

# Property Driven Three-Valued Model Checking on Hybrid Automata

Kerstin Bauer, Raffaella Gentilini, and Klaus Schneider

University of Kaiserslautern, Department of Computer Science, Germany  
{k\_bauer, gentilin, schneider}@cs.uni-kl.de

**Abstract.** In this paper, we present a three-valued *property driven* model checking algorithm for the logic CTL on hybrid automata. The technique of multi-valued model checking for hybrid automata aims at combining the advantages of classical methods based either on the preorder of simulation or on bounded reachability. However, as originally defined, it relies on the preliminary definition of special abstractions for combined over- and under-approximated reachability analysis, whose size is crucial and can be infinite. Our procedure avoids the above problem, since it is based on an incremental construction of the abstraction for the original hybrid automaton, that is suitably driven by the property under consideration.

## 1 Introduction

Hybrid automata [11,1] provide an appropriate modeling paradigm for systems where continuous variables interact with discrete modes. Such models are frequently used in complex engineering fields like embedded systems, robotics, automotive industries, avionics, and aeronautics [2,18,9]. In hybrid automata, the interaction between discrete and continuous dynamics is naturally expressed by associating a set of differential equations to every location of a finite automaton.

Finite automata and differential equations are well established formalisms in mathematics and computer science. Despite of their long-standing tradition, their combination in form of hybrid automata leads to surprisingly difficult problems that are often undecidable. In particular, the *reachability* problem is undecidable for most families of hybrid automata [14,15,16,10,1,5], and the few decidability results are built upon strong restrictions of the dynamics [3,12]. The reachability analysis of hybrid automata is a fundamental task, since checking *safety* properties of the underlying system can be reduced to a reachability problem for the set of bad configurations [11].

For this reason, a growing body of research is being developed on the issue of dealing with approximated reachability on undecidable – yet reasonably expressive – hybrid automata [6,19,8,17,18]. To this end, most of the techniques proposed so far either rely on bounded state-reachability or on the definition of finite abstractions. While the first approach suffers inherently of incompleteness, the quest for *soundness* is a key issue in the context of methods based on abstractions. In fact, abstractions can introduce unrealistic behaviors that may yield to spurious errors being reported in the safety analysis. Usually, a simulation preorder is required to relate the abstraction to the concrete

dynamics of the hybrid system under consideration, ensuring at least the correctness of each response of *non-reachability*. In general, the simulation preorder from the abstraction to the hybrid automaton allows for preservation of only *true* formulas in the *universal fragment* of a branching temporal logic. Recently, a novel framework has been proposed [7,4], featuring the capability of *both proving and disproving* properties expressed by means of the logic CTL or even the mu-calculus on (undecidable) hybrid automata. Such a method is based on the definition of a sound three-valued semantics [13] for the considered logics over so-called discrete bounded bisimulation (DBB) abstractions. Unfortunately, the size of the DBB quotients is not guaranteed to be finite for general hybrid systems. In fact, the method in [7,4] was applied to the only (undecidable) family of fully o-minimal hybrid automata.

In this paper, we sharpen the results concerning the size of the DBB abstractions on different classes of hybrid automata. Moreover, we design a new framework for three-valued model checking of the logic CTL on (undecidable) hybrid automata. Such a method has the key advantage of avoiding the a-priori construction of DBB abstractions that could result into an infinite partitioning. Rather, the abstraction is built on-the-fly and is driven by the ongoing three-valued model checking task.

## 2 Preliminaries

In this section, we introduce the basic definitions and the notations used in the paper.

**Definition 1 (Hybrid Automata [3]).** A hybrid automaton is a tuple  $H = (L, E, X, Init, Inv, F, G, R)$  with the following components:

- a finite set of locations  $L$
- a finite set of discrete transitions  $E \subseteq L \times L$
- a finite set of continuous variables  $X = \{x_1, \dots, x_n\}$  that take values in  $\mathbb{R}$
- an initial set of conditions:  $Init \subseteq L \times \mathbb{R}^n$
- an invariant location labeling  $Inv: L \mapsto 2^{\mathbb{R}^n}$
- a function  $F : L \times \mathbb{R}^n \mapsto 2^{\mathbb{R}^n}$  assigning to each location  $\ell \in L$  a differential rule defining the evolution of continuous variables within  $\ell$
- a guard edge labeling  $G : E \mapsto 2^{\mathbb{R}^n}$
- a reset edge labeling  $R : E \times \mathbb{R}^n \mapsto 2^{\mathbb{R}^n}$

We write  $\mathbf{v}$  to represent a valuation  $(v_1, \dots, v_n) \in \mathbb{R}^n$  of the variables' vector  $\mathbf{x} = (x_1, \dots, x_n)$ , whereas  $\dot{\mathbf{x}}$  denotes the first derivatives of the variables in  $\mathbf{x}$  (they all depend on the time, and are therefore rather functions than variables). A *state* in  $H$  is a pair  $s = (\ell, \mathbf{v})$ , where  $\ell \in L$  is called the *discrete component* of  $s$  and  $\mathbf{v}$  is called the *continuous component* of  $s$ . A *run* of  $H = (L, E, X, Init, Inv, F, G, R)$ , starts at any  $(\ell, \mathbf{v}) \in Init$  and consists of continuous evolutions (within a location) and discrete transitions (between two locations). Formally, a run of  $H$  is a path with alternating continuous and discrete steps in the *time abstract transition system* of  $H$ , defined below:

**Definition 2.** The time abstract transition system of the hybrid automaton  $H = (L, E, X, Init, Inv, F, G, R)$  is the transition system  $T_H = (Q, Q_0, \ell \rightarrow, \rightarrow)$ , where:

- $Q \subseteq L \times \mathbb{R}^n$  and  $(\ell, \mathbf{v}) \in Q$  if and only if  $\mathbf{v} \in \text{Inv}(\ell)$
- $Q_0 \subseteq Q$  and  $(\ell, \mathbf{v}) \in Q_0$  if and only if  $\mathbf{v} \in \text{Init}(\ell) \cap \text{Inv}(\ell)$
- $\ell_{\rightarrow} = E \cup \{\delta\}$  is the set of edge labels
- $\rightarrow \subseteq Q \times \ell_{\rightarrow} \times Q$  is the set of labeled transitions, that are determined as follows:
  - there is a continuous transition  $(\ell, \mathbf{v}) \xrightarrow{\delta} (\ell, \mathbf{v}')$ , if and only if there is a differentiable function  $f : [0, t] \rightarrow \mathbb{R}^n$ , with  $\dot{f} : [0, t] \rightarrow \mathbb{R}^n$  such that:
    1.  $f(0) = \mathbf{v}$  and  $f(t) = \mathbf{v}'$
    2. for all  $\varepsilon \in (0, t)$ ,  $f(\varepsilon) \in \text{Inv}(\ell)$ , and  $\dot{f}(\varepsilon) = F(\ell, f(\varepsilon))$ .
  - there is a discrete transition  $(\ell, \mathbf{v}) \xrightarrow{e} (\ell', \mathbf{v}')$  if and only if there exists an edge  $e = (\ell, \ell') \in E$ ,  $\mathbf{v} \in G(\ell)$  and  $\mathbf{v}' \in R((\ell, \ell'), \mathbf{v})$ .

A region is a subset of the states  $Q$  of  $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$ . Given a region  $B$  and a transition label  $a \in \ell_{\rightarrow}$ , the predecessor region  $\text{pre}_a(B)$  is defined as the region  $\{q \in Q \mid \exists q' \in B. q \xrightarrow{a} q'\}$ . The *bisimulation* and the *simulation* relations are two fundamental tools in the context of hybrid automata abstraction.

**Definition 3 ((Bi)simulation).** Let  $T_1 = \langle Q^1, Q_0^1, \ell_{\rightarrow}, \rightarrow^1 \rangle$ ,  $T_2 = \langle Q^2, Q_0^2, \ell_{\rightarrow}, \rightarrow^2 \rangle$ ,  $Q^1 \cap Q^2 = \emptyset$ , be two labeled transition systems<sup>1</sup> and let  $\mathcal{P}$  be a partition on  $Q_1 \cup Q_2$ . A simulation from  $T_1$  to  $T_2$  is a non-empty relation on  $\rho \subseteq Q^1 \times Q^2$  such that, for all  $(p, q) \in \rho$ :

- $p \in Q_0^1$  iff  $q \in Q_0^2$  and  $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$ , where  $[p]_{\mathcal{P}}$  ( $[q]_{\mathcal{P}}$ ) is the class of  $p$  ( $q$ ) w.r.t.  $\mathcal{P}$ .
- for each label  $a \in \ell_{\rightarrow}$ , if there exists  $p'$  such that  $p \xrightarrow{a} p'$ , then there exists  $q'$  such that  $(p', q') \in \rho$  and  $q \xrightarrow{a} q'$ .

If there exists a simulation from  $T_1$  to  $T_2$ , then we say that  $T_2$  simulates  $T_1$ , denoted  $T_1 \prec_S T_2$ . If  $T_1 \prec_S T_2$  and  $T_2 \prec_S T_1$ , then  $T_1$  and  $T_2$  are said similar, denoted  $T_1 \equiv_S T_2$ . If  $\rho$  is a simulation from  $T_1$  to  $T_2$ , and  $\rho^{-1}$  is a simulation from  $T_2$  to  $T_1$ , then  $\rho$  is said a bisimulation.

Definition 4 formally introduces the concept of abstraction for hybrid automata, on the ground of the notion of simulation.

**Definition 4.** Let  $H = (L, E, X, \text{Init}, \text{Inv}, F, G, R)$  be a hybrid automaton. An abstraction of  $H$  is a finite labeled transition system  $\mathcal{A} = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \rightarrow \rangle$  where:

- $Q_{\mathcal{A}}$  is a finite partition of  $Q$ ,  $Q_{\mathcal{A}}^0$  is a finite partition of  $Q_0$ ,  $\ell_{\rightarrow} = \{\delta\} \cup E$ , and  $\rightarrow \subseteq Q_{\mathcal{A}} \times Q_{\mathcal{A}}$ .
- $\mathcal{A}^* = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta}^* \rangle$  simulates<sup>2</sup>  $T_H$

Definition 6 recapitulates the semantics of the temporal logic CTL (where the  $\text{neXt}$  temporal operator is intensionally omitted because of the density of the underlying time framework) on hybrid automata [1,11].

**Definition 5 (CTL for Hybrid Automata).** Let  $\text{AP}$  be a finite set of propositional letters and  $\mathbf{p} \in \text{AP}$ . CTL is the set of formulas defined by the following grammar rules:

$$\phi ::= \mathbf{p} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid E(\phi_1 \cup \phi_2) \mid A(\phi_1 \cup \phi_2)$$

<sup>1</sup> A labeled transition system is a quadruple  $\langle Q, Q_0, \ell_{\rightarrow}, \rightarrow \rangle$ , where  $Q_0 \subseteq Q$ ,  $\rightarrow \subseteq Q \times \ell_{\rightarrow} \times Q$ .

<sup>2</sup> Given a relation  $\rightarrow$ , we denote by  $\rightarrow^*$  its transitive and reflexive closure.

**Definition 6 (CTL Semantics).** Let  $H = (L, E, X, \text{Init}, \text{Inv}, F, G, R)$  be a hybrid automaton, let  $Q$  ( $Q_0$ ) be the set of states (initial states) of  $H$ , and let  $\mathbf{AP}$  be a set of propositional letters. Consider  $\ell_{\mathbf{AP}} : L \times \mathbb{R}^n \mapsto 2^{\mathbf{AP}}$ . Given  $\phi \in \text{CTL}$  and  $s \in Q$ ,  $s \models \phi$  is inductively defined as follows:

- $s \models p$  if and only if  $p \in \ell_{\mathbf{AP}}(s)$
- $s \models \neg\phi$  if and only if not  $s \models \phi$
- $s \models \phi_1 \wedge \phi_2$  if and only if  $s \models \phi_1$  and  $s \models \phi_2$
- $s \models E(\phi_1 \cup \phi_2)$  if and only if there exists a run  $\rho$  and a time  $t$  such that:
  - $\rho(t) \models \phi_2$
  - $\forall t' \leq t. \rho(t') \models \phi_1$
- $s \models A(\phi_1 \cup \phi_2)$  if and only if for each run  $\rho$  there exists a time  $t$  such that:
  - $\rho(t) \models \phi_2$
  - $\forall t' \leq t. \rho(t') \models \phi_1$

Then,  $H \models \phi$  iff  $\forall s \in Q_0. s \models \phi$ .

### 3 Three-Valued Model Checking on Hybrid Automata and the Problem of Discrete Bounded Bisimulation Explosion

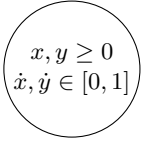
In this section we recall the notion of *discrete bounded bisimulation* abstraction on hybrid automata [7] and we sharpen the results relative to the size of its quotient on different families of hybrid systems. Discrete bounded bisimulation (cf. Definition 7) was introduced in [7] as a bisimulation approximation for hybrid automata, suitable to support sound *three-valued* model checking results on these systems.

**Definition 7 (Discrete Bounded Bisimulation (DBB)).** Consider the time abstract transition system  $T_H = (Q, Q_0, \ell_{\rightarrow}, \rightarrow)$  of a hybrid automaton  $H$ , and let  $\mathcal{P}$  be a partition on  $Q$ :

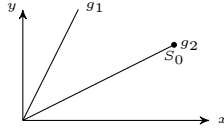
1.  $\equiv_0 \in Q \times Q$  is the maximum relation on  $Q$  such that for all  $p, q \in Q$ :  
if  $p \equiv_0 q$  then (a)  $[p]_{\mathcal{P}} = [q]_{\mathcal{P}}$  and  $p \in Q_0$  iff  $q \in Q_0$   
(b)  $\forall p'. p \xrightarrow{\delta} p' \Rightarrow \exists q'. p' \equiv_0 q' \wedge q \xrightarrow{\delta} q'$   
(c)  $\forall q'. q \xrightarrow{\delta} q' \Rightarrow \exists p'. p' \equiv_0 q' \wedge p \xrightarrow{\delta} p'$
2. Given  $n \in \mathbb{N}^+$ ,  $\equiv_n$  is the maximum relation on  $Q$  such that for all  $p, q \in Q$ :  
if  $p \equiv_n q$  then (a)  $p \equiv_{n-1} q$   
(b)  $\forall p'. p \xrightarrow{\delta} p' \Rightarrow \exists q'. p' \equiv_n q' \wedge q \xrightarrow{\delta} q'$   
(c)  $\forall q'. q \xrightarrow{\delta} q' \Rightarrow \exists p'. p' \equiv_n q' \wedge p \xrightarrow{\delta} p'$   
(d)  $\forall p'. p \xrightarrow{e} p' \Rightarrow \exists q'. p' \equiv_{n-1} q' \wedge q \xrightarrow{e} q'$   
(e)  $\forall q'. q \xrightarrow{e} q' \Rightarrow \exists p'. p' \equiv_{n-1} q' \wedge p \xrightarrow{e} p'$

Given  $n \in \mathbb{N}$ , the relation  $\equiv_n$  will be referred to as *n-DBB equivalence*.

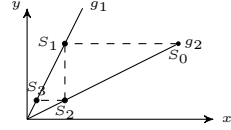
In [7] the notion of DBB was shown to provide finite abstractions on the family of fully o-minimal hybrid automata, for which the reachability problem is not decidable [14]. On this ground, it was possible to design a three-valued model checking algorithm for



**Fig. 1.** Single location rectangular automaton



**Fig. 2.** Initial partition of the state space



**Fig. 3.** Infinite bisimulation computation

the logic CTL and the  $\mu$ -calculus on fully o-minimal hybrid systems. Unfortunately, there are other important families of undecidable hybrid automata for which such a finiteness result does not hold. In particular, the following example (cf. Example 1) considers the class of (non-initialized) rectangular automata (RHA) for which the differential inclusions describing the continuous dynamics are *rectangular*, i.e. given by

$$\bigwedge_{i=1 \dots n} \dot{x}_i \in [a_i, b_i], a_i, b_i \in \mathbb{R}^+$$

RHA are well known to be undecidable w.r.t. the reachability problem [12], unless strong constraints such as the so-called *initialization*<sup>3</sup> are considered. Example 1 proves that the size of DBB abstractions cannot be guaranteed to be finite on RHA. Thus, the (DBB based) framework for the three-valued model checking of hybrid automata developed in [7] can not be applied to RHA.

*Example 1 (DBB explosion for RHA).* Consider the simple rectangular automaton given in Figure 1. Figure 2 provides an initial partition over the states-space of  $H$ , given by the line  $g_1 : y = 2x$ , the point  $S_0 = (8, 4)$ , and the segment  $g_2 : y = \frac{1}{2}x \wedge 0 \leq x < 8$ . Figure 3 depicts the construction procedure of the 0-DBB abstraction over  $H$ , w.r.t. the initial partition given in Figure 2. The procedure does not terminate since the regions  $g_1, g_2$  need to be split into the infinite set of regions  $\{(S_{2i+3}, S_{2i+1})\}_{i \geq 0}, \{(S_{2i+2}, S_{2i})\}_{i \geq 0}$ , with  $S_{2i+1} = (x_{2i}, \frac{1}{4}y_{2i})$  and  $S_{2i} = (\frac{1}{4}x_{2i-1}, y_{2i-1})$  due to the point  $S_0 = (8, 4)$ .

## 4 A General Algorithmic Framework for Property Driven Three-Valued CTL Model Checking on Hybrid Automata

In this Section, we develop a new algorithmic approach to the three-valued model checking of hybrid automata. Rather than relying on a preliminary static abstraction of the considered dynamical system, our method features a dynamic partitioning process, which is suitably driven by the three-valued model checking of the formula given as input. In other words, the three-valued verification task is accomplished *on the fly*, thus avoiding the problems related to the (possibly infinite) size of the DBB-abstraction.

As shown in Figure 4, our algorithm PROPERTYDRIVEN3MC accomplishes its task upon the subsequent usage of (1) the discrete switches (within the procedure D3MC, called at line 7) and (2) the continuous dynamics of the hybrid automaton given as input

<sup>3</sup> RHA are initialized, iff for each continuous variable during any discrete transition either the continuous flow does not change or the value of the variable is reinitialized.

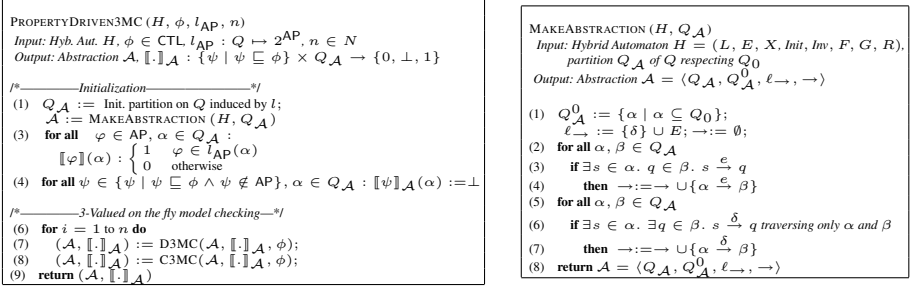


Fig. 4. The Property Driven Three-Valued Model Checking Procedure

(within the procedure C3MC, called at line 8). Both discrete and continuous dynamics are employed to grow dynamically an abstraction  $\mathcal{A}$  having the following key property. On the one hand,  $\mathcal{A}$  can be *globally* viewed as an *over*-approximation of the dynamics of the original hybrid automaton  $H$ . On the other hand, one can enucleate *local* underapproximations of the evolutions in  $H$ . These features permit to perform our on the fly three-valued model checking task. In particular, underapproximations (overapproximations) can be suitably used to deal with existential (universal) subformulas.

#### 4.1 The Procedure D3MC

Within each iteration of the main algorithm, the procedure D3MC in Figure 5 exploits the discrete dynamics of the underlying hybrid automaton to (1) refine a growing abstraction  $\mathcal{A}$ , and (2) proceed with its on the fly three-valued model checking. First, each discrete edge is used to split those classes in the current partition, for which some (sub)-formula of interest evaluates to the third value  $\perp$  (lines (2)–(4)). Such a split allows to separate set of states that *must* evolve to different regions via a discrete switch, and therefore should be assigned distinguished model checking values. Then, the paths through  $\mathcal{A}$  are used to suitably *propagate* the current information, in order to infer new sound true or false evaluations of the (sub)-formulas of interest. This is done within the subprocedure D3MCFORM, called at line 7. More in detail, D3MCFORM

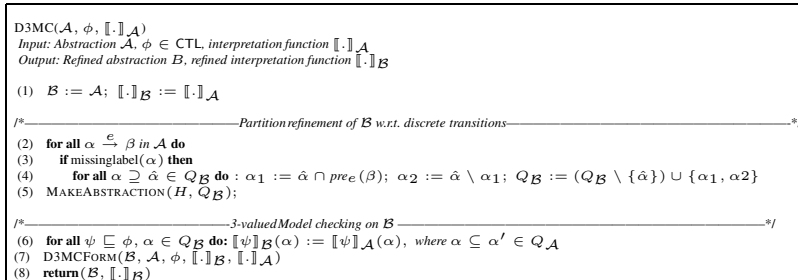


Fig. 5. The procedure D3MC within the algorithm PROPERTYDRIVEN3MC

```

D3MCFORM( $\mathcal{B}, \mathcal{A}, \phi, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ )
Input: Abstractions  $\mathcal{B}, \mathcal{A}$ , CTL formula  $\phi$ , interpretation functions  $\llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ 
Output: refined interpretation function  $\llbracket \cdot \rrbracket_{\mathcal{B}}$ 

(1) case  $\phi = \neg \varphi$  do
(2)   D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ ):  $\llbracket \phi \rrbracket_{\mathcal{B}} = \neg \llbracket \varphi \rrbracket_{\mathcal{B}}$ 

(3) case  $\phi = \varphi_1 \wedge \varphi_2$  do
(4)   D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_1, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ ); D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_2, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ ):  $\llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\mathcal{B}} = \llbracket \varphi_1 \rrbracket_{\mathcal{B}} \wedge \llbracket \varphi_2 \rrbracket_{\mathcal{B}}$ 

(5) case  $\phi = E(\varphi_1 U \varphi_2)$  do
(6)   D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_1, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ ); D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_2, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(7)   for all  $\alpha \in Q_{\mathcal{B}} : E(\varphi_1 U \varphi_2)(\alpha) = \perp$  do
(8)      $\llbracket E(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{B}}(\alpha) = 1$  iff
            $\llbracket \varphi_2 \rrbracket_{\mathcal{B}}(\alpha) = 1 \vee \llbracket \varphi_1 \rrbracket_{\mathcal{B}}(\alpha) = 1 \wedge \exists \alpha \xrightarrow{e} \beta. \llbracket E(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{A}}(\beta) = 1$ 
(9)      $\llbracket E(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{B}}(\alpha) = 0$  iff
            $\mathcal{B}$  does not contain any path starting in  $\alpha$  and modelling  $\varphi_1 U \varphi_2$  (by using a standard 2-valued model checking procedure on  $\mathcal{B}$ )

(10) case  $\phi = A(\varphi_1 U \varphi_2)$  do
(11)   D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_1, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ ); D3MCFORM( $\mathcal{B}, \mathcal{A}, \varphi_2, \llbracket \cdot \rrbracket_{\mathcal{B}}, \llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(12)   for all  $\alpha \in P : A(\varphi_1 U \varphi_2)(\alpha) = \perp$  do
(13)      $\llbracket A(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{B}}(\alpha) = 0$  iff
            $\llbracket \varphi_2 \rrbracket_{\mathcal{B}}(\alpha) = 0 \vee \llbracket \varphi_1 \rrbracket_{\mathcal{B}}(\alpha) = 1 \wedge \exists \alpha \xrightarrow{e} \beta. \llbracket A(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{A}}(\beta) = 0$ 
(14)      $\llbracket A(\varphi_1 U \varphi_2) \rrbracket_{\mathcal{B}}(\alpha) = 1$  iff
           All paths in  $\mathcal{B}$  starting in  $\alpha$  model  $\varphi_1 U \varphi_2$  (by using a standard 2-valued model checking procedure on  $\mathcal{B}$ )

```

**Fig. 6.** The subprocedure D3MCFORM, used within the algorithm PROPERTYDRIVEN3MC

(cf. Figure 6) refines the interpretation function following different schemes, depending on the existential (resp. universal) character of the considered operator. For existential formulas, only paths consisting of a single discrete switch are followed, since the latter provide (local) under-approximations of the concrete evolutions (lines 5–9). For universal formulas instead, global paths are followed (lines 10–14).

On the ground of Lemma 1, Lemma 2 below states formally the correctness of each on the fly model checking macro-step based on the usage of discrete switches, and performed within the procedure D3MC. Consider a call to  $D3MC(\mathcal{A}, \phi, \llbracket \cdot \rrbracket_{\mathcal{A}})$ , where  $\phi$  is a CTL formula,  $\mathcal{A}$  is an abstraction of the hybrid automaton  $H$ , and  $\llbracket \cdot \rrbracket_{\mathcal{A}} : \Gamma = \{\varphi \mid \varphi \sqsubseteq \phi\} \times Q_{\mathcal{A}} \mapsto \{0, 1, \perp\}^4$ . Moreover, let from now on  $\preceq$  be the *information ordering* on the set of truth values  $\{\perp, 0, 1\}$ , in which  $\perp \preceq 1$ ,  $\perp \preceq 0$ , and  $x \preceq x$  (for all  $x \in \{\perp, 0, 1\}$ ).

**Lemma 1.** *The abstraction refinement  $\mathcal{B}$  of  $\mathcal{A}$  produced upon the execution of the procedure  $D3MC(\mathcal{A}, \phi, \llbracket \cdot \rrbracket)$  is such that<sup>5</sup>:*

1.  $T_H \prec_S \mathcal{B}^* \prec_S \mathcal{A}^*$ .
2. *If  $\alpha \xrightarrow{e} \beta$  in  $\mathcal{B}$ , then each state in  $\alpha$  has a discrete successor in the unique class of  $\mathcal{A}$  containing  $\beta$ .*

**Lemma 2.** *Consider  $D3MC(\mathcal{A}, \phi, \llbracket \cdot \rrbracket_{\mathcal{A}})$  and assume that  $\llbracket \cdot \rrbracket_{\mathcal{A}}$  is sound w.r.t. the model checking of  $\phi$  on  $H$ , i.e.:  $\forall \alpha \in Q_{\mathcal{A}}. \forall \varphi \sqsubseteq \phi. \llbracket \varphi \rrbracket_{\mathcal{A}}(\alpha = [s]) \preceq \llbracket \varphi \rrbracket_H(s)$   
Then, the refined interpretation function  $\llbracket \cdot \rrbracket_{\mathcal{B}}$  computed by  $D3MC(\mathcal{A}, \phi, \llbracket \cdot \rrbracket_{\mathcal{A}})$  satisfies:*

$$\forall \alpha \in Q_{\mathcal{B}}. \forall \varphi \sqsubseteq \phi. \llbracket \phi \rrbracket_{\mathcal{A}}(\hat{\alpha}) \preceq (\llbracket \varphi \rrbracket_{\mathcal{B}}(\alpha = [s]) \preceq \llbracket \varphi \rrbracket_H(s), \text{ where } \alpha \subseteq \hat{\alpha} \in \mathcal{A}$$

<sup>4</sup> We use Kleene's definition of three-valued logic [13].

<sup>5</sup> Recall that given an abstraction  $\mathcal{A} = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta} \rangle$  for the hybrid automaton  $H$ , we denote by  $\mathcal{A}^*$  the structure  $\mathcal{A}^* = \langle Q_{\mathcal{A}}, Q_{\mathcal{A}}^0, \ell_{\rightarrow}, \xrightarrow{e} \cup \xrightarrow{\delta^*} \rangle$ , according to Definition 4.

```

C3MC( $\mathcal{A}$ ,  $\phi$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )
Input: Abstraction  $\mathcal{A}$ , CTL formula  $\phi$ , interpretation functions  $\llbracket \cdot \rrbracket_{\mathcal{B}}$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ 
Output: refined Abstraction  $\mathcal{B}$ , interpretation function  $\llbracket \cdot \rrbracket_{\mathcal{B}}$ 
Requires:  $T_H \prec_S \mathcal{B}^* \prec_S \mathcal{A}^*$ 
 $\forall \alpha \in Q_{\mathcal{B}}. \forall \varphi \sqsubseteq \phi. \llbracket \phi \rrbracket_{\mathcal{A}}(\hat{\alpha}) \preceq (\llbracket \varphi \rrbracket_{\mathcal{B}}(\alpha = [s]) \preceq \llbracket \varphi \rrbracket_H(s))$ , where  $\alpha \subseteq \hat{\alpha} \in \mathcal{A}$ 
(1) case  $\phi = \neg \varphi$  do
(2)   C3MC( $\mathcal{A}$ ,  $\varphi$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ );  $\llbracket \phi \rrbracket_{\mathcal{A}} = \neg \llbracket \varphi \rrbracket_{\mathcal{A}}$ 
(3) case  $\phi = \varphi_1 \wedge \varphi_2$  do
(4)   C3MC( $\mathcal{A}$ ,  $\varphi_1$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ ); C3MC( $\mathcal{A}$ ,  $\varphi_2$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ );  $\llbracket \phi \rrbracket_{\mathcal{A}} = \llbracket \varphi_1 \rrbracket_{\mathcal{A}} \wedge \llbracket \varphi_2 \rrbracket_{\mathcal{A}}$ 
(5) case  $\phi = E(\varphi_1 U \varphi_2)$  do
(6)   C3MC( $\mathcal{A}$ ,  $\varphi_1$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ ); C3MC( $\mathcal{A}$ ,  $\varphi_2$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(7)   SPLIT&CHECKEU( $\mathcal{A}$ ,  $\varphi_1 \varphi_2$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(8) case  $\phi = A(\varphi_1 U \varphi_2)$  do
(9)   C3MC( $\mathcal{A}$ ,  $\varphi_1$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ ); C3MC( $\mathcal{A}$ ,  $\varphi_2$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(10)  SPLIT&CHECKAU( $\mathcal{A}$ ,  $\varphi_1$ ,  $\varphi_2$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )
(11) return ( $\mathcal{A}$ ,  $\llbracket \cdot \rrbracket_{\mathcal{A}}$ )

```

Fig. 7. The subprocedure C3MC within the algorithm PROPERTYDRIVEN3MC

## 4.2 The Subprocedure C3MC

The purpose of the procedure C3MC in our main algorithm is that of gaining information from the continuous dynamics of the given hybrid automaton to proceed – on the fly – in the three-valued model checking of the considered property. The continuous dynamics associated to hybrid automata can be governed by differential inclusions (like in RHA) or by differential equations, the latter defined in a variety of theories over the reals (linear [11], semi-algebraic [6,17], o-minimal [14]). To cope with such a variety of patterns, we proceed bottom-up by defining first a simple interface for our procedure C3MC (cf. Figure 7). Within such an interface boolean formulae are dealt with using standard techniques. Instead, the resolution of temporal operators is only required to be implemented by adhering to the following constraints: If  $\langle \mathcal{B}, \llbracket \cdot \rrbracket_{\mathcal{B}} \rangle$  are the refined abstraction and the interpretation function computed by the function SPLIT&CHECKEU (resp. SPLIT&CHECKAU) then

- $T_H \prec_S \mathcal{B}^* \prec_S \mathcal{A}^*$
- $\forall \alpha \in Q_{\mathcal{B}}. \forall \varphi \sqsubseteq \phi. \llbracket \phi \rrbracket_{\mathcal{A}}(\hat{\alpha}) \preceq (\llbracket \varphi \rrbracket_{\mathcal{B}}(\alpha = [s]) \preceq \llbracket \varphi \rrbracket_H(s))$ , where  $\alpha \subseteq \hat{\alpha} \in \mathcal{A}$

In the next section, we will propose various implementations of the interface for the procedure C3MC, specialized for different classes of hybrid automata. Theorem 1, states the correctness of the main algorithm PROPERTYDRIVEN3MC, assuming a sound implementation of the interface for C3MC.

**Theorem 1.** *The algorithm PROPERTYDRIVEN3MC( $H, \phi, l_{AP}, n$ ), where the subprocedure C3MC implements correctly the corresponding interface, is sound w.r.t. the three-valued model checking of  $\phi$  on  $H$ .*

## 5 Specializing the General Algorithmic Framework for Different Classes of Hybrid Automata

In this section we consider various classes of hybrid automata that do not admit a finite discrete bounded bisimulation. Thereof, we propose suitable specializations of our general on the fly three-valued model checking algorithm. In particular, we appropriately

instantiate the interface given in the previous section for the procedure C3MC, taking into account different possible concrete descriptions of the continuous dynamics.

### 5.1 The Case of Hybrid Automata Based on Autonomous ODEs and Decidable Theories over the Reals: A Symbolic Approach

The first family of hybrid automata considered is the class of autonomous hybrid automata, for which the continuous evolution is described by means of a system of autonomous ordinary differential equations (ODEs). In autonomous hybrid automata, the continuous evolution originated from a given state is the same regardless of when it starts, and thus is *uniquely* determined. An important subfamily of autonomous hybrid systems is the class of o-minimal hybrid automata [14], for which continuous evolutions are also defined within an o-minimal theory. Such an assumption guarantees the finiteness of the discrete bounded bisimulation abstraction, while its removal leads easily to the realm of infiniteness<sup>6</sup> [14].

To implement the interface for C3MC (cf. Figure 7), we need to consider the operators EU, AU. In virtue of the deterministic character of the continuous component in autonomous systems, we can provide suitable symbolic descriptions of the sets of states satisfying  $E(\varphi \cup \psi)$ ,  $A(\varphi \cup \psi)$ . In particular, let  $f = E(\varphi \cup \psi)$ ,  $g = E(\varphi \cup \psi)$ , and suppose that you want to refine two corresponding (previously computed) sound interpretation functions  $\llbracket f \rrbracket$  and  $\llbracket g \rrbracket$ , using the continuous evolution in a given location  $\ell$ . Let  $\Phi : \mathbb{R}_0^+ \times \mathbb{R}^n \mapsto \mathbb{R}^n$  be the flow of the system of ODE associated to  $\ell$ . Then, the symbolic formulas (2)–(5), below, determine the sets of states within the considered location for which  $f$  and  $g$  evaluate to  $v \in \{0, 1, \perp\}$ , due to the continuous evolution among the regions induced by  $\llbracket f \rrbracket$  and  $\llbracket g \rrbracket$ .

$$\iota(x, t) := \forall 0 \leq t' \leq t. \Phi(x, t') \in \text{Inv}(\ell) \quad (1)$$

$$f_{\langle 1 \rangle}(x) := \exists t. \iota(x, t) \wedge \llbracket f \rrbracket(\Phi(x, t)) = 1 \wedge \forall 0 \leq t' \leq t. \llbracket \varphi \rrbracket(\Phi(x, t')) = 1 \quad (2)$$

$$f_{\langle \perp, 1 \rangle}(x) := \exists t. \iota(x, t) \wedge (\llbracket \psi \rrbracket(\Phi(x, t)) \neq 0 \vee \quad (3)$$

$$\exists y. \Phi(x, t) \xrightarrow{e} y \wedge \llbracket f \rrbracket(y) \neq 0) \wedge \forall 0 \leq t' \leq t. \llbracket \varphi \rrbracket(\Phi(x, t')) \neq 0$$

$$g_{\langle 0 \rangle}(x) := \forall t. \llbracket \psi \rrbracket(\Phi(x, t)) = 0 \wedge \iota(x, t) \quad (4)$$

$$\vee \exists t. \llbracket g \rrbracket(\Phi(x, t)) = 0 \wedge \iota(x, t) \wedge \forall 0 \leq t' < t. \llbracket \psi \rrbracket(\Phi(x, t')) = 0$$

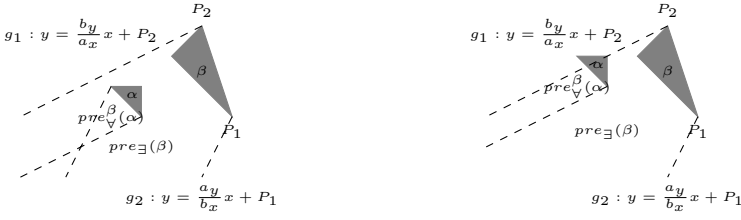
$$g_{\langle 0, \perp \rangle}(x) := \forall t. \iota(x, t) \wedge \llbracket \psi \rrbracket(\Phi(x, t)) \neq 1 \vee \forall t. \llbracket g \rrbracket(\Phi(x, t)) = 1 \wedge \iota(x, t) \quad (5)$$

$$\rightarrow \exists 0 \leq t' < t. \llbracket \varphi \rrbracket(\Phi(x, t')) \neq 1 \vee \exists y. \Phi(x, t') \xrightarrow{e} y \wedge \llbracket g \rrbracket(y) \neq 1$$

Let  $\mathcal{A}^{dec}$  be the subclass of autonomous hybrid automata for which all relevant sets (i.e. flow, invariants, initial states, . . .) are defined within a decidable theory of the reals. By Lemma 3 and 4, below, formulas (1)–(5) correctly implement SPLIT&CHECKEU and SPLIT&CHECKAU on  $\mathcal{A}^{dec}$ , w.r.t. the constraints imposed in the interface C3MC.

**Lemma 3.** *Consider the abstraction  $\mathcal{A}$  of the hybrid automaton  $H$  and assume that  $\llbracket \cdot \rrbracket_{\mathcal{A}}$  is sound w.r.t. the model checking of  $E(\phi \cup \psi)$ . Then, the following holds true:*

<sup>6</sup> [14] proves that autonomous (non o-minimal) *continuous* dynamical systems do not admit a finite bisimulation, implying infiniteness of 0-DBB for the corresponding hybrid systems.



**Fig. 8.** Geometric approach to obtain  $pre_{\exists}(\beta)$  and  $pre_{\forall}^{\beta}(\alpha)$

- $x \in f_{\langle 1 \rangle} \Rightarrow \llbracket E(\varphi U \psi) \rrbracket_H(x) = 1$
- $\llbracket E(\varphi U \psi) \rrbracket_H(x) = 1 \Rightarrow x \in f_{\langle \perp, 1 \rangle}$

**Lemma 4.** Consider the abstraction  $\mathcal{A}$  of the hybrid automaton  $H$  and assume that  $\llbracket \cdot \rrbracket_{\mathcal{A}}$  is sound w.r.t. the model checking of  $A(\phi U \psi)$ . Then, the following holds true:

- $x \in g_{\langle 0 \rangle} \Rightarrow \llbracket A(\varphi U \psi) \rrbracket_H(x) = 0$
- $\llbracket A(\varphi U \psi) \rrbracket_H(x) = 0 \Rightarrow x \in g_{\langle 0, \perp \rangle}$

## 5.2 The Case of (Uninitialized) Rectangular Automata: A Geometric Approach

In this subsection we consider the class of rectangular hybrid automata (RHA) that we proved in Section 3 not to admit a finite discrete bounded bisimulation.

Unlike for hybrid automata based on autonomous ODEs, in RHA the continuous evolution traversing a state  $x$  is not uniquely determined, since it is defined by differential inclusions. In order to cope with such a non deterministic behavior for the continuous component, we introduce two basic operators ( $pre_{\exists}(\alpha)$ ,  $pre_{\forall}^{\beta}(\alpha)$ ) that allow to quantify existentially (resp. universally) on the set of continuous paths evolving to the (convex) region  $\alpha \subseteq Inv(\ell)$ , within a given location  $\ell$ . The operators  $pre_{\exists}(\alpha)$ ,  $pre_{\forall}^{\beta}(\alpha)$  are formally defined by the Equations 6 and 7, below. Here, we use the symbol  $\rightsquigarrow_{\ell}$  to represent a (feasible) continuous path within location  $\ell$ .

$$pre_{\exists}(\alpha) := \{v \in Inv(\ell) \mid \exists v' \in \alpha. v \rightsquigarrow_{\ell} v'\} \quad (6)$$

$$pre_{\forall}^{\beta}(\alpha) := \{v \in pre_{\exists}(\beta) \mid \forall v' \in \beta. v \rightsquigarrow_{\ell} v' \Rightarrow \exists v'' \in \alpha. v \rightsquigarrow_{\ell} v'' \rightsquigarrow v'\} \quad (7)$$

Note that for the operator  $pre_{\forall}^{\beta}(\alpha)$ , the additional parameter  $\beta$  allows to restrict the universal quantification to the only paths that will eventually reach  $\beta$ . In particular, if  $\beta$  is the entire state space the operator boils down to collect the set of states for which all the paths evolve to  $\alpha$ . The implementation of the operators  $pre_{\exists}(\beta)$ ,  $pre_{\forall}^{\beta}(\alpha)$  on RHA can be obtained by simple geometrical reasoning. Consider e.g. the two regions  $\alpha, \beta$  depicted in Figure 8, and the rectangular differential inclusion  $a \leq x \leq b$ . The figure illustrates the geometrical approach allowing to obtain the regions  $pre_{\exists}(\beta)$ ,  $pre_{\forall}^{\beta}(\alpha)$ . Such approach can be easily generalized to consider arbitrary located regions  $\alpha, \beta$ .

Let  $f = E(\phi U \psi)$ ,  $g = A(\phi U \psi)$  for which the three-valued on the fly model checking is left open for C3MC in Figure 7. Let  $\ell$  be a location of the hybrid automaton, let  $\mathcal{P}_{\ell}$

be a partition of  $Inv(\ell)$  and let  $\llbracket f \rrbracket$  (resp.  $\llbracket g \rrbracket$ ) a sound three-valued interpretation of  $f$  (resp.  $g$ ) over  $\mathcal{P}_\ell$ . Formulas (8)–(11), determine the sets of states within the location  $\ell$  for which  $f$  and  $g$  evaluate to  $v \in \{0, 1, \perp\}$ , due to the continuous evolution over  $\mathcal{P}_\ell$ .

$$f_{\langle 1 \rangle}(x) = \bigcup_{\alpha \in Inv(\ell): \llbracket E(\phi \cup \psi) \rrbracket(\alpha) = 1} (pre_{\exists}(\alpha) \setminus pre_{\forall}^{\alpha}(\bigcup_{\beta: \llbracket \phi \rrbracket(\beta) \neq 1} \beta)) \quad (8)$$

$$f_{\langle \perp, 1 \rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell): \llbracket \psi \rrbracket(\alpha) \neq 0 \vee \\ \exists \alpha \xrightarrow{e} \gamma. \llbracket E(\phi \cup \psi) \rrbracket(\gamma) \neq 0}} (pre_{\exists}(\alpha) \setminus pre_{\forall}^{\alpha}(\bigcup_{\beta: \llbracket \phi \rrbracket(\beta) = 0} \beta)) \quad (9)$$

$$g_{\langle 0 \rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell): \llbracket A(\phi \cup \psi) \rrbracket(\alpha) = 0 \vee \\ \inf(\alpha) = 1 \wedge \llbracket \psi \rrbracket(\alpha) = 0}} (pre_{\exists}(\alpha) \setminus pre_{\forall}^{\alpha}(\bigcup_{\beta: \llbracket \psi \rrbracket(\beta) \neq 1} \beta)) \quad (10)$$

$$g_{\langle 0, \perp \rangle}(x) = \bigcup_{\substack{\alpha \in Inv(\ell): \llbracket \psi \rrbracket(\alpha) \neq 1 \vee \\ \exists \alpha \xrightarrow{e} \gamma. \llbracket A(\phi \cup \psi) \rrbracket(\gamma) \neq 1}} (pre_{\exists}(\alpha) \setminus pre_{\forall}^{\alpha}(\bigcup_{\beta: \llbracket \psi \rrbracket(\beta) = 1} \beta)) \quad (11)$$

The above sets can be used to appropriately split each class  $\alpha \in \mathcal{P}_\ell$  for which  $\llbracket f \rrbracket(\alpha) = \perp$  (resp.  $\llbracket g \rrbracket(\alpha) = \perp$ ), refining on the fly the current information for the three-valued model checking of  $f$  (resp.  $g$ ). Lemma 5 and Lemma 6, below, ensure that the sets  $f_{\langle 1 \rangle}(x)$ ,  $f_{\langle \perp, 1 \rangle}(x)$ ,  $g_{\langle 0 \rangle}(x)$ ,  $g_{\langle 0, \perp \rangle}(x)$  are sound, and can be used to implement the functions SPLIT&CHECKEU and SPLIT&CHECKAU in procedure C3MC according to the requirements.

**Lemma 5.** *Consider the abstraction  $\mathcal{A}$  of the hybrid automaton  $H$  and assume that  $\llbracket \cdot \rrbracket_{\mathcal{A}}$  is sound w.r.t. the model checking of  $E(\phi \cup \psi)$ . Then, the following holds true:*

- $x \in f_{\langle 1 \rangle} \Rightarrow \llbracket E(\varphi \cup \psi) \rrbracket_H(x) = 1$
- $\llbracket E(\varphi \cup \psi) \rrbracket_H(x) = 1 \Rightarrow x \in f_{\langle \perp, 1 \rangle}$

**Lemma 6.** *Consider the abstraction  $\mathcal{A}$  of the hybrid automaton  $H$  and assume that  $\llbracket \cdot \rrbracket_{\mathcal{A}}$  is sound w.r.t. the model checking of  $A(\phi \cup \psi)$ . Then, the following holds true:*

- $x \in g_{\langle 0 \rangle} \Rightarrow \llbracket A(\varphi \cup \psi) \rrbracket_H(x) = 0$
- $\llbracket A(\varphi \cup \psi) \rrbracket_H(x) = 0 \Rightarrow x \in g_{\langle 0, \perp \rangle}$

## 6 Conclusions

We sharpen the results relative to the size of the DBB abstraction for HA, showing that the family of RHA does not admit a finite DBB. On this ground, we extend the applicability of three-valued model checking on classes of HA having infinite DBBs. In particular, we provide a property driven three-valued model checking algorithm that does not require an a-priori discretization of the states-space by means of the DBB.

## References

1. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)

2. Alur, R., Henzinger, T., Ho, P.-H.: Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering* 22(3), 181–201 (1996)
3. Alur, R., Henzinger, T., Lafferriere, G., Pappas, G.: Discrete abstractions of hybrid systems. *Proceedings of the IEEE* 88(7), 971–984 (2000)
4. Bauer, K., Gentilini, R., Schneider, K.: A uniform approach to three-valued semantics for  $\mu$ -calculus on abstractions of hybrid automata. In: Hu, A., Chockler, H. (eds.) *Haifa Verification Conference (HVC)*, Haifa, Israel. LNCS. Springer, Heidelberg (2008)
5. Brihaye, T., Michaux, C., Rivière, C., Troestler, C.: On O-minimal hybrid systems. In: Alur, R., Pappas, G.J. (eds.) *HSCC 2004*. LNCS, vol. 2993, pp. 219–233. Springer, Heidelberg (2004)
6. Fränzle, M.: What will be eventually true of polynomial hybrid automata? In: Kobayashi, N., Pierce, B.C. (eds.) *TACS 2001*. LNCS, vol. 2215, pp. 340–359. Springer, Heidelberg (2001)
7. Gentilini, R., Schneider, K., Mishra, B.: Successive abstractions of hybrid automata for monotonic CTL model checking. In: Artemov, S.N., Nerode, A. (eds.) *LFCS 2007*. LNCS, vol. 4514, pp. 224–240. Springer, Heidelberg (2007)
8. Ghosh, R., Tiwari, A., Tomlin, C.: Automated symbolic reachability analysis with application to delta-notch signaling automata. In: Maler, O., Pnueli, A. (eds.) *HSCC 2003*. LNCS, vol. 2623, pp. 233–248. Springer, Heidelberg (2003)
9. Ghosh, R., Tomlin, C.: Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) *HSCC 2001*. LNCS, vol. 2034, pp. 232–245. Springer, Heidelberg (2001)
10. Henzinger, M., Henzinger, T., Kopke, P.: Computing simulations on finite and infinite graphs. In: Seberry, J., Pieprzyk, J. (eds.) *Annual Symposium on Foundations of Computer Science (FOCS)*, p. 453. IEEE Computer Society Press, Los Alamitos (1995)
11. Henzinger, T.: The theory of hybrid automata. In: *Verification of Digital and Hybrid Systems*. NATO Advanced Study Institute Series F: Computer and Systems Sciences, vol. 170, pp. 265–292. Springer, Heidelberg (2000)
12. Henzinger, T., Kopke, P., Puri, A., Varaiya, P.: What’s decidable about hybrid automata? *Journal of Computer and System Sciences* 57(1), 94–124 (1998)
13. Kleene, S.: *Introduction to Metamathematics*. North-Holland, Amsterdam (1952)
14. Lafferriere, G., Pappas, G., Sastry, S.: O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems* 13(1), 1–21 (2000)
15. Lafferriere, G., Pappas, J., Yovine, S.: A new class of decidable hybrid systems. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) *HSCC 1999*. LNCS, vol. 1569, pp. 137–151. Springer, Heidelberg (1999)
16. Miller, J.: Decidability and complexity results for timed automata and semi-linear hybrid automata. In: Lynch, N.A., Krogh, B.H. (eds.) *HSCC 2000*. LNCS, vol. 1790, pp. 296–309. Springer, Heidelberg (2000)
17. Piazza, C., Antoniotto, M., Mysore, V., Policriti, A., Winkler, F., Mishra, B.: Algorithmic algebraic model checking I: Challenges from systems biology. In: Etesami, K., Rajamani, S.K. (eds.) *CAV 2005*. LNCS, vol. 3576, pp. 5–19. Springer, Heidelberg (2005)
18. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation based abstraction refinement. In: Morari, M., Thiele, L. (eds.) *HSCC 2005*. LNCS, vol. 3414, pp. 573–589. Springer, Heidelberg (2005)
19. Tiwari, A., Khanna, G.: Series of abstractions for hybrid automata. In: Tomlin, C.J., Greenstreet, M.R. (eds.) *HSCC 2002*. LNCS, vol. 2289, pp. 465–478. Springer, Heidelberg (2002)