Sound Conformance Testing for Cyber-Physical Systems: Theory and Implementation

Hugo Araujo[a], Gustavo Carvalho[a], Morteza Mohaqeqi[b], Mohammad Reza Mousavi[c,d], Augusto Sampaio[a]

[a] *Universidade Federal de Pernambuco, Brazil*
[b] *Uppsala University, Sweden*
[c] *Halmstad University, Sweden*
[d] *University of Leicester, UK*

Abstract

Conformance testing is a formal and structured approach to verifying system correctness. We propose a conformance testing algorithm for cyber-physical systems, based on the notion of hybrid conformance by Abbas and Fainekos. We show how the dynamics of system specification and the sampling rate play an essential role in making sound verdicts. We specify and prove error bounds that lead to sound test-suites for a given specification and a given sampling rate. We use reachability analysis to find such bounds and implement the proposed approach using the CORA toolbox in Matlab. We apply the implemented approach on a case study from the automotive domain.

Keywords: Cyber-physical Systems, Model-Based Testing, Conformance Testing, Soundness, Reachability Analysis

1. Introduction

A cornerstone of model-based testing (MBT) [1] is to systematically generate test-cases from a test model, i.e., a specification of system's correct behavior. A rigorous notion of MBT aims at establishing a *conformance relation* by running a number of such generated test-cases [2, 3, 4, 5, 6]. Conformance relation is the mathematical formalization of the test technique, which is typically defined as a relation on a common semantic domain of both the test model and the system under test. We refer to such a mathematically-founded notion of MBT, where a rigorous notion of conformance is defined and is properly related to the testing technique, as *conformance testing*.

Figure 1 provides a schematic view of the notions of conformance relation, conformance testing and the correspondence between them. On the top-left corner, the MBT trajectory starts with a test model. This model is the result of a translation of the requirements, which typically describes the "interesting" scenarios of interaction to test as well as their expected outputs. The top part of Figure 1 depicts the practical part of MBT, which starts with generating test cases from the test model, applying them to the system under test, and analyzing the outcomes in an iterative manner. The bottom part of Figure 1 puts these practical activities on a firm theoretical ground. To this end, the test model is given a mathematical meaning, through a mapping called formal semantics. It is also assumed that the system under test *can* be given a mathematical meaning; however, the mathematical object (e.g., the state machine) capturing the behavior of the system under test may be too large to generate and analyse. Hence, the dashed line depicts a purported (not practically available) formal semantics of the system under test, bringing both the test model and the system under test in a common semantic model. The common semantic domain is used to

Email addresses: `hlsa@cin.ufpe.br` (Hugo Araujo), `ghpc@cin.ufpe.br` (Gustavo Carvalho), `morteza.mohaqeqi@it.uu.se` (Morteza Mohaqeqi), `m.r.mousavi@hh.se` (Mohammad Reza Mousavi), `acas@cin.ufpe.br` (Augusto Sampaio)

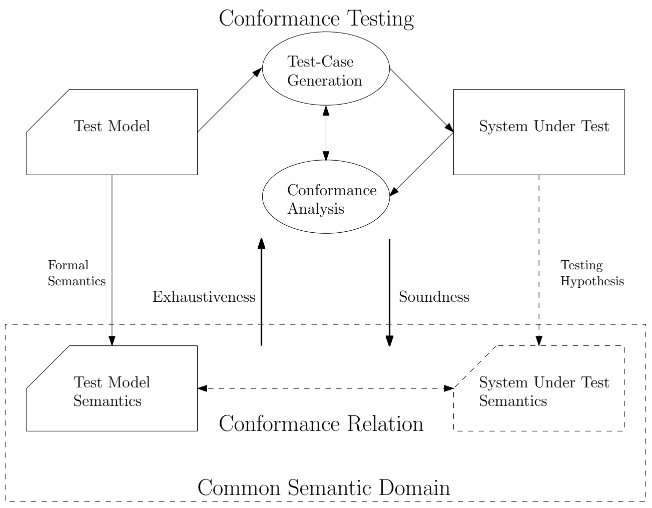


Figure 1: Schematic View of Conformance Testing and Conformance Relation

facilitate specifying and reasoning about the conformance relation, but in practice, one does not typically have access to such a rigorous description of behavior, particularly of the system under test (hence, the common semantic domain and the conformance relation are drawn in dashed line denoting that they only exist in theory). Using these two models and the common semantic domain, one devises a conformance relation defining what it means mathematically for a system to conform to a specification. The model-based testing trajectory should ideally precisely characterise the mathematical conformance relation, denoted by the downward and upward arrows in Figure 1 representing soundness and exhaustiveness, respectively.

Some notions of conformance testing for ordinary reactive systems are both sound and exhaustive [7, 8]. However, for cyber-physical systems (CPSs), it is far from trivial to come up with a sound and exhaustive, yet practical, notion of conformance testing. Conformance testing of CPSs [4, 9] often involves discrete sampling of continuous signals and hence, for any realistic notion of conformance testing, error margins should be accommodated to allow for slight deviations (e.g., measurement errors) in time and value [10, 2, 3]. The interaction among the continuous dynamics, the sampling rates and the error margins is an intricate one and if the aforementioned parameters are not in sync, the resulting conformance testing method can be unsound. Exhaustiveness is even more intricate and requires detailed information about the continuous dynamics of the implementation under test (as well as the specification); we focus on soundness and address exhaustiveness briefly and in passing towards the end of this paper.

1.1. Problem definition and contributions

Given a specification and a sampling rate, we seek sufficient conditions for a test-suite under which conformance testing is sound with respect to a given conformance relation. That is, the test-suite only fails on non-conforming implementations.

To make this problem more specific, we take the notion of hybrid conformance by Abbas and Fainekos [10, 2, 3] as our conformance relation. We then define a straightforward conformance testing algorithm based on this notion and study its soundness. As it turns out, for all reasonable specifications (e.g., specifications with countable minima and maxima in each finite interval) such a conformance testing algorithm may in general result in unsound verdicts. Hence, we specify and prove soundness criteria, based on the error margins (in time and value), the properties of specification's continuous dynamics, and the sampling rate, that guarantee soundness of the test verdicts. These results were first presented in a conference publication [11].

In this paper, we revise our earlier results [11] and significantly extend them. Particularly, finding the right bounds for conformance testing requires a thorough analysis of the system dynamics. To this end, we propose a method based on reachability analysis and implement it using the CORA toolbox [12] in Matlab.

To validate our strategy, along with its implementation, we consider a controller to an automotive air-fuel ratio (AFR) control system [13].

1.2. Running Example

To illustrate the notions introduced in the paper, we consider the model of a thermostat [6] as our running example. The thermostat has two operation modes to control the temperature. In mode *ON*, a heater is turned on in order to warm up the environment. During the mode *OFF*, the heater is turned off, which leads to a steady decrease in the environment temperature. There is a threshold for the minimum environment temperature, which triggers the thermostat to switch to mode *ON*. A similar threshold exists for the maximum temperature which makes the thermostat to switch to mode *OFF*. A formal model of the thermostat is defined as a hybrid automaton in Figure 2a, which is explained in the next section.

1.3. Organization

The rest of this paper is organized as follows. In Section 2, we review some notions from the literature concerning hybrid system specification and hybrid conformance. In Section 3, we define our notion of conformance testing for hybrid systems. In Section 4, we first show that in all practical cases, conformance testing can lead to unsound test verdicts and, subsequently, define sufficient conditions on test-suites to produce sound verdicts.

The remainder of this work is a substantial extension of the work presented earlier [11]. In Section 5, we give an overview of reachability analysis for hybrid systems, and show how we use it in our solution to yield sound results. In Section 6, we present how this sound conformance testing strategy is mechanised. In Section 7, we use our tool to validate our approach on a case study based on a combustion engine controller.

In Section 8, we review the related work. Finally, we conclude the paper and present the directions of our ongoing research in Section 9.

2. Preliminaries

In the remainder of this paper, $\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}_+$ denote the set of non-negative integers, real numbers, and non-negative real-numbers, respectively. Consider a set of real-valued variables V. A valuation of V is a function of type $V \to \mathbb{R}$, which assigns a real number to each variable $v \in V$. The set of all valuations of V is denoted by $\mathit{Val}(V)$. Furthermore, the domain of a function f is denoted by $\mathrm{dom}(f)$.

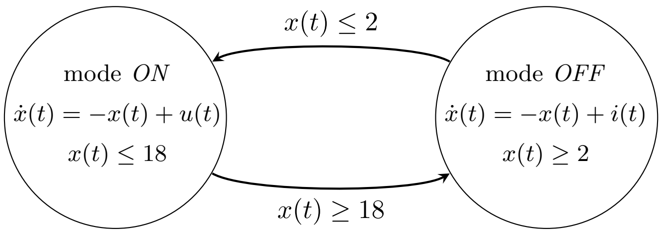
In the following subsections, we first formally specify the notion of hybrid systems and the corresponding concepts. Then, we elaborate a formal definition of a conformance relation for hybrid systems.

2.1. Hybrid System Specifications

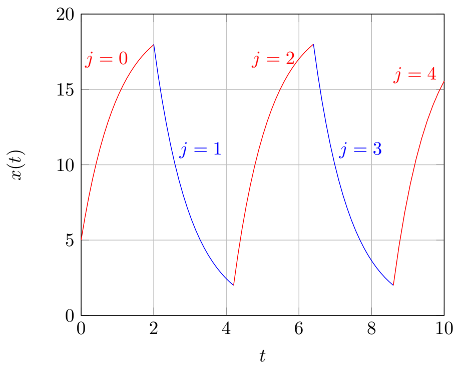
To specify test models for hybrid systems, we use the Hybrid Automata formalism, defined below.

Definition 1 (Hybrid Automata [14]). *A hybrid automaton $\mathcal{HA}$ is defined as a tuple (Loc, V, (l_0,v_0), $\to$, I, F), where*

- *Loc is the finite set of locations;*
- *$V = V_I \uplus V_O$ is the set of continuous variables, where V_I and V_O denote the disjoint sets of input variables and output variables, respectively;*
- *l_0 denotes the initial location and v_0 is an initial valuation of V;*
- *$\to \subseteq Loc \times \mathcal{B}(V) \times Reset(V) \times Loc$ is the set of jumps where:*
  - *$\mathcal{B}(V) \subseteq \mathit{Val}(V)$ indicates the guards under which the jump may be performed, and*
  - *$Reset(V) = \bigcup_{V' \subseteq V} \mathit{Val}(V')$ is the set of value assignments to the variables in V after the jump;*



(a) Hybrid automaton of the thermostat



(b) A sample of the continuous dynamics of the system

Figure 2: Thermostat example

- *$I : Loc \to \mathcal{B}(V)$ determines the allowed valuation of variables in each location (called the invariant of the location);*
- *$F : Loc \to \mathcal{B}\left(V \cup \dot{V}\right)$ describes some constraints on variables and their derivatives and specifies the allowed continuous behavior in each location.*

We denote the set of all hybrid automata by $\mathbb{H}$.

We typically write $l \xrightarrow{g,r} l'$ to denote $(l, g, r, l') \in \to$, where l is the source location of the jump (transition), g is its guard, r is the reset condition of the jump, and l' is its target.

Example 1. *Fig. 2a shows the hybrid automaton of the thermostat described in Section 1.2 with $Loc = \{ON, OFF\}$, $V_I = \{u, i\}$, $V_O = \{x\}$, $(l_0, v_0) = (ON, 5)$, $\to = \{(ON, x(t) \geq 18, \{\}, OFF), (OFF, x(t) \leq 2, \{\}, ON)\}$, $I(ON) = x(t) \leq 18$, $I(OFF) = x(t) \geq 2$, $F(ON) = \dot{x}(t) = -x(t) + u(t)$, and $F(OFF) = \dot{x}(t) = -x(t) + i(t)$.*

The evolution of a hybrid system is defined over a domain of *hybrid* time, defined below.

Definition 2 (Hybrid Time Domain [10])**.** *A hybrid time domain E is a subset of $\mathbb{R}_+ \times \mathbb{N}$ defined as*

$$E = \bigcup_{j=0}^{J-1} [t_j, t_{j+1}] \times \{j\} \tag{1}$$

where J denotes the maximum number of discrete jumps and $0 = t_0 \leq t_1 \leq t_2 \leq \ldots \leq t_J$. We denote the set of all hybrid time domains by $\mathbb{T}$.

The hybrid time domain is used to model the evolution of a hybrid system regarding both evolution of system dynamics (using continuous time intervals $[t_j, t_{j+1}]$) and discrete jumps (using integer numbers j).The following notion of solution gives a semantics to hybrid automata using the notion of hybrid time domain.

Definition 3 (Solution)**.** *A solution to a hybrid automaton $\mathcal{HA} = (Loc, V, (l_0, v_0), \to, I, F)$ is a function $s : E \to Loc \times Val(V)$, where*

- *$s(0, 0) = (l_0, v_0)$;*
- *for each $(t, j) \in \mathrm{dom}(s)$: x satisfies $I(l)$ and $F(l)$, where $(l, x) = s(t, j)$ is the pair of location and valuation at time (t, j); and*

- *for each* $(t_j, j) \in \mathrm{dom}(s)$ *with* $j > 0$*: there exists* $l \xrightarrow{g,r} l'$ *such that* x *satisfies* g *and* (x, x') *satisfies* r*, where* $(l, x) = s(t_j, j-1)$ *and* $(l', x') = s(t_j, j)$ *are the pairs of location and valuation at times* $(t_j, j-1)$ *and* (t_j, j)*, respectively.*

In order to capture the evolution of system dynamics and abstract away from the internal discrete states (i.e., locations), we use the following notion of *trajectory.*

Definition 4 (Trajectory [10][1])**.** *Take a hybrid time domain* E *and a set of variables* V*. A trajectory over* E *is a function* $\phi : E \to \mathit{Val}(V)$*, where* $\forall j, (t, \phi(t, j))$ *is absolutely continuous in* t *over the interval* $I_j = \{t | (t, j) \in E\}$*. The set of all trajectories defined over the variable set* V *is denoted by* $\mathit{Trajs}(V)$*.*

Definition 5 (Trajectory for Hybrid Automata)**.** *Given a hybrid automaton* $\mathcal{HA}$*, a function* $\phi : E \to Val(V)$ *is a trajectory for* $\mathcal{HA}$*, if there exists some solution* s *to* $\mathcal{HA}$ *for which* $\forall (t, j) \in E$*,* $\exists l \in Loc$ *such that* $(l, \phi(t, j)) = s(t, j)$*. The set of all trajectories for* $\mathcal{HA}$ *is denoted by* $\mathit{Trajs}(\mathcal{HA})$*.*

To discriminate input trajectories from output trajectories in a solution, the notion of *solution pair* is defined next. To this aim, we first define the notion of trajectory restriction.

Definition 6 (Trajectory Restriction [5])**.** *Consider a set of variables* V*. The restriction of a valuation* $val \in \mathit{Val}(V)$ *to* $V' \subset V$*, denoted by* $val \downarrow V' \in \mathit{Val}(V')$*, such that* $\forall v \in V'$*,* $(val' \downarrow V')(v) = val(v)$*. Further, the restriction of a trajectory* $\phi : E \to Val(V)$ *to* $V' \subset V$ *is a trajectory* $E \to Val(V')$*, denoted by* $\phi \downarrow V'$*, for which* $(\phi \downarrow V')(t, j) = \phi(t, j) \downarrow V'$*,* $\forall (t, j) \in \mathrm{dom}(\phi)$*.*

Example 2. *Figure 2b shows a trajectory to the thermostat hybrid automaton over a hybrid time domain of* $E = ([0, 2], 0) \cup ([2, 4.2], 1) \cup ([4.2, 6.4], 2) \cup ([6.4, 8.6], 3) \cup ([8.6, 10], 4)$ *after restriction to* $\{x\} \subset V$*.*

Definition 7 (Solution Pair [10])**.** *Let* u *and* y *be two trajectories of types* $E \to Val(V_I)$ *and* $E \to Val(V_O)$*, respectively; (u, y) is a solution pair to a hybrid automaton* $\mathcal{HA}$ *if*

- $\mathrm{dom}(u) = \mathrm{dom}(y)$*, and*
- *there exists a trajectory* ϕ *for* $\mathcal{HA}$ *such that* $\mathrm{dom}(\phi) = \mathrm{dom}(u)$*,* $u = \phi \downarrow V_I$*, and* $y = \phi \downarrow V_O$*.*

Note that by requiring $\mathrm{dom}(\phi) = \mathrm{dom}(u) = \mathrm{dom}(y)$, we make sure that the solution and its input and output pairs are all defined on the same hybrid time domain.

For an example, consider a trajectory ϕ over the hybrid time domain specified in Example 2 where $\phi(t, j)(u) = 20$ and $\phi(t, j)(i) = 2$ for all $(t, j) \in E$. Also, let y be the trajectory described in Example 2. The pair (ϕ, y) constitutes a solution pair to the respective hybrid automaton.

To simplify the forthcoming developments in the current work, we focus on deterministic hybrid automata, defined below. The extension to the non-deterministic case is straightforward and requires iterating over all possible output solutions for a given input. Some initial ideas to this effect are provided by Abbas, Mittelmann and Fainekos [3].

Definition 8 (Deterministic Hybrid Automata)**.** *A hybrid automaton* $\mathcal{HA}$ *with the set of solution pairs* Φ *is deterministic if*

$$\forall u \in E \to Val(V_I), \forall y_1, y_2 \in E \to Val(V_O), ((u, y_1) \in \Phi \text{ and } (u, y_2) \in \Phi) \Rightarrow y_1 = y_2 \tag{2}$$

In this case, we write $y = out_{\mathcal{HA}}(u)$ *to denote* $(u, y) \in \Phi$*.*

[1]Abbas, in his Ph.D. thesis [10], uses the term *Hybrid Arc* to refer to a trajectory.

2.2. Conformance Relation

To define a conformance relation, we assume that both the specification and the purported underlying semantics of the implementation can be captured by some hybrid automata. We use the notion of (τ,ϵ)-closeness, which is defined on the continuous behavior (solution) associated to a hybrid automaton. (We abstract away from the number of discrete jumps, as we consider them irrelevant regarding the observable behavior of the system.)

Definition 9 ((τ,ϵ)-closeness [10][2]). *Consider a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and $\tau, \epsilon > 0$; then two trajectories y_1 and y_2 are said to be (τ,ϵ)-close, denoted by $y_1 \approx_{(\tau,\epsilon)} y_2$, if*

1. *for all $(t,i) \in \mathrm{dom}(y_1)$ with $t \leq T, i \leq J$, there exists $(s,j) \in \mathrm{dom}(y_2)$ such that $|t-s| \leq \tau$ and $\|y_1(t,i) - y_2(s,j)\| \leq \epsilon$, and*
2. *for all $(t,i) \in \mathrm{dom}(y_2)$ with $t \leq T, i \leq J$, there exists $(s,j) \in \mathrm{dom}(y_1)$ such that $|t-s| \leq \tau$ and $\|y_2(t,i) - y_1(s,j)\| \leq \epsilon$.*

Definition 10 (Conformance Relation [10]). *Consider two hybrid automata $\mathcal{HA}_1$ and $\mathcal{HA}_2$. Given a test duration $T \in \mathbb{R}_+$, a maximum number of jumps $J \in \mathbb{N}$, and $\tau, \epsilon > 0$, $\mathcal{HA}_2$* conforms to *HA_1, denoted by $\mathcal{HA}_2 \approx_{(\tau,\epsilon)} \mathcal{HA}_1$, if and only if for all solution pairs (u, y_1) of $\mathcal{HA}_1$, there exists a solution pair (u, y_2) of $\mathcal{HA}_2$ such that the corresponding output trajectories y_1 and y_2 are (τ,ϵ)-close.*

3. Conformance Testing

The conformance relation defined in the previous section assumes access to the solutions of the hybrid automaton underlying the implementation. This is not a realistic assumption in practice. To remedy this, in this section, we define a notion of conformance testing that instead uses sampling of the specification solution to test the implementation outputs at various discrete points. Such a notion of conformance testing should be sound with respect to the conformance relation. Ideally, the notion of conformance testing should be able to generate a test suite that can exhaustively check the conformance relation; however, this turns out to be difficult to achieve in practice.

We define below a sampling mechanism, which involves sampled sequences of the continuous (output) signals.

Definition 11 (Hybrid-Timed State Sequence (TSS) [2]). *Let $N \in \mathbb{N}$ and V be a set of variables. A hybrid-timed state sequence (TSS) is defined as function $x : \mathbb{R}_+ \times \mathbb{N} \to Val(V)$, with $\mathrm{dom}(x) \in (\mathbb{R}_+ \times \mathbb{N})^N$. The value of function x at a specific point $(t,j) \in \mathrm{dom}(x)$ is denoted by $x(t,j)$. Also, we denote the set of all TSSs defined over the set of variables V by $\mathbb{TSS}(V)$.*

Definition 12 (Sampling Function). *Consider $N \in \mathbb{N}$. Any $P \in (\mathbb{R}_+ \times \mathbb{N})^N$ is called a set of sampling points. Take a set of trajectories Y with a set of variables V. Given a set of sampling points P, a sampling function over Y is defined as $\pi_P : Y \to \mathbb{TSS}(V)$, for which, $y_s = \pi_P(y)$ only if*

- $\mathrm{dom}(y_s) = \mathrm{dom}(y) \cap P$
- $\forall (t,j) \in \mathrm{dom}(y_s) : y_s(t,j) = y(t,j)$

A sampling function is periodic when its sampling points are equally distanced. In other words, a sampling function with the set of sampling points P is periodic with period p if and only if $\forall (t_1, j), (t_2, k) \in P$ with $t_1 < t_2$, it must hold that $t_2 - t_1 \geq p$ and moreover, $\nexists (t,l) \in P : t_1 < t < t_2 \Rightarrow t_2 - t_1 = p$. In this case, P is called a periodic set of sampling points with period p.

Next we define the notions of test-suite and test-case. Given an input trajectory and a sampling function, a test case (test suite) provides the expected output valuations at the specified sampling points.

[2]Unlike [10], here we allow different jump numbers (i.e. $i \neq j$) in the definition of (τ,ϵ)-closeness.

Definition 13 (Test-Suite and Test-Case)**.** *A test-suite is defined as a finite set* $TS \subset Trajs(V_I) \times \mathbb{TSS}(V_O)$*. A test-suite* TS *is a valid one for a given hybrid automaton* $\mathcal{HA}$ *only if, for any* $(u, y) \in TS$ *there exists a sampling point set* P *such that* $y = \pi_P(out_{\mathcal{HA}}(u))$*. Each member of a valid test-suite is called a test-case.*

It is worth noting that in the above definition, we provide continuous inputs and observe sampled (discrete) outputs. This is because, in practice, the system receives a continuous input.[3] However, the output behavior of a system is usually observed using a sampled signal [6], which provides a sequence of values in a number of discrete instants.

Based on the definition of periodic sampling points, we define a class of *periodic* test cases.

Definition 14 (Periodic Test-Case)**.** *A test-case* (u, y) *is periodic with period* p *if* dom(y) *is a periodic set of sampling points with period* p*.*

In this paper, we only deal with periodic test-cases. However, the presented results can be extended to the general case.

Execution of a test case $tc = (u, y) \in TS$ on a system representable by a hybrid automaton $\mathcal{HA}_I$ consists of applying u to $\mathcal{HA}_I$, obtaining the system output at the same sampling points of the TSS y, and making a decision on the correctness of $\mathcal{HA}_I$.

Definition 15 (Test Verdict)**.** *For a given hybrid automaton* $\mathcal{HA}$ *and a test-suite* TS*, a test verdict function is a mapping* $(TS, \mathcal{HA}) \to \{Pass, Fail\}$*.*

A class of test verdict functions is defined by Algorithm 1. A test verdict function defined by this algorithm compares the expected outcomes of TS with the solutions $\mathcal{HA}_I$ at the sampling points of TS within the neighborhood of T and E in time and value, respectively.

In a test verdict, *Fail* means that the system does not conform to the specification, while *Pass* means that, using the considered test-suite, no evidence has been found to conclude that the system does not conform to the specification. A test case with a *Pass* verdict is also inconclusive unless it is part of an exhaustive test suite. Thus, we do not include *Inconclusive* in our verdicts. We use Algorithm 1 as the test verdict for conformance testing of cyber-physical systems.

Algorithm 1 Test Verdict Algorithm: Given a test suite, a hybrid system implementation and conformance testing parameters, it determines the implementation passes or fails the test suite.

```
 1: inputs: A test-suite $TS$; A hybrid automaton $\mathcal{HA}_I$; Conformance parameters $T, E$
 2: output: Pass or Fail
 3: for each $(u, y) \in TS$ do
 4:     $y_I \leftarrow out_{\mathcal{HA}_I}(u)$
 5:     $P \leftarrow \mathrm{dom}(y)$
 6:     $y_I^s \leftarrow \pi_P(y_I)$
 7:     for each $(t, j) \in \mathrm{dom}(y_I^s)$ do
 8:         $I_t = [t - T, t + T] \cap \{t \mid \exists j : (t, j) \in \mathrm{dom}(y)\}$
 9:         if $\exists t' \in I_t, \exists i, k \in \mathbb{N}$ s.t. $\|y(t', i) - y_I^s(t, k)\| \leq E$ then
10:             continue;
11:         else
12:             return Fail
13:         end if
14:     end for
15: end for
16: return Pass
```

[3]Note that even in discrete-time systems (e.g. a system equipped with a digital controller) a *hold* circuit is used [15], which leads to a continuous input to the system.

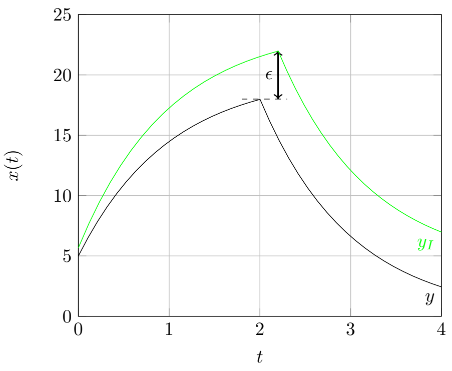


Figure 3: The output trajectory of the thermostat specification (y) and that of a sample implementation (y_I)

Definition 16 (Soundness)**.** *Considering a specification $\mathcal{HA}$, a test-suite TS is sound under a specified test verdict algorithm if the following proposition holds*

$$\forall \mathcal{HA}_I : \left(\mathcal{HA}_I \approx_{(\tau,\epsilon)} \mathcal{HA}\right) \Rightarrow \mathcal{HA}_I \textit{ passes } TS \tag{3}$$

4. Sound Test-Suites

The aim of this section is to establish the conditions under which the soundness of a test-suite can be guaranteed when Algorithm 1 is used as the test verdict.

4.1. Unsoundness result

A straightforward method for (τ, ϵ)-conformance testing is to use Algorithm 1, considering τ and ϵ as the algorithm parameters T and E, respectively. However, the problem with this approach is that, for all practical specifications, it can produce unsound results, irrespective of the sampling period used in the test-suite. The following example illustrates this problem.

Example 3. *Consider the thermostat system described in Section 1.2, with the hybrid automaton given in Figure 2a as its specification. Figure 3 shows an output trajectory obtained from the specification (labeled as y), and an output trajectory of a conforming implementation (labeled as y_I). The trajectory y_I has been obtained by shifting y to the right by 0.2 and to the above by 4. Considering $\tau = 1 > 0.2$ and $\epsilon = 4$, it is seen that y_I satisfies the (τ, ϵ)-closeness condition. Assume that for (τ, ϵ)-conformance testing of the implementation, Algorithm 1 is used with $T = \tau$ and $E = \epsilon$. Also, assume a test-case containing a TSS obtained from y by a sampling function with sampling period of 0.03. If the set of sampling points P is selected such that $t = 2.2 \in P$, then $2 \notin P$*[4]*. However, there is only one point in y, namely at time $t = 2$, which satisfies the closeness condition specified in line 9 of Algorithm 1. But under the described sampling function, this point is not included in the test-case; as a result, the implementation fails.*

The following theorem generalizes the observation made in Example 3 to a large class of specifications (covering all thinkable practical cases). Namely, we define a general class of specifications $\mathbb{S}$ such that for each $\mathcal{HA} \in \mathbb{S}$, there is no sampling period for which Algorithm 1 with parameter selection of $T = \tau$ and $E = \epsilon$ is guaranteed to be sound for all (τ, ϵ)-conforming implementations.

Theorem 1. *For a given $\tau > 0$, consider a class $\mathbb{S}$ of all specifications that exhibit at least one output trajectory, y, which satisfies the following property:*

[4]We omit the jump index from a hybrid time instance for notation brevity.

- *There exists a point s in time when y has a minimum (or maximum) and there exists an interval $I = [s-\tau, s+\tau]$ where there are at most a countable number of minima (maxima) points in I.*

Also, for any $\mathcal{HA} \in \mathbb{S}$, let $\mathbb{TS}_{\mathcal{HA}}$ denote the set of all valid test suits for $\mathcal{HA}$. Then, given $\tau, \epsilon > 0$, the following holds:

$$\forall \mathcal{HA} \in \mathbb{S}, \exists TS \in \mathbb{TS}_{\mathcal{HA}}, \exists \mathcal{HA}_I \in \mathbb{S} \text{ such that}: \ \mathcal{HA}_I \approx_{\tau,\epsilon} \mathcal{HA} \text{ and } \mathcal{HA}_I \text{ fails } TS$$

Proof. We present a proof by construction. The intuition is based on a generalization of Example 3. Consider a test-suite containing a test-case with sampling period p and let s be the point in time in which y has a minimum (or maximum) considered in the hypothesis of the theorem. Furthermore, let M be the set of minima (maxima) also described in the hypothesis. Take a sample implementation which is obtained from y by shifting the values of its variables by ϵ to above (below), and its hybrid time axis by $\delta \leq \tau$ to the right such that

$$\forall (t', j) \in M : \frac{|t' + \delta - s|}{p} \notin \mathbb{N}, \tag{4}$$

As a result, when the sampling points are adjusted such that s is a sampling point, none of the mentioned minima (maxima) are included in the sampled TSS of the implementation output. Hence, the implementation is failed, because only the points in M could satisfy the closeness condition specified in line 9 of Algorithm 1. On the other hand, it can be easily seen that the described implementation conforms to the specification, which leads to the unsoundness of the test-suite. □

4.2. Reinstating Soundness

As mentioned before, the goal of this section is to guarantee the soundness of test-suites. To this end, we define the following measure on the specifications.

Definition 17 (Specification Maximum Change)**.** *Given a specification $\mathcal{HA}$, a periodic test-case (u, y) for it with period $p > 0$, and a test duration T, the maximum change of $\mathcal{HA}$ with respect to (u, y) and T is defined as $\Delta_p = \max_{t \in T} \Delta_p(t)$, where*

$$\Delta_p(t) = \max_{s \in [t-p/2, t+p/2]} y(s) - \min_{s \in [t-p/2, t+p/2]} y(s).$$

The following lemma paves the way for the subsequent soundness result. Namely, it states that if the maximum changes of the specification are confined, then by extending the error margins for output values, one can always obtain sound results.

Lemma 1. *Given $p \geq 0$, a test duration T, a specification $\mathcal{HA}$, and an arbitrary output trajectory $y \in \mathbb{Y}$, we always have*

$$\begin{gathered} \forall \epsilon > 0, \forall c \in \mathbb{R}, \forall s_2 > s_1 \geq 0 : \text{ if } |s_1 - s_2| \leq p \text{ then} \\ (\|y(s_1) - c\| > \epsilon + \Delta_p \text{ and } \|y(s_2) - c\| > \epsilon + \Delta_p) \Rightarrow \\ \forall s \in (s_1, s_2) : \|y(s) - c\| > \epsilon \end{gathered} \tag{5}$$

Proof. The proof is by contradiction. Assume that the left side of (5) holds, but the right side does not hold, namely $\exists s \in (s_1, s_2) : \|y(s) - c\| \leq \epsilon$, or equivalently,

$$-\|y(s) - c\| \geq -\epsilon \tag{6}$$

Summing up (6) with $\|y(s_1) - c\| > \epsilon + \Delta_p$ from (5) yields

$$\|y(s_1) - c\| - \|y(s) - c\| > \Delta_p \tag{7}$$

Besides, using the general rule $\|a - b\| \geq \|a\| - \|b\|$, we can conclude

$$\|y(s_1) - y(s)\| > \Delta_p \tag{8}$$

which implies in a maximum change of y larger than Δ_p in an interval smaller than p. But this is inconsistent with the definition of Δ_p, which contradicts the initial assumption. □

Definition 18 (Robust Test-Suites)**.** *Given $\tau, \epsilon > 0$, assume that we use Algorithm 1 with parameter assignment of $T = \tau$ and $E = \epsilon + \Delta$, where $\Delta > 0$. Then, given a specification $\mathcal{HA}$, a test-case $tc = (u, y) \in \mathbb{TS}$ with a sampling period p is said to be robust if*

$$\Delta \geq \Delta_p, \tag{9}$$

Theorem 2. *A robust test-case is always sound, according to the definition of soundness in Definition 16.*

Proof. The proof is by contradiction. Assume the theorem does not hold. Thus, there exists a system $\mathcal{HA}_I$ conforming to a specification $\mathcal{HA}$ which fails a robust test case $tc = (u, y)$ with respect to a sampling period p. Let y_I be the sampled output trajectory of the system for the input u. Furthermore, let $\phi = out_{\mathcal{HA}}(u)$. Based on the mentioned assumption, Algorithm 1 returns *Fail* for y and y_I. As a result,

$$\begin{aligned} \exists t \in \operatorname{dom}(y_I) \textit{ such that } \forall t' \in [t-\tau, t+\tau] \cap \textit{dom}(y) : \\ \|y_I(t) - y(t)\| > \epsilon + \Delta_p \end{aligned} \tag{10}$$

(see lines 8-13 in Algorithm 1). According to Lemma 1, and based on the definition of Δ_p, (10) yields that

$$\forall t' \in [t-\tau, t+\tau] : \|y_I(t) - \phi(t)\| > \epsilon \tag{11}$$

which means that the implementation is not conforming to the specification. However, this result is in conflict with our assumption that $\mathcal{HA}_I$ conforms to the specification $\mathcal{HA}$, proving the theorem. □

It is worth noting that, in practice, an appropriate value for Δ_p, and thus, for Δ, can be determined based on the formal specification of the system. For instance, the evolution of the system may be specified using a hybrid automaton with a number of differential equations without a discrete jump in the values. In this case, if such equations yield a bounded derivative for the respective functions, then the maximum value of the derivative can be used as a valid value for Δ_p. We elaborate on the practical aspects of finding Δ_p in Sections 5 and 6.

4.3. Towards Exhaustiveness

In order for a test-suite to be exhaustive, it should fail each and every nonconforming implementation.

Definition 19 (Exhaustiveness)**.** *A test-suit TS is said to be exhaustive if*

$$\forall \mathcal{HA}_I : \left(\mathcal{HA}_I \not\approx_{(\tau,\epsilon)} \mathcal{HA}\right) \Rightarrow \mathcal{HA}_I \textit{ fails } TS$$

To this end, an implementation must pass a test suit only if it is conforming. However, the approach discussed up to now has the following two shortcomings to achieve this goal.

First, according to Definition 9, two criteria must be satisfied by an implementation to make sure that it conforms to the specification. However, the test verdict algorithm only checks one of them (lines 7 to 14 in Algorithm 1). In fact, for each sampled point in the implementation, the method checks the possibility of the satisfaction of the closeness condition, and the implementation fails if such possibility is not established. For an exhaustive test, however, we need to check also the other way around. For this goal, a similar information (namely maximum change as in Definition 17) should be available for the implementation trajectory. However, in this manner, the conformance testing cannot be seen as a black box technique any more and some inside information about the implementation will be required.

Second, Definition 9 calls for the conditions over all points in a continuous interval. However, it is not possible to concretely check all the points as there are uncountably many points in a given continuous interval. To resolve this issue, we need to check a *restrictive* condition on the tested points such that we can conclude some result for the other points in their vicinity, including the untested ones. For this goal, our conjecture is that, using Algorithm 1 with a parameter assignment of $T = \tau - p/2$ and $E = \epsilon - \Delta_{p/2}$ provides an exhaustive test verdict method. This is based on the following observation.

Observation 1. *Let y and y_I be two trajectories and assume that $\epsilon > 2 * \Delta_{p/2}$. Then, the following holds for any $t \in \mathrm{dom}(y)$:*

$$\exists t_I \in [t - \tau + \frac{p}{2}, t + \tau - \frac{p}{2}] \; : \; \|y_I(t_I) - y(t)\| \leq \epsilon - 2 * \Delta_{p/2} \;\Rightarrow\; \forall t' \in [t - \frac{p}{2}, t + \frac{p}{2}] : \|y(t') - y_I(t_I)\| \leq \epsilon$$

This observation states that if the two trajectories are sufficiently close to each other in two points, then they would be so for the other neighboring points (provided that the variation of the dynamics in the implementation is bounded in the same manner as the specification). This is motivated by the fact that on the one hand, any two points in the interval can vary at most up to $\Delta_{p/2}$ from the measured points. On the other hand, we have found two points that vary from each other up to $\epsilon - 2 * \Delta_{p/2}$. Hence, it follows from the last two statements that the total distance of any two points will be bounded by $\epsilon - 2 * \Delta_{p/2} + \Delta_{p/2} + \Delta_{p/2}$, i.e., ϵ.

To summarize, once the two sets of criteria are met, we require two runs of conformance testing (possibly run simultaneously) with two different bounds in the two respective directions to guarantee soundness and completeness.

5. Restoring Soundness via Reachability Analysis

Reachability analysis is a technique to explore the state space of a system in order to determine whether a particular set of states is reachable from a given set of initial states. It is often used to overcome the complexity of model-checking safety properties by calculating an over-approximation of the reachable set of states (through a less computationally intensive algorithm); then once this over-approximation satisfies the safety property, the actual system is also proven safe. In case a spurious counter-example is detected, the overapproximation is refined to exclude such a counter-example and the procedure is repeated until either safety is proven or a real counter-example is found. This approach (i.e., combining reachability with counter-example guided abstraction refinement) has proven useful in model-checking hybrid systems were basic problems are often intractable [16].

In our context, we use reachability analysis to provide an overapproximation of the changes in the system dynamics around the sampling points. This will in turn provide us with an overapproximation for the value of Δ_p used in our conformance testing algorithm. To this end, we use the reachability analysis algorithm and tool developed by Matthias Althoff [17].

In Section 5.1, we present an overview of the work regarding reachable sets for hybrid systems [17, 18]. In Section 5.2, we present the definition of zonotopes, which are the geometric forms we use to represent reachable sets. Afterwards, in Section 5.3, we show how we use reachability analysis to obtain sound test cases, taking advantage of the results presented in Section 4.

5.1. Reachable sets for hybrid systems

Reachable sets for hybrid systems are defined to capture both the discrete ($\mathcal{R}^z(t)$) and the continuous ($\mathcal{R}^e(t)$) behaviour for a given absolute point in time t [17]. Concerning the discrete behaviour, $\mathcal{R}^z(t)$, the calculation of its reachable sets is required only for the discrete jumps. This can be achieved by verifying which guard sets are reached by the continuous dynamics. This is done by computing possible intersections between the reachable sets that represent the trajectories bounds and the guard sets. Since discrete reachable sets yield the reachable locations of the associated hybrid automaton, and in testing we can typically observe only the valuations of the continuous variables, we restrict our focus to continuous reachable sets.

Regarding the continuous dynamics, the computation of the reachable set is performed considering how the system variables evolve with time. We have adapted the formal definition of the continuous reachable sets [17], in order to be used in our context, which involves a slight variation on the formal definition of hybrid automata. For instance, instead of using just the absolute time point t, we generalize for a super-dense time domain (with jumps).

Definition 20 (Exact Continuous Reachable Set for Hybrid Automata)**.** *The continuous reachable set* $\mathcal{R}^e$ *of a hybrid automaton* $\mathcal{HA}$ *at a time* t *is defined as:*

$$\mathcal{R}^e(t) = \Big\{ \phi(t, j) \downarrow V_O \mid \phi \in \mathit{Trajs}(\mathcal{HA}) \wedge j \in \mathbb{N} \Big\}$$

Each element in the above set is a valuation (at time t after j jumps) restricted to the output variables. The set contains all such elements, for all trajectories of the automaton $\mathcal{HA}$ jumps. Figure 4 illustrates the temperature trajectory of the thermostat example (see Figure 2) starting from multiple initial states. It depicts the exact continuous reachable set at a time t, namely, $\mathcal{R}^e(t)$.

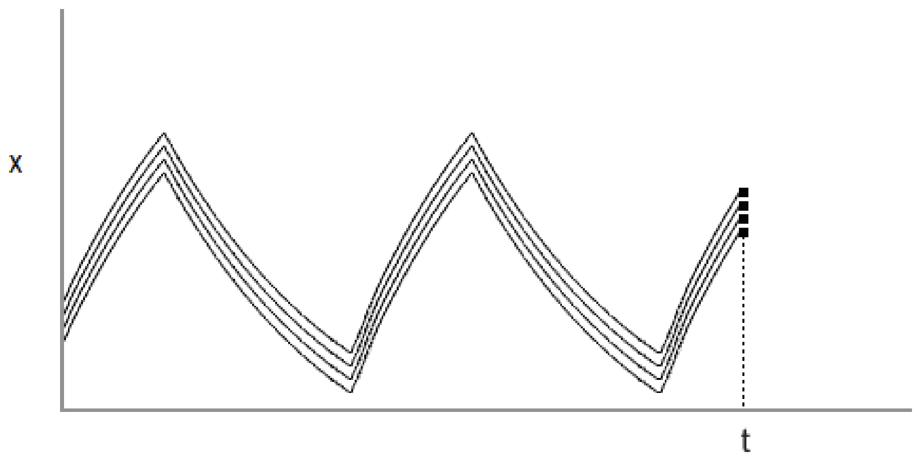


Figure 4: Thermostat - Trajectories for Multiple Initial States

In practice, however, exact reachable sets are only computable for some restricted forms of dynamical systems [16]. Therefore, over-approximations of the exact reachable sets, represented by $\mathcal{R}(t) \supseteq \mathcal{R}^e(t)$, are used. The details of the computation of $\mathcal{R}(t)$ is out of the scope of this paper; we use the approach proposed by Althoff [17, 18] and refer to it for more details. Moreover, the definition of $\mathcal{R}(t)$ can be lifted to time intervals: $\mathcal{R}([0, t])$.

Definition 21 (Reachable Set of a Time Interval [17])**.** *The reachable set of a time interval is the union of reachable sets at points in time within the interval* $t' \in [0, t]$*:*

$$\mathcal{R}([0, t]) = \bigcup_{t' \in [0,t]} \mathcal{R}(t').$$

Figure 5 shows the computable over-approximative reachable set of a time interval. In this case, it represents the reachable set which is a result of $\mathcal{R}([0, t])$. The central lines are the system behaviour and the surrounding area corresponds to the reachable sets.

5.2. Reachable set representation

There are several ways to represent reachable sets; ellipsoids [19], orthogonal polyhedras [20] and oriented rectangular hulls [21] have been tried in the past. In this work, reachable sets are represented in the form of zonotopes, which consist of a series of points that characterize geometric forms. Zonotopes are a compact and efficient way to portray high-dimensional sets due to their efficient computability [18].

However, using zonotopes has a downside when it comes to calculating intersections with other geometric forms (e.g., guard sets). Some techniques for over-approximating zonotopes [22] and [23] cope with time intervals and intersections.

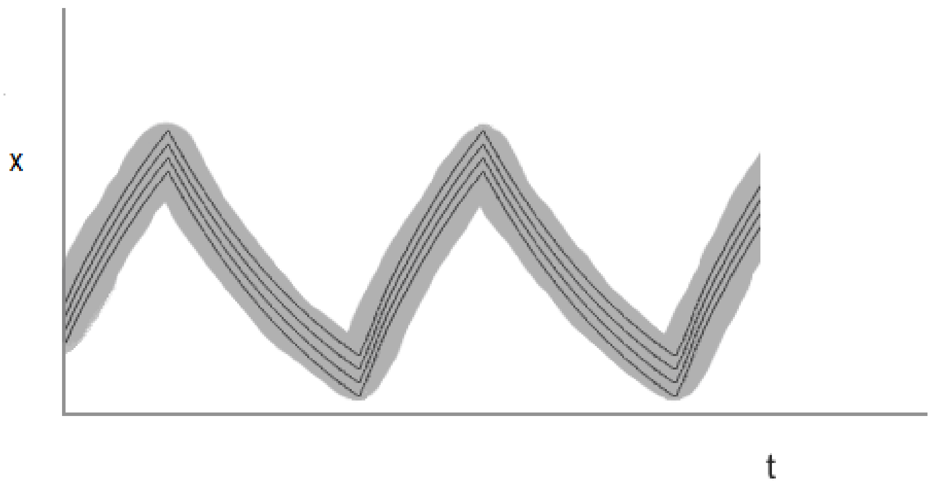


Figure 5: Thermostat - Overapproximation of Reachable Set from 0 to t

The generator representation ($\mathcal{G}$-representation) of a zonotope is defined by a center, c, and a set of points, i.e., the generators:

Definition 22 ($\mathcal{G}$-representation of a zonotope)**.** *A zonotope is a set*

$$\mathcal{Z} = \Big\{ z \in \mathbb{R}^n | z = c + \sum_{i=1}^{e} \beta_i \cdot g^{(i)}, -1 \leq \beta_i \leq 1 \Big\},$$

with $c, g^{(1)}, ..., g^{(e)} \in \mathbb{R}^n$*.*

This definition can be interpreted as the Minkowski sum of a finite set of line segments, where $\hat{l}_i = [-1, 1] \cdot g^{(i)}$. The Minkowski sum is a geometric operation used to add sets of points that represent shapes. The general formula is given by $A \oplus B = \{a + b \mid a \in A, b \in B\}$. For instance, Mikowski sum of two triangles $A = \{(1, 0), (0, 1), (0, -1)\}$ and $B = \{(0, 0), (1, 1), (1, -1)\}$ would result in shape similar to an hexagon represented by the set of points $\{(1, 0), (2, 1), (2, -1), (0, 1), (1, 2), (0, -1), (1, -2)\}$.

Figure 6 (taken from [17]) illustrates the iterative construction of a zonotope by adding generators, which results in better defined forms at each step.

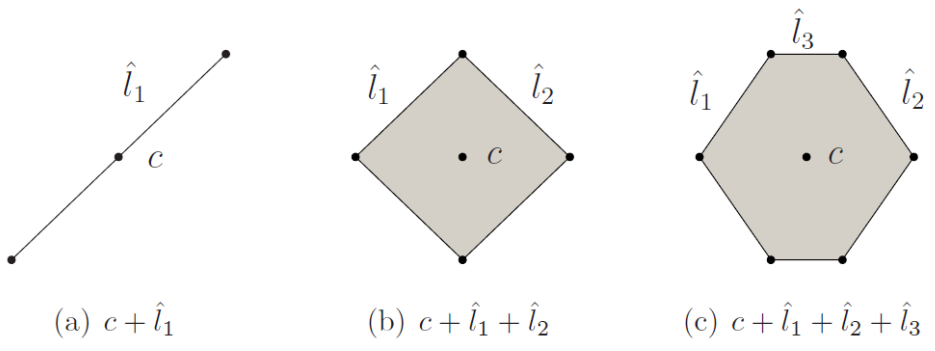


Figure 6: Iterative construction of a zonotope

Furthermore, Figure 7 illustrates the zonotopes representation of those reachable sets of the thermostat example. The geometric forms enclosing the central line are the computed zonotopes.

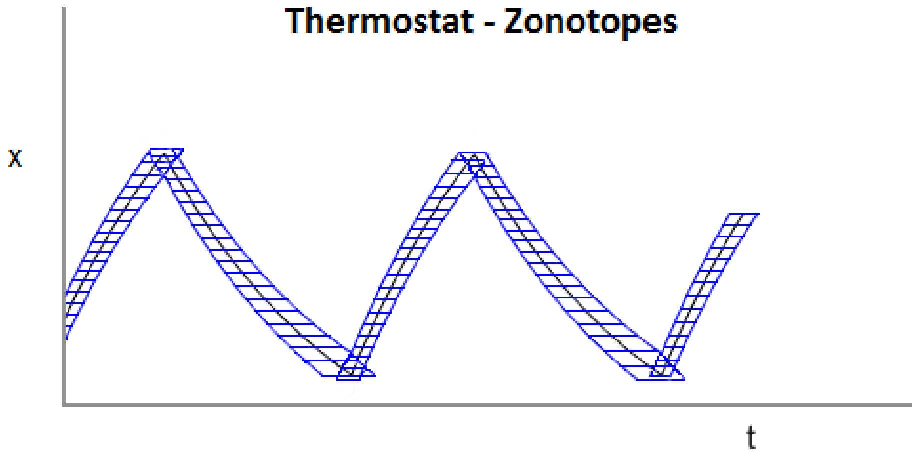


Figure 7: Thermostat - Zonotopes

5.3. Computing the specification maximum change via reachability analysis

In this section, we explain how to use reachable sets to compute the specification maximum change described in Definition 17. Our approach consists of calculating the reachable sets for time intervals $[t - p/2, t + p/2]$, where p stands for the period of a periodic test case (see Definition 17), considering every point in the set of sampling points.

To restore the soundness of our testing strategy, we need to compute the maximum value deviation within the obtained reachable sets. In order to compute this maximum value, we find the extreme points of each reachable set and calculate their difference. Once we have computed the maximum changes of the specification, we can obtain sound results (see Lemma 1).

We define the specification maximum change in terms of its reachable sets.

Definition 23 (Specification Maximum Change based on Reachable Sets)**.** *Given a specification $\mathcal{HA}$, a periodic test-case (u, y) for it with period $p > 0$, and a test duration T, the maximum change of $\mathcal{HA}$ with respect to (u, y) and T is defined as $\Delta_p^{RS} = \max_{t \in T} \Delta_p^{RS}(t)$, where*

$$\Delta_p^{RS}(t) = \max(\mathcal{R}([t - p/2, t + p/2])) - \min(\mathcal{R}([t - p/2, t + p/2])). \tag{12}$$

Considering the results represent an over-approximation of the trajectory boundaries, we can assume that the specification maximum change will be contained within the zonotopes. It is safe to infer that the distance between the extreme points of a zonotope will be at least equal to the distance between the extreme points in the specification trajectory itself. Therefore, we conclude that $\Delta_p^{RS} \geq \Delta_p$ for all hybrid automata. Thus, if we consider Δ_p^{RS}, computed via reachability analysis, we also have a sound conformance testing strategy.

Algorithm 2 gives an overview of the Δ_p^{RS} computation process. The specification model serves as input, which is fed into an external tool (see Section 6) where the zonotope computation is handled (line 3). Then, we iterate over the results in order to determine the difference between the extreme points in each zonotope (lines 4-6). The resulting Δ_p^{RS} yields from the zonotope with the greatest vertical amplitude (lines 8-15).

Algorithm 2 Δ_p^{RS} Computation

```
1: inputs: A hybrid automaton $\mathcal{HA}_I$, a period $p$;
2: output: $\Delta_p^{RS}$
3: $Zonotopes$ = computeReachableSets($\mathcal{HA}_I, p$)
4: for each $z_t \in Zonotopes$ do
5:     $\Delta_p^{RS}(t) = max(z_t) - min(z_t)$
6: end for
7: $\Delta_p^{RS} = 0$
8: $t = 0$
9: while $t \leq T$ do
10:     if $\Delta_p^{RS}(t) > \Delta_p^{RS}$ then
11:         $\Delta_p^{RS} = \Delta_p^{RS}(t)$
12:     end if
13:     $t = t + p$
14: end while
15: return $\Delta_p^{RS}$
```

6. Tool Support

In this section, we describe a prototype implementation of the ideas presented in this paper. The prototype is based on the tool we developed for conformance testing of hybrid systems [24]. We refer the reader to the tool paper [24] for the basic implementation of the prototype tool and, in this section, we focus mostly on the novel aspects concerning the soundness of conformance testing. Namely, in addition to test case generation and conformance analysis, already implemented in [24], the tool now interacts with a reachability analysis tool called CORA [12] to calculate sound conformance analysis margins. CORA (COntinuous Reachability Analizer) [12] is a Matlab toolbox for reachability analysis of continuous and hybrid systems. It is used in our tool to compute the specification maximum change, considering the given sampling rate. Moreover, we also show some examples where we explore the tool usage regarding different scenarios. The tool prototype and the models for the examples and the case study are available from the following URL: `http://ceres.hh.se/mediawiki/SCP_TASE_2016`.

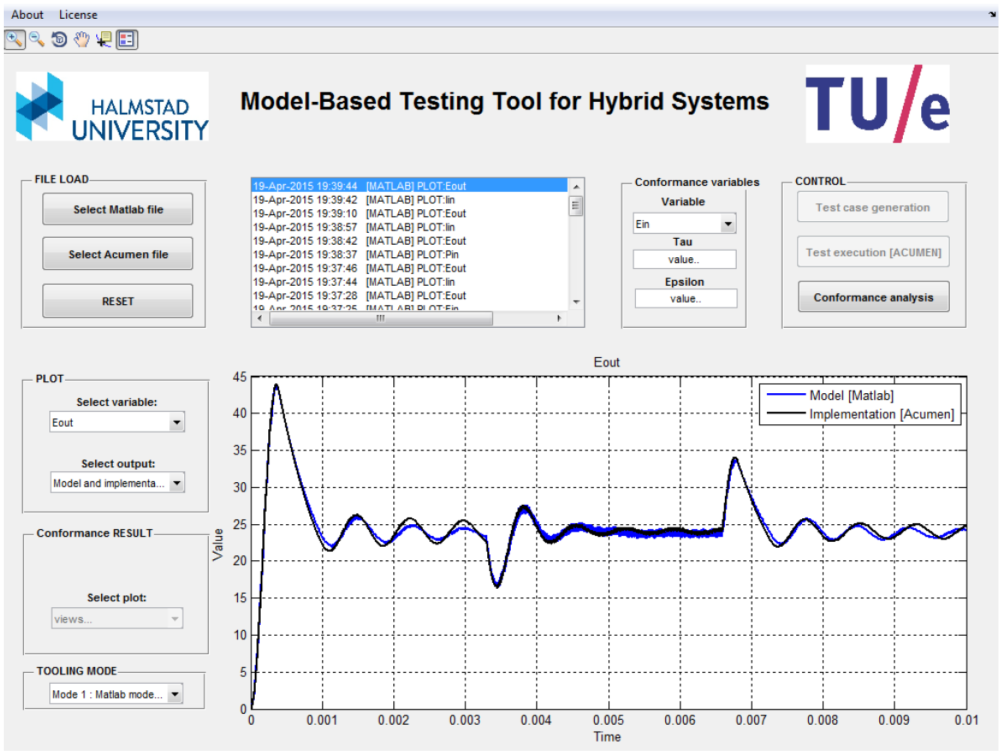
6.1. Generating Sound Test Cases

The overall goal of the tool is to perform conformance testing in three stages, namely, test-case generation, test-case execution, and conformance analysis. To this end, we sample the specification using the given input signal with the given sampling rate, calculate the appropriate error margins (for sound analysis) and, subsequently, execute the test cases on the system under test and, finally, analyzing the results with respect to the given conformance bounds and the calculated error margin. The starting point are, hence, models representing the specification and the system under test.

The application of test-case generation and execution methods results in generating input-output data for both the model and the implementation under test. The application of the conformance analysis is based on the notion described in Definition 10. This results in a conformance verdict, possibly accompanied with a counter-example for conformance violation, which is fed into the GUI (see Figure 8).

6.2. Integration with CORA

In order to apply the strategy described in Section 5, our tool executes CORA in background to perform reachability analyses. The seamless integration of the tools is made possible because both are implemented in the Matlab environment. However, the specification model format for our tool and and the one for CORA are different, so an additional CORA-readable model is required for the reachability analysis. Currently, there is no method for direct translation. Moreover, it is also expected that the name of variables and the considered sampling period in both models are the same.



```
19-Apr-2015 19:39:44  [MATLAB] PLOT:Eout
19-Apr-2015 19:39:42  [MATLAB] PLOT:Iin
19-Apr-2015 19:39:10  [MATLAB] PLOT:Eout
19-Apr-2015 19:38:57  [MATLAB] PLOT:Iin
19-Apr-2015 19:38:42  [MATLAB] PLOT:Eout
19-Apr-2015 19:38:37  [MATLAB] PLOT:Pin
19-Apr-2015 19:37:46  [MATLAB] PLOT:Eout
19-Apr-2015 19:37:44  [MATLAB] PLOT:Iin
19-Apr-2015 19:37:28  [MATLAB] PLOT:Eout
19-Apr-2015 19:37:25  [MATLAB] PLOT:Ein
```

Figure 8: Graphical User Interface of the tool

Once given an input model, CORA calculates the trajectory bounds for the respective model. These bounds are meant to delineate over-approximations of the reachable sets of the trajectories. CORA uses zonotopes for reachable sets representation. In summary, CORA calculates and returns to our tool the zonotopes obtained from the input model.

Zonotope manipulation is done via the Matlab toolbox named Multi-Parametic Toolbox (MPT) [25]. Using this tooblox, we find the extreme points for each zonotope, i.e., the minimum and maximum values contained in the reachable sets. These values are used to find the maximum variation possible within the trajectory, as shown in Definition 23. Each reachable set account for the region delimited by the time constraints $t - p/2$ and $t + p/2$, where each t belongs to the set of sampling points.

Once the specification maximum change is calculated, its value is displayed to the user, and it is automatically added to the value of ϵ for the verdict algorithm (see Algorithm 1). The user can then determine whether the calculated margin is tight enough to yield practically meaningful conformance results or increase the sampling rate and reiterate to obtain tighter margins. We illustrate these aspects by means of a number of examples.

6.3. Thermostat Example: Revisited

In the remainder of this section, we present three cases based on our running example. Using these cases, we demonstrate different scenarios of conformance testing using our tool as well as their interaction with the implemented steps for guaranteeing soundness.

We first explain the main scenario in using the tool. Test case generation is performed on the specification presented in Example 1 by simulating the specification model with a fixed sampling rate and a fixed input trajectory. For the sake of simplicity, the inputs ($u(t)$ and $i(t)$ – see Figure 2) are constant signals whose values are selected at the start of the simulation. In other words, the input values never change once selected. Additionally, all models described in this section use the same set of input values, namely $u(t) = 20.0$ and $i(t) = 0.0$.

Each case presented in the following subsections reflects an aspect of the tool application. Implementation 1 represents the best case scenario, where the implementation passes the test with the strict error margins τ and ϵ, without any adjustment calculated using the maximum changes in the specification. However, this scenario is not necessarily the usual case. We, hence, present Implementation 2, where the test cases fail, but the failure is not due to the non-conforming implementation, but rather unsoundness of the test cases with respect to the chosen sampling rate and conformance bounds. Hence, the bounds are corrected automatically in the tool by the reachability analysis. Finally, Implementation 3 represents the case where even with the adjusted sound bounds the implementation is found to be non-conforming and, hence, the bounds have to be relaxed by the user. This approach can provide the user with a quantitative degree of how incorrect the implementation is.[5]

6.3.1. Case 1 – strictly sound conformance results

The first case concerns an implementation that is a modified version of the specification (see Example 1) in such a way that its temperature trajectory is shifted 3 degrees below and 0.1 seconds to the left compared to the specification's behaviour. Thus, the implementation is (τ,ϵ)-conforming with respect to the specification, once we take $\tau = 0.1$ and $\epsilon = 3$. Both trajectories are shown in Figure 9.

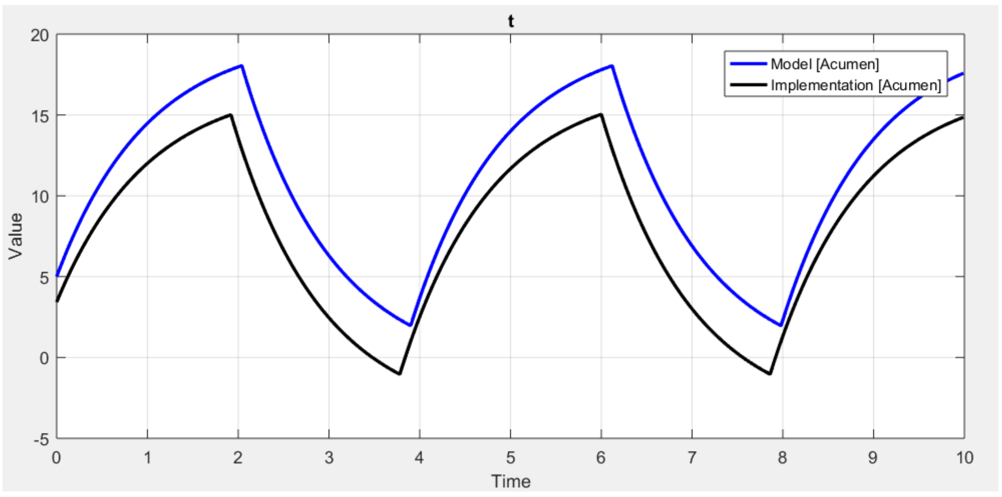


Figure 9: Temperature trajectory

For a fixed implementation, it is possible to find an adequate sampling rate that leads only to sound test cases. For this particular example, we have considered a sampling rate of 0.01 seconds, resulting in 1000 test vectors. This example represents the best case scenario, where, due to an appropriate choice of sampling rate, only sound test vectors are generated.

Table 1 displays some of the test vectors we have obtained from the specification, whereas Table 2 displays the temperature values for the corresponding implementation timestamps.

The difference in output values between the specification and the implementation for all sample points are within the conformance bounds ($\tau = 0.1$ and $\epsilon = 3.0$) and therefore the implementation passes the test case.

6.3.2. Case 2 – unsound test cases

As shown in Section 4, taking the strict bounds of τ and ϵ may result in unsound conformance verdicts. Considering the thermostat example, assume that we use the higher sampling rate of 0.03 for the same implementation. At 2.04 seconds of simulation time, the temperature in the specification trajectory reaches

[5]All the models used in this work are available in `http://www.cin.ufpe.br/~hlsa/cps/conftest/`

$Trajs(V_I)$		$\mathbb{TSS}(V_O)$	
$u(t)$	$i(t)$	t	$x(t)$
20	0	1.00	14.48
20	0	4.00	4.18
20	0	7.00	6.49
20	0	10.00	17.74

Table 1: Test vectors generated from the specification

$Trajs(V_I)$		$\mathbb{TSS}(V_O)$	
$u(t)$	$i(t)$	t	$x(t)$
20	0	0.90	11.48
20	0	3.90	1.18
20	0	6.90	3.49
20	0	9.90	14.74

Table 2: Values obtained from the implementation

the value of 18.049 degrees, which triggers a discrete change and the temperature starts to fall. In a similar way, the implementation trajectory reaches 15.049 degrees at 1.94 seconds and should also start to decrease. This can be seen in Figure 9. These values are the expected ones and are still within the conformance bounds.

However, since we set the sampling period to 0.03, it means that 2.04 belongs to the set of sampled points, whereas 1.94 does not. At exactly 2.04 seconds, the closest sampling point to 1.94 that respects the 0.1 seconds margin (value of τ) is 1.95. At this moment, the temperature in the implementation has already started to decrease and has a value of 14.448 degrees, which makes it non-conforming because this value is outside the 3 degrees margin (value of ϵ).

Note that we demonstrated that the implementation under test is indeed $(0.1, 3)$-conforming to the specification; hence, the verdict is clearly unsound. Figure 10 depicts this unsound non-conforming example.

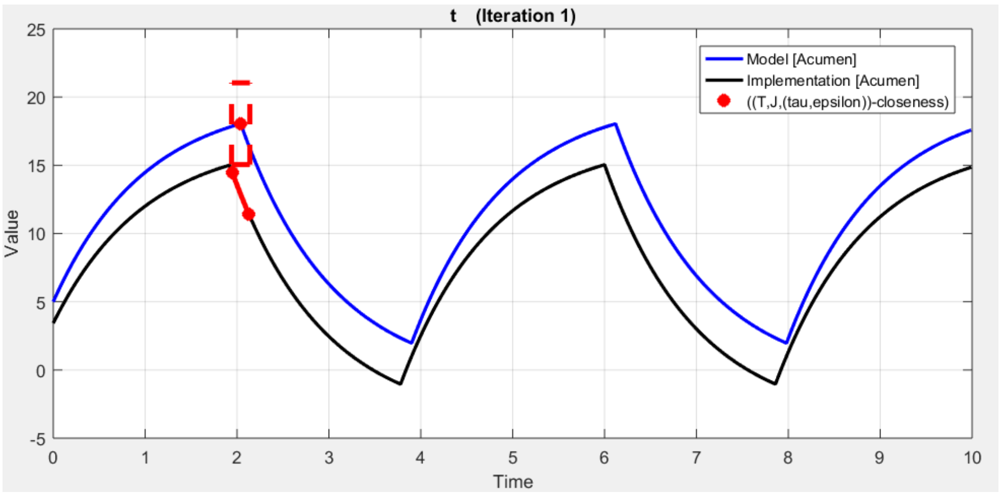


Figure 10: Temperature trajectory

The issue arises because information is always lost whenever a continuous signal is sampled. For example, it is possible for the value of the signal to abruptly increase and decrease between two sampling points, causing the sampling function not to capture this behaviour. Our strategy consists of increasing the error margin to take into consideration these sharp variations.

For that, we have built a similar model mirroring the specification's behaviour that serves as input to CORA. The obtained results using CORA are over-approximation bounds, in the form of geometric objects (zonotopes), for each sample of the trajectory, as depicted in Figure 7. The next step is to find the extremities of these geometric objects, as they represent the maximum possible trajectory variation within each sample. By repeating this process for all samples, we can find the specification maximum change. (This process can also be done locally for each group of sampling points and their corresponding conformance analysis; we leave both the theoretical framework and the practical implementation of this idea for future work.) For the thermostat example, this process yields 0.984 degrees as a result. This process is automatic and the result is displayed in the tool's interface.

This value represents the maximum possible theoretical deviation between samples. As a consequence, for this example, by extending the error margin, ϵ, of 0.984, we can always obtain sound results (see Lemma 1). Therefore, setting the error margin value to 3.984 instead of 3, results in a **pass** verdict. This occurs because 3.984 is greater than the difference between the temperature value in the specification and implementation samples, 18.049 and 14.448 respectively, yielding a sound verdict.

6.3.3. Case 3 - adjusting τ and ϵ to achieve conformance

For a non-conforming implementation, even if one extends the margins with Δ_p, the implementation will fail the conformance testing. In such a case, one might want to adjust the conformance parameters τ and ϵ in order to find the degree of non-conformance between the implementation and the specification.

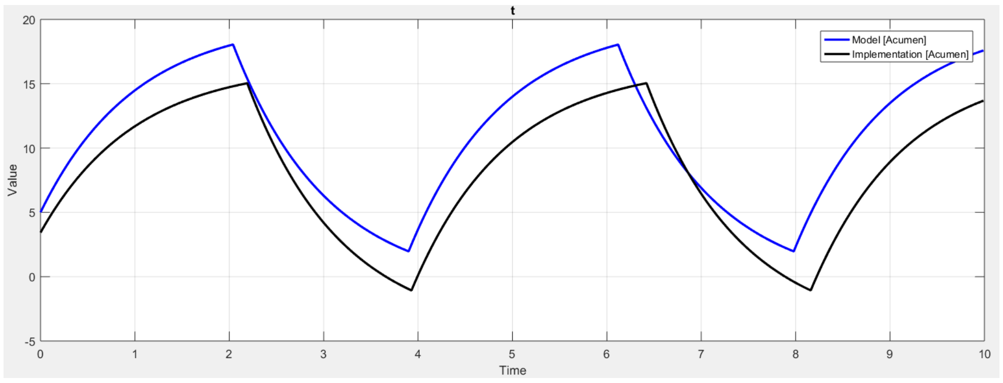


Figure 11: Unconforming Thermostat Implementation

Consider the faulty implementation depicted in Figure 11 that leads to a trajectory which, at some points, does not respect the intended conformance bounds of 3 value units and 0.1 time units. Even after adjusting the bounds with Δ_p, the specification and the implementation are found to be non-conforming; then, it is possible to adjust the parameters gradually until this condition is satisfied. This determines the degree of (non-)conformance of the current implementation with respect to the specification.

For the implementation depicted in Figure 11, if we fix $\tau = 0.1$, the minimum value for ϵ that makes the implementation conforming with respect to the implementation is 3.602[6]. Similarly, with a fixed $\epsilon = 3$, the minimum value for τ is 1.5. These margins might not be practically admissible and, hence, the implementation may be rejected altogether.

However, it is worth noting that, depending on the dynamics and the chosen fixed value for ϵ, it is possible that there is no value for τ that results in a conforming verdict. For instance, considering the thermostat implementation described in Section 6.3.1, if one fixes $\epsilon = 2.5$, there are certain points in the specification signal (e.g., the peaks of the signal) that will never be conforming, regardless of τ.

7. Case study: an automotive air-fuel ratio control system

This section presents a validation of our strategy considering a more complex case study. We present a model of a controller for an automotive air-fuel ratio (AFR) control system [26]. The AFR is an important measure in an internal combustion engine, which directly affects its performance and pollution.

In its formal model [13], the full system is comprised of two subsystems, namely, the plant and the controller. The plant captures and manipulates the physical aspects of certain components of the engine, such as the throttle and the intake manifold. It is a complex system that relies on time constraints and has a high degree of non-linear dynamics, which makes the application of formal techniques without significant

[6]This result was found with the tool's assistance, by iteratively increasing the parameter and checking conformance

simplifications non-trivial. The controller, however, can be viewed as a multi-state hybrid system that considers continuous inputs and outputs and switches the operating mode as required.

In this section, we focus on modeling and analyzing the controller's behavior. In what follows, we present a hybrid automaton representation of the controller and then we consider all steps of the proposed strategy for sound conformance verification discussed in the previous sections.

7.1. *Hybrid automaton representation*

The controller receives 5 inputs from the outside components: the inlet air mass flow rate measurement ($\dot{m}_{af}$), the air-fuel ratio (λ_m), the throttle angle (θ), the engine speed (ω) and the sensor failure event ($fail\_event$). It outputs only the commanded fuel (F_c), which serves as input to the plant. It also makes use of internal variables, which are the rate of air mass pumped into the cylinder (p_e) and the integrator state for the PI controller (i). Its dynamics also considers several coefficients that are detailed in its formal model [13].

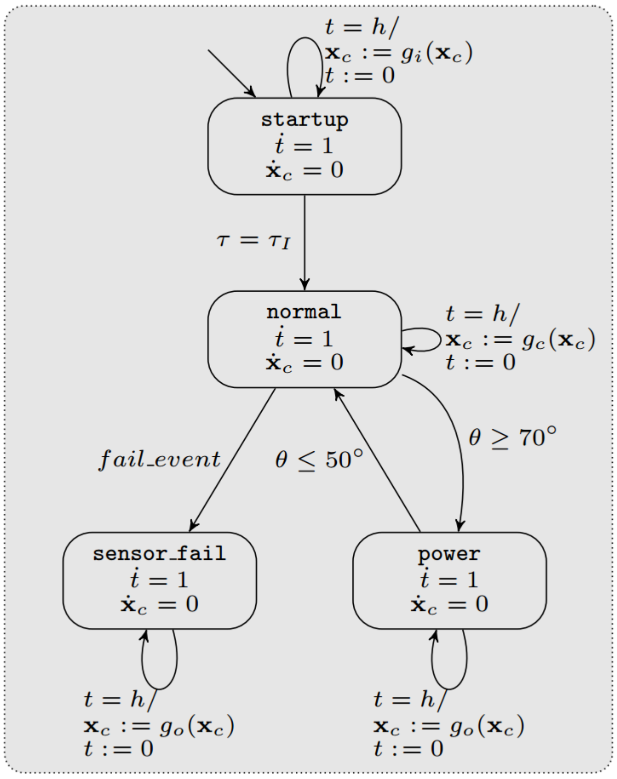


Figure 12: Hybrid automata representation of the controller [13]

A visual representation of the controller hybrid automata is shown in Figure 12 (taken from [13]). We refer to [13] for the full equations regarding the continuous dynamics. The explanation for the discrete states of the controller are as follows:

- Startup mode. The controller operates in this mode whilst the engine is below a certain temperature threshold. In our model, we simulate this behaviour by using a timer. Whenever the timer condition is met, the controller switches to the normal mode.
- Normal mode. The controller remains in this mode whenever all components and variables are operating within the expected bounds.

- Power enrichment mode. The controller enters this mode whenever the engine has to operate in a high powered mode. The conditions to switch from and to this mode are based on the throttle angle (θ) being below or above a certain limit.

- Sensor fail mode. This mode represents the situation when a problem is detected in the system and a failure signal is sent to the controller. Once this situation occurs, the system cannot recover, which means that the controller will remain indefinitely in this mode.

As shown in Figure 12, the controller remains in the startup mode up to $t1$ seconds, when it switches to the normal mode. The transition to the power enrichment mode is triggered whenever the throttle angle input (θ) is greater than 70°, and it remains there until θ drops below 50°. The controller can also enter and shall remain in the failed mode if a failure event is detected.

7.2. Conformance analysis

In order to simulate the hybrid automaton model, it is necessary to model how the system inputs change over time. To mimic the system execution in a normal environment, we have simulated it using dynamic inputs. With the exception of the $fail\_event$, the other inputs change over time while respecting their constraints. Similar to the thermostat example (Section 6), we have also created modified implementations by shift the output trajectory F_c. However, in this case study, we have only considered time-shifts (τ). Thus, the implementation has a time-shift of 0.5 seconds with respect to the specification model. The period used in the simulation was 0.03.

Considering $\tau = 0.5$ and $\epsilon = 0$, a sound conformance analysis should lead to a pass verdict. However, this is not the case and there are a few counter-examples as it can be seen in Tables 3 and 4. Despite using the same input trajectories, the output value in the implementation is different from the expected one, given by the specification. Since the implementation only had a time-shift, the values should remain the same 0.5 time units later, which did not happen. As we explained in Example 3, this discrepancy occurs because the values obtained from the specification do not belong to the sampling points taken from the implementation.

$Trajs(V_I)$					$\mathbb{TSS}(V_O)$	
$\dot{m}_{af}(t)$	$\lambda_m(t)$	θ	ω	$fail\_event$	t	$F_c(t)$
0.32	14.7	35	2577	0	0.32	9.86
0.67	14.7	64	2853	0	1.14	11.73
0.85	12.5	75	3245	0	4.57	15.68

Table 3: Test vectors generated from the specification

$Trajs(V_I)$					$\mathbb{TSS}(V_O)$	
$\dot{m}_{af}(t)$	$\lambda_m(t)$	θ	ω	$fail\_event$	t	$F_c(t)$
0.32	14.7	35	2577	0	0.82	9.79
0.67	14.7	64	2853	0	1.64	11.64
0.85	12.5	75	3245	0	5.07	15.55

Table 4: Values obtained from the implementation

As explained in details in Section 5, we computed Δ_p^{RS} using reachability analysis in order to restore the soundness of our conformance analysis. For this particular example, the resulting value is 0.153, and adding Δ_p^{RS} to ϵ, before the conformance check, resulted in a **pass** verdict, as expected.

As for performance metrics, the computational time of Δ_p^{RS} is directly related to the sampling rate of the input model. As expected, the smaller the sampling rate, the slower it takes to yield a result. However, the time seems to grow linearly, even when using a more complex example. Table 5 displays the average amount of time, in seconds, that the tool takes to compute Δ_p^{RS} for both the thermostat and the fuel controller.

Thermostat		Fuel Controller	
Period	Time	Period	Time
0.5	2.3	0.5	4.3
0.1	3.9	0.1	6.8
0.03	5.8	0.03	8.4
0.01	7.1	0.01	10.2

Table 5: Computational time of Δ_p^{RS}

As a concluding remark, we note how the delta computation does not require any further step, apart from having the appropriate model. Its is a way to account for the lossy discretization process and to obtain sound conformance results.

8. Related work

Conformance testing for cyber-physical systems has been approached from different perspectives. We refer to our earlier work [5, 27] for comparisons of some of these approaches. Inspired by the notion of input/output conformance relation (ioco) [28], which is used for discrete-event systems, van Osch [29, 6] proposed a notion of hybrid input output conformance (hioco). He used the hybrid labeled transition systems [30] formalism in order to specify the hioco relation. Intuitively, a system under test is conforming to a specification based on hioco if all possible behaviors of the implementation is a subset of behaviors allowed by the specification.

From a control-theoretic perspective, Abbas et al. proposed a more practical notion of conformance relation for cyber-physical systems [10, 2], which is used in this paper. Their conformance relation is defined based on the notion of time and value closeness. Their conformance relation allows for the implementation to deviate from the specification behavior to some extent, while it is still regarded as a conforming implementation. In an earlier work[31], we took the first step towards unifying the approach of Abbas et al. with that of van Osch by interpreting discrete actions in the framework of Abbas.

We are not aware of any study devoted to soundness (or robustness) of test suites for the above-mentioned approximate conformance relations. In a slightly different context, Fainekos and Pappas [32] studied the robust sampling of a continuous trajectory (called signal in that context). For this purpose, a notion of *robustness estimate* is defined for a given signal [32]. In summary, for a given Metric Interval Temporal Logic (MITL) formula and a given point in the signal, the robustness estimate specifies how deeply the formula is satisfied by that point. For example, consider the simple formula $t < 10$; then, the robustness of a signal point with value 7 is 3. This is done using the notions of *depth* and *distance*, already defined as the distance between a point and the boundary of a given set. Our soundness criteria bear close similarly to their notion of robustness for MITL. We not only require bounds on the depth of conformance at the sampling points, but also require them in their vicinity (by assuming an upper bound on the signal variation).

Different notions of robustness have been proposed within the control theory, e.g., the notion of input to state stability [33]; these notions have been recently considered in the context of cyber-physical systems [34]. We would like to study these notions of robustness and investigate their possible application to our setting for generating sound test cases.

9. Conclusions

In this paper, we have defined a straightforward notion of conformance testing for cyber-physical systems. This notion is supposed to capture the conformance relation proposed by Abbas and Fainekos. To this end, we have formulated a soundness requirement on test suites and we have shown that test suites are not generally sound. To remedy this, we defined some criteria on the test suite (relative to the sampling points and the system dynamics around those sampling points) and have proven that they indeed guarantee soundness. These criteria are practically and efficiently computed via reachability analysis, and we have shown how our approach can be used in practice considering a combustion engine controller. We have also explored some preliminary ideas regarding exhaustiveness. We implemented the soundness checks using the reachability analysis tool CORA and extended our prototype for conformance testing with the interface to reachability analysis.

The interface with CORA requires a separate model for reachability analysis. This model is almost identical to the model used for test-case generation (modulo syntactic changes) and we would like to automate the transformation of a single model to both perform test-case generation and reachability analysis. Providing a notion of coverage and providing minimal (Pareto-optimal) bounds with guaranteed coverage of the specification are among other areas of our future work. The soundness criteria are currently defined globally over the whole trajectory of system dynamics; however, these bounds can be calculated piece-wise in order to obtain local soundness bound and/or variable sampling rates, as it is common in control theory and simulation of hybrid systems. Adapting our theory and its implementation to this piece-wise calculation is another avenue for our future work.

Acknowledgment

The comments of the anonymous reviewers of TASE 2016 and the Science of Computer Programming Journal are gratefully acknowledged. We also gratefully acknowledge the help and support provided by Matthias Althoff and Hendrik Roehm in using the CORA toolset.

The work of M. R. Mousavi has been partially supported by the Swedish Research Council (Vetenskapsradet) award number: 621-2014-5057 (Effective Model-Based Testing of Concurrent Systems), the Swedish Knowledge Foundation (Stiftelsen for Kunskaps- och Kompetensutveckling) in the context of the AUTO-CAAS HoG project (number: 20140312), and the ELLIIT Strategic Research Area.

The work of Hugo Araujo, Gustavo Carvalho and Augusto Sampaio was partially supported by the CIn-UFPE/Motorola cooperation project, as well as CNPq grants 303022/2012-4 and 132332/2015-9.

References

[1] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner, Model-Based Testing of Reactive Systems, Vol. 3472 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005.

[2] H. Abbas, B. Hoxha, G. E. Fainekos, J. V. Deshmukh, J. Kapinski, K. Ueda, WiP abstract: Conformance testing as falsification for cyber-physical systems, in: Proceedings of the ACM/IEEE 5th International Conference on Cyber-Physical Systems (ICCPS 2014), IEEE CS, 2014, p. 211, available online: http://arxiv.org/abs/1401.5200.

[3] H. Abbas, H. Mittelmann, G. E. Fainekos, Formal property verification in a conformance testing framework, in: 12th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE 2014), IEEE, 2014, pp. 155–164.

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

[5] N. Khakpour, M. R. Mousavi, Notions of conformance testing for cyber-physical systems: Overview and roadmap (invited paper), in: Proc. of the 26th International Conference on Concurrency Theory, CONCUR 2015, Vol. 42 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pp. 18–40.

[6] M. van Osch, Automated model-based testing of hybrid systems, Ph.D. thesis, Eindhoven University of Technology, The Netherlands (2009).

[7] J. Tretmans, Model based testing with labelled transition systems, in: Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers, Vol. 4949 of Lecture Notes in Computer Science, Springer, 2008, pp. 1–38.

[8] M. Yannakakis, D. Lee, Testing of finite state systems, in: Computer Science Logic, Vol. 1584 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1999, pp. 29–44.

[9] J. V. Deshmukh, R. Majumdar, V. S. Prabhu, Quantifying conformance using the skorokhod metric, in: International Conference on Computer Aided Verification, Springer, 2015, pp. 234–250.

[10] H. Y. Abbas, Test-based falsification and conformance testing for cyber-physical systems, Ph.D. thesis, Arizona State University (2015).
URL `http://hdl.handle.net/2286/R.A.150686`

[11] M. Mohaqeqi, M. Mousavi, Sound test-suites for cyber-physical systems, in: TASE 2016, July 17–19, Shanghai, China, IEEE Computer Society, 2016.

[12] M. Althoff, An introduction to CORA 2015, in: Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems, 2015.

[13] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, K. Butts, Powertrain control verification benchmark, in: Proceedings of the 17th international conference on Hybrid systems: computation and control, ACM, 2014, pp. 253–262.

[14] R. Goebel, R. Sanfelice, A. Teel, Hybrid dynamical systems, IEEE Control Systems 29 (2), note. doi:10.1109/MCS.2008.931718.
URL `http://arxiv.org/abs/1009.3306`

[15] K. J. Aström, B. Wittenmark, Computer-controlled systems, Prentice Hall Englewood Cliffs, NJ, 1997.

[16] G. Lafferriere, G. J. Pappas, S. Yovine, Symbolic reachability computation for families of linear vector fields, J. Symb. Comput. 32 (3) (2001) 231–253.

[17] M. Althoff, Reachability analysis and its application to the safety assessment of autonomous cars, Dissertation, Technische Universitt Mnchen, Mnchen (2010).

[18] M. Althoff, B. H. Krogh, Zonotope bundles for the efficient computation of reachable sets, in: 2011 50th IEEE Conference on Decision and Control and European Control Conference, 2011, pp. 6814–6821. doi:10.1109/CDC.2011.6160872.

[19] X. Chen, Reachability analysis of non-linear hybrid systems using taylor models, Ph.D. thesis, RWTH Aachen University (2015).
URL `https://www.cs.colorado.edu/~xich8622/papers/thesis.pdf`

[20] O. Bournez, O. Maler, A. Pnueli, Orthogonal polyhedra: Representation and computation, in: Schuppen (Eds.), Hybrid Systems: Computation and Control, LNCS 1569, Springer, 1999, pp. 46–60.

[21] O. Stursberg, B. H. Krogh, Efficient representation and computation of reachable sets for hybrid systems, in: Proceedings of the 6th International Conference on Hybrid Systems: Computation and Control, HSCC'03, Springer-Verlag, Berlin, Heidelberg, 2003, pp. 482–497.
URL `http://dl.acm.org/citation.cfm?id=1768100.1768137`

[22] M. Althoff, O. Stursberg, M. Buss, Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes, Nonlinear Analysis: Hybrid Systems 4 (2) (2010) 233 – 249, {IFAC} World Congress 2008. doi:http://dx.doi.org/10.1016/j.nahs.2009.03.009.
URL `http://www.sciencedirect.com/science/article/pii/S1751570X09000442`

[23] A. Girard, C. Guernic, Zonotope/hyperplane intersection for hybrid systems reachability analysis, in: Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control, HSCC '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 215–228.

[24] A. Aerts, M. R. Mousavi, M. Reniers, A tool prototype for model-based testing of cyber-physical systems, in: M. Leucker, C. Rueda, F. D. Valencia (Eds.), Proceedings of the 12th International Colloquium on Theoretical Aspects of Computing (ICTAC 2015), Vol. 9399 of Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 563–572. doi:10.1007/978-3-319-25150-9_32.

[25] M. Kvasnica, P. Grieder, M. Baotić, Multi-Parametric Toolbox (MPT) (2004).
URL `http://control.ee.ethz.ch/~mpt/`

[26] J. A. Cook, J. Sun, J. H. Buckland, I. V. Kolmanovsky, H. Peng, J. W. Grizzle, Automotive powertrain controla survey, Asian Journal of Control 8 (3) (2006) 237–260.

[27] M. Mohaqeqi, M. R. Mousavi, W. Taha, Conformance testing of cyber-physical systems: A comparative study, in: Post-Proceedings of the 14th International Workshop on Automated Verification of Critical Systems (AVOCS 2014), Vol. 70 of ECEASST, 2014.
URL `http://journal.ub.tu-berlin.de/eceasst/article/view/982`

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

[29] M. van Osch, Hybrid input-output conformance and test generation, in: Formal Approaches to Software Testing and Runtime Verification, Vol. 4262 of Lecture Notes in Computer Science, Springer, 2006, pp. 70–84.

[30] P. J. L. Cuijpers, M. A. Reniers, W. P. M. H. Heemels, Hybrid transition systems, Computer Science Reports 02-12, Technische Universiteit Eindhoven, Department of Mathematics and Computer Science, 2002.

[31] M. Mohaqeqi, M. R. Mousavi, Approximate conformance for hybrid I/O automata, in: Proceedings of the First International Workshop on Verification and Validation of Cyber-Physical Systems (VVCPS 2016), Electronic Proceedings in Theoretical Computer Science, 2016.

[32] G. Fainekos, G. Pappas, Robust sampling for MITL specifications, in: J.-F. Raskin, P. Thiagarajan (Eds.), Formal Modeling and Analysis of Timed Systems, Vol. 4763 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 147–162.

[33] E. D. Sontag, Input to state stability: Basic concepts and results, in: Nonlinear and optimal control theory, Springer, 2008, pp. 163–220.

[34] M. Rungger, P. Tabuada, A notion of robustness for cyber-physical systems, IEEE Transactions on Automatic Control 61 (8) (2016) 2108–2123. doi:10.1109/TAC.2015.2492438.