A Tool Prototype for Model-Based Testing of Cyber-Physical Systems

Arend Aerts[1], Mohammad Reza Mousavi[2⋆], and Michel Reniers[1]

[1] Control Systems Technology Group
Eindhoven University of Technology, Netherlands
a.aerts@student.tue.nl, m.a.reniers@tue.nl
[2] Center for Research on Embedded Systems
Halmstad University, Sweden
m.r.mousavi@hh.se

Abstract. We report on a tool prototype for model-based testing of cyber-physical systems. Our starting point is a hybrid-system model specified in a domain-specific language called Acumen. Our prototype tool is implemented in Matlab and covers three stages of model-based testing, namely, test-case generation, test-case execution, and conformance analysis. We have applied our implementation to a number of typical examples of cyber-physical systems in order to analyze its applicability. In this paper, we report on the result of applying the prototype tool on a DC-DC boost converter.

Keywords: Model-Based Testing, Conformance Testing, Cyber-Physical Systems, Hybrid Systems, Acumen, Matlab

1 Introduction

Cyber-physical systems have been the focus of much research in the past few years: their structure and behavior are complex in nature and they often involve critical applications. Correctness of such systems is a major concern and, hence, rigorous validation and verification techniques are to be developed to ensure their correctness. Model-based testing [6] is a rigorous verification technique that is used to established that the behavior of an implementation conforms to the specified behavior of a model.

There are some proposals for extending the theory of model-based testing to the domain of cyber-physical systems [14, 11, 8, 7, 3, 4]. In this paper, we report on a prototype model-based testing tool for cyber-physical systems, based on a variant of the theory presented in [3, 4]. We extend the theory of [3, 4] by an offline test-case generation algorithm. Subsequently, in a prototype tool, we

⋆ M.R. Mousavi has been partially supported by the Swedish Research Council (Vetenskapsrådet) with award number 621-2014-5057 (Effective Model-Based Testing of Parallel Systems) and the Swedish Knowledge Foundation (Stiftelsen för Kunskaps- och Kompetensutveckling) with award number 20140302 (AUTO-CAAS).

implement the three steps of our model-based testing trajectory, namely, test-case generation, test-case execution, and conformance analysis.

Our prototype tool is implemented in Matlab and starts off with a hybrid-system model in a domain-specific language called Acumen Modeling Language [13]. Our choice of Acumen is motivated by the local knowledge and expertise in this particular language. However, the principles described in this paper are defined generically for hybrid-timed state sequences and hybrid automata and, hence, are applicable to a wide set of languages. Based on a model in Acumen, we generate offline test cases that are robust (up to a given threshold) with respect to minor deviations between the model and its implementation. Subsequently, we implement a test-case execution module that interfaces Matlab with the system under test. In our case, we interfaced Matlab with the Acumen simulator which simulates a model of the system under test.

In order to evaluate its applicability, we applied our tool to a few typical examples of cyber-physical systems. In this paper, we focus on one such example, namely the DC-DC boost converter to illustrate the functionality of the tool.

Organization. In Section 2, we review our variant of the conformance theory based on the approach of [3, 4]. Then, we describe our test-case generation technique for this theory of conformance. In Section 3, we describe the general architecture of the tooling. In Section 4, we report on the application of our tool to the DC-DC boost converter case study. In Section 5, we conclude the paper and present the directions of our future research and implementation activities.

This paper is based on previous work reported in [5].

2 Theory

In this section, we explain the underlying theory of our tool implementation based on and extending the theory of [3, 4].

2.1 Semantic Domain

In order to have a model of hybrid-systems behavior, we need to model the input and output trajectories of the system dynamics. In [3, 4], it is decided to take a discretized sampling of these trajectories as the basic starting point for conformance testing. The following notion of timed state sequences is defined to this end.

Definition 1 (Hybrid-Timed State Sequence (TSS) [3]). *Consider a sample size $N \in \mathbb{N}$, a dense time domain $\mathbb{T} = \mathbb{R}_{\geq 0}$, and a set of variables V. A hybrid-timed state sequence (TSS) is defined as pair (x,t), where $x \in Val(V)^N$, $t \in \mathbb{T}^N$, and $Val(V) : V \to \mathbb{R}$. The i'th element of a TSS (x,t) is denoted by (x_i,t_i), where $x_i \in Val(V)$ and $t_i \in \mathbb{T}$. 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 hybrid system according to [3], defined below, is a mapping from the initial condition and timed sequences of input variables to timed sequences of output variables.

Definition 2 (Hybrid System [3]). *Hybrid system $\mathcal{H}$ with initial condition $H \subset 2^{Val(V)}$, sample size N and input and output variables, respectively, V_I and V_O is modeled as a mapping: $\mathcal{H} : H \times \mathbb{TSS}(V_I, N) \mapsto \mathbb{TSS}(V_O, N)$. We write $y_{\mathcal{H}}(h_0, (u, t_u))$ to denote the output TSS to which the pair (u, t_u) is mapped by $\mathcal{H}$, considering h_0 as the initial condition.*

2.2 Conformance

The conformance notion [3, 4], presented below, compares the output reaction of the model and the system under test to the same input stimuli. The system under test is said to conform to the model, if the output behavior is "similar", i.e., they differ temporally or in signal values not more than the pre-defined τ and ϵ threshold, respectively.

Definition 3 ($(\tau, \epsilon)-$Conformance). *Consider a test duration $T \in \mathbb{T}$ and $\tau, \epsilon > 0$; then TSS (y, t)* (τ, ϵ)-conforms to *TSS (y', t') (both with sample size N and defined on the set V of variables), denoted by $(y, t) \approx_{\tau,\epsilon,V} (y', t')$, if and only if*

1. *for all $i \in [1, N]$ such that $t_i \leq T$, there exists $k \in [1, N]$ such that $t_k \leq T$, $|t_i - t_k| < \tau$ and for each $v \in V$, $||y_i(v) - y'_k(v)|| < \epsilon$, and*
2. *for all $i \in [1, N]$ such that $t'_i \leq T$, there exists $k \in [1, N]$ such that $t_k \leq T$, $|t'_i - t_k| < \tau$ and for each $v \in V$, $||y'_i(v) - y_k(v)|| < \epsilon$.*

A hybrid system $\mathcal{H}$ (τ, ϵ)-conforms to *a hybrid system $\mathcal{H}'$ (both with the same sample size and sets of input and output variables), denoted by $\mathcal{H} \approx_{\tau,\epsilon} \mathcal{H}'$, when for each initial condition h_0 and each TSS (u, t_u) on the common input variables V_I, $y_{\mathcal{H}}(h_0, (u, t_u)) \approx_{\tau,\epsilon,V_O} y_{\mathcal{H}'}(h_0, (u, t_u))$.*

Choosing the right conformance value for τ and ϵ is application dependent and is left to the user. However, in order to give some insight about the degree of conformance between a specification and a system under test, one may fix a value for τ and determine the minimal value of ϵ for which (τ, ϵ)-conformance holds. The following definition formalizes this concept.

Definition 4 (Conformance degree). *If $\mathcal{H}_1$ and $\mathcal{H}_2$ are two hybrid systems, given a predefined τ, the conformance degree of $\mathcal{H}_1$ to $\mathcal{H}_2$, denoted by $\boldsymbol{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2)$, is defined as $\boldsymbol{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2) := inf\{\epsilon : \mathcal{H}_1 \approx_{\tau,\epsilon} \mathcal{H}_2\}$.*

Note that our notion of conformance (degree) simplifies that of [3, 4] in a couple of ways: firstly, in our notion the number of discrete jumps is immaterial for our notion of conformance; secondly, we take the sample size of the specification and the implementation to be the same; finally, we simplified the super-dense time domain into a dense time domain. All of these are for the sake of simplicity in presentation (while keeping the definitions still applicable to our practical settings). Generalization to the original setting of [3, 4] is straightforward.

2.3 Test-Case Generation

In order to check conformance, we need to stimulate both the model and the system under test and then compare their outputs. To this end, we need to make sure that the inputs fed into the system are valid. Validity in our context has two aspects: firstly our Acumen input models (as well as other typical models of cyber-physical systems) feature state-dependent behavior. In other words, not all combinations of input valuations are valid for system specification. (This aspect is not addressed in the proposal of [3, 4] where models are assumed to be input-enabled.) Moreover, since the notion of conformance allows for some deviation between the model and the implementation, the inputs should not be too close to the boundaries of specification states (closer than the specified thresholds τ and ϵ in time and values, respectively); otherwise, the generated test cases may cease to be applicable in the course of test-case execution.

In order to give a generic exposition of our approach, we formulate it using the notion of hybrid automata, quoted below.

Definition 5 (Hybrid Automata [9]). *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;*
- *$V = V_I \uplus V_O$ 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*
  - *$Reset(V) \subseteq Val(V)^2$ 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);*
- *$F : Loc \rightarrow \mathcal{B}\left(V \cup \dot{V}\right)$ describes some constraints on variables and their derivatives and specifies the allowed continuous behavior in each location.*

In order to generate test cases for a hybrid automaton, we take two issues into account: validity of inputs in each location and the distance of the values from the location boundaries. These two aspects are summarized in the following notion of "sound and robust" test case. This notion is inspired by the notion of solution of hybrid automata [12].

Definition 6 (Solution). *A solution to the hybrid automaton $\mathcal{HA} = (Loc, V, (l_0, v_0), \rightarrow, I, F)$ is a function $s : [1, J] \rightarrow \mathbb{T} \rightarrow Loc \times Val(V)$ for some J, where for each $1 \leq j \leq J$: $\mathrm{dom}(s(j)) = [t_j, t_{j+1}]$ for some $t_j, t_{j+1} \in \mathbb{T}$, $t_1 = 0$, and*

- *$s(1)(0) = (l_0, v_0)$;*
- *for each $1 \leq j \leq J$ and $t \in [t_j, t_{j+1}]$: x satisfies $I(l)$ and $F(l)$, where $(l, x) = s(j)(t)$; and*
- *for each $1 \leq j < J$: there exists $l \xrightarrow{g,r} l'$ such that x satisfies g and (x, x') satisfies r, where $(l, x) = s(j)(t_{j+1})$ and $(l', x') = s(j + 1)(t_{j+1})$.*

Definition 7 (Sound and Robust Test Case). *A sound and (τ, ϵ)-robust test case of size N for a hybrid automaton is a TSS (y, t) with sample size N on the set V_I of variables if and only if there exists a solution s of the hybrid automaton such that*

1. *for each $i \leq N$, there exists a $j \in \mathrm{dom}(s)$, $t \in \mathrm{dom}(s(j))$, $y_i = s(j)(t) \downarrow V_I$ (soundness),*
   - *$t - \tau \in \mathrm{dom}(s(j))$ and $t + \tau \in \mathrm{dom}(s(j))$ (τ-robustness), and*
   - *for each $\epsilon' \leq \epsilon$, there exists a $t' \in \mathrm{dom}(s(j))$ such that for each variable $v \in V_O$ it holds that $||val(s(j)(t))(v) - val(s(j)(t'))(v)|| = \epsilon'$ (ϵ-robustness).*

When τ and ϵ are known from the context, we simply use the term "sound and robust test case".

3 Tool

In this section, the implementation of the conformance method in the tooling is discussed. In Fig. 1, an architectural view of our tool is presented. The grey area corresponds to the Graphical User Interface (GUI) which interacts with the tool functionality. The tooling is created in the Matlab R2013b environment. This environment was preferred to keep the implementation generic and also to be able to use the Java compatibility of Matlab in order to interface with various modeling and implementation frameworks.

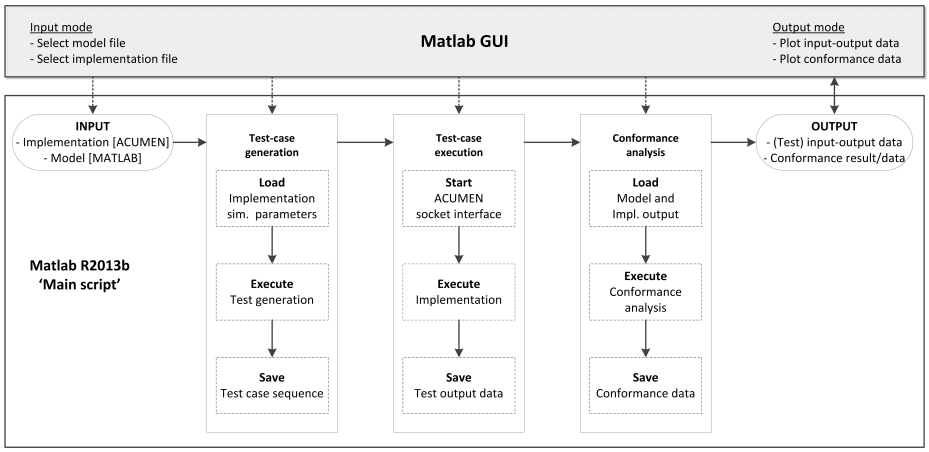


Fig. 1: Tool architecture overview

The three main steps of the conformance method are test-case generation, test-case execution, and conformance analysis and they can easily be recognized

in the architecture of the tool depicted in Fig. 1. The application of test-case generation and execution methods results in generating input-output data for both the model and the implementation under test. Application of the conformance analysis, subsequently, results in a conformance judgment possibly accompanied with an additional witness for conformance violation, which is fed into the GUI for visualization purposes.

As depicted in Fig. 1, there is a clear separation between the "Main script" module and the GUI. This division provides us with two builds of the tool, namely the Script Build and the GUI Build. The Script Build contains the full functionality of the tooling which is implemented using Matlab scripting methods (.m files), and is controllable from a command-line interface. The GUI Build contains selected functionality of the tool and offers a GUI for intuitive and easy use, especially for non-expert users. In Fig. 2, a preview of the GUI Build is provided.[3]

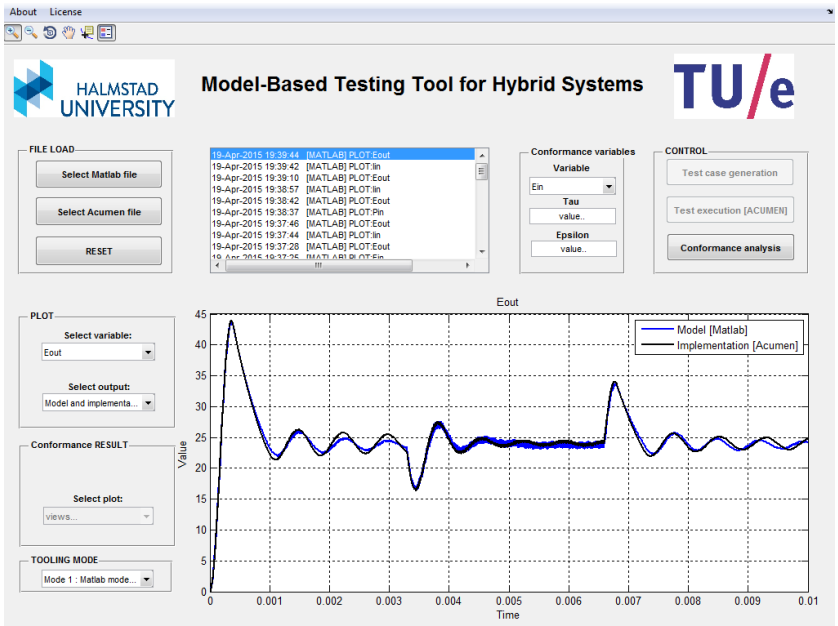


Fig. 2: Tool GUI

In the remainder of this section, we focus on the three main phases of the conformance method.

Test-case generation. In Fig. 1, before the test-case generation algorithm is executed, the simulation parameters as specified in the Acumen file are loaded into

[3] The prototype tool can obtained from `http://ceres.hh.se/mediawiki/Tool_Prototype_for_Conformance_Testing_of_CyPhy_Systems`.

Matlab. This process is performed by an Acumen file parser which extracts the specified simulation parameters and all model variables from the implementation modeled in Acumen. Definition 7 is then implemented in order to generate sound and robust test cases [**?**]. We made a slight simplification, by focusing on a subset of hybrid systems in which, firstly, the guards are not time-dependent and secondly, the invariants are only specified as intervals of input variable valuations. This simplified the implementation of the soundness and robustness checks.

Test-case execution. The test-case execution refers to the application of generated test cases on the implementation modeled in Acumen. In this step, a combination of Java and Matlab code is used in order to execute test cases / inputs on an Acumen (hybrid-system) model. This process involves communication between the Matlab tooling and the Acumen runtime environment. Note that further use of Acumen refers to the Acumen runtime environment.

The (simulator) data that is transferred between Matlab and Acumen uses the JSON-format for information exchange of the Acumen simulator state; see Fig. 3. Since existing JSON parsers failed to unwrap the simulation data from Acumen correctly, a custom Matlab JSON parser was designed and implemented for this purpose. This custom Matlab JSON parser uses the preloaded model variables of the Acumen file parser to extract all model variables of the implementation.

To initiate the communication between the Matlab tooling and Acumen, a command line interface is used (from within Matlab) to start up Acumen in the background. Moreover, Acumen automatically loads a prespecified Acumen file, in this case the implementation, and starts the simulation of this model. Since this start-up sequence is performed with Acumen in server mode, it creates a socket connection to execute a co-simulation. Hence, when the simulation of the implementation is automatically started, Acumen waits for a valid socket connection or client, in this case the Matlab tooling. In Fig. 1, this start-up process of the socket connection between Matlab and Acumen is shown as the first step of the test-case execution.

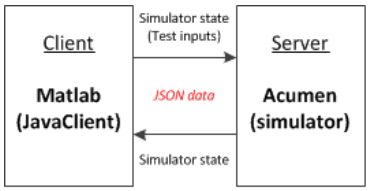


Fig. 3: Socket connection

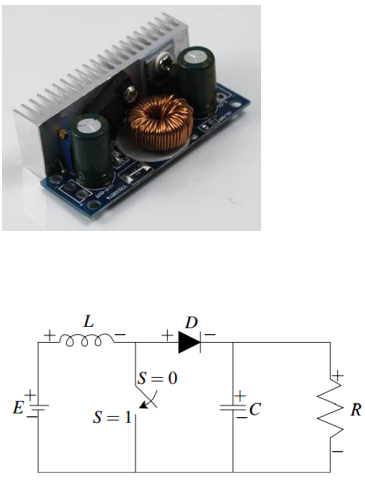
As soon as the Matlab tooling initiates the socket connection by running its embedded Javaclient, the (initial) simulator state is send over to Matlab. When Matlab returns the simulator state to Acumen, one simulation step of the implementation (in Acumen) is performed. This process repeats itself for every timestep of the full simulation duration as specified in the Acumen model file.

Conformance analysis. The conformance analysis is an implementation of Definition 3. In addition to providing a yes/no answer, the tool provides a visualization of the counter-example in case the conformance relation does not hold. This

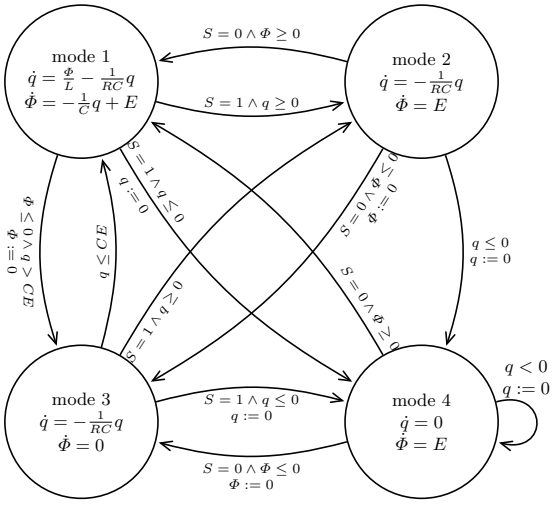
is achieved by plotting both the specification and the implementation trajectories and depicting the case of violation by a (τ, ϵ) box around the specification point which does not find a counterpart in the implementation (or the other way around). Additionally, our implementation automatically calculates the conformance degree based on Definition 4.

4 Experiment: DC-DC boost converter

The developed tool has been experimented with on several classical hybrid-system examples, such as the bouncing ball, the thermostat, and the DC-DC boost converter. The DC-DC boost converter example is discussed below. The DC-DC boost converter is a hybrid-system example (see [10]) that originates from the field of electrical engineering and is used to "boost" an input DC voltage to an increased output value. In Fig. 4a, such a boost converter is shown together with a schematic that shows the principle of operation. The boost of the DC voltage is a consequence the combined physical properties of the inductor L and capacitor C, which are controlled by the switch S and diode D. This process transforms the input voltage E to an increased output voltage that is applied to the resistive load R. Note that the control elements of the boost converter transform the otherwise continuous system into a hybrid system. Finally, the system is made input dependent by tuning the resistive load R which results in internal stabilizing behavior of the boost converter.



(a) PCB [2] and schematic [1]



(b) Hybrid automaton [1]

In Fig. 4b, the hybrid-automaton model of the DC-DC boost converter is shown. The four discrete states of the system are solely dependent on the position

of the switch S and the mode of the diode D (conducting/blocking). In addition, the physical properties of the system are modelled by the electric charge q of the capacitor and the magnetic flux ϕ of the inductor. For further understanding of the specified dynamics, state guards and reset maps see [10].

In Fig. 5, the output power of a specific boost converter is shown. The blue and black lines indicate the response of the model (in Matlab) and implementation (Acumen) respectively, which are visibly diverging. Hence, conformance analysis is needed in order to evaluate the conformance (degree) of the implementation with respect to the model. Non-conformance is detected and is indicated in red. In the lower sub-plot, an automatic zoom of the non-conformance area is performed in order to provide visual feedback of the τ-ϵ area around the corresponding data point. The following values are used in the conformance analysis of Fig. 5: $\tau = 0.00001$, $\epsilon = 7$.

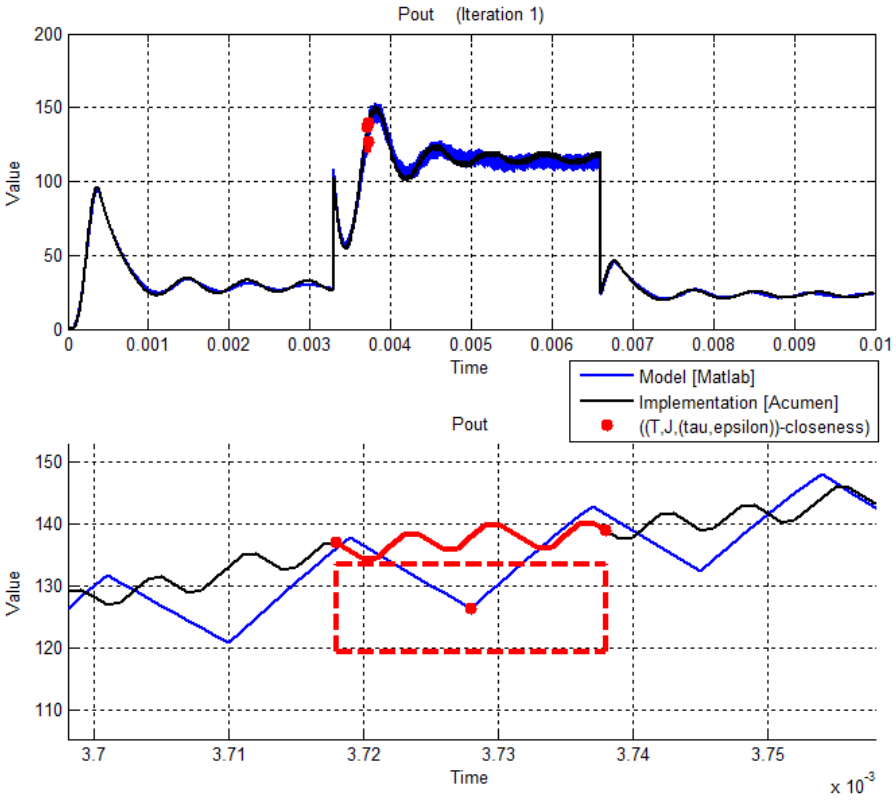


Fig. 5: DC-DC boost converter conformance analysis

5 Conclusions and Future Work

In this paper, we reported on an implementation of a conformance testing theory for cyber-physical systems, based on the conformance notion of [3, 4]. To this end, we have developed the notion of sound and robust test cases. We have used this notion to generate off-line test cases from a hybrid-system model in the domain specific language Acumen [13]. We have implemented the test-case generation, test-case execution, and conformance analysis in a Matlab-based prototype.

In order to manage the complexity of the implementation, we have made several simplifying assumptions on the structure of the invariants and guards in the specification. Relaxing these assumptions requires non-trivial numerical analysis of the specification and is left for future work. Turning our off-line test-case generation into an on-line test-case generation algorithm is another non-trivial extension. This is particularly interesting when non-determinism is allowed in the specification. Defining a notion of coverage along the lines of [7, 8] and adapting our test-case generation algorithm in order to maximize specification coverage is another avenue for our future research.

References

1. DC-DC boost converter [figures]. `http://goo.gl/rstOKi`, April 2015.
2. DC-DC boost converter pcb [figure]. `http://www.goo.gl/pDNyw3`, April 2015.
3. H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Conformance testing as falsification for cyber-physical systems. In *ICCPS*, 2014.
4. H. Abbas, H. Mittelmann, and G. Fainekos. Formal property verification in a conformance testing framework. In *MEMOCODE*, 2014.
5. A. Aerts. Model-based testing tool for hybrid systems in Acumen. Technical Report CST 2015.073, TU/e, 2015.
6. M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. *Model-Based Testing of Reactive Systems: Advanced Lectures*. LNCS 3472. Springer, 2105.
7. T. Dang. Model-based testing of hybrid systems. In *Model-based Testing for Embedded Systems*. CRC Press, 2011.
8. T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.
9. R. Goebel, R. Sanfelice, and A. Teel. Hybrid dynamical systems. *IEEE Control Systems Magazine*, 29(2):28–93, April 2009.
10. W. P. M. H. Heemels and B. de Schutter. Modeling and Control of Hybrid Dynamical Systems. TU/e, Lecture notes course 4K160, 2013.
11. A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In *HSCC*, 2007.
12. M. Lemmon. On the existence of solutions to controlled hybrid automata. In *HSCC*, 2000.
13. W. Taha, P. Brauner, Y. Zeng, R. Cartwright, V. Gaspes, A. Ames, and A. Chapoutot. A core language for executable models of cyber-physical systems (preliminary report). In *ICDCS*, 2012.
14. M. van Osch. Hybrid input-output conformance and test generation. In *FATES/RV*. 2006.