From Non-Zenoness Verification to Termination

Pierre Ganty[1], Samir Genaim[2], Ratan Lal[1], and Pavithra Prabhakar[1]

[1] IMDEA Software Institute, Madrid, Spain
[2] Universidad Complutense de Madrid, Spain

***Abstract*—We investigate the problem of verifying the absence of zeno executions in a hybrid system. A zeno execution is one in which there are infinitely many discrete transitions in a finite time interval. The presence of zeno executions poses challenges towards implementation and analysis of hybrid control systems. We present a simple transformation of the hybrid system which reduces the non-zenoness verification problem to the termination verification problem, that is, the original system has no zeno executions if and only if the transformed system has no non-terminating executions. This provides both theoretical insights and practical techniques for non-zenoness verification. Further, it also provides techniques for isolating parts of the hybrid system and its initial states which do not exhibit zeno executions. We illustrate the feasibility of our approach by applying it on hybrid system examples.**

Keywords*—*Hybrid System, Non-Zenoness, Verification

I. Introduction

The ubiquitous deployment of embedded control systems has motivated the study of hybrid systems – systems which exhibit both discrete and continuous behaviors. Hybrid automata [4], [18] are a popular formal model for hybrid systems, which combine the finite state automaton model for discrete systems and the differential equation/inclusion formalism for continuously evolving dynamical systems. The interactions between the discrete and continuous components gives rise to a class of behaviors which are unique to hybrid systems, namely, *zeno*. A zeno execution of a hybrid system is one in which there are infinitely many discrete transitions in a finite interval of time.

In this paper, we present automated techniques for detecting the absence of zeno behaviors in a hybrid system. Zeno behaviors have been studied extensively in the hybrid systems community owing to the fact that they present technical challenges in the design, analysis and implementability of hybrid control systems. Below we highlight some of them.

In the logic design of switched systems, the absence of zeno behaviors – also referred to as "chattering" or "sliding mode" – is desirable, as they allow for a mathematically simpler definition of solutions or executions. Hence, this is often an objective of the control design [20], [29]. Also, several analysis techniques assume the absence of zeno-behaviors due to lack of methods to deal with it. For instance, Duggirala and Mitra [13] assume "non-blocking", same as absence of zeno executions, to establish the soundness of the region stability analysis.

Further, the presence of zeno behaviors poses a challenge in the simulation of hybrid systems, e.g., it affects the efficiency and accuracy of simulations [22]. For instance, failure to simulate beyond the limit time could end up in an erroneous analysis. Efforts have been made to develop methods to "remove" zenoness. For instance, regularizations [23], [14] attempt to construct a sequence of non-zeno hybrid system approximations which converge to the given hybrid system.

Finally, the presence of zeno executions raises concerns regarding the implementability of the controller. Typically, a weaker notion – the absence of time-locks – is considered as a requirement for implementability [18], [31]. A state is said to be in a *time-lock* if all executions starting from it are zeno executions. Note that absence of zeno executions imply absence of time-locks. Though our notion is stronger, we believe that absence of zeno executions is a reasonable assumption for the implementability of the controller, since, their presence implies that the controller cannot be realized exactly.

Given the importance of zeno in hybrid systems design and analysis, there has been substantial effort towards understanding zeno behaviors. Several results in the literature focus on deducing necessary and/or sufficient conditions for the existence/non-existence of zeno executions. Zhang *et al.* [22] provide sufficient conditions for the non-existence of zeno behaviors; Camlibel *et al.* [9], [30] show that certain subclasses of switched linear systems do not exhibit zeno-behaviors; and Lamperski and Ames [24] provide Lyapunov like-sufficient conditions for the existence of zeno-behaviors. Non-zenoness verification is shown to be decidable for timed petri nets and timed pushdown automata [1], [17], [2]. Heymann *et al.* [21] provide sufficient conditions for analyzing non-zenoness of hybrid systems with constant rate and bounded rate dynamics.

In this paper, we present a simple, yet powerful, transformation which reduces the non-zenoness verification problem to the termination problem, that is, a given hybrid system does not have any zeno execution if and only if the transformed system does not have any non-terminating execution. Essentially, we introduce an extra variable, which is non-deterministically initialized to some value a in the interval $[0, \infty)$ and decreases at a constant rate of -1 until its value becomes 0. Hence, every execution of the original system is constrained to an interval $[0, a]$ in the transformed system, for some a. A non-terminating execution in the transformed system corresponds to a zeno execution of the original system, since such an execution exhibits an infinite number of discrete transitions in some finite interval $[0, a]$. On the other hand, if the original system has a zeno execution with a limit point $< a$, then the execution in the transformed system which starts with the extra variable value a is non-terminating. This transformation provides new theoretical insights, but, importantly, opens up a plethora of techniques from termination analysis [10], [3], [7], [8], [11], [16], [25].

Below we summarize the main results of the paper. From a theoretical point of view, we explore the boundary of

decidability for non-zenoness verification. We show that non-zenoness verification problem is undecidable for the class of multi-rate hybrid automata, whereas it is decidable for *initialized* multi-rate hybrid automata, which includes the class of timed automata. The latter is a new result which relies on the transformation into a termination problem, and our approach provides a new algorithm for deciding non-zenoness for the class of timed automata. From a practical point of view, we use the transformation to establish the following: (a) non-zenoness of hybrid systems, and (b) identify initial conditions which exclude zeno executions. This transformation is general and can be applied to any class of hybrid systems for which a "reachability relation" can be computed. We observe that our method for non-zenoness verification is sound for an over-approximation of the reachability relation, and hence, can be applied to a large class of systems. We demonstrate the applicability of the approach on hybrid system examples.

II. Preliminaries

We introduce some notations and concepts that we will use in the later sections.

A. Notation

a) Numbers: Let $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ respectively denote the sets of real numbers and non-negative real numbers. Let $[n]$ denote the set of n natural numbers given by $\{1, 2, \ldots, n\}$. Let $\mathcal{I}$ denote the set of intervals of the form $[a, b]$, $(-\infty, b]$ and $[a, \infty)$, where a and b are real numbers.

b) Euclidean space $\mathbb{R}^n$: Given $\vec{v} \in \mathbb{R}^n$ and $\vec{a} \in \mathbb{R}^n$, we use $(\vec{v})_i$ to denote the i-th component of $\vec{v}$, and $\vec{v} \times \vec{a}$ to denote the vector $((\vec{v})_1, \ldots, (\vec{v})_n, \vec{a})$. Given $f_1 : [0, t] \to \mathbb{R}^n$ and $f_2 : [0, t] \to \mathbb{R}$, let $f_1 \circ f_2 : [0, t] \to \mathbb{R}^{n+1}$ represent the function $t' \mapsto f_1(t') \circ f_2(t')$. Given sets $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}$, let $A \times B = \{a \times b \,|\, a \in A, b \in B\}$.

c) Rectangular sets: A n-dimensional rectangular set S is of the form $I_1 \times I_2 \times \ldots \times I_n$, where $I_i \in \mathcal{I}$, for $1 \leq i \leq n$. Given a rectangular set $S \subseteq \mathbb{R}^n$, let $\lfloor S \rfloor$ and $\lceil S \rceil$ denote the least upper bound and the greatest lower bound of S in $(\mathbb{R} \cup \{-\infty, +\infty\})^n$. Note that it always exists.

d) Trajectories: They will be used to represent the continuous dynamics. A trajectory is a function from an interval $[0, t]$ to a Euclidean space $\mathbb{R}^n$, where $[0, t]$ represents the time domain of the function. Let *Traj*(n) denote the set of all differentiable functions $f : [0, t] \to \mathbb{R}^n$, for some t. Let *dom*(f) denote the domain of f.

B. Timed transitions systems

Next, we introduce a semantic model of hybrid systems, namely, timed transition systems, and define the concepts of non-zenoness and termination.

Definition 1. *A* timed transition system *is a tuple $\mathcal{T} = (\mathcal{S}, \mathcal{S}_0, \Sigma, \to)$, where:*

1) *$\mathcal{S}$ is a set of states;*
2) *$\mathcal{S}_0 \subseteq \mathcal{S}$ is a set of initial states;*
3) *Σ is a set of action symbols; and*
4) *$\to \subseteq \mathcal{S} \times \mathbb{R}_{\geq 0} \times \Sigma \times \mathcal{S}$ is a set of timed transitions.*

From now on, by a transition system, we mean a timed transition system. We will use $s \overset{t,a}{\to} s'$ to denote $(s, t, a, s') \in \to$. Also, we will use $\to_{\mathcal{T}}$ to denote the set of timed transitions of a transition system $\mathcal{T}$.

An *execution* of a transition system $\mathcal{T} = (\mathcal{S}, \mathcal{S}_0, \Sigma, \to)$ is a finite or infinite sequence $\sigma = s_0(t_0, a_0)s_1(t_1, a_1)s_2 \ldots$, such that $s_0 \in \mathcal{S}_0$ and $s_i \overset{t_i,a_i}{\to} s_{i+1}$ holds for every i. The execution σ is said to be *finite* if σ is a finite sequence; otherwise, it is *infinite*. The execution σ is *zeno* if it is infinite and $\sum_{i=0}^{\infty} t_i \in \mathbb{R}_{\geq 0}$, that is, the sum of times is finite; otherwise, it is *non-zeno*.

Definition 2. *A transition system $\mathcal{T}$ is said to be* terminating *if none of its executions are infinite; $\mathcal{T}$ is said to be* non-zeno *if none of its executions are zeno.*

Next, we present a transformation of a transition system such that the non-zenoness of the first is equivalent to the termination of the second. We will later present a concrete method to realize this transformation for transition systems which arise from hybrid systems.

Definition 3. *Given a transition system $\mathcal{T} = (\mathcal{S}, \mathcal{S}_0, \Sigma, \to)$, we define a transformed transition system term$(\mathcal{T})$ $= (\mathcal{S}', \mathcal{S}'_0, \Sigma, \to')$ where*

1) *$\mathcal{S}' = \mathcal{S} \times \mathbb{R}_{\geq 0}$;*
2) *$\mathcal{S}'_0 = \mathcal{S}_0 \times \mathbb{R}_{\geq 0}$; and*
3) *$\to' = \{((s_1, \tau_1), t, a, (s_2, \tau_2)) \mid (s_1, \tau_1), (s_2, \tau_2) \in S', s_1 \overset{t,a}{\to} s_2, \tau_2 = \tau_1 - t\}$.*

The above transformation basically augments the state with an extra non-negative component, such that in each step it is decremented by the time t used when moving from the source to the target state in that step. The next theorem relates the zenoness of $\mathcal{T}$ to the termination of *term*$(\mathcal{T})$.

Proposition 1. *$\mathcal{T}$ is non-zeno if and only if term$(\mathcal{T})$ is terminating.*

Proof: ($\Rightarrow$) Suppose *term*$(\mathcal{T})$ is not terminating. Then there exists an infinite execution *term*$(\sigma) = (s_0, \tau_0)(t_0, a_0)(s_1, \tau_1)(t_1, a_1)(s_2, \tau_2) \ldots$ of *term*$(\mathcal{T})$. Note that $t_i = \tau_i - \tau_{i+1}$, by definition of *term*$(\mathcal{T})$. Therefore, $\sum_{i=0}^{\infty} t_i = \sum_{i=0}^{\infty}(\tau_i - \tau_{i+1}) = \lim_{k\to\infty} \sum_{i=0}^{k}(\tau_i - \tau_{i+1}) = \lim_{k\to\infty}(\tau_0 - \tau_{k+1}) = \tau_0 - \lim_{k\to\infty} \tau_{k+1}$. Since $t_i \geq 0$ and $\tau_i \geq 0$, we have $\tau_0 \geq \tau_1 \geq \ldots \geq 0$. Therefore, $\tau_0 \geq \lim_{k\to\infty} \tau_{k+1} \geq 0$. Hence, $\sum_{i=0}^{\infty} t_i \in [0, \tau_0]$. Therefore, $\sigma = s_0(t_0, a_0)s_1(t_1, a_1)s_2 \ldots$ is a zeno execution of $\mathcal{T}$.

($\Leftarrow$) Suppose $\mathcal{T}$ is not non-zeno. Then there exists a zeno execution $\sigma = s_1(t_1, a_1)s_2 \ldots$ of $\mathcal{T}$. Let $\sum_{i=0}^{\infty} t_i = T$. Define *term*$(\sigma) = (s_0, T_0)(t_0, a_0)(s_1, T_1)\ (t_1, a_1)(s_2, T_2) \ldots$, where T_is are defined, inductively, as $T_0 = T$, and $T_{i+1} = T_i - t_i$ for $i \geq 0$. Note that $T_i = T - (t_0 + \ldots + t_{i-1}) \geq 0$, for all i. Therefore, $(s_i, T_i) \overset{(t_0,a_0)}{\to} (s_{i+1}, T_{i+1})$ is a timed transition of *term*$(\mathcal{T})$, and *term*(σ) an infinite execution of *term*$(\mathcal{T})$. ∎

III. Hybrid Automata

Hybrid automata [4], [18] are a popular formalism for modeling systems with mixed discrete-continuous behaviors. The discrete dynamics is modeled by a finite state system and the continuous dynamics by differential equations or inclusions. Next, we introduce a formal definition of hybrid automata and highlight some special subclasses which are needed in the sequel.

Definition 4. *An n-dimensional* hybrid automaton *or* hybrid system *is a tuple $\mathcal{H} = (\mathcal{Q}, \mathit{Init}, \mathit{Inv}, \mathit{Flow}, \mathit{Edges})$, where:*

- *$\mathcal{Q}$ is a finite set of modes or locations;*
- *$\mathit{Init} : \mathcal{Q} \to 2^{\mathbb{R}^n}$ specifies the initial set of states;*
- *$\mathit{Inv} : \mathcal{Q} \to 2^{\mathbb{R}^n}$ provides invariants associated with the locations;*
- *$\mathit{Flow} : \mathcal{Q} \to 2^{\mathit{Traj}(n)}$ specifies the set of continuous trajectories associated with the locations; and*
- *$\mathit{Edges} \subseteq \mathcal{Q} \times 2^{\mathbb{R}^n \times \mathbb{R}^n} \times \mathcal{Q}$ is the set of edges representing the discrete mode changes and the jumps in the continuous states.*

The state-space of the hybrid automaton $\mathcal{H}$ is $\mathcal{Q} \times \mathbb{R}^n$. An execution of the hybrid system starts in an initial state $(q, \vec{v})$, where $\vec{v} \in \mathit{Init}(q)$. Then, it takes a sequence of continuous trajectories and edges. The system can take a continuous trajectory $f \in \mathit{Flow}(q_1)$ with $f(0) = \vec{v}_1$ from a state $(q_1, \vec{v}_1)$ if the continuous states of the trajectory satisfy the invariant of the location q_1 at all times, that is, $f(t) \in \mathit{Inv}(q_1)$ for all times t in its domain. Traversing an edge (q_1, r, q_2) from a state $(q_1, \vec{v}_1)$ to a state $(q_2, \vec{v}_2)$ is possible if the continuous states before and after the transition satisfy the jump relation r, that is, $(\vec{v}_1, \vec{v}_2) \in r$.

Next, we define the semantics of the hybrid automaton as a timed transition system. We first define auxiliary relations ρ_q, ρ_δ and $\rho_{q,\delta}$ where, ρ_q defines the relation between the states $\vec{v}$ and $\vec{v}'$, and time t such that $\vec{v}$ and $\vec{v}'$ are the first and last states of a trajectory of q of duration t; ρ_δ defines the relation between $\vec{v}$ and $\vec{v}'$ which corresponds to states before and after taking the edge δ; and $\rho_{q,\delta}$ is a "composition" of ρ_q and ρ_δ. Let $q \in \mathcal{Q}$ and $\delta = (q, r, q') \in \mathit{Edges}$, and define

$$\rho_q(\vec{v}, t, \vec{v}') \equiv \quad \exists f \in \mathit{Traj}(n) : [f \in \mathit{Flow}(q) \wedge \mathit{dom}(f) = [0, t]$$
$$\wedge \quad \vec{v} = f(0) \wedge \vec{v}' = f(t) \wedge (\forall t' \in \mathit{dom}(f), f(t') \in \mathit{Inv}(q))]$$

$$\rho_\delta(\vec{v}, \vec{v}') \equiv \quad (\vec{v}, \vec{v'}) \in r$$

$$\rho_{q,\delta}(\vec{v}, t, \vec{v}') \equiv \quad \exists \vec{v}'' : \rho_q(\vec{v}, t, \vec{v}'') \wedge \rho_\delta(\vec{v}'', \vec{v}')$$

Definition 5. *Let $\mathcal{H} = (\mathcal{Q}, \mathit{Init}, \mathit{Inv}, \mathit{Flow}, \mathit{Edges})$ be a n-dimensional hybrid automaton, its semantics is the transition system $[\![\mathcal{H}]\!] = (\mathcal{S}, \mathcal{S}_0, \mathit{Edges}, \to)$, where:*

- *$\mathcal{S} = \mathcal{Q} \times \mathbb{R}^n$;*
- *$\mathcal{S}_0 = \{(q, \vec{v}) \mid \vec{v} \in \mathit{Init}(q) \cap \mathit{Inv}(q)\}$; and*
- *$\to$ is given by $\{((q_1, \vec{v}), t, \delta, (q_2, \vec{v}')) \mid \delta = (q_1, r, q_2) \in \mathit{Edges},\ \rho_{q_1,\delta}(\vec{v}, t, \vec{v}')\}$.*

A. Special classes of hybrid automata

In this section, we present certain special subclasses of hybrid automata and concrete representations of the same.

1) Rectangular hybrid automata: Rectangular hybrid automata are a subclass of hybrid automata in which the invariants are rectangular sets, the flows are given by rectangular differential inclusions and the continuous state changes during mode changes are specified by a reset operation which is a combination of a non-deterministic assignment to a rectangular set and an identity operation which leaves the values unchanged. Rectangular hybrid automata are a simple, albeit, useful subclass of hybrid systems; they serve as useful approximations of hybrid systems with rich dynamics [28].

More precisely, an n-dimensional rectangular hybrid automaton is a hybrid automaton which satisfies the following:

- The sets $\mathit{Inv}(q)$ and $\mathit{Init}(q)$ associated with a location q are rectangular sets.
- The flows associated with a location q are specified as the solutions of a rectangular differential inclusion. Each location q is associated with a rectangular set $\mathit{Rate}(q)$, such that $\mathit{Flow}(q)$ is the set of all differentiable functions $f : [0, t] \to \mathbb{R}^n$ such that $\frac{d}{dt} f(t') \in \mathit{Rate}(q)$, for all $t' \in [0, t]$.
- An edge (q_1, r, q_2) of *Edges* is specified using a tuple (q_1, g, h, q_2), where g, referred to as the *guard*, is a rectangular set, and h, referred to as the *reset*, is a function $h : [n] \to \mathcal{I} \cup \{\mathit{Id}\}$. Here, *Id* is a unique symbol not in $\mathcal{I}$. The pair (g, h) represents the set r given by $\{(\vec{v}, \vec{v'}) \,|\, \vec{v} \in g, \forall 1 \le i \le n, [(h(i) = \mathit{Id} \Rightarrow (\vec{v})_i = (\vec{v'})_i) \wedge (h(i) \neq \mathit{Id} \Rightarrow (\vec{v'})_i \in h(i))]\}$.

Example 1. *Fig. 1 shows a one-dimensional rectangular hybrid automaton $\mathcal{H}_{\mathit{approx}}$, with two locations q_1 and q_2, and two edges a_1 and a_2. The continuous dynamics is captured using the variable x. The rectangular sets are represented by constraints which compare variables (or their dotted versions) to a constant or by specifying an interval to which the values belong. For instance, the constraint $\dot{x} \in [-2.2, -1.8]$ in location q_1, specifies that $\mathit{Rate}(q_1)$ is $[-2.2, -1.8]$. The invariant of q_1, represented by the constraint $18 \le x \le 22$, is the rectangular set $[18, 22]$. The guard along the edge $a_1 = (q_1, r, q_2)$, specified by the constraint $x \le 19$, represents the set $(-\infty, 19]$. The resets are represented by constraints on primed variables; if there are no such constraints on an edge, the reset is taken to be Id. A sample execution of $\mathcal{H}_{\mathit{approx}}$ is as follows. The control starts in location q_1 with $x = 21$, and lets time $t = 1$ elapse resulting in x having a value in $[18.8, 19.2]$, say 19. Since the value of x is less than or equal to 19, the control can change to location q_2 using the edge a_1, and the value of x remains unchanged upon mode change.*

Remark 1. *For a rectangular hybrid automaton, $\rho_q(\vec{x}, t, \vec{x'})$ and $\rho_\delta(\vec{x}, \vec{x}')$ can be expressed as a conjunction of linear constraints as follows. Below the guard and reset of δ are denoted g and h, respectively.*

$$\begin{aligned}\rho_q(\vec{x}, t, \vec{x}') \equiv \quad & \vec{x}, \vec{x}' \in \mathit{Inv}(q) \wedge \vec{x} + t \cdot \lfloor \mathit{Rate}(q) \rfloor \le \vec{x}' \\ & \le \vec{x} + t \cdot \lceil \mathit{Rate}(q) \rceil \\ \rho_\delta(\vec{x}, \vec{x'}) \equiv \quad & \vec{x} \in g \wedge \textstyle\bigwedge_{i \in [n], h(i) = \mathit{Id}} (\vec{x})_i = (\vec{x}')_i \\ & \wedge \textstyle\bigwedge_{i \in [n], h(i) \neq \mathit{Id}} (\vec{x}')_i \in h(i)\end{aligned}$$

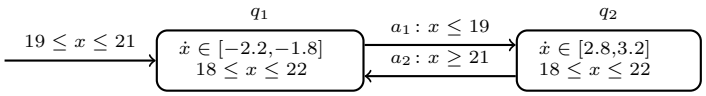

Fig. 1: A rectangular hybrid automaton $\mathcal{H}_{approx}$

The definition of ρ_q uses the fact that there exists a linear trajectory (differential at all points is a constant) between any two points reachable by some differentiable trajectory; see the work of Alur et al. [6] for details. Hence, it suffices to check invariants only at the end-points. In the sequel, we also allow invariants and guards to be polyhedral sets: those specified by linear constraints. Note that ρ_q and ρ_δ can still be expressed using linear constraints.

a) Initialized rectangular hybrid automata: Initialized rectangular hybrid automata are a subclass of rectangular hybrid automata that provide decidability of several properties including safety by restricting the discrete dynamics [19]. A n-dimensional rectangular hybrid automaton $\mathcal{H}$ is said to be *initialized* if for every $\delta = (q_1, r, q_2) \in$ *Edges*, where r is represented by a guard g and reset h, and for all $i \in [n]$, the following holds: *Rate*$(q_1)(i) \neq$ *Rate*$(q_2)(i)$ implies that $h(i) \neq$ *Id*. The above condition states that if the flows associated with the source and the target of an edge are different for a particular variable, then the reset is not identity in which case the dynamics associated with the two locations are "decoupled". For instance, the hybrid automaton $\mathcal{H}_{approx}$ in Figure 1 is not initialized, because the dynamics on q_1 and q_2 are different for x, but the reset for x is identity on the edge a_1.

b) Multi-rate hybrid automata and timed automata: Multi-rate and timed automata are subclasses of rectangular hybrid automata, that are apt for modeling timing constraints [5]. A *multi-rate hybrid automaton* is rectangular hybrid automaton such that *Rate*(q) is a singleton set. Fig. 3 shows a multi-rate hybrid automaton. A *timed automaton* is a multi-rate hybrid automaton in which *Rate*$(q)(i) = \{1\}$, for all q and i, and for all the resets h, $h(i) \in \{Id, \{0\}\}$. Note that a timed automaton is initialized, since, the rate never changes, and hence, does not have any constraints on the type of resets allowed on the edges.

2) Affine hybrid automata: Affine dynamics are an important class of dynamical systems extensively investigated in control theory. In particular, control design and analysis of non-linear systems is often conducted by considering their linearizations. An *affine hybrid automaton* is a rectangular hybrid automaton except that the flows are the solutions of an affine dynamical system. More precisely, each location q of an n-dimensional affine hybrid automaton is provided with a $n \times n$ matrix, $A(q)$, and a $n \times 1$ matrix $b(q)$ such that the flows are solutions of the affine dynamical system $\dot{x} = A(q)x + b(q)$, that is, *Flow*(q) consists of trajectories $f : [0, t] \to \mathbb{R}^n$ such that $\frac{d}{dt} f(t') = A(q)f(t') + b(q)$ for all $t' \in [0, t]$. Even when $b(q) = 0$, the solution of an affine dynamical starting from a values $\vec{v}$ is given by $f(t) = e^{A(q)t}\vec{v}$, where $e^{A(q)t}$ is a matrix exponential. In one dimension, the matrix exponential is essentially the exponential function. Hence, an affine dynamical system ρ_q cannot, in general, be represented by linear constraints. However, there are several techniques which can be used to over-approximate ρ_q by linear constraints [28], [27], [15].

Example 2. *Fig. 2 shows a one-dimensional affine hybrid system depicting the operation of a thermostat. Location q_1 is the "off" mode in which the temperature, represented by the variable x, decreases according to the differential equation $\dot{x} = -0.1x$ and in location q_2, the "on" mode, it increases according to $\dot{x} = 5 - 0.1x$. For instance, the flow f in the "off" mode starting at $x = x_0$ is given by $f(t) = e^{-0.1t}x_0$.*

For the purpose of representability, we will assume that all the constants appearing in the representation of the hybrid automata are rational numbers.

IV. Reduction of non-zenoness problem to termination problem

In this section, we present a transformation of a hybrid system such that the non-zenoness verification problem on the original system is reduced to the termination verification problem on the transformed system. In Section V, we exploit this observation to obtain decidability results for non-zenoness verification problem; and in Section VI, we exploit the state-of-the-art termination verification techniques and tools for automatic non-zenoness verification.

Definition 6. *A hybrid automaton $\mathcal{H}$ is said to be* non-zeno *if $[\![\mathcal{H}]\!]$ is non-zeno. Similarly, a hybrid automaton $\mathcal{H}$ is said to be* terminating *if $[\![\mathcal{H}]\!]$ is terminating.*

Example 3. *Consider the hybrid system $\mathcal{H}_{zeno}$ of Fig. 3. The executions are constrained to be in the rectangle $x \in [0, 2], y \in [0, 1]$ as specified by the invariants. In location q_1, the system evolves at 45 degrees. It can switch to location q_2 either when it hits the line $y = 1$ (the top edge of C and D) or the line $x + y = 1$ (the diagonal edge of A). In location q_2, the system evolves parallel to the negative y-axis. It can then switch back to location q_1, when it hits the x-axis. A sample zeno execution is shown by the dotted lines in Fig. 3, and corresponds to*

$$(q_2, 2/10, 9/10) \overset{9/10,a_3}{\to} (q_1, 2/10, 0) \overset{4/10,a_2}{\to} (q_2, 6/10, 4/10) \overset{4/10,a_3}{\to} (q_1, 6/10, 0) \overset{2/10,a_2}{\to} (q_2, 8/10, 2/10) \overset{2/10,a_3}{\to} (q_1, 8/10, 0) \overset{1/10,a_2}{\to} (q_2, 9/10, 1/10) \overset{1/10,a_3}{\to} (q_1, 9/10, 0) \cdots$$

It is zeno since the time spent in each re-entry into $q1$ (or $q2$) is halved compared to the previous entry. In fact, every execution starting in the region $B \cup D$ with control in q_1, or starting in region D with control in q_2, is non-zeno. From every other state, there is a zeno execution.

Let us consider the non-zenoness verification problem.

Problem 1. *Given a hybrid system $\mathcal{H}$, is it non-zeno?*

We will reduce it to the termination verification problem.

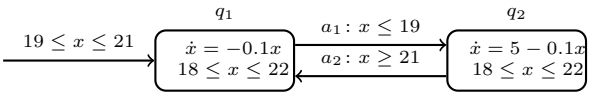

Fig. 2: A affine hybrid automaton $\mathcal{H}_{thermo}$ model of a thermostat

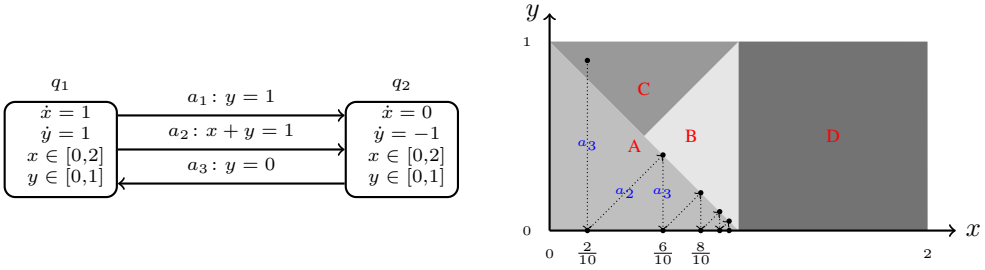

Fig. 3: (left) a hybrid automaton $\mathcal{H}_{zeno}$, (right) invariant $0 \le x \le 2 \wedge 0 \le y \le 1$ of $\mathcal{H}_{zeno}$ divided into: (A) $x + y \le 1$ (B) $x > y \wedge x + y > 1$ (C) $y \ge x \wedge x + y > 1$ (D) $x > 1$.

Problem 2. *Given a hybrid system $\mathcal{H}$, is it terminating?*

Let us return to the question of non-zenoness verification. Given a hybrid automaton $\mathcal{H}$, we wish to verify if it is non-zeno. We define a transformation of $\mathcal{H}$ into another hybrid automaton *Ext*($\mathcal{H}$) such that $\mathcal{H}$ is non-zeno iff *Ext*($\mathcal{H}$) is terminating. Essentially, we introduce an extra variable, which is non-deterministically initialized to some value a in the interval $[0, \infty)$ and decreases at a constant rate of -1 until its value becomes 0. Hence, every execution of the original system is constrained to an interval $[0, a]$ in the transformed system, for some a. A non-terminating execution in the transformed system corresponds to a zeno execution of the original system, since such an execution exhibits an infinite number of discrete transitions in some finite interval $[0, a]$. On the other hand, if the original system has a zeno execution with a limit point $< a$, then the execution in the transformed system which starts with the clock value a and "simulates" the zeno execution is non-terminating.

Let *decr*(t) denote the set of all functions $f : [0, t] \to \mathbb{R}_{\ge 0}$ such that $f(t') = f(0) - t'$, for all $0 \le t' \le t$. Given $r \subseteq \mathbb{R}^n \times \mathbb{R}^n$, let *Ext*($r$) denote the set $\{(\vec{v} \times a, \vec{v'} \times a) \,|\, (\vec{v}, \vec{v'}) \in r, a \in \mathbb{R}_{\ge 0}\}$.

Definition 7. *Let $\mathcal{H} = (\mathcal{Q}, Init, Inv, Flow, Edges)$ be a n-dimensional hybrid automaton, we define Ext($\mathcal{H}$) $= (\mathcal{Q}, \widehat{Init}, \widehat{Inv}, \widehat{Flow}, \widehat{Edges})$, a $(n+1)$-dimensional hybrid automaton, where:*

- *for all q, $\widehat{Init}(q) = Init(q) \times \mathbb{R}_{\ge 0}$;*
- *for all q, $\widehat{Inv}(q) = Inv(q) \times \mathbb{R}_{\ge 0}$;*
- $\widehat{Flow}(q) = \{f \circ f' \,|\, f : [0,t] \to \mathbb{R}^n \in Flow(q), f' \in decr(t)\}$; *and*
- $\widehat{Edges} = \{(q_1, Ext(r), q_2) \,|\, (q_1, r, q_2) \in Edges\}$.

Remark 2. *Ext($\mathcal{H}$) consists of augmenting $\mathcal{H}$ with a variable clk that decreases at a constant rate of -1, that is, $\dot{clk} = -1$; adding invariants $clk \ge 0$ to all the locations, and not allowing the value of clk to change while traversing an edge. For instance, Figure 5a shows a rectangular hybrid automaton $\mathcal{H}$, and Figure 5b shows its transformation Ext($\mathcal{H}$). Note that if $\mathcal{H}$ is affine, rectangular or multi-rate, then so is Ext($\mathcal{H}$), respectively. Further, if $\mathcal{H}$ is initialized, then so is Ext($\mathcal{H}$). However, Ext($\mathcal{H}$) is not a timed automaton, even if $\mathcal{H}$ is.*

The next proposition states that the effect of transforming $\mathcal{H}$ using *Ext*($\cdot$) is that its transition system is transformed using *term*($\cdot$).

Proposition 2. *Let $a = (q_1, r, q_2) \in \mathcal{H}$ and $\hat{a} = (q_1, Ext(r), q_2) \in Ext(\mathcal{H})$, then*

$((q_1, \vec{v}_1), \tau_1) \xrightarrow{t,a}_{term([\![\mathcal{H}]\!])} ((q_2, \vec{v}_2), \tau_2)$ *iff* $(q_1, \vec{v}_1 \times \tau_1) \xrightarrow{t,\hat{a}}_{[\![Ext(\mathcal{H})]\!]} (q_2, \vec{v}_2 \times \tau_2)$.

Proof: Let $\mathcal{H} = (\mathcal{Q}, Init, Inv, Flow, Edges)$ be of dimension n, and let $Ext(\mathcal{H}) = (\mathcal{Q}, \widehat{Init}, \widehat{Inv}, \widehat{Flow}, \widehat{Edges})$.

$$
\begin{aligned}
& ((q_1, \vec{v}_1), \tau_1) \xrightarrow{t,a}_{term([\![\mathcal{H}]\!])} ((q_2, \vec{v}_2), \tau_2) \\
\Leftrightarrow\ & (q_1, \vec{v}_1) \xrightarrow{t,a}_{[\![\mathcal{H}]\!]} (q_2, \vec{v}_2) \wedge \tau_1, \tau_2 \ge 0 \wedge \tau_2 = \tau_1 - t \\
\Leftrightarrow\ & \exists \vec{v}'' \exists f \in Traj(n) : [f \in Flow(q_1) \wedge dom(f) = [0, t] \\
& \wedge \vec{v} = f(0) \wedge \vec{v}'' = f(t) \wedge (\forall t' \in dom(f), f(t') \in Inv(q)) \\
& \wedge (\vec{v}'', \vec{v}') \in r], \tau_1, \tau_2 \ge 0, \tau_2 = \tau_1 - t \\
\Leftrightarrow\ & \exists \hat{v}'' \exists \hat{f} \in Traj(n+1) : [\hat{f} \in \widehat{Flow}(q_1) \wedge dom(\hat{f}) = [0, t] \\
& \wedge \vec{v} = \hat{f}(0) \wedge \vec{v}'' = \hat{f}(t) \wedge (\forall t' \in dom(\hat{f}), \hat{f}(t') \in \widehat{Inv}(q)) \\
& \wedge (\hat{v}'', \vec{v}') \in \hat{r}], \vec{v} = \vec{v} \times \tau_1, \vec{v}' = \vec{v}' \times \tau_2 \\
& \boxed{\text{Here } \hat{f}(t') = f(t') \times (\tau_1 - t') \text{ and } \vec{v}'' = \vec{v}'' \times \tau_2.} \\
\Leftrightarrow\ & (q_1, \vec{v}_1 \times \tau_1) \xrightarrow{t,\hat{a}}_{[\![Ext(\mathcal{H})]\!]} (q_2, \vec{v}_2 \times \tau_2)
\end{aligned}
$$

■

Theorem 1. *$\mathcal{H}$ is non-zeno iff Ext($\mathcal{H}$) is terminating.*

Proof: $\mathcal{H}$ is non-zeno iff $[\![\mathcal{H}]\!]$ is non-zeno. From Proposition 1, the non-zenoness of $[\![\mathcal{H}]\!]$ is equivalent to termination of *term*($[\![\mathcal{H}]\!]$), which is equivalent to termination of $[\![Ext(\mathcal{H})]\!]$, by Proposition 2, and hence, equivalent to termination of $[\![\mathcal{H}]\!]$. ■

V. Boundary of decidability for non-zenoness verification problem

In this section, we identify a boundary of decidability for the non-zenoness verification problem. More precisely, we show that the problem is undecidable for the class of multi-rate hybrid automata, where as, it is decidable for its subclass which consist of initialized automata.

Theorem 2. *Non-zenoness verification problem is decidable for the class of initialized multi-rate hybrid automata (and hence, for the class of timed automata).*

Proof: First, observe that the transformation *Ext*$(\cdot)$ transforms an initialized multi-rate automaton to an initialized multi-rate automaton. Hence, the non-zenoness verification problem for initialized multi-rate automata can be reduced to the termination verification problem for initialized multi-rate automata. We can reduce the problem further to the termination verification of timed automata. Henzinger *et al.* [19] provide a transformation from the class of initialized multi-rate automata to the class of timed automata, such that the timed transition systems of the initialized multi-rate automaton and its timed automaton transformation are bisimilar. Since, bisimulation of the timed transition systems preserves termination, we reduce the problem of termination verification from the class of multi-rate automata to that of timed automata.

It remains to show that the termination verification problem is decidable for the class of timed automata. Using the region construction [5], we can construct a finite state automaton which is bisimilar to the timed transition system of a timed automaton. Hence, we reduce the problem of termination verification for the class of timed automata to that for the class of finite state automata. The latter corresponds to the problem of checking for the existence of cycles reachable from an initial state of the automaton, which can be effectively computed. Hence, non-zenoness verification problem for the class of initialized multi-rate automata can be effectively reduced to termination verification problem of a finite state automaton which can be effectively checked. ■

Next, we show that non-zenoness verification problem is undecidable for the class of multi-rate automata. Hence, in general, we can only hope for techniques that in addition to a yes or no answer, they can also answer *don't know*. In the next section, we discuss some such techniques for non-zenoness verification based on termination verification.

Theorem 3. *Non-zenoness verification problem is undecidable for the class of multi-rate automata (and hence for the class of rectangular hybrid automata).*

Proof: We reduce the location reachability problem to the non-zenoness verification problem. The location reachability problem is the following:

Problem 3. *Given a hybrid system $\mathcal{H}$ and a location q_f, is q_f reachable by an execution of $\mathcal{H}$?*

The location reachability problem is undecidable for the class of multi-rate automata [19]. Given a multi-rate automaton $\mathcal{H}$ and a control state q_f, we construct a multi-rate automaton $\mathcal{H}'$ such that q_f is reachable in $\mathcal{H}$ iff $\mathcal{H}'$ is zeno.

We first construct a multi-rate automaton $\mathcal{H}_1$ which is equivalent to $\mathcal{H}$ in terms of reachability of the location q_f, but does not contain any zeno executions. For this, we replace every location q of $\mathcal{H}$ by a sub-automaton G_q as shown in Figure 4. G_q has locations q_1, q_2, q_3; all the incoming transitions of q are directed to q_1 and all outgoing transitions from q are directed out of q_3. The location q_3 is identical to q in terms of the invariants and rates. In location q_1 all the variables evolve at rate 1 (as shown by the variable $\dot{x} = 1$) for one time unit, followed by spending one time unit in q_2 with rates -1 (as shown by the variable $\dot{x} = -1$). The variable t is used to ensure that exactly one time unit is spent in locations q_1 and q_2. Note the values of all the variables of $\mathcal{H}$ have the same values while entering q_3 as while entering q_1. The new automaton essentially introduces a time elapse of two time units before entering any location, thereby eliminating any zeno-executions.

Next, we construct $\mathcal{H}'$. $\mathcal{H}'$ is similar to $\mathcal{H}_1$ except that from q_f, there is an edge to a subautomaton (disjoint from $\mathcal{H}_1$) with zeno-runs. Hence, if q_f is reachable in $\mathcal{H}$, then q_f is reachable in $\mathcal{H}_1$, and therefore $\mathcal{H}'$ has zeno executions. On the other hand, if $\mathcal{H}'$ has zeno executions, then q_f is reachable in $\mathcal{H}_1$, because, the part of $\mathcal{H}'$ corresponding to $\mathcal{H}_1$ does not have any zeno executions. Therefore, q_f is reachable in $\mathcal{H}$. ■

VI. Practical Aspects

In this section we demonstrate how the transformation, which reduces non-zenoness to an equivalent termination problem, is applied in practice. Termination analysis has been studied extensively for the class of linear constraints automata [7], [16], [3]. Hence, we reduce the termination analysis problem for hybrid automata to that of linear constraint automata. The formal definitions and details of the transformation are given in Subsection VI-A. In the rest of the section, we demonstrate how state of the art termination analysis tools can be used for proving non-zenoness, and uncover zeno runs in hybrid automata: Section VI-B shows how to use the termination proof in order to prove non-zenoness; Section VI-C shows that when failing to prove termination one can infer a precondition on the initial states that guarantees termination; Section VI-D show that in some cases one can detect the presence of zeno executions using termination analysis results; and Section VI-E shows that abstractions can be used to establish non-zenoness of hybrid systems with rich dynamics.

A. Linear constraint automaton

Given a hybrid automaton $\mathcal{H}$, the main steps in the analysis of non-zenoness, are the following:

Step 1. $\mathcal{H}$ is transformed, following Definition 7, into $\mathcal{H}' =$ *Ext*$(\mathcal{H})$;
Step 2. $\mathcal{H}'$ is transformed into a *linear constraints automaton* *Lin*$(\mathcal{H}')$ that precisely captures the executions of $\mathcal{H}'$; and
Step 3. We apply existing termination analysis tools to study the termination behaviour of *Lin*$(\mathcal{H}')$.

a) Step 1.: The transformation of a hybrid automaton $\mathcal{H}$ intro *Ext*$(\mathcal{H})$ is done following Definition 7, that is, we introduce an additional continuous variable *clk*, augment all the locations with invariant *clk* $\in [0, \infty)$ and flow given by $\dot{clk} = -1$. The variable does not change its value on an edge. For

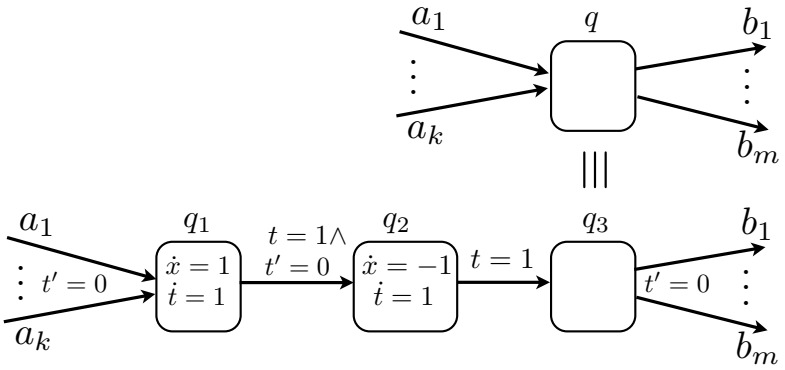

Fig. 4: Gadget G_q replacing a location q

example, the hybrid automaton $\mathcal{H}$ of Figure 5a is transformed into $\mathcal{H}'$ shown in Figure 5b.

b) Step 2.: A *linear constraints automaton* is given by a finite directed graph (V, E) and a finite set $\vec{x}$ of real-valued variables, where each edge $e \in E$ is labeled by conjunctions of linear equalities and inequalities over $\vec{x}$ and $\vec{x}'$. Because it is convenient, we represent such automaton as a pair (ρ, ψ), where

- ρ is a disjunction of formulas over variables $pc, \vec{x}, \vec{x}'$ and pc' of the form $pc = q \wedge \varphi(\vec{x}, \vec{x}') \wedge pc' = q'$ where $(q, q') \in E$ and φ is its label; and
- $\psi \equiv \bigvee_{q \in V}(pc = q \wedge \psi_q(\vec{x}))$ is a formula over variables pc and $\vec{x}$ where each ψ_q is a conjunction of linear constraints over $\vec{x}$, describing a set of valid initial states.

The semantics is given by a set of traces where a trace is a sequence of the form $(q_1, \vec{v}_1), (q_2, \vec{v}_2), \ldots$ where $(pc = q_1 \wedge \vec{x} = \vec{v}_1) \Rightarrow \psi$ and $(pc = q_i \wedge \vec{x} = \vec{v}_i \wedge pc' = q_{i+1} \wedge \vec{x}' = \vec{v}_{i+1}) \Rightarrow \rho$ are valid for every i.

We can encode a hybrid automaton $\mathcal{H}$ as a linear constraints automaton *Lin*($\mathcal{H}$), provided $\rho_{q,\delta}(\vec{x}, t, \vec{x}')$ and *Init*(q) are specified as a conjunction of linear constraints. Each edge $\delta = (q_1, r, q_2)$ in $\mathcal{H}$ yields a formula $pc = q_1 \wedge \rho_{q_1,\delta}(\vec{x}, t, \vec{x}') \wedge pc' = q_2$ in *Lin*($\mathcal{H}$). Also the description on the initial states ψ in *Lin*($\mathcal{H}$) is a disjunction of the formulas of the form $pc = q \wedge$ *Init*(q). It can be proved that the executions of $\mathcal{H}$ and the traces of *Lin*($\mathcal{H}$) are in one-to-one correspondence. Hence, termination problem is reduced to that of linear constraints automata. Note that, following Remark 1, in the case of rectangular hybrid automaton, it is guaranteed that $\rho_{q_1,\delta}(\vec{x}, t, \vec{x}')$ is a conjunction of linear equalities and inequalities. The same holds for *Init*(q). As an example, the hybrid automaton H' of Figure 5b is transformed to the linear constraints automaton *Lin*($\mathcal{H}'$) of Figure 5c. However, for more general dynamics, $\rho_{q,\delta}(\vec{x}, t, \vec{x}')$ can be over-approximated by a conjunction of linear constraints. We will see an example of this in Section VI-C.

B. Verifying non-zenoness

In this section, we illustrate through a simple example how non-zenoness of a hybrid system can be inferred from a termination analysis on its linear constraint automaton.

Consider the hybrid automaton $\mathcal{H}$ of Figure 5a. Its transformation *Ext*($\mathcal{H}'$) is shown in Figure 5b, and the linear constraint automaton in Figure 5c. Feeding *Lin*($\mathcal{H}'$), the linear constraint automaton corresponding to the hybrid automaton $\mathcal{H}$ of Figure 5a to a termination analyser [7], returns a proof of termination. The proof annotates locations q_1 and q_2 with the following ranking functions $f_{q_1}(x, y, clk) = x + 20 \cdot y + 8 \cdot clk$ and $f_{q_2}(x, y, clk) = x + 8 \cdot clk + 1$. Note that $f_{q_i}(\vec{v}) \geq 0$ for each $\vec{v} \in$ *Inv*(q_i) and $i = 1, 2$; and that when entering q_i with $\vec{v}$ and then move to location q_j with $\vec{v}'$, using any of the transitions, it holds that $f_{q_i}(\vec{v}) \geq f_{q_j}(\vec{v}') + 1$. This prevents infinite traces since an infinite trace $(q_{i_0}, \vec{v}_0), (q_{i_1}, \vec{v}_1), \ldots$ would induce an infinite decreasing chain $f_{q_{i_0}}(\vec{v}_0) \geq f_{q_{i_1}}(\vec{v}_1) + 1 \geq f_{q_{i_2}}(\vec{v}_2) + 1 \geq \cdots$, which contradicts that the f_{q_i} are bounded from below. Hence, termination of *Lin*($\mathcal{H}'$) is established. We can then conclude the non-zenoness of $\mathcal{H}$.

C. Inference of pre-conditions for non-zenoness

Consider the hybrid automaton $\mathcal{H}$ of Figure 3. Define $\mathcal{H}'$ to be *Ext*($\mathcal{H}$). The corresponding *Lin*($\mathcal{H}'$) is defined by $(\rho_1 \vee \rho_2 \vee \rho_3, true)$ where

$$
\begin{aligned}
\rho_1(\langle \vec{x}, pc \rangle, \langle \vec{x'}, pc' \rangle) \quad &\equiv \exists t\colon \big(pc = q_1 \wedge t \geq 0 \wedge clk \geq 0 \\
&\wedge 0 \leq x \leq 2 \wedge 0 \leq y \leq 1 \\
&\wedge x' = x + t \wedge y' = y + t \wedge y' = 1 \\
&\wedge clk' = clk - t \wedge pc' = q_2\big) \\
\rho_2(\langle \vec{x}, pc \rangle, \langle \vec{x'}, pc' \rangle) \quad &\equiv \exists t\colon \big(pc = q_1 \wedge t \geq 0 \wedge clk \geq 0 \\
&\wedge 0 \leq x \leq 2 \wedge 0 \leq y \leq 1 \\
&\wedge x' = x + t \wedge y' = y + t \wedge x' + y' = 1 \\
&\wedge clk' = clk - t \wedge pc' = q_2\big) \\
\rho_3(\langle \vec{x}, pc \rangle, \langle \vec{x'}, pc' \rangle) \quad &\equiv \exists t\colon \big(pc = q_2 \wedge \ t \geq 0 \wedge clk \geq 0 \\
&\wedge 0 \leq x \leq 2 \wedge 0 \leq y \leq 1 \\
&\wedge x' = x \wedge y' = y - t \wedge y' = 0 \\
&\wedge clk' = clk - t \wedge pc' = q_1\big) \ .
\end{aligned}
$$

Feeding *Lin*($\mathcal{H}'$) to a termination analyser [7], it fails to prove termination. However, it provides us with useful information that can be used to infer sets of states from which termination is guaranteed. Next we explain how.

As in the case of Section VI-B, the termination analyzer annotates each location q with a (ranking) function f_q, however, unlike that case, some of the transitions satisfy $f_{q_i}(\vec{v}) \geq f_{q_i}(\vec{v'}) + 1$, call them *decreasing transitions*, while the rest satisfy $f_{q_i}(\vec{v}) \geq f_{q_i}(\vec{v'})$, call them *non-increasing transitions*. Now since all f_q are well-founded, it is easy to see that an

(a) $\mathcal{H}$

(b) $\mathcal{H}' = \mathit{Ext}(\mathcal{H})$

$$
\begin{aligned}
\rho_1(\langle \vec{x}, pc\rangle, \langle \vec{x'}, pc'\rangle) &\equiv \exists t\colon pc = q_1 \wedge \rho_{q_1,a_1}(\vec{x}, t, \vec{x}') \wedge pc' = q_2 \\
&\equiv \exists t\colon \big(pc = q_1 \wedge t \geq 0 \wedge \mathit{clk} \geq 0 \wedge x \geq 0 \wedge 0 \leq y \leq 8 \wedge \\
&\qquad x' = x - 12t \wedge 8 = y + t \wedge y' = 0 \wedge \mathit{clk}' = \mathit{clk} - t \wedge pc' = q_2\big) \\
\rho_2(\langle \vec{x}, pc\rangle, \langle \vec{x'}, pc'\rangle) &\equiv \exists t\colon pc = q_2 \wedge \rho_{q_2,a_2}(\vec{x}, t, \vec{x}') \wedge pc' = q_1 \\
&\equiv \exists t\colon \big(pc = q_2 \wedge t \geq 0 \wedge \mathit{clk} \geq 0 \wedge x \geq 0 \wedge 0 \leq y \leq 10 \wedge \\
&\qquad x' = x + 8t \wedge 10 = y + t \wedge y' = 0 \wedge \mathit{clk}' = \mathit{clk} - t \wedge pc' = q_1\big)
\end{aligned}
$$

(c) $\mathit{Lin}(\mathcal{H}')$

Fig. 5: An example of a hybrid automata $\mathcal{H}$ [26], its instrumentation $\mathit{Ext}(\mathcal{H})$ with variable *clk*, and the corresponding linear constraints automata $\mathit{Lin}(\mathcal{H}')$.

infinite trace cannot involve decreasing transitions infinitely often, and thus it must have a suffix that consists only of non-increasing ones. Clearly, if we stay away from states that allow infinite traces of non-increasing transitions, then termination is guaranteed. Those states are exactly defined by the following greatest fixpoint [16]:

$$\mathcal{Z} = \mathit{gfp}\lambda X.\, \mathcal{S} \cap \mathit{pre}[\rho'](X)$$

where ρ' are the non-increasing transitions, $\mathit{pre}[\rho'](X) \stackrel{def}{=} \{\vec{v} \mid \exists \vec{v}' \in X\colon \rho(\vec{v}, \vec{v}')\}$, and $\mathcal{S}$ is the space of all states. Then, to guarantee termination, traces have to start from those initial states guaranteed not to reach any state in $\mathcal{Z}$. To compute these, it is enough to complement the set $\mathcal{V}$ of states that *may* reach a state in $\mathcal{Z}$ or, stated equivalently, the predecessors of $\mathcal{Z}$. Formally, we have:

$$\mathcal{V} = \mathit{lfp}\lambda X.\, \mathcal{Z} \cup \mathit{pre}[\rho](X) \ .$$

where ρ is the set of all transitions. Back to our example, we first infer that the non-increasing transitions are ρ_2 and ρ_3. Then, we compute $\mathcal{Z}$:

$$\{(x, y, \mathit{clk}, q_1) \mid x + y \leq 1 \wedge \mathit{Inv}(q_1)\}$$
$$\cup\{(x, y, \mathit{clk}, q_2) \mid x \leq 1 \wedge \mathit{Inv}(q_2)\} \ .$$

and then $\mathcal{V} = \mathcal{Z} \cup \{(x, y, \mathit{clk}, q_1) \mid x \leq y\}$. Finally, the set of states from which $\mathit{Lin}(\mathcal{H}')$ always terminate is the complement of $\mathcal{V}$:

$$\{(x, y, \mathit{clk}, q_1) \mid x + y > 1 \wedge x > y \wedge \mathit{Inv}(q_1)\}$$
$$\cup\{(x, y, \mathit{clk}, q_2) \mid x > 1 \wedge \mathit{Inv}(q_2)\} \ .$$

Note that this set coincides with those initial states of Example 3 that are claimed to have non-zeno executions.

D. Detecting zeno executions

Consider the hybrid automaton $\mathcal{H}$ of Figure 6, and let $\mathcal{H}' = \mathit{Ext}(\mathcal{H})$. The corresponding *linear constraints automaton* $\mathit{Lin}(\mathcal{H}')$ includes as many relation as edges in $\mathcal{H}$. For simplicity, we only give the those corresponding to a_1 and b_2:

$$
\begin{aligned}
\rho_1(\langle \vec{x}, pc\rangle, \langle \vec{x'}, pc'\rangle) &\equiv \exists t\colon \big(pc = q_1 \wedge t \geq 0 \wedge \mathit{clk} \geq 0 \\
&\quad \wedge x \geq 0 \wedge y \geq 5 \wedge x' = x \wedge y' = y - 5t \\
&\quad \wedge 0 \leq x' \leq 50 \wedge \mathit{clk}' = \mathit{clk} - t \wedge pc' = q_2\big) \\
\rho_2(\langle \vec{x}, pc\rangle, \langle \vec{x'}, pc'\rangle) &\equiv \exists t\colon \big(pc = q_2 \wedge t \geq 0 \wedge \mathit{clk} \geq 0 \wedge 0 \leq x \\
&\quad \leq 50 \wedge y \geq 5 \wedge x' = x + 10t \wedge y' = y - 5t \\
&\quad \wedge x' \geq 45 \wedge \mathit{clk}' = \mathit{clk} - t \wedge pc' = q_1\big)
\end{aligned}
$$

The termination analyser [3], [7] applied on $\mathit{Lin}(\mathcal{H}')$ fails to prove termination. Using the output of the termination analyser, and considerations similar to those of Section VI-C, we conclude that every non-terminating execution of $\mathit{Lin}(\mathcal{H}')$, and thus every zeno execution of $\mathcal{H}$, if any, necessarily contains an infinite suffix in which ρ_1 and ρ_2 alternate. Indeed, $\mathcal{H}$ has a zeno execution obtained by the alternation of a_1 and b_2. Unfortunately, in this case, we were not able to infer any *non-trivial* precondition that guarantees termination of $\mathit{Lin}(\mathcal{H}')$.

Remark that it is easy to prove non-termination of a linear constraints automata that consists of the transitions ρ_1 and ρ_2, simply by checking that $\rho_1 \circ \rho_2 \wedge \vec{x} = \vec{x}'$ is satisfiable, e.g., for $x = 45$, $y = 6$ and $\mathit{clk} = 1$, which is a valid initial state. This means that $\mathcal{H}$ has an infinite execution where time does not elapse. More involved automatic reasoning can prove non-termination even when time elapses in q_1 and q_2.

E. Approximation based non-zenoness analysis

In the examples we have considered so far, the relation $\rho_{q,\delta}(\vec{x}, t, \vec{x}')$ was expressible as a conjunction of linear constraints. Next, we will consider an example for which this is not the case. Consider the affine hybrid automaton $\mathcal{H}_{thermo}$ of Figure 2 which models a thermostat. We use the observation that over-approximations preserve termination (and non-zenoness), that is, if we can show that an over-approximation of a

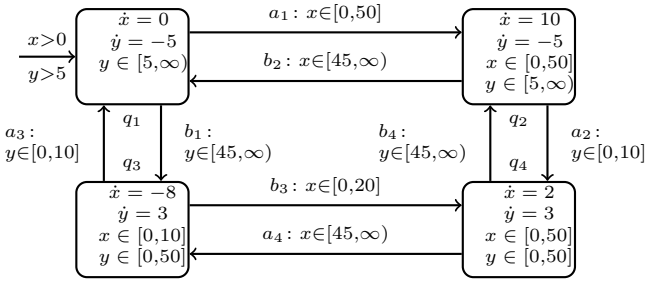

Fig. 6: Two tank(s) hybrid automata $\mathcal{H}$ [26].

linear constraint automaton is terminating, then so is the linear constraint automaton. The over-approximation of a linear constraint automaton *Lin*(*Ext*($\mathcal{H}$)) can be obtained by over-approximating $\rho_{q,\delta}(\vec{x}, t, \vec{x}')$ of hybrid automaton $\mathcal{H}$ as a conjunction of linear constraints.

There are several approaches in the hybrid systems literature for over-approximating the solutions of affine hybrid systems by a conjunction of linear constraints including sampling based methods [27], [15] and hybridization based methods [28], [12]. Figure 1 shows a rectangular hybrid automaton over-approximation of the hybrid automaton $\mathcal{H}_{approx}$ using the techniques of Doyen and Henzinger [12]. Here, the affine dynamics in each location is over-approximated by a rectangular inclusion dynamics by optimizing over the invariants. For instance, in location q_1, the value of $-0.1x$ (the right hand side of the differential equation) is bounded by the interval $[-2.2, -1.8]$ when the value of x satisfies $18 \leq x \leq 22$ (the invariant). Hence, the approximate dynamics is $\dot{x} \in [-2.2, -1.8]$. Then, we apply the termination analysis on the linear constraint automaton corresponding to the abstract automaton Figure 1, for which we obtain a termination proof. This illustrates the feasibility of approximation based analysis of non-zenoness.

VII. Conclusion

In this paper, we investigated automated methods for analyzing the absence of zeno behaviors in hybrid systems. Our broad approach consisted of reducing the problem to that of termination analysis. This enabled us to obtain decidability results for certain subclasses of hybrid automata, as well as practical techniques for proving non-zenoness and for generating initial conditions from which the system has no zeno executions. We have shown that these methods can be applied to systems with non-trivial dynamics by using over-approximations. Our future efforts will focus on conducting case studies and rigorous experimentation of non-zeno analysis involving linear and non-linear hybrid systems. Also, we intend to explore techniques for analysing the weaker notion of absence of livelocks in hybrid systems.

Acknowledgment

The work of Samir Genaim was funded partially by the EU project FP7-ICT-610582 ENVISAGE: Engineering Virtualized Services (http://www.envisage-project.eu), by the Spanish MINECO project TIN2012-38137, and by the CM project S2013/ICE-3006. The work of Pierre Ganty was funded partially by the EU project FP7-ICT-610686 POLCA and by the Madrid Regional Government CM project S2013/ICE-2731 (N-Greens).

References

[1] P. Abdulla, P. Mahata, and R. Mayr. Decidability of zenoness, syntactic boundedness and token-liveness for dense-timed petri nets. In *FSTTCS '04*, volume 3328 of *LNCS*, pages 58–70. Springer, 2004.

[2] P. A. Abdulla, M. F. Atig, and J. Stenman. Zenoness for timed pushdown automata. In *Proceedings 15th International Workshop on Verification of Infinite-State Systems, INFINITY 2013, Hanoi, Vietnam, 14th October 2013.*, pages 35–47, 2014.

[3] C. Alias, A. Darte, P. Feautrier, and L. Gonnord. Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In *SAS '10: Proc. 20th Int. Static Analysis Symp.*, volume 6337 of *LNCS*, pages 117–133. Springer, 2010.

[4] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1993.

[5] R. Alur and D. Dill. A theory of timed automata. *Theor. Comput. Science*, 126:183–235, 1994.

[6] R. Alur, T. A. Henzinger, and P. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng.*, 22(3):181–201, 1996.

[7] A. M. Ben-Amram and S. Genaim. Ranking functions for linear-constraint loops. *J. ACM*, 61(4):26, 2014.

[8] M. Brockschmidt, B. Cook, and C. Fuhs. Better termination proving through cooperation. In *CAV '13: Proc. 23rd Int. Conf. on Computer Aided Verification*, volume 8044 of *LNCS*, pages 413–429. Springer, 2013.

[9] M. K. Camlibel, J.-S. Pang, and J. Shen. Conewise linear systems: Non-zenoness and observability. *SIAM J. Control and Optimization*, 45(5):1769–1800, 2006.

[10] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *PLDI '06: Proc. 27th ACM-SIGPLAN Conf. on Programming Language Design and Implementation*, pages 415–426. ACM, 2006.

[11] B. Cook, A. See, and F. Zuleger. Ramsey vs. lexicographic termination proving. In *TACAS '13: Proc. 19th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *LNCS*, pages 47–61. Springer, 2013.

[12] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Automatic rectangular refinement of affine hybrid systems. In *FORMATS '05*, volume 3829 of *LNCS*, pages 144–161. Springer, 2005.

[13] P. S. Duggirala and S. Mitra. Abstraction refinement for stability. In *ICCPS*, pages 22–31, 2011.

[14] M. Egerstedt, K. H. Johansson, S. Sastry, and J. Lygeros. On the regularization of Zeno hybrid automata. *Systems and Control Letters*, 38:141–150, 1999.

[15] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. 2011 (to appear).

[16] P. Ganty and S. Genaim. Proving termination starting from the end. In *CAV '13: Proc. 23rd Int. Conf. on Computer Aided Verification*, volume 8044 of *LNCS*, pages 397–412. Springer, 2013.

[17] R. Hadjidj, H. Boucheneb, and D. Hadjidj. Zenoness detection and timed model checking for real time systems. In *VECoS'07: Proc. of the 1st Int. Conf. on Verification and Evaluation of Computer and Communication Systems*, pages 120–134. British Computer Society, 2007.

[18] T. A. Henzinger. The theory of hybrid automata. In *LICS '96: Proc. 11th Annual IEEE Symp. on Logic in Computer Science*, pages 278–292, 1996.

[19] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Journal of Computer and System Sciences*, pages 373–382. ACM Press, 1995.

[20] J. Hespanha and A. Morse. Stabilization of nonholonomic integrators via logic-based switching. *Automatica*, 35(3):385–393, Mar 1999.

[21] M. Heymann, G. Meyer, and S. Resmerita. Analysis of zeno behaviors in hybrid systems. In *CDC '02: Proc. of the 41st IEEE Conference on Decision and Control*, pages 2379–2384, 2002.

[22] J. L. J. Zhang, K. Johansson and S. Sastry. Zeno hybrid systems. *International Journal of Robust and Nonlinear Control*, 11:435–451, 2001.

[23] K. H. Johansson, J. Lygeros, S. Sastry, and M. Egerstedt. Simulation of zeno hybrid automata. In *CDC '99: Proc. of the 38th IEEE Conference on Decision and Control*, volume 4, 1999.

[24] A. G. Lamperski and A. D. Ames. Lyapunov theory for zeno stability. *IEEE Trans. Automat. Contr.*, pages 100–112, 2013.

[25] D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Proving termination of imperative programs using max-smt. In *FMCAD '13: Proc. 13th Int. Conf. of Formal Methods in Computer-Aided Design*, pages 218–225. IEEE, 2013.

[26] A. Podelski and S. Wagner. A method and a tool for automatic verification of region stability for hybrid systems. Technical Report MPI-I-2007-2-001, Max-Planck-Institut für Informatik, 2007.

[27] P. Prabhakar and M. Viswanathan. A dynamic algorithm for approximate flow computations. pages 133–143, 2010.

[28] A. Puri, V. S. Borkar, and P. Varaiya. Epsilon-approximation of differential inclusions. In *Hybrid Systems*, volume 1066 of *LNCS*, pages 362–376, 1995.

[29] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Transactions on Automatic Control*, 1999.

[30] L. Q. Thuan and M. K. Camlibel. Continuous piecewise affine dynamical systems do not exhibit zeno behavior. *IEEE Trans. Automat. Contr.*, 56(8):1932–1936, 2011.

[31] S. Tripakis. Verifying progress in timed systems. In J.-P. Katoen, editor, *Formal Methods for Real-Time and Probabilistic Systems*, volume 1601 of *Lecture Notes in Computer Science*, pages 299–314. Springer Berlin Heidelberg, 1999.