Chapter 5

PRACTICAL ALGORITHMS FOR CONFORMANCE TESTING

5.1 Introduction

One of the benefits of Model-Based Design is that it enables the iterative development of CPS. During this process, a series of models and implementations are developed, with varying fidelity, for the same underlying CPS. One of the problems that arises, is to quantify how "close" two systems are to each other. We refer to this as the conformance problem.

Due to the non-linear, hybrid nature of these systems, in general, it is not possible to certify that two systems are conformant to each other. We can only detect non-conformance. In this work, we review and adopt the semi-formal approach for conformance testing of CPS presented in [3]. We propose an automated black/grey box conformance testing framework. In this framework, we simulate two systems with the same input signal and then evaluate the output signals with our notion of conformance. A stochastic optimizer is then utilized to select the next input signal. Our method returns the most non-conformant behavior found after running a predetermined number of tests. Our framework is based on a notion of closeness between signals that encapsulates both spatial and temporal differences. Spatial difference by itself is not sufficient since in many cases, two system trajectories, due to factors such as jitter, may be slightly shifted. This shift is generally acceptable in model based design and therefore we do not wish to classify this as non-conformant behavior.

Finally, we introduce and integrate code coverage metrics for conformance testing of Cyber-Physical Systems. By doing so, we can ensure that particular locations of

the control logic have been covered in the testing process.

We illustrate our methods with an industrial scale high-fidelity engine model from Toyota.

In this chapter:

- We review and adopt the notion of conformance introduced in [3].
- We introduce code coverage methods for conformance testing of CPS.
- We illustrate our work on an industrial scale high-fidelity engine model from Toyota.

5.2 Problem Formulation

In the conformance testing framework, we consider two systems: Σ_1 for the Model and Σ_2 for the implementation. Utilizing the notation from Sect. 2.1, both systems can be viewed as functions:

$$\Delta_{\Sigma_1} : X_0 \times \mathbf{U} \to Y^N \times \mathfrak{T}$$
$$\Delta_{\Sigma_2} : X_0 \times \mathbf{U} \to Y^N \times \mathfrak{T}$$

The systems take as an imput the same initial conditions $x_0 \in X_0$ and input signals $u \in \mathbf{U}$ to produce:

$$\text{Output signal } \mathbf{y} : N \to Y \text{ and timing function } \tau : N \to \mathbb{R}_+ \text{ for } \Delta_{\Sigma_1}.$$
$$\text{Output signal } \mathbf{y}' : N \to Y \text{ and timing function } \tau' : N \to \mathbb{R}_+ \text{ for } \Delta_{\Sigma_2}.$$

We recall that $\mu = (\mathbf{y}, \tau)$ and $\mu' = (\mathbf{y}', \tau')$ are referred to as *timed state sequences* (TSS). TSS is a widely accepted model for reasoning about real-time systems [11]. A timed state sequence can represent a computer-simulated trajectory of a CPS or the sampling process that takes place when we digitally monitor physical systems.

We remark that a timed state sequence can represent both the internal state of the software/hardware (usually through an abstraction) and the state of the physical system.

Our high level goal is to determine whether there exists a pair of (initial conditions, input signal) that cause the model and its implementation to produce significantly different outputs; and if such a pair exists, to find it and present it to the user.

We define conformance over two trajectories as follows.

Definition 5.2.1 (*$(\delta, \varepsilon) - closeness$* **[3]**) *Two timed state sequences, or* trajectories, *$\mu = (\mathbf{y}, \tau)$ and $\mu' = (\mathbf{y}', \tau')$ are $(\delta, \varepsilon) - close$ if*
(a) for all $i \in N$, there exists $k \in N$ where

$$|\tau(i) - \tau'(k)| < \delta \text{ and } \|\mathbf{y}(i) - \mathbf{y}'(k)\| < \varepsilon$$

(b) for all $i \in N$, there exists $k \in N$ where

$$|\tau(i) - \tau'(k)| < \delta \text{ and } \|\mathbf{y}'(i) - \mathbf{y}(k)\| < \varepsilon$$

If these conditions are met we say that μ and μ' are conformant with degree (δ, ε).

The definition says that within any time window of size 2δ, there must be a time when the trajectories are within ε or less of each other. Allowing some 'wiggle room' in both time and space is important for conformance testing: when implementing a Model, there are inevitable errors. These are due to differences in computation precision, clock drift in the implementation, the use of inexpensive components, unmodeled environmental phenomena, etc, leading to the Implementation's output to differ in value from the Model's output, and to have different timing characteristics.

Problem 5.2.1 (System Conformance) *Given two Systems Σ_1 and Σ_2, for every $\mu \in \mathcal{L}(\Sigma_1)$ and $\mu' \in \mathcal{L}(\Sigma_2)$, μ and μ' are conformant with degree (δ, ε), where*

$$\mathcal{L}(\Sigma_1) = \{(\mathbf{y}, \tau) \mid \exists x_0 \in X_0 \,.\, \exists u \in \mathbf{U} \,.\, (\mathbf{y}, \tau) = \Delta_{\Sigma_1}(x_0, u)\} \text{ and}$$
$$\mathcal{L}(\Sigma_2) = \{(\mathbf{y}', \tau') \mid \exists x_0 \in X_0 \,.\, \exists u \in \mathbf{U} \,.\, (\mathbf{y}', \tau') = \Delta_{\Sigma_2}(x_0, u)\}.$$

In many cases, the implementation of the model is given as a black-box or gray-box where the internals of the model are hidden or partially known, respectively. In this case, we can only analyze the system by observing the input-output behavior. For gray-box systems, we might have partial information such as a subset of states of the system or the current locations or modes of the underlying control logic of the system. Therefore, in general, Prob. 5.2.2 does not lend itself to complete analytical techniques. However, since we can detect non-conformant behavior, we can pose this problem as a falsification problem. Namely, we can convert this problem to a testing problem where we search for non-conformant behavior.

Problem 5.2.2 ((δ, ε) - Conformance Testing) *Given two Systems Σ_1 and Σ_2, find $\mu_1 = \Delta_{\Sigma_1}(x_0, u)$ and $\mu_2 = \Delta_{\Sigma_2}(x_0, u)$ such that μ_1 and μ_2 are non-conformant with degree (δ, ε).*

An overview of the solution is presented in Fig. 5.1. Given two systems, the sampler produces a point x_0 from the set of initial conditions, and an input signal u. The initial conditions and input signal are passed to the system simulator for both systems which returns two trajectories. The trajectories are than analyzed and a conformance degree is returned. Here, either δ or ϵ are fixed a priori. The conformance degree is used by the stochastic sampler to decide on next initial conditions and inputs. The process terminates once a maximum number of tests is reached. The method returns non-conformant outputs (if they are found) along with the corresponding initial conditions and input signal.

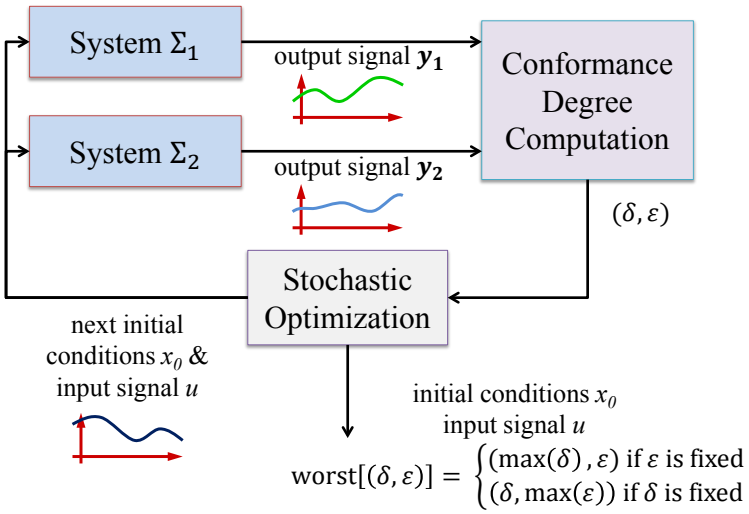


Figure 5.1: Overview of the solution to Prob. 5.2.2, the conformance testing problem for CPS.

5.3 Code Coverage for CPS

In software testing, many techniques and standards have been developed to ensure that the developed software operates as intended. Techniques ranging from static/dynamic analysis [58] and fault injection [110] to mutation testing [18] and code coverage [136]. In the following we investigate code coverage techniques for conformance testing of CPS.

Code coverage is of particular importance when dealing with CPS. In order to ensure that the testing process includes parts of the system that are rarely visited/tested. In closed-loop testing, a CPS is composed of a controller and a plant, as illustrated in Fig. 5.2. The physical representation of the system that needs to be controlled is called the plant. The controller processes signal data and generates an actuation signal using a control law that has a specific objective. From the testing perspective, the control logic can be instrumented automatically [55] and we can

conduct coverage over the states of the resulting finite state machine.

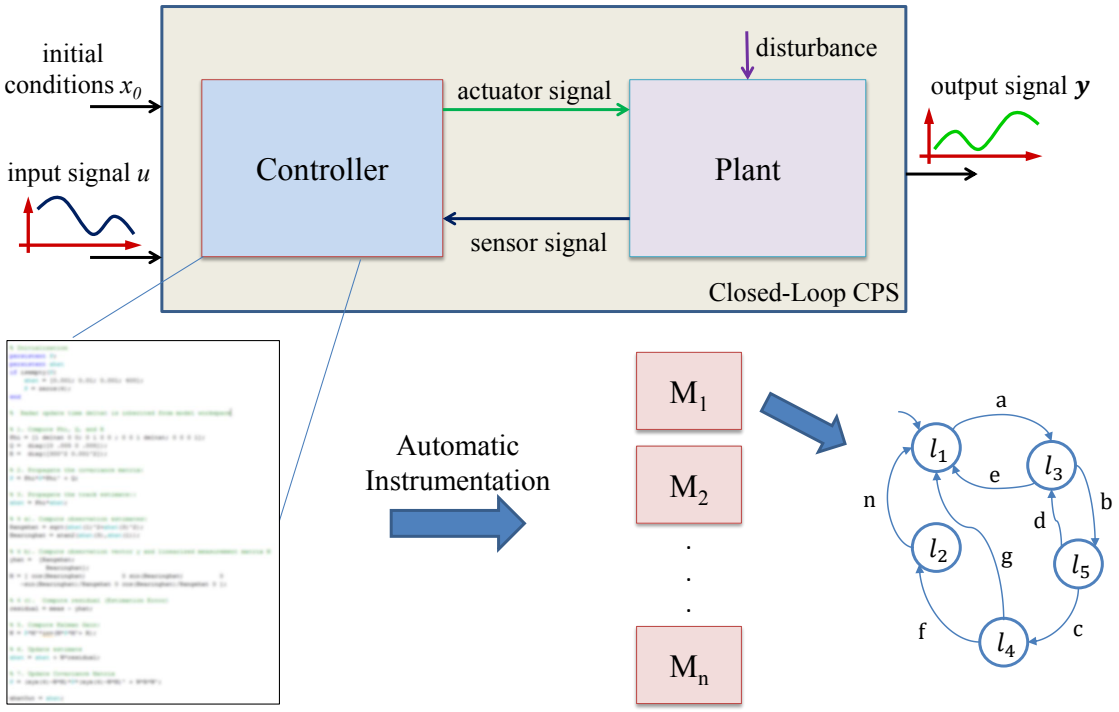


Figure 5.2: Typical closed-loop cyber-physical system. The discrete logic of the controller code is automatically instrumented and modeled using Extended Finite State Machines (EFSM) [10]. Namely, for each conditional statement code block in the controller code, an EFSM is generated.

Example 5.3.1 (Model Instrumentation) *We illustrate the model instrumentation process with our running example from Sect. 2.5.3. Given a simulink model of the hybrid nonlinear system, by analysing the function block, we are able to extract the control logic of the controller. In addition, we modify the model so that we are able to observe in which location the model is in when analyzing the output trajectories. This process is illustrated in Fig. 5.3 .*

In a typical V process in MBD, as presented in Fig. 1.1, a model is developed

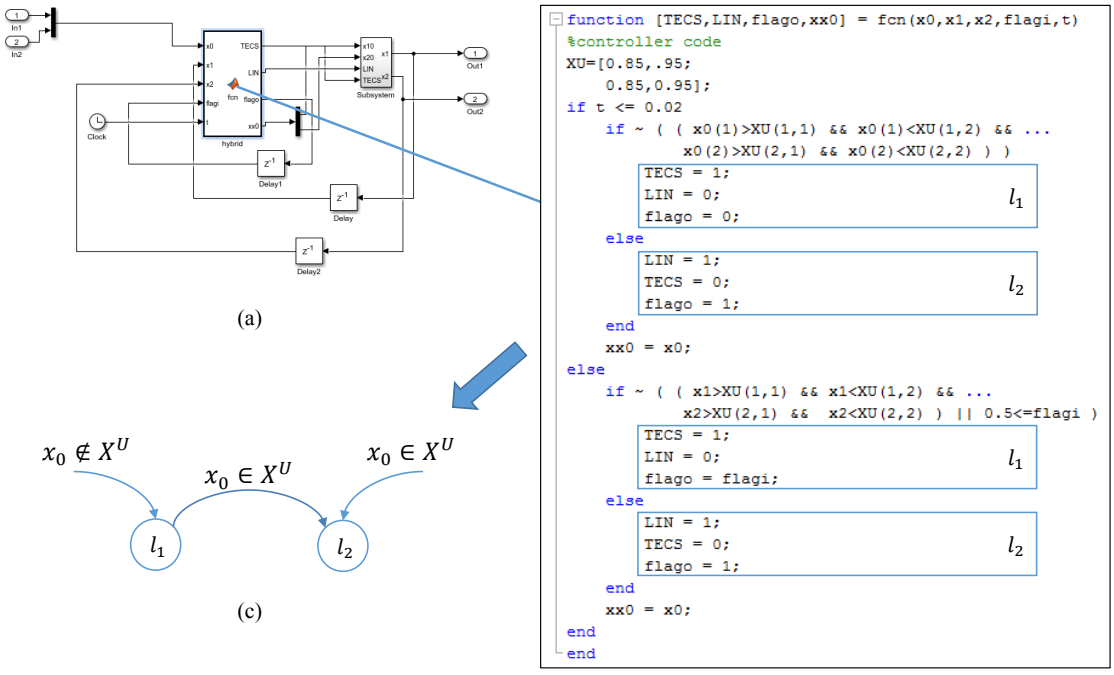
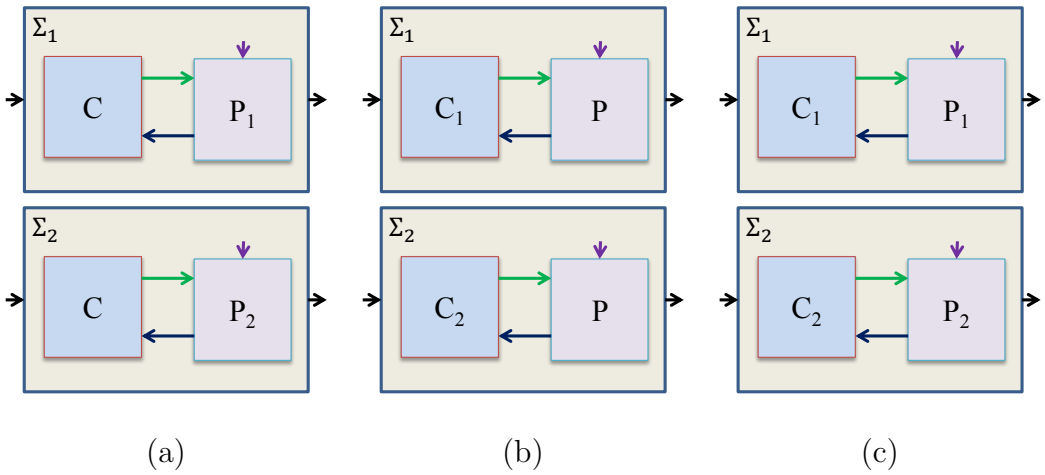


Figure 5.3: Model instrumentation of the hybrid nonlinear system (HS) presented in Sect. 2.5.3. (a) Simulink model of the HS system. (b) Controller code of the HS simulink model inside the function block. (c) Automatically extracted control logic from the controller code.

iteratively. In this iterative process, depending on the development stage, various conformance testing scenarios may be encountered. In Fig. 5.4, we list the three possible scenarios. In the following, we will provide a conformance testing solution for each. First, we will consider the case where the controller is the same in both the model and the implementation but the plant is different. After, we will consider the case when the plant is the same but the controller is different.



(a) (b) (c)

Figure 5.4: Conformance testing scenarios in the V process in MBD. (a) Two systems with the same controller and different plant. (b) Two systems with different controllers and same plant. (c) Two systems with different controllers and different plants.

5.4 Controller Coverage with Different Plants

In the right hand side of the V process, presented in Fig. 1.1, the model gets developed and tested through Model in the loop (MIL), Software in loop (SIL), Processor in loop (PIL), Hardware in loop (HIL) and implementation stages. Typically, the main components that change in this process are the plant and the controller of the model: from the most abstract (Model), to the most realistic (Implementation). In this section, we consider the case where the controller remains the same in both the model and implementation but the plant is different.

First, we extract the controller code through an instrumentation process to obtain the structure of the control logic of the controller but not the actual numerical computations. In this process, for every conditional statement code block, an Extended Finite State Machine (EFSM) M_i is generated. Depending on the size of the resulting machines, the practitioner may desire to test all or some specific combinations of

locations for each M_i. In most practical applications, the number of machines and locations is large and therefore testing all the combinations is not a feasible task. We discuss this case further in Sect. 5.6. In the rest of this section, we assume that the practitioner has a predetermined combination of locations that they would like to cover.

For a particular combination of locations, the problem is posed as an optimization problem. We modify the solution presented in Fig. 5.1 by changing the cost function such that the system is driven towards the target locations. We accomplish this by utilizing the hybrid distance metric introduced in [2]. We refer the reader to [2] for the technical details. The intuition behind the metric is as follows. When the current location is different from the target location, then the distance is the distance to the closest guard that will enable the transition to the next control location that reduces the path distance. When the system is in the target locations, the cost function is set to the (δ, ε) conformance metric. In this way, not only do we attempt to reach the target locations, but also attempt to find the most non-conformant behavior there.

5.5 Coverage with Different Controllers

Throughout the entire V-process of MBD, it is often the case that the controller is either modified, abstracted and refined depending on the application. When we encounter this case, we would like to ensure conformance between the previous and the new version of the controller. To achieve this, coverage of both controller locations is necessary. Given system Σ_1 with controller C_1 and Σ_2 with controller C_2, we extract the controller logic through an automatic process $\mathcal{A}(C_1) = M$ and $\mathcal{A}(C_2) = N$. Next, we conduct conformance testing with coverage over the product automaton $\mathcal{P} = M \times N$ which contains the states of both controller locations. The search process follows similarly as in Sect 5.4.

5.6 Covering Arrays for Code Coverage of CPS

In many cases, the number of conditional statements inside the controller code is very large, and generating system simulations to cover all the possible interactions is not feasible. In this section, we propose the use of covering arrays [42] to enable conformance testing testing of a subset of interactions for the conditional statements reached in the testing process. In particular, we will utilize methods from the Covering Array literature, to improve on existing conformance testing methods. Covering arrays have been successfully utilized in a variety of applications such as blackbox testing of hardware systems, learning an unknown Boolean function, sequence alignment of DNA, compressing inconsistent data, etc.

For conformance testing, consider the following example. A controller has 4 conditional statement blocks where M_1 has 15 locations, M_2 has 10 locations, M_3 has 8 locations and M_4 has 25 locations. To check all interactions of the locations we would have to attempt to reach $15 \times 10 \times 8 \times 25 = 30,000$ combinations of locations with a predetermined number of tests each. This approach becomes infeasible very quickly. However, we can utilize covering arrays to drastically decrease the number of interactions of locations. As a trade-off, we do not test for all interactions of locations but cover n-interactions. For example, a 3-interaction covering array reduces the number of interactions of locations to 3750, while a 2-interaction covering array reduces it to 376 location interactions. Furthermore, there is research that shows that a significant number of bugs in software can be found within n-interactions [102]. With covering arrays, the number of sequences of locations needed for coverage, for a fixed number of interactions, grows logarithmically in the number of parameters [102].

Covering arrays have been studied extensively over the last 40 years. Researchers have been primarily focused on the problem of determining the minimum size of

covering arrays. For an overview, see [73]. In general, there has been significant progress in determining upper bounds for the minimum size of the covering arrays. Furthermore, many algorithms [67, 33, 42] and tools [41] have been developed for the construction of covering arrays. In our framework, we utilize the AETG covering array generation system [41].

With covering arrays, we can define seeds, or combinations of locations that must be covered by the covering array. We can define avoids, or interactions that do not need to be covered. We can define constraints, which are interactions that should not be covered. We can also define the strength of some factors over others. All this allows for a very flexible framework when conducting conformance testing.

5.7 Case Study: Toyota Engine Controller

We consider two models of varying fidelity of an internal combustion engine. The models are academic, but of industrial complexity, and are provided by the Toyota Technical Center [1] . The first model Σ_1 is a high-fidelity engine plant and controller while the second one Σ_2 is a simplified, polynomial approximation of Σ_1. In this case, with conformance testing, we aim to explore how close the two models are and whether we can utilize Σ_2 for model predictive control [37, 38, 84, 85]. Both Σ_1 and Σ_2 are modeled in Matlab/Simulink. Σ_1 has 1878 blocks, including 10 integrator blocks, 47 lookup tables, 19 saturation blocks, 27 switch blocks, and 44 subsystem blocks. The model of the plant has 11 continuous and 68 discrete states. The controller is defined in a function block and contains $\approx$ 500 lines of code with multiple if-else conditional statements. Model Σ_2 has 1858 blocks, including 47 lookup tables, 17 saturation blocks, 27 switch blocks and 49 subsystem blocks with no continuous states and 60

[1] We note that due to confidentiality agreements, we have removed the unit measurements of the y-axes from the figures presented in this section.

discrete states. Both Σ_1 and Σ_2 have two inputs and one output each. The inputs are *Fuel Level* and the number of engine revolutions (Ne). The output is the pressure intake manifold (Pin). In the following, we run several experiments with different optimization functions. We also provide partial coverage of the input space as well as branch coverage for the controller code.

5.7.1 Experimental Design Parameters

Simulated Annealing

The search space over system inputs is defined over control points which are interpolated to generate an input signal as defined in Section 2.4. Here, both inputs have 10 control points. The search space also includes timing of the control points. Since the initial and terminal timing control points are fixed at the start and end of the simulation, respectively, for each input, we have an additional 8 search variables . The signals are interpolated through a piecewise constant interpolation function. An example input signal is presented in Fig. 5.5 (a) (Left). In total, we have 36 search variables. The stochastic optimizer utilized is a simulated annealing algorithm [2]. After 1000 tests, we found the system inputs that generate the outputs in Fig. 5.5. In the left figure, the resulting system inputs are presented. Figure 5.5 (a) (Middle) shows the system outputs, where the red out is from the complex model Σ_1, and the blue line is from the simplified model Σ_2. The figure on the right is the Pareto front over (δ, ϵ) which illustrates the ϵ difference over the δ range. The ϵ value at $\delta = 0$ represents the instantaneous difference between the outputs of the two models at the same time t, while the ϵ value at $\delta = 0.2$ represents the largest difference between the outputs of the two models while comparing the values of the two signals in a moving window of width 0.2s. Note that the Pareto Front is over 1000 tests and is therefore

an under-approximation of the *true* Pareto Front over all system behaviors. We can guarantee that (δ, ϵ) is at least as large as shown in Fig. 5.5 (a) (Right).

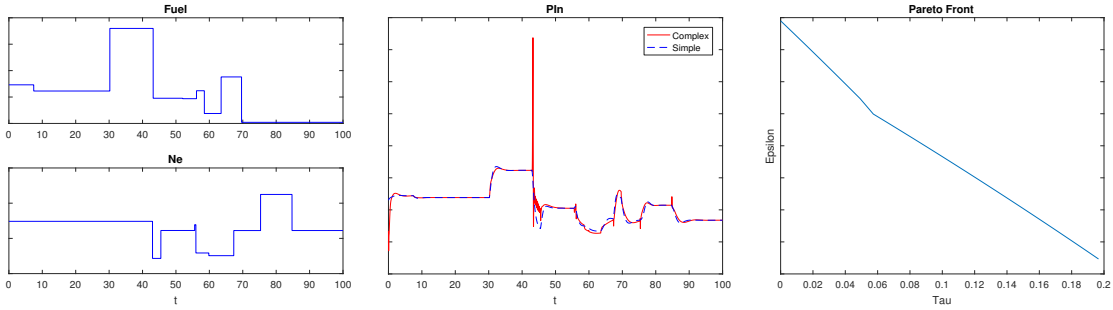
Grid Search

One of the goals in the conformance testing process is to provide a level of input search space coverage. To do so, we developed a grid search algorithm which divides the input search space in a grid. Formally, search is conducted over the following set $S = \{(x, y) : x \in [min(U_1) : \frac{range(U_1)}{g-1} : max(U_1)]\ and\ y \in [min(U_2) : \frac{range(U_2)}{g-1} : max(U_2)]\}$, where U_1, U_2 are input signals for inputs 1 and 2, respectively. Here, g is the granularity of the grid. The results of the grid search algorithm are presented in Fig. 5.5 (b). The middle figure shows highly non-conformant behavior between Σ_1 and Σ_2, indicating a possible singularity in either model at this particular input. Here, since the maximal difference between the two trajectories is constant over a period of $2 \times \delta$, the Pareto Front on the right figure is flat.

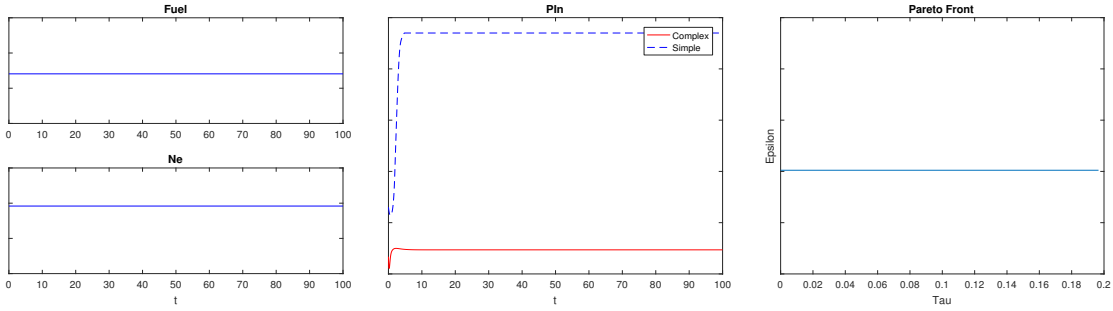
In the future, we plan to implement a grid search algorithm which includes several grids with respect to time and the input signals are interpolated linearly between them. This way, we can also study switching behaviors of the system.

Controller Branch Coverage

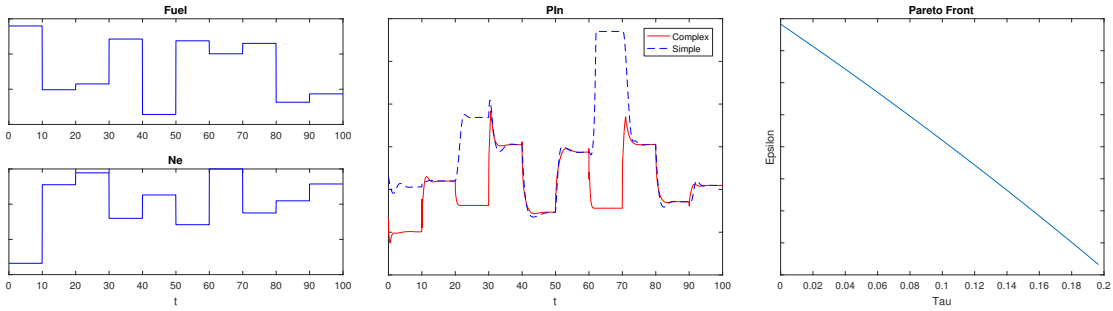
The next step in the analysis is to conduct conformance testing while making sure that we have controller branch coverage. We will consider three if-else blocks to be of particular importance. Namely M_1 with 12 branches, M_2 with 2 branches, and M_3 with 4 branches. These if-else blocks are instrumented automatically from the Simulink model. The instrumentation process extracts the branch information from the Embedded Matlab code function block and passes it to S-TaLiRo for conformance testing. We follow the approach presented in Sect. 5.4 which is related to the



(a) Simulated Annealing



(b) Grid Search



(c) Controller Branch Coverage

Figure 5.5: Experimental results under various conformance testing methods. Left: System inputs for *Fuel* and *Ne*. Middle: System output for *Pin*. Right: Pareto front over all tested system behaviors that illustrates the ε difference over the δ range.

testing approach in [55].

After running our algorithm, we found non-conformant behavior in locations $(6, 1, 4)$ for M_1, M_2, and M_3. The results are presented in Fig. 5.5 (c). The target locations $(6, 1, 4)$ were reached after 803 tests. After reaching the target locations, the stochastic optimizer was focused on maximizing the (δ, ε) metric between trajectories.

Following our testing results, Ken Butts, an Executive Engineer from the Powertrain Control Department at Toyota Technical Center provided the following statement:

"*Their tool has pointed out where the high-fidelity model is fragile and producing erroneous results. It is good to know that the polynomial model performs well at these cases.*"

5.8 Related Works

Conformance notions have been studied in the past. Tretmans [137] defined an Input-Output conformance (IOCO) notion for discrete labeled transition systems. This notion is defined as a relationship where the implementation does not generate an output that is not producible by the specification. Also, the implementation always produces an output when it is required by the specification. Later, Van Osch [114] extended IOCO to hybrid transition systems (HTS) by incorporating continuous-time inputs.

In [141], the authors extend the work by [137] where the implementation is a black box that generates trajectories. In this framework, the specification is represented as a timed automaton. Here, every trajectory in the language of the implementation needs to satisfy the timed automaton representing the specification.

In the work by Brandl et al. [28] the authors present a method for conformance checking through qualitative reasoning techniques. The method utilzes mutation-

based test case generation on action systems for IOCO [137] conformance checking.

In [48], the authors propose a notion of conformance based on the Skorokhod metric. This notion captures both timing and spacial differences between trajectories and supports transference of properties in the development process. However, the physical interpretation and computation of the Skorokhod distance is not as straightforward as for the (τ, ϵ) metric [3].

In [92], the authors provide an overview of confrmance testing methods for hybrid systems. They compare different notions of robustness for conformance testing and they list current challenges in the area.

5.9 Conclusions and Future Work

In this chapter, we presented a conformance testing framework to test how "close" two systems are. We presented a black/gray box framework that includes controller code coverage methods for improved testing. Next, we discussed the use of covering array techniques to dramatically reduce the number of tests necessary to test n-interactions of the branching conditions in the controller code. Finally, we demonstrated our methods with prototype high-fidelity models from Toyota.

In the future, we plan to incorporate a learning algorithm to obtain the regular language of the transitions between location n-tuples and estimate the accompanying probabilities for non-conformant behavior. This could possibly be represented as a Markov Decision Process. This would make it easy to observe "problematic" or non-conformant location changes and facilitate the debugging process.

Bibliography

[1] H. Abbas and G. Fainekos. Linear hybrid system falsification through local search. In *Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 503–510. Springer, 2011.

[2] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 12(s2), May 2013.

[3] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Conformance testing as falsification for cyber-physical systems. Technical Report arXiv:1401.5200, January 2014.

[4] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Wip abstract: Conformance testing as falsification for cyber-physical systems. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, pages 211–211. IEEE, 2014.

[5] H. Abbas, B. Hoxha, G. Fainekos, and K. Ueda. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, pages 1–6. IEEE, 2014.

[6] E. Ábrahám-Mumm, U. Hannemann, and M. Steffen. Verification of hybrid systems: Formalization and proof rules in pvs. In *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on*, pages 48–57. IEEE, 2001.

[7] T. Akazaki and I. Hasuo. Time robustness in mtl and expressivity in hybrid system falsification. In *International Conference on Computer Aided Verification*, pages 356–374. Springer, 2015.

[8] A. Alfonso, V. Braberman, N. Kicillof, and A. Olivero. Visual timed event scenarios. In *Proceedings of the 26th Int. Conference on Software Engineering*, pages 168–177. IEEE Computer Society, 2004.

[9] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233 – 249, 2010.

[10] R. Alur. *Principles of cyber-physical systems*. MIT Press, 2015.

[11] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In J. Mitchell, editor, *5th Annual IEEE Symp. on Logic in Computer Science (LICS)*, pages 414–425. IEEE Computer Society Press, June 1990.

[12] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[13] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*. Springer, 1993.

[14] R. Alur and D. L. Dill. Theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[15] R. Alur, K. Etessami, S. La Torre, and D. Peled. Parametric temporal logic for model measuring. *ACM Trans. Comput. Logic*, 2:388–407, July 2001.

[16] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.

[17] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 592–601. ACM, 1993.

[18] P. Ammann and J. Offutt. *Introduction to software testing*. Cambridge University Press, 2016.

[19] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[20] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise linear dynamical systems. In *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, pages 21–31. Springer, 2000.

[21] E. Asarin, A. Donzé, O. Maler, and D. Nickovic. Parametric identification of temporal properties. In *Runtime Verification*, volume 7186 of *LNCS*, pages 147–160. Springer, 2012.

[22] M. Autili, P. Inverardi, and P. Pelliccione. Graphical scenarios for specifying temporal properties: an automated approach. *Automated Software Engineering*, 14(3):293–340, 2007.

[23] S. Bacherini, A. Fantechi, M. Tempestini, and N. Zingoni. A story about formal methods adoption by a railway signaling manufacturer. In *FM 2006: Formal Methods*, pages 179–189. Springer, 2006.

[24] C. Baier, J.-P. Katoen, et al. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[25] S. Bak, S. Bogomolov, and T. T. Johnson. Hyst: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.

[26] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 142–156. Springer, 2004.

[27] L. Bozzelli and S. La Torre. Decision problems for lower/upper bound parametric timed automata. *Formal Methods in System Design*, 35(2):121–151, 2009.

[28] H. Brandl, M. Weiglhofer, and B. K. Aichernig. Automated conformance verification of hybrid systems. In *Quality Software (QSIC), 10th International Conference on*, pages 3–12. IEEE, 2010.

[29] G. Brat, D. Drusinsky, D. Giannakopoulou, A. Goldberg, K. Havelund, M. Lowry, C. Pasareanu, A. Venet, W. Visser, and R. Washington. Experimental evaluation of verification and validation tools on martian rover software. *Formal Methods in System Design*, 25(2-3):167–198, 2004.

[30] G. Bruns and P. Godefroid. Temporal logic query checking. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*, pages 409 – 417. IEEE Computer Society, 2001.

[31] W. Chan. Temporal-logic queries. In *Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *LNCS*, pages 450–463, London, UK, 2000. Springer.

[32] R. N. Charette. This car runs on code. *IEEE spectrum*, 46(3):3, 2009.

[33] M. Chateauneuf, C. J. Colbourn, and D. L. Kreher. Covering arrays of strength three. *Designs, Codes and Cryptography*, 16(3):235–242, 1999.

[34] M. Chechik and A. Gurfinkel. Tlqsolver: A temporal logic query checker. In *Proceedings of the 15th International Conference on Computer Aided Verification*, volume 2725, pages 210–214. Springer, 2003.

[35] T. Chen, M. Diciolla, M. Z. Kwiatkowska, and A. Mereacre. A simulink hybrid heart model for quantitative verification of cardiac pacemakers. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 131–136. ACM, 2013.

[36] X. Chen, E. Abraham, and S. Sankaranarayanan. Flow*: An analyzer for nonlinear hybrid systems. In *Computer-Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 258–263. Springer-Verlag, 2013.

[37] R. Choroszucha, J. Sun, and K. Butts. Closed-loop model order reduction and mpc for diesel engine airpath control. In *American Control Conference (ACC), 2015*, pages 3279–3284. IEEE, 2015.

[38] R. Choroszucha, J. Sun, and K. Butts. Nonlinear model order reduction for predictive control of the diesel engine airpath. In *American Control Conference (ACC), 2016*, pages 5081–5086. IEEE, 2016.

[39] A. Chutinan and K. R. Butts. Dynamic analysis of hybrid system models for design validation. Technical report, Ford Motor Company, 2002.

[40] A. Chutinan and B. Krogh. Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations. In *Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 76–90. Springer, 1999.

[41] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton. The aetg system: An approach to testing based on combinatorial design. *IEEE Transactions on Software Engineering*, 23(7):437–444, 1997.

[42] C. J. Colbourn and J. H. Dinitz. *Handbook of combinatorial designs*. CRC press, 2006.

[43] M. Conrad and I. Fey. Testing automotive control software. In *Automotive Embedded Systems Handbook*. CRC Press, 2008.

[44] J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Păsăreanu, R. Bby, and H. Zheng. Bandera: Extracting finite-state models from java source code. In *Software Engineering, 2000. Proceedings of the 2000 International Conference on*, pages 439–448. IEEE, 2000.

[45] T. Dang. *Verification and Synthesis of Hybrid Systems*. PhD thesis, INPG, 2000.

[46] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[47] Y. Deng, A. Rajhans, and A. A. Julius. Strong: A trajectory-based verification toolbox for hybrid systems. In *Quantitative Evaluation of Systems*, pages 165–168. Springer, 2013.

[48] J. V. Deshmukh, R. Majumdar, and V. S. Prabhu. Quantifying conformance using the skorokhod metric. In *International Conference on Computer Aided Verification*, pages 234–250. Springer, 2015.

[49] A. Deshpande, A. Gollu, and P. Varaiya. Shift: A formalism and a programming language for dynamic networks of hybrid automata. In P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems*, volume 1273 of *LNCS*, pages 113–133. Springer, 1996.

[50] B. Di Giampaolo, S. La Torre, and M. Napoli. Parametric metric interval temporal logic. In A.-H. Dediu, H. Fernau, and C. Martin-Vide, editors, *Language and Automata Theory and Applications*, volume 6031 of *LNCS*, pages 249–260. Springer, 2010.

[51] B. Di Giampaolo, S. La Torre, and M. Napoli. Parametric metric interval temporal logic. In *Language and Automata Theory and Applications*, pages 249–260. Springer, 2010.

[52] A. Dokhanchi, B. Hoxha, and G. Fainekos. On-line monitoring for temporal logic robustness. In *Runtime Verification*, volume 8734 of *LNCS*, pages 231–246. Springer, 2014.

[53] A. Dokhanchi, B. Hoxha, and G. Fainekos. Metric interval temporal logic specification elicitation and debugging. In *Formal Methods and Models for Codesign (MEMOCODE), 2015 ACM/IEEE International Conference on*, pages 70–79. IEEE, 2015.

[54] A. Dokhanchi, B. Hoxha, C. E. Tuncali, and G. Fainekos. An efficient algorithm for monitoring practical tptl specifications. In *Formal Methods and Models for System Design (MEMOCODE), 2016 ACM/IEEE International Conference on*, pages 184–193. IEEE, 2016.

[55] A. Dokhanchi, A. Zutshi, R. T. Sriniva, S. Sankaranarayanan, and G. Fainekos. Requirements driven falsification with coverage metrics. In *Proceedings of the 12th International Conference on Embedded Software*, pages 31–40. IEEE Press, 2015.

[56] A. Donze. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 167–170. Springer, 2010.

[57] A. Donze and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modelling and Analysis of Timed Systems*, volume 6246 of *LNCS*. Springer, 2010.

[58] V. D'silva, D. Kroening, and G. Weissenbacher. A survey of automated techniques for formal software verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1165–1178, 2008.

[59] P. S. Duggirala, S. Mitra, and M. Viswanathan. Verification of annotated models from executions. In *Proc. of the Eleventh ACM Int. Conf. on Embedded Software*, page 26. IEEE Press, 2013.

[60] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2e2: A verification tool for stateflow models. In *TACAS*, pages 68–82, 2015.

[61] D. L. Dvorak and M. Lyu. Nasa study on flight software complexity. *NASA office of chief engineer*, 2009.

[62] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan. 2003.

[63] F. Fages and A. Rizk. On temporal logic constraint solving for analyzing numerical data time series. *Theor. Comput. Sci.*, 408(1):55–65, 2008.

[64] G. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using s-taliro. In *Proceedings of the American Control Conference*, 2012.

[65] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications. In *Formal Approaches to Testing and Runtime Verification*, volume 4262 of *LNCS*, pages 178–192. Springer, 2006.

[66] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[67] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn. Refining the in-parameter-order strategy for constructing covering arrays. *Journal of Research of the National Institute of Standards and Technology*, 113(5):287, 2008.

[68] G. Frehse, C. L. Guernic, A. Donz, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*, 2011.

[69] S. Gao. *Computable analysis, decision procedures, and hybrid automata: a new framework for the formal verification of cyber-physical systems*. PhD thesis, PhD thesis, Carnegie Mellon University, 2012.

[70] S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*, pages 208–214. Springer, 2013.

[71] A. Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, volume 3414 of *LNCS*, pages 291–305, 2005.

[72] A. Gurfinkel, B. Devereux, and M. Chechik. Model exploration with temporal logic query checking. *SIGSOFT Softw. Eng. Notes*, 27(6):139–148, 2002.

[73] A. Hartman. Software and hardware testing using combinatorial covering suites. In *Graph theory, combinatorics and algorithms*, pages 237–266. Springer, 2005.

[74] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[75] M. Hinchey, C. Wang, and M. Josh. Formal methods for system/software engineering: Nasa & army experiences. `https://www.nasa.gov/sites/default/files/585641main_FormalMethodsforSystemSoftwareEngineering.pdf`. Accessed: 2017-07-04.

[76] G. J. Holzmann. The model checker spin. *IEEE Transactions on software engineering*, (5):279–295, 1997.

[77] G. J. Holzmann. The logic of bugs. In *Proc. of the 10th ACM SIGSOFT symp. on Foundations of soft. eng.*, pages 81–87. ACM, 2002.

[78] B. Hoxha, H. Abbas, and G. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2014.

[79] B. Hoxha, H. Abbas, and G. Fainekos. Using s-taliro on industrial size automotive models. *Proc. of Applied Verification for Continuous and Hybrid Systems*, 2014.

[80] B. Hoxha, H. Bach, H. Abbas, A. Dokhanchi, Y. Kobayashi, and G. Fainekos. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *Int. Workshop on Design and Implementation of Formal Tools and Systems*. October 2014.

[81] B. Hoxha, A. Dokhanchi, and G. Fainekos. Mining parametric temporal logic properties in model-based design for cyber-physical systems. *International Journal on Software Tools for Technology Transfer*, pages 1–15, 2017.

[82] B. Hoxha and G. Fainekos. Pareto front exploration for parametric temporal logic specifications of cyber-physical systems. In *Workshop on Monitoring and Testing of Cyber-Physical Systems*, 2016.

[83] B. Hoxha, N. Mavridis, and G. Fainekos. Vispec : A graphical tool for elicitation of mtl requirements. In *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[84] M. Huang, H. Nakada, S. Polavarapu, K. Butts, and I. Kolmanovsky. Rate-based model predictive control of diesel engines. In *7th IFAC Symposium on Advances in Automotive Control*, pages 177 – 182, 2013.

[85] M. Huang, H. Nakada, S. Polavarapu, R. Choroszucha, K. Butts, and I. Kolmanovsky. Towards combining nonlinear and predictive control of diesel engines. In *American Control Conference (ACC), 2013*, pages 2846–2853. IEEE, 2013.

[86] D. Isbell, M. Hardin, and J. Underwood. Mars climate orbiter team finds likely cause of loss. *NASA news release*, 1999.

[87] ISO/IEC/IEEE 29148:2011(E). International Standard - Systems and software engineering - Life cycle processes - Requirements engineering. pages 1–94, Dec 2011.

[88] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, and A. Platzer. A formally verified hybrid system for the next-generation airborne collision avoidance system. In *TACAS*, volume 9035, pages 21–36, 2015.

[89] Z. Jiang, M. Pajic, and R. Mangharam. Cyber-physical modeling of implantable cardiac medical devices. *Proceedings of the IEEE*, 100(1):122–137, 2012.

[90] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid systems: computation and control*, pages 43–52. ACM, 2013.

[91] J. Kapinski, J. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-guided approaches for verification of automotive powertrain control systems. In *American Control Conference (ACC), 2015*, pages 4086–4095. IEEE, 2015.

[92] N. Khakpour and M. R. Mousavi. Notions of conformance testing for cyber-physical systems: Overview and roadmap. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 42. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[93] E. S. Kim, M. Arcak, and S. A. Seshia. Directed specifications and assumption mining for monotone dynamical systems. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 21–30. ACM, 2016.

[94] G. Klein, J. Andronick, K. Elphinstone, G. Heiser, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, et al. sel4: formal verification of an operating-system kernel. *Communications of the ACM*, 53(6):107–115, 2010.

[95] S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *TACAS*, volume 15, pages 200–205, 2015.

[96] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282. ACM, 2014.

[97] P. Koopman. *Better Embedded System Software*. Drumnadrochit Education LLC, 2010.

[98] Y. Kouskoulas, D. W. Renshaw, A. Platzer, and P. Kazanzides. Certifying the safe design of a virtual fixture control algorithm for a surgical robot. In C. Belta and F. Ivancic, editors, *Hybrid Systems: Computation and Control (part of CPS Week 2013), HSCC'13, Philadelphia, PA, USA, April 8-13, 2013*, pages 263–272. ACM, 2013.

[99] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[100] H. Kugler, D. Harel, A. Pnueli, Y. Lu, and Y. Bontemps. Temporal logic for scenario-based specifications. In *Tools and Alg. for the Construction and Analysis of Systems*, pages 445–460. Springer, 2005.

[101] D. R. Kuhn and R. Chandramouli. Cost effective uses of formal methods in verification and validation. In *Foundations Verification and Validation Workshop*, 2002.

[102] D. R. Kuhn, D. R. Wallace, and A. M. Gallo. Software fault interactions and implications for software testing. *IEEE transactions on software engineering*, 30(6):418–421, 2004.

[103] J. Legriel, C. Le Guernic, S. Cotton, and O. Maler. Approximating the pareto front of multi-criteria optimization problems. In *TACAS*, pages 69–83. Springer, 2010.

[104] N. G. Leveson and C. S. Turner. An investigation of the therac-25 accidents. *Computer*, 26(7):18–41, 1993.

[105] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38(1):89–105, 2015.

[106] J.-L. Lions, L. Lbeck, J.-L. Fauquembergue, G. Kahn, W. Kubbat, S. Levedag, L. Mazzini, D. Merle, and C. O'Halloran. Ariane 5, flight 501 failure, report by the inquiry board. Technical report, CNES, July 1996.

[107] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, volume 3253 of *LNCS*, pages 152–166, 2004.

[108] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa. Robotic surgery. *Robotics & Automation Magazine, IEEE*, 18(3):24–32, 2011.

[109] R. H. Myers, D. C. Montgomery, and C. M. Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*. Wiley & Sons.

[110] R. Natella, D. Cotroneo, and H. S. Madeira. Assessing dependability with software fault injection: A survey. *ACM Computing Surveys (CSUR)*, 48(3):44, 2016.

[111] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancic, A. Gupta, and G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 211–220. ACM Press, 2010.

[112] W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, 38(3):209–272, 2002.

[113] D. of Defense. Dod modeling and simulation (m&s) verification, validation, and accreditation (vv&a), December 2009.

[114] M. Osch. Hybrid input-output conformance and test generation. In K. Havelund, M. Nez, G. Rou, and B. Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification*, volume 4262 of *Lecture Notes in Computer Science*, pages 70–84. Springer Berlin Heidelberg, 2006.

[115] D. G. V. Oss. Computer software in civil aircraft. In *Digital Avionics Systems Conference, 1991. Proceedings., IEEE/AIAA 10th*, pages 324–330. IEEE, 1991.

[116] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Falsification of ltl safety properties in hybrid systems. In *Proc. of the Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 5505 of *LNCS*, pages 368 – 382, 2009.

[117] A. Platzer and E. M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design*, 35(1):98–120, 2009.

[118] A. Platzer and J.-D. Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In A. Armando, P. Baumgartner, and G. Dowek, editors, *International Joint Conference on Automated Reasoning*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[119] J.-D. Quesel, S. Mitsch, S. Loos, N. Aréchiga, and A. Platzer. How to model and prove hybrid systems with keymaera: A tutorial on safety. *International Journal on Software Tools for Technology Transfer*, 18(1):67, 2016.

[120] N. Ramdani, N. Meslem, and Y. Candau. Reachability of uncertain nonlinear systems using a nonlinear hybridization. In *HSCC '08: Proceedings of the 11th international workshop on Hybrid Systems*, pages 415–428, Berlin, Heidelberg, 2008. Springer-Verlag.

[121] A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *International Conference on Computational Methods in Systems Biology*, number 5307 in LNCS, pages 251–268. Springer, 2008.

[122] H. Roehm, T. Heinz, and E. C. Mayer. Stlinspector: Stl validation with guarantees. In *Proceedings of the 29th International Conference on Computer Aided Verification*, 2017.

[123] S-TaLiRo: Temporal Logic Falsification Of Cyber-Physical Systems. `https://sites.google.com/a/asu.edu/s-taliro/s-taliro`, 2013. [Online; accessed April-2014].

[124] A. Saadat. Defect information report. `http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.pdf`, 2014.

[125] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *ACM International Conference on Hybrid Systems: Computation and Control*, 2012.

[126] S. Sankaranarayanan and G. Fainekos. Simulating insulin infusion pump risks by in-silico modeling of the insulin-glucose regulatory system. In *International Conference on Computational Methods in Systems Biology*, 2012. [To Appear].

[127] S. Sankaranarayanan, H. Homaei, and C. Lewis. Model-based dependability analysis of programmable drug infusion pumps. In *Formal modeling and analysis of timed systems*, pages 317–334. Springer, 2011.

[128] C. Scott. Industry-nominated technology brakethroughs of nsf industry/university cooperative research centers. *Washington DC: National Science Foundation*, 2012. Online at: `http://faculty.washington.edu/scottcs/NSF/2012/NSF_Compendium_2012-WEB.pdf`.

[129] C. Scott. Industry-nominated technology brakethroughs of nsf industry/university cooperative research centers. *Washington DC: National Science Foundation*, 2014. Online at: `http://faculty.washington.edu/scottcs/NSF/2014/NSF_Compendium_2014.pdf`.

[130] B. I. Silva and B. H. Krogh. Formal verification of hybrid systems using CheckMate: a case study. In *Proceedings of the American Control Conference*, volume 3, pages 1679 – 1683, June 2000.

[131] Simuquest. Enginuity. `http://www.simuquest.com/products/enginuity`. Accessed: 2013-10-14.

[132] A. Singh, C. Ramakrishnan, and S. A. Smolka. Query-based model checking of ad hoc network protocols. In *Proceedings of Concurrency Theory*, pages 603–619. Springer, 2009.

[133] M. H. Smith, G. J. Holzmann, and K. Etessami. Events and constraints: A graphical editor for capturing logic requirements of programs. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 14–22. IEEE, 2001.

[134] S. Srinivas, R. Kermani, K. Kim, Y. Kobayashi, and G. Fainekos. A graphical language for ltl motion and mission planning. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 704–709. IEEE, 2013.

[135] L. Tan, J. Kim, O. Sokolsky, and I. Lee. Model-based testing and monitoring for hybrid embedded systems. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration*, pages 487–492, 2004.

[136] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake. A classification and survey of analysis strategies for software product lines. *ACM Computing Surveys (CSUR)*, 47(1):6, 2014.

[137] J. Tretmans. Testing concurrent systems: A formal approach. In *CONCUR 1999 Concurrency Theory*, pages 46–65. Springer, 1999.

[138] S. Tripakis and T. Dang. *Model-Based Design for Embedded Systems*, chapter Modeling, Verification and Testing using Timed and Hybrid Automata, pages 383–436. CRC Press, 2009.

[139] R. Vinter, M. Loomes, and D. Kornbrot. Applying software metrics to formal specifications: A cognitive approach. In *Software Metrics Symposium, 1998. Metrics 1998. Proceedings. Fifth International*, pages 216–223. IEEE, 1998.

[140] A. Wasylkowski and A. Zeller. Mining temporal specifications from object usage. In *24th IEEE/ACM International Conference on Automated Software Engineering*, 2009.

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

[142] T. Wongpiromsarn, S. Mitra, A. Lamperski, and R. M. Murray. Verification of periodically controlled hybrid systems: Application to an autonomous vehicle. *ACM Trans. Embed. Comput. Syst.*, 11(S2):53:1–53:24, Aug. 2012.

[143] H. Yang, B. Hoxha, and G. Fainekos. Querying parametric temporal logic properties on embedded systems. In *Int. Conference on Testing Software and Systems*, 2012.

[144] B. Yordanov, J. Tmov, I. ern, J. Barnat, and C. Belta. Formal analysis of piecewise affine systems through formula-guided refinement. *Automatica*, 49(1):261 – 266, 2013.

[145] P. Zhang, B. Li, and L. Grunske. Timed property sequence chart. *Journal of Systems and Software*, 83(3):371–390, 2010.

[146] P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pages 243–252, 2010.

[147] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski. Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, page 5. ACM, 2014.

[148] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski. A trajectory splicing approach to concretizing counterexamples for hybrid systems. In *IEEE Conference on Decision and Control*, 2013.