THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

On Optimization-Based Falsification of Cyber-Physical Systems

Zahra Ramezani

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2022

On Optimization-Based Falsification of Cyber-Physical Systems

Zahra Ramezani
ISBN 978-91-7905-732-9

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5198
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000
www.chalmers.se

Printed by Chalmers Reproservice
Gothenburg, Sweden, October 2022

To my family

Abstract

In what is commonly referred to as cyber-physical systems (CPSs), computational and physical resources are closely interconnected. An example is the closed-loop behavior of perception, planning, and control algorithms, executing on a computer and interacting with a physical environment. Many CPSs are safety-critical, and it is thus important to guarantee that they behave according to given specifications that define the correct behavior. CPS models typically include differential equations, state machines, and code written in general-purpose programming languages. This heterogeneity makes it generally not feasible to use analytical methods to evaluate the system's correctness. Instead, model-based testing of a simulation of the system is more viable.

Optimization-based falsification is an approach to, using a simulation model, automatically check for the existence of input signals that make the CPS violate given specifications. Quantitative semantics estimate how far the specification is from being violated for a given scenario. The decision variables in the optimization problems are parameters that determine the type and shape of generated input signals.

This thesis contributes to the increased efficiency of optimization-based falsification in four ways. (i) A method for using multiple quantitative semantics during optimization-based falsification. (ii) A direct search approach, called line-search falsification that prioritizes extreme values, which are known to often falsify specifications, and has a good balance between exploration and exploitation of the parameter space. (iii) An adaptation of Bayesian optimization that allows for injecting prior knowledge and uses a special acquisition function for finding falsifying points rather than the global minima. (iv) An investigation of different input signal parameterizations and their coverability of the space and time and frequency domains.

The proposed methods have been implemented and evaluated on standard falsification benchmark problems. Based on these empirical studies, we show the efficiency of the proposed methods. Taken together, the proposed methods are important contributions to the falsification of CPSs and in enabling a more efficient falsification process.

Keywords: Cyber-Physical Systems, Model-Based Testing, Optimization-Based Falsification, Quantitative Semantics, Bayesian Optimization, Input Generators

List of Publications

This thesis is based on the following publications:

[A] **Zahra Ramezani**, Nicholas Smallbone, Martin Fabian, Knut Åkesson, "Evaluating Two Semantics for Falsification using an Autonomous Driving Example". in 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), IEEE, vol. 1, pp. 386–391, 2019.

[B] **Zahra Ramezani**, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, Knut Åkesson, "Multiple Objective Functions for Falsification of Cyber-Physical Systems". IFAC-PapersOnLine, vol. 53, no. 4, pp. 417–422, 2020.

[C] **Zahra Ramezani**, Koen Claessen, Nicholas Smallbone, Martin Fabian, Knut Åkesson, "Testing Cyber-Physical Systems Using a Line-Search Falsification Method". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 8, pp. 2393–2406, 2022.

[D] **Zahra Ramezani**, Kenan Šehić, Luigi Nardi, Knut Åkesson, "Falsification of Cyber-Physical Systems using Bayesian Optimization". Submitted for possible journal publication.

[E] **Zahra Ramezani**, Alexandre Donzé, Martin Fabian, Knut Åkesson, "On Input Generators for Cyber-Physical Systems Falsification". Submitted for possible journal publication.

Other publications by the author, not included in this thesis, are:

[F] **Zahra Ramezani**, Alexandre Donzé, Martin Fabian, Knut Åkesson, "Temporal Logic Falsification of Cyber-Physical Systems using Input Pulse Generators". *EPiC Series in Computing, vol. 80, pp. 195–202, 2021.*

[G] **Zahra Ramezani**, Jonas Krook, Zhennan Fei, Martin Fabian, Knut Åkesson, "Comparative Case Studies of Reactive Synthesis and Supervisory Control". *in 2019 18th European Control Conference (ECC), IEEE,* pp. 1752–1759, 2019.

[H] Johan Lidén Eddeland, Koen Claessen, Nicholas Smallbone, **Zahra Ramezani**, Sajed Miremadi, Knut Åkesson, "Enhancing Temporal Logic Falsification with Specification Transformation and Valued Booleans". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5247–5260, 2020.

[I] Koen Claessen, Nicholas Smallbone, Johan Lidén Eddeland, **Zahra Ramezani**, Knut Åkesson, "Using Valued Booleans to Find Simpler Counterexamples in Random Testing of Cyber-Physical Systems". *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 408–415, 2018.

[J] Koen Claessen, Nicholas Smallbone, Johan Lidén Eddeland, **Zahra Ramezani**, Knut Åkesson, Sajed Miremadi, "Applying valued booleans in testing of cyber-physical systems". *in 2018 IEEE Workshop on Monitoring and Testing of Cyber-Physical Systems (MT-CPS), IEEE* pp. 8–9, 2018.

Acknowledgments

Any story or adventure has a start as well as an end. It is now my turn to reach the end of my PhD journey. I should say that everything in life has a meaning when you look back in time and review everything. In spite of being alone on the way, any individual has much support from colleagues, family, and friends. I list a few of them to thank for their presence in each part of my PhD journey in Sweden.

First and foremost, I would like to express my sincere gratitude to my supervisor Knut Åkesson, for his unique support, patience, and friendship since the beginning of my PhD studies. Without his trust and meticulous approach, it was impossible to proceed and make progress throughout this important and exciting journey. He has continuously been motivating me to be curious and to try new ideas, which concludes in-depth knowledge regarding many scientific concepts. Indeed having such a great opportunity to work with a person like Knut cannot be forgotten.

Secondly, my sincere gratitude goes to my co-supervisor, Martin Fabian. His presence provided me with an incredible opportunity to discuss and analyze different scientific aspects. I always received wise, challenging, and constructive comments from your side, which forced me to be creative and made me think and prepare much better and more mature drafts. I have to add that Martin is not only a good supervisor but also a very supportive boss.

Thousands of thanks and appreciation to Alexandre Donzé for his assistance, support, and guidance during this work. I learned a lot, in particular constructive ideas, a deep understanding of the field, and being creative. I would like to dedicate a special thanks to Koen Claessen for the nice collaboration and discussions. I also greatly appreciate the collaboration that I had with Nicholas Smallbone. I also would like to thank my colleagues in the SyTec project: John Hughes and Mary Sheeran. I would like to express a specific thanks to Johan Lidén Eddeland for very nice discussions and collaborations with the papers and generally in the field.

I would like to thank Luigi Nardi, whom I had a chance to work with, and his group in the last year of my PhD. Especially working with Kenan Šehić was a good opportunity to learn new scientific concepts. Thanks for being available and supportive when I had any technical or even trivial questions. In addition, I would like to thank Zhennan Fei and Jonas Krook for the collaboration that resulted in a paper in a different field from mine.

I would like to express my sincere thanks to all seniors in the Automation group, Bengt Lennartson, Petter Falkman, Emmanuel Dean, Kristofer Bengtsson, and especially Sahar Mohajerani, for her advice. Thanks to all my colleagues, friends, and administrators in the Automation groups and also in the Electrical Engineering Department. One of the best parts of my journey was being in a nice office with kind office mates, Sabino, Constantin, and Mattias. Thanks for all the happiness and good feelings that we had together at the office, in particular the discussions and the warm atmosphere.

I would like to acknowledge my very kind and supportive friends who live here in Göteborg, which makes it my second hometown, and those who are far away from me, but I still feel their presence in my life. A special thanks to my friends who have been with me in happiness and sadness, in particular Roodabeh, Tahereh, Yasaman, Maryam R, and Ali F. Finally, my sincere gratitude goes to Masoumeh and Shirin for their non-stop and unique support.

Now it is time to turn to dedicate my words to two precious persons in my life who gave me the love of life and tranquility, my parents. I truly have to confess that without their emotional support, I could not have finished this journey. Words can not express my feelings, sensations, and emotions toward the persons who dedicated their life to me. I just want to simply say I love you, Mom and Papa!

My appreciation goes beyond the words to thank my dearest sisters, Parisa and Sepideh. Your emotional supports were countless and provided me with happiness and strength. Since childhood, we have had much togetherness in happiness and sadness, playing, studying, and many other occasions. There are always some secrets among close friends that no one knows, these are only some portions of love that my sweetheart sisters gave me. We had true sisterhood that made us inseparable. Last but not least, thanks to Hesam, my brother-in-law, for his support and being a real brother.

This work was supported by the Swedish Research Council (VR) project SyTeC VR 2016-06204 and by the Swedish Agency for Innovation Systems (VINNOVA) under project TESTRON 2015-04893.

The results for all benchmark problems in this thesis can be performed at High Performance Computing Center North (HPC2N), Umeå University, a Swedish national center for Scientific and Parallel Computing; and also resources at Chalmers Centre for Computational Science and Engineering (C3SE) provided by the Swedish National Infrastructure for Computing (SNIC).

Acronyms

AD:	Autonomous Driving
BO:	Bayesian Optimization
CPS:	Cyber-Physical System
NM:	Nelder-Mead
SNOBFIT:	Stable Noisy Optimization by Branch and Fit
STL:	Signal Temporal Logic
SUT:	System Under Test
VBool:	Valued Boolean

Contents

Part I

Overview

CHAPTER 1

Introduction

1.1 Cyber-Physical Systems

Computational and physical resources are closely interconnected in many real-world systems. These systems are commonly referred to as cyber-physical systems (CPSs). CPSs contain both *cyber* and *physical* elements together with sensors and actuators. CPSs collect and evaluate data and communicate with other systems in their environment [1]. Applications of CPSs are found in a wide range of domains [2], factory automation, medical systems, power grids, aerospace, autonomous driving, communications, robotics, etc.

A tool-chain for modeling, simulation, and optimization of CPSs typically needs multiple tools to describe different aspects of the CPSs. Modelica [3] and Matlab/Simscape [4] are equation-based languages tailormade to model the physical parts of a CPS, while Matlab/Simulink [5] is often used to implement control algorithms. General purpose programming languages, e.g. C [6] or Rust [7], are used to implement device drivers and communication protocols, and TensorFlow [8] and PyTorch [9] are libraries used to model machine learning components, for example, related to perception. This also illustrates the challenges of using formal methods to evaluate the correctness of CPSs. Mathematically, the continuous dynamics, for example, the physical plant, can typically be described using continuous differential equations, and the software parts by using discrete state machines. Due to this combination of continuous and discrete dynamics, CPSs are referred to as *hybrid systems*.

The integration of the cyber and physical and the combination of discrete and continuous dynamics makes CPSs complex and challenging to analyze. Additionally, they are often safety-critical and must correctly fulfill expected behavior. Two common methods used to assess their correctness are *formal verification* and *testing*.

To test or verify a system under test (SUT), the expected behavior of the system must be described. The term *specification* is used to describe this behavior. The specification can be expressed in natural language or using mathematical notation. In an automatic evaluation of the specification, variants of temporal logic are typically used since temporal logic allows specifications to be expressed in an unambiguous way.

This thesis considers the problem of testing CPSs, but we start with a brief introduction to formal verification before introducing testing.

1.2 Formal Verification

Formal verification is the process of ensuring the functional correctness of a design by applying mathematical techniques. Essentially, verification aims to find whether a given CPS model does fulfill a formal specification or not. Model checking [10] and deductive verification [11] are two commonly used formal verification techniques.

Model Checking

Model checking [10], [12] refers to the process of determining whether a finite-state model meets a given specification. Typical properties are freedom from deadlocks if invariants hold, or if request-response properties are satisfied.

Model checking requires minimal human intervention and produces a counterexample if the property is not satisfied. For finite-state systems with temporal logic specifications, model checking is a decidable problem. Thus, the correctness of the system with respect to the given specifications can be determined, assuming that the model is correct.

However, model checking has some disadvantages. One is the state explosion problem, where the number of states in a system of sub-systems increases exponentially with the number of sub-systems. An approach to mitigate the state-space explosion problem is to replace a sub-system with an alternative abstract representation having a smaller number of states. Model checking such an abstraction [13] is potentially less computationally expensive compared to model checking the original model. A potential challenge with using abstractions is that an abstracted model might remove behavior relevant for the verification.

Performing model checking of software is challenging because it is often less structured than hardware and concurrent software is often asynchronous. Hence, the state-space explosion is particularly problematic when software is an essential part of the system. Though model checking can be suitable for specific sub-systems where it is both possible and worth the time, in general, it is not a viable option for large-scale CPSs.

Deductive Verification

Deductive verification [14], [15], also called theorem proving, proves the correctness of a system by using axioms and proof rules. In deductive verification, mathematical models of the given system are formulated using appropriate logic, and the formal proof uses mathematical reasoning within a theorem prover. One of the theorem proving tools used for hybrid systems is KeYmaera X [16]. KeYmaera X combines deductive reasoning and computer algebraic prover technology, where differential dynamic logic is used for the model implementation and specifications.

One of the advantages of deductive verification is that it is applicable to both finite-state and infinite-state reactive systems. On the other hand, deductive verification requires logical reasoning skills and must be performed by experts with extensive experience; it is also time-consuming. When deductive verification fails to show if a property holds, the methods generally provide little feedback to explain why. In such a case, it is up to the user to decide whether the problem lies with the property, the system, or the proof itself.

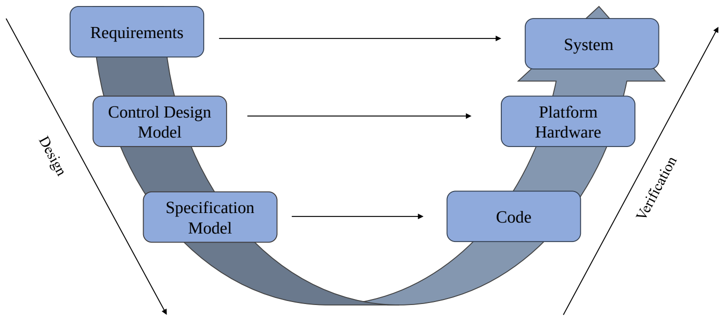

Figure 1.1: The V-model in model-based development design [17].

1.3 Testing

Testing is an essential part of engineering that is widely used in industry to evaluate the quality of embedded systems.

Traditionally embedded systems were developed by manually generating the code and then doing verification and validation using extensive tests, often on the physical system. This approach in developing embedded systems has a number of disadvantages. First, testing is often done at the late stages of development, where it is often time-consuming and expensive to handle identified problems. Second, embedded systems often interact with a physical environment that affects closed-loop behavior. Models of physical environments are often used also for other purposes, for example, for performance optimization and control design. For this reason, the V-model in the *model-based development* (MBD) process for CPSs was developed, see Figure 1.1. This method uses automatic code generation from high-level languages, modeling of formal specifications, and supports simulation-based evaluation at early development stages. The requirements, and models, on the design part, define the behaviors that should be exhibited by the corresponding systems during verification. A control design, including implementation details, is created based on the requirements. This results in a specification model. Code is generated automatically based on the specification model and is then compiled and finally executed on the hardware platform.

The main goal in this thesis is to support the development phases in the left branch of the V-model where the system is evaluated during simulation. This is also where it is relatively easy and cheap to discover and fix potential problems in the design. In the right branch, the physical system is used, of course, verification and validation need to be done at this level as well, but involving physical hardware makes it more expensive to scale up testing.

Conformance Testing

Conformance testing of CPSs, [18], [19] checks whether the observable behavior of a SUT corresponds to a prescribed input-output behavior. CPSs typically contain continuous signals, and during conformance testing, discrete sampling of the continuous signals is necessary. Due to discrete sampling, deviations both in time and value of the signals have to be allowed. In [20], the notion of (τ, ϵ)-closeness is introduced. In [19], the interaction between the continuous dynamics, sampling rates, and error margins are discussed with respect to building sound conformance testing algorithms. The main focus in conformance testing is on establishing whether an implementation fulfills the prescribed input-output behavior, while in this thesis, we actively search for input signals that might violate given properties.

Coverage-guided Testing

For software, the modified condition/decision coverage (MC/DC) [21] is a structural code coverage criteria. MC/DC is recommended to use for the highest Automotive Safety Integrity Level (ASIL D) classification in the standard ISO 26262 [22]. In [23], it is shown that MC/DC fulfillment is sensitive to the code structure. Thus, fulfilling the MC/DC criteria does not necessarily mean that the test suite is sufficient.

For CPSs, coverage-guided test criteria [24], [25] can be used to evaluate the quality of a test suite. Different coverage criteria can be used to evaluate how much of a system's behavior has been explored with a test suite. In [24], the star discrepancy is used to measure how well equidistributed a set of tested points is in the state space. Furthermore, several other coverage metrics are proposed in [26], which also include information about the discrete states of the SUT. In [27], structural coverage-based criteria using hybrid automata for CPSs are proposed. An approach that generates hybrid automata from a Modelica model is presented.

Coverage-guide test criteria can help when to keep generating more test cases. In general, they do not help in determining when to stop generating test cases since fulfilling a coverage test criteria does not imply that no counterexample exists.

Falsification

Falsification based on monitoring the behavior of a system [28] is a practical testing method that can be used both in the early stages of development when only simulations are available but also at later stages when all or parts of the system execute on physical hardware. *Simulation-based* falsification is typically done in the early phases of development [17]. A major advantage of simulation-based approaches is that they can handle all types of systems that can be simulated and that they scale in the sense that a large computer cluster can be used to simulate different instances of the system in parallel. Simulations are used for the design and debugging of embedded control system designs, validating the functional behavior, obtaining initial calibration parameter values, obtaining estimates of system performance, and serving as the basis for the functional and software specifications. Simulation can be applied to large industrial systems at any scale. For many applications, simulation is the only viable option since analytical models are not available for the full system, and even if they were, it would be practically infeasible to use deductive verification due to the extensive expertise and effort needed to apply those methods.

Simulation-based falsification of temporal logic specifications is a testing method that attempts to find *counterexamples* of given specifications. It is a black-box method that relies only on input and output traces of the SUT. Falsification can be used when a model of the system can be simulated, and formal specifications exist. The model of the system approximates the behavior of the SUT. Falsification can be done using optimization-free, e.g. random search or optimization-based methods. Both these methods are considered in this thesis. The optimization-based falsification process uses *quantitative semantics* associated with the specifications. A quantitative semantics [29],

[30] is used to define an objective function that is evaluated for given inputs and results in a measure of the distance to the specification being falsified. The problem of finding new test cases in the optimization-based approach uses the objective function values from previous simulations and modifies the test cases such that the new test case is expected to be more likely to falsify the specification.

For optimization-based falsification, to optimize over the parameters, the input signal needs to be parameterized. For example, a sinusoidal signal can be defined by two parameters, amplitude, and frequency. Piecewise constant signals take different values at different times, which can be parametrized using the values and the times at which the signals change. The input parameters are typically chosen within defined ranges to reduce the search space. The objective function, based on a temporal logic formula, is later evaluated based on a simulation of the system with the chosen parameters. In this thesis, we refer to a vector of input parameters as a *point* in the parameter space.

Monitoring of properties or specifications [30] can be done offline or online. Offline monitoring means that the entire signal is available before evaluating if a specification is violated. Online monitoring means that the evaluation is done while the SUT is running. Online monitoring is typically used when implemented in a real system, but offline methods can be used when the SUT is simulated. In this work, we use offline monitoring since the SUT is simulated; thus, we allow the simulation to complete before we evaluate a specification.

Formal verification aims to prove that a CPS model is free of failures. The purpose of test generation is, on the other hand, to provide evidence of failures if they exist. These two methods are compared in the following section.

Testing versus Formal Verification

For many reasons, a formal verification approach is not practically feasible for industrial systems. Many industrial systems are not available as a formal model suitable for formal verification [17]. Formal verification of large systems might be intractable even if formal models exist due to the verification algorithms' time and space complexity [10], [12]. It is, in general, an undecidable problem [31] to verify systems exhibiting both discrete and continuous dynamics. Formal verification is feasible and practical for systems with restricted behavior and/or size that either have a finite state space or a special structure that makes the verification problem decidable. Moreover, for many industrial applications, it is only possible to simulate the SUT. Thus, formal verification is not a viable option. Hence, this thesis considers testing of CPSs using the falsification approach.

There are limitations to the simulation of large-scale CPSs. Simulations are expensive to run, and the simulation time is also a limiting factor. Thus, it is important to enhance the capability of falsification by reducing the number of needed simulations. This thesis tackles the problem of how to enhance the falsification process for CPSs.

1.4 Tools for Testing of CPSs

CPSs typically exhibit a complex mix of continuous and discrete behavior. In practical testing, it is often assumed that a *black-box* model of the SUT is available. Thus, there is no internal information about the model available, only the input-output behavior of the SUT can be observed.

Below we introduce the academic tools, FalStar, Falsify, S-TaLiRo, ARIsTEO, FalCAuN, Breach, ForeSee, VerifAI, and HyConf, available for testing of CPSs.

FalStar [32] is a falsification tool that explores the idea of constructing the inputs incrementally in time where a probabilistic search is implemented using increasingly fine-grained spatial and temporal discretizations of the input space. Different probabilistic algorithms are implemented, like a two-layered framework combining Monte-Carlo tree search [33], a probabilistic algorithm [32], and adaptive Las-Vegas tree search [34]. This tool uses Signal Temporal Logic (STL) [30] to model the specifications.

Falsify [35] is an experimental tool that implements two reinforcement learning algorithms, Asynchronous Advantage Actor-Critic and Double Deep-Q Network, to learn the system's behavior by observing the outputs during the simulation. The attempt is to reduce the number of simulation runs necessary to falsify the SUT. STL is used in this tool to model the specifications.

S-TaLiRo [36], [37] is a tool that can perform falsification of Metric Temporal Logic (MTL) [29]. S-TaLiRo also supports parameter mining [38] and runtime monitoring [39]. S-TaLiRo uses stochastic optimization algorithms and has recently also included methods based on Bayesian Optimization methods like the Stochastic Optimization with Adaptive Restarts (SOAR) [40]; min-SOAR [41], an extension of SOAR that can be used for multiple conjunctive requirements.

ARIsTEO [42] is developed on top of S-TaLiRo. This tool is designed for computer-intensive CPS models, a large category of CPS models that may take several hours to simulate. A surrogate model that is faster to run than the original model is generated from sampled inputs and outputs of the SUT. The surrogate model is then subjected to black-box testing. The original model is checked when a failure test is identified for the surrogate model. Surrogate models are refined based on test results if the identified failure shows to be spurious. Otherwise, the test indicates a valid failure.

FalCAuN [43], is a tool that uses automated testing based on active learning and model checking [44] for testing MATLAB/Simulink models. FalCAuN discretizes the Simulink model's inputs and outputs, both in time and in value, and approximates the black-box model with a learned Mealy machine. Reusing the learned Mealy machine, this tool can falsify Simulink models against multiple STL specifications. A counterexample is found by FalCAuN by learning the Mealy machine and conducting model checking.

Breach [45] is a MATLAB/Simulink toolbox for formal monitoring of STL specifications that include different search methods for falsification together with optimization-based falsification methods. Breach also supports the mining of requirements [46]. In [47], a method to automatically translate specifications from Simulink charts to STL is presented. Since engineers are typically more proficient in using high-level domain specific languages like Simulink than in writing specifications in STL directly, this automatic translation is important in allowing Breach to be applied to industrial problems.

ForeSee [48] is implemented on top of Breach to solve the scale problem that can occur when the signals in the given specification have different scales. Then, when the objective function value is calculated, the contribution of a signal can be masked by others. In this tool, a combination of quantitative robustness and classical Boolean satisfaction is suggested, called QB-robustness.

VerifAI [49] is a software toolkit that supports the design and analysis of systems that include artificial intelligence (AI) and machine learning (ML) components. VerifAI is based on simulation-based falsification guided by formal models and specifications. Scenic [50] is used as a probabilistic programming language for modeling environments which is important, for example, when working with systems where perception is a vital part.

HyConf [51] is a MATLAB-based tool for conformance testing. This tool covers the test-case generation, test-case execution, and conformance analysis. In order to calculate sound conformance analysis margins [19], this tool interacts with the reachability analysis tool CoRA (COntinuous Reachability Analyzer) [52].

Breach has been used for the implementation of the proposed methods in this thesis. Breach has a modular architecture that allows implementing different quantitative semantics and optimization and is well documented, and is actively being developed. Breach has also been shown capable of handling the falsification of large-scale industrial problems. In particular, the use of the automatic translation to STL from specifications in Simulink has made it possible to apply Breach to industrial problems. S-TaLiRo has similar characteristics as Breach, and we believe the methods in this thesis can also be implemented on top of S-TaLiRo.

1.5 Research Questions

The goal of this thesis can be summarized in the four research questions below.

RQ1. *What are the strengths and weaknesses of the different quantitative semantics for the falsification of cyber-physical systems?*

In optimization-based falsification, it is necessary to estimate a distance to falsifying a specification, this estimation is computed using a quantitative semantics. The efficiency of the falsification is affected by which quantitative semantics that is used. Thus, it is important to understand the strengths and weaknesses of different quantitative semantics.

RQ2. *How does the combination of using different quantitative semantics affect the efficiency of falsification of cyber-physical systems?*

The main purpose of calculating an objective function value using a quantitative semantics during falsification is to guide the falsification process towards a point in which the specification is not satisfied. This is done by choosing the next set of parameters for the input signals to the system, such that those input signals are closer to falsifying the specification, if possible. The purpose of this research question is to understand how the combination of different quantitative semantics can be used to improve the falsification process.

RQ3. *How do different test-case generation methods affect the efficiency of the falsification process?*

Falsification of temporal logic properties is a testing method that may or may not formulate the problem of generating test cases as an optimization problem. Falsification can be performed using either optimization-free or optimization-based methods. In optimization-free methods, a quantitative semantics is not needed. Optimization-based falsification generates new input parameters that aim to find a lower objective function value as estimated by the quantitative semantics. The used optimization method affects the efficiency of the falsification process. This research question aims to evaluate the performance of both different optimization-free and optimization-based methods and to understand how they work and can be best used during the falsification process.

RQ4. *How do different input parameterizations affect the falsification process?*

Falsification of temporal logic properties is a black-box approach where only the input-output behavior of the SUT is available. Defining suitable input parameters is challenging. First, expert knowledge of the SUT is needed to define input signals that are relevant for the particular application. A drawback with being liberal in what kind of input signals are allowed is that signals that violate the specifications may be found during the falsification process but not considered to be possible or relevant by the engineers. It is often an arbitrary process to choose the number of control points or the interpolation scheme to generate an input signal. The way control points are set, and the number of control points can affect the success or failure of the falsification process. Moreover, having few parameters is important for optimization-based methods because optimization algorithms typically struggle with a high number of dimensions. Still, the input generators are needed to be rich enough so that they can describe the input signals that are necessary to falsify the specifications. This research question aims to evaluate the effect of different input generators on the falsification process.

1.6 Methodology

The main purpose of the falsification methods for CPSs is to find bugs and faults in the system with little manual work for the engineers. The purpose of the research presented in this thesis is to improve the understanding of the falsification problem and enhance the capabilities of the falsification process. The research in this field is typically based on the experimental evaluation done on benchmark problems. This thesis focuses on practical evaluations, particularly evident in the formulation of research questions 3 and 4. The research area of testing of CPSs is also heavily application dependent. Hence, it has been an important goal to conduct research that can be applied to large-scale and complex systems rather than only academic examples. However, simple systems are easy to analyze and can give valuable insight into how and why different methods perform better than others. In our case, a simple example resulted in the formulation of the first research question, and after answering this research question, research question 2 was formulated.

Method

Different testing methods and tools for CPSs have been introduced. Simulation-based falsification is picked as the testing method in this thesis because, for many large-scale systems, the only viable option for evaluation is simulation.

The reason to focus on quantitative semantics in Paper A and Paper B was that when the journey of this thesis started, a new quantitative semantics was introduced, and it was not clear how different quantitative semantics affected the falsification performance. After further investigations, the effect of the optimization methods and input generators became clear, which caused the research work in Paper C, Paper D, and Paper E.

All evaluation results in the appended papers are done on problems that have been implemented in MATLAB. An example of an adaptive cruise controller for autonomous driving from a MATLAB example is considered in Paper A, together with a specification that was developed as part of this work. The benchmark problems in Paper B are a collection of various problems collected from different publications and benchmarks [26], [53], [54]. However, some specifications have been evaluated with varying time intervals in their temporal formula and simulation times or different input ranges or model parameters in order to simplify or make the problems harder to falsify but also to create a larger set of falsification problems. However, after writing Paper B, we have used the standard benchmark problems that were used in the ARCH19 workshop [55]. In Paper C and Paper D, these benchmark problems plus variants of those problems and additional problems from Paper B are considered. Paper E considers all benchmark problems in Paper C and a few additional specifications in the updated version of the ARCH21 workshop [56].

For the implementation of the examples in this thesis, Breach [45], which is a MATLAB toolbox supporting falsification problems, has been used. Some used optimization methods, the used Bayesian optimization methods in Paper D and Paper E, were implemented in Python. Hence, in the Paper D and Paper E, Python code interacts with MATLAB. In this implementation, the optimization process is done in Python, and the simulation and evaluation of the objective function are done in MATLAB.

Running the evaluated benchmark problems in the appended papers sequentially takes a lot of time. Hence, all benchmark problems and the proposed methods in Paper C, Paper D, and Paper E are executed on a computing cluster which makes it possible to evaluate them in parallel.

The appended papers, particularly papers B, C, D, and E, consider the evaluated algorithms' randomness properties; hence, each problem and specification are executed in a number of runs. 20 runs are considered in papers B, C, and D, while 5 runs are considered in Paper E. The last paper uses fewer runs because of the need to run many experimental sets in limited time.

1.7 Contributions

In Figure 1.2, the overall optimization-based falsification process is shown to illustrate how the contributions connect together.

Contribution 1. This work investigates different quantitative semantics to increase our understanding of how they affect the falsification process. This investigation leads to the proposal of using multiple quantitative semantics during falsification. The Nelder-Mead [57] optimization method was extended to support the use of several quantitative semantics.

Contribution 2. A direct-search based optimization method suitable for falsification is proposed. The method, called *line-search falsification*, combines randomly generated lines in the parameter space and local search and also emphasizes exploring parameter values on the bounds of the allowed range.

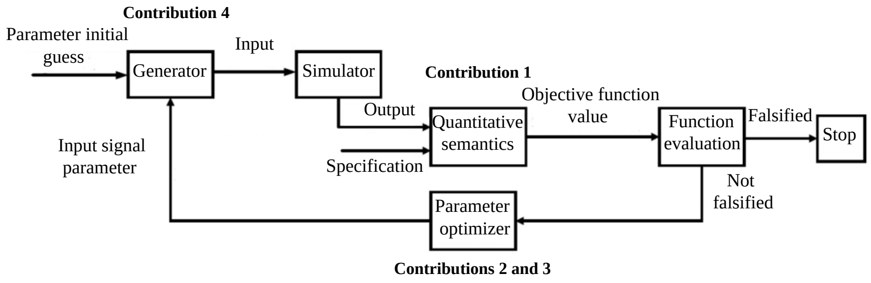

Figure 1.2: The optimization-based falsification process to show the contributions of this thesis.

Contribution 3. Getting the attraction of the falsification community for an efficient class of model-based optimization methods, two different methods have been evaluated. The first is a method that can be used for high-dimensional CPSs evaluated in this work. The second one allows injecting prior knowledge about promising parameter values to emphasize. In particular, we use it to emphasize parameter values that are close to the upper or lower bound of the allowed parameter range.

Contribution 4. An investigation of different input generator's parameterization and evaluation for falsification of CPSs to address the shortcoming of their effect on falsification. This leads to proposing a new parameterization for input signals. Included is also an analysis of coverage measures in the space and time domains and frequency domains of the proposed input generators.

1.8 Outline

This thesis consists of two parts. Part I is a general introduction and overview of the field and gives the readers the understanding needed for the appended papers. Part II contains the appended papers. Part I is organized as follows: Chapter 2 presents the concept of falsification of temporal logic properties. This chapter includes details about Breach [45], the testing tool used in this thesis. Chapter 3 introduces the specification formalisms, the concept of Valued Booleans (VBools), and the different semantics used in this thesis. In Chapter 4, the test generation methods, including both optimization-free and optimization-methods are presented. Chapter 5 describes how to generate input generators for falsification. Chapter 6 introduces the benchmark problems evaluated in the attached papers. Chapter 7 contains a summary of the appended papers. The thesis ends with conclusions in Chapter 8.

CHAPTER 2

Falsification of Cyber-Physical Systems

The falsification process is a strategy that searches for a counterexample where a system does not fulfill its specification. Falsification can be done using optimization-free or optimization-based methods. In the optimization-based approach, falsification uses an objective function defined by a *quantitative semantics* for the temporal logic formalisms. This objective function estimates the distance to the specification being falsified, which is used to guide the process towards falsification by choosing the next set of input points.

The falsification problem is formulated as:

- **Given**: a system *model* $\mathcal{S}$, and a *specification* φ.
- **Results for falsification**: a *counterexample* that falsifies the specification, φ, if found in the falsification process.

To find a counterexample, the falsification process is formulated as an optimization problem by parameterizing the input signal domain, $x \in \mathcal{P}_u$, such that its corresponding output $\mathcal{S}(x)$ falsifies the specification φ, if such input exists, i.e., $\mathcal{S}(x) \nvDash \varphi$. Otherwise, the falsification process is not successful, which does not mean that faults do not exist.

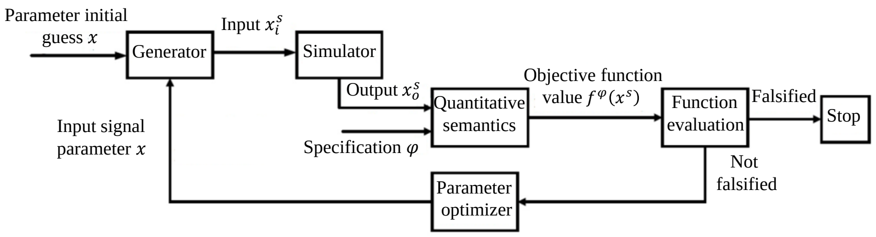

Figure 2.1: A flowchart, from [58], describing the optimization-based falsification procedure in Breach.

2.1 Optimization-based Falsification

The falsification procedure for the optimization-based approach that is used in this thesis is presented in Figure 2.1.

Input Generators

To generate the input trace $x_i^s[k]$ at a time k to the SUT, the *Generator* needs to take input parameters x (points), where each element is allowed to be within a defined range. The index k ranges from the start to the end of the simulation on a discretized time model. The space of permissible input signals is parametrized by m input parameters $a = (a_1, \dots, a_m)$ that take values from the set $\mathcal{P}_u$. The actual input $x_i^s[k]$ is created using a generator function g such that $x_i^s[k] = g(v(\mathbf{a}))[k]$, where $v(\mathbf{a}) \in \mathcal{P}_u$ is a valuation of the parameter vector $\mathbf{a}$. Often multiple parameters are used to parameterize each input signal, resulting in the dimension of x_i^s being lower than the dimension of x.

To interact with the system, different parameterized input generators can be used. For instance, the parameters can represent control points, with the input being generated by interpolation between these points. Also, it is possible to have variable step inputs, and different numbers of control points and interpolation methods, like *pchip* or *linear* interpolation. On the other hand, parameters can represent, for example, amplitude and frequency to define a sinusoidal signal. The effect of the input generators is considered in Paper E and discussed in Chapter 5.

Simulation

When the SUT is available and the input $x_i^s[k]$ is generated by a Generator, the *Simulator* is used to generate the corresponding output simulation trace x_o^s. Specifications are evaluated by using both the input and output signals, and the combination of x_i^s and x_o^s is denoted by trace x^s.

Quantitative Semantics

To determine whether a specification φ is satisfied or not, the input x_i^s and output vectors x_o^s are used with an objective function $f^\varphi(x^s)$. The quantitative semantics discussed in Chapter 3 define this objective function. A negative objective function value means the specification is falsified, and a positive value shows it is not falsified. If the specification is not falsified for the current input signal, the semantics will also give a value of how convincingly the test passed.

The current input and output traces set is a counterexample if the specification is falsified. Therefore, the process of falsification is terminated. In the sequel, we will examine the situation where the specification is not falsified for the current input signals and the goal is to find input traces that result in a lower objective function when evaluated using the quantitative semantics and possibly falsify the given specification.

Function Evaluation

Given φ, the traces x^s, and the choice of quantitative semantics, the evaluation of x^s will return $\langle v, z\rangle$, where v denotes if φ is falsified or not, and z is the measure.

Define $f^\varphi(x^s)$ such that

$$f^\varphi(x^s) = \begin{cases} z \;\; \mathit{if} \;\; v = \top \\ -z \;\; \mathit{if} \;\; v = \bot \end{cases}$$

where $\top$ and $\bot$ denote *true* and *false*, respectively. In this thesis, with slight abuse of notation, $f(x)$ will be considered as shorthand for $f^\varphi(x^s)$, where x_i^s is defined by the *Generator* by using the parameters x, and x_o^s is the output of the *Simulator* with x_i^s as the input trace.

The falsification procedure will stop when $f^\varphi(x^s)$ is negative. Different quantitative semantics are defined in the literature as *Max* [59], *Additive* [58], Mean Alternative Robustness Value *(MARV)*, Root Mean Square Alternative Robustness Value [60], and averaged STL [61]. *Max*, *Additive*, and *MARV* used in the papers appended to this thesis are introduced in Chapter 3. The performance of the quantitative semantics depends on the system and the specification. Papers A and B evaluate the effect of different quantitative semantics.

Parameter Optimization

If the objective function value $f^{\varphi}(x^s) \geq 0$, this means that the specification is not falsified; then new parameter values are selected and the process is repeated. This is done in the *Parameter optimizer* procedure in Figure 2.1. The Parameter optimizer tries to generate new parameter values x within given ranges to optimize the objective function f^{φ} to find lower objective function values.

Much research has been devoted to parameter optimization for falsification. In [62], the gradients are used to guide the optimization. On the other hand, the gradient-free optimization approaches for testing include Ant Colony Optimization [63] and Local Stochastic Tabu search [64]. In [65], the optimization methods, SNOBFIT [66], Nelder-Mead (NM) [57], Simulated Annealing [67], and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [68]; were evaluated on benchmark problems to evaluate the effect of optimization on falsification problems. The evaluation shows that there is a significant difference in the efficiency of the falsification process, with SNOBFIT being the overall top-performing optimization algorithm. In HyConf [51], simulated annealing and genetic algorithm heuristics are used in a multi-objective strategy [69] that maximizes the distance between the output of the system and its ideal target, achieves structural coverage by using control elements of hybrid system models, and generates additional tests covering different areas and shapes by using a diversity metric. The effect of the optimization method is considered in Paper C, where a new gradient-free optimization is suggested. NM and SNOBFIT, both considered in this thesis, are presented in Chapter 4. Moreover, Bayesian optimization (BO) is presented in Chapter 4 and evaluated in Paper D.

CHAPTER 3

Quantitative Semantics

In this chapter, the specification formalisms and quantitive semantics are discussed. A quantitative semantics provides a value that is used by the optimizer to decide on the next parameter values to evaluate. In a general sense, all defined quantitative semantics are equally good in the sense that a specification is satisfied or not, by all valid quantitative semantics However, the efficiency of the optimization-based falsification process depends on which quantitative semantics is used. It is, in general, not analytically possible to decide which semantics is the best because it depends on the SUT and the specification.

The concept of Valued Booleans (VBools) for supporting multiple quantitative semantics is introduced. Some quantitative semantics are evaluated on an autonomous driving-related example.

3.1 Specification Formalisms

To test CPSs, the expected behavior of the system, i.e., the desired properties the system must satisfy, needs to be defined and expressed in a mathematically precise way. The term *specification* describes the properties that must be fulfilled by the SUT.

Different mathematical formalisms might be used to model specifications. Linear Temporal Logic (LTL) [70] is a modal logic proposed to specify the properties of programs. LTL consists of a finite set of propositions, the logical and temporal operators. Metric Temporal Logic (MTL) [29] is an extension of LTL that expresses timing constraints on temporal operators. Metric Interval Temporal Logic (MITL) [71] is a fragment of MTL where the temporal operator interval is not allowed to be a single time-point. Signal Temporal Logic (STL) [30] extends MITL, used for expressing the temporal properties of dense-time, real-valued signals. The specifications can also be defined using high-level languages. In [47], it is described how STL specifications can be automatically generated from Simulink charts, supporting engineers in the specification of formal specifications without being trained in temporal logic.

STL cannot sufficiently distinguish certain oscillatory properties; hence STL* is suggested in [72] as an extension of STL. STL* is a logic formalism that includes an additional freezing operator that can distinguish variously oscillatory properties. A modified version of STL called averaged STL [61] where two time-averaged temporal operators ($\Box$ and $\Diamond$) are introduced and have a different robustness value, i.e., how far away the specification is from the be falsified, than the standard quantitative semantics. Time-Frequency Logic [73] is a specification formalism for real-valued signals that combines temporal logic properties in the time domain with frequency-domain properties.

Max semantics refers to the standard STL quantitative semantics, defined in [74]. Instead of only checking the boolean satisfaction of an STL formula, the notion of a quantitative value, i.e., an objective function value, also known as the *robustness value*, will be defined to measure how far away a specification is from being falsified. Two *Max* and *Additive*, which are expressed using VBools, will be introduced in the next section.

3.2 Quantitative Semantics Expressed in Valued Booleans

A VBool [58] $\langle v, z \rangle$ is a combination of a Boolean value v together with a real number z that is an estimate of how true or false the specification is. This real value z will be used to measure how convincingly a test passed or how severely it failed. In the VBool definition, the real value z is always defined to be non-negative, possibly infinitely large, so the domain is $\mathbb{V} = \mathbb{B} \times \mathbb{R}_{\geq 0}$.

VBools are an abstraction that can be used to provide different quantitative semantics to a formula in STL. Two quantitative semantics, *Max* and *Additive*, are expressed using VBools. Both semantics are defined below to show their formulation in detail. For these two semantics, we define the operators *and* $\wedge$, *or* $\vee$, *always* $\Box_{[a,b]}$, *eventually* $\Diamond_{[a,b]}$, and *until* $\mathcal{U}_{[a,b]}$ [30].

Max Semantics

Using VBools, the Max-*and* operator is defined as:

$$\begin{aligned}(\top, s) \wedge_{\text{Max}} (\top, z) &= \Big(\top, \min(s, z)\Big),\\ (\top, s) \wedge_{\text{Max}} (\bot, z) &= (\bot, z),\\ (\bot, s) \wedge_{\text{Max}} (\top, z) &= (\bot, s),\\ (\bot, s) \wedge_{\text{Max}} (\bot, z) &= \Big(\bot, \max(s, z)\Big).\end{aligned} \tag{3.1}$$

The first case expresses how to evaluate a conjunction where both VBools are true. The rationale behind setting the value in this way to take the minimum of the two real values is that the value represents a distance to how true each part of the conjunction is closer to falsifying their conjunction.

The two middle cases express the conjunction of two VBools where one is true, and one is false. The conjunction will be false, and the value of the real number is given by the false VBool. In the fourth case, both VBools are false. The conjunction will be false, and the value of the VBool is determined by the maximum of the respective real numbers. Similar to when both VBools are true, here, the highest real value shows the shortest distance that a specification is falsified.

The Max-*or* operator is defined in terms of Max-*and* as:

$$(v_s, s) \vee_{\text{Max}} (v_z, z) = \neg_{\text{v}}(\neg_{\text{v}}(v_s, s) \wedge_{\text{Max}} \neg_{\text{v}}(v_z, z)).$$

where VBools negation is defined as $\neg_{\text{v}}(v_s, s) = (\neg v_s, s)$.

The Max-*always* operator over the interval $[a, b]$ is defined in terms of the

Max-*and* operator as:

$$\square_{\mathrm{Max,[a,b]}}\, \varphi = \bigwedge_{k=a}^{b}{}_{Max}\ \varphi[k], \tag{3.2}$$

where φ is a finite sequence of VBools defined for all the discrete-time instants in [a, b].

The timed Max-*eventually* operator is defined in terms of Max-*always* as:

$$\Diamond_{Max,[a,b]}\varphi = \neg(\square_{\mathrm{Max,[a,b]}}(\neg_{\mathrm{v}}\, \varphi)).$$

Finally, the *Max until*-operator as:

$$\begin{aligned} &\varphi\ \mathcal{U}_{Max,[a,b]}\ \psi \\ &= \bigvee_{k=a}^{b}{}_{Max} \left(\psi[k] \wedge_{Max} \left(\bigwedge_{k'=a}^{b-1}{}_{Max}\ \varphi[k']\right)\right). \end{aligned}$$

Additive Semantics

The second semantics expressed in VBools is *Additive*. The operators of Additive-*and*, Additive-*or*, Additive-*always*, Additive-*eventually* and *Additive until* will be introduced in this section.

The Additive-*and* operator is defined as:

$$\begin{aligned} (\top, s) \wedge_{\mathrm{Additive}}(\top, z) &= \left(\top, \frac{1}{\frac{1}{s}+\frac{1}{z}}\right), \\ (\top, s) \wedge_{\mathrm{Additive}}(\bot, z) &= (\bot, z), \\ (\bot, s) \wedge_{\mathrm{Additive}}(\top, z) &= (\bot, s), \\ (\bot, s) \wedge_{\mathrm{Additive}}(\bot, z) &= \Big(\bot, (s+z)\Big). \end{aligned} \tag{3.3}$$

The first case expresses conjunction where both VBools are true. In this case the real value of the VBools is determined as $\frac{1}{\frac{1}{s}+\frac{1}{z}}$. This formula is inspired by the formula for parallel resistance that gives a value that is less than the maximum of s and z. The two middle cases express the conjunction of two VBools where one is true, and one is false. This conjunction is false, and the real number is given by the false VBool. The fourth case expresses a situation where both VBools are false. This conjunction is false, and the real number of the VBool is defined as $s+z$. In this formula, we consider the sum of s and z, not just whichever of them is the largest.

The Additive-*or* operator is defined as:

$$(v_s, s) \vee_{\text{Additive}} (v_z, z) = \neg_{\text{v}}(\neg_{\text{v}}(v_s, s) \wedge_{\text{Additive}} \neg_{\text{v}}(v_z, z)).$$

The Additive-*always* operator over the interval [a, b] is defined in terms of the Additive-*and* operator as:

$$\Box_{\text{Additive,[a,b]}}\, \varphi = \bigwedge_{k=a}^{b}{}_{Additive}\, \varphi[k] \,\#'\, \delta t, \tag{3.4}$$

where φ is a finite sequence of VBools defined for all the discrete-time instants in [a, b]; δt is the simulation step size that makes the quantitative value independent of the simulation time, and $\#'$ is:

$$\begin{aligned}(\bot, s) \,\#'\, \delta t &= (\bot, s \cdot \delta t)\\ (\top, s) \,\#'\, \delta t &= (\top, s/\delta t).\end{aligned}$$

The timed Additive-*eventually* operator is defined in terms of always as:

$$\Diamond_{Additive,[a,b]} \varphi = \neg(\Box_{\text{Additive,[a,b]}}(\neg_{\text{v}}\, \varphi)).$$

The *Additive until*-operator is defined as:

$$\begin{aligned}&\varphi\, \mathcal{U}_{Additive,[a,b]}\ \psi\\ &\quad= \bigvee_{k=a}^{b}{}_{Additive} \left((\psi[k]\#'\delta t) \wedge_{Additive} \left(\bigwedge_{k'=a}^{b-1}{}_{Additive} (\varphi[k']\#'\delta t) \right) \right).\end{aligned}$$

The *implication* is defined slightly differently from classical logic:

$$\phi \rightarrow_{Additive} \psi = \neg(\phi\#k) \vee \psi.$$

Here k is an arbitrary constant, and # scales the robustness of its argument:

$$(\bot, s)\#k = (\bot, s \cdot k)$$
$$(\top, s)\#k = (\top, s \cdot k).$$

The left-hand side of the implication is scaled, so the parameter optimizer will attempt to make the left-hand side true before falsifying the right-hand side.

The quantitative semantics *MARV* [60] used in Paper A is not expressed using VBools. Only the timed always operator is defined for *MARV*. The reader is referred to this paper for detail of how it is formulated.

The behavior of the quantitative semantics, *Max* and *MARV*, are discussed in Paper A, but Paper A does not include a discussion about the *Additive* semantics. In the next section, we include a comprehensive discussion of the *Additive* semantics on the autonomous driving example to prepare the reader for the multiple objective functions used in Paper B.

3.3 Evaluating Different Quantitative Semantics Using an Example

We consider a simple system to gain insight to evaluate and better understand the performance of different approaches. The adaptive cruise controller (ACC) from Paper A will be used to illustrate the differences between different quantitative semantics. In this example, there are two cars, the *ego* car is an autonomous car, and the *lead* car is in front of it. The ACC is designed with two modes; *speed* mode, where there is the desired speed, i.e., cruise control speed, and *safety* mode, where a safe distance between the cars must be maintained. The *relative distance* (d_{rel}) between two cars must always be greater than the safe distance (d_{safe}) to avoid collision. This property can be expressed by the following formula:

$$\Box_{[0,T]}(d_{rel} > d_{safe}), \tag{3.5}$$

where (d_{safe}) is from paper [75], see also Paper A. In this evaluation, no optimization method is used, instead, the objective function values of the different quantitative semantics are calculated using a discrete grid of possible input parameters. The aim of this evaluation is not to find the counterexamples but to

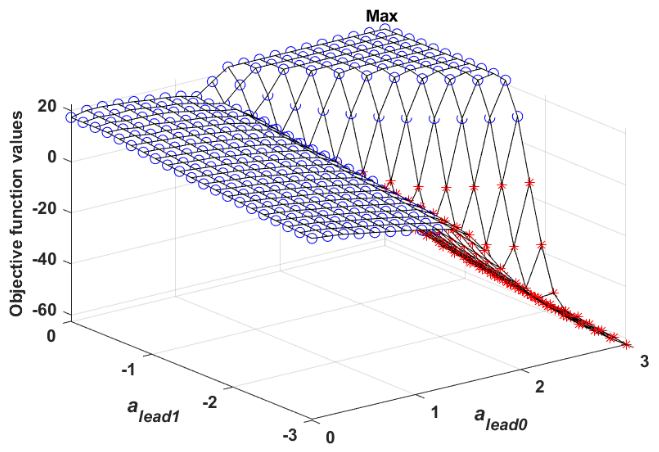

Figure 3.1: Objective function values for combinations of accelerations (a_{lead0}, a_{lead1}) using *Max* semantics, presented in Paper A. Positive values (○) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified.

understand how objective values change for different quantitative semantics.

The example uses the acceleration of the lead car a_{lead} as input to the simulation. Before a simulation of the closed-loop system starts, the values of the input parameter are selected by the falsification algorithm. This input is generated using two control points, a_{lead0} and a_{lead1}. The simulation time is $T = 30\,s$ and the simulation starts with a_{lead0} chosen in the range $[0; 3]$. At time $7.5\,s$, the acceleration changes and takes a value a_{lead1} in the range $[-3; 0]$. The objective function value will be calculated for different combinations of the parameters a_{lead0} and a_{lead1}. Each parameter is discretized into 20 equidistant points, resulting in 400 simulations.

The objective function values, using *Max*, *MARV*, and *Additive*, and gradient estimates for each of the 400 simulations are shown in figures 3.1–3.6. Some of these figures are not in Paper A but are included here to show more detail between the different quantitative semantics.

As seen in Figure 3.1, in the upper right plateau (where $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$, approximately), the objective function values are the same with the *Max* semantics. This situation happens because when the lead car continuously accelerates, the ego car also increases its speed. But the ego

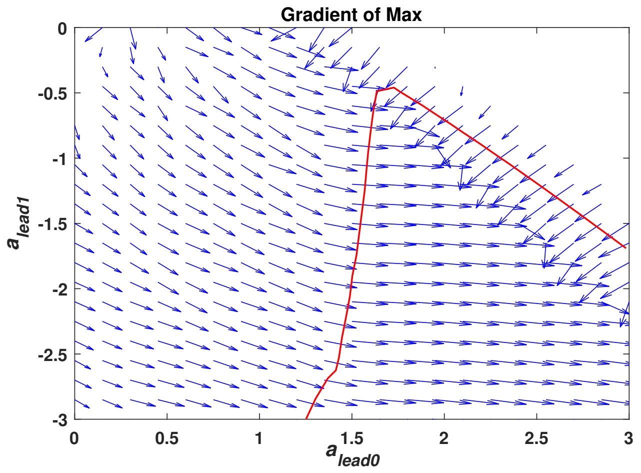

Figure 3.2: The gradient direction using *Max*. The triangular curve shows the edge of negative objective function values, i.e., the falsification area.

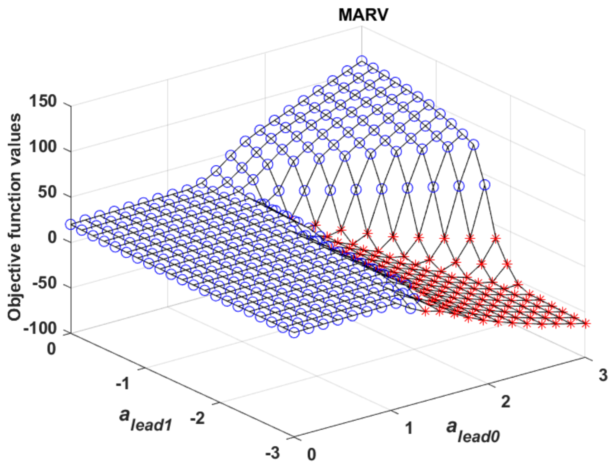

Figure 3.3: Objective function values for combinations of the accelerations (a_{lead0}, a_{lead1}) using *MARV* semantics, presented in Paper A. Positive values (○) mean that the specification is satisfied, while negative values (∗) mean that it is falsified.

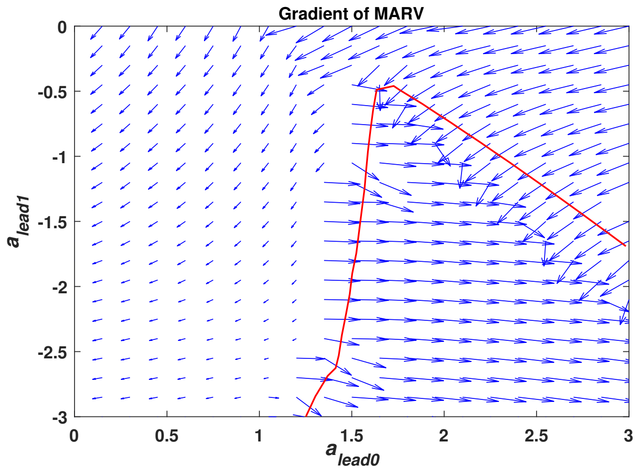

Figure 3.4: The gradient direction using *MARV*. The triangular curve shows the edge of negative objective function values, i.e., the falsification area.

car has a driver-set velocity limit of 30 m/s thus, the relative distance between the vehicles increases during the simulation. The minimum objective function value, therefore, occurs at the beginning of the simulation $t = 0$, where the relative and safe distances are closest to each other. For this reason, as can be seen in Figure 3.2, for points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$, there is no useful gradient, or sense of direction, in order to get closer to a falsification point. On the other hand, for other points the objective function values change. Therefore, those points can be used to guide the optimization algorithm towards a falsification point inside the triangular curve of Figure 3.2. Thus the *Max* semantics considers simulations to be equally good/bad for parameters that are close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$, resulting in no useful information for the optimization.

On the other hand, in Figure 3.3, we can see that the points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$ do have different values under *MARV* semantics. We can also see in Figure 3.4, the gradient for the points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$ is decreasing. But here, the situation is the opposite as for *Max*. For points close to $0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$ the gradients do not point towards the falsifiable area.

The objective function values using *Additive* and its gradients are shown

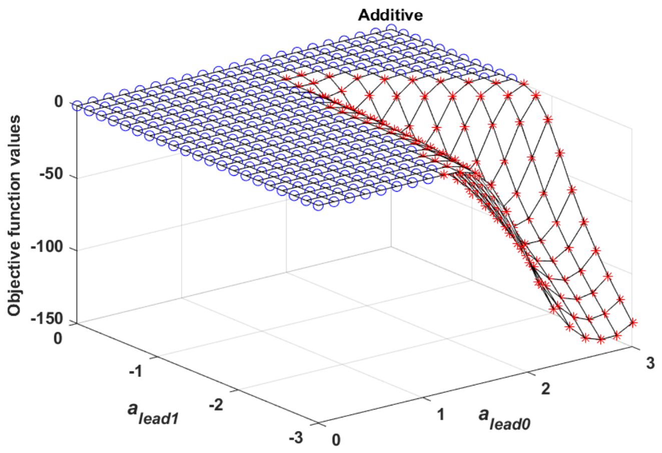

Figure 3.5: Objective function values for combinations of the accelerations (a_{lead0}, a_{lead1}) using *Additive* semantics. Positive values (∘) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified.

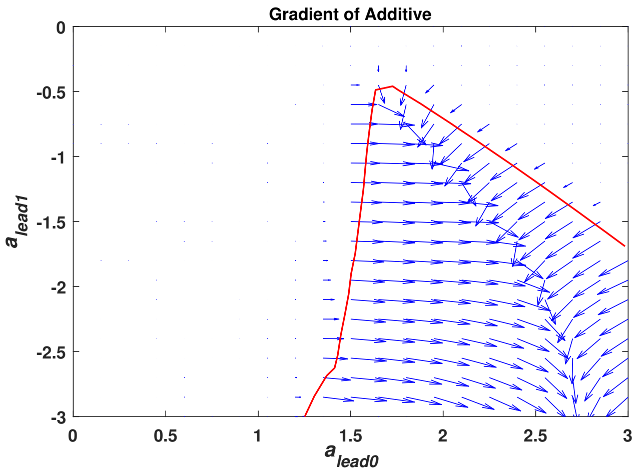

Figure 3.6: Gradient directions using *Additive*. The triangular curve shows the edge of negative objective function values, i.e., the falsification area.

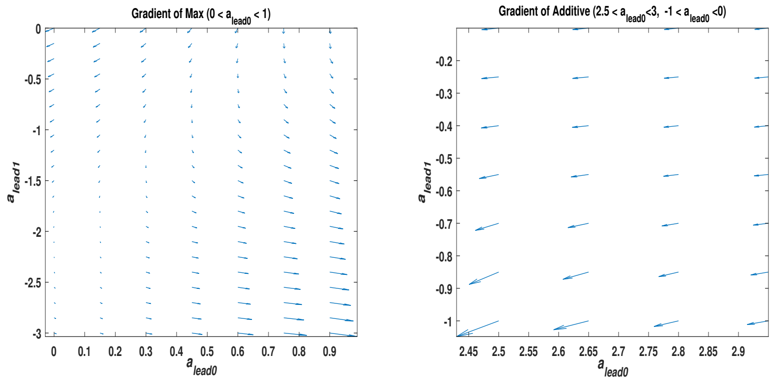

Figure 3.7: Gradient directions using *Additive*, detail of Figure 3.6. To the left, gradients for $0 < a_{lead0} < 1$, and to the right, $2.5 < a_{lead0} < 3$.

in figures 3.5 and 3.6. It might seem as there are no gradients in the areas ($0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$) and ($2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$), but this is not the case. Because of the very small difference between the objective function values in these areas, the arrows representing the gradients do not appear in Figure 3.6. For better illustration, magnitude gradients are shown in Figure 3.7. As can be seen here, for the points in the range ($0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$), there is a different direction. On the other hand, there is a clear direction for the points in ranges $0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$.

According to the results and discussions, it can be concluded that when only one quantitative semantics is used, no semantics always guides the optimization towards a point where the specification is falsified. Moreover, for some areas, if a semantics results in a constant objective function value for an area, it cannot guide the optimization algorithm in any direction. Thus, using a combination of different quantitative semantics might provide valuable information, and it might be possible to avoid using the semantics that does not have a clear direction to be used by the optimizer.

A problem with the *Max* semantics for the $\wedge$ operator is that it is only sensitive to whichever of x or y has the lowest value. *Additive* is sensitive to both values of x and y. For the *always*-operator of *MARV*, for a positive, all signal values affect the value of *MARV*. In this aspect, it is similar to

Additive. Thus, *Additive* and *MARV* might give useful information that might be exploited by the optimization algorithm during the falsification process. This example illustrates the effect of different quantitative semantics.

CHAPTER 4

Searching for Test Inputs

Falsification of CPSs can be done using either optimization-free or optimization-based methods to search for inputs that falsify specifications. For optimization-based falsification, a quantitative semantics is important because it is used to guide the optimization process in the right direction. However, optimization-free methods guide the search in other ways, for example, by using a random or structured exploration of the parameter values. Below, both optimization-free and optimization-based methods for falsification will be introduced.

4.1 Optimization-Free Methods

An optimization-based approach is guided by an objective function that is to be minimized or maximized, an optimization-free method is not guided by an objective function; instead, the search is guided by other criteria. Possible optimization-free methods include random search or a search procedure that generates a deterministic sequence for evaluation. In this work, we consider two optimization-free methods, corners and random search methods, and also their combination, described below.

Corners

The input parameters are chosen within their defined ranges $[l, u]$ where l and u refer to lower and upper bounds as an $n-$dimensional vector, respectively. Corner points refer to input parameters where the parameter values are at either the lower or upper bound. As discussed in Paper C and [56], corner points are effective as a falsification procedure.

For a system with n input parameters, there are 2^n corner points. Figure 4.1 illustrates corners in two dimensions. In this case, there are four corner points:

$$\begin{pmatrix} l_1 \\ l_2 \end{pmatrix}, \begin{pmatrix} l_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} u_1 \\ l_2 \end{pmatrix}, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

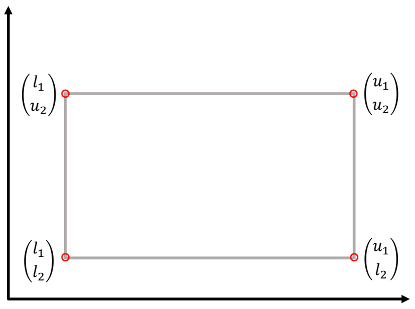
$\begin{pmatrix} l_1 \\ u_2 \end{pmatrix}$
$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$
$\begin{pmatrix} l_1 \\ l_2 \end{pmatrix}$
$\begin{pmatrix} u_1 \\ l_2 \end{pmatrix}$

Figure 4.1: An example in two dimensions to show the corner points.

Random Search

Random testing (RT) is an optimization-free approach where inputs are generated using a random generator. Despite its simplicity and ease of implementation, RT has proven to be an efficient method for test generation [76], [77] and is used extensively in practice. Random testing together with input generators is used in the tool QuickCheck [78], which has been applied to the testing of large software systems. QuickCheck was also extended to handle CPSs in [58]. A potential shortcoming of random test case generation is that counterexamples might be complicated; thus, a shrinking procedure was also introduced in [58] to simplify a counterexample after it has been found.

In Paper C, a uniform random sampling method is used. In this method, each sample test is generated from a uniform random distribution, which means that each value in the input ranges has an equal probability of being sampled. It is shown that RT is a viable strategy for many problems in Paper C. Random testing may be inefficient for more challenging problems since no information is used to guide the falsification.

Hybrid-Corner-Random

Hybrid-Corners-Random (HCR), presented in Paper C, combines the corners and random methods to take advantage of both. This method may be used as a baseline method for the optimization-based approaches to beat. From an evaluation of the benchmark problems, it turns out that this strategy, although simple, performs well and is, for some problems, more efficient than using an optimization-based approach.

4.2 Optimization-Based Methods

In optimization-based methods, an objective function is either maximized or minimized by choosing values for the decision variables within an allowed set. Gradient-based approaches, like gradient descent, can be used when an explicit model of the objective function is available. But for many industrially relevant falsification problems, only a black-box model of the system is available. In this thesis, we assume that the SUT is available as a black-box model, which thus allows only gradient-free optimization methods [79], [80].

Optimization-based falsification aims at reducing the number of tests by using an optimization method to determine the next set of inputs from an evaluation of the previous simulations. The optimization problem can be defined as

$$\min_{x \in \mathcal{X}} f(x), \tag{4.1}$$

where $f(x)$ is the objective value given by the used quantitative semantics evaluated for a given specification and SUT. Note that evaluating f for given parameters x requires a simulation of the system with the inputs generated using the parameters x, followed by an evaluation of input and output signals using quantitative semantics.

Let $x_{\min}$ be the global minimum of the objective function in the search space domain $\mathcal{X} \subseteq \mathbb{R}^n$ with the input parameters in the interval:

$$[l, u] := \{x \in R^n \,|\, l_i \leq x_i \leq u_i,\, i = 1, \ldots, n\},$$

where $l, u \in R^n$. The $[l, u]$ are bounded with a nonempty interior, and n is the number of dimensions in the optimization problem.

The optimization aims to get a falsified point with a negative objective function value, i.e., $f(x) < 0$, if possible. A challenge in falsification is that before the falsification starts, we have no information about the value or sign of $f(x_{\min})$ and if $f(x_{\min} \geq 0)$, it is not possible to falsify the system. All valid input points are in a box with a lower and upper value. The SUT is simulated at given input points, and the corresponding objective function values are calculated using the used quantitative semantics.

There are, in general, two types of black-box optimization methods: direct search and model-based. Direct-search methods do not require any information about gradients. On the other hand, model-based optimization methods build a surrogate model that can be used for guiding the search, for example, by providing estimations of gradients. These two methods will be discussed in the rest of this section.

When a new set of input signal values (a point) is generated for evaluation in the falsification process, the SUT is simulated with this point. Then, the objective function is calculated to determine whether or not the specification is falsified. If the specification is falsified, the optimization terminates with the falsified point. Otherwise, the process will continue to find a new point, likely having a lower objective function value, i.e., being closer to being falsified. The optimization process works until the specification is falsified or the maximum number of simulations has been reached.

The Direct-Search Methods

A direct-search method, [81], refers to the sequential consideration of trial solutions generated by a particular strategy. By comparing only objective function values for a given set of input parameters (points), without using gradient approximations, direct-search methods determine new candidate points for future exploration.

Direct-search methods can be used if the objective function is not continuous or differentiable, are relatively easy to implement, and can also be applied to nonlinear optimization problems. They require minimal effort to be used and often have few hyperparameters that need to be set [82]. For falsification problems, gradients are, in general, not available. Thus direct-search methods have been applied. Nelder-Mead (NM), a direct-search optimization method [57], has been the default optimizer in Breach. Also, Line-Search-Falsification (LSF) presented in Paper C will briefly be introduced in this section.

Nelder-Mead

The NM [57] method is a gradient-free numerical method to find the minimum of an objective function in a multidimensional space. NM needs only function evaluations making it suitable for minimizing non-differentiable functions. This method uses objective function values at $n+1$ vertices in n dimensions, where n is the number of decision variables, to compute the next point.

NM is a simplex-based direct search method that begins with a set of $n+1$ points (vertices). Based on the objective values at these points, new points for evaluation will be generated. In each iteration, one or more test points are evaluated. The worst point, i.e., the point with the highest objective function value, will be replaced with a new point.

In this method, first, the $n+1$ points are ordered from the lowest objective value to the highest objective value, *ordering*, and labeled. Then, in the case of *reflection*, *expansion*, or *contraction* exactly one new point is generated. In the case of *shrinking* n new points are generated. Figure 4.2 shows how new points are generated for the different cases by an illustration in two dimensions, from [83].

NM is modified in Paper B to work with multiple semantics. It is also evaluated as one of the optimization methods in Paper C.

Line-Search Falsification (LSF)

The Line-Search Falsification (LSF) method presented in Paper C is a direct-search optimization method tailor-made for falsification problems. Corners and random points are combined with a local search. An important property of this method is that it includes a crawling procedure similar to NM for local optimization but also has the ability to get out of local optima and continue the optimization from a new but related point. LSF needs three initial input

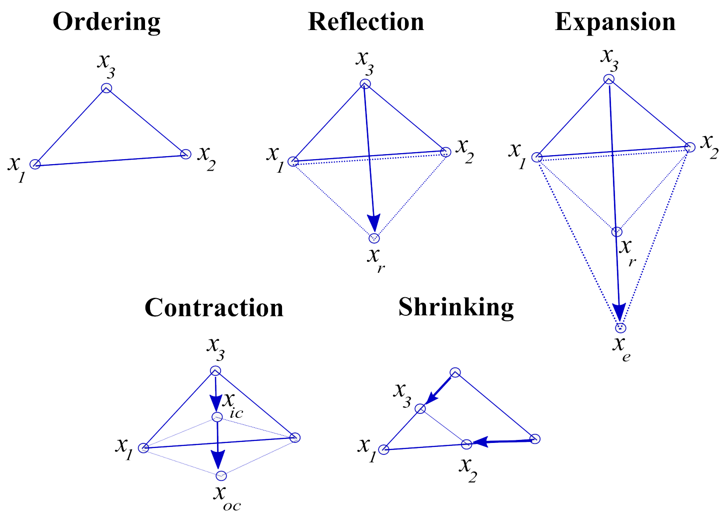

Figure 4.2: A two-dimensional illustration of the Nelder-Mead method of generating new points.

points to start the optimization process. The optimization is done over a line that passes through one of these points and cuts off at the edges of the upper/lower input ranges or a corner point. Paper C discusses the details of LSF.

Model-Based Optimization Methods

Model-based optimization methods try to predict the performance of the objective function by building a surrogate model of it and then use this surrogate model during the optimization process [80]. Overall, building a surrogate model reduces the number of required function evaluations. For CPSs, an evaluation typically requires a simulation of the SUT, which is often time-consuming. Potentially, model-based methods can be used to select each sample before evaluation more carefully. In this work, SNOBFIT and Bayesian Optimization (BO) are used, and both are briefly described below.

Stable Noisy Optimization by Branch and Fit (SNOBFIT)

SNOBFIT [66] solves a global optimization problem assuming only a black-box model and continuous decision variables. SNOBFIT is developed to support scenarios where it is expensive to evaluate parameters. SNOBFIT is also developed to handle noisy evaluations, i.e., if evaluating the same point multiple times it is possible to get slightly different evaluation results. SNOBFIT combines global and local search and searches simultaneously in several promising sub-regions. SNOBFIT operates by building local models of the function around the evaluated points. SNOBFIT generates points suitable for evaluation and puts them in one of five classes. The points chosen for evaluation are picked from these classes, and the user can influence in which proportions the algorithm picks points from these classes and may in this way influence if global exploration or local exploitation should be emphasized.

Bayesian Optimization

Bayesian optimization (BO) [84] is a global optimization method that has shown to be efficient for solving optimization problems where it is costly to evaluate functions and gradients are not available. BO also allows the function to optimize to be non-convex and multimodal (a function with more than one optima), and have multiple local optimal. BO can also handle the optimization of functions perturbed by noise.

The main idea behind BO is to build a probabilistic model of the objective function. This model includes an estimate of the function, including uncertainty. A BO algorithm contains two main components. The first is a probabilistic *surrogate model*, which represents the mapping between the input parametrization $x \subseteq \mathbb{R}^n$ and the objective function value $y = f(x) \subseteq \mathbb{R}$. This model consists of a prior distribution of the behavior of the unknown objective function based on observations. The second component is an acquisition function built on top of the surrogate model, to determine where to evaluate the function next. When a new point is searched, the surrogate model is updated, and the process continues until the best point of the function is found. The overall BO method is presented in Algorithm 1. At the start, M points x_i (for $i = 1, \dots, M$) in the search space, $\mathcal{X} \subseteq \mathbb{R}^n$ are generated randomly. After calculating the objective function values $y_i = f(x_i)$ for these points, an initial sample set $\mathcal{D}_0 = \{(x_i, y_i)\}_{i=1}^{M}$ is created. Using a predefined acquisition function, the algorithm selects the next point x^* for evaluation. Then, the sample set $\mathcal{D}$ is updated, and the previous steps are repeated until the total number of evaluations, N, is exhausted.

Algorithm 1 Bayesian Optimization

```
1: Input: Input space $\mathcal{X} \subseteq \mathbb{R}^n$, the initial design size $M$, the max number of evaluations, $N$.
2: Initialize: $\{x_i\}_{i=1}^{M} \sim \mathbb{U}(x), \{y_i \longleftarrow f(x_i)\}_{i=1}^{M}$, {Sample random points from the uniform distribution and evaluate the objective function.}
3: $\mathcal{D}_0 \longleftarrow \{(x_i, y_i)\}_{i=1}^{M}$ {Collect the initial design set.}
4: for $j$=1, 2, ..., $N$ do
5:     $x^* \longleftarrow \arg\min_{x \in \mathcal{X}} \alpha(\mathrm{x}, \mathcal{D}_{j-1})$ {Train a probabilistic model as $p(y|\mathcal{D}) = \mathcal{GP}(y; \mu_{y|\mathcal{D}}, K_{y|\mathcal{D}})$ with $\mathcal{D}_{j-1}$ and find the next sample $x^*$.}
6:     $y^* \longleftarrow f(x^*)$ {Evaluate the objective function.}
7:     $\mathcal{D}_j = \mathcal{D}_{j-1} \cup \{(x^*, y^*)\}$ {Update the sample set.}
8: end for
```

A variety of approaches are proposed in the literature to model a function, and its uncertainties, including random forests [85] and Gaussian processes ($\mathcal{GP}$) [86]. $\mathcal{GP}$ is a collection of random variables that are jointly normally distributed. In this thesis, $\mathcal{GP}$ is used to model the function being optimized and is described in more detail below.

A surrogate model of a function: Gaussian processes $\mathcal{GP}$ is the most popular model for building a posterior surrogate model. Typically, $\mathcal{GP}$ consists of a finite number of randomly distributed variables,

$$p(y) = \mathcal{GP}(y; \mu, K), \tag{4.2}$$

where μ is the mean approximation of $f(x)$ and K is the covariance or positive–definite kernel. Given M number of observations $\mathcal{D}_0 \longleftarrow \{(x_i, y_i)\}_{i=1}^{M}$, the distribution p is

$$p(y|\mathcal{D}) = \mathcal{GP}(y; \mu_{y|\mathcal{D}}, K_{y|\mathcal{D}}). \tag{4.3}$$

An example[1] of how a surrogate model can be built is shown in Figure 4.3. There are three initial samples, $M = 3$, red circles in the top left graph, with their objective function values, $f = \left[f_1, f_2, f_3\right]^T$. Now, we want to make a prediction about the function value at a new point x_*. Different possible

[1]Inspired from `https://www.cs.ubc.ca/~nando/540-2013/lectures/l6.pdf`

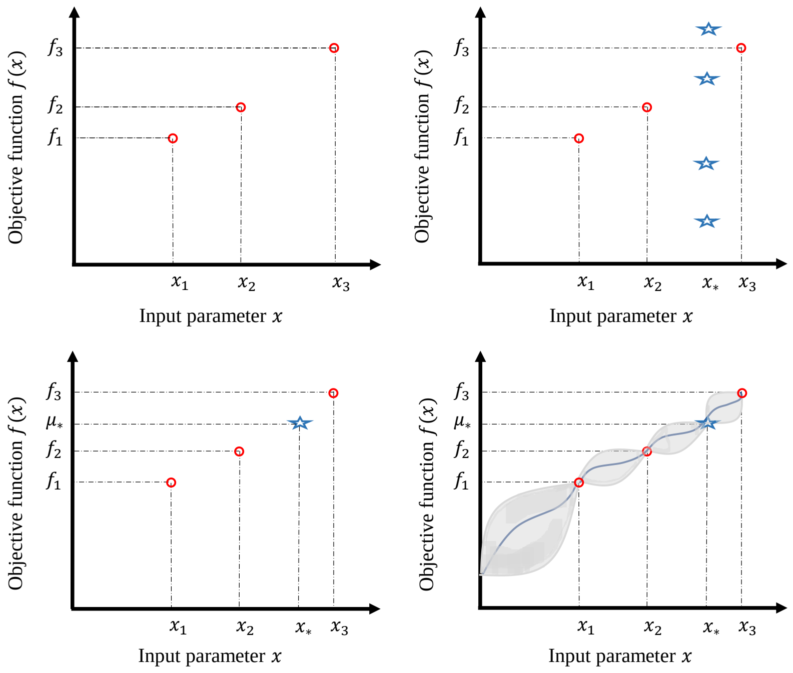

Figure 4.3: Gaussian process model with three initial samplings (points) x_1, x_2, x_3 shown by $\circ$. A new point is searched, shown by $\star$.

objective values can be predicted for x_* as shown in the upper right graph by the stars. The reasonable estimate would be between the objective values of the two points, x_2 and x_3, where f_* is less than f_3 and larger than f_2, as in the bottom left graph.

To calculate the mean and variance at any point x_*, we have:

$$\mu[x_*] = K\left[x_*, X\right] K\left[X, X\right]^{-1} f, \tag{4.4}$$

$$\sigma^2[x_*] = K\left[x_*, x_*\right] - K\left[x_*, X\right] K\left[X, X\right]^{-1} K\left[X, x_*\right], \tag{4.5}$$

where

$$K[X,X] = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}, \; K[X, x_*] = \begin{bmatrix} k_{1*} \\ k_{2*} \\ k_{3*} \end{bmatrix},$$

$$K[x_*, X] = \begin{bmatrix} k_{*1} & k_{*2} & k_{*3} \end{bmatrix}, \; K[x_*, x_*] = k_{**},$$

where k_{ij} refers to the covariance of the two points x_i and x_j. Different possible kernels will be discussed in the rest of this section.

Similar to what was discussed for a point x_* in (4.4) and (4.5), we can calculate the mean and variance for any point. In the bottom right graph of Figure 4.3, a surrogate model that predicts the function from three observations is built. The model describes a probability distribution over functions that pass through the three points. The mean function can be visualized as a curve with a gray area around each point representing the uncertainty. By sequentially adding points, the mean of the function changes, and the uncertainty decreases as each additional point provides more information.

Kernels An important part of $\mathcal{GP}$ is the kernel, which determines the shape of the prior and posterior. We can determine if two similar data points have similar target values using kernels. Kernels are divided into two categories: stationary and non-stationary. In a stationary kernel, the distance between data points is considered, not their absolute value. In contrast, a non-stationary kernel is determined by the value of the specific data points. Some commonly used kernels will be introduced below.

1. **Squared Exponential Kernel**: This is a stationary kernel

   $$k_{sq}[x, x'] = \sigma^2 \exp\big(-\frac{1}{2}d^2\big),$$

   where σ^2 is a scaling factor that determines the variation of function values from their mean, and d is the Euclidean distance between the points, i.e., $d^2 = (x - x')^T(x - x')$.

2. **Periodic Kernel**: The stationary periodic kernel is

   $$k_{Periodic}[x, x'] = \sigma^2 \exp\Big(-2\big(\sin\big(\frac{\pi d}{\tau}\big)\big)^2\Big),$$

   where τ is the period of the oscillation.

3. **Matérn kernel**: The stationary Matérn kernels are

$$\begin{aligned} k_{Matérn1}[x, x'] &= \sigma^2 \exp\big(-d\big), \\ k_{Matérn3}[x, x'] &= \sigma^2 \exp\big(-\sqrt{3}d\big)\big(1+\sqrt{3}d\big), \\ k_{Matérn5}[x, x'] &= \sigma^2 \exp\big(-\sqrt{5}d\big)\Big(1+\sqrt{5}d+\frac{5}{3}d^2\Big). \end{aligned}$$

4. **Linear Kernel**: The linear kernel is a non-stationary kernel

$$k_{Linear}[x, x'] = x^T x'.$$

Gaussian processes with the squared exponential kernel as covariance function have mean square derivatives of all orders and are thus very smooth. Matérn is a generalization of the squared exponential kernel that is a very flexible class of stationary kernels. On the other hand, kernels such as the squared exponential do not have the periodic property and could produce inaccurate measures of similarity [87].

Acquisition Functions A BO method also requires an acquisition function to decide on the next sample. The next point is selected based on the acquisition function being minimized or maximized. It is often easier to optimize acquisition functions than the original objective function because the analytical form makes them easy to evaluate. The performance of BO depends on the balance between exploration and exploitation. The result of too much exploitation is greedy optimization, which means that a surrogate model can easily be trapped in a local minimum. Conversely, too much exploration would result in an inefficient performance, where a surrogate model is kept improving with every new iteration without any exploitation. Thus, selecting the best next point for evaluation is a trade-off between exploration and exploitation.

Depending on the application, one acquisition function may be preferred over another. This thesis focuses on four acquisition functions: Thompson Sampling (TS) [88], Lower Bound Confidence (LCB) [89], Probability of Improvement (PI) [90], and Expected Improvement (EI) [91]. All presented acquisition functions are shown in one example with five initial samplings in Figure 4.4 [2].

[2]These figures are inspired from `https://www.borealisai.com/research-blogs/`

Each prediction made by a surrogate model comes with the confidence interval explained with a corresponding standard deviation. LCB refers to the lower bound of the uncertainties of the surrogate model. On the other hand, TS is a randomized acquisition that samples a function from the posterior with the lowest value. Then, it optimizes over that function to generate the next point. TS performs well in practice, is fast, and mostly focuses on exploitation. PI measures the probability that the best next point leads to an improvement upon a target. However, PI does not consider how much the improvement will be. Moreover, improving directly upon the current best solution can lead to trapping the search in a local optimum. A solution would be to weigh each improvement to avoid small improvements over larger ones. In Expected Improvement (EI), the objective is to find the best next point that maximizes the expected value of the improvement.

Except for EI, the acquisition functions are presented in detail in Paper D. Here we give the formal definition of EI:

$$\alpha_{\mathrm{EI}}(x^*) \in \operatorname*{argmax}_{x \in \mathcal{X}} \big(\tau - \mu(x)\big) \cdot \Phi\left(\frac{\tau - \mu(x)}{\sigma(x)}\right) + \sigma(x) \cdot \phi\left(\frac{\tau - \mu(x)}{\sigma(x)}\right), \quad (4.6)$$

where Φ is the standard normal cumulative distribution function and ϕ is the standard normal probability density function. For both PI and EI, the target value, τ, is the best-observed value with the lowest objective value. This target value for PI can lead to an overly greedy optimization [92]. On the other hand, in practice for EI, setting the target to the minimum best point works reasonably well [93]. If the objective function is very noisy, using the lowest mean value as the target is reasonable [94].

For high-dimensional applications, typically with $n > 15$, traditional BO does not scale well [95], for several reasons: (i) Typically high-dimensional applications require more evaluations to converge; using BO in this setting eventually becomes impractical [96], [97]. (ii) A good input space coverage is required to achieve a global optimum. The number of evaluations is increased to cover the input space due to the exponential increase in search space with increasing dimensions. (iii) When the search space grows faster than the number of evaluations allowed to be done, an overemphasis on exploration results in poor exploitation. The emphasis is on the uncertainty of

`tutorial-8-bayesian-optimization/`.

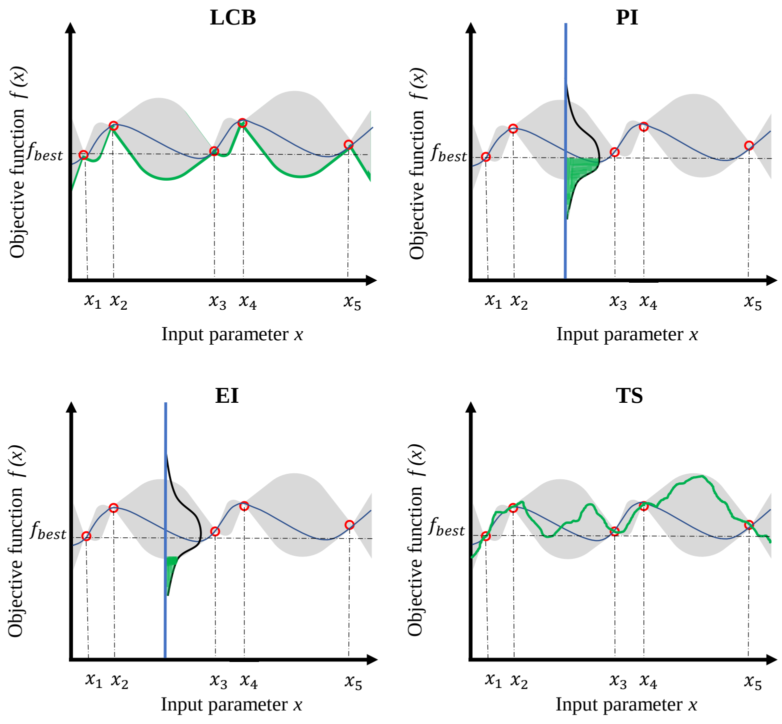

Figure 4.4: Four different acquisition functions: LCB, PI, EI, and TS with five initial samplings $\{x_1, \ldots, x_5\}$. The grey region demonstrates the variance, i.e., the uncertainty around the mean. The green parts show how each acquisition function is generated. The blue curve represents the mean function. f_{best} refers to the best (lowest) objective values found so far.

the surrogate model rather than high model prediction. (iv) Fitting a global surrogate model in high dimensions poses a challenge. (v) For the posterior to be computed, the inverse of the kernel matrix has to be computed, whose size ultimately depends on the number of function evaluations made so far. Thus, with an increase in dimensions, computational and storage costs for posterior distributions also increase exponentially as the number of function evaluations increases. To overcome the BO scalability problem, some works have been done in the literature [98]–[100].

Typically, in high-dimensional problems the objective function is only affected by a few dimensions [98]. This property was a motivation for some works [96], [99], [101]. In [101], the subspace identification first estimates the subspace on which the function is supposed to project inputs into low-dimension, and then BO optimizes the function on the learned subspace. The sliced inverse regression method [102] was suggested, which could automatically learn the intrinsic structure of the objective function during the optimization. [101] uses the sliced inverse regression technique for dimension reduction. Hash-enhanced Subspace BO [96] is another method that recovers the vector in the original space from the low dimension vectors using a hash function [103]. In [99], Random Embedding Bayesian Optimization (REMBO) was suggested based on different dimensionality reduction techniques that generate a low-dimensional embedding of the original search space, which is practical for exploration and exploitation. In REMBO, the high-dimensional space is embedded in a lower-dimensional manifold via a random linear embedding. However, selecting the optimal size of a linear embedding is an open question and the main limitation of REMBO and other similar high-dimensional BO methods [96], [104].

Another way to scale BO to high-dimensional problems is to model a richer class of functions instead of low dimensionality assumption, called *additive structure* [105]. In the additive structure, the objective function is decomposed as a sum of independent low-dimension functions, each dependent on a disjoint subset of dimensions. In [106], the additive model on the projected data is considered to address the restriction of requiring decomposition to be axis-aligned in [105]. In the models based on the additive structure method, the problem of knowledge about the decomposition groups and learning them is challenging. To deal with these problems, randomly sampling the decomposition is suggested in [105], [106].

Another approach to scaling BO methods to high-dimensional problems is to adopt a local strategy [100]. A local BO method, called TuRBO is suggested in [100]. A subset of the regions, called *trust regions* (TR) [107], is built around the current best solution to approximate an objective function. This method is evaluated for falsification in Paper D. In [108], a trust region framework was proposed to optimize with global convergence, escaping local optima while preserving the asymptotic properties of BO.

A test engineer may also have a prior belief about the potential location of an optimum [109], [110]. The standard BO approach fails to incorporate this valuable source of information. To address this issue [109], [110] proposed methods to inject this prior to BO. In Paper D, πBO [110] is evaluated.

In the following, TuRBO and πBO will be introduced.

Trust Region Bayesian Optimization (TuRBO) BO methods rely on building a surrogate model that might not be accurate enough to fit a global model. For example, if we want to build a surrogate model using all three initial points in Figure 4.3, it is harder to predict a good value for the new point x_*. On the other hand, it is easier to build a local model with only x_2 and x_3 in a small region. The idea behind TuRBO is to build local surrogate models. For this purpose, a surrogate model is built inside a TR [107]. A TR can be a sphere, polytope, or hyperrectangle with the center located at the current best optimal; this means the point with the lowest objective function found so far during optimization. Large enough TR would be equivalent to standard global BO methods. So, TR should be large enough to contain good solutions while small enough to build an accurate local model. Hence, there are limitations for the size of TR (L_{min}, L_{max}). In TuRBO, $\mathcal{GP}$ models within a hyperrectangle TR are used. The TR is expanded when a new point with a better objective function is found; otherwise, TuRBO shrinks.

At the beginning of the TuRBO process, a base side length is initialized for TR, L_{init}. An acquisition function is used at each iteration, i, to select a batch of q candidates $x_1^{(i)}, \ldots, x_q^{(i)}$ within TR. If within TR, the better points were searched consecutively, the size of TR is doubled, i.e, $\min(L_{min}, 2L)$. If TuRBO fails to find better points, TR is halved in size, $L/2$. If the size of TR is less than L_{min}, or larger than L_{max}, the current TR is discarded, and a new TR with L_{init} is initialized.

π**BO** This is a simple method that allows the injection of prior knowledge about promising areas into BO [110]. With πBO, it is possible to predict, by using a prior distribution, where a point is located with a lower objective function value within the input space. πBO is a method that aims to modify an acquisition function by multiplying it with a predefined probability distribution, see Paper D for more details.

Comparing Different Test Generation Methods

Each optimization method has specific features, which are compared from different perspectives. Their performance from experimental results evaluated on benchmark problems in [56] and Paper B is discussed in the attached papers and in Section 6. Comparing the optimization-free methods with optimization-based methods, the former is simpler. On the other hand, the optimization-free methods may or may not be efficient since no information is provided to guide the test generation. As it is shown in Paper C, these methods show a good performance for those problems that are falsifiable using corner points, random points, or their combination. Although, some problems need to use optimization-based methods to guide the falsification process to falsify a specification.

Comparing the optimization-based methods, direct search methods do objective function evaluation using the real objective function directly to search a new point. On the other hand, in model-based optimization methods, a surrogate model is built first, and this model is then used to search for a new point for the next iteration. Hence, model-based methods would be more efficient when the objective function is expensive to evaluate, or there is a low simulation budget.

Randomness has proven useful for testing, as discussed in Section 4.1. Hence, randomness properties have been considered in optimization-based methods. For example, LSF is a method that combines random exploration with local search where the lines in the n-dimensional parameter space are generated in random directions. For BO methods, random points are generated to build a surrogate model.

Different methods need to start the optimization process with different initial sampling points. In SNOBFIT and BO methods, these initial sampling points can be any number, depending on the system's dimensions or a fixed number. On the other hand, NM needs $n+1$ samples, while the LSF method needs three initial samples. The initial sampling for NM, SNOBFIT, and BO methods are generated randomly within the allowed parameter ranges. On the other hand, LSF is designed such that both corner points or the points on the board of ranges can be considered for the initial samplings.

When SNOBFIT does not get any hints from the objective function in which direction to continue the search, it tends to explore new parameter values towards the corner points. On the other hand, TuRBO explores the points more within the ranges, especially when the local optima is in a very narrow area, i.e., small input space, it might perform well, as was shown in Paper D. However, πBo can search both corners, edges of input ranges, and even within input ranges. Comparing the BO methods, TuRBO builds a surrogate model locally, while a standard BO does it globally.

As is shown in Paper C, Paper D, and Paper E, depending on the evaluated system and specification, one approach performs better than the other. Since we have worked with the black-box models, it is not possible to conclude which optimization works better for which SUT and specification. However, as is shown in Paper C and Paper D, the LSF method shows a better performance in general for the evaluated benchmark problems. On the other hand, TuRBO outperforms LSF in Paper D for some problems that were hard to falsify in Paper C. The HCR method performs the best for some easy problems that be falsified at corners or using random points.

CHAPTER 5

Parameterization of Input Signals

In optimization-based falsification, input signals are parameterized such that parameters are decision variables in the optimization problem. The optimization method attempts to find parameters such that the specifications are violated. In this chapter, we discuss how to parameterize input signals such that they are suitable for falsification.

5.1 Input parameters

CPS falsification problems, in general, include continuous-time input signals; hence, the search space is of infinite dimension. The optimization problem is simplified if we consider input signals that are parameterized by a finite number of parameters, which are also the decision variables in the optimization problem. The actual input signals are generated using an *input generator*.

A variety of parameterized input generators can be used. For example, the parameters can represent control points, and the actual input signals are created by interpolating between the control points. Since system dynamics are complex and often unknown, especially in large-scale industrial systems, defining suitable input signals and parameters is a challenging problem that requires expert knowledge and good insight into the SUT.

The dimensionality of the optimization problem affects optimization-based approaches. A large number of decision variables are problematic for the optimizer since the search space increases rapidly with the number of decision variables. On the other hand, if few control points or parameters are chosen, this decreases the problem's dimensionality. Although constraining inputs so that counterexamples may not be possible to construct with the current parameters, it might be possible for a more general input signal to be formulated. Hence, it is crucial to find a balance between the input generation's flexibility and the optimization problem's dimensionality.

The problem input signals for CPSs are considered in some work [32], [64], [111]. In [64], new control points are incrementally added to constant signals as needed. A CPS state space is explored using rapidly exploring random tree techniques in [111]. More recent methods come from the reinforcement learning domain and use tree-like structures [32] to partially evaluate inputs. However, because they require adding numerous control points, such methods might fail to converge on counterexamples located in high-frequency regions, as they would need to target a relatively small region of the input signal space. For a high-dimensional input space, when there are multiple inputs to the system, by decoupling independent input signals in [112], the effectiveness of the optimization method is increased. In [113], the authors present a timed automaton input generator used to produce complex periodic behaviors.

The rest of this chapter will discuss different input parameterizations and interpolating between them to construct an input signal.

5.2 Signal Generators

Some possible input parameterizations are discussed in this section. A signal can be represented by parameters in the time and signal domains. In the time domain, the signal is expressed as a function of time, limited by the simulation time T. On the other hand, the signal domain refers to the amplitude that the signal is allowed to be in, $[l, u]$.

Constant Input

A constant input is a simple signal that only requires one control point that defines the signal's value, which remains constant throughout the simulation.

This value can be any value inside or on the allowed range $[l, u]$, for the entire simulation time, 0 to T. Figure 5.1 shows an example of the constant input signal.

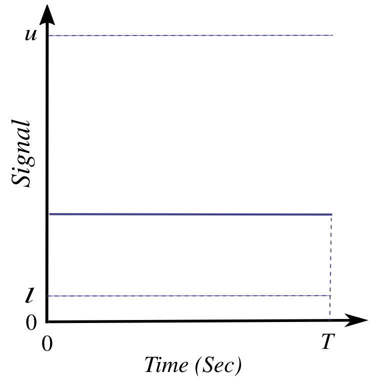

Figure 5.1: A constant input signal with one control point.

Piecewise Input

A piecewise input signal has multiple sub-segments, where each can have a different value and a different interval.

Two parameters need to be defined to create this input signal: the number of control points and the interpolation between them. The time domain is parameterized based on the number of control points. Figure 5.2 shows three input signals where three control points in the signal domain are distributed uniformly in the time domain; $t_1 = \frac{T}{3}$ and $t_2 = \frac{2T}{3}$. The three control points can be interpolated in different ways, such as *previous*, *linear*, and *pchip*. The *previous* interpolation corresponds to the previous sample value.The *linear* interpolation involves linearly interpolating between the control points. Finally, *pchip* is cubic polynomial interpolation between the control points, each with specified derivatives. Many other interpolations are possible, although not investigated in this work, for example *cubic*, *makima*, and *spline*.

In Figure 5.2, the input signals are distributed uniformly in the time domain, i.e., the same interval. It is also possible to vary the time domain to have different time intervals as shown in Paper E, where there are three control points with non-uniform intervals $(0, t_1)$, (t_1, t_2) and (t_2, T). In these

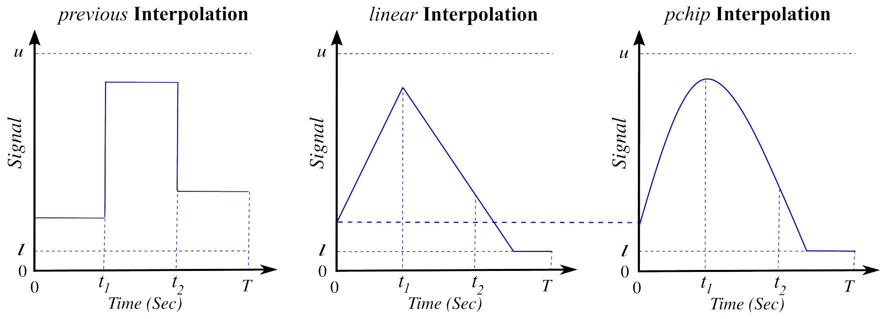

Figure 5.2: Three piecewise input signals with three control points distributed uniformly in time with *previous*, *linear*, and *pchip* interpolation.

signals, five control points are needed to build a signal, two extra compared to Figure 5.2. These extra control points define the length of the intervals in the time domain.

If the values of each three sub-segments between $(0, t_1)$, (t_1, t_2) and (t_2, T) are all the same, a constant input signal is generated as in Figure 5.1, no matter what values t_1 and t_2 have. On the other hand, if $t_1 = t_2 = 0$, a step signal is generated that switches from the lower to the upper value.

Pulse Input

A pulse input is a periodic square wave pulse can be defined by five parameters *period*, *base*, *amplitude*, *delay*, *width*, as shown in Figure 5.3. Different types of signals can be obtained using the pulse, depending on the ranges for the different parameters. For example, if $period \geq 2T$ and $width \geq 0.5$, this describes a constant signal. When $period = 2T$ and *delay* having any value in $[0, T]$, it describes a single step input.

While Figure 5.3 shows the suggested pulse generator in [114], a modified version of the pulse that has *low* and *high* parameters instead of *base* and *amplitude*, is presented in Paper E. When *base* and *amplitude* are used, as in [114], it is hard to define suitable input ranges. When only *amplitude* is considered as a parameter, it can be assumed to vary between a lower and higher bound while the *base* is set to the lower bound. On the other hand, if *base* also needs to be considered as an input parameter besides *amplitude*, including values for both the *amplitude* and *base* increases the complexity.

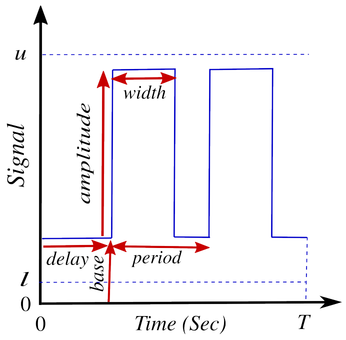

Figure 5.3: Pulse input signals [114].

Hence, Paper E suggests a modified version to deal with this issue.

Sinusoidal Input

Five parameters *period*, *amplitude*, *base*, *delay* and *decay* are used to define a sinusoidal input signal, as shown in Figure 5.4. The amplitude refers to the maximum distance from *base*. The *period* gives the time from one peak to the next. The right graph shows an exponentially decaying sine wave. In Paper E, the parameter *frequency* is used which is $period = \frac{2\pi}{frequency}$. To avoid the mentioned problem in the pulse generator when both *base* and *amplitude* are used, the sinusoidal generator is parameterized in a similar model to the pulse generator in Paper E.

Other types of input signals, not listed in this thesis, can also be used for falsification. Paper E evaluates the effect of different input generators on the falsification performance from experimental results and also coverage measures.

5.3 Testing Coverage Measures

The falsification process can only prove the presence of faults, not absence. Hence, if a counterexample is not identified during the falsification runs, this does not mean that violations of the specifications do not exist. After extensive testing with a falsification, it is useful for a designer to find out how much

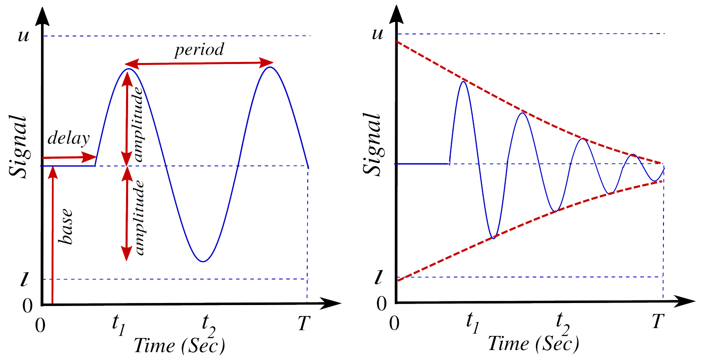

Figure 5.4: Sinusoidal input signal.

of the search space that has been covered. The works in [24], [111] focus on state coverage measures, these are measures to determine what portion of a system's state space is covered by a test suite. In [115], the authors focused on the coverage of input signal spaces. Paper E evaluates and suggests new coverage test cases on both space and time, and frequency domains.

CHAPTER 6

Benchmark Problems

This chapter introduces the benchmark problems used for evaluation in the attached papers. It also describes how the benchmarking was implemented and evaluated.

6.1 Benchmark Problems

Simulation-based falsification is considered for all benchmark problems from the ARCH19 workshop [55] and the benchmark problems of Paper B.

For the ARCH19 benchmark problems [55], two variants of input signals are considered, we refer to them as Instance 1 and Instance 2, respectively.

- *Instance 1: Arbitrary piecewise continuous input signals.* The input signals were allowed to be freely chosen but with a finite number of discontinuities in the ARCH19 competition.
- *Instance 2: Constrained input signals.* The input signal format is fixed, but discontinuities are allowed. An example input signal would be a piecewise constant signal with k uniformly spaced control points, with ranges for each input dimension.

For both variants of input signals, the same parameterization as used by Breach and S-TaLiRo in [55] is used in the evaluations.

Each benchmark example is introduced below briefly, although their specifications are introduced in Paper D. The evaluation in Paper E was done on benchmark [56], which includes four extra specifications rather than in Paper D. Hence, these four specifications will be introduced in this chapter as well.

Automatic Transmission

Automatic transmission (*AT*) has two inputs: $0 \leq \textit{throttle} \leq 100$ and $0 \leq \textit{brake} \leq 325$ where both inputs can be active at the same time. The controller in this model selects a gear from 1 to 4, depending on the inputs (throttle, brake), rotations per minute (ω), car speed (v), and the current engine load [116]. This problem includes ten specifications and has been evaluated with two input instances:

- Instance 1: Both *throttle* and *brake* are piecewise constant input signals, meaning *previous* interpolation. *throttle* has three signal domain control points, which means a piecewise signal with three different values, plus two control points for time intervals. *brake* has two signal domain control points, two piecewise signals with different values, plus one control point for the time interval.

- Instance 2: Constrained input signals with discontinuities, i.e., a piecewise constant input signal with 20 control points for each input signal.

Chasing Cars

Five cars are used in the chasing cars (*CC*) [117] model, where the first car is controlled by inputs $0 \leq \textit{throttle} \leq 1$ and $0 \leq \textit{brake} \leq 1$, and the algorithm of Hu et al controls the other four. The location of the five cars y_1, y_2, y_3, y_4, y_5 is the system output. Six specifications included in this problem with two instances of inputs are:

- Instance 1: The input specifications allow any piecewise continuous distributed equally where four control points for each input are considered.

- Instance 2: The input specifications constrain inputs to piecewise constant signals with control points every 5 seconds, which means 20 control points for each input.

Wind Turbine

A simplified wind turbine (WT) model [118] has only the wind speed v as input. The outputs are blade pitch angle θ, generator torque $M_{g,d}$, rotor speed Ω, and demanded blade pitch angle θ_d. The single input signal for this problem is constrained $8.0 \leq v \leq 16.0$, with four specifications in total. The input signal of this problem is piecewise with *spline* interpolation. There are 126 control points for this problem.

Neural Network Controller

Based on the NARMA-L2 [119] controller, the neural network controller (*NN*) works for a system that levitates a magnet above an electromagnet at a reference position. A reference value *Ref* for the position, where $1 \leq Ref \leq 3$ (or $1.95 \leq Ref \leq 2.05$), is the only input of this model. The current position of the levitating magnet *Pos* is the output. There are three specifications for this problem with two variants of inputs considered for this model are:

- Instance 1: The input specification requires discontinuities to be at least 3 time units long, i.e., 12 signal domains.

- Instance 2: An input signal with exactly 3 constant segments, i.e., 3 signal domains, is required.

Fuel Control of an Automotive Power Train

Fuel control of an automotive power train (AFC) is modeled on [120]. There is one constrained input signal that fixes the throttle θ to be piecewise constant with 10 uniform segments over a time horizon of 0 with two modes that each varies in $[0, 61.1]$ (or $[61.2, 81.2]$) and the engine speed ω to be a constant signal with $900 \leq \omega \leq 1100$. Three specifications are included in this problem.

Aircraft Ground Collision Avoidance System

The aircraft ground collision avoidance system (*F16*) aircraft and its inner-loop controller are modeled for Ground Collision avoidance. 16 continuous variables with piecewise nonlinear differential equations are modeled [121]. The system is required to always avoid hitting the ground during its maneuvers, starting from all the initial conditions $0.2\pi \leq roll \leq 0.2833\pi$, $-0.5\pi \leq pitch \leq -0.54\pi$, and $0.25\pi \leq yaw \leq 0.375\pi$. This problem has only one specification.

Steam Condenser with Recurrent Neural Network Controller

The steam condenser with a recurrent neural network controller (*SC*) is a dynamic model of a steam condenser based on energy balance and cooling mass water balance, controlled with a Recurrent Neural network in feedback [122]. This problem includes only one specification. The input can vary in the $[3.99, 4.01]$ range, and there are two variants of input signals for only one specification.

- Instance 1: The input signal is piecewise constant with 12 evenly spaced segments.
- Instance 2: The input signal is piecewise constant with 20 evenly spaced segments.

Automatic Transmission (*AT′*)

The automatic transmission (AT'), presented in Section 6.1, and evaluated in Paper C, Paper D, and Paper E, is different from the AT ARCH problem. It means that it has different specifications and input ranges. The inputs to the model are the $0 \leq throttle \leq 100$ and $0 \leq brake \leq 500$ of a vehicle. The model outputs are the vehicle speed v, the engine speed ω, and the gear; see [53] for details. There are 7 control points for *throttle* and 3 for *brake* distributed uniformly with *pchip* interpolation.

Potentially, this problem has eight different specifications, but some of the specifications have been evaluated with varying time intervals in the STL formula and simulation times. Hence, the number of evaluated specifications for this problem is fifteen.

Third Order Modulator

The third order $\Delta - \Sigma$ modulator is a model of a technique for analog to digital conversion [54]. It has one input U, three states x_1, x_2, x_3, and three initial conditions $x_1^{init}, x_2^{init}, x_3^{init}$. Three different input ranges are assumed for only one specification of this problem. These ranges are: $-0.35 \leq U \leq 0.35$, $-0.40 \leq U \leq 0.40$, and $-0.45 \leq U \leq 0.45$.

Static Switched

The static switched (SS) system is a model without any dynamics that is included as a simple case. The model is inspired from [26]. In this problem, the gradient cannot point toward the falsification area, which means that the optimization methods get the wrong gradient direction. This example has two inputs that can vary in the range $[-1, 1]$. This problem has only one specification and three different values are considered for parameter $thresh = 0.7, 0.8, 0.9$.

The four extra specifications used in Paper E, are from [56], and are listed in Table. 6.1. These specifications are related to *AT*, *AFC*, *CC*, and *NN*.

Table 6.1: Specifications to falsify for four extra evaluated in Paper E. Note that the specifications $\varphi_7^{AT}, \varphi_8^{AT}, \varphi_9^{AT}$ are defined in Paper D.

Specifications	STL Formula
φ_{10}^{AT}	$\varphi_7^{AT} \wedge \varphi_8^{AT} \wedge \varphi_9^{AT}$
φ_3^{AFC}	$\Box_{[11,50]} \lvert\mu\rvert < 0.007$ $61.2 \leq \theta \leq 81.2$
φ_6^{CC}	$\bigwedge_{i=1,\ldots,4} \Box_{[0,50]} \Big((y_{i+1} - y_i) > 7.5\Big)$
φ_3^{NN}	$\Diamond_{[0,1]}(Pos > 3.2) \wedge \Diamond_{[}1, 1.5] \Big(\Box[0, 0.5] \Big(1.75 < Pos < 2.25\Big)\Big)$ $\wedge \Box_{[}2, 3] \Big(1.825 < Pos < 2.175\Big)$ $1.95 \leq Ref \leq 2.05$

6.2 Large-Scale Testing

A key strength of using a model-based development method is that the models can be used for various purposes, including simulation, control design, and testing. The closed-loop behavior can be evaluated when both the control logic, the physical parts, and the environment have models that support high-fidelity simulation.

Testing benefits from running a large number of simulations. If only software models are needed to run the simulations, testing can take advantage of computing clusters to run the models in parallel, using thousands of computing nodes.

To support the evaluation of the different strategies used in this thesis, we have set up a system such that all benchmark problems in this thesis can be executed on resources at High-Performance Computing Center North (HPC2N), Umeå University, a Swedish national center for Scientific and Parallel Computing (SNIC)[1].

6.3 Benchmark Setup and Evaluation

This section briefly presents the setup of the presented methods in the attached Paper B, Paper C, Paper D, and Paper E. The maximum number of simulations of each falsification problem is set to 1000. Since the optimization methods contain stochastic features, the falsification is run 20 times for each optimization method and objective function. Two values for every evaluated method are presented in each table of the mentioned papers. The first is the relative success rate for falsification expressed in percent. For each parameter value and specification, there will be 20 falsification runs, so the success rate will be a multiple of 5%. The second value, inside parentheses, is the average number of simulations (rounded) per successful falsification. In Paper E, 5 runs are considered for each evaluated example and specification. This was due to a lot of experimental results being needed, while there is the limitation of computing hours running problems on the cluster. The success rate in Paper E is 20%.

The suggested methods in the attached papers are evaluated based on the setups mentioned above. In Paper C, different optimization methods are eval-

[1]`https://www.hpc2n.umu.se/`

uated on the problems. It should be mentioned here that in Paper C, because of the journal's page number limitation, the results of *Max* are removed, and only *Additive* results are presented. A complete evaluation of both semantics can be found in [123].

As was fully discussed in Chapter 5, it is often challenging to define suitable input generator parameters since the dynamics of the systems are complex and often unknown, especially for a large-scale SUT. To be able to write Paper E, an experimental setup has been done in a Technical Report [124] to show which pulse parameters are most important to falsify the benchmark problems. In this report, for this purpose, the pulse generator suggested in Paper E is applied to the benchmark problems evaluated in Paper C using the default TuRBO with Thompson Sampling [88] as the acquisition function. In the technical report all benchmark problems were first evaluated using only one input parameter for optimization. Then, to show which combination of input generators might work better, we evaluated if at least one input parameter is successful in falsifying a specification; regardless of the success rate, their combination will also be successful. After finding the best combination, we did the experimental setup, and consequently, based on this assessment, the best combination of different inputs is presented in Paper E. The reader is referred to the technical report [124] to see the effect of dimensionality on the optimization process for falsification.

CHAPTER 7

Summary of Papers

The main focus of this work is to improve the falsification performance by reducing the number of simulations needed to falsify a problem. This chapter provides a summary of the included papers.

7.1 Paper A

Zahra Ramezani, Nicholas Smallbone, Martin Fabian, Knut Åkesson
Evaluating Two Semantics for Falsification using an Autonomous Driving Example
in 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), IEEE, vol. 1, pp. 386-391, 2019.
©2019 IEEE DOI: 10.1109/INDIN41052.2019.8972229.

The optimization-based falsification process is guided by a quantitative semantics. Different quantitative semantics are introduced in the literature. It is important to understand the strengths and weaknesses of the different semantics. This paper evaluates two, the *Max* and *MARV* semantics, on an adaptive cruise controller of an autonomous car. The evaluation shows that the *Max* semantics results in a constant objective value for some parts of the parameter space. In contrast, *MARV* results in non-constant objective value, i.e., it has a direction that can be exploited by an optimization algorithm. This paper shows the importance of choosing a suitable semantics for the problem at hand.

7.2 Paper B

Zahra Ramezani, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, Knut Åkesson
Multiple Objective Functions for Falsification of Cyber-Physical Systems
IFAC-PapersOnLine, vol. 53, no. 4, pp. 417–422, 2020
©https://doi.org/10.1016/j.ifacol.2021.04.040.

The results of Paper A show that the efficiency of the falsification is affected by the quantitative semantics used. This paper suggests the use of multiple quantitative semantics during the falsification process. The combination of *Max* and *Additive* semantics is evaluated on a set of benchmark problems. The evaluation shows that using multiple quantitative semantics can reduce the number of simulations necessary to falsify a specification compared to when a single quantitative semantics is used. We show how the Nelder-Mead algorithm can be extended to support the multiple objective functions defined by the different quantitative semantics.

7.3 Paper C

Zahra Ramezani, Koen Claessen, Nicholas Smallbone, Martin Fabian, Knut Åkesson
Testing Cyber-Physical Systems Using a Line-Search Falsification Method
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 8, pp. 2393–2406, 2022
©IEEE DOI: 10.1109/TCAD.2021.3110740.

Falsification of cyber-physical systems can be done using random search methods or by exploring the parameter space in a more structured way, for example, by using optimization. In this work, we evaluate the performance of a simple strategy based on combining random parameters together with lower and upper values of in the allowed parameter range. The evaluation using benchmark problems shows that a simple strategy works well. We propose using this approach as a baseline method when evaluating falsification methods. The second part of this paper discusses a simple gradient-free optimization method, named line-search falsification, that does optimization in falsification over a line. This method is compared to the Nelder-Mead and SNOBFIT methods. The comparison shows that the line-search optimization enhances the efficiency of the falsification while still having a simple implementation.

7.4 Paper D

Zahra Ramezani, Kenan Šehić, Luigi Nardi, Knut Åkesson
Falsification of Cyber-Physical Systems using Bayesian Optimization
Submitted for possible journal publication, 2022.

Falsification of specifications for cyber-physical systems (CPSs) can be done using optimization methods. In this work, we explore Bayesian optimization (BO), a sample-efficient method that learns a surrogate function that models the relationship between the evaluation of the specification and the input parameterization. We use two prominent BO methods, TuRBO and πBO, and compare their performance with the state-of-the-art methods for falsification on standard benchmark examples. TuRBO is less sensitive to the complexity of the CPS as a local search approach and nullifies the bias for exploiting the edges. The comparison shows that TuRBO falsifies some of the hard to falsify benchmark problems. On the other hand, πBO allows the test engineers to include their prior knowledge about falsification where there is a higher chance of falsifying the system. To reduce the number of simulations in this paper, searching more corners or close to them is injected in BO as prior knowledge in πBO. Based on experiments, the BO methods have a clear advantage over previously presented methods.

7.5 Paper E

Zahra Ramezani, Alexandre Donzé, Martin Fabian, Knut Åkesson
On Input Generators for Cyber-Physical Systems Falsification
Submitted for possible journal publication, 2022.

Falsification is a black-box approach where only the input-output behavior of the system under test is observed. The falsification process searches for inputs that falsify the specification. This work focuses on the input generators, a mapping from parameters used as optimization variables to signals that form the actual test cases for the system. Different input generators, pulse, sinusoidal, and piecewise, are proposed and evaluated on benchmark problems for practical performance and testing coverage measures. Based on the evaluation, pulse input generators perform well on all benchmark problems. In particular, all benchmark problems are falsified in at least one of the five falsification runs we did when used together with Bayesian optimization. The sinusoidal generator in general, not as efficient as the pulse input generator and piecewise with *previous* interpolation.

CHAPTER 8

Answer to Research Questions and Conclusions

This thesis investigates optimization-based falsification of cyber-physical systems. The falsification process is enhanced by (i) using multiple quantitative semantics, (ii) new optimization methods based on a direct-search approach, (iii) a model-based optimization method that actively learns a surrogate function, and (iv) by carefully parameterizing input signals that have few parameters but are still rich enough to be able to falsify specifications. We evaluate the proposed methods using standard benchmark problems.

An example from an autonomous driving model is analyzed to understand better the consequences of using different quantitative semantics. The evaluation results show that the objective function values do not change in some regions of the parameter space for *Max* semantics. By using the *MARV* semantics, the objective function values for samples in the mentioned regions change, giving a sense of direction to the optimization algorithm to exploit. The results from Paper A leads to the idea of Paper B, where the use of multiple objective functions during the optimization procedure is evaluated. The evaluation shows the potential of using multiple quantitative semantics, but how to best integrate multiple semantics is future research.

The optimization method used affects the efficiency of the falsification process. Evaluation results on benchmark problems show that for some models and specifications, a surprisingly efficient falsification method combines evaluating extreme parameter values (called corners) with randomly generated parameter values. Hence, this idea can be used as a baseline method that more involved falsification methods could be evaluated against. This baseline method is suggested in Paper C, based on corners and random parameter values. The evaluation shows that optimization-based methods outperform the baseline method for more challenging falsification problems, while the baseline method is efficient for easier falsification problems.

A new direct-search-based and gradient-free optimization method developed specifically for falsification problems is proposed in Paper C. The method is designed such as to use prior information that corner points are efficient for many falsification problems. Although the method in Paper C improves the falsification performance compared to previously used methods, it does not make an effort to learn the objective function. This motivated the investigation of Bayesian optimization (BO) methods, which are model-based optimization methods that learn a surrogate model of the objective function in order to better select the next sample of the parameter values to evaluate. BO has received relatively little attention in the falsification community. In our work, we investigate BO methods and the choice of acquisition functions on the performance of the falsification process. An acquisition function tailor-made for falsification problems and using prior knowledge about promising parameter values are developed in Paper D.

The performance of optimization-based methods typically suffers from having a large number of parameters. We thus proposed input signal generators with a simple but rich structure to keep the number of optimization parameters down. We show that these input signals are rich enough to allow falsification of specifications that both require specific values of signals but also input signals with different frequencies.

With the above investigations, we attempt to answer the research questions formulated in Chapter 1:

RQ1. *What are the strengths and weaknesses of the different quantitative semantics for the falsification of cyber-physical systems?*

Paper A answers this question by evaluating a model of an adaptive cruise controller used for autonomous driving. The performance was evaluated for two semantics, *Max*, and *MARV*. In Paper A, it is shown that using *Max* semantics results in constant objective values for regions of the parameter space. Thus there is no valuable information for an optimization algorithm to guide the direction to pick the next sample to evaluate. On the other hand, *MARV* results in objective values that guide the optimization to parameter ranges where the specification can be falsified. The autonomous driving example illustrates some shortcomings with the standard quantitative semantics, i.e., *Max*, and shows a potential value of using alternative quantitative semantics. Unfortunately, it is hard to make any definite conclusions about when to use which quantitative semantics.

RQ2. *How does the combination of using different quantitative semantics affect the efficiency of falsification of cyber-physical systems?*

Evaluation results from Paper A show that using a single objective function may cause the optimization to have no clear direction to follow during the falsification process or even be directed in the wrong direction. Thus, a combination of using multiple quantitative semantics, resulting in different objective function values, during the optimization is proposed in Paper B. Multiple objective functions are combined with an extended version of the Nelder-Mead optimization method that uses multiple objective functions. This is done so that the optimizer avoids using semantics that does not have a clear direction to get closer to a falsification point. Using multiple quantitative semantics guides the optimization better than single quantitative semantics. We show that, for the evaluated benchmark problems, the use of multiple semantics improves the efficiency of the falsification process.

RQ3. *How do different test-case generation methods affect the efficiency of the falsification process?*

In this thesis, we consider optimization-based falsification. However, many specifications can be falsified quickly by evaluating combinations of extreme values, i.e., the min and max values in the allowed parameter ranges, or by random parameter combinations. Using an optimization-based approach means that it is necessary to have quantitative semantics defined. However, this thesis shows that the preferred quantitative semantics are not easily chosen. Optimization-based methods also have an additional cost related to solving the optimization problem. For falsification of cyber-physical systems, gradients are not generally available since only simulations are possible for non-trivial systems. Thus gradient-free optimization methods have to be used. To motivate the complexity and cost of using an optimization-based falsification approach, an optimization-based approach should be significantly more efficient than a non-optimization-based approach. No baseline method for comparison is recognized within the optimization-based falsification research community. Thus, we propose a non-optimization-based baseline method suitable for comparison. This baseline approach is a hybrid method that combines extreme parameter values with a random strategy for selecting parameter values within the allowed parameter ranges. The evaluation on standard benchmark problems shows that this approach performs well on many problems except the hardest ones. This approach and the evaluations are presented in Paper C.

On the other hand, there are also problems where optimization-based methods perform significantly better. In Paper C, we suggest a new simple direct-search-based optimization method. The optimization uses randomly generated lines in the parameter space, with a local search and an emphasis on evaluating extreme parameter values. The evaluation of benchmark problems shows that this method enhances the falsification process while still having a simple implementation.

In Paper D, we have also investigated Bayesian optimization-based methods that learn a surrogate objective function and use that for selecting the following sample for evaluation. We show that using prior knowledge with a carefully designed acquisition function that favors falsification can significantly improve the falsification process compared to other previously proposed methods.

The results from Paper D indicate that a carefully chosen formulation based on Bayesian optimization is the new start-of-art optimization method for simulation-based falsification of CPSs.

RQ4. *How do different input parameterizations affect the falsification process?*

How to parameterize input signals for optimization-based falsification has received little attention in the scientific community. However, this is a highly relevant problem because optimization methods suffer from the curse of dimensionality and do not scale well as the number of optimization parameters increases. Thus, we investigate different parametrizations of input signals intending to have few parameters but still with high coverage of different values and frequencies. A periodic input generator is proposed in [114], which can falsify problems that previous optimization-based methods failed to falsify. This method was a motivation to investigate different input generators in Paper E. While pulse generators provide average performance in both the space and time and frequency domain coverage, pulse generators together with a Bayesian optimization algorithm were able to falsify all benchmark problems. The sinusoidal generator was the best for covering space and frequency but was less efficient than the other input generators for falsification.

Combined, the experiments conducted in this thesis have improved our understanding of optimization-based falsification of cyber-physical systems. The increased understanding has resulted in several proposed methods that result in a significantly improved falsification process. We hope this will have consequences not only on the scientific community but also on industrial applications where efficient falsification processes are in much need, given the increasing complexity of cyber-physical systems.

8.1 Future Work

For future work, it would be interesting to continue the work on enhancing the falsification of CPSs, particularly focusing on the optimization part. While we only assume a black-box model in this work, in many cases, we have grey-box models available since parts of the systems are often expressed using high-level domain specific language or by models in machine learning frameworks. An interesting problem is to analyze further how we can exploit these models during the falsification process. One possibility would be to exploit the model more by estimating gradients directly from the model. Also, for future work, it would be interesting to apply the methods to autonomous systems that include both perception, decision, and actuation sub-systems. Especially the sensor aspects when using optical sensors, radar, or lidar sensors might pose a challenge for falsification due to the very large input space.

References

[1] E. A. Lee, “Cyber physical systems: Design challenges,” in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, IEEE, pp. 363–369, 2008.

[2] J. Shi, J. Wan, H. Yan, and H. Suo, “A survey of cyber-physical systems,” in *2011 international conference on wireless communications and signal processing (WCSP)*, IEEE, pp. 1–6, 2011.

[3] P. Fritzson and V. Engelson, “Modelica—a unified object-oriented language for system modeling and simulation,” in *European Conference on Object-Oriented Programming*, Springer, 1998, pp. 67–90.

[4] MathWorks, *Simscape—model and simulate multidomain physical systems*, `https://www.mathworks.com/products/simscape.html`, Online: accessed 1 September, 2022.

[5] ——, *Simulink is for model-based design*, `https://www.mathworks.com/products/simulink.html`, Online: accessed 1 September, 2022.

[6] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd. Prentice Hall, 1988, ISBN: 0131103709.

[7] N. D. Matsakis and F. S. Klock II, “The rust language,” in *ACM SIGAda Ada Letters*, ACM, vol. 34, 2014, pp. 103–104.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*, Software available from tensorflow.org, 2015.

[9] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[10] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[11] J.-C. Filliâtre, "Deductive software verification," *International Journal on Software Tools for Technology Transfer*, vol. 13, no. 5, pp. 397–403, 2011.

[12] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 2000.

[13] E. M. Clarke, O. Grumberg, and D. E. Long, "Model checking and abstraction," *ACM transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 5, pp. 1512–1542, 1994.

[14] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, "Deductive software verification—the KeY book," *Lecture notes in computer science*, vol. 10001, 2016.

[15] W. Bibel, *Automated theorem proving*. Springer Science & Business Media, 2013.

[16] A. Platzer, *Logical Foundations of Cyber-Physical Systems*, 1st edition. Springer, 2018, ISBN: 3319635875.

[17] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, 2016.

[18] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda, "Conformance testing as falsification for cyber-physical systems," *CoRR*, vol. abs/1401.5200, 2014.

[19] H. Araujo, G. Carvalho, M. Mohaqeqi, M. R. Mousavi, and A. Sampaio, "Sound conformance testing for cyber-physical systems: Theory and implementation," *Science of Computer Programming*, vol. 162, pp. 35–54, 2018.

[20] H. Abbas, "Test-based falsification and conformance testing for cyber-physical systems," Ph.D. dissertation, Arizona State University, Tempe, USA, 2015.

[21] K. J. Hayhurst and D. S. Veerhusen, "A practical approach to modified condition/decision coverage," in *20th DASC. 20th Digital Avionics Systems Conference (Cat. No. 01CH37219)*, IEEE, vol. 1, 2001, 1B2–1.

[22] ISO, "ISO 26262:2018–Road vehicles–Functional safety," *International Standard ISO/FDIS*, 2018.

[23] G. Gay, A. Rajan, M. Staats, M. Whalen, and M. P. Heimdahl, "The effect of program and model structure on the effectiveness of mc/dc test adequacy coverage," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 3, pp. 1–34, 2016.

[24] T. Dang and T. Nahhal, "Coverage-guided test generation for continuous and hybrid systems," *Formal Methods in System Design*, vol. 34, no. 2, pp. 183–213, 2009.

[25] T. Nahhal and T. Dang, "Test coverage for continuous and hybrid systems," in *International Conference on Computer Aided Verification*, Springer, 2007, pp. 449–462.

[26] A. Dokhanchi, A. Zutshi, R. T. Sriniva, S. Sankaranarayanan, and G. Fainekos, "Requirements driven falsification with coverage metrics," in *2015 International Conference on Embedded Software (EMSOFT)*, IEEE, 2015, pp. 31–40.

[27] J. Eddeland, J. G. Cepeda, R. Fransen, S. Miremadi, M. Fabian, and K. Åkesson, "Automated mode coverage analysis for cyber-physical systems using hybrid automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9260–9265, 2017.

[28] E. Bartocci, J. Deshmukh, A. Donzé, *et al.*, "Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications," in *Lectures on Runtime Verification*, ser. Lecture Notes in Computer Science, vol. 10457, 2018, pp. 135–175.

[29] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[30] O. Maler and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Springer, 2004, pp. 152–166.

[31] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, “What’s decidable about hybrid automata?” In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC ’95, Las Vegas, Nevada, USA: ACM, 1995, pp. 373–382.

[32] G. Ernst, S. Sedwards, Z. Zhang, and I. Hasuo, “Fast falsification of hybrid systems using probabilistically adaptive input,” in *International Conference on Quantitative Evaluation of Systems*, Springer, pp. 165–181, 2019.

[33] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo, “Two-layered falsification of hybrid systems guided by monte carlo tree search,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2894–2905, 2018.

[34] G. Ernst, S. Sedwards, Z. Zhang, and I. Hasuo, “Falsification of hybrid systems using adaptive probabilistic search,” *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 31, no. 3, pp. 1–22, 2021.

[35] T. Akazaki, S. Liu, Y. Yamagata, Y. Duan, and J. Hao, “Falsification of cyber-physical systems using deep reinforcement learning,” in *International Symposium on Formal Methods*, Springer, 2018, pp. 456–465.

[36] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, “S-TaLiRo: A tool for temporal logic falsification for hybrid systems,” in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer, 2011, pp. 254–257.

[37] G. Fainekos, B. Hoxha, and S. Sankaranarayanan, “Robustness of specifications and its applications to falsification, parameter mining, and runtime monitoring with s-taliro,” in *Runtime Verification*, B. Finkbeiner and L. Mariani, Eds., Cham: Springer International Publishing, 2019, pp. 27–47.

[38] B. Hoxha, A. Dokhanchi, and G. Fainekos, “Mining parametric temporal logic properties in model-based design for cyber-physical systems,” vol. 20, no. 1, 2018, ISSN: 1433-2779.

[39] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *Runtime Verification*, B. Bonakdarpour and S. A. Smolka, Eds., Cham: Springer International Publishing, 2014, pp. 231–246.

[40] L. Mathesen, G. Pedrielli, S. H. Ng, and Z. B. Zabinsky, "Stochastic optimization with adaptive restart: A framework for integrated local and global learning," *Journal of Global Optimization*, vol. 79, no. 1, pp. 87–110, 2021.

[41] L. Mathesen, G. Pedrielli, and G. Fainekos, "Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2021, pp. 732–737.

[42] C. Menghi, S. Nejati, L. Briand, and Y. I. Parache, "Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, IEEE, 2020, pp. 372–384.

[43] M. Waga, "Falsification of cyber-physical systems with robustness-guided black-box checking," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–13.

[44] D. Peled, M. Y. Vardi, and M. Yannakakis, "Black box checking," in *Formal Methods for Protocol Engineering and Distributed Systems*, Springer, 1999, pp. 225–240.

[45] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*, Springer, 2010, pp. 167–170.

[46] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1704–1717, 2015.

[47] J. L. Eddeland, K. Claessen, N. Smallbone, Z. Ramezani, S. Miremadi, and K. Åkesson, "Enhancing temporal logic falsification with specification transformation and valued booleans," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 5247–5260, 2020.

[48] Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, "Effective hybrid system falsification using Monte Carlo tree search guided by QB-robustness," in *International Conference on Computer Aided Verification*, Springer, 2021, pp. 595–618.

[49] T. Dreossi, D. J. Fremont, S. Ghosh, *et al.*, "VERIFAI: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *International Conference on Computer Aided Verification*, Springer, 2019, pp. 432–442.

[50] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: A language for scenario specification and scene generation," New York, NY, USA: Association for Computing Machinery, 2019, ISBN: 9781450367127.

[51] A. Aerts, M. R. Mousavi, and M. Reniers, "A tool prototype for model-based testing of cyber-physical systems," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2015, pp. 563–572.

[52] M. Althoff, "An introduction to CORA 2015," in *Proc. of the workshop on applied verification for continuous and hybrid systems*, 2015, pp. 120–151.

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

[54] T. Dang, A. Donzé, and O. Maler, "Verification of analog and mixed-signal circuits using hybrid system techniques," in *International Conference on Formal Methods in Computer-Aided Design*, Springer, 2004, pp. 21–36.

[55] G. Ernst, P. Arcaini, A. Donzé, *et al.*, "ARCH-COMP 2019 category report: Falsification," in *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 61, EasyChair, 2019, pp. 129–140.

[56] G. Ernst, P. Arcaini, I. Bennani, *et al.*, "ARCH-COMP 2021 category report: Falsification with validation of results.," in *ARCH@ ADHS*, 2021, pp. 133–152.

[57] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, Jan. 1965, ISSN: 0010-4620.

[58] K. Claessen, N. Smallbone, J. Eddeland, Z. Ramezani, and K. Åkesson, "Using valued booleans to find simpler counterexamples in random testing of cyber-physical systems," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 408–415, 2018.

[59] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems*, K. Chatterjee and T. A. Henzinger, Eds., Berlin, Heidelberg: Springer, 2010, pp. 92–106.

[60] J. Eddeland, S. Miremadi, M. Fabian, and K. Åkesson, "Objective functions for falsification of signal temporal logic properties in cyber-physical systems," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017, pp. 1326–1331.

[61] T. Akazaki and I. Hasuo, "Time robustness in MTL and expressivity in hybrid system falsification," in *International Conference on Computer Aided Verification*, Springer, 2015, pp. 356–374.

[62] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius, "Functional gradient descent method for metric temporal logic specifications," in *2014 American Control Conference*, 2014, pp. 2312–2317.

[63] Y. S. R. Annapureddy and G. E. Fainekos, "Ant colonies for temporal logic falsification of hybrid systems," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 91–96.

[64] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, "Stochastic local search for falsification of hybrid systems," in *International Symposium on Automated Technology for Verification and Analysis*, Springer, 2015, pp. 500–517.

[65] J. L. Eddeland, S. Miremadi, and K. Åkesson, "Evaluating optimization solvers and robust semantics for simulation-based falsification," *EPiC Series in Computing*, vol. 74, pp. 259–266, 2020.

[66] W. Huyer and A. Neumaier, “SNOBFIT–stable noisy optimization by branch and fit,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 35, no. 2, pp. 1–25, 2008.

[67] R. Smith and H. Romeijn, “Simulated annealing for constrained global optimization,” *Journal of Global Optimization*, vol. 5, Sep. 1994.

[68] N. Hansen, “The CMA evolution strategy: A comparing review,” in *Towards a new evolutionary computation*, Springer, pp. 75–102, 2006.

[69] H. Araujo, G. Carvalho, M. R. Mousavi, and A. Sampaio, “Multi-objective search for effective testing of cyber-physical systems,” in *International Conference on Software Engineering and Formal Methods*, Springer, 2019, pp. 183–202.

[70] A. Pnueli, “The temporal logic of programs,” in *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*, 1977, pp. 46–57.

[71] R. Alur, T. Feder, and T. A. Henzinger, “The benefits of relaxing punctuality,” *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[72] L. Brim, P. Dluhoš, D. Šafránek, and T. Vejpustek, “STL*: Extending signal temporal logic with signal-value freezing operator,” *Information and computation*, vol. 236, pp. 52–67, 2014.

[73] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka, “On temporal logic and signal processing,” in *International Symposium on Automated Technology for Verification and Analysis*, Springer, 2012, pp. 92–106.

[74] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, “Reactive synthesis from signal temporal logic specifications,” in *Proceedings of the 18th international conference on hybrid systems: Computation and control*, ACM, 2015, pp. 239–248.

[75] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” *CoRR*, vol. abs/1708.06374, 2017.

[76] J. W. Duran and S. C. Ntafos, “An evaluation of random testing,” *IEEE transactions on Software Engineering*, no. 4, pp. 438–444, 1984.

[77] A. Arcuri, M. Z. Iqbal, and L. Briand, “Random testing: Theoretical results and practical implications,” *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 258–277, 2012.

[78] K. Claessen and J. Hughes, “QuickCheck: A lightweight tool for random testing of haskell programs,” in *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, 2000, pp. 268–279.

[79] S. Amaran, N. Sahinidis, B. Sharda, and S. Bury, “Simulation optimization: A review of algorithms and applications,” *Annals of Operations Research*, vol. 240, May 2016.

[80] C. Audet and W. Hare, *Derivative-free and blackbox optimization.* Springer, 2017, vol. 2.

[81] R. Hooke and T. A. Jeeves, “‘Direct search’ solution of numerical and statistical problems,” *Journal of the ACM (JACM)*, vol. 8, no. 2, pp. 212–229, 1961.

[82] R. M. Lewis, V. Torczon, and M. W. Trosset, “Direct search methods: Then and now,” *Journal of computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 191–207, 2000.

[83] H. P. Gavin, “The Nelder-Mead algorithm in two dimensions,” *CEE 201L. Duke U*, 2013.

[84] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of Bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[85] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[86] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. MIT Press, 2006, p. 248.

[87] H. Su and H. Zhang, “On stationary periodic kernels,” in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2019, pp. 43–46.

[88] W. R. Thompson, “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples,” *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[89] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Madison, WI, USA: Omnipress, 2010, pp. 1015–1022.

[90] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, pp. 97–106, 1964.

[91] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[92] B. Echard, N. Gayton, and M. Lemaire, "AK-MCS: An active learning reliability method combining kriging and Monte Carlo simulation," *Structural Safety*, vol. 33, no. 2, pp. 145–154, 2011.

[93] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, vol. 25, 2012.

[94] Z. Wang and N. de Freitas, "Theoretical analysis of Bayesian optimisation with unknown gaussian process hyper-parameters," *arXiv preprint arXiv:1406.7758*, 2014.

[95] M. Malu, G. Dasarathy, and A. Spanias, "Bayesian optimization in high-dimensional spaces: A brief survey," in *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, IEEE, 2021, pp. 1–8.

[96] A. Nayebi, A. Munteanu, and M. Poloczek, "A framework for Bayesian optimization in embedded subspaces," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 4752–4761.

[97] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[98] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[99] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, pp. 361–387, 2016.

[100] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local Bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[101] J. Djolonga, A. Krause, and V. Cevher, "High-dimensional gaussian process bandits," *Advances in neural information processing systems*, vol. 26, 2013.

[102] M. Zhang, H. Li, and S. Su, "High dimensional Bayesian optimization via supervised dimension reduction," *arXiv preprint arXiv:1907.08953*, 2019.

[103] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *International Colloquium on Automata, Languages, and Programming*, Springer, 2002, pp. 693–703.

[104] B. Letham, R. Calandra, A. Rai, and E. Bakshy, "Re-examining linear embeddings for high-dimensional Bayesian optimization," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, vol. 33, 2020, pp. 1546–1558.

[105] K. Kandasamy, J. Schneider, and B. Póczos, "High dimensional Bayesian optimisation and bandits via additive models," in *International conference on machine learning*, PMLR, 2015, pp. 295–304.

[106] C.-L. Li, K. Kandasamy, B. Póczos, and J. Schneider, "High dimensional Bayesian optimization via restricted projection pursuit models," in *Artificial Intelligence and Statistics*, PMLR, 2016, pp. 884–892.

[107] Y.-X. Yuan, "A review of trust region algorithms for optimization," in *Proceedings of the 4th International Congress on Industrial & Applied Mathematics (ICIAM 99)*, 2000, pp. 271–282.

[108] Y. Diouane, V. Picheny, R. L. Riche, and A. S. Di Perrotolo, "TREGO: A trust-region framework for efficient global optimization," *arXiv preprint arXiv:2101.06808*, 2021.

[109] A. Souza, L. Nardi, L. B. Oliveira, K. Olukotun, M. Lindauer, and F. Hutter, "Bayesian optimization with a prior for the optimum," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2021, pp. 265–296.

[110] C. Hvarfner, D. Stoll, A. Souza, M. Lindauer, F. Hutter, and L. Nardi, "πBO: Augmenting acquisition functions with user beliefs for Bayesian optimization," *arXiv preprint arXiv:2204.11051*, 2022.

[111] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods Symposium*, Springer, 2015, pp. 127–142.

[112] A. Aerts, B. T. Minh, M. R. Mousavi, and M. A. Reniers, "Temporal logic falsification of cyber-physical systems: An input-signal-space optimization approach," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2018, pp. 214–223.

[113] B. Barbot, N. Basset, T. Dang, A. Donzé, J. Kapinski, and T. Yamaguchi, "Falsification of cyber-physical systems with constrained signal spaces," in *NASA Formal Methods Symposium*, Springer, 2020, pp. 420–439.

[114] Z. Ramezani, A. Donzé, M. Fabian, and K. Åkesson, "Temporal logic falsification of cyber-physical systems using input pulse generators," *EPiC Series in Computing*, vol. 80, pp. 195–202, 2021.

[115] A. Adimoolam, T. Dang, A. Donzé, J. Kapinski, and X. Jin, "Classification and coverage-based falsification for embedded control systems," in *International Conference on Computer Aided Verification*, Springer, 2017, pp. 483–503.

[116] B. Hoxha, H. Abbas, and G. Fainekos, "Benchmarks for temporal logic requirements for automotive systems," in *ARCH@CPSWeek*, 2014.

[117] J. Hu, J. Lygeros, and S. Sastry, "Towards a theory of stochastic hybrid systems," in *Hybrid Systems: Computation and Control*, N. Lynch and B. H. Krogh, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 160–173.

[118] S. Schuler, F. D. Adegas, and A. Anta, "Hybrid modelling of a wind turbine," in *ARCH16. 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, G. Frehse and M. Althoff, Eds., ser. EPiC Series in Computing, vol. 43, EasyChair, 2017, pp. 18–26.

[119] MathWorks, *Design NARMA-L2 neural controller in Simulink*, `https://au.mathworks.com/help/deeplearning/ug/design-narma-l2-neural-controller-in-simulink.html`, Online: accessed 1 March 2021, 2020.

[120] A. Dokhanchi, S. Yaghoubi, B. Hoxha, *et al.*, "ARCH-COMP18 category report: Results on the falsification benchmarks," in *ARCH@ADHS*, 2018.

[121] G. Frehse, A. Abate, D. Adzkiya, *et al.*, "ARCH-COMP18 category report: Hybrid systems with piecewise constant dynamics," in *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, G. Frehse, Ed., ser. EPiC Series in Computing, vol. 54, EasyChair, 2018, pp. 1–13.

[122] S. Yaghoubi and G. Fainekos, "Gray-box adversarial testing for control systems with machine learning components," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 179–184.

[123] Z. Ramezani, K. Claessen, N. Smallbone, M. Fabian, and K. Åkesson, "Testing Cyber-Physical Systems Using a Line-Search Falsification Method," *10.36227/techrxiv.14555826.v3*, Jul. 2021.

[124] Z. Ramezani and K. Åkesson, "Technical report: The effect of input parameters on falsification of cyber-physical systems," *arXiv preprint arXiv:2209.07131*, 2022.