Pós-Graduação em Ciência da Computação

HUGO LEONARDO DA SILVA ARAUJO

A process for sound conformance testing of Cyber-Physical Systems

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

HUGO LEONARDO DA SILVA ARAUJO

A process for sound conformance testing of Cyber-Physical Systems

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: Prof. Dr. Augusto Cezar Alves Sampaio

CO-ORIENTADOR: Prof. Dr. Gustavo Henrique Porto de Carvalho

RECIFE
2017

Catalogação na fonte

Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A663p Araujo, Hugo Leonardo da Silva

A process for sound conformance testing of cyber-physical systems / Hugo Leonardo da Silva Araujo. – 2017.

63 f.: il., fig., tab.

Orientador: Augusto Cezar Alves Sampaio.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.

Inclui referências.

1. Engenharia de software. 2. Métodos formais. I. Carvalho, Gustavo Henrique Porto de (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2017-245

Hugo Leonardo da Silva Araujo

A Process for Sound Conformance Testing of Cyber-Physical Systems

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 17/08/2017

BANCA EXAMINADORA

_____

Prof. Dr. Carlos Alexandre Barros de Mello
Centro de Informática / UFPE

_____

Prof. Dr. Marcelo Vinicius Medeiros de Oliveira
Departamento de Informática e Matemática Aplicada / UFRN

_____

Prof. Dr. Augusto Cezar Alves Sampaio
Centro de Informática / UFPE

Acknowledgements

Firstly, I would like to thank my family for all the support that has been given throughout my life and academic journey. All those years were filled with love, dedication and partnership that helped moulding my future. I thank them for the incentive to challenge myself and for their own life history of overcoming difficulties that served as inspiration to me.

I would also like to give deep thanks to my advisor and co-advisor, Augusto Sampaio and Gustavo Carvalho, for the excellent support and personal attention that was offered during this work. They were the anchors who provided me with stability and knowledge when the research difficulties appeared. I'm also very grateful to Mohammad Mousavi who invited me for a brief stay in Sweden. The joint work we performed ended up being essential for my research.

Thanks to Embraer and Braulio Horta, for providing me with research problems and for the knowledge we have exchanged. Thanks to Mathias Althoff and Roehm Hendrik for helping me with CORA and to Arend Aerts and Georgios Fainekos for the meetings we held. I also thank professors Enrique Droguett and Carlos Mello for their inputs on some of my research problems.

Furthermore, I express gratitude to the Centro de Informática (UFPE) and all its employees for the education they have been providing, which made this dissertation possible. The knowledge I acquired throughout the years has proved invaluable to me and I might not have had the opportunity to perform this work if were not for this centre. Last but not least, thanks to the financial support provided by CNPq and Motorola.

Thank you all who directly or indirectly contributed to this work.

Abstract

The term Hybrid System is used to describe a modelling formalism of systems that combine discrete and continuous aspects; for instance, a system where a controller (discrete component) is connected to a physical system (continuous component). Systems that encompass tightly integrated digital and physical components and deal with spatial and temporal metrics, besides involving human interaction, are known as Cyber-Physical Systems (CPS). Model-based testing of CPSs is a recent subject in the literature, and it is still being actively researched and developed. The analysis of CPSs is usually complex due to their multidisciplinary nature, with such systems dealing with aspects of different subject areas such as computer science, physics and control systems. In this work, we propose a process for sound conformance testing of cyber-physical systems. The main goal of this process is to provide a practical and semi-automatic solution to testing CPSs. Some of the steps of our process were mechanized through the use of a prototype tool that we have developed. This project was conceived during the literature review in our research when we realized the absence of a structured process with systematic steps for conformance testing of CPSs. We first focused on studying the existing conformance testing strategies of hybrid systems and settled on working with (τ, ε)-conformance relation. In this conformance notion, the outputs of both specification and implementation models are compared under the same input stimuli. It makes use of temporal (τ) and spatial (ε) margins of error to determine if the output behaviours are close enough to each other. In conformance verification strategies based on this relation, an issue related to soundness was brought to our attention, which made us shift our focus to solve this problem through reachability analysis. We noticed that the sampling rate, used to observe the system behaviour at discrete points, was closely related to the soundness problem identified. This motivated the definition and partial automation of a process to support conformance testing of CPSs. The proposed process involves five steps: (i) automatic sampling rate computation; (ii) margins of error definition (temporal and spatial); (iii) performing reachability analysis to obtain sound verdicts; (iv) conformance testing (test generation, test execution and verdict attainment); (v) result analysis and parameters tuning. Additionally, we have performed an empirical analysis to shown how our approach can be used in practice describing a few usage scenarios as well as implementing two case studies: a combustion engine controller and a pneumatic suspension system.

Keywords: Cyber-physical systems. Hybrid systems. Conformance testing. Reachability analysis

Resumo

O termo Sistema Híbrido é usado para descrever sistemas que combinam elementos contínuos e discretos; por exemplo, um sistema em que um controlador digital (elemento discreto) está conectado à um sistema físico (elemento contínuo). Sistemas desse tipo, que envolvem componentes físicos e digitais altamente integrados e que lidam com métricas temporais e espaciais, além de envolverem interação humana, são conhecidos como sistemas ciber-físicos (SCF). Neste contexto, o uso de técnicas de teste baseadas em modelos (do inglês, Model Based Testing) em sistemas ciber-físicos é um assunto recente e está sendo ativamente pesquisado e desenvolvido. A análise de SCFs é de alta complexidade devido à multidisciplinaridade de tais sistemas, que combinam aspectos de diversas áreas como ciência da computação, física e sistemas de controle. Neste trabalho, nós propomos um processo para teste de conformidade de sistemas ciber-físicos. O objetivo desse processo é oferecer uma abordagem prática que provê uma solução semi-automática para o teste de SCFs. Algumas etapas do processo foram mecanizadas a partir de um protótipo de ferramenta desenvolvido. Este projeto foi concebido durante a revisão da literatura, quando percebeu-se a falta de um processo estruturado com passos sistematizados para a realização de testes de conformidade em SCFs. Em primeiro plano, a pesquisa foi direcionada para o estudo das relações de conformidade existentes, o que resultou em um foco maior na relação *(τ, ε)-conformance*. Nesta relação de conformidade, as saídas dos modelos da especificação e da implementação são comparadas sob o mesmo estímulo de entrada. Ela faz uso de margens de erro temporais (τ) e espaciais (ε) para determinar se o comportamento de saída dos modelos estão suficientemente próximas. Em estratégias de verificação de conformidade com base nesta relação, um problema relacionado à propriedade de inconsistência (*soundness*) da relação foi percebido, o que fez com que o foco da pesquisa fosse voltado a resolver esse problema via análise de alcançabilidade. Identificou-se que a taxa de amostragem, utilizada para observar o comportamento do sistema em pontos discretos, estava fortemente relacionada ao problema de inconsistência encontrado. Isto motivou a definição e automação parcial de um processo para apoiar o teste de conformidade de SCFs. O processo é organizado em cinco passos: (i) computação automática da taxa de amostragem; (ii) definição das margens de erro temporais e espaciais (τ e ε, respectivamente); (iii) execução da análise de alcançabilidade com o objetivo de assegurar a consistência da análise; (iv) teste de conformidade (geração e execução dos testes e obtenção do veredito); (v) análise dos resultados e ajuste de parâmetros. Além disso, foi realizada uma análise empírica para mostrar como essa abordagem pode ser usada na prática. Descrevemos alguns cenários de uso e dois estudos de caso: um controlador de um motor de combustão e um sistema de suspensão pneumática.

Palavras-chave: Sistemas ciber-físicos. Sistemas híbridos. Teste de conformidade. Análise de alcançabilidade

List of Figures

List of Tables

List of Symbols

$\mapsto$	Maplet (to denote ordered pairs)
:	Set membership
$\times$	Cartesian product
$\dot{x}$	First derivative of variable x
$\lvert n \rvert$	Modulo of a number n
$\lVert v \rVert$	Norm of vector v (*e.g.* Euclidean norm)
$\uplus$	Disjoint union

Contents

1 INTRODUCTION

The incorporation of computational systems in physical environments, also known as Cyber-Physical Systems (CPSs), has been deemed necessary in modern products such as automobiles and airplanes [MZ16]. In order to model the continuous and discrete dynamics often present in CPSs, we make use of hybrid systems models. As an example of a hybrid system, we have a system where sensors feed input signals to a digital controller (discrete component) attached to a physical system (continuous component) that outputs different continuous signals.

Such systems are considerably complex since they are usually part of multidisciplinary critical systems. It is not uncommon for a system component to deal with aspects of different subject areas such as computer science, physics and control systems [HDS13]. Thus, there is a natural concern regarding the safety and reliability of such systems, which warrants the need for further research directed to increasing their credibility.

To this end, Model-Based Testing techniques (MBT) can contribute with an important role in the verification of these systems by providing precise mathematical assurances [Tre08]. For instance, one can design a test strategy based on a mathematical relation that decides whether the System Under Test (SUT) behaves as expected. These mathematical relations are also known as conformance relations, and their goal is to verify compatibility between the specification and the resulting implementation.

A test strategy is said to be sound if when a test fails, it necessarily means that the SUT does not conform to the specification considering the adopted conformance relation. The work described in [AHF+14] introduces a practical test strategy for cyber-physical systems based on the (τ, ε)-conformance relation, which allows for margins of error. In summary, it compares the output signals of both implementation and specification models under the same input stimuli, and decides whether the signals are close enough. In other words, the outputs do not need to be identical. The parametrised margins of error, namely τ and ε, allow for slight temporal and spatial variation.

It is important to note that, in practice, it is not feasible to check all the points in a continuous signal, since they are infinite. Hence, due to this intrinsic characteristic of cyber-physical systems, it is often required to sample the system. The sampling process results in dealing with only a few selected points in the curve. The lower the sampling rate, the more sparse the points are, and the higher is the data loss. Thus, in [MM16a], the author brings to light that a test strategy based on the (τ, ε)-conformance relation might yield unsound verdicts (i.e., yield a fail verdict for conforming implementations) due to the sampling process. In order to restore soundness, the authors consider an additional margin of error to deal with the sampling loss.

Despite the aforementioned contributions, applying a test strategy, such as the one based on the (τ, ε)-conformance relation, to analyse the behaviour of a CPS, is not a straightforward task. Typically, one needs to decide on the value of different testing parameters (*e.g.*, τ, ε, and the sampling rate), besides taking into account the impact of these decisions.

Therefore, in this work we propose a process for sound conformance testing of cyber-

physical systems, particularly considering the (τ, ε)-conformance relation.

1.1 OBJECTIVES AND CONTRIBUTIONS

The main goal of this work is to propose a sound process for conformance testing of cyber-physical systems. We make use of the results we have obtained in [ACS+17a] and embody them into a potentially scalable and systematic process for checking conformance of CPSs. An initial version of the process we propose first appeared in [ACS+17b].

The scientific and technological contributions achieved by this work are the following:

- Proposal of a process for sound conformance testing of CPSs;
- Introduction of an automatic strategy for defining a sampling rate that minimizes sampling loss;
- Usage of reachability analysis in order to compute an additional margin of error that guarantees soundness of the testing strategy;
- Integration of tools for conformance testing and reachability analysis in order to support the proposed process;
- Application of the proposed process concerning three examples of CPSs: a thermostat (running example), an automotive air-fuel ratio control system, and an automotive pneumatic suspension system.

The first main contribution is the definition of the process itself. The process receives the specification and implementation models as input and the result is a test suite that only yields sound verdicts. The process is organized into five steps: (i) automatic sampling rate computation; (ii) margins of error definition (temporal and spatial); (iii) performing reachability analysis to obtain sound verdicts; (iv) conformance testing (test generation, test execution and verdict attainment); (v) result analysis and parameters tuning, which includes verifying whether the results are acceptable and revise definitions accordingly. We have also shown how this process can be used discussing possible scenarios, as well as considering two case studies, besides a running example.

The second main contribution is the implementation of the solution proposed in [MM16a]. We have designed a practical solution involving reachability analysis [Alt10] and applied our research results, besides extending an existing prototype tool that already considers the (τ,ε)-conformance relation in its test strategy.

1.2 DISSERTATION STRUCTURE

This dissertation is organized as follows:

- Chapter 2 provides the necessary background information required to understand the work we present here.

- Chapter 3 presents our process for sound conformance testing of cyber-physical systems.
- Chapter 4 introduces the tools we have used and developed in order to mechanise our testing strategy.
- Chapter 5 gives a summary of the results we have achieved and considers related work. Furthermore, it presents the next steps in our research agenda.

2 BACKGROUND

In this chapter, we present the necessary background information required to understand the work we present here. Section 2.1 gives a brief introduction to formal methods whilst section 2.2 focus on explaining Cyber-Physical Systems (CPSs) and how we formally define them. Finally, section 2.3.1 presents the notion of conformance that we use in CPSs throughout this work.

2.1 MODEL-BASED TESTING

Formal methods consist in the application of theoretical fundamentals that aim to aid hardware and software engineers to build more reliable systems supported by mathematically-based techniques. The use of formal methods is driven by the assurance that proper mathematical analysis provides robustness and reliability for the design.

One of the technical challenges in formal verification lies in handling large search spaces. However, when used appropriately, such techniques can be fast, automatic and identify faults through counterexamples, which usually represent subtle errors in design.

Model-based testing (MBT) is a notion that originates from software and system testing [UL10, BJK$^+$05], which makes use of formal models in order to generate and execute test cases and, thus, verifying compliance with system requirements. One of the key purposes of such a notion is to facilitate the automation of the testing process. Figure 2.1 depicts some common elements of an MBT approach. The test suite is generated by a tool from the formal specification model and the requirements are taking into account when creating the model. Different types of models have been used, such as timed automata [AD94], labelled transitions systems [Tre96] and process algebras [CFMS12]. The tests are then executed and the verdict (fail, pass or inconclusive) is used to make adjustments to the System Under Test (SUT). Pass and fail mean that the testing process found evidence that shows whether the test was successful or not, respectively. Inconclusive means that no such evidence was found.

Conformance testing is a variation of MBT where the key idea is to verify whether the SUT conforms to a given specification considering a mathematical relation, also known as a conformance relation. Moreover, when considering an MBT framework, two concepts are important to evaluate the quality of the testing framework, namely soundness and completeness. The former states that if a SUT is correct, a pass verdict is achieved for any generated test suite. The latter states that, for incorrect models, there will always exist a generated test case that detects the failure. The last aspect, completeness, is a hard property to achieve in practice, since it might imply in generating infinite test suites. Several frameworks and tools exist to perform MBT for software models. For example, in the case of LTSs, a sound and complete testing framework based on a conformance relation named ‘ioco’ (Input-Output COnformance) is introduced in [Tre08].

In our context of Cyber-Physical Systems, model-based testing is a relatively recent subject in the literature [ARM16, DDD+15, HBA+14, KM15a] and it is still being actively researched and developed. Hence, in order to apply MBT on CPSs in this work, state-of-the-art techniques are adopted and further developed.

In what follows, we first discuss how CPSs can be represented as hybrid automata (Section 2.2). Afterwards, we focus on one of the existing conformance testing strategies for CPSs, namely (τ, ε)-conformance, which is adopted in this work and explained in more details (Section 2.3.1).

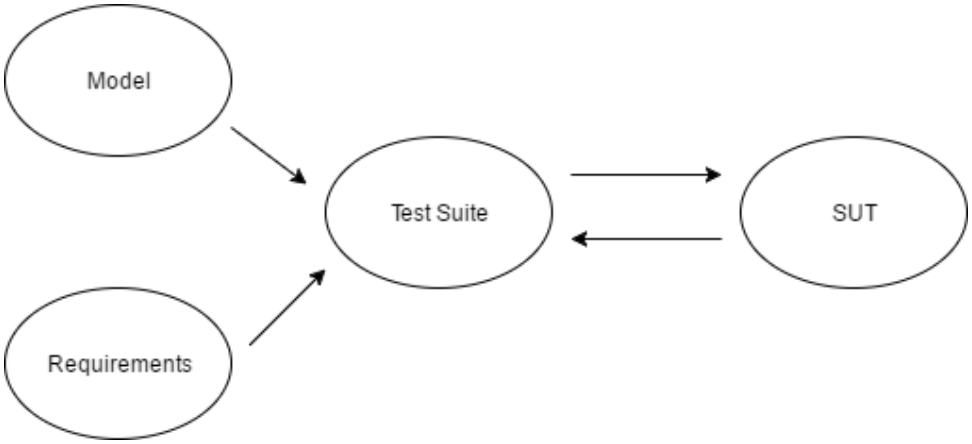


Figure 2.1: Model-based testing overview

2.2 CYBER-PHYSICAL SYSTEMS

The term Cyber-Physical System (CPS) represents a combination of discrete and continuous dynamics, often found in systems where a digital controller unit (discrete) is connected with some physical system (continuous). Particularly, a CPS might involve complex networks of these elements. also including human interaction. In order to model the continuous and discrete dynamics often present in CPSs, we make use of hybrid systems models [ACH+95a].

Such systems require special attention since they are usually part of multidisciplinary critical systems. It is not uncommon for a component to deal with aspects of different subject areas such as computer science, physics and control systems [HDS13]. Many classes or representations of hybrid systems can be found [DSHL+09], due to the wide range of modeling formalisms in these areas. In this work, we use hybrid automata [ACHH93] to model CPSs since it is a well-established and solid formalism, with an intuitive semantics, and that can be processed by several tools.

2.2.1 Hybrid automata

Hybrid Automata can be seen as an extension of discrete event models, more specifically finite timed automata. Guards, reset maps, invariants and specific dynamics for each location are added to these kind of models, in order to support continuous dynamics.

Definition 1 (Hybrid Automata [GST09]). *A hybrid automata is defined as a tuple (Loc, V, (l_0,v_0), $\rightarrow$, 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 and output variables, respectively;*
- *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 jump may be performed, and*
  - *$Reset(V) = \bigcup_{V' \subseteq V} Val(V')$ is the set of value assignments to the variables in V after the jump;*
- *$I : Loc \rightarrow \mathcal{B}(V)$ 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.*

We denote the set of all hybrid automata by $\mathbb{H}$. Following standard notation, we write $l \xrightarrow{g,r} l'$ to denote $(l, g, r, l') \in \rightarrow$.

Locations are discrete states that operate independently and each one can be viewed as a purely continuous system. Furthermore, the continuous behaviour of the whole hybrid system is captured by the valuation of a set V of continuous variables. We assume that V is partitioned into disjoint sets of input variables, denoted by V_I, and output variables, denoted by V_O.

A jump represents a change in the current operating location. To perform a jump, the transition guard always needs to be true. In general, if the guard is true, but the location invariant also holds, the jump might not be performed; this results in a non-deterministic specification. Differently, if the invariant becomes false and the guard is true, the jump must be performed. There is also the situation where both the location invariant and the corresponding transition are false; this scenario is assumed not to happen. Moreover, a jump is also an immediate action, which does not require time to pass. During a jump event, the valuation of the continuous variables can be reset.

2.2.2 Running example

We use the model of a thermostat [vO09a] as our running example. Consider a thermostat that has two modes, ON and OFF. The thermostat controls a heater and the temperature of the room changes towards the temperature represented by the inputs $h(t)$ and $i(t)$. For instance, if h and i are two constant signals equal to 20 and 0 respectively, when the thermostat is in mode ON, the temperature of the room would increase towards 20 degrees. Analogously, when it is in mode OFF, the temperature would decrease towards 0 degrees.

The main goal of the thermostat is to keep the room temperature at 10 degrees. However, to avoid switching the radiator on an off all the time, there are two thresholds for the minimum and

maximum values where the thermostat does not attempt to turn the heater on or off until they are reached. Therefore, considering the previously mentioned values for $h(t)$ and $i(t)$ (20 and 0, respectively), the thermostat only switches to mode ON when the temperature falls below 2 degrees and to OFF when the temperature rises above 18 degrees.

In Figure 2.2, the hybrid model of the thermostat example is shown, which contains two discrete states representing its modes, namely, q_0 (ON) and q_1 (OFF). Moreover, the dynamics of each mode represents the temperature behaviour in the room with the variable $x \in \mathbb{R}$ modeling temperature. As previously explained, the temperature is regulated between 2 and 18 degrees, hence the corresponding guard conditions consider these values. .

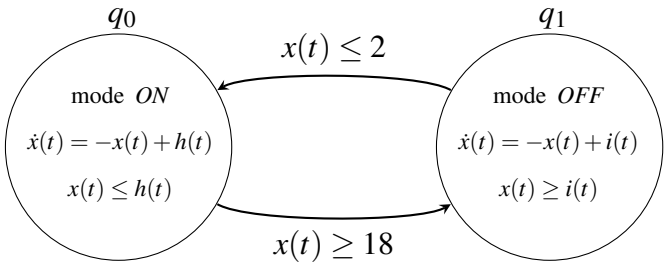


Figure 2.2: Hybrid automaton of the thermostat

Example 1. *Figure 2.2 shows the hybrid automaton of the thermostat with* $Loc = \{ON, OFF\}$, $V_I = \{h, i\}$, $V_O = \{x\}$, $(l_0, v_0) = (ON, 5)$, $I(ON) = x(t) \leq h(t)$, $\rightarrow = \{(ON, x(t) \geq 18, \{\}, OFF)$, $(OFF, x(t) \leq 2, \{\}, ON)\}$, $I(OFF) = x(t) \geq i(t)$, $F(ON) = \dot{x}(t) = -x(t) + h(t)$, *and* $F(OFF) = \dot{x}(t) = -x(t) + i(t)$.

Figure 2.3, reproduced from [MM16a], shows the output signal of the temperature for the thermostat example.

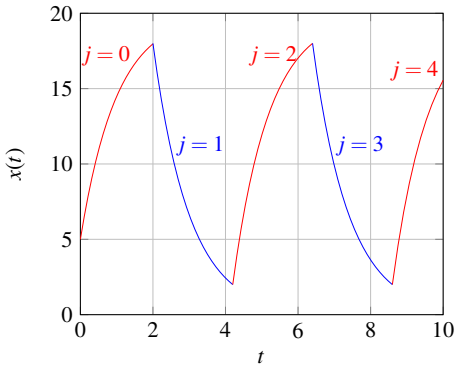


Figure 2.3: A sample of the continuous dynamics of the thermostat (taken from [MM16a])

2.2.3 Formal testing approaches to CPS

Testing and verifying the correctness of all physical and cyber components of a complex CPS system present a big challenge. A typical research problem is the different approaches to

system modeling and that comes from different areas such as computer science, control theory and physics. In [Alu11], the author presents an overview of key research works that have been devoted to formal verification of CPSs.

Since conformance testing based on hybrid-systems models is a relatively recent subject, it still requires more work on theoretical and practical aspects. A recent survey presented in [AIH15] concludes that existing model checking and other formal techniques are insufficient to meet CPS application needs. According to the study participants, there is still a gap in language between formal models of computing and communications and models of physics that makes the joint application challenging.

Model verification can be performed by providing the modelled system with test inputs and evaluating the model output behaviour. Moreover, assuming that the model represents the overall desired system behaviour, system verification is performed by providing the same test inputs to both models and evaluating their respective output behaviour based on a conformance notion.

In what follows, we present the conformance notion proposed in [AHF+14], which we deem to be one of the main existing proposals for a formal notion of conformance. The (τ, ε)-conformance notion is, to our knowledge, the most concrete formal notion in both theoretical and practical aspects, and the one we use in our approach. Thus, in what follows, we formally describe this notion and its basic concepts. Later, in Chapter 5, we present other conformance notions as related works. Furthermore, a comparison between these notions of conformance can be found in [KM15a].

2.3 (τ, ε)-CONFORMANCE NOTION

The authors of [AHF+14, AMF14] propose a conformance relation based on the output behaviour of a system specification and implementation models. In Figure 2.4, an overview of this conformance notion is depicted. Briefly speaking, under the same input stimuli u, that is given for both specification model (u_M) and implementation (u_I), the difference in the output behaviour of both systems (y_M and y_I) is analysed. That is, a distance metric is used to verify if output behaviours are close enough to each other.

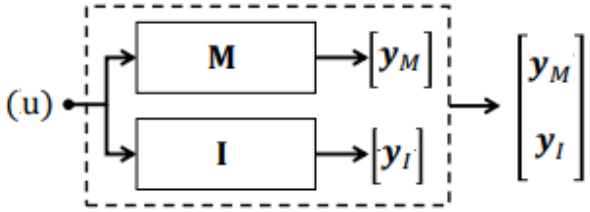


Figure 2.4: (τ, ε)-conformance concept (taken from [AHF+14])

The closeness notion that is used to determine how close two output signals are, in terms of valuation, is based on a maximum margin of error ε. However, this notion not does not require the output signals to behave in exact synchronization. In practice, due to un-modelled physical phenomena such as transport delays or mechanical backlash, implementation behaviour often

deviates in time with respect to the model [AMF14]. Thus, if this notion did not take into consideration these time shifts, that are natural occurrence in CPSs, it could result in undesired non-conformance verdicts in scenarios where slight time deviations are acceptable. Hence, in the (τ,ε)-conformance relation, a maximum temporal error of τ is allowed between the two output signals y_M and y_I to permit such time discrepancies.

2.3.1 Formal definition

In this section, we present the formal definition of the conformance relation that is used in the remainder of our approach, based on [MM16a]. First, we define some preliminary concepts based on the previous notion of hybrid automata (see Definition 1). Then, we elaborate on the formal definition of the conformance relation that we use in our process for sound conformance testing.

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

Definition 2 (Hybrid Time Domain [Abb15])**.** *A hybrid time domain E is a subset of $\mathbb{R}_+ \times \mathbb{N}$ defined as*

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

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

Example 2. *Figure 2.3 shows the output signal for temperature of the thermostat with a Hybrid Time Domain* $E = \{(0,0),...,(2.2,0),(2.2,1),...,(4.4,1),(4.4,2),...,(6.6,2),(6.6,3),...,(8.8,3),(8.8,4),...,(10,4)\}$

The hybrid time domain models the development of a hybrid system regarding the evolution of continuous time intervals ($[t_j, t_{j+1}]$) and discrete jumps (j). In [MM16a], the author combines the notions of hybrid automata and hybrid time domain to define the solution concept.

Definition 3 (Solution [MM16a])**.** *A solution to a hybrid automaton $\mathcal{HA} = (Loc, V, (l_0, v_0), \rightarrow, I, F)$ is a function $s : E \rightarrow Loc \times Val(V)$, 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.*
- *Let V be a set of real-valued variables, Val(V) denotes the set of all valuations of V, where a valuation of V is a function of type $V \rightarrow \mathbb{R}$, which assigns a real number to each variable $v \in V$.*

Example 3. *Considering the Hybrid Time Domain E for the thermostat described in Example 2, a Solution for this example should yield the following values* $s(0,0) = (ON,5)$, $s(2.2,0) = (ON,18)$, $s(2.2,1) = (OFF,18)$, $s(4.4,1) = (OFF,2)$, $s(4.4,2) = (ON,2)$.

The following notion of *trajectory* captures the evolution of system dynamics isolating it from the discrete locations.

Definition 4 (Trajectory [Abb15][1])**.** *Take a hybrid time domain E and a set of variables V. A trajectory over E is a function* $\phi : E \to Val(V)$*, 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* $Trajs(V)$.

Definition 5 (Trajectory for Hybrid Automata [MM16a])**.** *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* $Trajs(\mathcal{HA})$.

The trajectory for the temperature in thermostat example is depicted in Figure 2.3. The notions of trajectory restriction and solution pair are defined in order to distinguish between input and output trajectories in a solution.

Definition 6 (Trajectory Restriction [KM15b])**.** *Consider a set of variables V. The restriction of a valuation* $val \in Val(V)$ *to* $V' \subset V$*, denoted by* $val \downarrow V' \in Val(V')$*, 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* $(\phi \downarrow V')(t,j) = \phi(t,j) \downarrow V'$, $\forall (t,j) \in \mathrm{dom}(\phi)$.

Definition 7 (Solution Pair [Abb15])**.** *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* $y = \phi \downarrow V_O$.

The works we based ours on and the tools we describe and use in later sections of this dissertation do not support non-determinism. Hence, in our work, we focus on deterministic hybrid automata, defined below.

Definition 8 (Deterministic Hybrid Automata [MM16a])**.** *A hybrid automaton* $\mathcal{HA}$ *with the set of solution pairs* Φ *is deterministic if*

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

Finally, the following notion of (τ,ε)-closeness is defined based on the continuous behaviour (solution) associated to a hybrid automaton.

Definition 9 ((τ,ε)-closeness [Abb15][2])**.** *Consider a test duration* $T \in \mathbb{R}_+$*, a maximum number of jumps* $J \in \mathbb{N}$*, and* $\tau, \varepsilon > 0$*; then two trajectories* y_1 *and* y_2 *are said to be* (τ,ε)*-close, denoted by* $y_1 \approx_{(\tau,\varepsilon)} y_2$*, if*

[1] Called *Activity* in [ACH+95b] and *Hybrid Arc* in [Abb15]

[2] Unlike [Abb15], here we allow different jump numbers (i.e. $i \neq j$) in the definition of (τ,ε)-closeness.

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* $\|y_1(t,i) - y_2(s,j)\| \leq \varepsilon$, *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* $\|y_2(t,i) - y_1(s,j)\| \leq \varepsilon$.

Definition 10 (Conformance Relation [Abb15]). *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* $\tau, \varepsilon > 0$, $\mathcal{HA}_2$ conforms to $\mathcal{HA}_1$, *denoted by* $\mathcal{HA}_2 \approx_{(\tau,\varepsilon)} \mathcal{HA}_1$, *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.*

For this conformance relation, we assume that both the specification and the implementation can be modelled in the form of a hybrid automata; this is normally referred to as the *test hyphothesis*.

2.3.2 (τ,ε)-conformance testing

Conformance relations are typically defined in a theoretical framework involving the formal semantics of the specification model and the formal semantics of the SUT. However, the latter is practically impossible to obtain for sufficiently large systems. Hence, in practice, one checks conformance using test cases that are generated from the model and executed on the system under test. The result of this test execution must then be compared with the expected results of the model up to the specified conformance bounds.

As previously stated, based on the same input stimuli, (τ,ε)-conformance testing checks for conformance based on the output behaviour of both the specification model and the implementation. However, it assumes access to all points in the output continuous dynamics, which is not a realistic assumption in practice. Hence, discretised sampling of input/output trajectories is considered. To this end, in [AHF+14], a notion of timed state sequences is proposed as the basic starting point for conformance testing.

Definition 11 (Hybrid-Timed State Sequence (TSS) [AHF+14]). *Let* $N \in \mathbb{N}$ *and* V *be a set of variables. A hybrid-timed state sequence (TSS) is defined as a 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* $\mathbb{TSS}(V)$.

As observed in the above definition, a TSS contains a list of variable valuations indexed by the corresponding time stamps.

In [MM16a], the authors define a sampling function, notions of test suite and test case for hybrid automaton and propose a verdict algorithm based on the (τ,ε)-conformance notion defined in Definition 10.

Definition 12 (Sampling Function [MM16a]). *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*

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

Given an input trajectory and a sampling function, a test case (test suite) provides the expected output valuations at the specified sampling points.

Definition 13 (Test Suite and Test Case [MM16a]). *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.*

A sampling function is periodic when its sampling points are equally distanced. Furthermore, in this situation, it is possible to define *periodic* test cases. In this work, we limit ourselves to using periodic sampling functions and test cases for simplicity. This is because, in practical scenarios, samples are extracted at a fixed rate since they lead to more consistent results.

Definition 14 (Periodic Test Case [MM16a]). *A test case* (u,y) *is periodic with period p if* $\mathrm{dom(y)}$ *is a periodic set of sampling points with period p.*

Furthermore, we use Algorithm 1 [MM16a] as the test oracle for conformance testing of cyber-physical systems in our tool presented in Chapter 4. In summary, the algorithm compares the expected outcome of the test suit *TS* with the samples of the output trajectories of $\mathcal{HA}_I$. It testes whether each element in *TS* is within T and E (which represent the spatial and temporal bounds, namely, τ and ε) distance of any sampling points provided by the output trajectory of $\mathcal{HA}_I$.

As for the verdict itself, a *Fail* verdict means that the algorithm found evidence that the implementation does not conform to the specification, while a *Pass* verdict means that no such evidence was found. 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 the verdicts.

2.3.3 Soundness

It is possible to use Algorithm 1 as a direct application of (τ,ε)-conformance testing. However, as we show in this subsection, passing the exact values of τ and ε as parameters for Algorithm 1 (as values for T and E, respectively) can result in unsound verdicts. In the remainder of this subsection, we present the basic definitions for soundness in hybrid systems and the required formal conditions to achieve it. Then, in Chapter 3 we present a systematic process for CPS conformance testing that guarantees sound results.

In [MM16a], the authors define soundness for hybrid systems as follows:

Definition 15 (Soundness [MM16a]). *Considering a specification* $\mathcal{HA}$, *a test suite TS is sound under a specified test verdict algorithm if the following proposition holds*

$$\forall \mathcal{HA}_I : \left(\mathcal{HA}_I \approx_{(\tau,\varepsilon)} \mathcal{HA}\right) \Rightarrow \mathcal{HA}_I \textit{ passes } TS \tag{2.1}$$

Algorithm 1 Test Verdict Algorithm [MM16a]: Given a test suite, a hybrid system implementation and conformance testing parameters, it determines whether the implementation passes or fails the test suite.

inputs: A test suite TS; A hybrid automaton $\mathcal{HA}_I$; Conformance parameters T, E
output: Pass or Fail
for each $(u,y) \in TS$ **do**
    $y_I \leftarrow out_{\mathcal{HA}_I}(u)$
    $P \leftarrow \text{dom}(y)$
    $y_I^s \leftarrow \pi_P(y_I)$
    for each $(t,j) \in \text{dom}(y_I^s)$ **do**
        $I_t = [t-T, t+T] \cap \{t \mid \exists j : (t,j) \in \text{dom}(y)\}$
        if $\exists t' \in I_t$ s.t. $\|y(t',i) - y_I^s(t,k)\| \leq E$ **then**
            continue;
        else
            return Fail
        end if
    end for
end for
return Pass

This definition proposes that, for a test suite to be considered sound, the following condition must hold: if the implementation model (τ,ε)-conforms to the specification, then the implementation should pass the test suit. In summary, a test suit is sound if all *Fail* verdicts are correct.

Using τ and ε as arguments for Algorithm 1 can produce unsound results for all practical specifications. The following example (inspired by [MM16a]), illustrates this problem.

Example 4. *Consider the example of the thermostat described in Section 2.2.2. We can find an implementation such that its trajectory is similar to the specification one but it occurs with a delay of 0.1 time units and it is shifted in value by 4 degrees. An example of both trajectories is shown in Figure 2.5, with the specification output trajectory being labelled as y whereas the implementation is labelled as y_I. Note that y reaches its maximum value (18 degrees) when $t = 2.0$ and, similarly, y_I reaches its respective maximum value (22 degrees) when $t = 2.1$. Considering $\tau = 0.1$ and $\varepsilon = 4$ the implementation should satisfy the $(\tau,\varepsilon) - closeness$ condition and therefore it $(\tau,\varepsilon) - conforms$ with the specification. However, as stated, in order to test this in practice, the system should be sampled. If we use 0.03 as sampling rate, then $t = 2.1$ will belong to the set of points in the sampling pool whereas $t = 2.0$ will not. This results in the sampling function not capturing the specification trajectory's peak (18 degrees) and only capturing the surrounding values, which are lower than 18. Hence, when comparing the point in which the implementation is at its peak (22 degrees), Algorithm 1 will not find a single point in the specification that respects the margin of error $\varepsilon = 4$. This leads to a fail verdict and therefore an unsound one, since it won't satisfy the closeness condition when it should.*

The issue arises due to the sampling rate not selecting the points in the specification trajec-

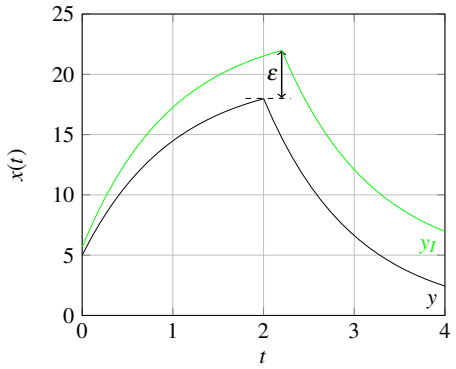


Figure 2.5: The output trajectory of the thermostat specification (y) and that of a sample implementation (y_I) (Taken from [MM16a])

tory that make the conformance to hold. This can happen when there is a sudden large variation in the trajectories, which causes the discretisation process not to capture this behaviour.

In order to reinstate soundness, [MM16a] introduces a new measure based on the specification. The Specification Maximum Change (Definition 16) takes into consideration the sampling period to compute the maximum spatial variation the specification achieves within each sampling interval.

Definition 16 (Specification Maximum Change [MM16a])**.** *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*

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

Moreover, the concept of robust test suites is also introduced and in [MM16a] it is proved that a robust test case is always sound, according to the definition of soundness presented in Definition 15.

Definition 17 (Robust Test Suites [MM16a])**.** *Given $\tau,\varepsilon>0$, assume that we use Algorithm 1 with parameter assignment of $T=\tau$ and $E=\varepsilon+\Delta$, 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*

$$\Delta \geq \Delta_p$$

This definition states that if we can find the boundaries respected by the specification, then we can have robust test suites by extending the margins of error of these bounds. Thus, achieving sound results. Furthermore, it is possible to compute the values for Δ_p based on the formal specification of the system. We elaborate on this computation process in Chapter 3.

3 A PROCESS FOR CPS CONFORMANCE TESTING

In the previous chapter, we presented a conformance testing algorithm for CPSs, based on the (τ,ε)-conformance notion defined in [AHF+14, AMF14]. We have shown that conformance verification for this notion is sensitive to the dynamics of the system model and to the sampling rate. Additionally, we described the work in [MM16a] that specifies the error bounds for a given model and the additional margin of error that is proved to ensure soundness of conformance testing. Here, these previous results are embodied into a potentially scalable, more general and systematic process for checking conformance of CPSs. An initial version of the process we propose first appeared in [ACS+17b].

This chapter is structured as follows: in Section 3.1, we give an overview of the process and briefly describe its steps. In Section 3.2, we elaborate on defining the initial parameters for the process, such as the conformance bounds (i.e. τ and ε) and the sampling rate. Section 3.3 describes our strategy to compute Δ_p (Definition 16) based on reachability analysis and, thus, reinstate soundness. Section 3.4 focuses on describing the last step of the process, which involves result analysis and parameter tuning.

3.1 GENERAL PROCESS STRUCTURE

The main motivation behind this process is to develop a practical approach that guarantees the strategy described in Section 2.3.2 to be sound. Particularly, we have explicitly defined the parameters of a modularised process, split into five steps, with the aim of providing support for an engineer to adopt a systematic strategy.

Figure 3.1 depicts the main steps of the process. The general idea is to offer a semi-automatic solution to achieve conformance testing. While most steps are mechanised, some steps require user interaction and knowledge of the system.

Regarding inputs, the process requires a hybrid model of a cyber-physical system in which the correct behaviour of the system (at its interface level) is specified. Such a model should capture both discrete behaviour and evolution of continuous dynamics of the system. The implementation model is also required, to which the test suite should be applied. Both specification and implementation models should be hybrid automata.

The first two steps of our proposed process is to determine the initial parameters (the sampling rate and the conformance bounds, respectively), which are based on the specification model. For the first step (Figure 3.1-step 1), we propose an automatic process for finding a sampling rate that is likely to achieve a good balance between precision and processing cost. For instance, changing the period from 0.2 to 0.1 doubles the amount of points to process but does not necessarily mean it also doubles the processing time, since this depends on the complexity of the system and its equations. Therefore, the sampling rate computed in this process should be considered as a starting point and it can be fine-tuned in case the results need

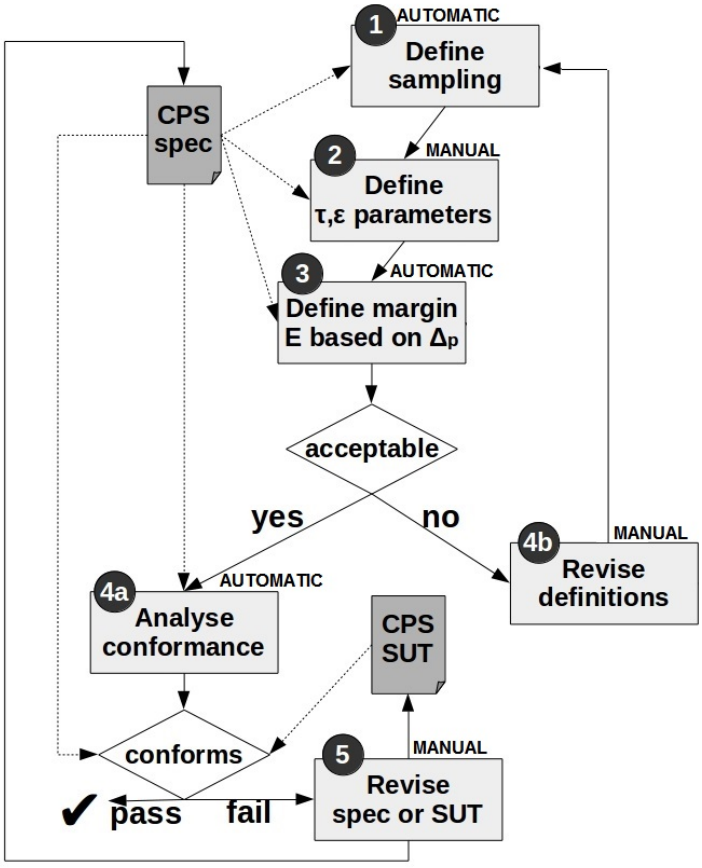


Figure 3.1: Process for sound conformance testing

adjustment. Section 3.2 describes in more details the process for computing the sampling rate.

In the second step (Figure 3.1-step 2), the conformance bounds, namely the τ and ε parameters, are the margins of error considered by the conforming testing strategy. Defining them is a manual step that requires a certain degree of knowledge of the system and its components. For instance, to properly define them, one should ideally take into consideration precision of sensors to account for input/output delay, noise and other small disturbances, thus making it impractical for our process to automatically compute such parameters.

The next step in our process (Figure 3.1-step 3) is automatically performed and involves another precision parameter for error mitigation. In this case, it is to accommodate for data loss in the sampling process. In summary, the discretisation step is lossy by nature and, thus, our strategy requires another layer of margin of error to guarantee that our conformance analysis is sound (see Section 2.3.3). Thus, we perform reachability analysis on the specification model and compute Δ_p, which should be added to ε when passing the spatial error argument ($E = \varepsilon + \Delta_p$) for the conformance algorithm (Algorithm 1), as shown in Definition 16 . See Section 3.3 for a detailed description of the reachability analysis process.

Then, the user should decide whether or not such a margin, *i.e.* E, is acceptable for testing the system. If positive, then the user should be able to perform conformance testing and verify whether the implementation (τ,ε)-conforms to the specification (Figure 3.1-step 4a). In the

event that the test suite fails, the user must revise the models in order to find the problem (Figure 3.1-step 5) and then restart the process. If the test suite passes, the system model of the SUT is consider to (τ,ε)-conform to the specification model.

However, the final value of E, *i.e.* ε in addition to Δ_p, might not be realistic, that is, this margin of error might be too big to be acceptable. In this case, we suggest the user to revise the initial parameters (Figure 3.1-step 4b). One option is to increase the sampling rate, which results in a lower Δ_p but affects processing cost. A second alternative is to revise and possibly reduce the conformance bounds τ and ε, which should also lower the value of E. We further develop on these options and consequences in Section 3.4.

3.2 DEFINING INITIAL PARAMETERS

Since conformance testing inherently involves comparing continuous dynamics, the key parameters of the process are (i) finding and adjusting the sampling rate of the dynamic behaviour, and (ii) defining the conformance bounds, *i.e.* margins of error, which determine whether two signals are sufficiently close to each other. The final parameter of this process is (iii) a method to measure closeness when comparing continuous signals.

3.2.1 Sampling rate

As stated before, discrete sampling of continuous signals is often a necessary step when dealing with CPSs and is closely related to the error margin involved in the discretisation step [MM16b]. The efficacy of conformance testing relies on choosing appropriate sampling rates that are proportional to the pace of changes in system dynamics; otherwise, as discussed in Section 2.3.3 conformance testing may yield unsound results due to the sampling function missing sharp signal variations.

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. The strategy described in Section 2.3.3 consists of increasing the spatial error margin to take into consideration these sharp variations.

The process to define an adequate sampling rate requires an intricate analysis of the system dynamics. We propose a strategy to find a suitable period by inspecting the system in the frequency domain and defining a frequency limit from which the system ceases to provide significant responses. Once the signal is band-limited, we can apply the Nyquist-Shannon theorem [Sha49, Sha01], which is explained later, to find a suitable period.

Hence, the first step of our process to find a suitable sampling rate is to analyse the output signal in the frequency domain, which allows the use of techniques that can determine the system stability.

There are a few ways to do this. A practical and common one is to apply the Fourier transform [BB86] to a signal. However, in the Matlab environment (see Chapter 4 for implementation details), we can only compute the Discrete Fourier Transform, which only works on sampled signals. This means that our signal would have to be already sampled for Fourier to work. Another method that does not share this restriction, and the one we use, is the Bode plot

[Yar10].

The advantage of this method is that it works based on the system specification instead of the sampled output signals. Figure 3.2 shows the thermostat example analysed in the frequency domain acquired via Bode plot. The y-axis represents the waves amplitude and the x-axis represents its frequency. The former is a measurement of how big the waves are and the latter represents the number of wave cycles passing a point in a time unit. In cyber-physical systems, the lower frequencies tend to have higher amplitude and thus are more relevant.

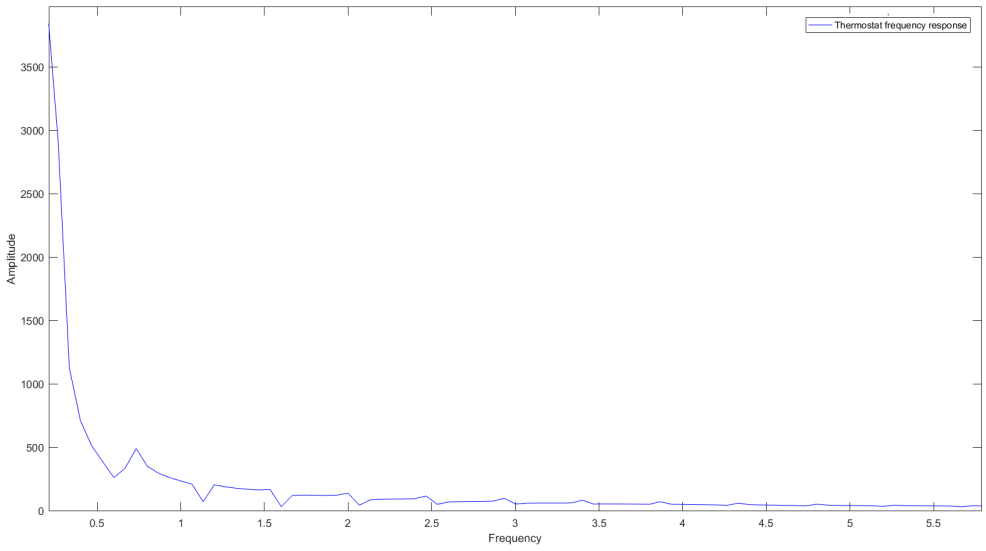


Figure 3.2: Frequency response for the thermostat example

The next step is to define the frequency threshold where the system's frequency response begins to be reduced, so we can apply the Nyquist-Shannon theorem. For that, we compute the cumulative energy percentage via cumulative integration. This value represents the percentage (from 0 to 1) at which the energy flowing through the system attenuates to negligible values. Figures 3.3 and 3.4 show the frequency response (continuous line) along with the cumulative energy (dotted line).

In our strategy, we use the cumulative energy limit of 0.999. Thus we limit the system frequency at the same point in which the cumulative integration reaches 0.999. Once we have a frequency limit, *i.e.* the system is band-limited, we can finally apply the Nyquist-Shannon theorem.

The theorem states that, for band-limited signals, a suitable period must be equal to, or greater than, twice the highest frequency component. Consider the signal depicted in Figure 3.5, which is band-limited at frequency B. According to the Nyquist-Shannon theorem, an adequate period p would be $p = \frac{1}{2B}$.

We have applied this process on the thermostat twice, using different input signals each time. On the first run we fixed the inputs to $h(t) = 20$ and $i(t) = 0$. For this particular set of inputs, using 0.999% as cumulative energy threshold, the yielded cutoff frequency is 14.6 Hz.

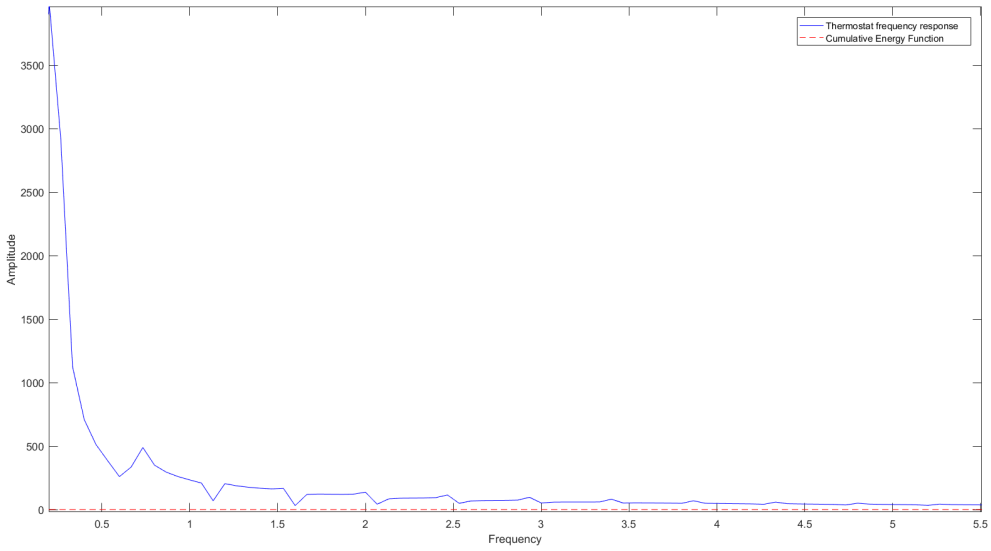


Figure 3.3: Cumulative integration applied to the thermostat

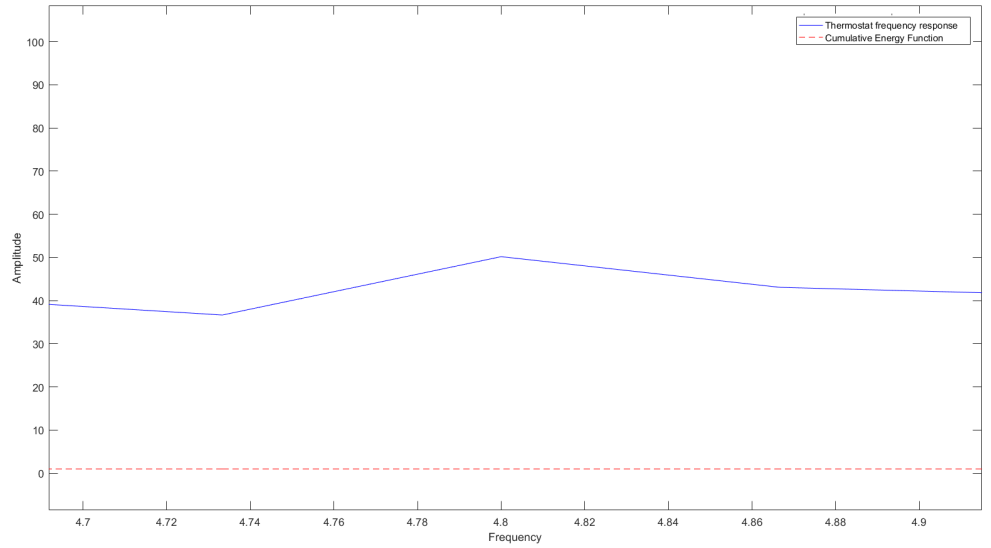


Figure 3.4: Cumulative integration applied to the thermostat - Zoomed in

Using the Nyquist rate (2 times the cutoff frequency), the sampling rate for the system should be approximately 0.034.

On the second run, the input signals are variable in time, increasing at a fixed rate. This affects how fast the temperature of the room changes. Using 0.999% as cumulative energy threshold as well, we reach the cutoff frequency of 18.4 Hz. Thus, the Nyquist rate yields approximately 0.027 as an adequate sampling rate. This shows, as expected, that the greater

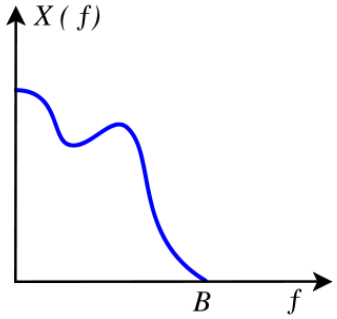


Figure 3.5: Band-limited frequency

the output variation over time, the lower the sampling rate value needs to be.

3.2.2 Precision

Our conformance testing strategy involves observing the system behaviour on a finite number of points and, hence, involves reducing the comparison of continuous behaviours to comparison of discretised samples; this process inherently deviates from the precise comparison of continuous signals. Moreover, conformance testing needs to take into account measurement errors introduced by sensors and noise in general, such that the smallest disturbance in the signal will not necessarily yield unsound conformance results. For these reasons we have opted to adopt the (τ, ε)-conformance in our strategy.

Another precision component was detailed in Section 2.3.3. There, we have shown that in order to generate sound test cases for the (τ, ε)-conformance relation, one has to incorporate a second layer of error mitigation (Δ_p) that is inversely proportional to the rate of changes of the dynamics in the specification within the sampling intervals, *i.e.*, the higher the sampling rate, the lower the error margin. This parameter has a direct impact on the applicability of the test strategy, since the imprudent increase of error margins can render the results inadmissible.

In our process, the τ and ε parameters are manually defined. The definition of these parameters requires intricate knowledge of the system, which renders an automatic definition difficult. However, for Δ_p, it is possible to define a mechanical process to find its value. The theoretical and practical details regarding its computation are discussed in Section 3.3 and Chapter 4, respectively.

3.2.3 Closeness

When comparing physical systems, it is often necessary to measure how "close" they are from each other. In conformance testing of CPSs, that can mean measuring the distance between two output signals or computing the intersection of a signal with a region. The most common approach considers Euclidean distance [DD09], which is a widely known method for computing the distance between points in a geometrical space.

In the process proposed in this chapter, although we consider Euclidean distance by default, it is possible to adopt other closeness notions, as well. In particular, we consider the (τ, ε)-conformance relation (Section 2.3.1), which takes the Euclidean measure for closeness

of signals..

In addition to the Euclidean distance, other approaches have also been reported in the literature. In [DMP15] and [CB02], Skorokhod distance is used to measure the closeness between continuous signals with promising results. In [DMP15], the author argues that Skorokhod is well suited for conformance checking and also provides benchmarks using a prototype tool that mechanises his strategy. In [CB02], however, the author questions the use of Skhorokod in a multiple input system and also in unstable systems.

3.3 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 over-approximation 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 [LPY01].

In our context, we use reachability analysis to provide an over-approximation of the changes in the system dynamics around the sampling points. This will in turn provide us with an over-approximation for the value of Δ_p used in the adopted conformance testing algorithm. To this end, we use the reachability analysis algorithm and tool developed by Matthias Althoff [Alt10].

In Section 3.3.1, we present an overview of the work regarding reachable sets for hybrid systems [Alt10, AK11]. In Section 3.3.2, we present the definition of zonotopes, which are the geometric forms used to represent reachable sets. Afterwards, in Section 3.3.3, we show how we use reachability analysis to obtain sound test cases, taking advantage of the results presented in Section 2.3.3. An initial version of this strategy was first described in [ACS+17a].

3.3.1 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 [Alt10]. 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 presented in [Alt10], 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 generalise it for a super-dense time domain (with jumps).

Definition 18 (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:*

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

Each element in the above set is a valuation (at time t after j jumps) restricted to the output variables. Recall that $\phi(t,j) \downarrow V_O$ denotes the trajectory restriction (see Definition 6). The set contains all such elements, for all trajectories of the automaton $\mathcal{HA}$. Figure 3.6 illustrates the temperature trajectory of the thermostat example (see Figure 2.2) starting from multiple initial states. It also depicts the exact continuous reachable set at a time t, namely, $\mathcal{R}^e(t)$.

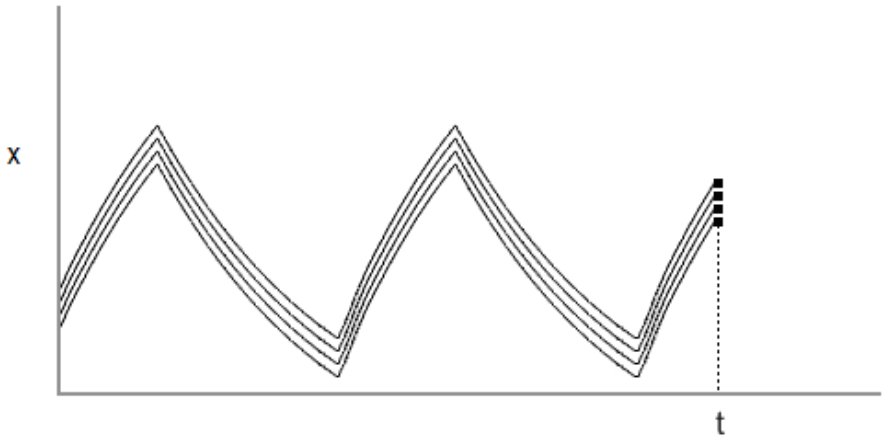


Figure 3.6: Thermostat - Trajectories for multiple initial states

In practice, however, exact reachable sets are only computable for some restricted forms of dynamical systems [LPY01]. 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 work, since we use the approach proposed by Althoff [Alt10, AK11] 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 19 (Reachable Set of a Time Interval [Alt10])**.** *The reachable set of a time interval is the union of reachable sets at points in time within the interval $t' \in [0,t]$:*

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

Concerning the thermostat example, Figure 3.7 shows the computed over-approximated reachable set of a time interval. In this case, it represents the reachable set $\mathcal{R}([0,t])$. The central

lines are the system behaviour and the surrounding area corresponds to the over-approximated reachable sets.

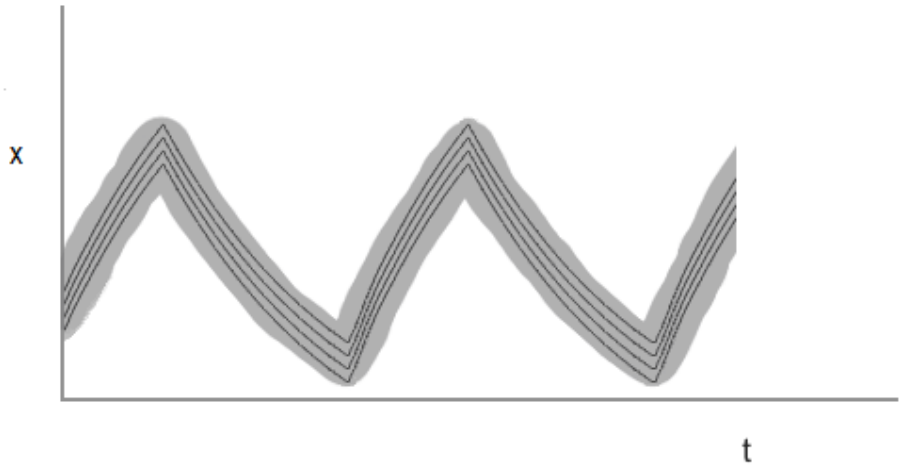


Figure 3.7: Thermostat - Overapproximation of reachable set from 0 to t

3.3.2 Reachable set representation

There are several ways to represent reachable sets; ellipsoids [Che15], orthogonal polyhedra [BMP99] and oriented rectangular hulls [SK03] 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 characterise geometric forms. Zonotopes are a compact and efficient way to portray high-dimensional sets due to their efficient computability [AK11].

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 [ASB10, GG08] cope with time intervals and intersections.

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

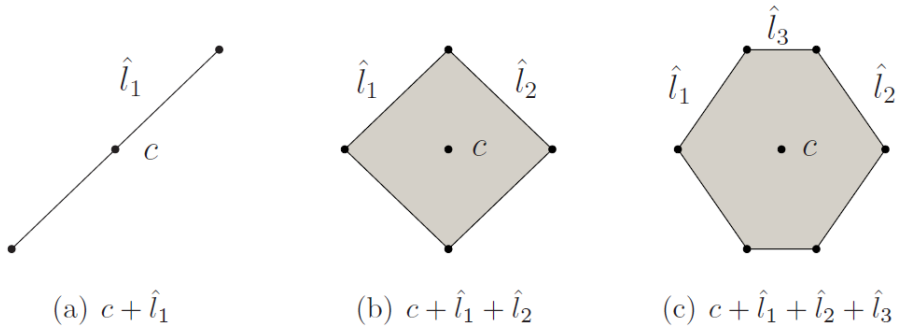
Definition 20 ($\mathcal{G}$-representation of a zonotope [Alt10])**.** *A zonotope is a set*

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

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

In this definition, c represents the centre point, $g^{(i)}$ are the generator points and β_i represents the values in the $[-1,1]$ interval. The product resulting from $[-1,1] \cdot g^{(i)}$ is a line $\hat{l}_i$ as can be seen in Figure 3.8(a), which illustrates the iterative construction of a zonotope. Note that adding generators, and therefore lines, results in better defined forms at each step.

Definition 20 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



(a) $c + \hat{l}_1$ (b) $c + \hat{l}_1 + \hat{l}_2$ (c) $c + \hat{l}_1 + \hat{l}_2 + \hat{l}_3$

Figure 3.8: Iterative construction of a zonotope (taken from [Alt10])

that represent shapes. The general formula is given by $A \oplus B = \{a + b \mid a \in A, b \in B\}$. For instance, the Mikowski sum of two sets of points forming triangles $A = \{(1,0),(0,1),(0,-1)\}$ and $B = \{(0,0),(1,1),(1,-1)\}$ would result in a 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 3.9 illustrates the zonotope representation of those reachable sets of the thermostat example. The geometric forms enclosing the central line are the computed zonotopes.

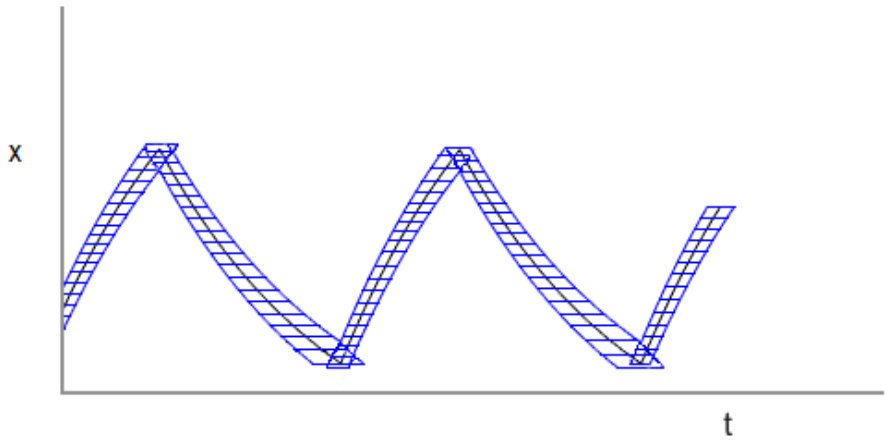


Figure 3.9: Thermostat - Zonotopes

3.3.3 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 16. 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 16), 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 Definition 17).

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

Definition 21 (Specification Maximum Change based on Reachable Sets)**.** *Given a specification $\mathcal{HA}$, a 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*

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

Considering that the result represents an over-approximation of the trajectory boundaries, we can assume that the actual specification maximum change (i.e., Δ_p) 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 achieve 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 Chapter 4) 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 HA, a period p, an input stimuli u;
2:  output: Δ_p^RS
3:  Zonotopes = computeReachableSets(HA, p, u)
4:  for each z_t ∈ Zonotopes do
5:      Δ_p^RS(t) = max(z_t) − min(z_t)
6:  end for
7:  Δ_p^RS = 0
8:  t = 0
9:  while t ≤ T do
10:     if Δ_p^RS(t) > Δ_p^RS then
11:         Δ_p^RS = Δ_p^RS(t)
12:     end if
13:     t = t + p
14: end while
15: return Δ_p^RS
```

3.4 RESULT ANALYSIS

In this section, we present four possible scenarios based on our running example (Section 2.2.2). Using these scenarios, we demonstrate different possible outcomes of conformance

testing using our process, as well as their interaction with the steps for guaranteeing soundness.

To obtain these results, we have used a prototype tool for conformance testing of hybrid systems. The tool generates test cases, performs reachability analysis to compute Δ_P^{RS} and applies the generated test suite in order to verify if the (τ,ε)-conformance holds. We refer to Chapter 4 for more details on the tool development, as well as other case studies more complex than the thermostat example. Here, we abstract from implementation details to focus on the aspects concerning our process.

The scenarios presented in the following sections reflect aspects of steps 4 and 5 of the proposed process. Implementation 1 (Section 3.4.1) represents the best case scenario, where the implementation passes the test with the strict error margins τ and ε, without the need of Δ_P^{RS}. However, this scenario is not necessarily the usual case. Then, in Section 3.4.2, we present Implementation 2 (Figure 3.1-step 4a), where the test suite fails (*i.e.*, at least one test case yields a fail verdict), but the failure is not due to a non-conforming implementation, but rather unsoundness of the test suite with respect to the chosen sampling rate and conformance bounds. Hence, the conformance testing algorithm takes into account $\varepsilon + \Delta_P^{RS}$ as the value of E.

In Section 3.4.3 we consider Implementation 3 (Figure 3.1-step 4b), in which the adjusted sound bounds are deemed unacceptable, which leads to the revision of the initial parameters. In this case, the sampling rate is increased resulting in tighter and acceptable bounds. In Section 3.4.4, Implementation 4 (Figure 3.1-step 5) represents the case where the adjusted sound bounds are considered acceptable, but even then, 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.

For all implementations, the 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 ($h(t)$ and $i(t)$ – see Figure 2.2) 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 $h(t) = 20.0$ and $i(t) = 0.0$.

3.4.1 Scenario 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 $\varepsilon = 3$. Both trajectories are shown in Figure 3.10.

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 cases. This example represents the best case scenario, where, due to an appropriate choice of sampling rate, only sound test vectors are generated.

Table 3.1 displays some of the test vectors we have obtained from the specification, whereas Table 3.2 displays the temperature values for the corresponding implementation sample points.

The difference in output values between the specification and the implementation for all

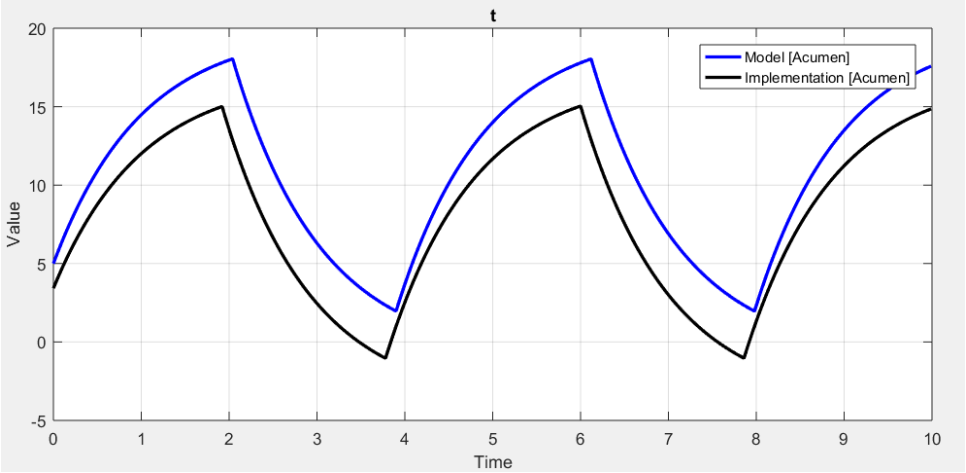


Figure 3.10: Temperature trajectory

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

Table 3.1: Test cases generated from the specification

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

Table 3.2: Values obtained from the implementation

sample points is within the conformance bounds ($\tau = 0.1$ and $\varepsilon = 3$) and therefore the implementation passes the test suite.

3.4.2 Scenario 2 – unsound test cases

As shown in Section 2.3.3, 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 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 3.10. 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 have demonstrated that the implementation under test is indeed (τ, ε)-conforming to the specification (for a given $\tau = 0.1$ and $\varepsilon = 3$); hence, the verdict is clearly unsound. Figure 3.11 depicts this unsound non-conforming example.

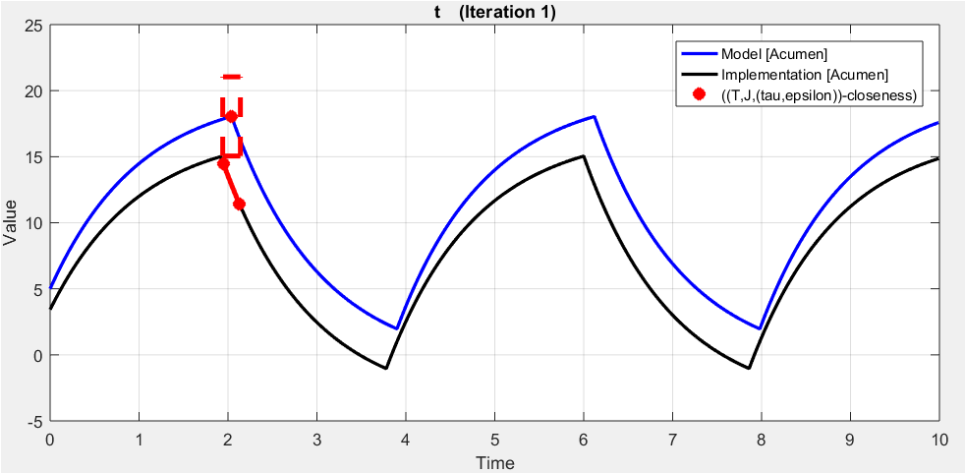


Figure 3.11: Temperature trajectory

As stated earlier, this issue is caused by the lossy discretisation process. A solution that guarantees soundness regardless of the sampling rate is to find the value that represents the maximum possible theoretical deviation of the specification trajectory between samples *i.e.*, find Δ_p. For that, our tool uses reachability analysis (for implementation details, see Chapter 4). For the thermostat example, this automatic process yields $\Delta_P^{RS} = 0.984$ degrees as a result.

Thus, for this example, by extending the error margin, ε, of 0.984, we obtain sound results (see Definition 17). 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.

3.4.3 Scenario 3 - increasing the sampling rate to tighten conformance bounds

Consider the same implementation used in Case 2. In light of the **pass** verdict, the resulting error margin value of 3.984 might be considered inadmissible (*i.e.*, too high). Considering that the original margin of error was 3.0, with this greater value, certain implementations that would otherwise result in **fail** verdicts, might now result in **pass**.

It is possible to mitigate this issue by increasing the sampling rate and, thus, lowering the Δ_P^{RS} value and, consequently, E. For instance, in this case, a sampling rate of 0.001, yields 0.437

as Δ_p. Note that choosing a higher rate leads to increased hardware usage and the maximum possible value might be limited by system or environment constraints.

Finally, it should also be possible to find the smaller sampling rate that results in sound verdicts, given a fixed Δ_P^{RS}. For example, one might be interested in the minimum sampling rate that achieves soundness for a Δ_p value of 0.5. However, this approach was left unexplored and open for further study.

3.4.4 Scenario 4 - adjusting τ and ε to achieve conformance

For some non-conforming implementations, 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.

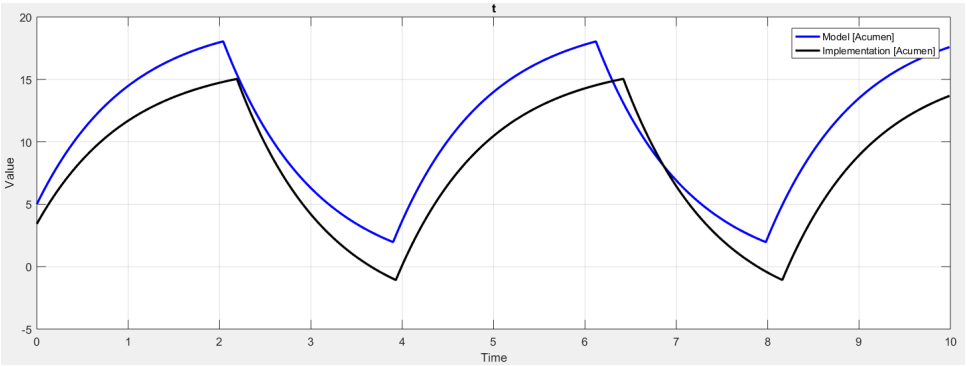


Figure 3.12: Unconforming thermostat implementation

Consider the faulty implementation depicted in Figure 3.12 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^{RS}, the specification and the implementation are found to be non-conforming; then, it is possible to adjust the parameters τ and ε gradually until the implementation is considered (τ, ε)-conforming. This can be used to determine the degree of (non-)conformance of the current implementation with respect to the specification.

For the implementation depicted in Figure 3.12, if we fix $\tau = 0.1$, the minimum value for ε that makes the implementation conforming with respect to the implementation is 3.602[1]. Similarly, with a fixed $\varepsilon = 3$, 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 3.4.1, if one fixes $\varepsilon = 2.5$, there are certain points in the specification signal (e.g., the peaks of the signal) that will never be conforming, regardless of τ.

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

4 EMPIRICAL ANALYSIS

In this section, we describe a prototype implementation of the ideas that are present in the previous chapter. This prototype is an extension of a previously developed tool for conformance testing of hybrid systems [AMR15a]. We give a brief explanation of the tool, but, we focus mostly on the new aspects concerning soundness of conformance testing. In addition to test case generation and conformance analysis, already implemented in the tool presented in [AMR15a], the tool now interacts with a reachability analysis tool called CORA [Alt15] to calculate sound conformance analysis margins.

CORA (COntinuous Reachability Analizer) [Alt15] 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 a given sampling rate.

4.1 TOOL OVERVIEW

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, it samples the specification considering an input signal and a given sampling rate, calculates the appropriate error margins (for sound analysis), executes the test cases on the system under test and, finally, reaches a verdict analysing the results with respect to the given conformance bounds and the calculated error margin. The starting point is, 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 4.1).

To explain the general usage of the tool we have partitioned Figure 4.1 in 3 areas. The area marked as 1 is the area where the user selects the models for input and also where the output console is located, which serves to exhibit messages to the user. Area 2 is for executing commands and parameter definition. Here, the user can generate and execute test cases and perform conformance analysis. The analysis can only be done once the user defines the ε and τ in the appropriate fields. In area 3, the user can select which output he wants to inspect. If the tool finds a counterexample during the conformance analysis for this output, the signal and the error are displayed here as well. We refer to [AMR15a], for further details on this tool.

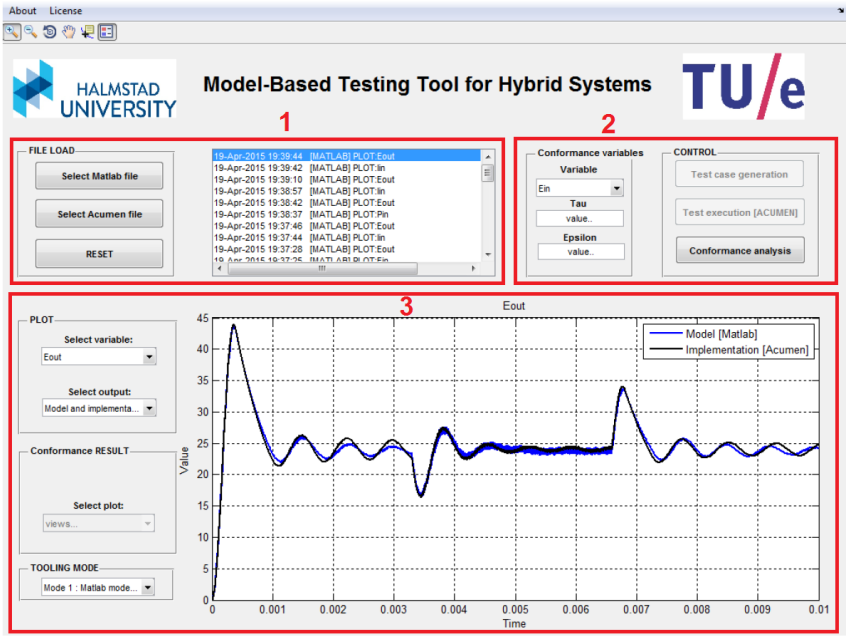


Figure 4.1: Graphical User Interface of the tool

4.1.1 Tool extension

In order to apply the strategy described in Section 3.3, our tool executes CORA in background to perform reachability analysis. The seamless integration of the tools is made possible because both are implemented in the Matlab environment. However, the specification model format required by our tool and the one required by CORA are different, so an additional CORA-readable model is required for performing the reachability analysis. Currently, although it is possible to mechanise, there is no method for automatic translation. Moreover, it is also expected that the name of variables and the sampling period in both models are the same.

Figure 4.2 shows the new tool interface extended with new features. For instance, there is a new button on the top right side to start the CORA process in the background and perform the reachability analysis as shown in Figure 4.3. The results of this computation will appear in the *Delta* (Δ_p) field and is added automatically to *Epsilon* (ε) resulting in the value of the parameter E of the verdict algorithm (Algorithm 1 - Section 2.3.2).

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) [KGB04]. Using this toolbox, 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 21. Each reachable set accounts 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.

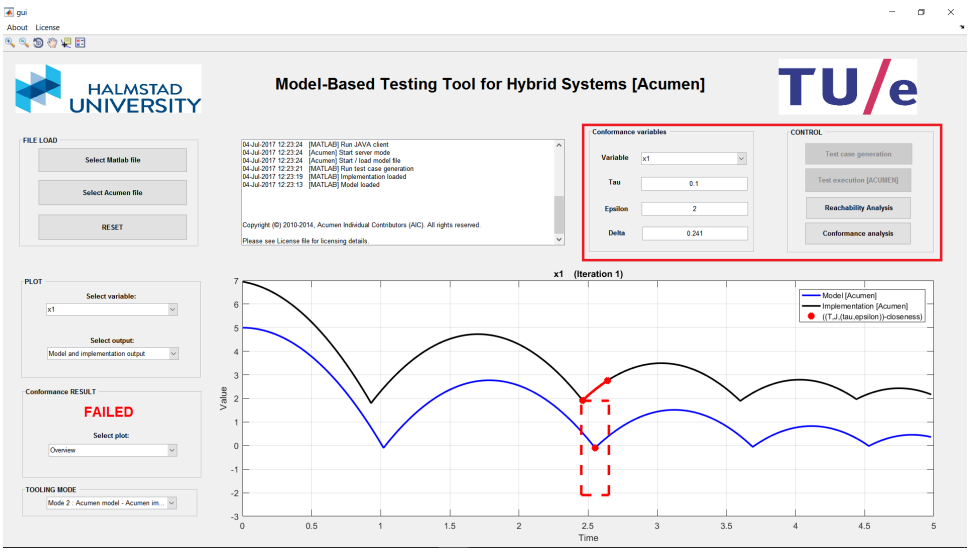


Figure 4.2: New Graphical User Interface of the tool

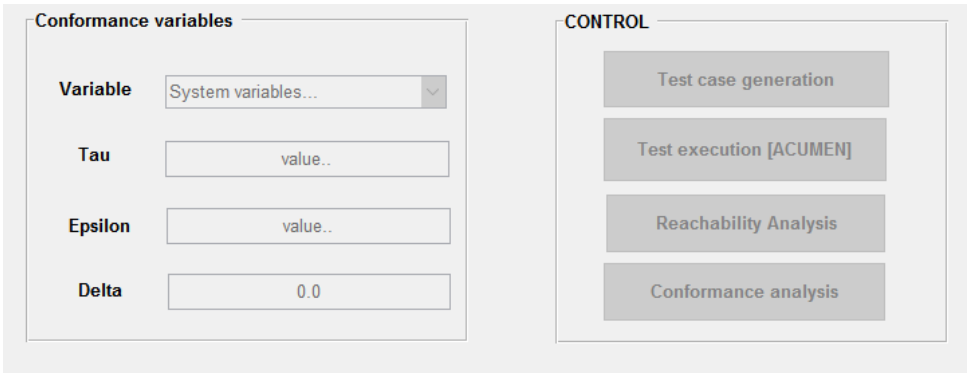


Figure 4.3: Graphical User Interface modifications

Once the specification maximum change is calculated, its value is displayed to the user, and it is automatically added to the value of *Epsilon*, as previously said. The user can then determine whether the calculated margin is tight enough to yield practically meaningful conformance results or whether he should increase the sampling rate to obtain tighter margins. We have illustrated these possibilities in examples described in Section 3.4, when we computed Δ_P^{RS} for certain scenarios. Now, with the aid of the extension we implemented in the aforementioned tool, we present two case studies where we apply the process for sound conformance testing proposed in this work.

4.2 CASE STUDY 1: A SUSPENSION SYSTEM

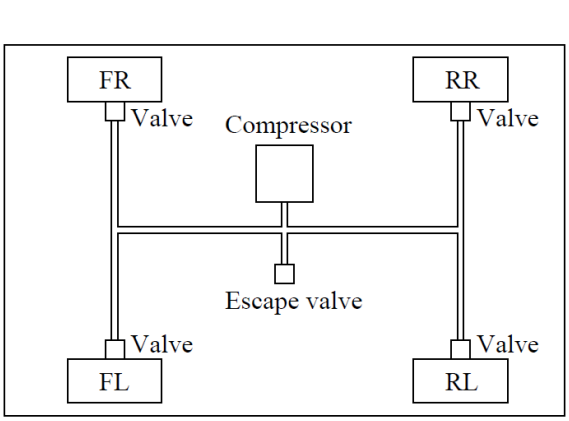
This section presents the applicability of our strategy considering a more complex case study: an automotive pneumatic suspension system [MS00] that can be modelled as a hybrid automaton. The system's goal is to increase driving comfort by adjusting the chassis level to

compensate for road disturbances. This is achieved by a pneumatic suspension that connects the valves attached to each wheel to a compressor and an escape valve (see Figure 4.4a).

The system aims to keep the chassis level as close as possible to a defined set point in each of the four wheels. Sensors can measure the deviation and those values are passed to a controller. The controller reads these values and operates the compressor and the escape valve to either pump air into or blow air off one of the wheel's valves. For example, to increase the level at the front right wheel, the compressor has to be turned on, the escape valve must be closed and the valve connected to the suspension of the front right wheel must be opened.

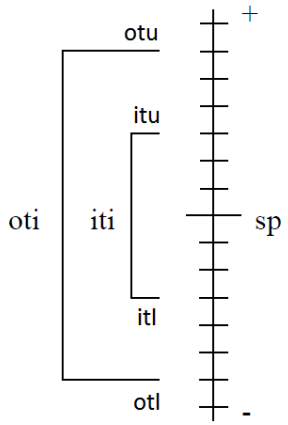
This decision to increase or decrease chassis level is based on tolerance intervals defined for each wheel (see Figure 4.4b). Whenever the chassis height exceeds the outer tolerance interval (*oti*) for one wheel, the controller decreases the chassis level by opening the escape valve and the wheel's valve. Once the chassis level is below the upper limit of the inner tolerance interval (*iti*), the controller closes both valves. The process to increase the chassis level is performed analogously using the compressor instead of the escape valve.

Furthermore, an input signal informs the controller whether the car is driving through a bend. If so, the compressor and escape valve must be switched off, as the priority is given to different stability controllers over this one that keeps chassis level within tolerance intervals. In this example, only two operating modes, namely *driving* and *bend*, are modelled. However, one could easily implement other modes such as *stopped* that could work similar to *driving*. Additionally, we consider $[sp + otl, sp + otu]$ and $[sp + itl, sp + itu]$ as the outer and inner tolerance interval in mode *driving*, respectively. Here, sp represents the set point which is the target value the chassis level and *otl, otu, itl* and *itu* represents the tolerance thresholds as seen in Figure 4.4b.



FR: front right wheel
FL: front left wheel
RR: rear right wheel
RL: rear left wheel

(a) The mechanic system



(b) The tolerance intervals

Figure 4.4: Suspension system [MS00]

4.2.1 Hybrid automaton representation

The automaton we present here is a modified version of the original one presented in [MS00]. We have made a few changes so that the example can work in the tools we use, since the original model contains certain unsupported features. For instance, it comprises several smaller modules that are supposed to work in parallel via labelled synchronisation. Additionally, the model contains non-deterministic differential equations where the variables derivative receive an interval instead of a direct value assignment, as we normally see and use in our examples.

Thus, we have modified the example in such a way to remove the parallelism but keeping the overall behaviour intact (*i.e.*, computing the parallel product of the constituent hybrid automata). Additionally, we have removed the non-determinism by adding inputs that can assume the same values contained in the aforementioned intervals and are assigned directly to the corresponding variable derivative. The model we present here contains 4 locations with several differential equations each, as it can be seen in Figure 4.5. Note that, in this figure, there is a transition from a 'super-state' to the state **bend**. This is a concept not supported in the tools we use. We chose to portrait the figure this way to simplify visualisation. In the actual model, there is a transition from each state to the state **bend**.

The system receives 4 inputs and it outputs the current chassis level c. The inputs are *bend*, $dist$, cp and ev. The first (*bend*) triggers the change between operating modes, e.g., switching from *driving* to *bend*. The second ($dist$) corresponds to the disturbance level coming from the environment, which indicates road perturbations such as small depressions or elevations. The last two (cp and ev) dictate the change in chassis level performed by the compressor and the escape valve, respectively.

This automata works as follows. The system starts on the **in_tolerance** state, which represents the state where the chassis is within the tolerance interval and all valves are closed as well as the compressor. Whenever the controller needs to increase the chassis level, the system switches to the **up** state. Analogously, the system switches to the **down** state when the chassis level needs to be decreased. The fourth state is called **bend** and it represents the *bend* operating mode that is triggered whenever the car is driving through a bend.

In order to switch locations, a transition guard must hold. For instance, the transition from **in_tolerance** to **up** is only performed when (1) the car is not driving through a bend, (2) the filtered chassis level f is less than the outer tolerance interval and (3) the timer t is equal to the sampling period t_sample. The first condition checks whether there is a change in the operating mode. Whenever the *bend* signal is equal to 1 the system switches to the **bend** state, regardless of the current state. Thus, this verification is performed in every transition. The second condition compares the filtered chassis level against the tolerance intervals, which is the major prerogative for a change in the chassis level and thus a change of location. The third condition also occurs in all transitions of the system and is there to guarantee that these transitions will only happen within the sampling points. Additionally, the timer t is always reset to 0 after any transition so that the third condition can be checked again during future transitions.

Almost all transitions follow this same pattern. There is a *bend* condition followed by a chassis level check to verify whether it warrants a state change. Finally, the timer needs to be

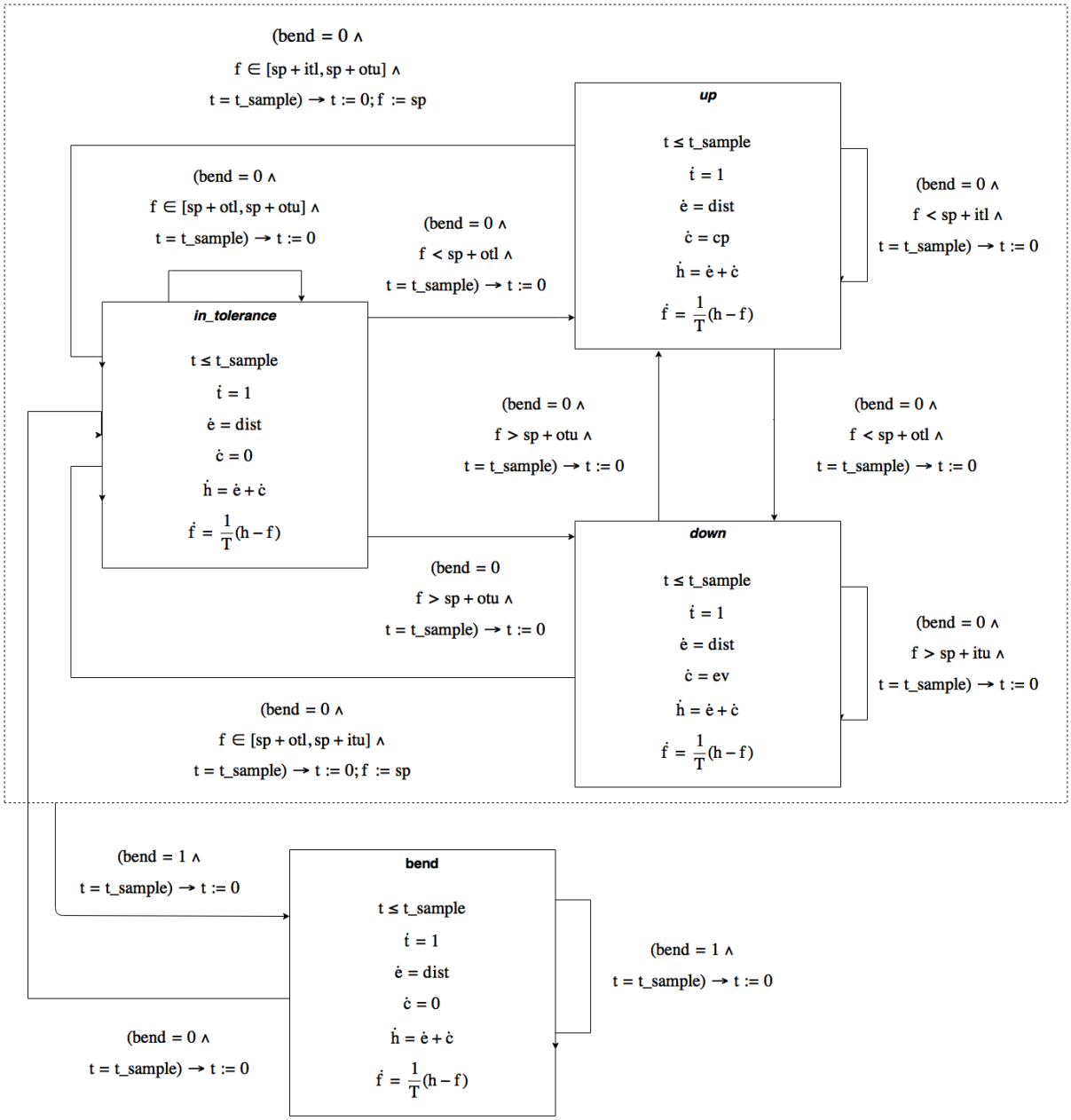


Figure 4.5: Hybrid Automaton of the Suspension System

equal to the system period for the transition to occur. Two exceptions to this pattern occur in transitions coming from **up** and **down**, and going to **in_tolerance**. In these cases, there is an additional operation, which is when the filtered chassis level is reset to the set point. The other exceptions are transitions involving the **bend** state; the reason for this is because the change to or from this state is triggered regardless of the chassis level.

Each location also has an invariant followed by 5 differential equations. The invariant checks if the timer is still less than the period, while the equations represent the required changes in the internal system variables as well as the output c. In the **up** and **down** states,

the chassis level changes are dictated by the compressor and escape valve inputs, respectively. However, it remains constant on the **in_tolerance** and **bend** states. Furthermore, we note that the system uses the filtered chassis level f instead of the current chassis level c on its guards. The f computation uses a combination of the current chassis level, environmental disturbances and a filter constant T.

4.2.2 Conformance testing

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 *bend*, the other inputs change over time while respecting their constraints. In this example, *bend* is always equal to 0. The reason behind this decision is because when the system is operating in the *bend* mode, the chassis level remains stationary, which negates the data loss that happens during the sampling process. Thus, there is no unsound verdict when the output trajectory is constant. In what follows, we apply the process steps in order to achieve sound results.

The first step in the process is to find a suitable period for the system. Using the steps described in Section 3.2.1, we have analysed the system in the frequency domain and used 0.999 as the limit value for cumulative energy. Hence, the cutoff frequency for this example is 13.3 Hz, leading to an approximate sampling rate of 0.0375, after applying the Nyquist-Shannon theorem. Similar to the thermostat example (Section 3.4), we have also created modified implementations by shifting the output trajectory F_c. However, in this case study, we have only considered a time-shift (τ) of 0.3 seconds. Thus, the implementation should be (τ,ε)-conforming when using $\tau = 0.3$ and $\varepsilon = 0$. Considering these values, 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 4.1 and 4.2. 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 0.3 time units later, which did not happen. As we explained in Example 4, this discrepancy occurs because the values obtained from the specification do not belong to the sampling points taken from the implementation.

	$Trajs(V_I)$				$\mathbb{TSS}(V_O)$
t	*bend*	*dist*	*cp*	*ev*	c
2.96	0	-0.49	2.31	0	4.86
5.32	0	0.54	0	1.57	3.71
8.85	0	-0.87	3.28	0	5.14

Table 4.1: Test cases generated from the specification using 0.0375 as period

As explained in details in Section 3.3, 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 0.102, and adding Δ_p^{RS} to ε, before the conformance check, resulted in a **pass** verdict, as expected.

	$Trajs(V_I)$				$\mathbb{TSS}(V_O)$
t	$bend$	$dist$	cp	ev	c
3.26	0	-0.49	2.31	0	4.81
5.62	0	0.54	0	1.57	3.77
9.15	0	-0.87	3.28	0	5.08

Table 4.2: Values obtained from the implementation using 0.0375 as period

Now, consider that the user finds the Δ_p^{RS} value too high. One way to lower it is to increase the sampling rate. Suppose that, the system uses a period of 0.03 instead of 0.0375. In this case, the unsoundness issue still persists at similar points in the trajectory as presented in tables 4.3 and 4.4. However, this time the reachability analysis leads to a Δ_p^{RS} of 0.074.

	$Trajs(V_I)$				$\mathbb{TSS}(V_O)$
t	$bend$	$dist$	cp	ev	c
2.94	0	-0.48	2.29	0	4.82
5.31	0	0.55	0	1.59	3.75
8.85	0	-0.87	3.28	0	5.14

Table 4.3: Test cases generated from the specification using 0.03 as period

	$Trajs(V_I)$				$\mathbb{TSS}(V_O)$
t	$bend$	$dist$	cp	ev	c
3.24	0	-0.48	2.29	0	4.77
5.61	0	0.55	0	1.59	3.81
9.15	0	-0.87	3.28	0	5.08

Table 4.4: Values obtained from the implementation using 0.03 as period

4.3 CASE STUDY 2: AUTOMOTIVE AIR-FUEL RATIO CONTROL

This second case study is based on a model of a controller for an automotive air-fuel ratio (AFR) control system [CSB+06]. The AFR is an important measure in an internal combustion engine, which directly affects its performance and pollution.

In its formal model [JDK+14], 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 simplifications non-trivial. The controller, however, can be viewed as a hybrid system that considers continuous inputs and outputs and switches the operating mode as required.

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

4.3.1 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 [JDK+14].

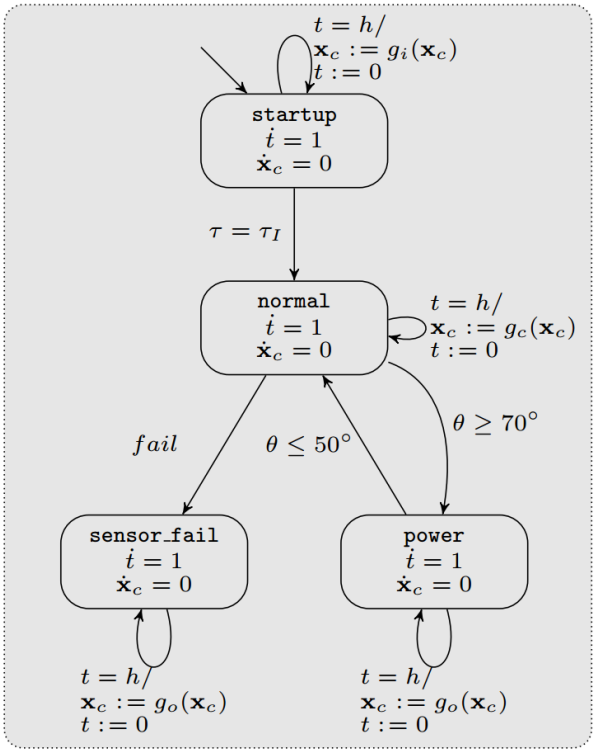


Figure 4.6: Hybrid automata representation of the controller [JDK+14]

A visual representation of the controller hybrid automaton is shown in Figure 4.6 (taken from [JDK+14]). 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 4.6, 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.

4.3.2 Conformance testing

Now, we present how we have applied our process for sound conformance testing in this example. As previously said, the first step is to find a suitable period for the system. Applying the strategy described in Section 3.2.1, it yields a cutoff frequency of 19,8 Hz, which results in a sampling rate of approximately 0.025 when using the Nyquist-Shannon theorem.

Similar to the other examples, we have also found an implementation that leads to a unsound verdict, which was a result of an intentional shift in the output trajectory. Thus, the implementation has a time-shift of 0.5 seconds with respect to the specification model. Thus, considering $\tau = 0.5$ and $\varepsilon = 0$, a sound conformance analysis should lead to a pass verdict.

The tables below illustrate the unsound case. The results presented in Tables 4.5 and 4.6 were obtained under the same input stimuli.

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

Table 4.5: Test cases generated from the specification

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

Table 4.6: Values obtained from the implementation

Since the implementation only had a time-shift, the values should have remained the same 0.5 time units later, which did not happen due to mismatching sampling points. Thus, this

implementation requires that we compute Δ_p^{RS} to obtain a sound verdict. We have used our extension with Reachability Analysis for this computation and the value obtained for Δ_P^{RS} was 0.153. As expected, considering this value yields a **pass** verdict. Considering the low value of Δ_p in this case, it is possible to accept these results if the sampling rate cannot be lowered further. Otherwise, using lower values for the periodic sampling might yield lower values of E.

4.4 PERFORMANCE ANALYSIS

We present the metrics for generating test scenarios using the examples presented in this work, namely the thermostat, the suspension system and the AFR controller. This empirical evaluation analyses the automatic steps in the proposed process and measure the time it takes for them to complete. We have also examined the use of different sampling rates and how it affects performance.

We focus on the metrics related to (i) the time taken to find a suitable sampling rate given a specification, which is the first step in the proposed process (Figure 3.1 - Step 1); (ii) the time taken for computing Δ_p^{RS} (Figure 3.1 - Step 3); (iii) the time taken to generate and execute the test cases and reach the conformance verdict (Figure 3.1 - Step 4a). In all tables presented below, we show the time in seconds.

The data in Table 4.7 is related to Step 1. We have noticed that the time it takes for computing a sampling rate is mostly constant. The slight difference in time seems to be affected by the amount of states and differential equations which are also shown in the table. However, we have not explored systems with a really large amount of states to reach a final conclusion and this is left for future work.

Example	States	Equations	Time
Thermostat	2	2	0.12 s
AFR Controller	4	8	0.13 s
Suspension System	4	20	0.16 s

Table 4.7: Computational time of Step 1

As for Steps 2 and 3, we have noticed that the computational time of Δ_p^{RS} is directly related to the sampling rate of the input model. Table 4.8 displays the average amount of time, in seconds, that the tool takes to compute Δ_p^{RS} for the thermostat, the AFR controller and the suspension system. Analogously, Table 4.9 displays the time for generating and executing test cases and reaching a verdict.

As expected, the smaller the period, the slower it takes to yield a result. Nonetheless, the time seems to grow linearly, even when using more complex examples.

	Time		
Period	**Thermostat**	**AFR Controller**	**Suspension System**
0.5	2.3 s	4.3 s	5.7 s
0.1	3.9 s	6.8 s	8.3 s
0.03	5.8 s	8.4 s	10.7 s
0.01	7.1 s	10.2 s	12.8 s

Table 4.8: Computational time of Step 3

	Time		
Period	**Thermostat**	**AFR Controller**	**Suspension System**
0.5	3.5 s	5.2 s	6.6 s
0.1	5.1 s	8.3 s	9.5 s
0.03	6.4 s	10.1 s	12.2 s
0.01	8.9 s	11.9 s	13.8 s

Table 4.9: Computational time of Step 4a

5 CONCLUSION AND FUTURE WORK

In this dissertation, we have proposed a process for sound conformance testing of hybrid systems. For that, we have used state-of-the-art results that include the conformance notion proposed in [AHF+14] and the tools presented in [AMR15b] and in [Alt15]. We have also used and implemented the strategy described in [MM16a] to guarantee that our strategy for conformance testing is sound.

This project was conceived when we realised the absence of a structured process with well-defined steps for conformance testing of hybrid systems. This project involved four stages, three of them being research-related and one implementation part. The first research subject was the study of existing conformance testing strategies for hybrid systems. When we had settled on working with (τ, ε)-conformance, the soundness issue was brought to our attention. That was when we shifted the focus to solve this problem through reachability analysis. The third stage was the implementation of our strategy. We have applied our research results regarding reachability analysis and extended an existing prototype tool that already used the (τ, ε)-conformance relation in its testing strategy. The final research component was designing the process itself and its inner aspects, such as finding a suitable sampling rate.

As stated earlier, the proposed process consists of 5 steps: sampling rate computation, parameters definition, reachability analysis, conformance analysis and specification/parameters review. While some steps are manual due to requiring some intricate analysis on the system, it was possible to automate some steps (sampling rate computation, reachability analysis, and conformance analysis) and, thus, they were implemented in the form of a tool extension. The implementation was fully done in Matlab, which is a well known language/environment for control systems analysis. This also allowed for a seamless integration with the external tool we use for reachability analysis (CORA).

We have also shown how our approach can be used discussing possible scenarios, as well as implementing two case studies: the pneumatic suspension system and the AFR controller. We have shown that the tool's performance is inversely proportional to the sampling rate of the system and the complexity seems to be linear. Although we believe the results are satisfactory, there is room for improvement. Next we discuss related work and then some topics for future research opportunities.

5.1 RELATED WORKS

During our literature review, we were not able to find any work that proposes a process for conformance testing as we do here. This is not surprising considering we are dealing with a relatively new subject and state-of-the-art strategies. Thus, we describe here two other conformance notions for hybrid systems that we could have considered, namely hybriod ioco [vO09b] and Approximate Simulation [GJP08]. Later, we also present S-TaLiRo [LFS], which is an-

other tool for conformance testing of hybrid systems, and highlight its key features.

5.1.1 Hybrid Input-Output Conformance

In [VO06], the author presents the hybrid input-output conformance relation (hioco) based on the already existing discrete-event input-output conformance relation of Tretmans [Tre08]. Input-output conformance testing refers to testing whether a system conforms to a model of the system itself based on input and output activity.

In short, hioco states that the output behaviour of the implementation model is a subset of the output behaviour of the specification model, but restricted to what has been specified. In other words, the implementation is free to exhibit any behaviour after inputs that have not been considered by the specification model. Figure 5.1 illustrates the hioco relation.

Model $\mathcal{H}_1$ contains two states S_0 and S_1, and an output trajectory that triggers the transition from the initial state S_0 to S_1. A second and identical output trajectory, originating from S_1, does not cause a change in state. Model $\mathcal{H}_2$ contains a similar behaviour. However, it has one additional output action (a_1), which triggers a transition from the initial state u_0 to state u_1. Subsequently, the output trajectory σ_1 results in a self loop of the state u_1.

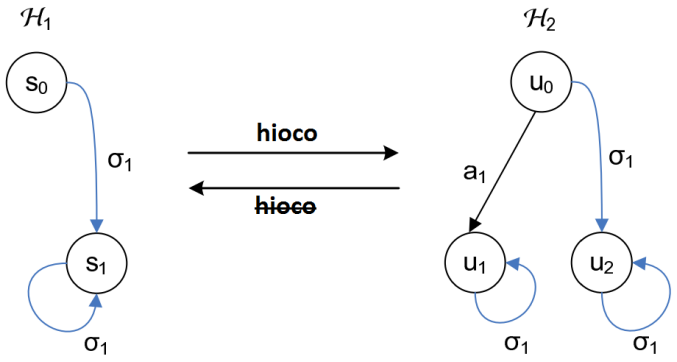


Figure 5.1: hioco example [VO06]

Based on the hioco notion, a bilateral conformance verdict is provided, as observed in Figure 5.1. Considering that all behaviour of $\mathcal{H}_1$ is contained in $\mathcal{H}_2$, $\mathcal{H}_1$ conforms to $\mathcal{H}_2$. Differently, $\mathcal{H}_2$ does not conform to $\mathcal{H}_1$ since the model $\mathcal{H}_2$ includes behaviour that is not contained or permitted in model $\mathcal{H}_1$. Note that, in this example, the non-conforming behaviour is due to the fact that a_1 is an output action. However, after new inputs, the implementation should be free to perform any actions without violating the conformance rules. For instance, if we consider that a_1 is an input, then $\mathcal{H}_2$ conforms to $\mathcal{H}_1$, but $\mathcal{H}_2$ does not conform to $\mathcal{H}_1$.

In hioco, the dynamical behaviour of an implementation should be precisely captured in the specification to establish conformance; any divergence of implementation behaviour (e.g., due to sensor disturbances) is evaluated as a non-conforming behaviour. This property discourages the application of the hioco notion in practice, since disturbances and imprecision are typically part of continuous dynamics.

5.1.2 Approximate Simulation

The authors of [GJP08, GP11] propose a conformance relation based on the output behaviour of CPSs. In this case, however, it deals with systems modeled by Metric Transition Systems (MTSs) [KM15b]. In Figure 5.2, this conformance notion is depicted. In summary, similar to the (τ, ε)-conformance notion, approximate simulation compares the output signals of both implementation and specification models under and decides whether the signals are close enough, based on a closeness notion.

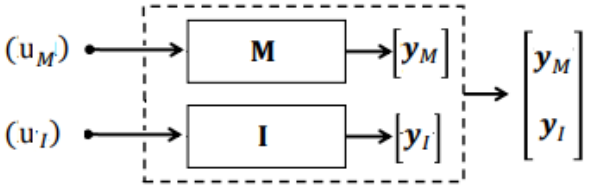


Figure 5.2: Approximate simulation concept (adapted from [AHF+14])

To compare the outputs y_M and y_I, a distance metric needs to be defined over the output space of both systems [GJP08]. The closeness notion that is used to determine how close two output signals are, in terms of valuation, is based on a maximum margin of error ε. Moreover, the output signals y_M and y_I are generated by two corresponding input signals, namely u_M and u_I, respectively. As a result, approximate simulation also requires that both input signals have a maximum difference of γ, similar to the output comparison. Thus, if these conditions are satisfied, then the implementation approximately simulates the specification model with precision (ε,γ). Analogously, to the (τ, ε)-conformance notion, the reliability degree of the verdict is determined by how lenient is the valuation of its parameters.

Thus, compared to (τ,ε)-conformance, Approximate Simulation differs in two ways, namely in its input signal dependency and its closeness notion. Firstly, the test suite executed on both model and implementation are not necessarily the same. Secondly, the closeness notion that is used to determine how near two output signals are (in terms of valuation) is based only on spatial margin of error and does not consider temporal margin of error.

5.1.3 S-TaLiRo

S-TaLiRo [LFS] is a Matlab toolbox that performs temporal verification and conformance testing of Hybrid Systems. In this case, temporal verification is used to prove or falsify temporal logic properties of the system by searching for system behaviours that falsify the specification, i.e., counterexamples to Metric Temporal Logic (MTL) properties.

In Figure 5.3, the tool architecture is depicted. When combined, the Stochastic Optimization Engine block, the Generate Input Signals block, the System Simulator Engine block, and the TaLiRo block form a closed loop. This loop represents the MTL falsification solution. More specifically, the Stochastic Optimization Engine and the Generate Input Signals components form the test case generation block of Figure 5.3, while the TaLiRo block represents the MTL robustness computation [FP09].

As for conformance testing, the tool uses the (τ, ε)-conformance notion, which was proposed by the same authors and, although it focuses on analysing Simulink models, it also woks with hybrid automata. Furthermore, it uses randomised testing based on stochastic optimisation techniques for input selection and test generation.

Although we consider S-TaLiRo as one of the main existing tools for conformance testing of hybrid systems, we have decided to extend the tool presented in [AMR15b] because of the support we have received from the people behind the latter tool. This support led to a joint research work and a cooperation that has been fruitful. Furthermore, S-TaLiRo also uses a slightly different hybrid automata. For instance, the tool does not support reset maps in the automata. It would require some initial work just to support some of the desired features.

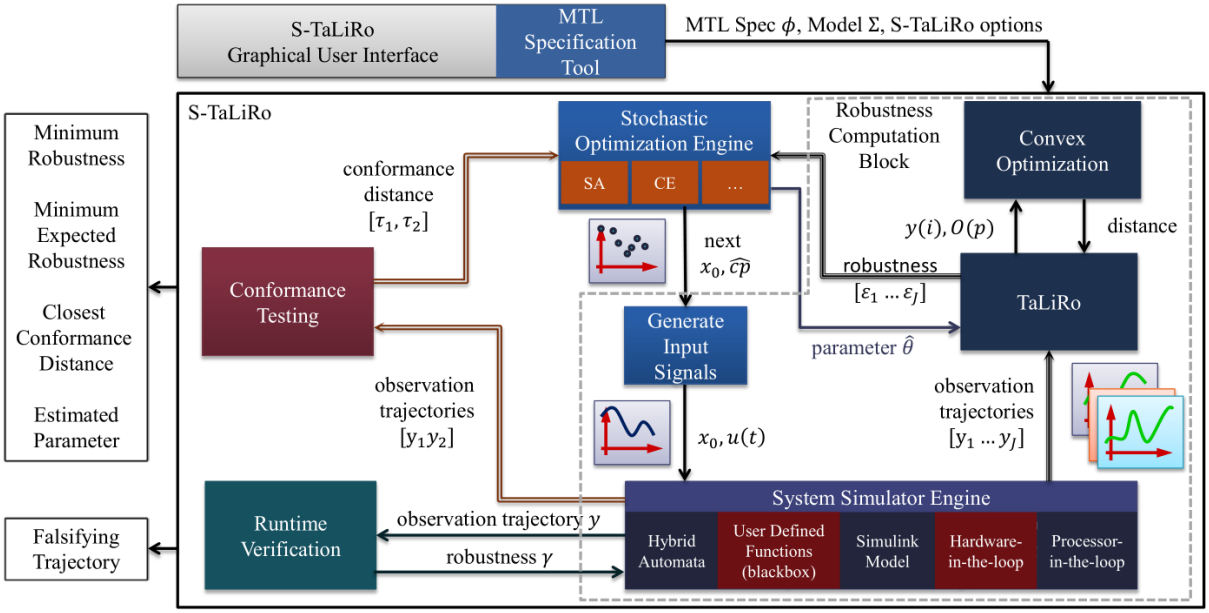


Figure 5.3: S-Taliro architecture [HBA+14]

5.2 FUTURE WORK

In what follows, we outline some future work to complement the research presented here.

- **Build a CORA model from the specification model**
  In the current state of the tool, the interface with CORA requires a separate model for reachability analysis. Although CORA's input model is nearly identical to the model we use for test case generation, there are some distinctions due to the way these tools work internally. Thus, we plan to automate the translation from the hybrid automaton model we use to the one expected by CORA.

- **Support for dynamic sampling rate**
  We believe that with some modifications to our conformance testing strategy and algorithms, we could make use of dynamic sampling. In this case, this technique should

allows us to use different rates to deal with different behaviours of the specification trajectories. For example, it could be possible to assign a sampling rate to each location in the automata and, therefore, compute its corresponding specification maximum change as well. This would allow us to achieve tighter error bounds in general.

- **Facilitate parameter tuning**
  One of the the last steps of our process (Figure 3.1 - Step 4b) is a manual revision of the parameters in order to reduce the error bounds. It is possible to reduce manual labour by performing certain analyses on the system and providing relevant data. For instance, given a fixed margin of error (ε), we could find the minimum sampling rate with which our strategy still guarantees soundness.

- **Integration with NAT2TEST**
  It is in our research agenda to provide test case generation from specifications written in natural language. Thus, we would like to integrate our strategy into the NAT2TEST tool [CBC+15]. This tool already generates test cases from natural language requirements based on different formalisms such as CSP [CSM13] and SCR [CFB+13] and allows simulation of hybrid systems from requirements that express discrete and continuous aspects [OCSM17]. We intend to extend these facilities to support conformance testing of CPS.

- **New conformance relation**
  We plan to define a new conformance relation for hybrid systems. We have noted that the existing relations do not have all the features we are looking for, such as support for partial specifications, discrete actions, margins of error and non-determinism. Certain notions support a few of these features but not all. Hybrid ioco, for example, allows for partial specifications and non-determinism but lacks support for margins of error, which makes it unfit for testing system in practical scenarios.

REFERENCES

[Abb15] Houssam Y Abbas. *Test-Based Falsification and Conformance Testing for Cyber-Physical Systems*. PhD thesis, Arizona State University, 2015.

[ACH+95a] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1):3–34, 1995.

[ACH+95b] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

[ACS+17a] Hugo Araujo, Gustavo Carvalho, Augusto Sampaio, Mohammad Reza Mousavi, and Morteza Mohaqeqi. Sound conformance testing for cyber-physical systems: Theory and implementation. *Science of Computer Programming*, 2017. To Appear.

[ACS+17b] Hugo Araujo, Gustavo Carvalho, Augusto Sampaio, Mohammad Reza Mousavi, and Masoumeh Taromirad. A process for sound conformance testing of cyber-physical systems. In *Software Testing, Verification and Validation Workshops (ICSTW), 2017 IEEE International Conference on*, pages 46–50. IEEE, 2017.

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

[AHF+14] Houssam Abbas, Bardh Hoxha, Georgios Fainekos, Jyotirmoy V Deshmukh, James Kapinski, and Koichi Ueda. Conformance testing as falsification for cyber-physical systems. *arXiv preprint arXiv:1401.5200*, 2014.

[AIH15] Sara Abbaspour Asadollah, Rafia Inam, and Hans Hansson. A survey on testing for cyber physical system. In *IFIP International Conference on Testing Software and Systems*, pages 194–207. Springer, 2015.

[AK11] M. Althoff and 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*, pages 6814–6821, Dec 2011.

[Alt10] Matthias Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. Dissertation, Technische Universitat Munchen, MÃ¼nchen, 2010.

[Alt15] M. Althoff. An introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[Alu11] Rajeev Alur. Formal verification of hybrid systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 273–278. IEEE, 2011.

[AMF14] Houssam Abbas, Hans Mittelmann, and Georgios Fainekos. Formal property verification in a conformance testing framework. In *Formal methods and models for codesign (memocode), 2014 twelfth acm/ieee international conference on*, pages 155–164. IEEE, 2014.

[AMR15a] Arend Aerts, Mohammad Reza Mousavi, and Michel Reniers. A tool prototype for model-based testing of cyber-physical systems. In Martin Leucker, Camilo Rueda, and Frank D. Valencia, editors, *Proceedings of the 12th International Colloquium on Theoretical Aspects of Computing (ICTAC 2015)*, volume 9399 of *Lecture Notes in Computer Science*, pages 563–572. Springer International Publishing, 2015.

[AMR15b] Arend Aerts, Mohammad Reza Mousavi, and Michel Reniers. A tool prototype for model-based testing of cyber-physical systems. In *International Colloquium on Theoretical Aspects of Computing*, pages 563–572. Springer, 2015.

[ARM16] Arend Aerts, Michel A Reniers, and Mohammad Reza Mousavi. Model-based testing of cyber-physical systems. 2016.

[ASB10] Matthias Althoff, Olaf Stursberg, and Martin Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233 – 249, 2010. {IFAC} World Congress 2008.

[BB86] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[BJK+05] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner. *Model-based testing of reactive systems: advanced lectures*, volume 3472. Springer, 2005.

[BMP99] Olivier Bournez, Oded Maler, and Amir Pnueli. Orthogonal polyhedra: Representation and computation. In *Schuppen (Eds.), Hybrid Systems: Computation and Control, LNCS 1569*, pages 46–60. Springer, 1999.

[CB02] Paul Caspi and Albert Benveniste. *Toward an Approximation Theory for Computerised Control*, pages 294–304. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

[CBC+15] Gustavo Carvalho, Flávia Barros, Ana Carvalho, Ana Cavalcanti, Alexandre Mota, and Augusto Sampaio. Nat2test tool: From natural language requirements to test cases based on csp. In *Software Engineering and Formal Methods*, pages 283–290. Springer, 2015.

[CFB+13] Gustavo Carvalho, Diogo Falcão, Flávia Barros, Augusto Sampaio, Alexandre Mota, Leonardo Motta, and Mark Blackburn. Test case generation from natural language requirements based on scr specifications. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1217–1222. ACM, 2013.

[CFMS12] Gustavo Carvalho, Diogo Falcão, Alexandre Mota, and Augusto Sampaio. A process algebra based strategy for generating test vectors from scr specifications. In *SBMF*, pages 67–82. Springer, 2012.

[Che15] Xin Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.

[CSB+06] Jeffrey A Cook, Jing Sun, Julia H Buckland, Ilya V Kolmanovsky, Huei Peng, and Jessy W Grizzle. Automotive powertrain control - a survey. *Asian Journal of Control*, 8(3):237–260, 2006.

[CSM13] Gustavo Carvalho, Augusto Sampaio, and Alexandre Mota. A csp timed input-output relation and a strategy for mechanised conformance verification. In *International Conference on Formal Engineering Methods*, pages 148–164. Springer, 2013.

[DD09] Michel Marie Deza and Elena Deza. Encyclopedia of distances. In *Encyclopedia of Distances*, page 94. Springer, 2009.

[DDD+15] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods Symposium*, pages 127–142. Springer, 2015.

[DMP15] Jyotirmoy V. Deshmukh, Rupak Majumdar, and Vinayak S. Prabhu. *Quantifying Conformance Using the Skorokhod Metric*, pages 234–250. Springer International Publishing, Cham, 2015.

[DSHL+09] B De Schutter, WPMH Heemels, J Lunze, Christophe Prieur, et al. Survey of modeling, analysis, and control of hybrid systems. *Handbook of Hybrid Systems Control–Theory, Tools, Applications*, pages 31–55, 2009.

[FP09] Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[GG08] Antoine Girard and Colas Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proceedings of the 11th International Workshop on Hybrid Systems: Computation and Control*, HSCC '08, pages 215–228, Berlin, Heidelberg, 2008. Springer-Verlag.

[GJP08] Antoine Girard, A Agung Julius, and George J Pappas. Approximate simulation relations for hybrid systems. *Discrete event dynamic systems*, 18(2):163–179, 2008.

[GP11] Antoine Girard and George J Pappas. Approximate bisimulation: A bridge between computer science and control theory. *European Journal of Control*, 17(5-6):568–578, 2011.

[GST09] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93, 2009.

[HBA+14] Bardh Hoxha, Hoang Bach, Houssam Abbas, Adel Dokhanchi, Yoshihiro Kobayashi, and Georgios 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*, 2014.

[HDS13] WPMH Heemels and B De Schutter. Modeling and control of hybrid dynamical systems. *TU/e, Lecture notes course 4K160*, 2013.

[JDK+14] Xiaoqing Jin, Jyotirmoy V Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 253–262. ACM, 2014.

[KGB04] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

[KM15a] Narges Khakpour and Mohammad Reza 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.

[KM15b] Narges Khakpour and Mohammad Reza 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*, volume 42 of *LIPIcs*, pages 18–40. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[LFS] Che Liu, Georgios E Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. Springer.

[LPY01] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, September 2001.

[MM16a] Morteza Mohaqeqi and Mohammad Reza Mousavi. Sound test-suites for cyber-physical systems. In *Theoretical Aspects of Software Engineering (TASE), 2016 10th International Symposium on*, pages 42–48. IEEE, 2016.

[MM16b] Morteza Mohaqeqi and Mohammad Reza Mousavi. Sound test-suites for cyber-physical systems. In *10th International Symposium on Theoretical Aspects of Software Engineering (TASE 2016)*, IEEE Computer Society, 2016.

[MS00] Olaf Müller and Thomas Stauner. Modelling and verification using linear hybrid automata–a case study. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):71–89, 2000.

[MZ16] Pieter J Mosterman and Justyna Zander. Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. *Software & Systems Modeling*, 15(1):5–16, 2016.

[OCSM17] Bruno Oliveira, Gustavo Carvalho, Augusto Sampaio, and Mohammad Reza Mousavi. Simulation of hybrid systems from natural-language requirements. *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017. To Appear.

[Sha49] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[Sha01] Claude E Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[SK03] Olaf Stursberg and Bruce 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, pages 482–497, Berlin, Heidelberg, 2003. Springer-Verlag.

[Tre96] Jan Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer networks and ISDN systems*, 29(1):49–79, 1996.

[Tre08] Jan Tretmans. Model based testing with labelled transition systems. *Formal methods and testing*, pages 1–38, 2008.

[UL10] Mark Utting and Bruno Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.

[VO06] Michiel Van Osch. Hybrid input-output conformance and test generation. In *Formal Approaches to Software Testing and Runtime Verification*, pages 70–84. Springer, 2006.

[vO09a] Michiel van Osch. *Automated Model-based Testing of Hybrid Systems*. PhD thesis, Eindhoven University of Technology, The Netherlands, 2009.

[vO09b] Michiel Pieter Willem Jacob van Osch. *Automated model-based testing of hybrid systems*. Eindhoven University of Technology, 2009.

[Yar10] RK Rao Yarlagadda. *Analog and digital signals and systems*. Springer Science & Business Media, 2010.