal{H}_1$ as explained in Section VI. A weighted graph $\mathcal{G}'$ is returned. Note that we do not use the weight of the graph $\mathcal{G}'$ except to deduce that all the executions move to the next facet. If there exists an infinite weight, then $\mathcal{H}_1$ is not region stable with respect to $\mathcal{Z}$. The nodes in $\mathcal{G}'$ corresponding to the interior of the ball $\mathcal{Z}$ are automatically removed from the graph $\mathcal{G}'$, since we just need to check that every path reaches some of the nodes referring to the facets of $\mathcal{Z}$. Recall that we restrict ourselves, in the system $\mathcal{H}$, to executions with initial value of $\mathrm{T}_I = 0$. Therefore we consider the nodes (q, f) in $\mathcal{G}'$ such that $f \cap \{\mathrm{T}_I = 0\} \neq \emptyset$ as initial nodes, which are selected automatically. All the non reachable nodes are automatically removed from the initial nodes in $\mathcal{G}'$, since they are not necessary for our analysis and the simpler the graph becomes the less the toperational time required.

Finally, the graph $\mathcal{G}'$ is analyzed for termination. First, a search for cycles in $\mathcal{G}'$ is performed and it returns negative. Finally, the nodes in $\mathcal{G}'$ with no outgoing edges is computed, and it is verified that these correspond only to the nodes representing Facets($\mathcal{Z}$). Hence, the implementation outputs that $\mathcal{H}$ is region stable with respect to $\mathcal{Z}$, and therefore, GAS is stated for the automatic gearbox instance we are considering.

XII. Conclusion

In this paper, we present an algorithmic approach to global asymptotic stability verificatoin that relies on a decomposition of the global stability into a local asymptotic stability verification problem and a region stability problem. We apply our algorithm to a case study involving a cruise control interacting with an automatic gearbox. In the future, we will develop computational methods for applying these techniques to the class of linear and non-linear switched systems.

References

[1] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Hybrid Systems, pages 209–229, 1992.

[2] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. Acta Informatica, 43(7):451–476, 2007.

[3] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In Proceedings of the International Conference on Hybrid Systems: Computation and Control, pages 11–20, 2010.

[4] Parasara Sridhar Duggirala and Sayan Mitra. Abstraction refinement for stability. In Proceedings of the International Conference on Cyber-Physical Systems, pages 22–31, 2011.

[5] 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.

[6] R. Goebel, R.G. Sanfelice, and A.R. Teel. Hybrid dynamical systems. IEEE Control Systems Magazine, 29:28–93, 2009.

[7] James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided lyapunov analysis for hybrid dynamical systems. In Proceedings of the International Conference on Hybrid Systems: Computation and Control, pages 133–142, 2014.

[8] H. K. Khalil. Nonlinear Systems. Prentice-Hall, Upper Saddle River, NJ, 1996.

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

[10] Hai Lin and Panos J. Antsaklis. Stability and stabilizability of switched linear systems: A survey of recent results. IEEE Transactions on Automatic Control, 54(2):308–322, 2009.

[11] Paolo Mason, Ugo V. Boscain, and Yacine Chitour. Common polynomial lyapunov functions for linear switched systems. SIAM J. Control and Optimization, 45(1):226–245, 2006.
[12] Eike Möhlmann and Oliver E. Theel. Stabhyli: a tool for automatic stability verification of non-linear hybrid systems. In Proceedings of the International Conference on Hybrid Systems: Computation and Control, pages 107–112, 2013.
[13] Antonis Papachristodoulou and Stephen Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In Conference on Decision and Control, 2002.
[14] Stefan Pettersson and Bengt Lennartson. Stability of hybrid systems using lmis - A gear-box application. In Proceedings of the International Conference on Hybrid Systems: Computation and Control, pages 381–395, 2000.
[15] 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.
[16] Andreas Podelski and Silke Wagner. Region stability proofs for hybrid systems. In Proceedings of Formal Modeling and Analysis of Timed Systems, pages 320–335, 2007.
[17] Pavithra Prabhakar and Miriam García Soto. Abstraction based model-checking of stability of hybrid systems. In Proceedings of the International Conference on Computer Aided Verification, pages 280–295, 2013.
[18] Pavithra Prabhakar and Miriam Garcia Soto. Counterexample guided abstraction refinement for stability analysis. In Proceedings of the International Conference on Computer Aided Verification, 2016.
[19] Pavithra Prabhakar and Miriam García Soto. Hybridization for stability analysis of switched linear systems. 2016.
[20] 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.
[21] Weehong Tan and Andrew K. Packard. Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming. IEEE Transactions on Automatic Control, 53(2):565–571, 2008.