Lyapunov Abstractions for Inevitability of Hybrid Systems*

Parasara Sridhar Duggirala
Department of Computer Science
University of Illinois at Urbana Champaign
Urbana, IL
duggira3@illinois.edu

Sayan Mitra
Coordinate Science Laboratory
University of Illinois at Urbana Champaign
Urbana, IL
mitras@illinois.edu

ABSTRACT

A set of states S is said to be inevitable for a hybrid automaton $\mathcal{A}$ if every behavior of $\mathcal{A}$ ultimately reaches S within bounded time. Inevitability captures various commonly occurring liveness properties. In this paper, we present an algorithm for verifying inevitability of Linear Hybrid Automata (LHA). The algorithm combines (a) Lyapunov function-based relational abstractions for the continuous dynamics with (b) automated construction of well-founded relations for the loops of the hybrid automaton. The algorithm is complete for automata that are symmetric with respect to the chosen Lyapunov functions. The algorithm is implemented in a prototype tool (*Ly*SHA) which is integrated with a Simulink/Stateflow frontend for modeling hybrid systems. The experimental results demonstrate the effectiveness of the methodology in verifying inevitability of hybrid automata with up to five continuous dimensions and forty locations.

Keywords

Stability; Switching Systems; Abstraction; Formal Methods;

1. INTRODUCTION

In this paper, we present a technique for verifying a type of liveness property called *inevitability* [21, 3] for systems involving both discrete and continuous dynamics. The hybrid system formalisms [1, 13] provide a convenient mathematical framework for modeling such systems. A behavior or an *execution* of a hybrid automaton is an alternating sequence of discrete state transitions and continuous trajectories, where the latter are often specified by differential and algebraic equations. A set of states S is said to be *inevitable* for a hybrid automaton (HA) $\mathcal{A}$ if starting from arbitrary initial states, every execution of $\mathcal{A}$ reaches S within bounded time. Inevitability captures practical liveness requirements such as: a robot must arrive within a neighborhood of a waypoint, a traffic control algorithm must allow vehicles to make progress along all intersecting routes, and a routing protocol must arrive at a valid forwarding scheme after some nodes fail.

The *switched system* model (see, for example the books [15] and [25] and the references therein) can be viewed as an abstraction of hybrid automata. Here the guards and resets are abstracted by exogenous switching signals. Naturally, the problem of verifying inevitability for hybrid automata is connected to the problem of analyzing stability of switched systems. Research on the latter problem has focused on establishing necessary and sufficient conditions for stability. These conditions typically assume (a) existence of common or multiple Lyapunov functions and (b) timing-dependent restrictions on families of switching signals for guaranteeing stability [5, 12]. Although, there are some examples of verification algorithms which build upon these results (for example [19]), in general, a systematic framework for liveness verification of hybrid systems is still missing. In this paper, we propose such a framework which combines Lyapunov functions with program analysis techniques. We present an instance of this framework as an abstraction-refinement algorithm for inevitability verification of Linear Hybrid Automata. Our prototype implementation of the algorithm automatically verifies inevitability and suggests counterexamples for HA with upto five continuous variables and forty locations.

1.1 Overview

Throughout this paper we will assume that the system model $\mathcal{A}$ is nonblocking, that is, it allows time to diverge. For reactive systems which are expected to run for long periods of time this assumption is reasonable. Now, suppose we want to prove the inevitability of hybrid automaton $\mathcal{A}$ to the set S. Let $\mathcal{B}$ be the automaton obtained by removing the set S from $\mathcal{A}$. In [9], we observed that $\mathcal{A}$ inevitably reaches S if and only if the automaton $\mathcal{B}$ is blocked within finite time. In other words, time cannot diverge to infinity in $\mathcal{B}$ if and only if S is inevitable for $\mathcal{A}$. This suggests that inevitability can be verified by verifying the blocking property. This blocking property is related to termination of programs and we will leverage available and recently developed techniques for program termination verification.

The standard technique for proving program termination involves finding a *well-founded relation* which subsumes the transitions of the program. A well-founded relation has no infinite chains, and hence, the program cannot have infinite executions. Unlike programs, however, HA have uncount-

*This research is supported by a research grant from the National Science Foundation (CNS-1016791).

Hybrid Systems Computation and Control '12 Beijing, China

able states which evolve both through discrete transitions and continuous trajectories. Furthermore, a single continuous trajectory can be decomposed into arbitrarily many small steps and seemingly violate the well-foundedness criteria. The first step in our approach is to define a new transition relation (called *hybrid step relation (HSR)*) which combines the continuous trajectories and the discrete transitions. Through the lens of program analysis, then, verifying inevitability of $\mathcal{A}$ becomes equivalent to proving well-foundedness of $\mathcal{B}$'s HSR.

The HSR captures detailed information about the state-evolution. For linear dynamics it involves real arithmetic formulas with exponentials and trigonometric functions. Therefore, the above methodology faces computational barriers from two directions: first in computing HSRs and in computing the composition of these HSRs, second in checking the well-foundedness of HSRs. To address these issues, we abstract the HSR with a collection of Lyapunov-like functions for the individual locations. A wide variety of standard techniques based on semidefinite and convex optimization are available for finding Lyapunov functions [14, 4, 22]. For instance, for linear HA, the focus of this paper, Lyapunov functions for the individual modes are obtained by solving the (linear) Lyapunov equations. We abstract the state space of the system using Lyapunov-like functions and this defines a *Lyapunov Step Relation*(LSR) which overapproximates the HSR. We show how LSR can be computed effectively for time-triggered linear HA. We also identify a class of HA for which there is no loss of precision in abstracting the HSR with Lyapunov abstractions.

Even with the Lyapunov abstractions, proving well-foundedness of the LSR of complex hybrid systems can become intractable. Leveraging recent results in program termination analysis which have led to effective new tools for termination analysis of realistic software systems [8, 6], we show that it suffices to find well-founded relations for the LSRs corresponding to individual loops of the HA. Roughly, if all the loops of $\mathcal{B}$ are well-founded, then none of them can be sustained forever, and we can conclude that the $\mathcal{B}$ is blocking, and that $\mathcal{A}$ inevitably reaches S.

One final barrier to the above approach is that the number of loops of $\mathcal{B}$ can be infinite. To address this, we abstract the individual loop LSRs with finitely many transition predicates. This guarantees termination of our algorithm, even though it may sometimes fail to prove inevitability or the blocking property. Putting it all together, we obtain a new abstraction-refinement based framework for verifying inevitability of HA. Based on our experimental results with a prototype implementation of the algorithm for linear HA, the method appears to scale to system models with more than forty locations and five or so continuous states.

1.2 Related Work

Abstractions over the state space of dynamical systems have been used to prove safety properties in [24, 16]. Both, of these approaches, use timed automaton for abstracting the behavior of dynamical system. In [24], however, Lyapunov functions were used for creating the timers in the timed automaton abstraction. This paper shows how similar Lyapunov-based abstractions can be applied to linear HA for verifying inevitability.

In [23], transitional abstractions (relational abstractions) were used to prove safety properties of the systems. These transitional abstractions capture the relation among the state variables from the beginning to the end of a trajectory (i.e. continuous evolution of the system). These transitional abstractions are computed from the eigenstructure of the relevant matrices and by performing quantifier elimination. In contrast, our abstractions differ from this approach primarily in two ways, First, we require the transitional abstractions to also capture the discrete behavior of the system by defining the Hybrid Step Relation which captures both the continuous trajectories and the discrete transitions of the system. Secondly, we require explicit Laypunov functions for abstraction of the state space of the system. The techniques of [23] may provide effective means of computing abstractions of the type we need for inevitability. This direction will be explored in the future.

A different algorithm for stability of a linear HA which relies on finiteness of certain sequences called snapshot sequences has been presented in [21]. In [21], stability of a linear hybrid system is characterized as *finiteness* of certain sequences called *snapshot sequences* . Three types of snapshot sequences are identified and a new hybrid automaton representing the execution of these three snapshot sequences are created. Further, by analyzing the unary reachability of these automaton using reachability tools like PHAVer [10], finiteness of these sequences is determined and hence stability of the given automaton is verified. Our approach of first computing Lyapunov abstractions could be combined with this analysis of snapshot sequences.

2. PRELIMINARIES

We briefly introduce the basic concepts of the hybrid automaton framework and refer the reader to [1, 13, 18] for details. For a vector x in $\mathbb{R}^k$ then we denote its components by $x[1], x[2], \ldots, x[k]$. A relation $R \subseteq S \times S$ on a set S is *well-founded* if it does not contain any infinite sequence of pairwise related elements. A set of variables, say V, is used to define the state of a hybrid automaton. Each variable $v \in V$ is associated with a *type*, denoted by $type(v)$, which defines the set of values it can take. A valuation $\mathbf{v}$ for a set of variables V maps each $v \in V$ to an element in $type(v)$. We use the standard $\mathbf{v}.v$ notation to refer to the valuation of a variable $v \in V$ at $\mathbf{v}$. The set of all valuations of V is denoted by $val(V)$.

The state of a hybrid automaton, that is, the valuation of its variables can change through instantaneous *discrete transitions* and continuously over an interval of time by following a *trajectory*. A trajectory for a set of variables V is a function $\tau : [0, t] \to val(V)$, where either $t \in \mathbb{R}_{\geq 0}$ or ∞. The domain of τ is denoted by $\tau.dom$. The first state of τ, denoted by $\tau.\mathsf{fstate}$, is $\tau(0)$. A trajectory is closed if t is finite and in this case we define $\tau.\mathsf{ltime} \triangleq t$, $\tau.\mathsf{lstate} \triangleq \tau(t)$, and we write $\tau.\mathsf{fstate} \xrightarrow{\tau} \tau.\mathsf{lstate}$.

Definition 1. *A Hybrid Automata (HA) $\mathcal{A}$ is a tuple $\langle V, Loc, A, \mathcal{D}, \mathcal{T} \rangle$ where*

(a) $V = X \cup \{loc\}$ is a set of variables. Here loc is a discrete variable of finite type Loc. Elements of Loc are called locations. *Each $x \in X$ is a continuous variable of type $\mathbb{R}$. Elements of $val(V)$ are called* states.

(b) A is a finite set of actions *or transition labels.*

(c) $\mathcal{D} \subseteq val(V) \times A \times val(V)$ is the set of discrete transitions.

A transition $(\mathbf{v}, a, \mathbf{v}') \in \mathcal{D}$ *is written as* $\mathbf{v} \xrightarrow{a} \mathbf{v}'$. *The discrete transitions are specified by finitely many guards and reset maps involving* V.

(d) $\mathcal{T}$ *is a set of trajectories for* X *which is closed under suffix, prefix and concatenation (see [13,* **?***] for details). For each* $l \in Loc$, *a set of trajectories* $\mathcal{T}_l$ *for location* l *are specified by differential equations* E_l *and an invariant* $I_l \subseteq val(X)$. *Over any trajectory* $\tau \in \mathcal{T}_l$, *loc remains constant and the variables in* X *evolve according to* E_l *such that for all* $t \in \tau.dom, \tau(t) \in I_l$. $\mathcal{T} \triangleq \cup_{l \in Loc} \mathcal{T}_l$.

Later in this paper, we introduce a class of HA called *time-triggered* hybrid automaton which have a special timer variable now that tracks the time spent in each location.

2.1 Semantics of Hybrid Automata

An execution of a hybrid automaton $\mathcal{A}$ records all the information (about variables) over a particular run. Formally, an *execution* is an alternating sequence $\tau_0 a_1 \tau_1 \ldots$ where each τ_i (except possibly the last) is a closed trajectory and $\tau_i.\mathsf{lstate} \xrightarrow{a_{i+1}} \tau_{i+1}.\mathsf{fstate}$. Note that executions may start from any state of the automaton. The first state of α is denoted by $\alpha.\mathsf{fstate} = \tau_0.\mathsf{fstate}$. The duration of α is defined as $\alpha.\mathsf{ltime} = \sum_i \tau_i.\mathsf{ltime}$. The property of interest in this paper is inevitability: A set of states $S \subseteq val(V)$ is said to be *inevitable* for $\mathcal{A}$ if every execution of $\mathcal{A}$ reaches S in finite time.

An execution is *finite* if it is a finite sequence, otherwise it is *infinite*. An execution α is said to be *nonblocking* if its duration is infinite or if along all its infinite extensions time diverges to infinity; otherwise α is said to be *blocking*. A hybrid automata $\mathcal{A}$ is *blocking (nonblocking)* if all its executions are blocking (nonblocking, respectively). A hybrid automaton is said to be *Zeno* if there exists an execution of finite duration with infinitely many transitions.

The notion of a simulation relation will be used later in our development.

Definition 2. *A relation* $R \subseteq val(V) \times val(V)$ *is a* simulation relation *if for every* $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_1' \in val(V)$ *and* $\mathbf{v}_1 \; R \; \mathbf{v}_2$, *(a) if* $\mathbf{v}_1 \xrightarrow{a} \mathbf{v}_1'$ *then there exists* $\mathbf{v}_2'$ *such that* $\mathbf{v}_1' \; R \; \mathbf{v}_2'$, $\mathbf{v}_2 \xrightarrow{a} \mathbf{v}_2'$, *and (b) if* $\mathbf{v}_1 \xrightarrow{\tau} \mathbf{v}_1'$ *then there exists* $\mathbf{v}_2'$ *such that* $\mathbf{v}_1' \; R \; \mathbf{v}_2'$, $\mathbf{v}_2 \xrightarrow{\tau} \mathbf{v}_2'$.

2.2 Three Assumptions

We assume that the hybrid automata to be verified is *nonblocking*. This is a natural assumption for systems which are expected to run for long time periods.

A hybrid automaton may be non-blocking because it stays in one location forever. In such cases, the verification problem reduces to the well-studied problem of analyzing the stability of the differential equations of that location with respect to the target set. Our goal is to study the hybrid aspect of inevitability verification, and therefore, we assume that the automaton is *locally blocking*. That is, time cannot diverge within a single location.

Finally, we will assume that the automaton does not have Zeno executions. Systems can be designed to be non-Zeno with built-in dwell times. Furthermore, several techniques have been developed for checking Zenoness and dwell time properties [2, 19].

3. VERIFYING INEVITABILITY

We begin this section by setting up inevitability verification as a problem of checking well-foundedness of a certain relation which we call the *Hybrid Step Relation (HSR)*. Next, we introduce Lyapunov function-based abstractions for over-approximating HSRs. We show that for a special class of HA which conform to the symmetries of the Lyapunov functions, Lyapunov abstractions provide a complete method for proving well-foundedness of HSR. Finally, we show how Lyapunov abstractions can be computed for time-triggered or periodically-controlled hybrid automata.

3.1 Inevitability to Well-Foundedness

Unlike programs which evolve in atomic discrete transitions, a trajectory can be split into infinitely many small trajectories. The relation relating all these intermediate states in a trajectory is not well-founded. The hybrid step relation combines a trajectory with a following transition, and thereby relates a state $\mathbf{v}$ with the maximal state $\mathbf{v}'$ that can follow a trajectory from $\mathbf{v}$.

Definition 3. *For HA* $\mathcal{A}$, *the* hybrid step relation (HSR) $\Gamma \subseteq val(V) \times val(V)$ *is defined as:* $(\mathbf{v}, \mathbf{v}') \in \Gamma$ *iff there exist* $\mathbf{v}'' \in val(V), a \in A, \tau \in \mathcal{T}$ *such that* $\mathbf{v} = \tau.\mathsf{fstate}, \mathbf{v}'' = \tau.\mathsf{lstate}$ *and* $\mathbf{v}'' \xrightarrow{a} \mathbf{v}'$.

In other words, from $\mathbf{v}$ there is a trajectory followed by a transition which takes it to $\mathbf{v}'$.

A key observation from [9] (restated as Theorem 1 below) relates inevitability to blocking: A set of states S is inevitable for HA $\mathcal{A}$ if an only if $\mathcal{A}_{S^c}$ is blocking, where $\mathcal{A}_{S^c}$ is obtained by (1) removing the states S from the state-space of $\mathcal{A}$ (2) removing the transitions into S from the transitions of $\mathcal{A}$ and (3) removing the non-trivial trajectories in S from the trajectories of $\mathcal{A}$.

Theorem 1. *(from [9]) A locally blocking, non-Zeno HA* $\mathcal{A}$ *is blocking iff* Γ *is well-founded.*

Proof. Suppose that $\mathcal{A}$ is not blocking, i.e. there exists an execution α in which time diverges. Since $\mathcal{A}$ is locally closed, the execution α is of the form $\tau_0, a_1, \ldots$. Let $\beta = \mathbf{v}_0, \mathbf{v}_1, \ldots$, where $\mathbf{v}_i = \tau_i.\mathsf{fstate}$. Hence, we have that $\forall i, (\mathbf{v}_i, \mathbf{v}_{i+1}) \in \Gamma$. Thus β is an infinite sequence and Γ is not well-founded.

Suppose that the HSR Γ is not well-founded. Hence, we have an infinite sequence $\beta = \mathbf{v}_0, \mathbf{v}_1, \ldots$, where $(\mathbf{v}_i, \mathbf{v}_{i+1}) \in \Gamma$. From β and Γ, we can construct a sequence $\beta' = \mathbf{v}_0$, $\mathbf{v}_0'$,$\mathbf{v}_1$,$\mathbf{v}_1'$,... such that $\forall i, \exists \tau_i, a_i$ such that $\mathbf{v}_i = \tau_i.\mathsf{fstate}, \mathbf{v}_i' = \tau_i.\mathsf{lstate}$ and $\mathbf{v}_i' \xrightarrow{a_{i+1}} \mathbf{v}_{i+1}$. Thus we have that $\alpha = \tau_0, a_1, \tau_1, \ldots$ is an admissible execution of the system. Since $\mathcal{A}$ is non-Zeno, it means that time diverges along execution α and thus $\mathcal{A}$ is not blocking. □

For rectangular hybrid automata [11] the hybrid step relation can be computed exactly and a concrete algorithm is presented in our previous work in [9]. For general hybrid automata, computing the hybrid step relation is undecidable. For this purpose, we compute the abstractions of hybrid step relation with respect to an abstraction β over the state space of the system. The next theorem illustrates the utility of abstractions for proving well-foundedness.

Definition 4. *Given a HSR* Γ, *an abstract domain* D, *and a function* $\beta\colon val(V) \to D$, *we define* $\beta(\Gamma) \subseteq D \times D$ *as the relation* $(y, y') \in \beta(\Gamma)$ *iff* $\exists \mathbf{v}, \mathbf{v}' \in val(V)$, *such that* $(\mathbf{v}, \mathbf{v}') \in \Gamma$, $y = \beta(\mathbf{v})$ *and* $y' = \beta(\mathbf{v}')$.

Theorem 2 follows from Theorem 1 in [7]

Theorem 2. *If $\beta(\Gamma)$ is well-founded then Γ is also well-founded.*

Theorem 2 will be useful, for example, in the case of time-triggered linear hybrid automata, where computing the hybrid step relation involves real arithmetic with exponentials while computing the abstraction of the HSR will involve only linear real arithmetic.

3.2 Lyapunov Abstractions

We assume that all the locations of the hybrid automata are either asymptotically stable or asymptotically unstable. In the case of linear systems, this implies that we can effectively compute Lyapunov-like functions that exponentially decay or grow along the trajectories. Formally, for every location, $l \in Loc$, we have k such functions, denoted by $L_{l,1}, L_{l,2}, \ldots, L_{l,k}$. If the location l is stable, then for each $i \in \{1, \ldots, k\}$, $\exists \lambda_{l,i} < 0$ and $B_{l,i} > 0$, such that for every trajectory $\tau \in \mathcal{T}_l$, $L_{l,i}(\tau(t)) \leq B_{l,i} e^{\lambda_{l,i} t} L_{l,i}(\tau(0))$. If the location l is unstable, then $\exists \lambda_{l,i} > 0$ and $B_{l,i} > 0$, such that for every trajectory $\tau \in \mathcal{T}_l$, $L_{l,i}(\tau(t)) \leq B_{l,i} e^{\lambda_{l,i} t} L_{l,i}(\tau(0))$.

Definition 5 (Lyapunov Abstraction)**.** *Let $\mathfrak{L} = \{L_{\ell,i}\}, \ell \in Loc, i \in \{1, \ldots, k\}$ be an ordered collection of $k|Loc|$ Lyapunov functions. The* Lyapunov Abstraction *is a function $\mathcal{L} \colon val(V) \to Loc \times \mathbb{R}^k$ defined as:*

$$\mathcal{L}(\mathbf{v}) = \langle \mathbf{v}.loc, L_{\mathbf{v}.loc,1}(\mathbf{v}.X), \ldots, L_{\mathbf{v}.loc,1}(\mathbf{v}.X) \rangle.$$

For convenience, we also define $\mathcal{L}(\mathbf{v}).loc = \mathbf{v}.loc$ and

$$\mathcal{L}(\mathbf{v}).X = (L_{\mathbf{v}.loc,1}(\mathbf{v}.X), \ldots, L_{\mathbf{v}.loc,k}(\mathbf{v}.X)).$$

$\mathcal{L}(\Gamma) \subseteq (Loc \times \mathbb{R}^k) \times (Loc \times \mathbb{R}^k)$ is called the Lyapunov Step Relation*(LSR).*

It follows from Theorem 2 that if a LSR $\mathcal{L}(\Gamma)$ is well-founded then the corresponding HSR Γ is also well-founded. For computing LSR, we need to construct Lyapunov-like functions for each location. For linear hybrid systems, for asymptotically stable modes, a quadratic Lyapunov function $x^T P x$ can be effectively computed by solving the Lyapunov equations in the matrix form: $AP + PA^T = -Q$, where Q is a positive semidefinite matrix. Similarly, for asymptotically unstable modes, a quadratic Lyapunov-like function $x^T P x$ can be obtained by solving the matrix equation $AP + PA^T = Q$, where Q is a positive semidefinite matrix.

Definition 6 (Symmetric States)**.** *Given a HA $\mathcal{A}$ and a Lyapunov abstraction $\mathcal{L}$, we define the relation $\mathcal{S}_\mathcal{L}$ as $(\mathbf{v}, \mathbf{v}') \in \mathcal{S}_\mathcal{L}$ iff $\mathcal{L}(\mathbf{v}) = \mathcal{L}(\mathbf{v}')$. We say that two $\mathcal{S}_\mathcal{L}$-related states are symmetric with respect to $\mathcal{L}$.*

Definition 7 (Symmetric Hybrid Automata)**.** *Given a HA $\mathcal{A}$ and a Lyapunov abstraction $\mathcal{L}$, $\mathcal{A}$ is said to be symmetric with respect to $\mathcal{L}$ iff $\mathcal{S}_\mathcal{L}$ is a simulation relation for $\mathcal{A}$.*

Informally, an automaton is symmetric with respect to a collection of Lyapunov functions if any two states with identical valuation of those Lyapunov functions behave indistinguishably. Since the transitions and the trajectories of the HA preserve any simulation relation between states, it follows that the hybrid step relation also preserves symmetry.

Proposition 3. *Given a HA $\mathcal{A}$ that is symmetric with respect to a Lyapunov abstraction $\mathcal{L}$, for any $\mathbf{v}_1, \mathbf{v}_1', \mathbf{v}_2 \in val(V)$, if $(\mathbf{v}_1, \mathbf{v}_1') \in \Gamma$ and $(\mathbf{v}_1, \mathbf{v}_2) \in \mathcal{S}_\mathcal{L}$ then there exists $\mathbf{v}_2' \in val(V)$, and $(\mathbf{v}_1', \mathbf{v}_2') \in \mathcal{S}_\mathcal{L}$ such that $(\mathbf{v}_2, \mathbf{v}_2') \in \Gamma$.*

Example 1 We illustrate the *Lyapunov step relation* using a simple example. Consider the HA in Figure 1. At location $i \in \{1, 2, 3\}$, the dynamics of the system is defined by $\dot{x} = A_i x$. Notice that all the invariants depend only on the timer now. Suppose

$$A_1 = \begin{pmatrix} -1 & 0 \\ 5 & -3 \end{pmatrix} A_2 = \begin{pmatrix} 2 & 1 \\ 0 & -1 \end{pmatrix} A_3 = \begin{pmatrix} -4 & -2 \\ 0 & -9 \end{pmatrix}.$$

Eigenvalues for A_1 and A_3 are negative and those of A_2 are positive, therefore, locations 1 and 3 are asymptotically stable while 2 is unstable. Consider an execution starting from

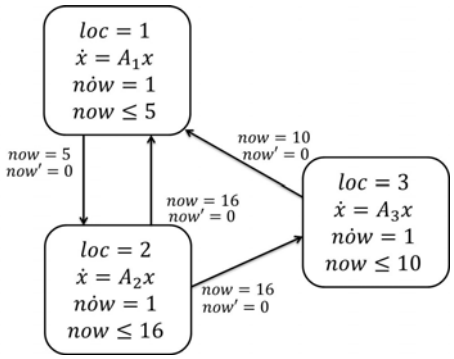

Figure 1: Linear hybrid automaton with 3 locations.

the state $\mathbf{v}$ where $\mathbf{v}.loc = 1, \mathbf{v}.x_1 = 2, \mathbf{v}.x_2 = 5$. Initially, the system will evolve according to the continuous dynamics $\dot{x} = A_1 x$. After 5 time units, the location changes to 2. By solving the differential equation for mode A_1, we obtain that $\mathbf{v} \xrightarrow{\tau} \mathbf{v}''$ where $\mathbf{v}''.loc = 1, \mathbf{v}''.x_1 = 0.0135$ and $\mathbf{v}''.x_2 = 0.0337$, and $\tau.\mathrm{ltime} = 5$. Further, the system changes its location from 1 to 2, i.e. $\mathbf{v}'' \to \mathbf{v}'$, we have $\mathbf{v}'.loc = 2, \mathbf{v}'.x_1 = 0.0135$ and $\mathbf{v}'.x_2 = 0.0337$. Therefore, we get that $(\mathbf{v}, \mathbf{v}') \in \Gamma$.

The two Lyapunov functions for locations 1 and 2 are obtained as

$$\begin{aligned} L_{1,1} &= 0.5x_1^2 + 1.25x_1x_2 + 1.2083x_2^2 \\ L_{1,2} &= 0.5x_1^2 + 1.75x_1x_2 + 1.625x_2^2 \\ L_{2,1} &= x_1^2/3 - x_1x_2/3 + x_2^2/2, \text{ and} \\ L_{2,2} &= x_1^2/6 + x_1x_2/3 + x_2^2/2. \end{aligned}$$

Let $y = \mathcal{L}(\mathbf{v}) = \langle 1, L_{1,1}(\mathbf{v}), L_{1,2}(\mathbf{v}) \rangle = \langle 1, 44.7705, 60.1250 \rangle$. Let $y' = \mathcal{L}(\mathbf{v}') = \langle 2, L_{2,1}(\mathbf{v}'), L_{2,2}(\mathbf{v}) \rangle = \langle 2, 4.7691 \times 10^{-4}, 7.4991 \times 10^{-4} \rangle$. Hence, we have $(y, y') \in \mathcal{L}(\Gamma)$. □

3.3 Refinement and Completeness

In general, the abstract relation $\mathcal{L}(\Gamma)$ may not be well-founded even if the HSR Γ is. In other words, $\mathcal{L}(\Gamma)$ may contain an infinite chain $\sigma = y_1, y_2, \ldots$ such that there is no concrete execution of $\mathcal{A}$ corresponding to the infinite sequence of concrete states $\mathcal{L}^{-1}(\sigma) \triangleq \mathcal{L}^{-1}(y_1), \mathcal{L}^{-1}(y_2), \ldots$. Such sequences are called *potential counterexamples*. If any one of these executions is indeed an execution of HA, then the system is not *blocking* . Otherwise, the counterexample is said to be *spurious*. In such cases, we would like to *refine* the abstraction to $\mathcal{L}'$, such that, the set of *potential counterexample* $\mathcal{L}'^{-1}(\sigma)$ decreases by some measure. Ideally, we would like this sequence of refinements to terminate. That is, proving the well-foundedness of

some kLSR should suffice for proving well-foundedness of the concrete HSR. In this section, we present a technique for refining Lyapunov abstractions and identify a class of hybrid automata for which this procedure is guaranteed to terminate.

Definition 8 (Distinguishing Lyapunov function)**.** *For a given set of k Lyapunov-like functions $L_1, \ldots, L_k$, a $(k+1)st$ function L_{k+1} is said to be* distinguishing *if $\forall x_1, x_2, \ldots, x_k \in \mathbb{R}_{\geq 0}$, if $\bigcap_{i=1}^{k} L_i^{-1}(x_i) \neq \emptyset$, then there exists $x_{k+1} \in \mathbb{R}_{\geq 0}$ such that*

$$\bigcap_{i=1}^{k+1} L_i^{-1}(x_i) \subsetneq \bigcap_{i=1}^{k} L_i^{-1}(x_i).$$

Informally, the $(k+1)^{st}$ function is able to distinguish points in any subspace defined by the intersection of level-sets of all the other k Lyapunov functions. We now prove that the number of *potential counterexamples* decrease when a distinguishing Lyapunov function is added to each location for the Lyapunov abstraction.

Proposition 4. *Given $\mathfrak{L} = \{L_{\ell,i}\}, \ell \in Loc, i \in \{1, \ldots, k\}$ define a Lyapunov abstraction $\mathcal{L}$, a refinement of this abstraction $\mathfrak{L}'$ can be obtained by adding a distinguishing Lyapunov function for each location.*

Proof. Suppose that $\sigma = y_1, y_2, \ldots$ be a counterexample for the abstraction $\mathcal{L}$ where $y_i \in Loc \times \mathbb{R}^k$ and let $\mathbf{v}_1, \mathbf{v}_2, \ldots$ be the set of *potential counterexamples* $\mathcal{L}^{-1}(\sigma)$. Now, consider the counterexample $\sigma' = (y_1, z_1), (y_2, z_2), \ldots$ of $\mathcal{L}'$ where $(y_i, z_i) \in Loc \times \mathbb{R}^{k+1}$ and $\mathbf{v}'_1, \mathbf{v}'_2, \ldots$ be the set of *potential counterexamples* $\mathcal{L}^{-1}(\sigma')$. Let $V_i = \{\mathbf{v}_i\}$ and $V'_i = \{\mathbf{v}'_i\}$ From Definitions 5 8, we have that whenever V_i is non-empty, we have that $V'_i \subsetneq V_i$, in which case, the number of potential counter examples decreases. If V_i is an empty set, then the sequence $y_1, y_2, \ldots$ is not an abstract counter example as (y_{i-1}, y_i) cannot be in $\mathcal{L}(\Gamma)$. □

We observe that Lyapunov function $x^T P_{k+1} x$ is *distinguishing* from $x^T P_1 x, \ldots, x^T P_k x$ only when $P_1, \ldots, P_{k+1}$ are *linearly independent* . In practice, we follow this strategy to add *distinguishing* Lyapunov functions during refinement of Lyapunov abstractions.

Example 2 To illustrate the refinement process, we consider the hybrid automata in Example 1. Consider the execution of the system starting from $\mathbf{v}$ where $\mathbf{v}.loc = 1, \mathbf{v}.x_1 = 2, \mathbf{v}.x_2 = 5$. We have established in Example 1 that $\mathbf{v}$ will evolve according to continuous dynamics for 5 time units and reaches the state $\mathbf{v}''$ where $\mathbf{v}''.loc = 1, \mathbf{v}''.x_1 = 0.0135$ and $\mathbf{v}''.x_2 = 0.0337$ and then takes a discrete transition to $\mathbf{v}'$ where $\mathbf{v}'.loc = 2, \mathbf{v}'.x_1 = 0.0135$ and $\mathbf{v}'.x_2 = 0.0337$. Therefore, we get that $\mathbf{v} \Gamma \mathbf{v}'$.

Now consider the abstraction $\mathcal{L}$ of the system where we have one Lyapunov function associated with each location, i.e. $L_{1,1}$ for location 1 and $L_{2,1}$ for location 2. Now in this abstraction, the Lyapunov step relation would be $y = \langle 1, L_{1,1}(\mathbf{v}) \rangle$ and $y' = \langle 2, L_{2,1}(\mathbf{v}') \rangle$. Thus we have that $y = \langle 1, 44.7705 \rangle$ and $y' = \langle 2, 4.7691 \times 10^{-4} \rangle$. Hence $(y, y') \in \mathcal{L}(\Gamma)$. However, this approximation is too coarse because this abstract transition corresponds to all the states on the level set of $L_{1,1} = 44.7705$, related to all the states on the level set of $L_{2,1} = 4.7691 \times 10^{-4}$. To make the abstraction *finer* we add 2 more Lyapunov functions $L_{1,2}$ and $L_{2,2}$ to $\mathcal{L}$ and obtain $\mathcal{L}'$. The kLSR for $\mathcal{L}'$ corresponding to the *refined* Lyapunov abstraction is $(y, y') \in \mathcal{L}(\Gamma)$, where $y = \mathcal{L}'(\mathbf{v}) = \langle 1, L_{1,1}(\mathbf{v}), L_{1,2}(\mathbf{v}) \rangle = \langle 1, 44.7705, 60.1250 \rangle$, $y' = \mathcal{L}'(\mathbf{v}') = \langle 2, L_{2,1}(\mathbf{v}'), L_{2,2}(\mathbf{v}) \rangle = \langle 2, 4.7691 \times 10^{-4}, 7.4991 \times 10^{-4} \rangle$. It can be observed that by adding a new Lyapunov function for each location, one can eliminate a lot of spurious transitions. In this example, this corresponds to eliminating some of the transitions where the states in level set $L_{1,1} = 44.7705$ are not related to $L_{2,1} = 4.7691 \times 10^{-4}$. □

Theorem 5. *For a symmetric hybrid automaton $\mathcal{A}$ w.r.t. a Lyapunov abstraction $\mathcal{L}$, we have that Γ is well-founded iff $\mathcal{L}(\Gamma)$ is well-founded.*

Proof. From Theorem 2 we have that whenever $\mathcal{L}(\Gamma)$ is well-founded, Γ is also well-founded. Now, we need to prove that when $\mathcal{L}(\Gamma)$ is not well-founded, Γ is also not well-founded. We prove this as follows: we consider an infinite chain $\mathcal{L}(\Gamma)$ and from this chain, we prove that there exists an infinite chain in Γ.

Let $y_1, y_2, \ldots$ be an infinite chain in $\mathcal{L}(\Gamma)$. Now, since $(y_1, y_2) \in \mathcal{L}(\Gamma)$, by Definition 5, we have that $\exists (\mathbf{v}_1, \mathbf{v}_2) \in \Gamma$ such that $\mathcal{L}(\mathbf{v}_1) = y_1, \mathcal{L}(\mathbf{v}_2) = y_2$. Now, since $(y_2, y_3) \in \mathcal{L}(\Gamma)$, we have that $\exists (\mathbf{v}'_2, \mathbf{v}'_3) \in \Gamma$ such that $\mathcal{L}(\mathbf{v}'_2) = y_2$ and $\mathcal{L}(\mathbf{v}'_3) = y_3$. Further since $\mathcal{L}(\mathbf{v}_2) = \mathcal{L}(\mathbf{v}'_2)$, we have that $\mathbf{v}_2 \mathcal{S}_{\mathcal{L}} \mathbf{v}'_2$. From Theorem 3, we have that Γ is also symmetric and hence $\exists \mathbf{v}_3$ such that $\mathcal{L}(\mathbf{v}_3) = \mathcal{L}(\mathbf{v}'_3) = y_3$ and $(\mathbf{v}_2, \mathbf{v}_3) \in \Gamma$. Similarly, we get $(\mathbf{v}_3, \mathbf{v}_4) \in \Gamma$. Hence, one can construct an infinite chain $\mathbf{v}_1, \mathbf{v}_2, \ldots$ such that $(\mathbf{v}_i, \mathbf{v}_{i+1}) \in \Gamma$ and thus Γ is not well-founded. □

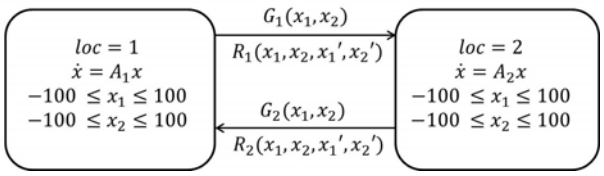

Figure 2: A Symmetric Linear hybrid automaton with 2 locations.

Next, we now look at examples of hybrid automata that are symmetric and asymmetric.

Example 3 Consider the automaton shown in Figure 2. It consists of two continuous variables variables x_1, x_2, where $A_1 = \begin{pmatrix} -2 & 0 \\ 1 & -3 \end{pmatrix}$ and $A_2 = \begin{pmatrix} -1 & 2 \\ 1 & -3 \end{pmatrix}$. Let $L_1(x_1, x_2) \triangleq 0.25x_1^2 + 0.1x_1x_2 + (0.55/3)x_2^2$ and $L_2(x_1, x_2) \triangleq 1.75{x_1}^2 + 1.25x_1x_2 + 0.375{x_2}^2$. The guards and reset maps are defined as $G_1(x_1, x_2) \triangleq L_1(x_1, x_2) = 6.25$, $R_1(x_1, x_2, x'_1, x'_2) \triangleq L_2(x'_1, x'_2) = 175$, $G_2(x_1, x_2) \triangleq L_2(x_1, x_2) = 25$ $R_1(x_1, x_2, x'_1, x'_2) \triangleq L_1(x'_1, x'_2) = 2$. We can observe that L_1 is a Lyapunov function for location 1 and L_2 is a Lyapunov function for location 2. Hence for a Lyapunov abstraction $\mathcal{L}$ with the set of Lyapunov function $\mathfrak{L} = \{L_1, L_2\}$, we get that the system is symmetric. In Section 5, we observe how to prove inevitability for this system using $\mathcal{L}$. □

Example 4 Now consider the trivial hybrid system shown in Figure 3. It consists of only one location and only one con-

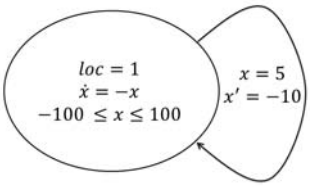

Figure 3: An asymmetric Linear hybrid automaton with 1 location.

tinuous variable. A quadratic Lyapunov function for the system is $L_{1,1} = x^2$. We can see that the system is not symmetric with respect to the Lyapunov abstraction because of the discrete transitions. We observe that the state where the value of $x = -5$ is symmetric with respect to $x = 5$. However, the state $x = 5$ can take a discrete transition and go to state $x = -10$ whereas $x = -5$ cannot have such a transition. Later, in Section 5, we observe that even though the given system satisfies inevitability, since the Lyapunov abstraction is not complete, we cannot prove it using Lyapunov abstractions. □

3.4 Time-Triggered Hybrid Automata

In this section, we will derive the *LSR* for Time-Triggered Hybrid Automata. In Time-Triggered Hybrid Automata, the transition guards are always predicated on the timer variable now and may also depend on other variables.

Definition 9. *A* time-triggered hybrid automaton *$\mathcal{A}$ is a HA with (a) a* guard $G : A \to 2^{val(V)}$ *which associates each action $a \in A$ with an enabling predicate, (b) a special clock variable called now, and (c) a* constraint *function $C : Loc \to \mathbb{R}_+$ which associates a timing constraint for each location. In addition, the following axioms are satisfied:*

(i) For a discrete transition $\mathbf{v} \xrightarrow{a} \mathbf{v}'$, $\mathbf{v} \in G(a)$ and $\mathbf{v}'.X = \mathbf{v}.X$. Further, if $\mathbf{v}.loc$ is stable, then $\mathbf{v}.now \geq C(\mathbf{v}.loc)$ and $\mathbf{v}'.now = 0$. If $\mathbf{v}.mode$ is unstable, then $\mathbf{v}.now \leq C(\mathbf{v}.loc)$ and $\mathbf{v}'.now = 0$.

(ii) For any trajectory τ in a stable location l, either $\tau.\mathsf{ltime} \geq C(l)$ or there exists a τ' for which τ is a prefix and $\tau'.\mathsf{ltime} \geq C(l)$.

(iii) For any trajectory τ in an unstable location l, $\tau.\mathsf{ltime} \leq C(l)$.

A hybrid system is called *strictly time-triggered* when all the switching conditions involve only the clock variable now. An example of such a hybrid system is given in Figure 1. We assume that one can come up with $\{L_{l,i}\}, l \in Loc, i \leq i \leq k$ Lyapunov functions which decays (grows) exponentially for stable (unstable) locations.

Now, compute the LSR for time-triggered HA. Consider $(\mathbf{v}, \mathbf{v}') \in \Gamma$ where $\mathbf{v}.loc = l$ and $\mathbf{v}'.loc = m$. Let $\mu_{l,m}$ be a constant such that $\forall \mathbf{v} \xrightarrow{a} \mathbf{v}', \mathcal{L}(\mathbf{v}').X[i] \leq \mu_{l,m}\mathcal{L}(\mathbf{v}').X[i]$. Let $max_{G(a)}$ and $min_{G(a)}$ be arrays of real numbers such that $1 \leq i \leq k$, $min_{G(a)}[i]$ and $max_{G(a)}[i]$ are the least upper bound and greatest lower bound of the set $S_i = \{y | \exists \mathbf{v} \in G(a), y = \mathcal{L}(\mathbf{v}).X[i]\}$.

Theorem 6. *We say that $(y, y') \in \mathcal{L}(\Gamma)$ when $(min_{G(a)}[i] \leq B_{l,i}e^{\lambda_{l,i} \times C(l)}y.X[i]) \wedge (y'.X[i] \leq \mu_{l,m}B_{l,i}e^{\lambda_{l,i} \times C(l)}y.X[i])$ where $l = y.loc$*

Proof. Let $(\mathbf{v}, \mathbf{v}') \in \Gamma$, then $\exists \mathbf{v}''$ such that $\exists \tau \in \mathcal{T}_{\mathbf{v}.loc}, \tau.\mathsf{fstate} = \mathbf{v}, \tau.\mathsf{lstate} = \mathbf{v}''$ and $\exists a \in \mathcal{D}, (\mathbf{v}'', a, \mathbf{v}')$. Hence, we have that $\mathbf{v}''.X \in G(a)$ and since the reset map is identity, we have $\mathbf{v}'.X \in G(a)$. Now, let $y = \mathcal{L}(\mathbf{v}), y' = \mathcal{L}(\mathbf{v}'), y'' = \mathcal{L}(\mathbf{v}''), l = \mathbf{v}.loc, l' = \mathbf{v}.loc$. Since we have that the system should stay in location l for at least $C(l)$ time if l is stable, or else it should stay at most for $C(l)$ time in unstable modes, we have that $y.loc = l$, $y''.loc = l$ and $y'.loc = l'$. Further, we have,

$$y''.X[i] \leq B_{l,i}e^{\lambda_{l,i} \times C(l)}y.X[i] \tag{1}$$

Further, since $v'' \in G(a)$, we have

$$min_{G(a)}[i] \leq y''.X[i] \leq max_{G(a)}[i] \tag{2}$$

And by the value of μ_{ij}, we also have

$$y'.X[i] \leq \mu_{l,m}y''.X[i] \tag{3}$$

By combining equations 1, 2 and 3, we get that $(y, y') \in \mathcal{L}(\Gamma)$ when

$$(min_{G(a)}[i] \leq B_{l,i}e^{\lambda_{l,i} \times C(l)}y.X[i]) \wedge \tag{4}$$

$$(y'.X[i] \leq \mu_{l,m}B_{l,i}e^{\lambda_{l,i} \times C(l)}y.X[i]) \tag{5}$$

□

Equation (4) represents the LSR relation for time-triggered HA.

In this section we have seen how to obtain the Lyapunov step relation $\mathcal{L}(\Gamma)$ for time-triggered hybrid systems. To prove the blocking property, we need to prove the well-foundedness of $\mathcal{L}(\Gamma)$ which we discuss next.

4. ABSTRACTION REFINEMENT FOR WELL-FOUNDEDNESS

We have established that inevitability can be verified by finding a well-founded relation R that contains the HSR Γ or its Lyapunov abstraction $\mathcal{L}(\Gamma)$. In general, finding such a well-founded relation R is difficult as it must subsume the hybrid-steps along all possible paths or sequences of locations that can be visited by the hybrid automaton. Theorem 1 from [20] gives an alternative method for searching for such relations by allowing us to come up with a well-founded relation for each loop (sequence of locations starting and ending at the same location). Restated in our context, this theorem gives that Γ is well-founded iff its transitive closure Γ^+ is contained in the disjunctive union of a finite collection of well-founded relations. That is, $\Gamma^+ \subseteq \cup_{i=1}^{m} R_i$, where R_i is a well-founded relation and m is a natural number. An algorithm for verifying inevitability in rectangular hybrid systems based on this observation is presented in [9].

Applying the same algorithm for linear hybrid systems poses two challenges. First, the HSR in the case of linear hybrid systems involves exponentials. In order to overcome this challenge, we prove the well-foundedness of LSR for the loops. Second, the number of loops to be considered can be possibly infinite. To overcome this challenge, we perform standard predicate abstraction of the loop LSRs. Even if the number of loops and hence the number of loop LSRs is infinite, with respect to a finite collection of predicates, the set of abstract loop LSRs becomes finite.

Proposition 7. *The Lyapunov step relation $\mathcal{L}(\Gamma)$ is blocking if and only if its transitive closure of $(\mathcal{L}(\Gamma))^+$ is contained in the disjunctive union of finite well-founded relations. That is, there exists a collection $\{R_1, \ldots, R_m\}$ of well-founded relations such that $(\mathcal{L}(\Gamma))^+ \subseteq \cup_{i=1}^m R_i$.*

For a given $(y, y') \in (\mathcal{L}(\Gamma))^+$, it corresponds to an execution fragment α of $\mathcal{A}$ starting from $\mathbf{v}$ and ending at $\mathbf{v}'$ such that $\mathcal{L}(\mathbf{v}) = y$ and $\mathcal{L}(\mathbf{v}') = y'$. If we can find a finite collection of well-founded relations containing the Lyapunov abstractions of the first and last states of every execution, then we are done. First, observe that α is such that $\mathbf{v}.loc = l_1$ and $\mathbf{v}'.loc = l_2$ and $l_1 \neq l_2$, then the trivial relation $\{(y, y') \mid y.loc = l_1 \wedge y'.loc = l_2\}$ proves the well-foundedness of the relation $\{(y, y')\}$. Since there are finitely many locations, there are finite number of relations of this type which cover all executions which do not start and end at the same location. Hence, it suffices to consider (y, y') pairs with $y.loc = y'.loc$.

Next, to restrict the set of loops to be checked, we perform predicate abstractions over LSR. An abstraction with respect to a collection of transition predicates $\mathcal{P}$ is a function that maps a relation ρ to another relation $abs_{\mathcal{P}}(\rho)$ where $\rho \subseteq abs_{\mathcal{P}}(\rho)$. For every relation $\rho \subseteq {\mathbb{R}_{\geq 0}}^k \times {\mathbb{R}_{\geq 0}}^k$, we define the abstraction of ρ with respect to the set of predicates $\mathcal{P}$ as the conjunction of all the predicates that are weaker than ρ. In other words, if $p_1, \ldots, p_n \in \mathcal{P}$ such that $\rho \subseteq p_i$, we have that $abs_{\mathcal{P}}(\rho) = p_1 \wedge \ldots \wedge p_n$. For a relation $\sigma = \rho_1 \circ \ldots \circ \rho_n$, obtained by composing several relations, we define $abs_{\mathcal{P}}(\sigma)$ inductively as follows

$$\begin{aligned} abs(\sigma) &= abs_{\mathcal{P}}(\rho_1 \circ abs_{\mathcal{P}}(\sigma_1)) \\ &\quad \text{where } \sigma_1 = \rho_2 \circ \ldots \circ \rho_n \\ abs_{\mathcal{P}}(\sigma_{n-1}) &= abs_{\mathcal{P}}(\rho_n) \end{aligned}$$

Now we present the algorithm for proving well-foundedness. The algorithm iteratively adds to a collection of predicates $\mathcal{P}$ and a collection of well-founded relations $\mathcal{R}$ as it checks each loop. For each loop of the hybrid automaton (say π), compute the LSR $\mathcal{L}(\Gamma)_\pi$ of the loop and its abstraction $abs_{\mathcal{P}}(\mathcal{L}(\Gamma)_\pi)$. If this abstract relation is subsumed by one of the well-founded relations in $\mathcal{R}$ then this loop cannot be sustained forever, and we move on to the next loop. Otherwise, there are three possibilities: (1) If provably well-founded by a new relation R (but not by any relation in $\mathcal{R}$), then add R to $\mathcal{R}$ increasing the arsenal of well-founded relations[1]. (2) Else if $\mathcal{L}(\Gamma)_\pi$ is well-founded (though its abstraction is not), then refine the abstraction by adding the path and loop predicated to $\mathcal{P}$. (3) If $\mathcal{L}(\Gamma)_\pi$ is not well-founded then the π loop could be sustained forever and it is suggested as a potential counter-example to the inevitability/blocking property. If the automaton is symmetric with respect to $\mathcal{L}$, then π will indeed correspond to an infinite execution that never satisfies the desired property. For general automata, as a practical measure, we resort to simulations for identifying real infinite executions corresponding to a suggested counter-example loop.

4.1 Abstraction Refinement for Time-Triggered Hybrid Automata

Checking well-foundedness of LSR for time-triggered LHA can be performed using real arithmetic. Observe that Theorem 6 gives the LSR for time-triggered HA as follows:

$$min_{G_a}[i] \leq B_l e^{\lambda_{l,i} \times C(l)} y.X[i] \wedge$$
$$y'.X[i] \leq \mu_{l,m} B_l e^{\lambda_{l,i} \times C(l)} y.X[i]$$

We observe that the values of $\lambda_{l,i}$ for linear system $\dot{x} = A_l x$ are dependent on the eigenvalues of A_l. Further, the values $\mu_{l,m}$, B_l and min_{G_a} can be calculated given the dynamics and guards of each of the modes. Note that computing these values is simple only because we are dealing with *polynomial functions*. Also, $C(l)$ is provided along with the description of hybrid automata. Hence, the above relation can be simplified to the form as shown:

$$y'.X[i] \leq \frac{y.X[i]}{K_i} \wedge y.X[i] \geq c_i \text{ where } c_i \geq 0, K_i > 0 \quad (6)$$

A useful property of the above structure in Equation (6) is that this structure is closed under composition. Hence, composition of several LSRs for each loop can be easily performed. Further, such a relation is well-founded iff $K > 1$. Also, we have that a relation $R_1 \triangleq \{y'.X[i] \leq \frac{y.X[i]}{K_1} \wedge y.X[i] \geq c_1 \text{ where } c_1 \geq 0, K_1 > 0\}$ is an abstraction of the relation $R_2 \triangleq \{y' \leq \frac{y.X[i]}{K_2} \wedge y.X[i] \geq c_2 \text{ where } c_2 \geq 0, K_2 > 0\}$ whenever $K_1 \geq K_2$ and $c_1 \leq c_2$.

Hence all the required operations— composition, checking well-foundedness, and abstraction with respect to a set of predicates can be computed effectively using real arithmetic for the relations described in Equation (6).

```
R ← ∅; P ← ∅                                                              1
while
  if exists π = v_1 ... v_n s.t. v_1.loc = v_n.loc, ∀i < n, (v_i, v_{i+1}) ∈ Γ   3
    and abs_P(L(Γ)_π) ⊈ R for any R ∈ R then
    if exists R ∈ R such that L(Γ)_π ⊆ R then                             5
      Refine Abstraction
      P_path ← ⋃_{i∈0..n} Preds(L(v_i, v_{i+1}) ∘ ... ∘ L(v_{n-1}, v_n))  7
      P_loop ← Preds(R) ∪
        ⋃_{i∈0..n} Preds(L(v_i, v_{i+1}) ∘ ... ∘ L(v_{n-1}, v_n) ∘ R)     9
      P ← P ∪ P_path ∪ P_loop
    else                                                                  11
      if L(Γ)_π is well-founded relation by a (new) ranking relation R then
        Weaken disjunctive well-founded relation                          13
        R ← R ∪ R
      else                                                                15
        return "Unable to infer the blocking property of A, v_1 ... v_n"
  else                                                                    17
    return "A is blocking"
end                                                                       19
```

Figure 4: Algorithm 1. Abstraction refinement based verification algorithm for blocking properties of hybrid automata.

5. EXPERIMENTS

We have implemented a prototype tool LySHA in MATLAB for verifying *inevitability* of hybrid automata. LySHA is integrated with Mathworks' Simulink/Stateflow (SLSF) modeling environment through HyLink which translates SLSF models to hybrid automata [17].

5.1 Case Studies

We have created a suite of linear time-triggered HA for evaluating LySHA. These automata are similar to the one shown in Figure 1. Each such HA consists of a set of locations Loc and a certain number (n) of continuous variables. When the

[1]The algorithm may fail if we are unable to infer well-founded ness of this relation.

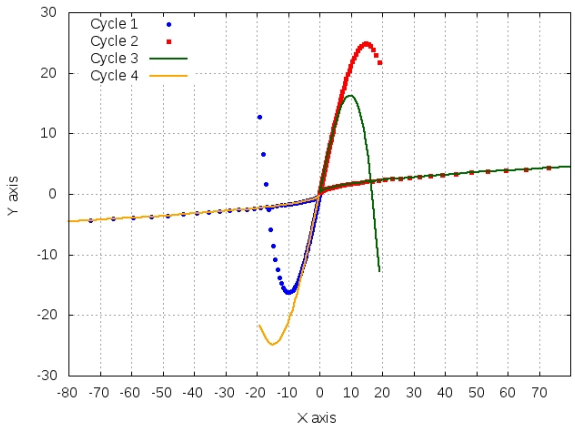

Figure 5: Non-inevitable behavior of the system described in Figure 1 discovered with LySHA . Four (of infinitely many) executions over the two locations and the corresponding values of x and y are shown.

Model $(n, \|L\|)$	Unstable modes	Time Taken (sec)	Is $\mathcal{B}_\epsilon$ inevitable
(2,5)	2	0.01	Yes
(2,20)	5	1.88	No
(2,50)	9	391.85	Yes
(3,20)	5	2.02	Yes
(3,40)	8	100.49	No
(4,20)	5	2.34	No
(4,40)	8	110.34	Yes
(5,20)	5	2.98	No
(5,40)	8	146.28	Yes

Figure 6: Experimental results: (Columns left to right) continuous dimensions (n), number of locations ($|L|$), number of unstable locations, execution time, and answer given by LySHA .

system is in a location $l \in Loc$, the continuous variables evolve according to the linear dynamics $\dot{x} = A_l x$, where x is n-dimensional vector. As shown in Table 6, some of the locations are stable while others are unstable. The system switches from one location to another based on the timer variable now and state predicates. We verify the inevitability property of the system for the set of states $\mathcal{B}_\epsilon$—an ϵ ball around origin with $\epsilon = 0.001$. Given the description of the system in SLSF, HyLink generates the intermediate hybrid automaton representation, which is then used by LySHA .

LySHA computes the LSR for each loop and checks whether its abstraction is well-founded. If not, we refine the *Lyapunov abstraction* by adding a *linearly independent* Lyapunov function to each location and again compute LSR for the loop. After adding n such Lyapunov functions (where n is the dimension of the system), it terminates with a "failure to verify". In such cases, we simulate the loop and check whether the loop satisfies the inevitability property or not. For the hybrid automaton shown in Figure 1, LySHA was unable to infer inevitability for the loop $1 \to 2 \to 1$. Observe in Figure 5, that the loop $1 \to 2 \to 1$ does not satisfy the inevitability property. The figure represents the evolution of the two continuous variables x and y for four different executions of the loop $1 \to 2 \to 1$. Observe that when the system enters into location 2 (sharp change in execution near $(0, 0)$), there is a very rapid growth in the values of x and y. Hence, for each loop, the state of the execution moves farther from origin. This is the reason for the non-inevitable behavior of the system. Further, LySHA confirmed that the loop $1 \to 2 \to 3 \to 1$ satisfies the inevitability property. The well-founded relation in this case is given by the relation:

$$y' \leq \frac{y}{2} \wedge y > 0.001 \tag{7}$$

In general, the technique may produce spurious counter-examples. For example, upon verification of the inevitability property for the HA in Figure 3, LySHA returns an abstract counter-example which does not correspond to any real counter-example. However, since the algorithm is complete for symmetric HA (HA of Figure 2), upon computing the Lyapunov step relation for the loop $1 \to 2 \to 1$, the tool computes the LSR as $(y, y') \in \mathcal{L}(\Gamma)$ iff $y.loc = y'.loc = 1$ and $y.X > 6.25 \wedge y'.X \leq 2$. This relation is well-founded and hence the system satisfies the inevitability property. Note that the algorithm presented for verifying inevitability is sound, i.e. whenever LySHA infers inevitability, it indeed holds.

We have tested LySHA for a variant of Navigation Benchmark with a different number of modes. The results are shown in Table6. Model TTLHA_n_m denotes the time-triggered linear hybrid automaton of n dimensions (i.e. continuous variables) with m locations.

As expected, with a larger number of locations LySHA has to analyze larger number of loops, and the running time increases. Observe that LySHA could prove inevitability of systems even when some of the locations are unstable. Whenever the LySHA inferred that the system does not satisfy inevitability, we checked the behavior of the system through simulations and the results matched with LySHA . Typically, in these examples, the HA fails inevitability because of a loop with several unstable locations.

5.2 Generating Timing Constraints for Inevitability

LySHA can be used for designing *strictly time-triggered* systems, where the linear HA changes its location purely based on the time spent at each location. Consider the scenario where one desires to design a time-triggered LHA for which $\mathcal{B}_\epsilon$ is inevitable. LySHA can aid in computing these timing constraints. LySHA will produce a set of linear inequalities on the time constraints for each location derived the from well-foundedness of a LSR for a loop. If the system is designed in such a way that these timing constraints are satisfied, then the HA is guaranteed to satisfy the inevitability property.

Give the time-triggered system shown in Figure 1, LySHA displays the following set of constraints and the values of the dwell-times of each of these modes:

$$\begin{aligned} -C(1) + 2C(2) - 4C(3) &< -3.991 \\ -C(1) + 2C(2) &< -3.2582 \end{aligned}$$

The two constraints specified in the above equation correspond to well-foundedness of loops $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ and $1 \rightarrow 2 \rightarrow 1$ respectively. We observe that $C(1) = 40$, $C(2) = 16$ and $C(3) = 10$ satisfies these constraints and we can check that the system satisfies the inevitability property. In general, the set of constraints produced by the LySHA can be fed into a linear programming solver to obtain (if feasible) the constraints for inevitability.

6. CONCLUSIONS & FUTURE WORK

Exploiting the relationship between inevitability, program termination and asymptotic stability, in this paper, we have presented a technique for automatic verification of the inevitability property of hybrid automata. We show how Lyapunov function-based abstractions and automated construction of well-founded relations for loops can be combined. The implementation of the resulting algorithm in a software tool (integrated with Simulink/Stateflow) shows promising results in analyzing time-triggered linear HA.

Instantiating the general verification for HA with nonlinear dynamics is an obvious direction for future exploration. For this, we have to use techniques for computing Lyapunov functions for such systems and develop algorithms for composing such Lyapunov relations.

7. REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] A. D. Ames, P. Tabuada, and S. Sastry. On the stability of zeno equilibria. In *Hybrid Systems: Computation and Control (HSCC), volume LNCS 3927, pages 34 Ű 48*, page 3927. Springer-Verlag. 148, 2006.

[3] S. Bogomolov, C. Mitrohin, and A. Podelski. Composing reachability analyses of hybrid systems for safety and stability. In *Proceedings of the 8th international conference on Automated technology for verification and analysis*, ATVA'10, pages 67–81, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] S. Boyd and C. Barratt. *Linear controller design: limits of performance*. Citeseer, 1991.

[5] M. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43:475–482, 1998.

[6] A. Chawdhary, B. Cook, S. Gulwani, M. Sagiv, and H. Yang. Ranking abstractions. In *ESOP'08/ETAPS'08: Proceedings of the Theory and practice of software, 17th European conference on Programming languages and systems*, pages 148–162, Berlin, Heidelberg, 2008. Springer-Verlag.

[7] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16:1512–1542, September 1994.

[8] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *PLDI '06: Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation*, pages 415–426, New York, NY, USA, 2006. ACM.

[9] P. S. Duggirala and S. Mitra. Abstraction-refinement for stability. In *Proceedings of International Conference on Cyber-physical systems (ICCPS 2011)*, Chicago, IL, April 2011.

[10] G. Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In M. Morari and L. Thiele, editors, *HSCC*, volume 3414 of *LNCS*, pages 258–273. Springer, 2005.

[11] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.

[12] J. Hespanha and A. Morse. Stability of switched systems with average dwell-time. In *Proceedings of 38th IEEE Conference on Decision and Control*, pages 2655–2660, 1999.

[13] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005. Also available as Technical Report MIT-LCS-TR-917.

[14] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, New Jersey, 3rd edition, 2002.

[15] D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.

[16] O. Maler and G. Batt. Approximating continuous systems by timed automata. In *Proceedings of the 1st international workshop on Formal Methods in Systems Biology*, FMSB '08, pages 77–89, Berlin, Heidelberg, 2008. Springer-Verlag.

[17] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo. A step towards verification and synthesis from simulink/stateflow models. In *Hybrid Systems: Computation and Control (HSCC 2011)*, 2011.

[18] S. Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.

[19] S. Mitra, D. Liberzon, and N. Lynch. Verifying average dwell time of hybrid systems. *ACM Trans. Embed. Comput. Syst.*, 8(1):1–37, 2008.

[20] A. Podelski and A. Rybalchenko. Transition invariants. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 32–41, Washington, DC, USA, 2004. IEEE Computer Society.

[21] A. Podelski and S. Wagner. Model checking of hybrid systems: From reachability towards stability. In *HSCC*, pages 507–521, 2006.

[22] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *In Proceedings of the 41st IEEE Conf. on Decision and Control*, pages 741–746, 2002.

[23] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *CAV*, 2011.

[24] C. Sloth and R. Wisniewski. Abstraction of continuous dynamical systems utilizing lyapunov functions. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 3760 –3765, dec. 2010.

[25] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, London, 2000.