Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 1369–1376

Synthesis of Implementable Control Strategies for Lazy Linear Hybrid Automata

Luigi Di Guglielmo
Department of Computer Science
University of Verona, Italy

Sanjit A. Seshia
Department of EECS
University of California, Berkeley, USA

Tiziano Villa
Department of Computer Science
University of Verona, Italy

***Abstract*—In the last few years hybrid automata have been widely applied in the modeling and verification of hybrid systems, but their related formal verification techniques usually rely on un-implementable assumptions to which a concrete control strategy cannot adhere. For this reason, once a hybrid model of the system has been proved to be correct with respect to the desired properties, it would be valuable to derive a correct-by-construction implementable control strategy for such a model. This work discusses a new methodology and a corresponding tool-chain that allows to synthesize an implementable control strategy for the class of hybrid automata named Lazy Linear Hybrid Automata (LLHA). LLHA model the discrete time behavior of control systems containing finite-precision sensors and actuators interacting with their environment under bounded delays.**

I. Introduction

HYBRID systems are dynamical systems whose behaviors cannot be characterized faithfully using either discrete or continuous models. They consist of a discrete part that operates in a continuous environment, and for this reason, they are sensitive not only to time-driven phenomena but also to event-driven ones. The presence of mixed dynamics makes the formal treatment of this kind of systems considerably hard.

Hybrid automata (HA) [1] are a powerful formalism for modeling hybrid systems. It extends the usual definition of finite state automata with continuous variables that evolve according to dynamics characterizing each discrete state. In the last few years, a wide spectrum of algorithmic techniques has been studied to solve the problems of simulation and verification for hybrid automata. Current state-of-art tools can verify hybrid systems with complex nonlinear dynamics [2], [3], or linear systems with a large number of continuous variables [4], thus becoming of interest also for real test-cases and application domains.

Another phase of the design flow where the interplay of continuous and discrete behaviors makes things complicated is the refinement and implementation phase. Indeed, formal verification techniques for hybrid automata usually rely on un-implementable assumptions, such as the *synchrony hypothesis*, i.e., the capability of performing any computation in zero time units and forcing a change in the dynamics of the hybrid system with no delays. As a consequence, the verification results on the correctness of the ideal model of the system cannot be directly applied to a real implementation [5].

For this reason, new semantics for hybrid automata, which do no rely on synchrony or other unrealistic assumptions, have been proposed in the literature [6], [7]. By formally verifying the correctness of the system using such semantics, it is possible to synthesize an implementable control strategy for the analyzed hybrid system, i.e., determine the *performance and latency bounds* to be satisfied by any conservative concrete hardware/software device that implements the system.

In [6], [8], the authors propose the Almost-ASAP semantics, a formal semantics that imposes a controller to react within a bounded delay, i.e., Δ, when a synchronization or a control action has to take place. The authors use reachability analysis [1] to look for the largest value Δ for which the controller is still correct w.r.t. the properties that the original instantaneous model has to enforce. Such a Δ-relaxed controller represents an implementable control strategy if $\Delta > 0$. The main limitation of this approach is that the problem of synthesizing such a value Δ may not be decidable.

The work in [7] proposes a similar approach for synthesizing implementable control strategies. The authors try to derive an implementable control strategy from the original instantaneous model by shrinking the guards of the latter so that, by assuming bounded reaction delays for synchronization and control actions, all behaviors of the former are non-blocking (i.e., shrinkability problem). This means that all timing requirements satisfied by the instantaneous control strategy, such as critical deadlines, are strictly respected by the derived one. The authors have shown that deciding the shrinkability problem can be checked in EXPTIME.

This work focuses on the problem of automating the synthesis of implementable control strategies for a relevant class of hybrid automata, named *Lazy Linear Hybrid Automata* (LLHA). Such a kind of automata takes into account all the typical implementation aspects (e.g., discrete time behaviors, finite precision of sensors and clock, sensing and actuation delays), thus, once they have been proved correct, they provide an implementable control strategy for the hybrid system.

The paper is organized as follows. Section II introduces the fundamental definitions and semantics of LLHA which motivate the assumptions at the base of the proposed methodology for synthesis of implementable control strategies described in Section III. Section IV describes some case studies to which the methodology has been applied. Finally, Section V is devoted to concluding remarks.

978-1-4673-4471-5/$25.00

II. BACKGROUND

In the following the class of LLHA is described. A LLHA is meant to be a model of a closed-loop system consisting of a digital controller interacting with a continuous environment [9]. The controller samples the state of the continuous environment at periodic discrete-time instants. The state of the environment consists of the values of the continuous variables as observed by the sensors. These values are digitized with finite precision and reported to the controller that may decide to switch the state of the environment. In such a case, the controller generates suitable output signals that, once transmitted to the actuators, will effect the desired change. Sensors will report the values of the current variables and actuators will change the evolution of the continuous variables with bounded delays.

A. The LLHA formal definition

Definition 1 (Lazy Linear Hybrid Automaton)**.** A finite precision Lazy Linear Hybrid Automaton (LLHA) is a tuple $\langle X, Q, init, inv, flow, E, jump, Act, P, D, \epsilon, B\rangle$. The components of a LLHA are as follows:

- *Variables*. A finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables. $\dot{X}$ stands for the set $\{\dot{x_1}, \ldots, \dot{x_n}\}$ of dotted variables and X' stands for the set $\{x'_1, \ldots, x'_n\}$ of primed variables.
- *Control modes*. A finite set Q of control modes. $Q_0 \subseteq Q$ denotes the set of initial modes.
- *Initial condition*. A labeling function $init$ that assigns to each control mode $q \in Q_0$ an initial predicate. The initial predicate $init(q)$ is a convex (non-)linear formula over the variables in X.
- *Invariant condition*. A labeling function inv that assigns to each control mode $q \in Q$ an invariant predicate. The invariant predicate $inv(q)$ is a convex (non-)linear formula over the variables in X.
- *Flow condition*. A labeling function $flow$ that assigns to each control mode $q \in Q$ a flow predicate. For each $i \in \{1, \ldots, n\}$, let $\dot{X}^i_q \subset \mathbb{Q}$ be the set of legal flow rates for the variable x_i in the control mode q. The flow predicate $flow(q)$ is of the form $(\dot{x}_1 \in \dot{X}^1_q) \wedge \ldots \wedge (\dot{x}_n \in \dot{X}^n_q)$.
- *Control switches*. A set E of edges (q, q') from a source mode $q \in Q$ to a target mode $q' \in Q$.
- *Jump condition*. A labeling function $jump$ that assigns to each control switch $e \in E$ a predicate. Each jump predicate $jump(e)$ from the control mode q to q', is given by the conjunction of a guard and a reset condition. The *guard* is given by a convex (non-)linear formula over the variables in X. The *reset* condition is given by the identity predicate over the variables in $X \cup X'$ (e.g., $x'_i = x_i$).
- *Actions*. A finite set Act of actions that the automaton uses either for internal synchronization or for synchronizing with other communicating automata. An edge labeling function $action : E \to Act$ assigns an action to each control switch.
- *Period*. P represents the sampling interval of the controller, i.e., control mode switches take place at times T_0, T_1, T_2, ... where $T_{k+1} = T_k + P$.
- *Delay parameters*. $D = \{g, \delta_g, h, \delta_h\} \subset \mathbb{Q}$ is the set of delay parameters such that $0 \leq g \leq g + \delta_g < h \leq h + \delta_h \leq P$, where g denotes the actuation delay, h denotes the sensing delay and δ_g, δ_h represent the uncertainty in actuation and sensing delay, respectively.
- *Precision*. ϵ_i is the precision of measurement of variable x_i.
- *Range*. $B_i = [B_{i_{min}}, B_{i_{max}}] \subset \mathbb{R}$ is the allowed range of the variable x_i such that $B_{i_{min}}$, $B_{i_{max}} \in \mathbb{Q}$ and $B_{i_{min}} < B_{i_{max}}$.

To keep the notation compact, in what follows, $q \xrightarrow{a,\varphi} q'$ is used to denote that there exists a control switch $e = (q, q')$ with $q \neq q'$, $a = action(e)$ and $\varphi = jump(e)$ in A.

Unlike the conventional definition of linear hybrid automata [10], invariants and guards in LLHA can be non-linear (i.e., polynomial). The flows in linear hybrid automata are represented using rectangular formulas which denote closed intervals of the form $[l, r] \subset \mathbb{R}$ with $l, r \in \mathbb{Q}$ and $l < r$. Under the assumption of finite precision, in a LLHA such rectangular formulas denote finite sets of rational values modeling the rate of change of the different continuous variables.

B. The LLHA formal semantics

Let A be a lazy linear hybrid automaton as defined above. The following definitions are required to specify the behavior of A in terms of a transition relation.

Definition 2 (Valuation for continuous variables)**.** Let $X = \{x_1, \ldots, x_n\}$ be a set of continuous variables. A valuation V for the variables in X is a member of $\mathbb{R}^n$ such that V assigns a real value $V(i)$ to each variable x_i.

Definition 3 (State of a LLHA)**.** A *state* of a lazy linear hybrid automaton A is a triple $(q, V, \hat{q})$ where q, $\hat{q}$ are control modes and V is a valuation. q is the control mode holding at the current time instant and $\hat{q}$ is the control mode that held at the previous time instant. V captures the actual values of the variables at the current instant. The state $(q, V, \hat{q})$ is feasible if and only if $V(i) \in [B_{min_i}, B_{max_i}]$ for every i.

Intuitively, the state of a LLHA stores information about current and previous control modes (i.e., q and $\hat{q}$, respectively) due to the fact that, as a result of a mode change, the change of rates of continuous variables will occur with bounded delays. As a consequence, the evolution of continuous variables in a mode q will depend not only on the flow predicates of q, but also on the flow predicate of the previous control mode $\hat{q}$.

The initial state is, by convention, the triple $(q_{init}, V_{init}, q_{init})$. It is assumed without loss of generality that the initial state is feasible. Let S_A denote the set of states of A.

For convenience, in what follows, it is assumed that the rate of change of continuous variables is constant in each control mode. Thus, each flow predicate $flow(q)$ can be described

by vector $\rho_q \in \mathbb{Q}^n$ that specifies the rate $\rho_q(i)$ at which each variable x_i evolves when the automaton is in the control mode q.

Definition 4 (Transition relation of a LLHA)**.** $\Longrightarrow \subseteq S_A \times (Act \cup \{\tau\}) \times S_A$ is such that:

- Let $(q, V, \hat{q})$, $(q', V', \hat{q}') \in S_A$ be states and $a \in Act$. Then $(q, V, \hat{q}) \overset{a}{\Longrightarrow} (q', V', \hat{q}')$ if and only if $\hat{q}' = q$ and there exist a control switch of the form $q \overset{a,\varphi}{\longrightarrow} q'$ in A and $t_1 \in \mathbb{Q}^n$, $t_2 \in \mathbb{Q}^n$ such that $\forall i \in [1, n]$, $t_1(i) \in [g, g+\delta_g]$, $t_2(i) \in [h, h+\delta_h]$ and the following conditions are satisfied:
  1) Let $v_i = V(i) + \rho_{\hat{q}}(i) \cdot t_1(i) + \rho_q(i) \cdot (t_2(i) - t_1(i))$ for each i. Then $(\langle v_1 \rangle, \ldots \langle v_n \rangle)$ satisfies φ and each $\langle v_i \rangle$ represents the digitized value of the variable x_i that has been rounded using the value of ϵ_i.
  2) $V'(i) = V(i) + \rho_{\hat{q}}(i) \cdot t_1(i) + \rho_q(i) \cdot (P - t_1(i))$ for each i.
- Let $(q, V, \hat{q})$, $(q', V', \hat{q}') \in S_A$ be states. Then $(q, V, \hat{q}) \overset{\tau}{\Longrightarrow} (q', V', \hat{q}')$ if and only if $q' = \hat{q}' = q$ and there exists $t_1 \in \mathbb{Q}^n$ and $\forall i \in [1, n]$, $t_1(i) \in [g, g + \delta_g]$ such that:
  1) $V'(i) = V(i) + \rho_{\hat{q}}(i) \cdot t_1(i) + \rho_q(i) \cdot (P - t_1(i))$ for each i.

The lazy semantics of linear hybrid automata means that if a control mode switch took place at time T_k, then the delay in actuating a change in flow rates lies between $[T_k + g, T_k + g + \delta_g]$. Similarly, a control decision made at time T_k is based on the variables values read by the controller at some time in the interval $[T_{k-1} + h, T_{k-1} + h + \delta_h]$. The parameters δ_g and δ_h represent the bounded uncertainty in actuation and sensing delay, respectively. The precision ϵ_i depends on the accuracy of the sensors measuring x_i from the continuous dynamical system. Guards and state invariants are evaluated on the digitized values $\langle x_i \rangle$ of the variables x_i that have been rounded using the value of ϵ_i. The parameter B, instead, reflects the range of values which can be taken by a state variable associated with a fixed width register.

From the semantics defined above, it is possible to derive the notions of trajectory of a LLHA and the reachability relation between states.

Definition 5 (Trajectory of a LLHA)**.** Let A be a lazy linear hybrid automaton and let $(q, V, \hat{q})$ be a state of A. A trajectory of A from $(q, V, \hat{q})$ is a sequence of states $(q_i, V_i, \hat{q}_i)$ with $i > 0$, such that $(q_0, V_0, \hat{q}_0) = (q, V, \hat{q})$ and $(q_{i-1}, V_{i-1}, \hat{q}_{i-1}) \overset{\alpha}{\Longrightarrow} (q_i, V_i, \hat{q}_i)$ for some $\alpha \in Act \cup \{\tau\}$.

Example 1. Figure 1(i) sketches a lazy linear hybrid automaton A with two control modes q_1 and q_2. Let i and j be such that $i, j \in \{1, 2\}$ and $i \neq j$. Each invariant condition $inv(q_i)$ is defined as a subset I_i of $\mathbb{R}$ and the LLHA can stay in the control mode q_i if the valuation of the variable x satisfies the invariant condition. The jump condition of a control switch $e_{ij} = (q_i, q_j)$ is specified by a guard set G_{ij} and a reset function that, by definition of LLHA, is always

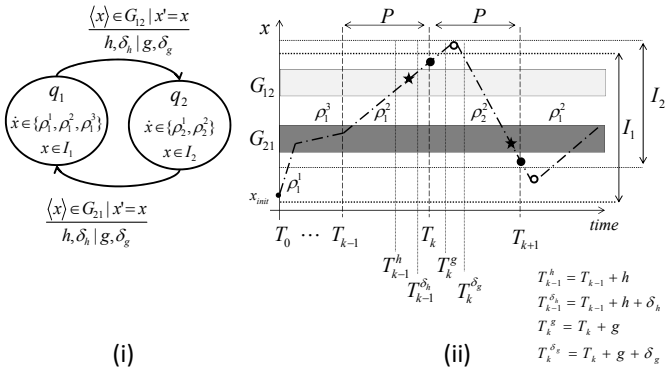

Fig. 1. Lazy linear hybrid automaton example.

the identity function. The control switch e_{ij} is enabled only if the digitized value $\langle x \rangle$ detected by the sensor belongs to G_{ij}. Moreover, sensing and actuation delays (i.e., h, δ_h and g, δ_g, respectively) are associated to the control switch. Finally, each flow condition $flow(q_i)$ constrains the evolution of the continuous variable x to one of the possible rates ρ_i^n allowed in the mode (e.g., $\{\rho_1^1, \rho_1^2, \rho_1^3\}$ in q_1).

Figure 1(ii) sketches part of a trajectory of such a LLHA starting from the initial state (q_1, x_{init}). In the example, the trajectory keeps following the dynamics $flow(q_1)$ until the time instant T_k. In fact, the control switch e_{12} is not enabled as soon as the trajectory reaches the guard set G_{12} because of the semantics of LLHA: a jump condition can be evaluated only at periodic time points and by considering the digitized values detected by the sensor at some instant (marked with $\star$) in the interval $[T_{k-1}^h, T_{k-1}^{\delta_h}]$. As shown in the figure, at T_k, the invariant condition of the mode q_1 is still satisfied, thus, the LLHA can either switch to q_2 or continue with the dynamics of q_1. Let assume that the automaton performs a control switch (marked with $\bullet$) and moves to q_2. When the LLHA switches from q_1 to q_2, it resets the continuous variable x according to the predicate specified by the jump condition, i.e., the identity function. Thus, in this case, the trajectory starts from the same state reached at T_k. Notice that the trajectory keeps following the dynamics of q_1 due to the presence of an actuation delay (i.e., g, δ_g) on the control switch. In fact, only at some time (marked with $\circ$) in the interval $[T_k^g, T_k^{\delta_g}]$ the trajectory changes according to the rates specified by the flow condition of q_2 (i.e., $\dot{x} \in \{\rho_2^1, \rho_2^2\}$). Then the trajectory follows that flow rate until the invariant I_2 is violated or the jump condition G_{21} is satisfied allowing the automaton to jump back in the mode q_1.

Definition 6 (Reachability relation between states of a LLHA)**.** Let A be a lazy linear hybrid automaton. A state $(q, V, \hat{q})$ reaches a state $(q', V', \hat{q}')$ if there exists a finite trajectory of states $(q_i, V_i, \hat{q}_i)$, with $0 \leq i \leq n$, such that $(q_0, V_0, \hat{q}_0) = (q, V, \hat{q})$ and $(q_n, V_n, \hat{q}_n) = (q', V', \hat{q}')$. $\mathcal{RC}(q, V, \hat{q})$ is used to denote the set of states reachable from $(q, V, \hat{q})$. $\mathcal{RC}$ is used to denote the set of all the possible states reachable from the initial ones.

III. Synthesis of Implementable Control Strategies for LLHA

The main contributions of this work can be summarized as follows:

- it proposes a Bounded Model Checking (BMC) [11] formulation for the problem of synthesizing implementable control strategies for LLHA that reduces such a problem to the state reachability problem on LLHA. A previous BMC formulation has been given in [12]. While that work assumes that precision and delay parameters are given, the present paper models them as parameters that must be synthesized. Then, by verifying the safety properties as reachability queries, it is possible to identify values for such parameters which make the control strategy implementable, i.e., the control strategy is able to handle the continuous plant by following discrete-time and finite-precision behaviors.
- it proposes a synthesis procedure that, starting from a set of feasible values for the different parameters, identifies for each of them the maximum values which enable a LLHA to satisfy its required safety properties.

The following sections describe all the details of the proposed approach.

A. Problem definition

The synthesis of an implementable control strategy for a LLHA consists of determining if there exist legal values for the sampling period (i.e., P), and upper bounds for sensing and actuation delays (i.e., $T_{SD} = h + \delta_h$ and $T_{AD} = g + \delta_g$, respectively) for which the control strategy modeled in the LLHA is able to satisfy the safety properties that the hybrid system has to ensure.

Let A be a LLHA such that $\mathcal{S}_A$ is the set of the possible states, $\mathcal{INIT}$ be a predicate that constrains the initial state, $\mathcal{TR}$ be the transition relation that models the lazy behavior of A and φ_{safe} be a function that tests whether the safety properties for the hybrid system hold in a given state. The synthesis problem summarized above can be formalized by a Quantified Boolean Formula (QBF), i.e., a formula in which propositional variables can be either quantified existentially or universally, as follows:

$$\begin{aligned}&\exists P, T_{SD}, T_{AD}, \forall\, n \in \mathbb{N}, \forall S_i \in \mathcal{S}_A : \mathcal{INIT}(S_0) \wedge \\ &\bigwedge_{i=0}^{n} \mathcal{TR}(S_i, S_{i+1}, P, T_{SD}, T_{AD}) \rightarrow \bigwedge_{i=0}^{n} \varphi_{safe}(S_i)\end{aligned} \tag{1}$$

Intuitively, the formula states that there exist suitable values for P, T_{SD} and T_{AD} for which at any step i, the state S_{i+1}, reachable from a previous state S_i, satisfies the safety property φ_{safe}.

An efficient way to solve this problem consists of deriving from Formula (1) a BMC problem on A. Such a BMC problem focuses on identifying the existence of *bad states*, i.e., states S_i violating the safety properties and reachable from the initial state of A:

$$\begin{aligned}&\mathcal{BMC}(A, \varphi_{safe}, n, P, T_{SD}, T_{AD}) \equiv \mathcal{INIT}(S_0) \wedge \\ &\bigwedge_{i=0}^{n} \mathcal{TR}(S_i, S_{i+1}, P, T_{SD}, T_{AD}) \wedge \bigvee_{i=0}^{n} \neg\varphi_{safe}(S_i)\end{aligned} \tag{2}$$

The identification of suitable values for the parameters n, P, T_{SD} and T_{AD} which cause the unsatisfiability of Formula (2), will prove the validity of Formula (1) w.r.t. the chosen P, T_{SD} and T_{AD}. Notice that, given the values for n, P, T_{SD} and T_{AD}, the satisfiability of Formula (2) may be proved or disproved by applying a Satisfiability Modulo Theory (SMT) [13] decision procedure on its propositional part.

Notice that the transition relation $\mathcal{TR}$ in the $\mathcal{BMC}$ formula may be unrolled a finite number n of times, where n is the *reachability diameter* [14] of the LLHA, i.e., the minimal number of steps for reaching all its reachable states. Thus, the formula checks if a bad state $S_{i \leq n}$ is reachable from the initial state S_0. Unfortunately, such a number n may require a very high number of copies of the transition relation in the $\mathcal{BMC}$ formula making the verification unfeasible due to memory problems.

Due to lack of space, the symbolic BMC encoding will be described in an extended version of the paper. In what follows a synthesis procedure is proposed for identifying the maximum suitable values of sampling period (P), sensing and actuation delays (T_{SD},T_{AD} respectively) to let the LLHA A satisfy the safety specification φ_{safe}.

B. Synthesis procedure

The definition of the BMC formula described in the previous section is based on a set of parameters whose values affect the correctness of the LLHA model. The synthesis engine aims at identifying the maximum values of such parameters for which the discrete-time and finite-precision behaviors specified by the LLHA are able to satisfy φ_{safe}.

In particular, the parameters reported into the formula $\mathcal{BMC}$ are the following:

- *sampling period* P. It specifies the periodicity at which it is possible to evaluate the guards for performing a mode switch;
- *sensing delay upper-bound* T_{SD}. It specifies the maximum delay admitted for notifying the controller that a mode switch can be performed (i.e., sensor latency);
- *actuation delay upper-bound* T_{AD}. It specifies the maximum delay admitted for changing the rate due to a mode switch (i.e., actuator latency).

At the moment, the precision ϵ of the observed values is not explicitly modeled as a parameter. Instead, it is assumed that it is fixed at some suitable level of granularity and that the constant values reported in the predicates that model guard and invariant conditions have been scaled accordingly to be represented as integers[1].

[1]Remember that the underlying structure used for the symbolic representation of variables is the bit-vector.

For reducing the time required in identifying the suitable values for the parameters summarized above, the user is asked to specify a desired sampling period P that the control strategy has to adopt. Then a synthesis procedure will automatically retrieve the maximum values for T_{SD} and T_{AD} that preserve the safety of the model according to the specified sampling period.

The procedure identifies the intended values by using a bisection method on a finite interval of feasible values for the parameters. According to the LLHA semantics, the actuation delay has to be smaller than the sensing delay and the sensing delay has to be smaller than the sampling period. Thus, it is necessary to search the suitable values of T_{AD} and T_{SD} in some intervals $\mathcal{I}_1 = [a, b]$ and $\mathcal{I}_2 = [c, d]$ such that $b < d$ and $b + d < P$. This is due to the fact that, in the BMC encoding, the upper bound T_{AD} for the actuation delay is considered as the most distant time instant from a sampling period T_i at which a control switch has occurred and, as a consequence, its starting search space should be given by the interval $[0, b]$. Similarly, in the BMC encoding, the upper-bound T_{SD} for the sensing delay is considered as the most distant time instant from a sampling period T_j, subsequent to T_i, at which a control decision may be taken. Thus, the starting search space for T_{SD} should be given by the interval $\mathcal{I}_2 = [0, d]$.

Algorithm 1 reports the pseudo-code implementing the synthesis procedure for parameters, and $BMC^n(P, T_{AD}, T_{SD})$ denotes the BMC encoding of the transition relation of the LLHA A unrolled n times (n is the reachability diameter).

At each step, the procedure divides the current subintervals of $[a, b]$ and $[c, d]$ in two by computing their midpoints mid_1 and mid_2. Then, using a SMT solver (i.e., $\mathcal{SMT}$), it verifies whether the formula $BMC^n(P, mid_1, mid_2)$ is valid by fixing $T_{AD} = mid_1$ and $T_{SD} = mid_2$. Now, according to the verification results, the method registers the current midpoints as candidate solutions and selects the subintervals to be used in the next step. In particular, if mid_1 and mid_2 make the formula valid, they become *candidate* maximal values for T_{AD} and T_{SD}, resp., and the new intervals of search will be $[mid_1, b]$ and $[mid_2, d]$, i.e., the procedure will check the validity of the formula on new values greater than the current midpoints. Otherwise, the procedure has to look for smaller ones. At first, it checks the formula validity by reducing only the current value for actuation delays. It computes the new candidate for T_{AD} (i.e., mid_{new}) and, by preserving the previous candidate mid_2 for T_{SD}, verifies the validity of the formula $BMC^n(P, mid_{new}, mid_2)$. If the verification returns a positive answer, then mid_{new} and mid_2 are recorded as new candidate maximal solutions for T_{AD} and T_{SD}, resp., and the new intervals of search will be $T_{AD} \in [mid_{new}, mid_1]$ and $T_{SD} \in [mid_2, d]$. On the contrary, the non-validity of the formula underlines that also a smaller sensing delay is required. For this reason the search continues on the intervals $[a, mid_1]$ and $[c, mid_2]$. In this way the intervals that contain the satisfying values of the parameters are reduced in width at least by 50% at each step. The process is continued until the maximum number N of iterations is reached.

Algorithm 1: The synthesis procedure of parameters for LLHA-based control strategies.

procedure find_values(BMC^n, P, a, b, c, d, N)
input: the BMC^n formula, the sampling period P, initial intervals [a,b] and [c,d] of feasible values for T_{AD} and T_{SD}, resp., and the maximum number N of iterations
output: maximal values of T_{AD} and T_{SD} for which the control strategy satisfies φ_{safe}, otherwise $T_{SD} = 0$ and $T_{AD} = 0$

```
1  it = 0;
2  T_SD = 0;
3  T_AD = 0;
4  mid_1 = ⌊(a + b)/2⌋;
5  mid_2 = ⌊(c + d)/2⌋;
6  while (it < N) do
7      if SMT(BMC^n(P, mid_1, mid_2)) --> valid then
8          a = mid_1;
9          T_AD = mid_1;
10         c = mid_2;
11         T_SD = mid_2;
12     else
13         b = mid_1;
14         mid_new = ⌊(a + b)/2⌋;
15         if SMT(BMC^n(P, mid_new, mid_2)) --> valid then
16             a = mid_new;
17             T_AD = mid_new;
18             c = mid_2;
19             T_SD = mid_2;
20         else
21             d = mid_2;
22     mid_1 = ⌊(a + b)/2⌋;
23     mid_2 = ⌊(c + d)/2⌋;
24     it = it + 1;
25 return (T_SD, T_AD);
```

IV. Experimental Results

This section reports the results obtained by applying the proposed LLHA parameter synthesis approach on four case studies. All experiments have been performed on a workstation with Intel Xeon 2.53 GHz processors and 16GB RAM. The hybrid models of the case studies have been described by means of the CIF [15] language. The *cif2uclid* tool has been implemented to automatically derive, from such models, the LLHA descriptions and the corresponding BMC encodings which have been automatically synthesized into equivalent SMT formulas by using the UCLID [16] modeling environment. Several SMT solvers have been used to verify the models and identify the maximum values for the sensing and actuation delay parameters appearing in the LLHA models. In particular, for each case-study the performances of the following SMT solvers have been compared: Beaver [17], Boolector[2] [18], and Yices [19]. Notice that any available SMT solver could be used as verification and parameter synthesis engine.

[2]MiniSat and PicoSat have been used as the underlying SAT engines.

A. Train-Gate Controller

The train gate controller ensures that the gate is closed when the train is approaching it. The train is assumed to move at a constant speed v on a circular track of length $d_{far-away}$ and the gate begins to close at a constant angular speed u when the train is at d_{max} distance from the gate. Once the train has moved d_{max} distance away from the gate, the gate begins to open again. The system is shown in Figure 2. The distance d of the train is measured in meters, the angle a of the gate in degrees and the time in seconds. The set of parameter values used in the running example is as follows: $v = 20m/s$, $u = 10°/s$, $d_{far-away} = 20000m$, $d_{max} = 400m$ and $d_{safe} = 160m$.

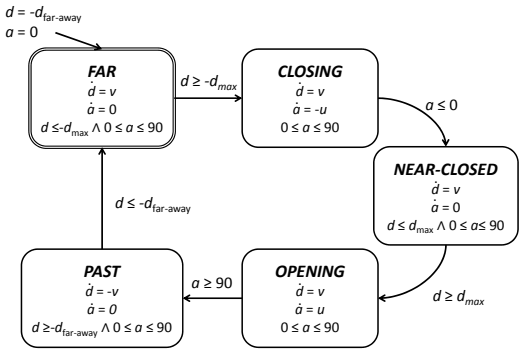

Fig. 2. LLHA model of the Train-Gate controller.

The system is considered safe, i.e., the train is never closer to the gate than d_{safe} unless the gate is completely closed, only if the following safety property is satisfied: $-d_{safe} \leq d \leq d_{safe} \rightarrow a <= 0$.

Such a property is used during the synthesis phase for identifying the maximum values for the sensing and actuation delay parameters (T_{SD} and T_{AD}, respectively) that are reported into the LLHA modeling the system. In particular, the parametric LLHA has been automatically generated by using a digitizing precision $\epsilon = 10^{-3}$ and a control switch period $P = 10^{-2}s$. Then, the parameter synthesis approach has determined that the coarse values for the sensing and actuation delays which ensure the correctness of such a LLHA are $T_{SD} = 4 \cdot 10^{-3}s$ and $T_{AD} = 2 \cdot 10^{-3}s$. The time required for synthesizing such values is reported in Table I.

TABLE I
SYNTHESIS TIMES USING DIFFERENT SMT SOLVERS.

$\mathcal{SMT}$	T_{SD} **s-Space**	T_{AD} **s-Space**	**# Bisect.**	**Time (s)**
Beaver	$[0; 5 \cdot 10^{-3}]$	$[0; 4 \cdot 10^{-3}]$	15	123.336
Boolector	$[0; 5 \cdot 10^{-3}]$	$[0; 4 \cdot 10^{-3}]$	15	101.544
Yices	$[0; 5 \cdot 10^{-3}]$	$[0; 4 \cdot 10^{-3}]$	15	49121.94

In particular, column $\mathcal{SMT}$ reports the name of the compared SMT solvers; columns T_{SD} *s-Space* and T_{AD} *s-Space* report the initial search spaces used for identifying suitable values for the sensing and actuation delays, respectively. Column *# Bisect.* shows the maximum number of bisection iterations allowed for synthesizing the parameter values and, finally, column *Time* reports the total time (in seconds) spent for the synthesis process.

B. Room Heating Controller

The room heating controller ensures that the temperature of a room is kept into a comfort interval by turning *on* and *off* the heater installed into the room. Figure 3 depicts the model of the system. Intuitively, the automaton is composed by two control modes *on* and *off*, representing the status of the heater. The variable x denotes the room temperature (measured in $°C$). When the heater is *off*, the temperature of the room falls according to any rate specified by the rectangular constraint $\dot{x} \in [-b, -a]$. Instead, when the heater is *on* the temperature of the room rises following any rate specified by the constraint $\dot{x} \in [b, c]$. The heater is turned on as soon as the falling temperature reaches x_{low}: the automaton moves to the control mode *on* and the temperature rises starting at a value $x \leq x_{low}$. The heater is turned off as soon as the temperature reaches x_{high}: the automaton moves to the control mode *off* and the temperature starts falling again at a value $x \geq x_{high}$. This control strategy guarantees that the temperature of the room will remain between x_{min} and x_{max} starting at the initial temperature x_{init} such that $x_{low} < x_{init} < x_{high}$. The set of parameter values used in the running example is as follows: $a = 2 \cdot 10^{-1}\ °C/s$, $b = 3 \cdot 10^{-1}\ °C/s$, $c = 4 \cdot 10^{-1}\ °C/s$, $x_{min} = 17.8°C$, $x_{low} = 18°C$, $x_{high} = 22°C$, $x_{max} = 22.5°C$ and $x_{init} = 19°C$.

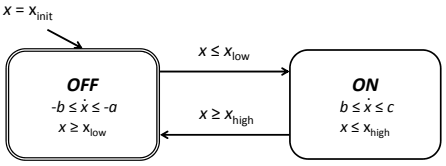

Fig. 3. The LLHA model of the room heating controller.

The property $x_{min} < x < x_{max}$ is used for synthesizing the maximum values of the sensing delay T_{SD} and the actuation delay T_{AD} in the parametric LLHA modeling the finite-precision lazy heating controller. Such a model has been generated from the one depicted in Figure 3 by choosing a digitizing precision $\epsilon = 10^{-4}$ and a control switch period $P = 10^{-2}s$. The synthesis procedure found that the values $T_{SD} = 11 \cdot 8^{-3}s$ and $T_{AD} = 10 \cdot 10^{-3}s$ guarantee that the lazy controller keeps the temperature into the comfort bounds (i.e., x_{min} and x_{max}). The time required for synthesizing such values is reported in Table II.

C. Watertank Controller

The watertank system is centered on a water tank, which is characterized by an uncontrolled outbound water flow, while the inbound water flow is controlled by the aperture of a valve. The controller acts on the aperture of the valve y in order to keep the water level x in a safe interval $x_{min} < x <$

TABLE II
SYNTHESIS TIMES USING DIFFERENT SMT SOLVERS.

$\mathcal{SMT}$	T_{SD} s-Space	T_{AD} s-Space	# Bisect.	Time (s)
Beaver	$[0; 4 \cdot 10^{-2}]$	$[0; 3 \cdot 10^{-2}]$	15	343.888
Boolector	$[0; 4 \cdot 10^{-2}]$	$[0; 3 \cdot 10^{-2}]$	15	1584.864
Yices	$[0; 4 \cdot 10^{-2}]$	$[0; 3 \cdot 10^{-2}]$	15	90903.836

TABLE III
SYNTHESIS TIMES USING DIFFERENT SMT SOLVERS.

$\mathcal{SMT}$	T_{SD} s-Space	T_{AD} s-Space	# Bisect.	Time (s)
Beaver	$[0; 5 \cdot 10^{-2}]$	$[0; 4 \cdot 10^{-2}]$	15	531.56
Boolector	$[0; 5 \cdot 10^{-2}]$	$[0; 4 \cdot 10^{-2}]$	15	2413.02
Yices	$[0; 5 \cdot 10^{-2}]$	$[0; 4 \cdot 10^{-2}]$	15	107236.98

x_{max}. The system model is shown in Figure 4. The water level is measured in deciliters, the valve aperture in degrees and the time in seconds. The set of parameter values used in the running example is as follows: $a = 1dl$, $b = 2dl$, $c = 4dl$, $d = 7dl$, $v = 20°/s$, $x_{high} = 850dl$, $x_{low} = 550dl$, $y_{min} = 0°$, $y_{max} = 360°$, $x_{max} = 870dl$ and $x_{min} = 540dl$.

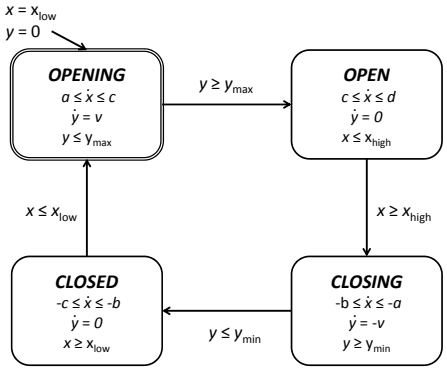

Fig. 4. LLHA model of the watertank controller.

In the model, the water level begins to increase according to a rectangular constraint $\dot{x} = [a, c]$ when the valve starts to open at constant angular speed v. As soon as the valve reaches its full aperture, the incoming flow reaches its maximum value, filling faster the water tank. Once the water level crosses an upper threshold x_{high}, the valve starts to close in order to avoid a water overflow. Once the valve is completely closed, no inbound water flow is present and the water level keeps decreasing. When the water level reaches its lower threshold x_{low}, the valve begins to open again.

The property $x_{min} < x < x_{max}$ is used for synthesizing the maximum values of the sensing delay T_{SD} and the actuation delay T_{AD} in the parametric LLHA modeling the finite-precision lazy watertank controller. Such a model has been generated from the one shown in Figure 4 by choosing a digitizing precision $\epsilon = 10^{-3}$ and a control switch period $P = 10^{-1}s$. At the end of the synthesis phase, the verification has determined that the values $T_{SD} = 23 \cdot 10^{-3}s$ and $T_{AD} = 11 \cdot 10^{-3}s$ guarantee that the lazy controller keeps safely the water level into the x_{min} and x_{max} bounds. The time required for synthesizing such values is reported in Table III.

D. Automated Highway Control System

Automated Highway Control System (AHS) is an arbiter which ensures that there is no collision between cars running on a highway by imposing legal speed ranges. The linear hybrid automaton representing the case of four cars is shown in Figure 5. This example is a small variant of the original model reported in [20]. The distance between cars is measured in km, time in hours and speeds in km/h. The set of parameter values used in the running example is as follows: $rl = 20km/h$, $b = 30km/h$, $c = 40km/h$, $d = 50km/h$, $e = 60km/h$, $ru = 70km/h$, $f = 100km/h$, $\alpha_{min} = 2 \cdot 10^{-3}km$, $\alpha_{max} = 1km$ and $\alpha'_{min} = 5 \cdot 10^{-4}km$.

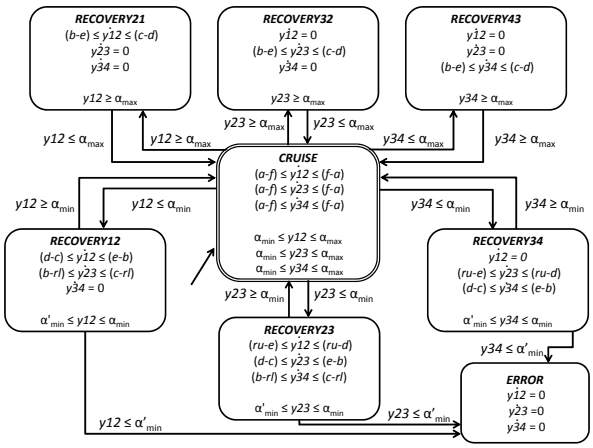

Fig. 5. LLHA model of the Automated Highway Control System.

To avoid collisions, the arbiter specifies speed limits (i.e., $[a, f]$) for each vehicle. When two vehicles i and j come within a distance $y_{ij} \leq \alpha_{min}$ of each other, there exists a possible collision event. The arbiter asks the approaching car to slow down by reducing the speed into the interval $[b, c]$, and asks the leading car to speed up by keeping a speed into the interval $[d, e]$; it also requires that all other cars not involved in the possible collision slow down to a constant recovery mode velocity rl for cars behind the critical region and ru for cars in front of the critical region. When the distance between the two vehicles involved in the possible collision exceeds α, the arbiter model goes back to the dynamics of the cruise mode. Moreover, the arbiter keeps all the vehicles below a maximal distance α_{max} of each other. When two vehicles i and j exceed such a distance (i.e., $y_{ij} \geq \alpha_{max}$), the arbiter asks the leading car to slow down by reducing the speed into the interval $[b, c]$ and asks the approaching car to speed up by keeping a speed into the interval $[d, e]$; it also requires that all other cars keep the current distance constant (i.e., speeding up for cars behind the critical region and slowing down for cars in front of the critical region). When the distance between the

two vehicles decreases below α_{max}, the arbiter model goes back to the dynamics of the cruise mode.

The only safety property to be satisfied by the model is that the control mode is never the *error* mode.

Again, once the parametric LLHA model has been extracted by choosing a digitizing precision $\epsilon = 10^{-5}$ and a clock period $P = 10^{-2}$, the safety property is used as a constraint for identifying the coarse values of the sensing and actuation delay parameters (T_{SD} and T_{AD}, respectively). In this case study, the synthesis phase determined that the values $T_{SD} = 218 \cdot 10^{-5}h$ and $T_{AD} = 112 \cdot 10^{-5}h$ guarantee the safety of the system, i.e., the lazy controller is able to avoid cars' collision. The time required for synthesizing such values is reported in Table IV.

TABLE IV
SYNTHESIS TIMES USING DIFFERENT SMT SOLVERS.

$\mathcal{SMT}$	T_{SD} **s-Space**	T_{AD} **s-Space**	**# Bisect.**	**Time (s)**
Beaver	$[0, 5 \cdot 10^{-3}]$	$[0, 4 \cdot 10^{-3}]$	15	1472.75
Boolector	$[0, 5 \cdot 10^{-3}]$	$[0, 4 \cdot 10^{-3}]$	15	1844.05
Yices	$[0, 5 \cdot 10^{-3}]$	$[0, 4 \cdot 10^{-3}]$	15	188523.11

V. CONCLUSION

The development of methodologies for the synthesis of implementable control strategies for models based on hybrid automata is a new and valuable research area. This work focused on defining a new methodology which enables the synthesis of implementable control strategies for the interesting subclass of lazy linear hybrid automata. To support the methodology, a tool, i.e., *cif2uclid*, and a synthesis procedure were implemented in order to provide a complete toolchain for synthesizing the implementable control strategy in a systematic way. *cif2uclid* is able to extract from a hybrid model a corresponding parametric-LLHA description, that is automatically synthesized into an equivalent SMT formula by using the UCLID modeling environment. The verification of such a formula retrieves constraints for the parameters which guarantee that the control strategy is implementable, i.e., the verification retrieves the performance and latency bounds which make the control strategy realizable by a concrete hardware/software device. The synthesis procedure may use any available SMT solver.

The proposed procedure for the automatic synthesis of parameters that guarantee implementability of control strategies is supported by a complete toolchain and improves the state-of-art with respect to previous proposals, however scalability of the overall approach is still a serious issue, when the objective is to study test cases of industrial strength. This may require from one side further restrictions to the class of allowed hybrid automata, still preserving enough expressivity for practical purposes, and from the other side advances in the computational engines and how they are used (for instance, gaining efficiency by using incrementally the SMT solvers).

REFERENCES

[1] T. Henzinger, "The Theory of Hybrid Automata," in *Proc. of IEEE Symposium on Logic in Computer Science (LICS)*, pp. 278 – 292, 1996.

[2] S. Ratschan and Z. She, "Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement," *ACM Transactions in Embedded Computing Systems*, vol. 6, no. 1, 2007.

[3] L. Benvenuti, D. Bresolin, P. Collins, A. Ferrari, L. Geretti, and T. Villa, "Assumeguarantee verification of nonlinear hybrid systems withAriadne," *International Journal of Robust and Nonlinear Control*, p. doi: 10.1002/rnc.2914, 2012.

[4] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable Verification of Hybrid Systems," in *Proc. of International Conference on Computer Aided Verification (CAV)*, pp. 379–395, 2011.

[5] D. Bresolin, L. D. Guglielmo, L. Geretti, R. Muradore, P. Fiorini, and T. Villa, "Open Problems in Verification and Refinement of Autonomous Robotic Systems," in *Proceedings of the 15th EUROMICRO Conference on Digital System Design (DSD 2012)*, pp. 469–476, 2012.

[6] M. Wulf, L. Doyen, and J. Raskin, "Almost ASAP Semantics: from Timed Models to Timed Implementations," *Formal Aspects of Computing*, vol. 17, no. 3, pp. 319 – 341, 2005.

[7] O. Sankur, P. Bouyer, and N. Markey, "Shrinking Timed Automata," in *Proc. of IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. 90–102, 2011.

[8] D. Bresolin, L. D. Guglielmo, L. Geretti, and T. Villa, "Correct-by-Construction Code Generation from Hybrid Automata Specification," in *Proceedings of First IEEE Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy'11)*, pp. 1660–1665, 2011.

[9] M. Agrawal and P. Thiagarajan, "The Discrete Time Behavior of Lazy Linear Hybrid Automata," in *Proc. of International Conference on Hybrid Systems: Computation and Control (HSCC)*, pp. 55–69, 2005.

[10] T. Henzinger and P. Kopke, "Discrete-Time Control For Rectangular Hybrid Automata," in *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 582–593, Springer, 1997.

[11] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, "Bounded Model Checking," *Advances in Computers*, vol. 58, pp. 117–148, 2003.

[12] S. Jha, B. A. Brady, and S. A. Seshia, "Symbolic Reachability Analysis of Lazy Linear Hybrid Automata," in *Proc. of International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pp. 241–256, 2007.

[13] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability Modulo Theories," in *Handbook of Satisfiability* (A. Biere, H. van Maaren, and T. Walsh, eds.), vol. 4, ch. 8, IOS Press, 2009.

[14] D. Kroening and O. Strichman, "Efficient Computation of Recurrence Diameters," in *Proc. of International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pp. 298–309, 2003.

[15] C. Sonntag, R. Schiffelers, D. van Beek, J. Rooda, and S. Engell, "Modeling and Simulation using the Compositional Interchange Format for Hybrid Systems," in *International Conference on Mathematical Modelling (MATHMOD)*, pp. 640–650, 2009.

[16] R. E. Bryant, S. K. Lahiri, and S. A. Seshia, "Modeling and Verifying Systems using a Logic of Counter Arithmetic with Lambda Expressions and Uninterpreted Functions," in *Proc. of International Conference on Computer Aided Verification (CAV)*, pp. 78–92, 2002.

[17] S. Jha, R. Limaye, and S. Seshia, "Beaver: Engineering An Efficient SMT Solver for Bit-Vector Arithmetic," in *Proc. of International Conference on Computer Aided Verification (CAV)*, pp. 668–674, 2009.

[18] R. Brummayer and A. Biere, "Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays," *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 174–177, 2009.

[19] B. Dutertre and L. De Moura, "The Yices SMT Solver." http://yices.csl.sri.com/tool-paper.pdf, 2006.

[20] S. K. Jha, B. H. Krogh, J. E. Weimer, and E. M. Clarke, "Reachability for Linear Hybrid Automata Using Iterative Relaxation Abstraction," in *Proc. of International Conference on Hybrid Systems: Computation and Control (HSCC)*, pp. 287–300, 2007.