Verifying Average Dwell Time by Solving Optimization Problems*

Sayan Mitra[1], Nancy Lynch[1], and Daniel Liberzon[2]

[1] Computer Science and AI Laboratory,
Massachusetts Inst. of Technology,
32 Vassar Street, Cambridge, MA 02139
{lynch, mitras}@csail.mit.edu,
[2] Coordinated Science Laboratory,
Univ. of Illinois at Urbana-Champaign,
Urbana, IL 61821
liberzon@uiuc.edu

Abstract. In the switched system model, discrete mechanisms of a hybrid system are abstracted away in terms of an exogenous switching signal which brings about the mode switches. The *Average Dwell time (ADT)* property defines restricted classes of switching signals which provide sufficient conditions for proving stability of switched systems. In this paper, we use a specialization of the Hybrid I/O Automaton model to capture both the discrete and the continuous mechanisms of hybrid systems. Based on this model, we develop methods for automatically verifying ADT properties and present simulation relations for establishing equivalence of hybrid systems with respect to ADT. Given a candidate ADT for a hybrid system, we formulate an optimization problem; a solution of this problem either establishes the ADT property or gives an execution fragment of the system that violates it. For two special classes of hybrid systems, we show that the corresponding optimization problems can be solved using standard mathematical programming techniques. We formally define equivalence of two hybrid systems with respect to ADT and present a simulation relation-based method for proving this equivalence. The proposed methods are applied to verify ADT properties of a linear hysteresis switch and a nondeterministic thermostat.

1 Introduction

In order to accurately represent hybrid phenomena that arise in typical applications, hybrid system models must provide discrete and continuous valued state variables and must have mechanisms to capture both instantaneous state transitions and state trajectories spanning time intervals. The standard approach for describing hybrid behavior is to assume that every state of the system belongs to one of $\mathcal{P}$ *modes*, where $\mathcal{P}$ is a finite index set. When the state is in mode p, for some $p \in \mathcal{P}$, the continuous variables $\mathbf{x}$ evolve according to $\dot{\mathbf{x}} = f_p(\mathbf{x})$ and the discrete variables remain constant. Discrete transitions alter both continuous and discrete variables, which may lead to mode change. Analyzing the stability

* Supported by the MURI project:DARPA/AFOSR MURI F49620-02-1-0325 grant.

of hybrid systems is challenging because the stability of the continuous dynamics of each individual mode does not necessarily imply the stability of the whole system (see [2] for an example). The basic tool for studying stability of hybrid systems relies on the existence of a *Common Lyapunov function*, whose derivative along the trajectories of all the modes must satisfy suitable inequalities. When such a function is not known or does not exist, *Multiple Lyapunov* functions [2] are useful for proving stability of a chosen execution. These and many other stability results are based on the *switched system* [8, 15] view of hybrid systems. Switched systems may be seen as higher-level abstractions of hybrid systems. A switched system model neglects the details of the discrete behavior of a hybrid system and instead relies on an exogenous switching signal to bring about the mode switches. If the individual modes of the system are stable, then the *Dwell Time* [14] and the more general *Average Dwell Time (ADT)* criteria, introduced by Morse and Hespanha [6], define restricted classes of switching signals that guarantee stability of the whole system. In this paper, we present techniques for automatically verifying ADT properties using a model for hybrid systems that captures both their discrete and continuous mechanisms. Thus we provide a missing piece in the toolbox for analysis of stability of hybrid systems.

We use the *Hybrid Input/Output Automaton (HIOA)* framework of Lynch, Segala, and Vaandrager [10] to develop methods for checking ADT properties, which in turn provide sufficient conditions for proving stability of hybrid systems. A hybrid system $\mathcal{A}$ has ADT τ_a if, in every execution fragment of $\mathcal{A}$, any τ_a interval of time, on an average, has at most one mode switch. A large ADT means that the system spends enough time in each mode to dissipate the transient energy gained through mode switches. Having a large ADT itself is not sufficient for stability; in addition, the individual modes of the automaton must also be stable. In fact, the problem of testing the stability of a hybrid system can be broken down into (a) finding Lyapunov functions for the individual modes and (b) checking the ADT property. We assume that a solution to part (a)—a set of Lyapunov functions for the individual modes—is known from existing techniques, and we present automatic methods for part (b).

Our approach for checking if a given automaton $\mathcal{A}$ has ADT τ_a, is to formulate an optimization problem $\mathsf{OPT}(\tau_a)$. From the solution of $\mathsf{OPT}(\tau_a)$ we can either get a counterexample execution fragment of $\mathcal{A}$ that violates the ADT property τ_a, or else we can conclude that no such counterexample exists and that $\mathcal{A}$ has ADT τ_a. We show that for certain useful classes of HIOA, $\mathsf{OPT}(\tau_a)$ can indeed be formulated and solved using standard mathematical programming techniques. In [11], an invariant-based method for proving ADT is proposed. This method transforms the given automaton $\mathcal{A}$ to a new automaton $\mathcal{A}_{\tau_a}$, so that $\mathcal{A}$ has ADT τ_a if and only if $\mathcal{A}_{\tau_a}$ has a particular invariant property $\mathcal{I}_{\tau_a}$. This method is applicable to any HIOA; however, for general HIOA, the invariant $\mathcal{I}_{\tau_a}$ cannot be checked automatically. The optimization-based approach presented here is automatic and complements the invariant method of [11] because the two can be used in combination to find the ADT of hybrid systems. We can start with some candidate value of $\tau_a > 0$ and search for a counterexample execution fragment

for it, using the optimization-based approach. If such an execution fragment is found, then we decrease τ_a (say, by a factor of 2) and try again. If eventually the optimization approach fails to find a counterexample execution fragment for a particular value of τ_a, then we use the invariant approach to prove that this value of τ_a is an ADT for the given system.

Contributions and overview. In Section 2 we introduce a new model for hybrid systems called *Structured Hybrid Automaton (SHA)*. We define stability and the Average Dwell Time (ADT) property of SHA. In Section 3 we introduce the optimization problem $\mathsf{OPT}(\tau_a)$, which searches for an execution fragment of the given SHA that violates the ADT property in question. We formally define what it means for one SHA to switch faster than another and for two SHA to be ADT-equivalent. We define a new type of simulation relation, called *switching simulation*, that provides sufficient conditions for establishing the "faster than" and the equivalence relationships between SHA. In Section 4 we explore the class of Graph SHA, for which solving $\mathsf{OPT}(\tau_a)$ reduces to detecting a negative cost cycle in a weighted graph. We verify the ADT property of a linear, scale-independent hysteresis switch taken from [5] by first finding a Graph SHA $\mathcal{B}$ that is ADT-equivalent to it and then showing how $\mathsf{OPT}(\tau_a)$ for $\mathcal{B}$ can be solved efficiently using standard graph algorithms, like the Bellman-Ford algorithm [3]. In Section 5, we study a more general class of SHA, called Initialized SHA and show that $\mathsf{OPT}(\tau_a)$ can be solved by detecting a cyclic execution fragment with "extra" mode switches. We show that for rectangular initialized SHA, $\mathsf{OPT}(\tau_a)$ can be formulated as a Mixed Integer Linear Program. We use this formulation along with switching simulations to verify the ADT property of a nondeterministic thermostat.

2 Hybrid system model and stability definitions

The Hybrid Input/Output Automaton (HIOA) model [10] with its invariant and simulation based proof methods has been used to verify the safety properties of several hybrid systems (see, e.g., [9, 13, 4]). In this paper, we are concerned with internal stability of hybrid systems, so we use a specialization of the HIOA model called Structured Hybrid Automata (SHA), that does not have input/output variables and does not distinguish among input, output, and internal actions. On the other hand, SHA have extra structure called "state models" for describing the trajectories using differential and algebraic equations.

2.1 Structured Hybrid Automaton model

We denote the domain of a function f by $f.dom$. For a set $S \subseteq f.dom$, we write $f \lceil S$ for the restriction of f to S. If f is a function whose range is a set of functions containing Y, then we write $f \downarrow Y$ for the function g with $g.dom = f.dom$ such that for each $c \in g.dom$, $g(c) = f(c) \lceil Y$. For a tuple or an array b with n elements, we refer to its i^{th} element by $b[i]$.

We fix the *time axis* T to be $\mathsf{R}_{\geq 0}$. Let X be a set of state variables; X is partitioned into X_d, the set of discrete variables, and X_c, the set of continuous

variables. Each variable $x \in X$ is associated with a *type*, which is the set of values that x can assume. Each $x \in X_d$ (respectively, X_c) has *dynamic type*, which is the pasting closure of the set of constant (resp. continuous) functions[3] from left-closed intervals in T to the type of x. A *valuation* $\mathbf{x}$ for the set of variables X is a function that associates each $x \in X$ to a value in its type. The set of all valuations of X is denoted by $val(X)$. A *trajectory* $\tau : J \to val(X)$ specifies the values of all variables X on a time interval J with left endpoint of J equal to 0, with the constraint that evolution of each $x \in X$ over the trajectory should be consistent with its dynamic type. A trajectory with domain $[0, 0]$ is called a *point trajectory*. If $\tau.dom$ is right closed then τ is *closed* and its *limit time* is the supremum of $\tau.dom$ and is written as $\tau.ltime$. The *first valuation* of τ, $\tau.fval$ is $\tau(0)$, and if τ is closed, then the *last valuation* of τ, $\tau.lval$, is $\tau(\tau.ltime)$.

Definition 1. *A* state model *F for a set of variables X is a set of differential equations for X_c of the form $\dot{\mathbf{x}}_c = f(\mathbf{x}_c)$, such that: (1) For every $\mathbf{x} \in val(X)$, there exists a trajectory τ with $\tau.fval = \mathbf{x}$, with the property that $\tau \downarrow X_c$ satisfies F, and (2) for all $t \in \tau.dom$, $(\tau \downarrow X_d)(t) = (\tau \downarrow X_d)(0)$. The prefix and suffix closure of the set of trajectories of X that satisfy the above conditions is denoted by $traj(X, F)$.*

Definition 2. *A* Structured Hybrid Automaton *(SHA) is a tuple $\mathcal{A} = (X, Q, \Theta, A, \mathcal{D}, P)$, where (1) X is a set of* variables*, including a special discrete variable called* mode*. (2) $Q \subseteq val(X)$ is the set of states, (3) $\Theta \subseteq Q$ is a nonempty set of* start states*, (4) A is a set of* actions*, (5) $\mathcal{D} \subseteq Q \times A \times Q$ is a set of* discrete transitions*, and (6) P is an indexed family F_i, $i \in \mathcal{P}$, of state models, where $\mathcal{P}$ is an index set.*

A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$ or as $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ when $\mathcal{A}$ is clear from the context. A transition $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ is a *mode switch* if $\mathbf{x} \lceil mode \neq \mathbf{x}' \lceil mode$. The set of *mode switching transitions* is denoted by M. The *guard predicate* of action a is $G_a \triangleq \{\mathbf{x} \in Q \mid \exists \mathbf{x}', \mathbf{x} \xrightarrow{a} \mathbf{x}' \in \mathcal{D}\}$. In this paper, we assume that the right hand sides of the differential equations in the state models are well behaved (locally Lipschitz), and the differential equations have solutions defined globally in time. Therefore, for each F_i, $i \in \mathcal{P}$ and $\mathbf{x} \in Q$ with $\mathbf{x} \lceil mode = i$, there exists a trajectory τ starting from $\mathbf{x}$ that satisfies F_i and if, $\tau.dom$ is finite then $\tau.lval \in G_a$ for some $a \in A$. The set $\mathcal{T}$ of trajectories of SHA $\mathcal{A}$ is defined as $\mathcal{T} \triangleq \bigcup_{i \in \mathcal{P}} traj(X, F_i)$. An *execution fragment* of an SHA $\mathcal{A}$ is an alternating sequence of actions and trajectories $\alpha = \tau_0 a_1 \tau_1 a_2 \ldots$, where (1) each $\tau_i \in \mathcal{T}$, and (2) if τ_i is not the last trajectory then $\tau_i.lstate \xrightarrow{a_{i+1}} \tau_{i+1}.fstate$. The *first state* of an execution fragment α, $\alpha.fstate$, is $\tau_0.fstate$. An execution fragment α is an *execution* of $\mathcal{A}$ if $\alpha.fstate \in \Theta$. The *length* of a finite execution fragment α is the number of actions in α. An execution fragment is *closed* if it is a finite sequence, and the domain of the last trajectory is closed. Given a closed

[3] This set of functions must be closed under time-shift, restriction to subintervals, and pasting. See [7] for formal definition of these closure properties

execution fragment $\alpha = \tau_0, a_1, \ldots, \tau_n$, its *last state*, $\alpha.lstate$, is $\tau_n.lstate$ and its *limit time* is defined as $\sum_i^n \tau_i.ltime$. We define the following shorthand notation for the valuation of the variables of $\mathcal{A}$ at $t \in [0, \alpha.ltime]$: $\alpha(t) = \alpha'.lstate$, where α' is the longest prefix of α with $\alpha'.ltime = t$. A state $\mathbf{x} \in Q$ is *reachable* if it is the last state of some execution of $\mathcal{A}$. An execution fragment α is *reachable* if $\alpha.fstate$ is reachable. A closed execution fragment α of SHA $\mathcal{A}$ is a *cycle* if $\alpha.fstate = \alpha.lstate$.

2.2 Stability and Average Dwell Time

Stability is a property of the continuous variables of SHA $\mathcal{A}$ with respect to the standard Euclidean norm in R^n. At a given state $\mathbf{x} \in Q$, we write the norm of the continuous variables $|\mathbf{x} \lceil X_c|$ in short as $|\mathbf{x}|$. We assume that for each $i \in \mathcal{P}$, the origin is an equilibrium point for the state model $\dot{\mathbf{x}}_c = f_i(\mathbf{x}_c)$ of $\mathcal{A}$.

SHA $\mathcal{A}$ is *stable* (also called stable in the sense of Lyapunov), if for every $\epsilon > 0$, there exists a $\delta > 0$, such that for every closed execution α of $\mathcal{A}$, for all $t \in [0, \alpha.ltime]$, $|\alpha(0)| \leq \delta$ implies $|\alpha(t)| \leq \epsilon$. $\mathcal{A}$ is *asymptotically stable* if it is stable and δ can be chosen so that, $|\alpha(0)| \leq \delta$ implies $\alpha(t) \to 0$ as $t \to \infty$. If the above condition holds for all δ then $\mathcal{A}$ is *globally asymptotically stable*.

Uniform stability guarantees that the stability property in question holds for execution fragments and not only for executions. $\mathcal{A}$ is *uniformly stable* if for every $\epsilon > 0$ there exists a constant $\delta > 0$, such that for any execution fragment α, $|\alpha.fstate| \leq \delta$ implies $|\alpha.lstate| \leq \epsilon$. An SHA $\mathcal{A}$ is said to be *uniformly asymptotically stable* if it is uniformly stable and there exists a $\delta > 0$, such that for every $\epsilon > 0$ there exists a T, such that for any execution fragment α with $\alpha.ltime \geq T$, $\forall t \geq T$, $|\alpha.fstate| \leq \delta$ implies $|\alpha(t)| \leq \epsilon$. It is said to be *globally uniformly asymptotically stable* if the above holds for all δ, with $T = T(\delta, \epsilon)$.

It is well known that a hybrid system is stable if all the individual modes of the system are stable and the switching is sufficiently slow, so as to allow the dissipation of the transient effects after each switch. The *dwell time* [14] and the *average dwell time* [6] criteria define restricted classes of switching patterns, based on switching speeds, and one can conclude the stability of a system with respect to these restricted classes.

Definition 3. *Given a duration of time $\tau_a > 0$, SHA $\mathcal{A}$ has* Average Dwell Time (ADT) *τ_a if there exists a positive constant N_0, such that for every reachable execution fragment α, $N(\alpha) \leq N_0 + \alpha.ltime/\tau_a$, where $N(\alpha)$ is the number of mode switches in α. The number of* extra switches *of α with respect to τ_a is $S_{\tau_a}(\alpha) \triangleq N(\alpha) - \alpha.ltime/\tau_a$.*

Theorem 1 from [6], adapted to SHA, gives a sufficient condition for stability based on ADT. Roughly, it states that a hybrid system is stable if the modes are individually stable and the switches do not occur too frequently on the average. See Section 3.2 of [8] for a proof.

Theorem 1. *If there exist positive definite, radially unbounded, and continuously differentiable functions $\mathcal{V}_i : R^n \to R^n$, for each $i \in \mathcal{P}$, and positive numbers λ_0 and μ such that:*

$$\frac{\partial \mathcal{V}_i}{\partial \mathbf{x}_c} f_i(\mathbf{x}_c) \leq -\lambda_0 \mathcal{V}_i(\mathbf{x}_c), \quad \forall \mathbf{x}_c, \quad \forall i \in \mathcal{P}, \text{and}$$
$$\mathcal{V}_i(\mathbf{x}'_c) \leq \mu \mathcal{V}_j(\mathbf{x}_c), \quad \forall \mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}', \text{ where } i = \mathbf{x}' \lceil \text{mode and } j = \mathbf{x} \lceil \text{mode}.$$

Then, $\mathcal{A}$ is globally uniformly asymptotically stable if it has an ADT $\tau_a > \frac{\log \mu}{\lambda_0}$.

This stability condition effectively allows us to decouple the construction of Lyapunov functions—the $\mathcal{V}_i$'s for each $i \in \mathcal{P}$, which we assume are known from available methods of system theory—from the problem of checking that the automaton has a certain ADT, which we discuss in the rest of the paper.

3 ADT: Optimization and Equivalence

From Definition 3 it follows that a given $\tau_a > 0$ is *not* an ADT of a given SHA $\mathcal{A}$ if and only if, for every $N_0 > 0$ there exists a reachable execution fragment α of $\mathcal{A}$ such that $S_{\tau_a}(\alpha) > N_0$. Thus, if we solve the following optimization problem:

$$\mathsf{OPT}(\tau_\mathsf{a}): \qquad \alpha^* \in \arg\max S_{\tau_a}(\alpha)$$

over all the execution fragments of $\mathcal{A}$, and the optimal value $S_{\tau_a}(\alpha^*)$ turns out to be bounded, then we can conclude that $\mathcal{A}$ has ADT τ_a, with $S_{\tau_a}(\alpha^*)$ as the corresponding value of the constant N_0. Otherwise, if $S_{\tau_a}(\alpha^*)$ is unbounded and α^* is reachable then we can conclude that τ_a is not an ADT for $\mathcal{A}$. However, the optimization problem $\mathsf{OPT}(\tau_a)$ may not be solvable using standard mathematical programming tools because, among other things, the executions of $\mathcal{A}$ may not have any finite descriptions. In Sections 4 and 5 we study particular classes of SHA for which $\mathsf{OPT}(\tau_a)$ can be solved, but prior to that we define what it means for two SHA to be ADT-equivalent and we develop a simulation relation-based technique for proving this relationship.

Definition 4. *Consider SHA $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{B}$ is* faster *than $\mathcal{A}$, written as $\mathcal{B} \geq_{ADT} \mathcal{A}$, if for all $\tau_a > 0$, τ_a is an ADT for $\mathcal{B}$ implies that τ_a is an ADT for $\mathcal{A}$. If $\mathcal{B} \geq_{ADT} \mathcal{A}$ and $\mathcal{A} \geq_{ADT} \mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ are* ADT-equivalent*, and this is written as $\mathcal{A} =_{ADT} \mathcal{B}$.*

Definition 5. *A* switching simulation relation *from $\mathcal{A}$ to $\mathcal{B}$ is a relation $\mathcal{R} \subset Q_\mathcal{A} \times Q_\mathcal{B}$ satisfying the following conditions:*

1. *If $\mathbf{x} \in \Theta_\mathcal{A}$ then there exists $\mathbf{y} \in \Theta_\mathcal{B}$ of $\mathcal{B}$ such that $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$.*
2. *If $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$ and α is an execution fragment of $\mathcal{A}$ consisting of a single action surrounded by two point trajectories with $\alpha.fstate = \mathbf{x}$, then $\mathcal{B}$ has a closed execution fragment β, such that $\beta.fstate = \mathbf{y}$, $N(\beta) \geq N(\alpha)$, $\beta.ltime = 0$, and $\alpha.lstate\ \mathcal{R}\ \beta.lstate$. Here $N(\beta)$ is the number of mode switches in β.*

3. *If* $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$ *and* α *is an execution fragment of* $\mathcal{A}$ *consisting of a single closed trajectory with* $\alpha.fstate = \mathbf{x}$*, then* $\mathcal{B}$ *has a closed execution fragment* β*, such that* $\beta.fstate = \mathbf{y}$*,* $\beta.ltime \leq \alpha.ltime$*, and* $\alpha.lstate\ \mathcal{R}\ \beta.lstate$*.*

Lemma 1. *Let* $\mathcal{R}$ *be a switching simulation relation from SHA* $\mathcal{A}$ *to* $\mathcal{B}$*. Then, for all* $\tau_a > 0$ *and for every reachable execution fragment* α *of* $\mathcal{A}$*, there exists a reachable execution fragment* β *of* $\mathcal{B}$*, such that* $S_{\tau_a}(\beta) \geq S_{\tau_a}(\alpha)$*.*

Owing to space limitations most of the proofs are omitted from this paper; complete proofs for all the results are available in the full version [12]. The above lemma is proved by inductively defining a sequence $\beta_0\beta_1\beta_2\ldots$ of closed execution fragments of $\mathcal{B}$ for a given an execution fragment $\alpha = \tau_0 a_1 \tau_1 a_2 \tau_2 \ldots$ of $\mathcal{A}$, such that for all i, $\beta_i.lstate = \beta_{i+1}.fstate$, $\alpha_i.lstate\ \mathcal{R}\ \beta_i.lstate$, and $S_{\tau_a}(\beta) \geq S_{\tau_a}(\alpha)$. We use Property 2 of the definition of switching simulation for the construction of the β_i's with i even. This gives us $\beta_i.ltime \leq \alpha_i.ltime$ for every even i. We use Property 1 of the definition of switching simulation for the construction of the β_i's with i odd. This gives us $\beta_i.ltime = \alpha_i.ltime$ and $N(\beta_i) \geq N(\alpha_i)$ for every odd i.

Theorem 2 states that the existence of a switching simulation relation from $\mathcal{A}$ to $\mathcal{B}$ is sufficient to establish that $\mathcal{B}$ is faster than $\mathcal{A}$.

Theorem 2. *If* $\mathcal{R}$ *is a switching simulation relation from* $\mathcal{A}$ *to* $\mathcal{B}$*, then* $\mathcal{B} \geq_{ADT} \mathcal{A}$*.*

Corollary 1. *Let* $\mathcal{R}_1$ *be a switching simulation from* $\mathcal{A}$ *to* $\mathcal{B}$ *and* $\mathcal{R}_2$ *be a switching simulation from* $\mathcal{B}$ *to* $\mathcal{A}$*. Then,* $\mathcal{A} =_{ADT} \mathcal{B}$*.*

4 Graph SHA and negative cost cycles

We introduce a special class of SHA, called Graph SHA for which $\mathsf{OPT}(\tau_a)$ (see Section 3) can be solved using classical graph algorithms. We show how Graph SHA and switching simulations can be used to prove ADT properties of a linear, scale-independent hysteresis switch.

Graph SHA model. Consider a graph G defined by: a set of vertices $\mathcal{V}$, a set of directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, a cost function $w : \mathcal{E} \to R_{\geq 0}$ for the edges, and a special start edge $e_0 \in \mathcal{E}$. The cost of a path in G is the sum of the costs of the edges in the path. Given $G = (\mathcal{V}, \mathcal{E}, w, e_0)$, the corresponding Graph SHA $Aut(G)$ is specified by the code in Figure 1. The source and the target vertices of an edge e are denoted by $e[1]$ and $e[2]$, respectively. The discrete transitions are written using the standard precondition-effect style. Each *trajdef* d defines a set of trajectories $\mathcal{T}_d$ in terms of the invariant $inv(d)$, the stopping condition $stop(d)$, and the state model written as an evolve clause $evolve(d)$. A trajectory τ is in $\mathcal{T}_d$, if an only if (1) τ satisfies $evolve(d)$, (2) $\forall\ t \in \tau.dom$, $\tau(t) \in inv(d)$, and (3) $\exists\ t \in \tau.dom$, $\tau(t) \in stop(d) \to t = \tau.ltime$. The set of trajectories of $Aut(G)$ is the union of the sets of trajectories defined by each trajdef.

Intuitively, the state of $Aut(G)$ captures the motion of a particle moving with unit speed along the edges of the graph G. The position of the particle is given by

the *mode*, which is the edge it resides on, and the value of x, which is its distance from the source vertex of the edge G. Thus, a switch from mode e to mode e' of $Aut(G)$ corresponds to the particle arriving at vertex $e[2]$ via edge e, and departing on edge e'. Within edge e the particle moves at unit speed from $e[1]$, where $x = 0$ to $e[2]$, where $x = w(e)$. The next theorem implies that to search

Variables:
$mode \in \mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, *initialy* e_0
$x \in R$, *initially* 0

Actions:
switch(e,e'), e,$e' \in \mathcal{E}$

Transitions:
switch(e,e')

Precondition
$mode = e \wedge e[2] = e'[1] \wedge x = w(e)$
Effect
$mode \leftarrow e'$, $x \leftarrow 0$

Trajectories:
Trajdef edge($mode$)
Evolve $d(x) = 1$
Invariant $x \leq w(mode)$
Stop when $x = w(mode)$

Fig. 1. Graph SHA $Aut(G)$, where $G = (\mathcal{V}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}, w : \mathcal{E} \rightarrow R_{\geq 0}$, and $e_0)$

for an execution of $Aut(G)$ that violates a ADT property τ_a, it is necessary and sufficient to search over the space of the cycles of G. See [12] for a proof.

Theorem 3. *Consider $\tau_a > 0$ and a graph SHA $Aut(G)$. $\mathcal{A}$ has average dwell time τ_a if and only if for all $m > 1$, the cost of any reachable cycle of G with m segments is at least $m\tau_a$.*

Thus, the problem of solving $\mathsf{OPT}(\tau_a)$ for $Aut(G)$ reduces to the problem of checking whether G contains a cycle of length m, for some $m > 1$, with cost less than $m\tau_a$. We can solve this problem as follows: we scale the weight of every edge e of G to be $\frac{w(e)}{\tau_a} - 1$, and check for the existence of a negative cost cycle in the new graph. Existence of a negative cost cycle in any graph $G = (\mathcal{V}, \mathcal{E})$ can be detected efficiently using the Bellman-Ford algorithm [3] in $O(\mathcal{V}\mathcal{E})$ time.

Example 1: Linear hysteresis switch. We verify the ADT properties of a linear, scale-independent hysteresis switch which is a subsystem of an adaptive supervisory control system taken from [5] (also Chapter 6 of [8]). An adaptive supervisory controller consists of a family of candidate controllers $u_i, i \in \mathcal{P}$, which correspond to the parametric uncertainty range of the plant in a suitable way. The controller operates in conjunction with a set of on-line estimators that provide *monitoring signals* $\mu_i, i \in \mathcal{P}$; intuitively, smallness of μ_i indicates high likelihood that i is the actual parameter value. Based on these signals, the switching logic unit changes the variable *mode*, which in turn determines the controller to be applied to the plant. Average dwell time property of this switching logic guarantees stability of the overall supervisory control system.

In building the linear SHA model $\mathcal{A}$ (shown in Figure 2), we consider monitoring signals generated by linear differential equations, such that for each $i \in \mathcal{P}$, if $mode = i$, then $d(\mu_i) = c_i\mu_i$, otherwise $d(\mu_i) = 0$. The switching logic unit

implements scale independent hysteresis switching as follows: at an instant of time when controller k is operating, that is, $mode = k$ for some $k \in \mathcal{P}$, if there exists a $i \in \mathcal{P}$ such that $\mu_i(1+h) \leq \mu_k$ for some fixed hysteresis constant h, then the switching logic sets $mode = i$ and applies output of controller i to the plant.

Variables:
$mode \in \mathcal{P}$, *initially* p_0
$\mu_p \in R$, $p \in \mathcal{P}$,
initialy $\mu_{p_0} = (1+h)C_0$
for all $i \neq p_0$, $\mu_i = C_0$
derived
$\mu_{min} = Min_{i \in \mathcal{P}} \mu_i$

Actions:
switch(p,q), p,$q \in \mathcal{P}$

Transitions:
switch(p,q)
Precondition
$mode = p \wedge (1+h)\mu_q \leq \mu_p$
Effect $mode \leftarrow q$

Trajectories:
Trajdef mode(p)
Evolve *for all* $i \in \mathcal{P}$,
if $i = p$ **then** $d(\mu_p) = c_p \mu_p$ **else** $d(\mu_i) = 0$
Stop when
$\exists\ q \in \mathcal{P}$ **such that** $(1+h)\mu_q \leq \mu_p$

Fig. 2. Linear hysteresis switch with parameters $\mathcal{P}, C_0, h$ and c_i for each $i \in \mathcal{P}$.

The switching behavior of $\mathcal{A}$, does not depend on the value of the μ_i's, but on the ratio of $\frac{\mu_i}{\mu_{min}}$, which is always within $[1,(1+h)]$. When $\mathcal{A}$ is in mode $p \in \mathcal{P}$, all the ratios remain constant, except $\frac{\mu_p}{\mu_{min}}$ which increases monotonically from 1 to either $(1+h)$ or to $(1+h)^2$, in time $\frac{1}{c_p}\ln(1+h)$ or $\frac{2}{c_p}\ln(1+h)$, respectively. Now we specify a Graph automaton $\mathcal{B}$, in terms of the parameters of $\mathcal{A}$, that captures the switching behavior of $\mathcal{A}$. Consider a graph $G = (\mathcal{V}, \mathcal{E}, w, e_0)$, where:

1. $\mathcal{V} \subset \{1,(1+h)\}^n$, such that for any $v \in V$, all the n-components are not equal. We denote the i^{th} component of $v \in V$, by $v[i]$.
2. An edge $(u,v) \in \mathcal{E}$ if and only if, one of the following conditions hold:
   (a) There exists $j \in \{1,\ldots,n\}$, such that, $u[j] \neq v[j]$ and for all $i \in \{1,\ldots,n\}$, $i \neq j$, $u[i] = v[i]$. The cost of the edge $w(u,v) \stackrel{\Delta}{=} \frac{1}{c_j}\ln(1+h)$ and we define $\zeta(u,v) \stackrel{\Delta}{=} j$.
   (b)There exists $j \in \{1,\ldots,n\}$ such that $u[j] = 1, v[j] = (1+h)$ and for all $i \in \{1,\ldots,n\}$, $i \neq j$ implies $u[i] = (1+h)$ and $v[i] = 1$. The cost of the edge $w(u,v) \stackrel{\Delta}{=} \frac{2}{c_j}\ln(1+h)$ and we define $\zeta(u,v) \stackrel{\Delta}{=} j$.
3. $e_0 \in \mathcal{E}$, such that $e_0[1][p_0] = (1+h)$ and for all $i \neq p_0$, $e_0[1][i] = 1$.

As an example, the graph for $n = 3$ is shown in Figure 3. Let $\mathcal{B}$ be the Graph SHA $Aut(G)$. Each edge of G corresponds to a mode of $\mathcal{A}$. In fact, *mode* of $\mathcal{A}$ equals $\zeta(e)$, where e is the edge corresponding to the *mode* of $\mathcal{B}$. We define a relation $\mathcal{R}$ on the state spaces on $\mathcal{A}$ and $\mathcal{B}$. Each vertex of G is an n-tuple; the i^{th} component of the source vertex of e is denoted by $e[1][i]$.

Definition 6. *For any* $\mathbf{x} \in Q_{\mathcal{A}}$ *and* $\mathbf{y} \in Q_{\mathcal{B}}$, $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$ *if and only if:*

1. $\zeta(\mathbf{y} \lceil mode) = \mathbf{x} \lceil mode$
2. *For all* $j \in \{1,\ldots,n\}$, *if* $j = \zeta(\mathbf{y} \lceil mode)$ *then (a)* $\frac{\mathbf{x}\lceil\mu_j}{\mathbf{x}\lceil\mu_{min}} = e^{c_j(\mathbf{y}\lceil x)}$ *else (b)* $\frac{\mathbf{x}\lceil\mu_j}{\mathbf{x}\lceil\mu_{min}} = (\mathbf{y} \lceil mode)[1][j]$ *and* $\frac{\mathbf{x}\lceil\mu_j}{\mathbf{x}\lceil\mu_{min}} = (\mathbf{y} \lceil mode)[2][j]$.

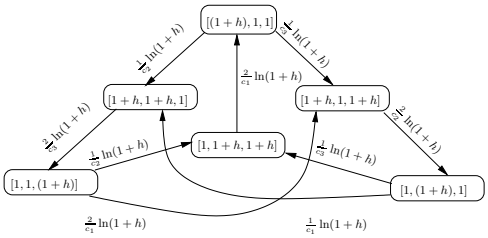

Fig. 3. Graph G with 3 modes. Here h and c_i's are the parameters from the hysteresis switch automaton $\mathcal{A}$.

Part 1 of Definition 6 states that if $\mathcal{A}$ is in *mode* j and $\mathcal{B}$ is in mode e, then $\zeta(e) = j$. Part 2 states that for all $j \neq \zeta(e)$, the j^{th} component of $e[1]$ and $e[2]$ are the same, and are equal to μ_j/μ_{min}, and for $j = \zeta(e)$, $\mu_j = \mu_{min}e^{c_j x}$. The next lemma states that $\mathcal{R}$ is a switching simulation relation from $\mathcal{A}$ and $\mathcal{B}$ and from $\mathcal{B}$ to $\mathcal{A}$. The first part is proved by showng that every state of $\mathcal{A}$ is related to some state of $\mathcal{B}$ and that every action and trajectory of $\mathcal{A}$ can be emulated by an execution fragment of $\mathcal{B}$ with more extra switches. The second part is proved using identical steps by interchanging $\mathcal{A}$ and $\mathcal{B}$.

Lemma 2. *$\mathcal{R}$ is a switching simulation relation from $\mathcal{A}$ to $\mathcal{B}$ and from $\mathcal{B}$ to $\mathcal{A}$.*

Remark 1. From Corollary 1 it follows that SHA $\mathcal{A}$ and the graph SHA $\mathcal{B}$ are ADT-equivalent. We already know that ADT properties of $\mathcal{B}$ can be automatically verified using Bellman-Ford algorithm. Therefore, we can also verify ADT of $\mathcal{A}$ automatically. The ADT of a linear hysteresis switch with $\mathcal{P}$ modes can be checked in $O(2^{|\mathcal{P}|})$ time.

5 Initialized SHA and mixed integer linear programming

Given $\tau_a > 0$ and an Initialized SHA $\mathcal{A}$, we show that existence of a cycle of $\mathcal{A}$ with extra switches with respect to τ_a is necessary and sufficient to conclude that τ_a is not an ADT for $\mathcal{A}$. We apply this result to rectangular initialized SHA to solve the ADT verification problem using Mixed Integer Linear Programming (MILP).

Initialized SHA model. A SHA $\mathcal{A}$ is *initialized* if every $a \in M$ is associated with a set $R_a \subseteq Q$, such that $\mathbf{x} \xrightarrow{a} \mathbf{x}'$ is a mode switching transition if and only if $\mathbf{x} \in G_a$ and $\mathbf{x}' \in R_a$. The set R_a is called the *initialization predicate* of a. We assume that all cyclic execution fragments of initialized SHA $\mathcal{A}$ are reachable. A SHA is *rectangular* if the differential equations in the state models have constant right hand sides, and the guard and the initialization predicates (restricted to the set of continuous variables) are polyhedra.

Theorem 4 implies that for an initialized SHA $\mathcal{A}$, it is necessary and sufficient to solve $\mathsf{OPT}(\tau_a)$ over the space of the cyclic fragments of $\mathcal{A}$ instead of the larger space of all execution fragments.

Theorem 4. *Given $\tau_a > 0$ and initialized SHA $\mathcal{A}$, τ_a is an ADT for $\mathcal{A}$ if and only if $\mathcal{A}$ does not have any cycles with extra switches with respect to τ_a.*

Here we sketch a proof of this theorem and refer the reader to [12] for the complete proof. Existence of a cycle α of $\mathcal{A}$ with $S_{\tau_a}(\alpha) > 0$ implies that τ_a is not an ADT, because by concatenating many α's we can construct an execution fragment $\alpha \frown \alpha \frown \alpha \ldots$ with an arbitrarily large S_{τ_a}. To prove that the existence of a cycle with extra switches is also necessary for violating the ADT property, we assume that τ_a is not an ADT for $\mathcal{A}$ and that $\mathcal{A}$ does not have any cycles with extra switches. We choose $N_0 > |\mathcal{P}|^3$; from the definition of ADT we know that there exists an execution fragment γ, such that $S_{\tau_a}(\gamma) > N_0$. Let $\alpha = \tau_0 a_1 \tau_1 \ldots \tau_n$ be the shortest such execution fragment. Since $N(\alpha \geq |\mathcal{P}|^3$ mode switches and $\mathcal{A}$ is initialized, α must cintain a cycle. As $\mathcal{A}$ does not have any cycles with extra switches, we get a contradiction to the assumption that α is the shortest execution fragment with more than N_0 extra switches.

The next lemma allows us to limit the search for cycles with extra switches to cycles with at most $|\mathcal{P}|^3$ mode switches. It is proved by showing that any cycle with extra switches that has more than $|\mathcal{P}|^3$ mode switches, can be decomposed into two smaller cycles, one of which must also have extra switches.

Lemma 3. *If initialized SHA $\mathcal{A}$ has a cycle with extra switches, then it has a cycle with extra switches that has fewer than $|\mathcal{P}|^3$ mode switches.*

MILP formulation of OPT(τ_a). Figure 4 shows the specification of a generic Initialized rectangular SHA $\mathcal{A}$. The automaton $\mathcal{A}$ has a single discrete variable called *mode* which takes values in the index set $\mathcal{P} = \{1, \ldots, N\}$, and a continuous variable vector $\mathbf{x} \in R^n$. For any $p, q \in \mathcal{P}$, the action that changes the mode from p to q is called $switch(p, q)$. The guard and the initialization predicates of this action are given by sets of linear inequalities on the continuous variables, represented in the matrix notation by: $G[p, q]\mathbf{x} \leq g[p, q]$ and $R[p, q]\mathbf{x} \leq r[p, q]$, respectively, where $G[p, q]$ and $R[p, q]$ are constant matrices with n columns and $g[p, q], r[p, q]$ are constant vectors. For each mode $p \in P$ of automaton $\mathcal{A}$, the

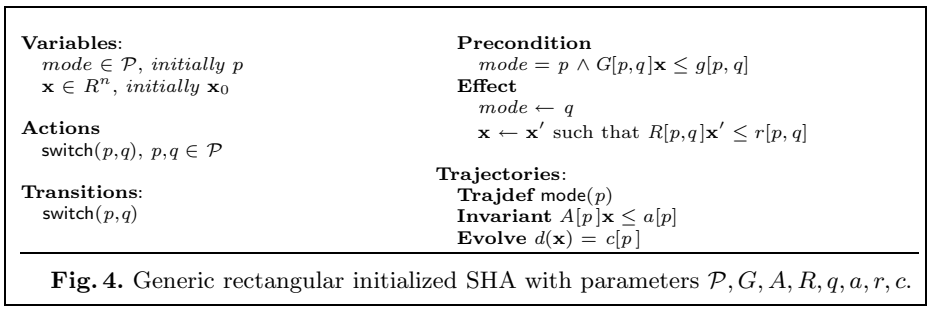

Fig. 4. Generic rectangular initialized SHA with parameters $\mathcal{P}, G, A, R, q, a, r, c$.

invariant is stated in terms of linear inequalities of the continuous variables

$A[p]\mathbf{x} \leq a[p]$, where $A[p]$ is a constant matrix with n columns and $a[p]$ is a constant vector. The evolve clause is given by a single differential equation $d(\mathbf{x}) = c[p]$, where the right hand side $c[p]$ is a constant vector.

We describe a MILP formulation $\mathsf{MOPT}(K, \tau_a)$ for finding a cyclic execution with K mode switches that maximizes the number of extra switches with respect to τ_a. If the optimal value is positive, then the optimal solution represents a cycle with extra switches with respect to τ_a, and we conclude that τ_a is not an ADT for $\mathcal{A}$. On the other hand, if the optimal value is not positive, then we conclude that there are no cycles with extra switches of length K. To verify ADT of $\mathcal{A}$, we solve a sequence of $\mathsf{MOPT}(K, \tau_a)$'s with $K = 2, \ldots, |\mathcal{P}|^3$. If the optimal value is positive for none of these, then we conclude that τ_a is an ADT for $\mathcal{A}$. By adding extra variables and constraints we are able to formulate a single MILP that maximizes the extra switches over all cycles with K or less mode switches, but for simplicity of presentation, we discuss sf MOPT(τ_a) instead of this latter formulation. The following are the decision variables for $\mathsf{MOPT}(K, \tau_a)$; the objective function and the constraints are shown in Figure 5.

- $\mathbf{x}_i \in R^n$, $i \in \{0 \ldots, K\}$, value of continuous variables
- $t_i \in R$, $i \in \{0, 2, 4, \ldots, K\}$, length of i^{th} trajectory
- $m_{ij} = \begin{cases} 1, \text{ if mode over } i^{th} \text{ trajectory is } j \\ 0, \text{ otherwise.} \quad \text{for each } i \in \{0, 2, \ldots, K\},\ j \in \{1, \ldots, N\} \end{cases}$
- $p_{ijk} = \begin{cases} 1, \text{ if mode over } (i-1)^{st} \text{ trajectory is } j \text{ and over } (i+1)^{st} \text{ trajectory is } k \\ 0, \text{ otherwise.} \quad \text{for each } i \in \{0, 2, 4, \ldots, K\}, j, k \in \{1, \ldots, N\} \end{cases}$

In $\mathsf{MOPT}(K, \tau_a)$, an execution fragment with K mode switches is represented as a sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_K$ of K valuations for the continuous variables. For

Objective function: $S_{\tau_a} : \dfrac{K}{2} - \dfrac{1}{\tau_a} \displaystyle\sum_{i=0,2,\ldots}^{K} t_i$

Mode: $\forall\, i \in \{0, 2, \ldots, K\},\ \displaystyle\sum_{j=1}^{N} m_{ij} = 1$ *and* $\forall\, i \in \{1, 3, \ldots, K-1\},\ \displaystyle\sum_{j=1}^{N}\sum_{k=1}^{N} p_{ijk} = 1$ (1)

Cycle: $\mathbf{x}_0 = \mathbf{x}_K$ and $\forall\, j \in \{1, \ldots, N\}, m_{0j} = m_{Kj}$ (2)

Preconds: $\forall\, i \in \{1, 3, \ldots, K-1\},\ \displaystyle\sum_{j=1}^{N}\sum_{k=1}^{N} G[j,k].p_{ijk}.\mathbf{x}_i \leq \sum_{j=1}^{N}\sum_{k=1}^{N} p_{ijk}.g[j,k]$ (3)

Initialize: $\forall\, i \in \{1, 3, \ldots, K-1\},\ \displaystyle\sum_{j=1}^{N}\sum_{k=1}^{N} R[j,k].p_{ijk}.\mathbf{x}_{i+1} \leq \sum_{j=1}^{N}\sum_{k=1}^{N} p_{ijk}.r[j,k]$ (4)

Invariants: $\forall\, i \in \{0, 2, \ldots, K\},\ \displaystyle\sum_{j=1}^{N} A[j].m_{ij}.\mathbf{x}_i \leq \sum_{j=1}^{N} m_{ij}.a[j]$ (5)

Evolve: $\forall\, i \in \{0, 2 \ldots, K\},\ \mathbf{x}_{i+1} = \mathbf{x}_i + \displaystyle\sum_{j=1}^{N} c[j].m_{ij}.t_i$ (6)

Fig. 5. The objective function and the linear and integral constraints for $\mathsf{MOPT}(K, \tau_a)$

each even i, $\mathbf{x}_i$ goes to $\mathbf{x}_{i+1}$ by a trajectory of length t_i. If this trajectory is in mode j, for some $j \in \{1, \ldots, N\}$, then $m_{ij} = 1$, else $m_{ij} = 0$. For each odd i, $\mathbf{x}_i$ goes to $\mathbf{x}_{i+1}$ by a discrete transition. If this transition is from mode j to mode k, for some $j, k \in \{1, \ldots, N\}$, then $p_{ijk} = 1$, else $p_{ijk} = 0$. These constraints are specified by Equation (1) in Figure 5. For each odd i, Constraints (3) and (4) ensure that $(\mathbf{x}_i, \mathit{switch}(j,k), \mathbf{x}_{i+1})$ is a valid mode switching transition. These constraints reduce to the inequalities $G[j,k]\mathbf{x}_i \leq g[j,k]$ and $R[j,k]\mathbf{x}_{i+1} \leq r[j,k]$ which correspond to the guard and the initialization conditions on the pre- and the post-state of the transition. For each even i, $\mathbf{x}_i$ evolves to $\mathbf{x}_{i+1}$ through a trajectory in some mode, say j. Constraint (5) ensures that $\mathbf{x}_i$ satisfies the invariant of mode j described by the inequality $A[j]\mathbf{x}_i \leq a[j]$. An identical constraint for $\mathbf{x}_{i+1}$ is written by replacing $\mathbf{x}_i$ with $\mathbf{x}_{i+1}$ in (5). Since the differential equations have constant right hand sides and the invariants describe pohyhedra in R^n, the above conditions ensure that all the intermediate states in the trajectory satisfy the mode invariant. Equation (6) ensures that, for each even i, $\mathbf{x}_i$ evolves to $\mathbf{x}_{i+1}$ in t_i time according to the differential equation $d(\mathbf{x}) = c[j]$.

Some of these constrains involve nonlinear terms. For example, $m_{ij}\mathbf{x}_i$ in (5) is the product of real variable $\mathbf{x}_i$ and boolean variable m_{ij}. Using the "big M" method [16] we can linearize this equation by replacing $m_{ij}\mathbf{x}_i$ with $\mathbf{y}_i$, and adding the following linear inequalities: $\mathbf{y}_i \geq m_{ij}\delta$, $\mathbf{y}_i \leq m_{ij}\Delta$, $\mathbf{y}_i \leq \mathbf{x}_i - (1 - m_{ij})\delta$, and $\mathbf{y}_i \geq \mathbf{x}_i - (1 - m_{ij})\Delta$, where δ and Δ are the lower and upper bounds on the values of $\mathbf{x}_i$.

Example 2: Thermostat. We use the MILP technique together with switching simulation relations to verify the ADT of a thermostat with nondeterministic switches. The thermostat SHA $\mathcal{A}$ (see Figure 6 *Left*) has two modes l_0, l_1, two continuous variables x and z, and real parameters $h, C, \theta_1, \theta_2, \theta_3, \theta_4$, where $0 < \theta_1 < \theta_2 < \theta_3 < \theta_4 < h$. In l_0 mode the heater is off and the temperature x decreases according to the differential equation $d(x) = -Cx$. While the temperature x is between θ_2 and θ_1, the on action *must* occur. As a result of which the mode changes to l_1. In mode l_1, the heater is on and the x rises according to the $d(x) = C(h - x)$, and while x is between θ_3 and θ_4, the off action must occur. The continuous variable z measures the total time spent in mode l_1

The thermostat SHA $\mathcal{A}$ is neither initialized nor rectangular; however, there is a rectangular initialized SHA $\mathcal{B}$, such that $\mathcal{B} \geq_{ADT} \mathcal{A}$. Consider the SHA $\mathcal{B}$ of Figure 6 (*Right*) with parameters L_0 and L_1. Automaton $\mathcal{B}$ has a reset timer r and two modes l_0 and l_1, in each of which r increases at a unit rate. When r reaches L_i in mode l_i, a switch to the other mode *may* occur and if it does then r is reset to zero. We define a relation $\mathcal{R}$ on the state spaces of $\mathcal{A}$ an $\mathcal{B}$ such that with appropriately chosen values of L_0 and L_1, $\mathcal{B}$ captures the fastest switching behavior of $\mathcal{A}$.

Definition 7. *For any* $\mathbf{x} \in Q_{\mathcal{A}}$ *and* $\mathbf{y} \in Q_{\mathcal{B}}$, $\mathbf{x}\ \mathcal{R}\ \mathbf{y}$ *if and only if: (1)* $\mathbf{x} \lceil \mathit{mode} = \mathbf{y} \lceil \mathit{mode}$, *and (2) if* $\mathbf{x} \lceil \mathit{mode} = l_0$ *then* $\mathbf{y} \lceil r \geq \frac{1}{C} \ln \frac{\theta_3}{\mathbf{x} \lceil x}$ *else* $\mathbf{y} \lceil r \geq \frac{1}{C} \ln \left(\frac{h - \theta_2}{h - \mathbf{x} \lceil x} \right)$.

Lemma 4. *If we set* $L_0 = \frac{1}{C} \ln \frac{\theta_3}{\theta_2}$ *and* $L_1 = \frac{1}{C} \ln \frac{h - \theta_2}{h - \theta_3}$, *then the relation* $\mathcal{R}$ *is a switching simulation from* $\mathcal{A}$ *to* $\mathcal{B}$.

The proof of this lemma is like that of Lemma 2 ; we show that every state of $\mathcal{A}$ is related to some state of $\mathcal{B}$ and that every action and trajectory of $\mathcal{A}$ can be emulated by an execution fragment of $\mathcal{B}$ with more extra switches. Lemma 4

Variables:
$mode \in \{l_0, l_1\}$, *initially* l_0
$x, z \in R$, *initially* $x = \theta_4, z = 0$

Actions
on, off

Transitions:
on **Precondition**
$mode = l_0 \wedge x \leq \theta_2$
Effect $mode \leftarrow l_1$

off **Precondition**
$mode = l_1 \wedge x \geq \theta_3$
Effect $mode \leftarrow l_0$

Trajectories:
Trajdef l_0
Evolve $d(x) = -Cx$; $d(z) = 0$
Invariant $x \geq \theta_1$ **Stop when** $x = \theta_1$

Trajdef l_1
Evolve $d(x) = C(h-x)$; $d(z) = 1$
Invariant $x \leq \theta_4$ **Stop when** $x = \theta_4$

Variables:
$mode \in \{l_0, l_1\}$, *initially* l_0
$r \in R$, *initially* $r = L_1$

Actions
switchto$_i$, $i \in \{0,1\}$

Transitions:
switchto$_1$
Precondition
$mode = l_0 \wedge r \geq L_0$
Effect $mode \leftarrow l_1$, $r \leftarrow 0$

switchto$_0$
Precondition
$mode = l_1 \wedge r \geq L_1$
Effect $mode \leftarrow l_0$, $r \leftarrow 0$

Trajectories:
Trajdef l_0
Evolve $d(r) = 1$

Trajdef l_1
Evolve $d(r) = 1$

Fig. 6. *Left:*Thermostat SHA $\mathcal{A}$ with parameters $\theta_1, \theta_2, \theta_3, \theta_4, C$, and h. *Right:* Rectangular SHA $\mathcal{B}$ with parameters L_0, L_1.

implies that $\mathcal{B} \geq_{ADT} \mathcal{A}$, and so, for any $\tau_a > 0$ if τ_a is an ADT for $\mathcal{B}$ then τ_a is also an ADT for $\mathcal{A}$. Also, $\mathcal{B}$ is rectangular and initialized, so we can use Theorem 4 and MILP technique to check any ADT property of $\mathcal{B}$.

We formulated the $\mathsf{MOPT}(K, \tau_a)$ for automaton $\mathcal{B}$ and used the GNU Linear Programming Kit [1] to solve it. Solving for $K = 4, L_0 = 40, L_1 = 15$, and $\tau_a = 25, 27, 28$, we get optimal costs $-0.4, -4.358E^{-13} (\approx 0)$ and 0.071, respectively. We conclude that the ADT of $\mathcal{B}$ is $\geq 25, \geq 27$, and < 28. Since $\mathcal{B} \geq_{ADT} \mathcal{A}$, we conclude that the ADT of the thermostat is no less than 27.

Remark 2. For finding counterexample execution fragments of proposed ADT properties, the MILP approach can be applied to non-initialized rectangular SHA as well. In such applications, however, the necessity part of Theorem 4 will not hold and from the failure to find a counterexample alone we cannot conclude that the automaton satisfies the ADT property in question.

Conclusions. We have presented optimization-based methods for automatically verifying Average Dwell Time (ADT) properties of certain classes of hybrid systems, which provides a tool for proving (uniform) stability. We have also defined equivalence of hybrid systems with respect to ADT and have presented a simulation relation-based method for proving these equivalence relationships. The

proposed methods have been applied to verify ADT of a linear, scale-independent hysteresis switch and a nondeterministic thermostat.

In this paper we examined internal stability only; however, the input and output variables of HIOA make the framework suitable for studying input-output properties of hybrid systems. Another direction of future research is to extend the ADT verification technique to probabilistic hybrid systems.

Acknowledgments. We would like to thank Debasish Chatterjee for his comments on this paper.

References

1. GLPK - GNU linear programming kit. http://www.gnu.org/directory/libs/glpk.html.
2. M. Branicky. Multiple Lyapunov Functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43:475–482, 1998.
3. T. H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.
4. C. Heitmeyer and N. Lynch. The generalized railroad crossing: A case study in formal verification of real-time system. In *Proceedings of the 15th IEEE Real-Time Systems Symposium*, 1994.
5. J.P. Hespanha, D. Liberzon, and A.S. Morse. Hysteresis-based switching algorithms for supervisory control of uncertain systems. *Automatica*, 39:263–272, 2003.
6. J.P. 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.
7. D. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. The theory of timed I/O automata. Technical Report MIT/LCS/TR-917a, MIT Laboratory for Computer Science, 2004. Available at `http://theory.lcs.mit.edu/tds/reflist.html`.
8. D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.
9. C. Livadas, J. Lygeros, and N. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, 1999.
10. N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, August 2003.
11. S. Mitra and D. Liberzon. Stability of hybrid automata with average dwell time: an invariant approach. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004.
12. S. Mitra, N. Lynch, and D. Liberzon. Verifying average dwell time by solving optimization problems, September 2005. Available from: http://theory.lcs.mit.edu/~mitras/research/hscc06-full.pdf.
13. S. Mitra, Y. Wang, N. Lynch, and E. Feron. Safety verification of model helicopter controller using hybrid Input/Output automata. In *HSCC'03, Hybrid System: Computation and Control*, 2003.
14. A. S. Morse. Supervisory control of families of linear set-point controllers, part 1: exact matching. *IEEE Transactions on Automatic Control*, 41:1413–1431, 1996.
15. A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, London, 2000.
16. H.P. Williams. *Model building in mathematical programming*. J. Wiley, New York, 1990. third edition.