Conformance Testing of Cyber-Physical Systems: A Comparative Study

Morteza Mohaqeqi[1], Mohammad Reza Mousavi[2] and Walid Taha[3] *

[1] m.mohaqeqi@ut.ac.ir
School of Electrical and Computer Engineering, University of Tehran, Iran
Center for Research on Embedded Systems, Halmstad University, Sweden

[2] m.r.mousavi@hh.se
Center for Research on Embedded Systems, Halmstad University, Sweden

[3] walid.taha@hh.se
Center for Research on Embedded Systems, Halmstad University, Sweden
Rice University, USA

Abstract: For systematic and automatic testing of cyber-physical systems, in which a set of test cases is generated based on a formal specification, a number of notions of conformance testing have been proposed. In this paper, we review two existing theories of conformance testing for cyber-physical systems and compare them. We point out their fundamental differences, and prove under which assumptions they coincide.

Keywords: Hybrid systems, conformance testing, (hybrid) timed state sequences, and hybrid labeled transition systems.

1 Introduction

Cyber-physical systems are interdisciplinary by their nature and hence, experts from different disciplines have brought in different techniques for their specification, design, and analysis. This enables cross-fertilization of approaches among several disciplines but to achieve that, it requires reconciliation among the different techniques.

Model-based testing of cyber-physical systems is a relatively new concept that has been studied by few researchers from different backgrounds [BK06, vO06, vO09, ABW09, BWA10, Dan10, WLT13, AHF+14, AFM+14] and hence such a reconciliation has yet to take place.

Two notable examples of model-based testing theories for these systems in the literature are: the hioco relation by van Osch in [vO06, vO09] and the conformance relation by Abbas et al. in [AHF+14], which we refer to as hconf. The hioco theory is based on hybrid transition systems formalism, which allows to specify both discrete input and output actions, and continuous behavior. On the other hand, under hconf, the conformance relation is considered as a measure of distance between the observable discretized (sampled) behavior of systems. These two approaches are substantially different since they stem, respectively, from the computer science-

* This author was supported by the US NSF CPS award 1136099.

and control theory point of view on model-based testing. Our general goal of research is to come up with a notion that reconciles these two views, and we start with a formal and in-depth comparison. Concerning the specific research question addressed in this paper, we were mainly intrigued by the following statement from [AHF+14]:

> It can be verified that the hioco relation by van Osch is an exact version of $(T,J,(\tau,\varepsilon))$-closeness ($\tau=\varepsilon=0$) with the role of inputs and outputs made explicit.

We aimed at formalizing and proving this claim. (Note that "the exact version of $(T,J,(\tau,\varepsilon))$-closeness" is what we call "hconf" in the remainder of this paper.)[1] However, even if we set aside the issue of explicit input and output, it turns out there are other fundamental differences between the two notions that went unnoticed in this statement. Hence, we will first point out the main differences between the two notions (which will also guide our future research in reconciling them) and subsequently, seek a restricted setting in which the above-mentioned claim does hold. We then formalize and prove the above-mentioned statement in this restricted setting.

As a common ground for our comparison, we use a restricted subset of Hybrid Automata, of which the restrictions are given and motivated in the remainder of this paper. We then translate this subset into the semantic domains of the two notions of conformance and show that the notions of conformance for the translated hybrid automata coincide in both domains.

1.1 Running Example

We consider a bouncing ball as a simple example of a system exhibiting both continuous and discrete behavior. The ball is dropped with an initial height and an initial velocity. Naturally, it is under the force of gravity. However, a vertical force can also be applied to the ball, which is considered as the input to this system. Collision with the floor leads to a bounce, which causes the ball to move in the reverse direction with the speed reduced by a factor ρ. The movement of the ball between two consecutive collisions is accounted as a continuous behavior, while the collision with the floor is considered a discrete jump.

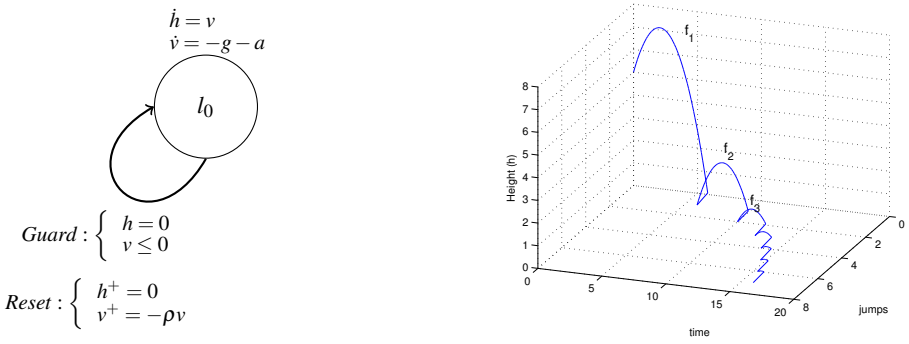
We can use a hybrid automaton to formally specify the behavior of the ball. We define three continuous variables h, v, and a for specifying the behavior of the ball, where h denotes the height, v denotes the vertical velocity, and a is the (vertical downward) acceleration caused by the force exerted on the ball. Consequently, the dynamics of this system can be captured by a hybrid automaton with one location, as shown in Fig. 1a. The dynamics associated to this location is specified as $\dot{h}=v$ and $\dot{v}=-g-a$, where g is the gravity constant.

Figure 1b shows the behavior of the system, starting with initial condition $h=5$ and $v=2$, while no vertical force is applied to the system.

1.2 Related Work

In the realm of model-based conformance testing, Tretmans proposed the theory of input/output (I/O) conformance (ioco) testing for discrete event systems [Tre96]. Afterwards, this theory was

[1] Informally, two observed behavior satisfy $(T,J,(\tau,\varepsilon))$-closeness relation if in each interval of length τ, there exists one point in each of them with difference less than ε.



(a) Hybrid automaton

(b) A sample behavior of the ball

Figure 1: Bouncing ball example

extended to real-time systems [BB05], where in addition to discrete states of the system, the evolution of a continuous variable, namely time, is modeled.

Conformance testing for cyber-physical systems (of which formal models are called hybrid systems in this paper), also, has been approached by a relatively small number of studies. In addition to the two aforementioned studies [vO06] and [AHF+14], which will be explained in detail in the next sections, conformance testing of hybrid systems using some notion of over-approximation has been proposed in [ABW09] and [BWA10]. These studies exploited a qualitative reasoning in order to obtain a discrete representation (abstraction) of a hybrid system. In a qualitative system, the continuous domains are abstracted into finite sets of quality levels and intervals. They proposed a method for obtaining a qualitative transition system from a hybrid system, which provides a discrete-event view to the system. Then, the ioco relation is applied to the obtained transition system. While this approach leads to finite representation of the hybrid system, it suffers from information loss occurred in the abstraction step.

Bringmann et al. [BK06] used stream-processing functions [MS97] for specifying test cases for hybrid systems. In this formalism, the continuous behavior of the variables is captured by the notion of streams. A stream-processing function or component relates a stream on input variables to a stream of output ones. Based on this notion, they propose a language for describing the test sets. They focus on how to specify test cases for hybrid systems. However, they provide no formal definition for conformance relation of hybrid systems.

In addition, Woehrle et al. [WLT13] investigated the problem of conformance testing of cyber-physical systems through measurement of a number of physical quantities like power consumption of the computing system. They employed timed automata to specify the system behavior. Then, the measured physical variable is compared against the automata to verify that if the system behavior conforms to its specification. Their method does not actually model the hybrid behavior of a system, but only considers time as a continuous variable. A conformance relation for hybrid systems has been also defined based on the notion of hybrid automata in [Dan10]. While the underlying formalism is different from the hybrid labeled transition system that is used by van Osch [vO06], the semantic model of the two conformance relations is similar [Dan10].

1.3 Paper Organization

The remainder of this paper is structured as follows. In Section 2, we review the hioco and hconf theories and point out the fundamental differences both between their semantic models and their conformance relations. In order to show the equivalence of the two conformance relations under specific assumptions, we first provide a method for translating a given hybrid automaton (in the restricted semantic domain) to each of the models in Section 3. Then, in Section 4, we show that, based on the proposed translation method, the two notions of conformance coincide. The paper concludes in Section 5, where we also describe our plan for future work.

2 Formal Definitions and Fundamental Differences

In this section, we first formally introduce system models that are used in the two aforementioned approaches, namely [vO06] and [AHF+14]. Afterwards, we point out the fundamental differences in their modeling approach and their assumptions. These will motivate the assumptions that we have made to pave the way for a meaningful comparison between the two conformance relations. Then, we review the conformance relations proposed by the two studies, and state the fundamental differences between them.

2.1 Formal Models

In the following, we first introduce some basic definitions. Then, we formally specify the system models employed in [vO06] and [AHF+14], namely, hybrid labeled transition system and hybrid-timed state sequence model, respectively.

2.1.1 Basic Definitions

A hybrid system is a system exhibiting both continuous and discrete behavior. The continuous behavior of the system is captured by the valuation of a set of continuous variables, denoted by V. We assume that V is partitioned into a set of input variables, denoted by V_I, and a set of output variables, denoted by V_O. A valuation on V is defined as a function which assigns a value to each variable $v \in V$; here, only variables of type real are considered. The set of all valuations of V is denoted by $Val(V)$ and is defined as the set of all functions $V \to \mathbb{R}$. To describe the (non-interrupted) continuous evolution of the system, we use the following notion of *trajectory* [LSV03].

Definition 1 (Trajectory [vO06]) Let $D \subset \mathbb{R}$ be an interval (here, singleton sets of numbers are also considered as intervals). A trajectory σ is a function $\sigma : D \to Val(V)$ that maps each element in interval D to a valuation. The set of all trajectories associated to V is denoted by $trajs(V)$.

Beside continuous evolution, a hybrid system features discrete changes, called jumps or switches. A jump happens instantly, leading to a possibly noncontinuous change.

Further, we consider a restriction operator on the functions as follows.

Definition 2 (Function Restriction) Consider an interval $D \subset \mathbb{R}$, a set V of variables, and a

function $f : D \to Val(V)$; we define the restricted function $f \downarrow V'$ for some $V' \subseteq V$ as the function of type $D \to Val(V')$, such that for each $d \in D$, $f(d) \downarrow V'$ is obtained from $f(d)$ after removing variables not in V'.

2.1.2 Hybrid Labeled Transition System

In the approach adopted by van Osch [vO06], a hybrid system is modeled as a hybrid labeled transition system (HLTS). An HLTS, formally defined below, consists of a set of states with discrete (action) and continuous (trajectory) transitions between them. Actions are partitioned into three classes, namely, input-, output-, and internal actions. The latter class is not observable from outside.

Definition 3 (Hybrid Labeled Transition System [vO06]) A hybrid labeled transition system (HLTS) $\mathcal{H}$, over a given set A of actions, is a 5-tuple $(S, s_0, V, L, \to)$, where

- S is a (possibly infinite) set of states;
- $s_0 \in S$ is the initial state;
- $V = V_I \uplus V_O$ is a set of (resp. input or output) continuous variables;
- $L = A \uplus trajs(V)$ is a set of (resp. action or trajectory) labels;
- $\to \subseteq S \times (L \cup \{\xi\}) \times S$ specifies the transition relation, where ξ denotes the internal action.

We may write $s \xrightarrow{l} s'$ instead of $(s, l, s') \in \to$. Moreover, we define a notion of generalized transition relation for an HLTS as follows. For the purpose of this paper, we assume the set of actions to be empty. We will also henceforth simplify the definitions for this restricted subset of HLTSs.

Definition 4 (Generalized Transition Relation) Consider an HLTS $\mathcal{H} = (S, s_0, V, L, \to)$. A generalized transition relation for $\mathcal{H}$ is defined as the smallest relation $\Rightarrow \subseteq S \times (L)^* \times S$ where

- $s \stackrel{\varepsilon}{\Rightarrow} s$;
- if $s \xrightarrow{\xi} s'$, then $s \stackrel{\varepsilon}{\Rightarrow} s'$,
- $\forall l \in L$, if $s \xrightarrow{l} s$, then $s \stackrel{l}{\Rightarrow} s$;
- $\forall \alpha, \beta \in L^*$, if $s \stackrel{\alpha}{\Rightarrow} s''$ and $s'' \stackrel{\beta}{\Rightarrow} s'$, then $s \stackrel{\alpha\beta}{\Rightarrow} s'$;

We write $s \stackrel{\alpha}{\Rightarrow}$ to denote that there exists a $s' \in S$ such that $s \stackrel{\alpha}{\Rightarrow} s'$. The behavior of a system is specified by its set of traces, which are finite sequences of trajectories. (The internal actions are abstracted away in the traces.)

Definition 5 (Trace) For HLTS $\mathcal{H}$, a trace is a finite sequence $\alpha \in L^*$ such that $s_0 \stackrel{\alpha}{\Rightarrow}$, where s_0 is the initial state of $\mathcal{H}$.

The length of a trace is defined as the number of elements of the sequence and is represented by $|\alpha|$. We denote the set of all traces of $\mathcal{H}$ by $traces(\mathcal{H})$.

Example 1 *Consider the bouncing ball example. Collision with the floor can be considered as an internal action. Also, the curves f_1, f_2, and f_3 in Fig. 1b denote some trajectories of the system (projected only to one variable h). Hence, the sequences $f_1 f_2$ and $f_1 f_2 f_3$ are two traces of this system.*

An HLTS can be input-enabled, as defined in the following.

Definition 6 (Input-Enabled HLTS) An HLTS is *input-enabled* if

$$\forall s \in S, \forall \tau \in trajs(V_I) : \exists \tau' \in trajs(V) \text{ such that } \tau' \downarrow V_I = \tau \wedge s \stackrel{\tau'}{\Rightarrow}.$$

2.1.3 Timed State Sequence (TSS) Model

The approach proposed in [AHF+14] specifies a hybrid system as a mapping from pairs of initial conditions and input signals to output signals. The input and output signals of the system are described through a notion of timed-state sequence (TSS). (We slightly adapt the notation to make it consistent among the different semantic models.)

Definition 7 (Hybrid-Timed State Sequence (TSS) [AHF+14]) Consider $N \in \mathbb{N}$, $\mathbb{T} = \mathbb{R}_{\geq 0} \times \mathbb{N}$, and a set of variables V. A hybrid-timed state sequence (TSS) is defined as pair (x,σ), where $x \in (Val(V))^N$ and $\sigma \in \mathbb{T}^N$. The i th element of a TSS (x,σ) is denoted by (x_i,σ_i), where $\sigma_i=(t_i, j_i) \in \mathbb{T}$. Also, we denote the set of all TSSs defined over the set of variables V, considering a specific $N \in \mathbb{N}$, by $\mathbb{TSS}(V,N)$.

A TSS describes the valuation of a set of (input/output) variables in a finite number of time instants. In this notion, a time instant is denoted by a pair (t, j), where t is a real number denoting the real time, and j is an integer which denotes the number of discrete jumps until that time instant. We refer to this semantic model as the TSS model.

Example 2 *The sequence $((1,1,1),((0,0),(4.5,0),(6,1)))$ is a TSS for the running example, defined for the variable a, where the tuple $(1,1,1)$ denotes the value of a in three time instants $(0,0)$,$(4.5,0)$, and $(6,1)$. For instance, the pair $(6,1)$ shows the instant that real time is 6 and the system has experienced one discrete jump. Similarly, the sequence*

$$((5,6.55,7.1,6.65,5.2,2.75,0.19),((0,0),(1,0),(2,0),(3,0),(4,0),(5,0),(6,1)))$$

denotes an output TSS for the considered system, denoting a valuation of the output variable h.

Assume a set of input variables V_I with a corresponding compact set of possible values $Val(V_I)$ and a set of output variables V_O with a set of possible valuations $Val(V_O)$. Accordingly, the set of input TSSs and output TSSs can be respectively defined by $\mathbb{TSS}(V_I,N)$ and $\mathbb{TSS}(V_O,N)$. Also, let H be the set of initial conditions.

A hybrid system can be viewed as a mapping between the initial condition and the input TSSs $(u,\sigma_u) \in \mathbb{TSS}(V_I,N)$ to the output TSSs $(y,\sigma_y) \in \mathbb{TSS}(V_O,N)$.

Definition 8 (Hybrid System [AHF+14]) Hybrid system $\mathcal{H}$ is modeled as a mapping:

$$\mathcal{H} : H \times \mathbb{TSS}(V_I, N) \mapsto \mathbb{TSS}(V_O, N)$$

We write $y_{\mathcal{H}}(h_0, (u, \sigma_u))$ to denote the output TSS to which the pair (u, σ_u) is mapped by $\mathcal{H}$, considering h_0 as the initial condition. In [AHF+14], it is assumed that the system is always input-enabled, with the following definition of input-enabledness.

Definition 9 (Input-Enabled System) A hybrid system $\mathcal{H}$ is *input-enabled* if

$$\forall h_0 \in H, \forall (u, \sigma_u) \in \mathbb{TSS}(V_I, N) : \exists (y, \sigma_y) \in \mathbb{TSS}(V_O, N) \text{ such that } (y, \sigma_y) = y_{\mathcal{H}}(h_0, (u, \sigma_u)).$$

In other words, the system is input-enabled if it produces an output for every initial condition and input TSS.

2.2 Differences in Semantic Models

The following list provides an overview of the fundamental differences among the two underlying semantic models, reviewed in Subsection 2.1. After explaining each difference, we state the assumption we have made for the common semantic domains that makes a meaningful comparison possible.

- Explicit discrete interactions vs. unlabeled jumps. In HLTSs, an explicit notion of input/output actions is introduced, while TSSs provide no explicit means for modeling discrete input/output actions of the system. In order to have a unified semantic model for the two models, we do not consider explicit interactions in HLTSs and model all discrete jumps of the system as internal actions when using HLTSs. (**A1**)

- Partial specifications vs. input-enabled models. HLTSs used for hioco allow for partial specifications, in which the (output) response to certain (input) traces is left unspecified. However, the input-enabledness assumption made in the TSS model implies that for every sequence of input signal, the model should specify an output behavior of the system, expressed in terms of an output TSS. To allow for a meaningful comparison, we only consider input-enabled models. (**A2**)

- Unique initial state vs. arbitrary initial condition. HLTSs specify a unique initial state, while TSSs allow for arbitrary initial conditions. In order to make the two models compatible, we assume a singleton set of initial condition for the TSS model. (**A3**)

- Nondeterministic specifications vs. deterministic models. HLTSs allow for non-deteminism both in the discrete and the continuous behavior, as an abstraction mechanism for unspecified / irrelevant details. However, hybrid systems in the hconf model are mappings from input TSSs to output TSSs and hence, are deterministic in nature. To make the comparison possible, we only consider deterministic models. (**A4**)

- Continuous trajectories vs. discretized samples. HLTSs allow for specification of continuous trajectories while hybrid systems in the hconf theory are necessarily discretized.

This alone may not be a major issue (i.e., hioco is also defined for discretized samples in [vO09, Chapter 6]); however, in combination with determinism and input-enabledness this assumption has far-reaching consequences, namely, if two continuous input trajectories have the same discretized behavior for one arbitrary sampling, they should lead to the same output behavior. To make the comparison feasible, we assume the latter property in the models studied in the remainder of this paper. (**A5**)

Considering these differences one comes away with the impression that the two models have distinct purposes and strengths. In particular, HLTSs as the semantic models of hioco are suitable for high-level partial specifications that leave some room for future design decisions and also only specify certain aspects of the system. However, TSSs used as the semantic models of hconf are suitable for low level specifications that provide a complete and deterministic (discretized) recipe for implementation. An ideal notion of conformance, in our view, should relax assumptions (**A1**)-(**A5**). In other words, an ideal theory of conformance should combine the liberal semantic domain of hioco with the practical conformance relation of hconf, as pointed out in the next section.

2.3 Conformance Relations

In this subsection, we review the two designated notions of conformance testing for hybrid systems, based on the formal models defined in Subsection 2.1.

2.3.1 HIOCO

In this section, we review the hybrid input-output conformance (*hioco*) theory [vO06], simplified according to the assumptions made in Section 2.2.

Definition 10 (**after** Operator) For an HLTS $\mathcal{H}$ and a trace $\alpha \in \mathit{traces}(\mathcal{H})$, we define

$$\mathcal{H} \textbf{ after } \alpha = \left\{ s \,\middle|\, s_0 \overset{\alpha}{\Rightarrow} s \right\}$$

Definition 11 (Trajectories of a State) For an HLTS $\mathcal{H}$ and a state s, **traj**(s) is defined as

$$\textbf{traj}(s) = \left\{ \sigma \in \mathit{trajs}(V) \mid s \overset{\sigma}{\Rightarrow} \right\}$$

This definition can also be extended to a set of states $C \subseteq S$ as $\textbf{traj}(C) = \bigcup_{s \in C} \textbf{traj}(s)$

Using the above-given definitions, we are now ready to define the notion of hioco (simplified by neglecting discrete actions and restriction to input-enabled specifications).

Definition 12 (Hybrid I/O Conformance [vO06]) Consider an HLTS $\mathcal{S}$; an input-enabled HLTS $\mathcal{I}$ is said to be hybrid input-output conforming to $\mathcal{S}$, denoted by $\mathcal{I} \mathit{hioco}\ \mathcal{S}$, if and only if for all traces $\alpha \in \mathit{traces}(\mathcal{S})$:

$$\textbf{traj}\ (\mathcal{I} \textbf{ after } \alpha) \subseteq \textbf{traj}\ (\mathcal{S} \textbf{ after } \alpha)$$

The symmetric kernel of *hioco*, is denoted by $=_{hioco}$. Under the assumption of Section 2.2, *hioco* is a pre-order and $=_{hioco}$ is an equivalence relation.

2.3.2 HCONF

To define the notion of conformance, the notion of (τ,ε)-closeness is defined to measure how much a given TSS deviates from another TSS. Informally, two TSSs are said to be (τ,ε)-close if, for any time interval of length τ, and each sampled point in one, there exists a sample point in the other such that the difference between the valuations in the two points is less than ε. Based on this notion, two systems are conforming if, for all combinations of initial conditions and input TSSs, the respective output TSS of the two systems are close to each other.

In order to be consistent with the hioco conformance relation, we only consider the conformance relation for TSS models for $(0,0)$-closeness [AHF+14]. Consider a maximum number of jumps $N \in \mathbb{N}$ for which we are to perform the test.

Definition 13 (hconf [AHF+14]) Two systems $\mathcal{H}_M$ and $\mathcal{H}_I$ are said to be conforming if for any input TSS $(u,\sigma_u) \in \mathbb{TSS}(V_I,N)$

$$y_{\mathcal{H}_M}(h_0,(u,\sigma_u)) = y_{\mathcal{H}_I}(h_0,(u,\sigma_u))$$

We write $\mathcal{H}_M$ hconf $\mathcal{H}_I$ to denote that two systems $\mathcal{H}_M$ and $\mathcal{H}_I$ are conforming.

2.4 Differences in Conformance Relations

In addition to the fundamental differences in the semantic domain, which stated in Subsection 2.2, there are also fundamental differences in the way the two conformance relations are defined. Below, we provide a concise list of such fundamental differences:

- Pre-order vs. equivalence relation. The conformance relation hioco is a pre-order (for input-enabled specifications), while hconf is an equivalence relation. This stems from a fundamental difference in whether the implementation may choose from the alternative behaviors of the specification (pre-order view), or the implementation should implement the behavior prescribed by the specification (equivalence view). (The notion of closeness in hconf to some extent remedies this, and allows for deviations in the implementation.) In order to make our comparison possible, we consider the symmetric kernel of hioco for input-enabled specification, denoted by $=_{hioco}$, and compare it with hconf.

- Projecting on specification traces/trajectories vs. considering all traces. Since the semantic model of hioco allows for partial specification, the conformance relation also exploits that by comparing the behavior of the implementation and specification only with respect to those traces specified in the specification. This can potentially create major differences between hioco and hconf. However, since we restricted our specifications to input-enabled and deterministic ones, this difference is not visible in our results to follow.

- Exact vs. approximate comparison of trajectories. Conformance relation hioco compares the continuous behaviors precisely, while hconf defines a notion of temporal and spatial closeness is used to represent and measure the deviation of the implementation from the specification. As indicated in the aforementioned claim of [AHF+14], we only consider the exact version of hconf and compare it with hioco.

- Sensitivity to quiescence. In order to reject implementations that do not produce any output when they should, the notion of quiescence is employed in hioco. Also, to denote existence or absence of continuous trajectories the notion of agile states is introduced. These notion are altogether absent in hconf, because hconf does not allow for the absence of outputs. In the case of deterministic input-enabled models (both for specifications and implementations, implied by our assumptions (**A1**)-(**A5**)), this difference is immaterial; however, once the models are relaxed to allow for non-deterministic and/or partial models, this difference may differentiate between the two notions of conformance.

3 Translating Hybrid Automata to the Other Models

In this section, we employ the notion of hybrid automata as a general formalism for hybrid systems and show how a hybrid automaton satisfying assumptions (**A1**)-(**A5**) can be translated to each of the two considered semantic domains. For the purpose of this paper, we deal with a minimal definition of hybrid automaton as considered in [GST09], in which there is no distinction between input and output actions.

Definition 14 (Hybrid Automata [GST09]) A hybrid automaton is defined as a tuple (Loc, V, (l_0,v_0), $\rightarrow$, I, F), where

- Loc is a finite set of locations;
- V is the set of continuous variables;
- l_0 denotes the initial location and v_0 is an initial valuation of V;
- $\rightarrow \subseteq Loc \times \mathcal{B}(V) \times Reset(V) \times Loc$ is the set of jumps where:
  - $\mathcal{B}(V) \subseteq Val(V)$ indicates the guards under which the switch may be performed, and
  - $Reset(V) = \bigcup_{V' \subseteq V} Val(V')$ is the set of all value assignments to all or a subset of the variables V;
- $I : Loc \rightarrow \mathcal{B}(V)$ determines the allowed valuation of variables in each location (called the invariant of the location);
- $F : Loc \rightarrow \mathcal{B}(V \bigcup \dot{V})$ describes some constraints on variables and their derivatives and specifies the allowed continuous behavior in each location.

As before, we write $l \xrightarrow{g,r} l'$ for $(l,g,r,l') \in \rightarrow$.
The dynamic behavior of a hybrid automaton can be specified by a set of *solutions*. A solution to a hybrid automaton is a function on a hybrid time domain, which is defined as follows.

Definition 15 (Hybrid Time Domain [GT06, GST09]) A set $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is a hybrid time domain if either:

- $E = \bigcup_{j=0}^{J-1} ([t_j, t_{j+1}], j)$, for a finite J, where $t_0 = 0$ and $t_0 \leq t_1 \leq t_2 \leq \ldots \leq t_J$, $t_J \leq \infty$; or

- $E = \bigcup_{j=0}^{\infty}([t_j, t_{j+1}], j)$, with $t_0 = 0$ and $t_0 \leq t_1 \leq t_2 \leq ...$

For a hybrid time domain, we define time interval I_j to be $[t_j, t_{j+1}]$. A time instant in the hybrid time domain E is defined as a pair $(t, j) \in E$.

Definition 16 (Solution of a Hybrid Automaton [Lem00]) A solution to the hybrid automaton $\mathcal{HA} = (Loc, V, (l_0, v_0), \rightarrow, I, F)$ is a function x : $E \rightarrow Loc \times Val(V)$, where E is a hybrid time domain partitioned into a finite number J of intervals, and

- $x(0,0) = (l_0, v_0)$;
- for a fixed j, $x(t, j) : t \rightarrow l_j \times Val(V)$ is a function over real time that satisfies $I(l_j)$ and $F(l_j)$; and
- $\forall j\ (J - 1 > j \geq 0) : \exists g \in \mathcal{B}(V), r \in Reset(V)$ such that $\left(l_j \xrightarrow{g,r} l_{j+1}\right) \wedge$ $(x(t_{j+1}, j) \downarrow V$ satisfies $g) \wedge (x(t_{j+1}, j+1) \downarrow V = r)$.

The set of all solutions of a hybrid automaton $\mathcal{HA}$ is denoted by $Solutions(\mathcal{HA})$. For a given hybrid automaton $\mathcal{HA}$, the determinism assumption **(A4)** yields:

$$\forall s, s' \in Solutions(\mathcal{HA}) : (s \downarrow V_I = s' \downarrow V_I) \Rightarrow (s \downarrow V_O = s' \downarrow V_O)$$

The length of a solution s is defined as the number of intervals in dom s.

3.1 Translating Hybrid Automata to Hybrid Labeled Transition Systems

According to Definition 3 and Definition 14, a hybrid automaton denotes an HLTS, as follows.

Definition 17 ($[\![.]\!]_{HLTS}$ [vO06]) A hybrid automaton $\mathcal{HA} = (Loc, V, (l_0, v_0), \rightarrow, I, F)$ can be converted to an equivalent HLTS $[\![\mathcal{HA}]\!]_{HLTS} = (Loc \times Val(V), (l_0, v_0), trajs(V), \rightarrow)$, where

- $\rightarrow = \{(l,u) \xrightarrow{\sigma} (l',u') | \left(\exists_{g,r} : l \xrightarrow{g,r} l'\right) \wedge (u \in g) \wedge (u \in I(l)) \wedge (u' \in r) \wedge (u' \in I(l')) \wedge$ $(u = \sigma.fval \wedge u' = \sigma.lval)\}$ $\bigcup$ $\{(l,u) \xrightarrow{\sigma} (l,u') | \exists_{\sigma \in trajs(V)} : (u = \sigma.fval) \wedge (u' = \sigma.lval) \wedge (\sigma\ satisfies\ F(l)) \wedge (\forall t \in dom(\sigma) :$ $\sigma(t) \in I(l))\}$
  where, $\sigma.fval$ and $\sigma.lval$ respectively denote the value of σ at its starting (first) point and ending (last) point.

3.2 Translating Hybrid Automata to Timed State Sequences

In order to convert a hybrid automaton to a TSS, we use the concept of the *solution* of the hybrid automata.

Definition 18 ($[\![.]\!]_{TSS}$) Consider a hybrid automaton $\mathcal{HA}$ with the set of input variables V_I and the set of output variables V_O. Further, let $Solutions(\mathcal{HA})$ denote the set of all solutions of $\mathcal{HA}$. Thus, the TSS model which is defined by $\mathcal{HA}$, denoted by $[\![\mathcal{HA}]\!]_{TSS}$, is the mapping:

$$\mathcal{H} : (u, \sigma_u) \mapsto (y, \sigma_y)$$

that is constructed from the set of solutions in such a way that, for any solution $x \in \mathit{Solutions}(\mathcal{HA})$, any input TSS (u, σ_u) obtained by discretizing $x \downarrow V_I$ is mapped to an output TSS (y, σ_y), where

- $\sigma_y = \sigma_u$;
- $y_i = x(t_i, j_i) \downarrow V_O$, where (t_i, j_i) is the i th element of σ_u.

4 Equivalence of hioco and hconf

Theorem 1 *Given a specification $\mathcal{S}$ in terms of a hybrid automaton and an implementation of which the behavior is expressed by a hybrid automaton $\mathcal{I}$. If both $\mathcal{S}$ and $\mathcal{I}$ satisfy conditions **A1-A5**, then it holds that $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$ if and only if $[\![\mathcal{S}]\!]_{HLTS} =_{hioco} [\![\mathcal{I}]\!]_{HLTS}$.*

Proof. The idea of the proof is that we define a notion of hybrid conformance relation for hybrid automata and show that both hioco and hconf are equivalent to this notion of conformance. The proof comprises the following two steps:

1. $[\![\mathcal{S}]\!]_{HLTS} =_{hioco} [\![\mathcal{I}]\!]_{HLTS} \Leftrightarrow$

$$\forall s \in \mathit{Solutions}(\mathcal{S}), \forall s' \in \mathit{Solutions}(\mathcal{I}) : (s \downarrow V_I = s' \downarrow V_I) \Rightarrow (s \downarrow V_O = s' \downarrow V_O)$$

2. $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS} \Leftrightarrow$

$$\forall s \in \mathit{Solutions}(\mathcal{S}), \forall s' \in \mathit{Solutions}(\mathcal{I}) : (s \downarrow V_I = s' \downarrow V_I) \Rightarrow (s \downarrow V_O = s' \downarrow V_O)$$

To prove step 1, we provide a method for converting a solution in a hybrid automaton $\mathcal{H}$ to an equivalent trace in the respective HLTS, and conversely, to convert a trace to a corresponding solution.

Lemma 1 (Traces and Solutions Equivalence) *Given a solution $s \in \mathit{Solutions}(\mathcal{H})$, one can uniquely obtain a trace $\tau \in \mathit{traces}([\![\mathcal{H}]\!]_{HLTS})$, with $|\tau| = |s|$, and vice versa.*

Proof. For a fixed $N \geq 0$, consider a trace $\tau = \alpha_0\alpha_1 ... \alpha_{N-1} \in \mathit{traces}([\![\mathcal{H}]\!])$ with $|\tau| = N$, and let D_j denote the continuous time interval over which α_j is defined. We can associate a solution $s \in \mathit{Solutions}(\mathcal{H})$ with τ as follows. We specify the domain of s as $\mathrm{dom}\, s = \bigcup_{j=0}^{N-1} I_j$, where I_j is defined as:

- $I_0 = [D_0, 0]$;
- $I_j = ([t_j, t_{j+1}], j)$, where $t_{j+1} = t_j + |D_j|$ for $j > 0$;

in which $|D_j|$ denotes the length of time interval D_j. Furthermore, the value of the solution s at time instant (t, j) is specified by the value of the respective trajectory at that instant, i.e., $s(t, j) = \alpha_j(t - t_j)$. Conversely, it can be shown that given a solution $s \in \mathit{Solutions}(\mathcal{H})$, one can uniquely obtain a trace $\tau \in \mathrm{traces}([\![\mathcal{H}]\!]_{HLTS})$, with $|\tau| = |s|$. □

Next, we show that $[\![\mathcal{S}]\!]_{HLTS} =_{hioco} [\![\mathcal{I}]\!]_{HLTS}$ means that $traces([\![\mathcal{S}]\!]_{HLTS}) = traces([\![\mathcal{I}]\!]_{HLTS})$. Then, using Lemma 1, the first step of the proof can be readily concluded.

Lemma 2 *Consider two hybrid automata $\mathcal{S}$ and $\mathcal{I}$. It holds that $[\![\mathcal{S}]\!]_{HLTS} =_{hioco} [\![\mathcal{I}]\!]_{HLTS}$ if and only if* $\text{traces}([\![\mathcal{S}]\!]_{HLTS}) = \text{traces}([\![\mathcal{I}]\!]_{HLTS})$.

Proof. For an HLTS $\mathcal{H}$, let $\text{traces}_N(\mathcal{H})$ denote the set of all traces of $\mathcal{H}$ which are of length N. In other words, $traces_N(\mathcal{H}) = \{\alpha \in \text{traces}(\mathcal{H}) : |\alpha| = N\}$. The goal is to show, given $[\![\mathcal{S}]\!]_{HLTS} =_{hioco} [\![\mathcal{I}]\!]_{HLTS}$, that $traces_N([\![\mathcal{S}]\!]_{HLTS}) = traces_N([\![\mathcal{I}]\!]_{HLTS})$ for any $N \geq 0$. We proceed by induction on N.

We consider $N = 0$ as the base of induction, which includes empty trace ε. According to Definition 4, and also with regard to Definition 5, it holds that $\varepsilon \in traces([\![\mathcal{S}]\!]_{HLTS})$ and $\varepsilon \in traces([\![\mathcal{I}]\!]_{HLTS})$. Therefore, $traces_0([\![\mathcal{S}]\!]_{HLTS}) = traces_0([\![\mathcal{I}]\!]_{HLTS})$.

For the induction step, assuming that $traces_N(\mathcal{S}) = traces_N(\mathcal{I})$, we need to show $traces_{N+1}(\mathcal{S}) = traces_{N+1}(\mathcal{I})$. To this end, consider that $traces_{N+1}(\mathcal{I})$ and $traces_{N+1}(\mathcal{S})$, which can be defined as follows:

$$traces_{N+1}(\mathcal{I}) = \{\tau\alpha \mid \tau \in traces_N(\mathcal{I}) \wedge \alpha \in \textbf{traj}\ (\mathcal{I}\ \textbf{after}\ \tau)\}$$

and

$$traces_{N+1}(\mathcal{S}) = \{\tau\alpha \mid \tau \in traces_N(\mathcal{S}) \wedge \alpha \in \textbf{traj}\ (\mathcal{S}\ \textbf{after}\ \tau)\}$$

We have that $traces_N(\mathcal{S}) = traces_N(\mathcal{I})$, and also according to Definition 12, $\textbf{traj}\ (\mathcal{S}\ \textbf{after}\ \tau) = \textbf{traj}\ (\mathcal{I}\ \textbf{after}\ \tau)$, we have $traces_{N+1}(\mathcal{S}) = traces_{N+1}(\mathcal{I})$, which completes the induction step. (Note that according to assumptions **A2** and **A4** in Subsection 2.2, Definition 12 establishes the former equality.) □

For step 2 of the proof of Theorem 1, we first show that if $Solutions(\mathcal{S}) = Solutions(\mathcal{I})$ then $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$. Note that the method for translating a hybrid automaton to a *TSS* described in Definition 18 is deterministic. Moreover, the translation method constructs the respective TSS solely based on the set of solutions of the hybrid automaton. Therefore, for two hybrid automata $\mathcal{S}$ and $\mathcal{I}$ with the same set of solutions, the resultant TSSs are trivially equal. According to Definition 13, this means that $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$.

To prove step 2, we further need to show that if $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$ then $Solutions(\mathcal{S}) = Solutions(\mathcal{I})$. We proceed with proof by contradiction. Assume that $[\![\mathcal{S}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$ and $Solutions(\mathcal{S}) \neq Solutions(\mathcal{I})$. Therefore, it holds that

$$\exists s \in Solutions(\mathcal{I}) \text{ such that } \forall s' \in Solutions(\mathcal{S}) : s \neq s'.$$

But, due the assumption that the considered hybrid automata are input-enabled, there exists an $s' \in Solutions(\mathcal{I})$ for which $s \downarrow V_I = s' \downarrow V_I$. But, as $s \neq s'$, we can conclude $s \downarrow V_O \neq s' \downarrow V_O$, which means that there is a time instant (t, j) for which $s(t, j) \neq s'(t, j)$. Consider the input TSS obtained by discretization of s, denoted as u that includes time instant (t, j). According to the translation method from hybrid automaton to TSS described in Definition 18, the output TSS to which $[\![\mathcal{S}]\!]_{TSS}$ maps the input TSS u differs from that of $[\![\mathcal{I}]\!]_{TSS}$, which contradicts with the assumption of $[\![\mathcal{I}]\!]_{TSS}$ hconf $[\![\mathcal{I}]\!]_{TSS}$. □

5 Conclusion

We studied two fundamental notions of conformance testing for cyber-physical systems, which are to our knowledge the most notable notions of their kind. We have pointed out fundamental differences, both in their semantic domains and in their definition of conformance. We identified a set of conditions under which the two notions are comparable. We proved that under such conditions, the two notions coincide.

While hioco is based on a richer and more expressive semantic domain, it suffers from practical concerns, which are related to its underlying infinite state-space. On the other hand, hconf provides a more practical approach to checking conformance, but it ignores some important aspects in modeling, including explicit discrete interactions, nondeterminism, and partial specifications.

The dichotomy reflected in the difference in the two notions suggests that a natural next step would be to define a more general notion of conformance testing for cyber-physical systems, which consolidates the theoretical expressiveness of hioco with the practical approach of hconf to conformance checking. Defining a test-case generation algorithm and proving soundness and adequacy of the generated test cases (given some testing hypothesis) are among the future milestones.

Acknowledgements: We would like to thank the anonymous reviewers of AVOCS'14, Houssam Abbas, Harsh Beohar, Pieter Cuijpers, Jim Kapinski, Mehdi Kargahi, and Mahsa Varshosaz for their helpful comments.

Bibliography

[AHF+14] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, K. Ueda. Conformance testing as falsification for cyber-physical systems. In *ICCPS*. 2014.
Full paper available from: http://arxiv.org/abs/1401.5200

[AFM+14] H. Abbas, G. Fainekos, and H. Mittelmann. Formal property verification in a conformance testing framework. In *MEMOCODE*. 2014.

[ABW09] B. K. Aichernig, H. Brandl, F. Wotawa. Conformance testing of hybrid systems with qualitative reasoning models. *ENTCS* 253(2):53–69, Oct. 2009.

[AD94] R. Alur, D. L. Dill. A theory of timed automata. *Theoretical computer science* 126(2):183–235, 1994.

[BB05] L. B. Briones, E. Brinksma. A Test Generation Framework for Quiescent Real-time Systems. In *Proceedings of the 4th International Conference on Formal Approaches to Software Testing*. FATES'04, pp. 64–78. Springer-Verlag, Berlin, Heidelberg, 2005.

[BK06] E. Bringmann, A. Krämer. Systematic Testing of the Continuous Behavior of Automotive Systems. In *Proceedings of the 2006 International Workshop on Software Engineering for Automotive Systems*. SEAS '06, pp. 13–20. ACM, 2006.

[BWA10] H. Brandl, M. Weiglhofer, B. Aichernig. Automated Conformance Verification of Hybrid Systems. In *International Conference on Quality Software (QSIC)*. Pp. 3–12. July 2010.

[CR08] P. J. Cuijpers, M. A. Reniers. Lost in translation: Hybrid-time flows vs. real-time transitions. In *Hybrid Systems: Computation and Control*. Pp. 116–129. Springer, 2008.

[CRH02] P. J. L. Cuijpers, M. A. Reniers, W. P. M. H. Heemels. *Hybrid transition systems*. Computer Science Reports 02-12, TU Eindhoven, 2002.

[Dan10] T. Dang. Model-based testing of hybrid systems. *Monograph in Model-Based Testing for Embedded Systems, CRC Press*, 2010.

[GST09] R. Goebel, R. Sanfelice, A. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine* 29(2):28–93, April 2009.

[GT06] R. Goebel, A. R. Teel. Solutions to Hybrid Inclusions via Set and Graphical Convergence with Stability Theory Applications. *Automatica* 42(4):573–587, Apr. 2006.

[Lem00] M. Lemmon. On the Existence of Solutions to Controlled Hybrid Automata. In *Hybrid Systems: Computation and Control*. LNCS 1790, pp. 229–242. Springer, 2000.

[Lib03] D. Liberzon. *Switching in systems and control*. Springer, 2003.

[LSV03] N. Lynch, R. Segala, F. Vaandrager. Hybrid i/o automata. *Information and Computation* 185(1):105–157, 2003.

[MS97] O. Müller, P. Scholz. Functional specification of real-time and hybrid systems. In *Hybrid and Real-Time Systems*. LNCS 1201, pp. 273–285. Springer, 1997.

[vO06] M. van Osch. Hybrid input-output conformance and test generation. In *Formal Approaches to Software Testing and Runtime Verification*. LNCS 4262, pp. 70–84. Springer, 2006.

[vO09] M. van Osch. Automated Model-based Testing of Hybrid Systems. *Faculty of Mathematics and Computer Science, TU Eindhoven*, 2009.

[Tre96] J. Tretmans. Test Generation with inputs, outputs and repetitive quiescence. *Software - Concepts and Tools* 17(3):103–120, 1996.

[WLT13] M. Woehrle, K. Lampka, L. Thiele. Conformance testing for cyber-physical systems. *ACM Trans. Embed. Comput. Syst.* 11(4):84:1–84:23, Jan. 2013.