CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

WATER-MARK

LICENTIATE THESIS

ZAHRA RAMEZANI • Enhancing Temporal Logic Falsification of Cyber-Physical Systems • 2020

Enhancing Temporal Logic Falsification of Cyber-Physical Systems

using multiple objective functions and a new optimization method

ZAHRA RAMEZANI

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020
www.chalmers.se

THESIS FOR THE DEGREE OF LICENTIATE OF ENGINEERING

Enhancing Temporal Logic Falsification of Cyber-Physical Systems

Zahra Ramezani

CHALMERS
UNIVERSITY OF TECHNOLOGY
Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2020

Enhancing Temporal Logic Falsification of Cyber-Physical Systems

Zahra Ramezani

Copyright © 2020 Zahra Ramezani
All rights reserved.

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, November 2020

To my parents

Abstract

Cyber-physical systems (CPSs) are engineering systems that bridge the cyber-world of communications and computing with the physical world. These systems are usually safety-critical and exhibit both discrete and continuous dynamics that may have complex behavior. Typically, these systems have to satisfy given specifications, i.e., properties that define the valid behavior. One commonly used approach to evaluate the correctness of CPSs is testing. The main aim of testing is to detect if there are situations that may falsify the specifications.

For many industrial applications, it is only possible to simulate the system under test because mathematical models do not exist, thus formal verification is not a viable option. *Falsification* is a strategy that can be used for testing CPSs as long as the system can be simulated and formal specifications exist. Falsification attempts to find *counterexamples*, in the form of input signals and parameters, that violate the specifications of the system. Random search or optimization can be used for the falsification process. In the case of an optimization-based approach, a quantitative semantics is needed to associate a simulation with a measure of the distance to a specification being falsified. This measure is used to guide the search in a direction that is more likely to falsify a specification, if possible.

The measure can be defined in different ways. In this thesis, we evaluate different quantitative semantics that can be used to define this measure. The efficiency of the falsification can be affected by both the quantitative semantics used and the choice of the optimization method. The presented work attempts to improve the efficiency of the falsification process by suggesting to use multiple quantitative semantics, as well as a new optimization method. The use of different quantitative semantics and the new optimization method have been evaluated on standard benchmark problems. We show that the proposed methods improve the efficiency of the falsification process.

Keywords: Cyber-Physical Systems, Testing, Falsification

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". Proceedings of 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki-Espoo, Finland.

[B] **Zahra Ramezani**, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, Knut Åkesson, "Multiple Objective Functions for Falsification of Cyber-Physical Systems". To appear in Proceedings 15th Workshop on Discrete Event Systems (WODES), Rio, Brazil, November 2020.

[C] **Zahra Ramezani**, Koen Claessen, Martin Fabian, Knut Åkesson, "Enhancing Temporal Logic Falsification with Line Optimization". To be submitted to possible journal publication.

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

[D] **Zahra Ramezani**, Jonas Krook, Zhennan Fei, Martin Fabian, Knut Åkesson, "Comparative Case Studies of Reactive Synthesis and Supervisory Control". *Proceedings 18th European Control Conference (ECC),* pp. 1752-1759, Naples, Italy, June. 2019.

[E] 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*, 2020.

[F] Koen Claessen, Nicholas Smallbone, Johan Lidén Eddeland, **Zahra Ramezani**, Sajed Miremadi, Knut Åkesson, "Using Valued Booleans to Find Simpler Counterexamples in Random Testing of Cyber-Physical Systems". *14th IFAC Workshop on Discrete Event Systems (WODES)*, 2018.

[G] Koen Claessen, Nicholas Smallbone, Johan Lidén Eddeland, **Zahra Ramezani**, Knut Åkesson, Sajed Miremadi, "Applying valued booleans in testing of cyber-physical systems". *Proceedings 3rd Workshop on Monitoring and Testing of Cyber-Physical Systems (MT-CPS),* Porto, Portugal, April, 2018.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisor Prof. Knut Åkesson for his unique support and patience 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 interesting period of someone's life. He continuously has been motivating me to feed my curiosity and will to learn, to try, and to investigate deeply various concepts. Secondly, my sincere gratitude is for my second supervisor Prof. Martin Fabian, his presence provides me an incredible opportunity to discuss and interpret different scientific aspects. The fact is that without any hard discussion and knowledge exchange it is impossible to finish a milestone. I always received wise, while challenging comments, that force me to respond creatively.

I also want to thank everyone who I have worked with. I would like to dedicate a special thanks to Prof. Koen Claessen for nice collaboration, discussions, and support. I also would like to thank my colleagues in the SyTec project: Dr. Nicholas Smallbone, Prof. John Hughes, Prof. Mary Sheeran, and Johan Lidén Eddeland. My biggest thanks to Dr. Alexandre Donzé, for his support, help, and guidance. I would like to express my sincere thanks to all my colleagues and friends in Automation groups and also at the Electrical Engineering Department.

Now it is the turn to dedicate my words to a person who gave me love of life. My role model in every moment of life, my Mom. Being robust, resilient, while keeping her warm, kind, and precious smile towards all her children. The words can not express my feelings and sensations for a real angel that has been with me unconditionally. Love you Mom. An incredible honor to my father, the man who has been dedicating his entire life to provide us with the prosperous, peaceful, and independent conditions. I really do not know how to describe and how to be thankful.

Life without friendship is vague and meaningless. My dearest sisters Parisa and Sepideh that their emotional supports provided me togetherness. Since childhood so far, playing together, being together in happiness and sadness, studying together, and many other things together. 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. At last but not least thanks to the new member of my family, Hesam, my brother in law, for his support.

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 simulation results for all benchmark problems in this thesis can be performed on resources at Chalmers Centre for Computational Science and Engineering (C3SE) provided by the Swedish National Infrastructure for Computing (SNIC).

Acronyms

AD:	Autonomous Driving
Additive:	Additive semantics
CPS:	Cyber-Physical System
MARV:	Mean Alternative Robustness Value semantics
Max:	Max semantics
MBD:	Model-Based Development
NM:	Nelder-Mead
SNOBFIT:	Stable Noisy Optimization by Branch and Fit
STL:	Signal Temporal Logic
SUT:	System Under Test
VBool:	Valued Boolean

Contents

Abstract i

List of Papers iii

Acknowledgements v

Acronyms vii

I Overview 1

1 Introduction 3
1.1 Research Questions 6
1.2 Method 8
1.3 Outline 9

2 Falsification of Cyber-Physical Systems 11
2.1 Cyber-Physical Systems 11
2.2 Testing of Cyber-Physical Systems 12
Tools for Testing of CPSs 13
2.3 Falsification of Temporal Logic Specifications 14
Falsification in Breach 14

2.4 An Autonomous Driving Example . . . 16
2.5 Large-Scale Testing . . . 18

3 Quantitative Semantics 19
3.1 Specification Formalisms . . . 19
3.2 Signal Temporal Logic . . . 20
3.3 Valued Booleans . . . 21
Max Semantics . . . 22
Additive Semantics . . . 23
3.4 Mean Alternative Robustness Value Semantics . . . 24
3.5 Discussion . . . 25

4 Optimization Methods 27
4.1 The Optimization Problem . . . 27
4.2 The Nelder-Mead Optimization Method . . . 28
4.3 Stable Noisy Optimization by Branch and Fit . . . 29

5 Contributions and Summary of Papers 33
5.1 Contributions . . . 33
5.2 Summary of Papers . . . 45
5.3 Paper A . . . 45
5.4 Paper B . . . 46
5.5 Paper C . . . 47

6 Concluding Remarks and Future Work 49
6.1 Future Work . . . 51

References 53

II Papers 59

A Evaluating Two Semantics for Falsification using an Autonomous Driving Example A1
1 INTRODUCTION . . . A4
2 Falsification . . . A5
2.1 Signal Temporal Logic . . . A6
2.2 Valued Booleans and *MAX* semantics . . . A6

2.3 Mean Alternative Robustness Value ($MARV$) semantics A7
3 Use-Case . . . A8
3.1 Specification of Safe Longitudinal Distance . . . A9
4 Evaluation Results . . . A10
5 Conclusion . . . A15
References . . . A16

B Multiple Objective Functions for Falsification of Cyber-Physical Systems B1
1 INTRODUCTION . . . B3
1.1 Quantitative Semantics and Objective Functions . . . B6
1.2 Quantitative Semantics . . . B7
2 Falsification Using Multiple Objective Functions . . . B7
3 Benchmark Problems . . . B9
3.1 Automatic Transmission (AT) Benchmark . . . B11
3.2 Third Order Modulator . . . B11
3.3 Static Switched (SS) System . . . B11
4 Experimental Setup and Results . . . B12
4.1 Results . . . B14
5 Conclusion . . . B15
References . . . B16

C Enhancing Temporal Logic Falsification with Line Optimization C1
1 Introduction . . . C3
2 Quantitative Semantics and Objective Functions . . . C7
2.1 Quantitative Semantics . . . C8
3 Optimization Methods . . . C9
3.1 Hybrid Corners-Random Method . . . C10
3.2 Line Optimization . . . C11
4 Benchmark Problems . . . C18
4.1 ARCH Benchmark Problems . . . C18
4.2 Additional Benchmark Problems . . . C21
5 Experimental Setup and Results . . . C22
5.1 Hybrid Corners-Random vs. Corners and Random . . . C29
5.2 Line Optimization vs. Nelder-Mead and SNOBFIT . . . C31
5.3 Comparison of all optimization methods . . . C33
6 Conclusions . . . C35

References C35

Part I

Overview

CHAPTER 1

Introduction

Computational and physical resources are closely interconnected in many real-world systems. These systems are commonly defined as cyber-physical systems (CPSs). CPSs have applications in communications, robotics, healthcare, energy [1].

CPSs are often developed in a model-based development (MBD) paradigm. MBD involves the different kinds of models that constitute the *closed-loop* system: plant and control [2]. The plant model contains the dynamical characteristics of the physical components of the system like mechanical, electrical, chemical components. The dynamical characteristics are represented based on differential and algebraic equations. The control model contains the embedded software components of the system that regulates the behavior of the physical system.

CPSs usually safety-critical and typically are *hybrid systems*, i.e., a combination of discrete and continuous dynamics which may have the complex behavior. These systems must fulfill their expected behavior. To guarantee their correctness two common methods are used: testing, and verification. Testing and verifying the correctness of CPSs are a big challenge. Since it is necessary to do hardware and software testing, communication and com-

putation testing, extra-functional testing for each component [3]. This thesis tackles the problem of testing CPSs.

To test the developed models, what needs to be defined is the expected behavior of the system. The term *specification* is used to describe this behavior. The specification can be expressed in natural language or mathematical language.

One motivating example in this thesis is an adaptive cruise control system for autonomous driving (AD). In this example, the autonomous car represents the plant, the adaptive cruise control is the control method implemented on the autonomous car. There are other cars and pedestrians around the autonomous car that affect the behavior of the autonomous car. Of course, like any car, an autonomous car must always drive safely. So, to avoid collisions, for instance, we can formulate the following specification in natural language: "*the relative distance between the autonomous car and a car in front of it, must always be greater than ten meters*". In mathematical terms, this can be expressed as "$d_{relative} > 10$".

Testing is an essential part in engineering that is widely considered in industry to guarantee the quality of a system under test (SUT). Testing can be done in different parts and component of a CPS application like [3]:

- **Hardware Testing:** This consists of the hardware testing including the testing of the functionality of each component.
- **Structure and Computation Testing:** This focuses on the computational part of the software part of a system.
- **Extra-Functional Properties Testing:** This is related to the inherent interaction with the system environment.
- **Network Testing:** Checks and verifies the used communication protocols.
- **Integration Testing:** This is the combination of the individual software modules in a group and testing the grouped module.
- **System Testing:** Hardware or software testing to test a complete integrated system when all units are integrated to fulfill the overall requirements of the system.

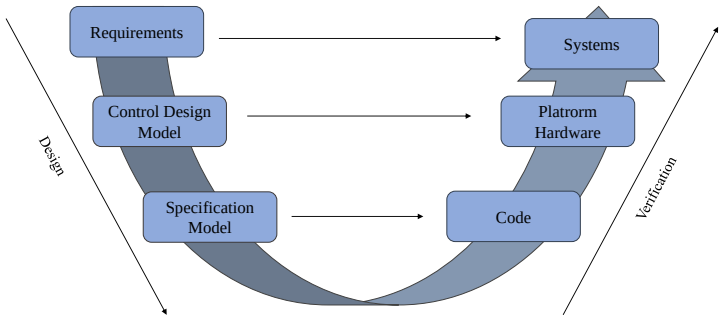

Figure 1.1: The model-based development design [4].

Figure 1.1 represents the MBD process and how different levels of testing can be related to the model. The requirements, or models, on the design part, define the behaviors that should be exhibited by the corresponding systems on the verification part. A control design including the implementation details is created based on the requirements. This control results in a specification model. The code is generated automatically or manually based on the specification model. The code is compiled and executed on the hardware platform.

In early development, *simulation-based* approaches are used for testing [4]. Simulations provide a model of the system that numerically approximates the system behavior. Simulations are used for debugging the 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 systems, such as industrial systems, at any scale. Also, for many industrial applications, analyzing the mathematical models is not possible and it is thus necessary to rely on simulations of the SUT. *Falsification* of temporal logic specifications can be used as a testing method as long as the system can be simulated. Falsification tries

to find *counterexamples* of given specifications. Random search or optimization can be used for the falsification process. The falsification process uses *quantitative semantics* associated with the specifications in the optimization-based approach. A quantitative semantics defines an objective function that can be evaluated for given inputs and gives 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 modifying the test cases such that the new test case is likely to have a lower objective value as defined by the quantitative semantics.

The possible input signals are typically parameterized, for example, a sinusoidal signal can be defined by two parameters, the amplitude, and the frequency. Other examples are constant signals that take different values at different times, this can be parametrized using the values and the times at which the signals change.

The input parameters are chosen within their defined ranges. The objective function, which is 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.

There are limitations for the simulation of large-scale CPSs. Simulations are costly and the simulation time is the most limiting factor, not the evaluation of the quantitative semantics. Thus, it is important to enhance the capability of falsification by trying to reduce the number of needed simulations. This thesis tackles the problem of enhancing the falsification process for CPSs.

1.1 Research Questions

The goal of this thesis can be summarized in three research questions connected to model-based testing of CPSs. These questions are presented below.

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

For the falsification of temporal logic, it is required to estimate the distance to falsify the specification, i.e., quantitative semantics. The efficiency of the falsification is affected by the used quantitative semantics. Different quantitative semantics have been proposed in the literature to guide the test case

generation to falsify the specification. These semantics have different properties and behavior. Thus, this is important to understand the strengths and weaknesses of different semantics.

RQ2. *How does the combination of different semantics affect and improve the falsification of cyber-physical systems?*

The main purpose of calculating an objective function value in terms of quantitative semantics in falsification is to guide the testing process towards the falsification point. This is done by choosing the next set of parameters for the input signals to the system being simulated, such that the likelihood of falsifying the specification is increased, if possible. According to the evaluation results on the falsification of CPSs, it is not possible to have a clear and certain answer to the question of which semantics performs better than others. There is a clear tendency that a specific semantics outperforms for some examples while might not work as well for other examples.

Using a single semantics may cause the optimization to get the wrong or even no information to guide the falsification process for some examples. On the other hand, the used semantics might work well for other examples. Hence, a combination of different objective functions might help to be more efficient and use the advantages of different semantics.

The purpose of this research question is to understand how different semantics can be used to improve the falsification process.

RQ3 *How do different optimization methods affect the falsification process?*

Falsification of temporal logic is a testing method they may formulate the problem of generating test cases as an optimization problem. The optimization generates new input parameters that aim at finding a lower objective value. The used optimization method may thus affect the efficiency of the falsification process.

Different optimization methods are used for falsification in testing toolboxes, and these methods affect the falsification performance. The purpose of this research question is to evaluate the performance of different optimization methods, to know how they work and how they can generate new input points such that the system goes closer to falsifying the specification.

1.2 Method

The main purpose of the falsification methods for CPSs is to find bugs and faults of the system without much additional 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 methods.

To support answering RQ1 we have developed an example of an adaptive cruise controller for an autonomous driving. This example is easy to understand, only having two input parameters, but still having complex dynamics. With this example it is possible to, in detail, analyze how different quantitative semantics affect the falsification performance.

RQ2 was formulated after working on answering RQ1 where it became unclear which quantitative semantics to use, and we wanted to evaluate if it could be beneficial to use a hybrid strategy that uses multiple quantitative semantics during the falsification process. The decision to focus on combinations of different semantics was that it cannot be analytically decided which semantics outperforms the others, as this depends on the specification and the system. The use of multiple quantitative semantics is suggested and evaluated on benchmark problems in Paper B. The benchmark problems are a collection of various problems collected from different publications.

RQ3 is defined since it became clear that the choice of optimization methods affect the falsification process. Since we in this thesis assume that we can only simulate the system and do not have access to the internals of the system, we cannot use a gradient-based optimization method, instead, we have to use a gradient-free optimization method. In Paper C, the benchmark problems came from the ARCH19 workshop [5] plus variants of those problems, and additional problems from Paper B.

Evaluation results on the benchmark problems gave us the feedback that some of them can be falsified by using the corners as input parameters, i.e the extreme input points that are generated from the given ranges of inputs to the SUT, or considering random input points. These results led to suggest a combination of the corners and random method in Paper C as another baseline to compare optimization-based falsification against. Still there are some specifications in the benchmark problems where optimization-based methods is needed. In [6] it is shown that one of the best optimization method, Nelder-Mead (NM) [7], is good at finding a local optima but have no built-in support

for getting out from local optima. While another optimization method is Stable Noisy Optimization by Branch and Fit (SNOBFIT) [8], outperformed NM also when a constant quantitative semantics is used. A constant semantics that has a constant positive value if the specification is fulfilled and a constant negative value if the specification is falsified. In Paper C, a new optimization method, line optimization, is presented that when evaluated on the benchmark problems is as good as or better than SNOBFIT, but having a much simpler implementation.

1.3 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 gives an overview of cyber-physical systems and testing methods for CPSs. This chapter also introduces the testing frameworks including details about Breach [9], the testing tool used in this thesis. The concept of falsification of temporal logic is also presented in this chapter. Chapter 3 introduces the Signal Temporal Logic (STL), the concept of Valued Booleans (VBools), and the three different semantics used in this thesis. In Chapter 4, two optimization methods, the Nelder-Mead method, and SNOBFIT are presented. Chapter 5 presents the contributions in the appended papers. The thesis ends with conclusions and future work.

CHAPTER 2

Falsification of Cyber-Physical Systems

This chapter starts with a presentation of cyber-physical systems in general and continues with testing methods for CPSs. This chapter presents, Breach, the tool that is used for testing in this thesis. We also introduce the falsification of temporal logic specifications, the main method used in this thesis. The chapter ends with large-scale testing where it discusses the benefits of running a large number of simulations on computing clusters.

2.1 Cyber-Physical Systems

The term cyber-physical systems refers to systems that are connected with other systems in the environment by integrating physical components and communication. They have the ability to collect and evaluate data, and communicate with other systems in their environment [10].

CPSs contain both *cyber* and *physical* elements with various sensors and actuators. CPSs typically contain a mix of continuous and discrete dynamics, and are thus *hybrid systems*. Therefore, the methods and tools of CPS engineering incorporate both continuous-valued formalisms of physical systems, like mechanical, electrical, electronic engineering, and discrete models of the

computing hardware and software [11]. CPS applications are found in a wide range of application domains, like factory automation, medical systems, power grids, aerospace, and transportation.

CPSs are often developed using a model-based development (MBD) paradigm [12]. The first step of the MBD process contains the *plant model*. The model of the plant captures the dynamic characteristics of the system's physical parts as described by differential, logic, and algebraic equations. Designing a controller is the next step of MBD. A controller is designed to regulate the behavior of the physical system by obeying control laws. The *closed-loop* model refers to the composition of the plant and the controller.

The integration of the cyber and physical makes CPSs complex and difficult to analyze. CPSs are often safety-critical, for example, autonomous cars and medical devices. Thus, verification and testing of CPSs are two important methods for the validation and correctness of these systems. Testing is the most commonly used method in industry. This thesis considers the problem of testing. In the next section a review of testing methods of CPSs is presented.

2.2 Testing of Cyber-Physical Systems

CPSs are often tested at different levels: Model-in-the-loop (MIL), Software-in-the-loop (SIL), and Hardware-in-the-loop (HIL). Testing functional requirements are done at the MIL and SIL level, while at the HIL level functional as well as non-functional requirements are tested by performing real-time simulations [13]. Testing of CPSs is costly because of: many variants to test; long test cases; simulation of the physical layer. Therefore, selecting suitable test cases is important. CPSs have large and complex sub-systems and multiple variants. Thus, simulations are gaining importance for testing, debugging, and estimation of the performance of CPSs.

For many industrial applications, it is not possible to analyze the mathematical model analytically, it is only possible to simulate the SUT. Thus formal verification is not a viable option. The *falsification* of temporal logic specifications, as a testing method, can be used as long as the model can be simulated and a formal specification exists. Tools that can be used for testing will be introduced in the next section.

Tools for Testing of CPSs

As was mentioned in previous sections, large industrial systems are a mix of continuous models and discrete models, this feature may make them complex. We assume that only a *black-box* model of the SUT is available to analyze. Thus, there is no internal information about the model and the controller of the system, and only the input-output behavior of the SUT can be observed. Different tools and methods are used for testing CPSs: TestWeaver [14], FALSTAR [15], falsify [16], S-TaLiRo [17], and Breach [9].

TestWeaver is a commercial testing software that can automatically generate thousands of test vectors. This is efficient because testing of the system needs to consider a large number of test vectors. The testing inputs are generated to maximize a certain notion of coverage, and the discretization of the continuous state space of the hybrid system [14].

FALSTAR is an experimental falsification tool that explores the idea to construct the inputs incrementally in time. In FALSTAR, different probabilistic algorithms are implemented, like a two-layered framework combining Monte-Carlo tree search [18], and a probabilistic algorithm [15].

falsify is an experimental falsification program that uses reinforcement learning algorithms to solve the falsification problem. falsify implements a deep reinforcement learning algorithm Asynchronous Advantage Actor-Critic [19]. falsify uses a *grey-box* method where it learns the behavior of the system by observing the system output during the simulation [16].

S-TaLiRo is a MATLAB toolbox used to perform falsification using Metric Temporal Logic (MTL). This tool searches for counterexamples to properties for nonlinear hybrid systems using a quantitative semantics metric in Simulink and Stateflow. The optimization uses stochastic optimization techniques that perform a random walk over the initial states, controls, and disturbances of the system [17].

Breach is another MATLAB/Simulink toolbox used for testing and falsification of specifications in Signal Temporal Logic (STL). The falsification process is guided by an optimization procedure. Breach is the tool used in this thesis. In the next section, the falsification method will be introduced and discussed then its implementation in Breach will be discussed.

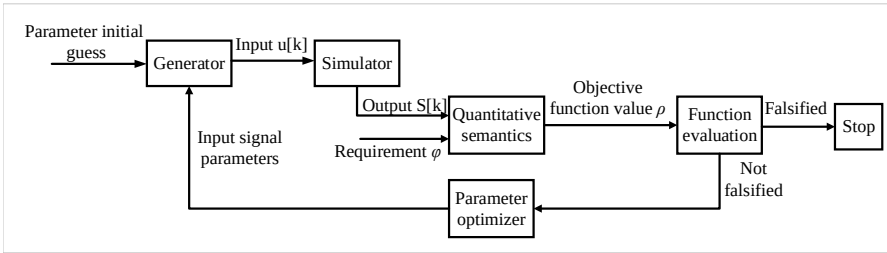

Figure 2.1: A flowchart describing the optimization-based falsification procedure in Breach [20].

2.3 Falsification of Temporal Logic Specifications

The falsification process is a strategy that performs over *quantitative semantics* of the specification in the optimization-based approach. A quantitative semantics defines an objective function to measure the distance to the specification being falsified and guide the process towards the falsification process by choosing the next set of input points. The objective function is determined by the definition of quantitative semantics for the temporal logic formalisms.

The falsification problem is formulated as:

- **Given**: a system *model* $\mathcal{S}$, *input* domain $\mathcal{P}_u$ to the system, and a *specification* φ.
- **Results for falsification**: a *counterexample* input $x \in \mathcal{P}_u$ such that its corresponding *output* $\mathcal{S}(x)$ violates the specification φ, if such input exists, i.e., $\mathcal{S}(x) \not\models \varphi$.

Falsification in Breach

This section reviews how the falsification procedure is implemented in Breach. The main falsification procedure in Breach is shown in Figure 2.1. This procedure works as follows.

Input generators

To generate the input trace $u[k]$ at a time k to SUT, the *Generator* needs to take some input parameters (points). The space of permissible input signals

is parametrized by m input parameters $a = (a_1, \ldots, a_m)$ that take values from a set $\mathcal{P}_u$. The actual input $u[k]$ is created using a generator function g such that $\mathbf{u}[k] = g(v(\mathbf{a}))[k]$, where $v(\mathbf{a}) \in \mathcal{P}_u$ is a valuation of the parameter vector $\mathbf{a}$. The output signal is calculated by system $\mathcal{S}$. The optimization problem is formulated as

$$\min_{(v(\mathbf{a})\in\mathcal{P}_u)} \quad \rho\Big(\varphi, \mathcal{S}(g(v(\mathbf{a})))\Big), \tag{2.1}$$

where ρ is a quantitative semantics that defines the objective function.

The input parameters are chosen within their defined ranges. To interact with the system we can use different parameterized input generators. For instance, the parameters can represent the control point, and the input generated by some interpolation between these points. Also, it is possible to have variable step inputs and also different numbers of control points and interpolation methods, like spline or liner. Piece-wise constant signals take different values at different times, this can be parametrized using the values and the times at which the signals change. Sinusoidal signal is another example that can be defined by two parameters, the amplitude, and the frequency.

Simulation

When the SUT is available in Simulink and the input $u[k]$ generated by the Generator, the *Simulator* can generate a simulation trace. The trace and the requirement φ are used together to evaluate the quantitative semantics, ρ.

Quantitative semantics

The *quantitative semantics* ρ is evaluated to see whether the specification is falsified or not. Quantitative semantics defines an objective function. A negative objective function value means the specification is falsified and a positive value shows it is not falsified. The falsification procedure will stop when ρ is negative. Different quantitative semantics are defined in the literature as *Max* [21], *Additive* [20], Mean Alternative Robustness Value *(MARV)*, Root Mean Square Alternative Robustness Value [22], and averaged STL (avSTL) [23]. *Max*, *Additive*, and *MARV* used in the appended papers of this thesis is introduced in detail in Chapter 3. The performance of the semantics depends on the system and the specification. Papers A and B evaluate the effect of different semantics.

Parameter optimization

If the objective function value ρ is positive, this means that the specification is not falsified, it leads to new parameters being sampled 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 k within given ranges to optimize the objective function ρ to find lower objective function values.

Much research has been devoted to parameter optimization for falsification. In [24], the gradients are used to guide the optimization. On the other hand, the gradient-free optimization approaches for testing include Ant Colony Optimization [25] and Local Stochastic Tabu search [26].

In [6] different optimization methods were evaluated on some CPSs benchmark problems to evaluate the effect of optimization on falsification. They are SNOBFIT [8]; NM [7], an optimization implemented in Breach; Simulated Annealing [27], was included in S-TaLiRo and is now implemented in Breach; Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [28]. The effect of the optimization method is considered in Paper C where a new gradient-free optimization is suggested. NM and SNOBFIT, that are considered in this thesis, are presented in Chapter 4.

2.4 An Autonomous Driving Example

In this section, an example of autonomous driving, AD, is presented to give more details about how the input generators and quantitative semantics work in Breach implementation. Autonomous driving applications aim to take the driving responsibility away from human drivers, thus, safety of these systems is paramount.

In this example, two cars are assumed on a road: *ego* car refers to the *autonomous car* and a *lead car* that is a normal car in front of the ego. The ACC is designed to automate the task of choosing the right acceleration, a_{ego}, for the ego car to maintain a safe distance from other cars.

The lead car can change its acceleration, a_{lead}. The acceleration of the lead car in front of the autonomous car can be assumed as the input to the system in Breach. Before a simulation of the closed-loop system starts, the values of this input must be selected. Then, it is needed to know what are the suitable ranges to consider for the acceleration of a non-autonomous car.

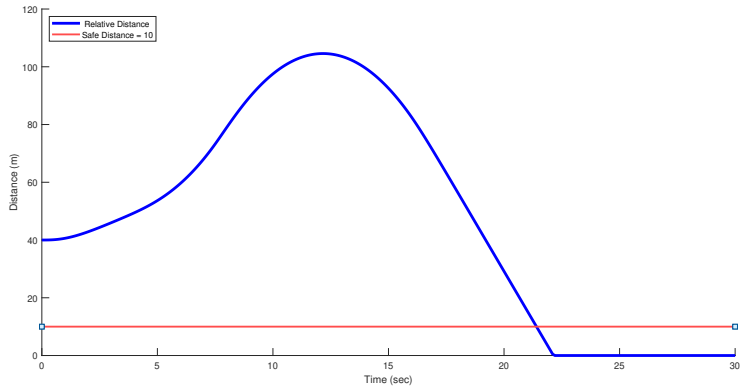

Figure 2.2: The relative and safe distance between two cars.

To avoid collision, the relative distance $d_{relative}$ between the ego and the lead car must always be greater than the safe distance. The safe distance is assuemd equal to a constant value 10. The specification is formulated as:

$$\varphi : \square_{[0,30]}(d_{relative} > 10).$$

Then ρ is defined as to minimize the difference $(d_{relative} - 10)$. Assume that the lead car can accelerate in the range $[0, 3]$ and brake in $[-3, 0]$. The input point is generated from Breach using two control points, a_{lead0} and a_{lead1}. The simulation time is $T = 30$ *sec*. The simulation starts with a_{lead0} chosen by Breach in the range $[0, 3]$. At time 7.5 *sec*, Breach chooses a new value a_{lead1} in the range $[-3, 0]$.

By giving the input points in the given ranges to the system, we would like to see how the ego car can react and the safety specification formula can be satisfied or not. Also, in which input point the specification is falsified. The relative and safe distance between the two cars is shown in Figure. 2.2, where $a_{lead0} = 1.5$ and $a_{lead0} = -1.5$. This figure is generated when the lead car accelerates and after some time decelerates, and results in the safety formula being violated at time 22 *sec*, where $d_{relative} < 10$ and a crash happens.

As was shown by the AD example, falsification finds counterexamples where a system does not satisfy the given specification. It can help to find bugs and faults on the systems.

2.5 Large-Scale Testing

A key strength of using a model-based development methods is that the models can be used for various purposes including testing.

When both the control logic, the physical parts, and the environment have models that support high-fidelity simulation, the closed-loop behavior can be evaluated.

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 the Chalmers Centre for Computational Science and Engineering (C3SE) provided by the Swedish National Infrastructure for Computing (SNIC)[1].

[1] https://www.c3se.chalmers.se/

CHAPTER 3

Quantitative Semantics

In this chapter, the specification formalisms are reviewed and the concept of Signal Temporal Logic is defined with details. The concept of Valued Booleans (VBools) and different semantics are presented, in this chapter.

3.1 Specification Formalisms

To test CPSs, the expected behavior of the system, i.e., the desired properties of the system must satisfy, needs to be defined, and expressed in a mathematically precise way. The term *specification* is used to describe the desired behavior.

There are different mathematical formalisms to model the specifications like *Metric Temporal Logic* (MTL), Metric Interval Temporal Logic (MITL) [29], and *Signal Temporal Logic* (STL) [30]. STL is an extension of MTL with real-valued signals and dense time, while MTL is an extension of Linear Temporal Logic (LTL) [31].

STL, that is is used to model the specifications in this thesis, is introduced below.

3.2 Signal Temporal Logic

The syntax of STL is defined below:

$$\varphi ::= \mu \,|\, \neg\mu \,|\, \varphi \wedge \psi \,|\, \varphi \vee \psi \,|\, \square_{[a,b]}\psi \,|\, \Diamond_{[a,b]}\psi \,|\, \varphi\, \mathcal{U}_{[a,b]}\psi,$$

where the predicate μ is $\mu \equiv \mu(s) > 0$ and s is a signal; φ and ψ are STL formulas; $\square_{[a,b]}$ denotes the *globally* operator between times a and b (with $a < b$); $\Diamond_{[a,b]}$ denotes the *finally* operator between a and b; and $\mathcal{U}_{[a,b]}$ denotes the *until* operator between a and b.

The satisfaction of the formula φ for the discrete signal s at the discrete-time instant k is defined as [32]:

$$\begin{aligned}
(s,k) &\models \mu && \Leftrightarrow\ \mu(s[k]) \\
(s,k) &\models \neg\mu && \Leftrightarrow\ \neg((s,k) \models \mu) \\
(s,k) &\models \varphi \wedge \psi && \Leftrightarrow\ (s,k) \models \varphi \wedge (s,k) \models \psi \\
(s,k) &\models \varphi \vee \psi && \Leftrightarrow\ (s,k) \models \varphi \vee (s,k) \models \psi \\
(s,k) &\models \square_{[a,b]}\varphi && \Leftrightarrow\ \forall k' \in [k+a, k+b], (s,k') \models \varphi \\
(s,k) &\models \Diamond_{[a,b]}\varphi && \Leftrightarrow\ \exists k' \in [k+a, k+b], (s,k') \models \varphi \\
(s,k) &\models \varphi\, \mathcal{U}_{[a,b]}\psi && \Leftrightarrow\ \exists k' \in [k+a, k+b]\ \ (s,k') \models \psi \\
& && \quad \wedge \forall k'' \in [k, k'], (x,k'') \models \varphi
\end{aligned}$$

A modified version of STL, called averaged STL (avSTL) [23] where two time-averaged temporal operators ($\square$ and $\Diamond$) are introduced and have a different robustness value than the standard robust semantics. STL* is a logic formalism that includes an additional freezing operator that specifies damped oscillations in a signal [33]. Time-Frequency Logic (TFL) [34] is a specification formalism for real-valued signals that combines temporal logic properties in the time domain with frequency-domain properties.

Instead of only checking the boolean satisfaction of an STL formula, the notion of a quantitative value, i.e., an objective value also known as the robustness value, will be defined to measure how far away a specification is from being falsified. Therefore, the concept of VBools first will be defined in the next section.

3.3 Valued Booleans

A VBool [20] (v, x) is a combination of a Boolean value v together with a real number x that is a measure of how true or false the specification is. This value will be used as a measure of how convincingly a test passed, or how severely it failed, respectively. VBools is a logical framework that allows choosing between several possible semantics for each connective. In the VBool definition, the value is defined to always be non-negative, possibly infinitely large

$$\mathbb{V} = \mathbb{B} \times \mathbb{R}_{\geq 0}$$

The VBool comparison operator is defined as

$$\leq_v : \mathbb{R} \times \mathbb{R} \to \mathbb{V}$$
$$x \leq_v y = \begin{cases} (\top, y - x) & \text{if } \ x \leq y \\ (\bot, x - y) & \text{otherwise,} \end{cases}$$

where $\top$ and $\bot$ denote true and false, respectively.

The other comparison operators are defined in terms of $\leq_v$, except for $=_v$ which is defined as

$$x =_v y = \begin{cases} (\top, K) & \text{if } \ x = y \\ (\bot, K) & \text{otherwise,} \end{cases}$$

where K is an arbitrary constant. Truth values and negation are defined as

$$\top_{\mathrm{v}} = (\top, \infty),$$
$$\bot_{\mathrm{v}} = (\bot, \infty),$$
$$\neg_v(b, x) = (\neg b, x).$$

The semantics of VBools is in terms of discrete-time signals. VBool includes two quantitative semantics: *Max* and *Additive*. The purpose of using *Max* semantics for VBools instead of STL quantitative is so that the tester can freely change between different semantics of VBools, even within a single

formula. We present these two quantitative semantics below.

Max Semantics

The operators of Max-*and* ($\wedge_{\text{Max}}$) , Max-*or* ($\vee_{\text{Max}}$), Max-*always* ($\Box_{\text{Max,[a,b]}}$), and Max-*eventually* ($\Diamond_{Max,[a,b]}$) will be introduced here.

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

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

The first clause expresses the case where both VBools are true and they are combined with a conjunction. In this case, the value of the VBools is determined by which of x or y has the lowest value. The second clause expresses a conjunction of two VBools where one is true and one is false. This conjunction is false and the value is given by the false VBool, x. Similarly in the third clause, the value of the false VBool, y is considered. The fourth clause expresses a case where both VBools are false. This conjunction is false and the value of the VBools is determined by whichever of x or y has the highest value.

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

$$(b_x, x) \vee_{\text{Max}} (b_y, y) = \neg_{\text{v}}(\neg_{\text{v}}(b_x, x) \wedge_{\text{Max}} \neg_{\text{v}}(b_y, y)).$$

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

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

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:

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

Finally, the *Max until*-operator as:

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

.

Additive Semantics

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

The Additive-*and* operator is defined as

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

The first clause expresses the case where both VBools are true and they are combined with a conjunction. In this case the value of the VBools is determined as $\frac{1}{\frac{1}{x}+\frac{1}{y}}$. This formula is inspired by the formula for parallel resistance that gives a value that is less than the maximum of x and y. The second clause expresses a conjunction of two VBools where one is true and one is false. This conjunction is false and the value is given by the false VBool, x. Similarly in the third clause, the value of the false VBool, y is considered. The fourth clause expresses a case where both VBools are false. This conjunction is false and the value of the VBools is defined as $x + y$. In this formula, we consider the values of both x and y, not just whichever of them has the highest value.

The Additive-*or* operator is defined as:

$$(b_x, x) \vee_{\text{Additive}} (b_y, y) = \neg_{\text{v}}(\neg_{\text{v}}(b_x, x) \wedge_{\text{Additive}} \neg_{\text{v}}(b_y, y)).$$

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

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

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:

$$(\bot, x)\,\#'\,\delta t = (\bot, x \cdot \delta t)$$
$$(\top, x)\,\#'\,\delta t = (\top, x/\delta t).$$

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

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

The *Additive until*-operator is defined as

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

The *implication* is defined slightly differently than in classical logic:

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

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

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

VBool includes two semantics that was introduced above. There is semantics that is not expressed in terms of VBool. One of these semantics is *MARV*, which is used in Paper A, will be introduced in the next section.

3.4 Mean Alternative Robustness Value Semantics

MARV, as defined in [22], is a semantics where only the timed always operator is defined. The discrete-time always operator is described by the following

formula.

$$\square_{MARV,[a,b]}\varphi = \begin{cases} \frac{1}{b-a}\sum_{i=0}^{M-1}\rho(\varphi,t_i)\,(t_{i+1}-t_i) & \rho \geq 0 \\ \rho(\varphi,t_i) & \rho < 0 \end{cases} \tag{3.5}$$

where $\rho(\varphi, t_i)$ is a real-valued function that gives the objective value at each time instant, t_i, and M is the number of sampling times over the interval [a, b].

For positive objective values, which represent the case when $\square_{MARV,[a,b]}\varphi$ is satisfied, *MARV* calculates the mean over the interval.

This semantics has different properties compared to *Max* and *Additive* that will be discussed in the next section.

3.5 Discussion

Each semantics has specific features and behaviors compared to others. The *and*-operator is the essential operator in VBool since the *or*-operator, *always*-operator and *eventually*-operator, can be expressed in terms of the *and*-operator for both *Max* and *Additive*, assuming negation is defined. There is no difference between the *Max* and *Additive* if one of VBools is false, in clauses 2 and 3 of equations 3.1 and 3.3. The differences appear in the first and fourth clauses.

In the *and*-operator of *Max*, if $x \wedge y$ is true only the semantics is sensitive to whichever of x or y has the lowest value. On the other hand, *Additive* is sensitive to both values of x and y, if $x \wedge y$ is true. If $x \wedge y$ is false in the fourth clause *and*-operator of both semantics, the *Max* is sensitive to whichever of x or y has the highest value. Contrary to *Max*, in the *Additive*, the semantics is sensitive to values of both x and y.

For the *always*-operator of *MARV*, it can be seen that for a positive ρ in equation 3.5, all signal values affect the value of *MARV*. It is similar to *Additive*, but opposite to *Max*.

Compared to *Additive*, *MARV* may result in larger objective values that lead to having bigger differences between objective function values of evaluated input points. This feature of *MARV* will be discussed further in Paper A and Chapter 5.

CHAPTER 4

Optimization Methods

Two main optimization methods are considered, in this thesis, are Nelder-Mead (NM) [7] and SNOBFIT [8]. This section introduces and compares these two optimization methods.

4.1 The Optimization Problem

The optimization problem is as follows.

$$\begin{aligned} &\min\ f(x) \\ &s.t.\ x \in [u, v], \end{aligned} \tag{4.1}$$

where $f(x)$ is the objective value of the used quantitative semantics; all input parameters are in an interval,

$$[u, v] := \{x \in R^n \mid u_i \leq x_i \leq v_i,\ i = 1, \ldots, n\},$$

where $u, v \in R^n$ and $u_i < v_i$ for $i = 1, \ldots, n$. $[u, v]$ is bounded with nonempty interior, n is the number of dimensions in the optimization problem.

For falsification, the optimization aims to get a falsified point with a negative objective function value, i.e., $f < 0$, if possible. All input points are in a box with a lower and upper value. The SUT is simulated at each given input point and the corresponding objective function values will be calculated. The specification is evaluated according to the sign of the objective function value $f(x)$. $f(x) < 0$ means the specification is falsified; Otherwise, it means the specification is not falsified.

4.2 The Nelder-Mead Optimization Method

NM is an optimization method that is easy to implement and can deal with functions with many variables. Moreover, NM needs only function evaluations that lead to it being suitable in the minimization of non-differentiable functions. This method uses function values at $n+1$ vertices in n dimensions. NM belongs to the general class of direct search methods [35], [36]. The NM algorithm is briefly presented in Algorithm 1.

NM is a simplex-based direct search method that begins with a set of $n+1$ points denote $x_1, \ldots, x_{n+1} \in R_n$. This method aims to decrease the function values at its vertices. The SUT needs to be simulated at each of $n+1$ points and calculate the objective function values, $f(x_1), \ldots, f(x_{n+1})$ using a quantitative semantics. If the specification is falsified at any of these points, the algorithm terminates with the falsified point. Otherwise, these points are ordered and labeled from the lowest to the highest objective function value.

In each iteration of this algorithm, one or more test points are evaluated with their objective function values and the worst point $n+1$ i.e., the point that has the highest objective function value, will be replaced with a new point. A new point is calculated according to which cases of *reflection*, *expansion*, or *contraction* happen, steps 11-33 of Algorithm 1. If *shrink* happens, n new points are generated in step 34 of algorithm 1.

This method will generate $n+1$ new points if one of the following conditions is fulfilled.

- The maximum number of iterations, *Max_Iter* is reached. The default is $200 \times n$.
- The maximum number of the cost function evaluation, *Max_Fun_Iter*, is reached. It is a constant value.

- $f(x_{n+1}) - f(x_1) < \epsilon$, where $\epsilon > 0$ is a small predetermined tolerance.

This algorithm works until the specification is falsified or the maximum number of the simulations, *max_nbr_simulations* is reached in the falsification process.

4.3 Stable Noisy Optimization by Branch and Fit

SNOBFIT is a MATLAB package designed for selecting continuous parameters for simulations or experiments. SNOBFIT solves the optimization problem as defined in eq. 4.1 and evaluates f at or near a given number of points. This method returns NaN to signal that a requested function value does not exist.

SNOBFIT builds internally, around each point, local models of the function to minimize and return, in each step, several points whose evaluation is expected to give better function values. For the global search, SNOBFIT searches simultaneously in several promising sub-regions. It also predicts their quality by producing cheaper models based on fitting the function values at safeguarded nearest neighbors.

In this method, an initial call with an empty list of points and function values is made. The same number of points is generated by each call to SNOBFIT, and the function is evaluated at the suggested points.

In one call to SNOBFIT, a set of points, X, and corresponding function values are considered. The algorithm then generates the requested number, n_{req}, of new evaluation points, if possible. These points and their function values should preferably be used as input for SNOBFIT. A box $[u', v'] \in R^n$ denotes the interval that the points generated by SNOBFIT are in.

The suggested evaluation points belong to five classes. These classes indicate how a point has been generated by a local or a global aspect of the algorithm. The points of classes 1 to 3 represent the local aspect of the algorithm. The points belonging to these classes are generated with the aid of local linear or quadratic models at positions where good function values are expected. The points of classes 4 and 5 represent more global aspects of the optimization, where they are generated in large unexplored regions. These points are reported together with a model function value computed from the appropriate local model.

SNOBFIT can be done in steps that are briefly described below. It should

Algorithm 1 Nelder-Mead Algorithm for Falsification

n+1 points. Consider $n+1$ points, x_i, denote the list of vertices in the current simplex, $i = 1, \ldots, n+1$.
while (nbr_of_simulations $\leq$ max_nbr_simulations) **do**
  if $f(x_{n+1}) - f(x_1) < \epsilon$ OR *the number of iterations > Max_Iter* OR *the number of function evaluation > Max_Fun_Evals* **then**
    Consider $n+1$ new points.
  end if
  Simulate the system at each $n+1$ and calculate their objective function values using the used semantics.
  if the spec is falsified at the current evaluated point **then**
    Terminates the algorithm with the falsified point.
  else
    Order. Order and re-label the $n+1$ points from lowest objective function value to highest like: $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$.
    Reflection: Compute the reflected point $x_r = \bar{x} + \rho(\bar{x} - x_{(n+1)})$, where $\bar{x}$ is the centroid of the n points with lowest objective function values, $\bar{x} = \sum \frac{x_i}{n},\ i = 1, \ldots, n$.
    if $f(x_1) < f(x_r) < f(x_n)$ **then**
      Replace: x_{n+1} with the point x_r.
    end if
    Expansion:
    if $f(x_r) < f(x_1)$ **then**
      Compute the expanded point x_e by $x_e = \bar{x} + \chi(x_r - \bar{x})$.
      if $f(x_e) < f(x_1)$ **then**
        Replace x_{n+1} with x_e.
      else
        Replace x_{n+1} with x_r.
      end if
    end if
    Contraction:
    if $f(x_r) \geq f(x_n)$ **then**
      Perform a contraction between $\bar{x}$ and the best among x_{n+1} and x_r.
      if $f(x_n) \leq f(x_r) < f(x_{n+1})$ **then**
        Calculate $x_{oc} = \bar{x} + \tau(x_r - \bar{x})$ *Outside contract.*
        if $f(x_{oc}) \leq f(x_r)$ **then**
          Replace x_{n+1} with x_{oc}.
        end if
      end if
    end if
    Shrink: Evaluate the n new vertices $x' = x_1 + \phi(x_i - x_1),\ i = 2, \ldots, n+1$.
    Replace the vertices $x_2, ..., x_{n+1}$ with the new vertices $x'_2, ..., x'_{n+1}$.
  end if
end while

be mentioned here that SNOBFIT is a complex method where five different classes are not described with any more details here.

1. **Searching box.** This method needs to start with a search box $[u, v]$. The vectors u and v are defined such that $[u, v]$ is the smallest box.

2. **Duplication**. Duplicates the *new* and in a continuation call also the *old* points, if we are not in the first iteration, from the previous iterations and the corresponding function value f and uncertainty Δf.

3. **Splitting all current boxes**. All current boxes containing more than one point are split according to the algorithm described in section 5 of [8].

4. For the current point set X:
   if $(|X| < n + \Delta n + 1)$ || $(f \neq NaN)$
   Go to step 11.
   else
   For each new point x, a vector pointing to $n + \Delta n$ safeguarded nearest neighbors is computed.
   end if

5. **Fictitious function f and uncertainties Δf.** For any new input point x with function value NaN, i.e., a requested function value does not exist, and for all old points marked infeasible whose nearest neighbors have changed, fictitious values f and Δf are calculated.

6. **Local fits.** Local fits around all new points and all old points with changed nearest neighbors are computed. Also, the potential points of class 2 and 3, and their estimated function values are determined.

7. **Best point and function value.** The current best point x_{best} is in $[u', v']$ and f_{best} refers to the current best function value. A local quadratic fit around the x_{best} is computed as described in Section 3 of [8] and a point belongs to class 1 is generated.
   if (the objective function has not been evaluated in $[u', v']$)
   Go to step 8.
   end if

8. **Generating points.** Generate the remaining evaluation points.

9. **Ordering.**
   if (X contains any local points)
   Take some points that are chosen in the order of ascending model function values.
   else
   For each new point x a vector pointing to $n + \Delta n$ safeguarded nearest neighbors is computed.
   end if

10. **Assigning the model function values.** The function values obtained from the local models are assigned to the points of class 4 (See step 10 in [8] for further details.)

11. **if** (The number of suggested evaluation points is still less than n_{req})
    The set of evaluation points is filled up with points of class 5.
    end if

    if (Local models are already available)
    The model function values for the points of class 5 are determined as the ones for the points of class 4.
    else
    They are set to NaN.
    end if

12. **Stop criteria**. The search continues as long as an outside agent, i.e., a program or human that oversees the optimization, finds it reasonable to continue.

In the evaluation of this optimization method for falsification, the algorithm works until the maximum number of simulations, *max_nbr_simulations* is reached, or the specification is falsified.

CHAPTER 5

Contributions and Summary of Papers

This chapter aims to give details about how the idea of the multiple objective functions was created and how the combination of the different semantics can work. Also, this chapter presents more details about the new optimization methods. The chapter ends with a summary of the papers.

5.1 Contributions

The purpose of falsification is to find *counterexamples* where a given specification is not satisfied. Falsification needs to estimate the distance to the falsification of the specification. Different measures in form of quantitative semantics have been proposed in the literature to improve the falsification process. It is not analytically possible to decide which semantics is the best because it depends on the SUT and the specification. Thus, it is needed to evaluate and analyze the different quantitative semantics on examples to understand their strengths and weaknesses.

The evaluation of large-scale systems is hard because of having many parameters. Simple systems are useful for gaining insight to evaluate and better understand the performance of different approaches. To evaluate different se-

mantics, a simple example of an adaptive cruise controller (ACC) from an autonomous driving example is considered in Paper A. In this example, there are two cars, the *ego* car refers to an autonomous car, and the *lead* car is in front of this car. The ACC is designed with two modes; *speed* mode, where there is a 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.

$$\square_{[0,T]}(d_{rel} > d_{safe}), \tag{5.1}$$

where (d_{safe}) is taken from paper [37]. In the evaluation of the results, the optimization method is not considered. The objective function values of the different quantitative semantics are calculated in the next section.

It should be mentioned here that the aim of the evaluation result in the AD example is not to evaluate the system and find the counterexamples where the safety formula of equation 5.1 is violated but to understand how objective values change for different parts of the parameter space and for different quantitative semantics.

Evaluation of Different Semantics on the AD Example

The model takes the acceleration of the lead car a_{lead} as an input and simulates the AD example. Before a simulation of the closed-loop system starts, the values of input parameters are selected by the falsification algorithm. The input is generated from Breach using two control points, a_{lead0} and a_{lead1}. The simulation time is $T = 30$ *sec* and the simulation starts with a_{lead0} chosen by Breach in the range $[0; 3]$. At time 7.5 *sec*, the acceleration changes and takes a value a_{lead1} in the range $[-3; 0]$. The objective value will be calculated for different combinations of the parameters a_{lead0} and a_{lead1} where these two acceleration points are divided into 20 equidistant points, resulting in 400 simulations.

The objective function values and the gradient vector are calculated for each of the 400 simulations using *Max*, *MARV*, and *Additive* are shown in Fig. 5.1-Fig. 5.6.

As can be seen in Figure 5.1, in the upper right corner (where $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$ approximately), the objective function values are the

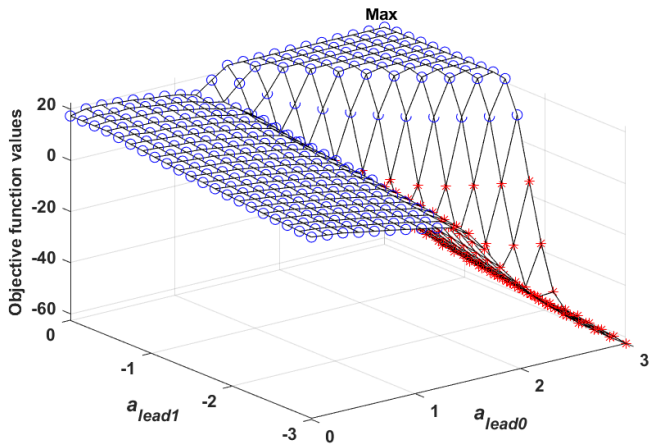

Figure 5.1: The objective values for combinations of the accelerations (a_{lead0}, a_{lead1}) for the *Max* semantics. Positive values ($\circ$) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified.

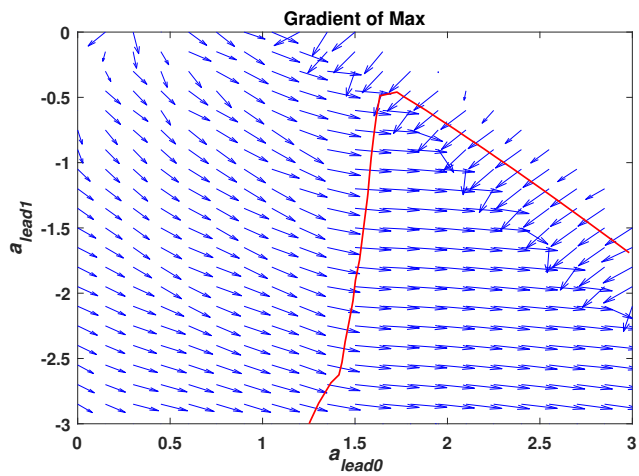

Figure 5.2: The gradient direction using *Max*. The red curve shows the edge of negative objective function values, i.e., falsification area.

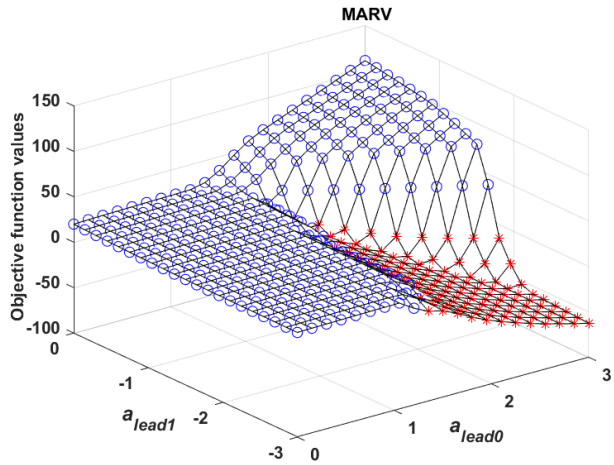

Figure 5.3: The objective values for combinations of the accelerations (a_{lead0}, a_{lead1}) for the *MARV* semantics. Positive values ($\circ$) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified.

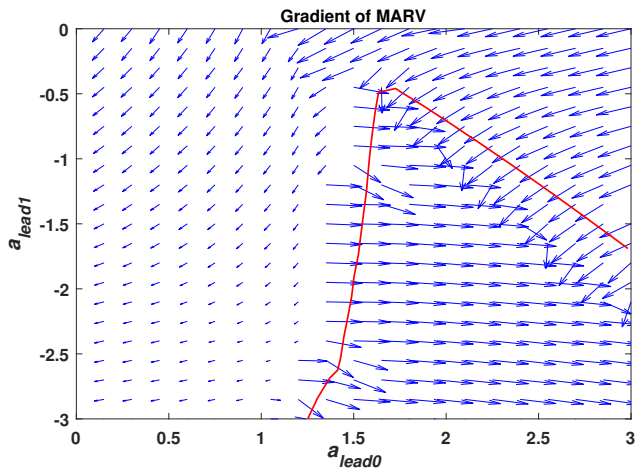

Figure 5.4: The gradient direction using *MARV*. The red curve shows the edge of negative objective function values, i.e., falsification area.

same for each point using the *Max* semantics. This condition happens because when the lead car continuously accelerates, the ego car also increases its speed. But the ego car has a driver-set velocity limit 30 m/s so that the relative distance between the vehicles increases and the minimum objective function value occurs at the beginning of the simulation $t = 0$ where the relative and safe distances are closest to each other. For this reason, as it can be seen in Figure 5.2, for the points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1=0} < 0$, there is no useful gradient, or sense of direction, to go in in order to get closer to a falsification point. On the other hand, for other points except points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1=0} < 0$, there are different objective values. Therefore, those points can guide towards a falsification point, i.e., the red points, where the specification is falsified.

As a result, *Max* semantics considers the different 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 information that can be used by the optimization algorithm and there is not any direction for these points.

On the other hand, by looking at Figure 5.3, we can see that the points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$ have different values under *MARV* semantics. Also, we can see that in figure 5.4, the gradient for the points close to $2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$ is decreasing. But here the situation is opposite for *Max*. For the points close to $0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$ the gradients do not point towards the falsification area.

The objective function values using *Additive* and its gradients are shown in Figures 5.5 and 5.6. One important note that should be mentioned here, although it seems that the gradient arrows do not exist in areas ($0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$) and ($2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$), it is not the case. Because of very small difference between the objective function values in these areas, the arrows, representing the gradients, do not appear in Figure 5.6. For this reason and for a better perspective, the magnitude gradient plots for these areas are shown in Figure 5.7. As can be seen, 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 $-3 < a_{lead1} < 0$.

The behavior of *Max* and *MARV* are discussed in Paper A of this thesis, but the discussion about the Additive semantics was new here.

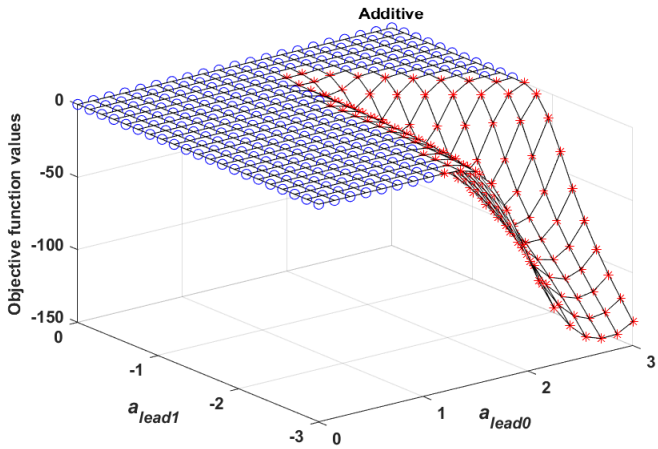

Figure 5.5: The objective values for combinations of the accelerations (a_{lead0}, a_{lead1}) for the *Additive* semantics. Positive values ($\circ$) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified.

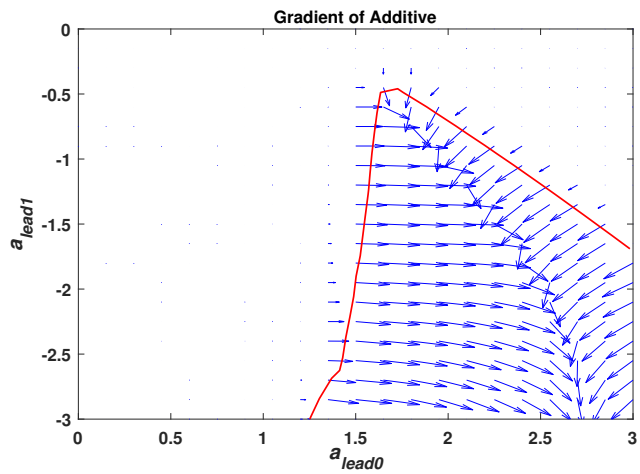

Figure 5.6: The gradient direction using *Additive*. The red curve shows the edge of negative objective function values, i.e., falsification area.

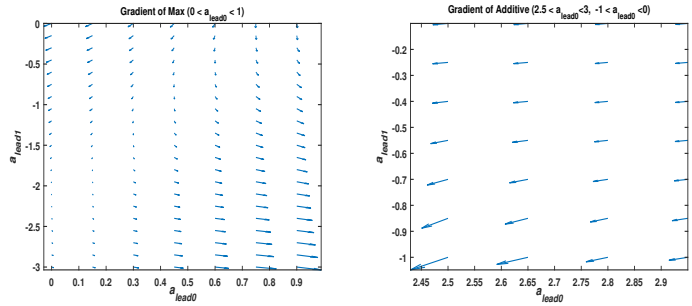

Figure 5.7: The gradient direction using *Additive*, special part of Fig. 5.6. In the left graph, the gradient is shown where $0 < a_{lead0} < 1$ and in the right graph, $2.5 < a_{lead0} < 3$.

Discussion About Using an Optimization Method for the AD Example

The results presented for the AD example in the previous section were generated without using any optimization method. But, it is important to evaluate each semantics when an optimization method is used. The Nelder-Mead optimization method, used in Breach, was introduced in Chapter 4. This method needs $n+1$, where n is the number of dimensions in the optimization problem, points to start the optimization.

Let's assume three initial points for the NM algorithm to be chosen in different areas of autonomous acceleration in figures 5.1, 5.3, and 5.5 for each semantics. Before analyzing, it should be mentioned that NM is a gradient-free optimization method and the aim of using gradients here is to show the objective values of each semantics change in the falsification method.

- The first area is where $0 < a_{lead0} < 1$ and $-3 < a_{lead1} < 0$. If *Max* is used, NM can towards the falsification area. Contrary, using *MARV*, it cannot find the right direction and it searches in the wrong direction where the specification is not falsified. Using *Additive* may need more simulations because there are two different directions and depending on how the new points are chosen by the NM method, it might go to the falsification area or not.

- The second area is where ($2.5 < a_{lead0} < 3$ and $-1 < a_{lead1} < 0$). Because there is no information about the direction in these areas using

Max, therefore NM gets stuck after some iterations, NM stops working without finding the falsification point. On the other hand, both *Additive* and *MARV* can continue towards a falsification of the specification.

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

Multiple Objective Functions

As was discussed in the previous section, *Max* semantics in a specific situation gave the same constant values that result in no information that can be used by the optimization method. This behavior causes, when NM optimization is used, that we get stuck in the area where no direction information is available or a local minimum point is found. This causes the next point to be generated by the NM algorithm to be close to the previous point. As a result, by multiple objective functions, we try to force the algorithm to choose the semantics that has a clear direction. The NM was explained in Algorithm 1 in Chapter 4. This algorithm needs to be modified to use multiple objective functions.

With the purpose of not choosing the semantics that does not have the suitable information for the optimization, different strategies for switching between objective functions can be used as described below.

$$Strategy\ 1 : \frac{\sum_{i=1}^{n+1} (f_i^k - \mu)^2}{n+1}, \tag{5.2}$$

$$Strategy\ 2 : \frac{f_{max}^k - f_{min}^k}{||x_{max}^k - x_{min}^k||}, \tag{5.3}$$

where f_i^k refers to the objective function value of each $n+1$ evaluated points that have the lowest value for each semantics (k). μ is the mean of the

$n+1$ points; f^k_{max} and f^k_{min} refer to the maximum and minimum objective function values of the $n+1$ points that have the lowest objective value for each semantics, x^k_{max} and x^k_{min} refer to these points.

The first strategy calculates the variance of the $n+1$ points. The second one calculates the distance between the points that have the largest and lowest objective function values. For each of these strategies, Eq. 5.2 - Eq. 5.3, the NM algorithm will choose a semantics that has the largest value according to the strategy used.

Properties of Strategies

Each strategy has different features. One common feature of the strategies is that if the objective values of one semantics for all available $n+1$ points, needed for the NM algorithm, are equal then the numerator of all strategies is equal to zero. The semantics that has this feature, will not be chosen by the NM algorithm.

In the first strategy where the variance of points is considered, the effect of all $n+1$ evaluated points is considered. In the second strategy only the effect of the points with the largest and lowest objective function values, in terms of the distance, is considered. If a point has the lowest objective value for a semantics, it does not mean that it has to be the minimum point for other semantics. Thus, by calculating the distance between these two best (largest objective value) and worst (lowest objective value), the semantics that has the larger distance can be chosen. In this strategy, it will avoid picking the semantics that its best and worst points close together. It might help to pick the right semantics to be able to be successful in the falsification.

One weakness of the multiple objective functions approach is if each used semantics is not successful in falsification, we cannot expect that multiple objective functions can find the falsified point.

Modified Nelder-Mead Method

This section discusses how multiple objective functions are implemented as an extension to the NM. It does not matter which strategy that is used, they can all be used by the extension to the NM algorithm that is presented in Paper A.

In the implementation of NM in Breach, two bunch of points are being

considered. First, before the algorithm starts to search for a new point, p random sample points are generated. The second points are $n+1$ points that NM needs to start its process.

When one point of p is generated, the SUT is simulated at the generated point. The objective function value of the evaluated point is calculated. If the given specification is violated, the algorithm terminates with violated input point. Otherwise, a new point is generated and the same evaluation will be done. It will continue until the p points are generated and evaluated. Since that objective value might be different for each used quantitative semantics, there will be one ordering for each semantics. This first step aims to find a minimum point that will be used to generate the $n+1$ points needed for the NM process. All the p random points are sorted from the lowest to highest according to the objective values for each semantics. Here is the first time that a strategy is needed to be used. The semantics that has larger values of used strategy, wins the competition. In the rest of the method, the objective value of the winner semantics will be considered for the NM process. The minimum point of winning semantics is taken here to generate n new points before starting the optimization method. After generating these points, NM starts searching to find a new point (or new points if *Shrink* happens).

Four different cases can happen in the NM process: Reflection, Expansion, Contraction, or Shrink. The new point (or n new points if *Shrink* happens) can be a good point (with lower objective value) or a bad point (with higher objective value) for each semantics. Hence, we save all the new and old points in each iteration. Again, all saved points must be sorted from lowest to highest according to the objective values for each semantics. Since NM needs $n+1$ in each iteration, we take the $n+1$ points for each semantics that have the lowest objective values. Now its time to reuse again a strategy to pick a semantics. This process continues until the stop condition is reached.

It is hard to compare the strategies in equations 5.2 and 5.2 and give a clear answer to the question that which of the suggested strategies performs better than others. Because it depends on the SUT and the specifications. According to the results of Paper B, it can just be said that for the benchmark set when using multiple objective functions and NM optimization, using multiple objective functions works better than using a single objective function.

Line Optimization Method

The optimization method may affect falsification performance. Evaluation results on benchmark problems in Paper C showed that some specifications and examples can be falsified using the corners points or by using only random points. It means that, for these types of problems, it is not necessary to use any optimization method at all. A combination of corners and random methods can be used. This method can work for falsification of the specifications that at least one of the corners or random method performs well. On the other hand, there are specifications and examples where neither corners, random, and even their combination is successful in falsifying the specifications. Thus, in these situations using optimization methods might be an interesting option.

There are different optimization methods in the literature used for falsification. Two main optimization methods are considered in Paper C are NM and SNOBFIT.

NM is an optimization for finding at local optima, while SNOBFIT is a complex method that can do both local and global search. A simple optimization, line optimization, the method will be introduced here.

To start the process of the line optimization method, three initial input points are needed. These three points are taken from a line, in n dimensions, that passes from a middle point and cut off at the edges of the upper and lower of input ranges. The first point is called middle point $x^* = \frac{l_b+u_b}{2}$, where l_b and u_b are the vectors that refer to the minimum and maximum of the input ranges, respectively. This optimization method aims to modify this point in each iteration, i.e., find a better point that has a lower objective value than x^*. In falsification, it means, we try to find the new points that a point falsifies the specification. Two other points are generated from the line that passes from x^* and cut off at the edges of the upper and lower of input ranges. These two points are called x_L and x_R.

According to the objective value calculated of the used semantics by simulating the system at each three points, $f(x^*)$, f_L, f_R, these points can have different cases. Irrespectively, of which case that occurs, in each iteration of the optimization process, we work with one of the new points $x_1 = \frac{x_L+x^*}{2}$ or $x_2 = \frac{x_R+x^*}{2}$. Then, one of the new points, x_1 with the objective function value $f(x_1)$ or x_2 with the objective function value $f(x_2)$, can be used instead of one of old points according to the following rules.

1. If f_L is the smallest objective value, then

Simulate the system at point $x_1 = \frac{x_L + x^*}{2}$ and calculate $f(x_1)$. let (x_L, x^*, x_R) be (x_L, x_1, x^*).

2. If f_R is the smallest objective value, then
   Simulate the system at point $x_2 = \frac{x_R + x^*}{2}$ and calculate $f(x_2)$. let (x_L, x^*, x_R) be (x^*, x_2, x_R).

3. If $f(x^*)$ is the smallest objective value:

   - If $||x^* - x_L|| \geq ||x_R - x^*||$
     Simulate the system at point $x_1 = \frac{x_L + x^*}{2}$ and calculate $f(x_1)$.
     a) If $f(x_1) < f(x^*)$
        let (x_L, x^*, x_R) be (x_L, x_1, x^*).
     b) Otherwise
        let (x_L, x^*, x_R) be (x_1, x^*, x_R).
   - Otherwise
     Simulate the system at point $x_2 = \frac{x_R + x^*}{2}$ and calculate $f(x_2)$.
     a) if $f(x_2) < f(x^*)$
        let (x_L, x^*, x_R) be (x^*, x_2, x_R).
     b) Otherwise
        let (x_L, x^*, x_R) be (x_L, x^*, x_2).

One important question that can be asked here is how long we should stay on a line. Because we have a maximum number of simulations, we need to get out from the picked line and pick a new line if the point is not improved more. Hence, a maximum number of iterations is used before trying a new line. In the implementation of this method, a *counter* is used to count how many iterations, to be in a line. In each iteration, if we can find a new point that has a lower objective value than the point that we want to improve, x^*, then *counter* resets to zero. On the other hand, if in one iteration the point does not improve, then, the *counter* is increased by 1. Finally, when this *counter* is reached to its maximum number of iterations, we get out from the loop and pick a new line.

When we want to pick a new line we need to consider some conditions to be able to enhance the falsification. For picking a new line, again a random line that passes from x^* needs to be considered. If the point that should be improved, x^*, was changed during the optimization process, then we again

similar to the beginning of this optimization, a random line is selected that causes new x_L and x_R to be generated. But, if the improving point was not changed when the *counter* reaches to the maximum number of iterations, we try to take x^* to be equal to the second-best point that is found. This assumption leads to not waste the simulations with a bad point that never can guide us to the falsification process. Then, the new x_L and x_R are generated from the line that passes from the new x^*.

One important property of this method is that is inspired by the crawling procedure in NM for local optimization, but also the ability to get out of local optima and continue the optimization from a new but related point. This method has a simpler implementation than SNOBFIT.

5.2 Summary of Papers

This section provides a summary of the included papers. In Paper A, the performance of two semantics, *Max* and *MARV*, are evaluated on a simple autonomous driving example. The main purpose of this evaluation was to understand the strengths and weaknesses of different semantics. Only, the objective function of the semantics is evaluated without considering any optimization method in Paper A. This paper leads that Paper B suggests the use of the combination of different semantics and produces multiple objective functions using a specific optimization method, in this case, NM.

Evaluation results on some benchmark examples showed that many specifications can be falsified quickly by evaluating combinations of the min and max values in the allowed parameter ranges or by evaluating random parameter combinations. We define a hybrid method that combines using corner parameter values with a random strategy for selecting parameter values within the allowed parameter ranges. On the other hand, some specifications need optimization methods to be falsified. Paper C suggests a new simple gradient-free optimization method for this purpose that is as efficient as SNOBFIT but is having a much simpler implementation.

5.3 Paper A

Zahra Ramezani, Nicholas Smallbone, Martin Fabian, Knut Åkesson
Evaluating Two Semantics for Falsification using an Autonomous Driv-

ing Example
Proceedings of 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), pp. 386-391, July. 2018.
©2019 IEEE DOI: 10.1109/INDIN41052.2019.8972229 .

The falsification process is an optimization procedure where the optimization is performed using a quantitative semantics. Different semantics that has different behavior are introduced in the literature. It is important to understand the strengths and weaknesses of the different semantics. This paper evaluates two *Max* and *MARV* semantics on an adaptive cruise controller of an autonomous car. In the evaluation of it is shown that the *Max* semantics results in constant objective value for some specific parts of the evaluated parameter space, while *MARV* results in a 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.

5.4 Paper B

Zahra Ramezani, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, Knut Åkesson
Multiple Objective Functions for Falsification of Cyber-Physical Systems
To appear in Proceedings 15th Workshop on Discrete Event Systems (WODES), November 2020.

According to the results achieved in Paper A, it was shown 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 as defined by the different quantitative semantics.

5.5 Paper C

Zahra Ramezani, Koen Claessen, Martin Fabian, Knut Åkesson
Enhancing Temporal Logic Falsification with Line Optimization
To be submitted to the possible journal publication .

Falsification of specifications for 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 explore how a simple strategy, based on combining random parameters together with considering extreme combinations of parameter values, performs. The evaluation is done on benchmark problems and the evaluation shows that this simple strategy performs surprisingly well on many of the benchmark problems. We thus consider using this approach as another base-line method when evaluating falsification methods. The second part of this paper discusses a simple gradient-free optimization method, named line optimization, that does the optimization in falsification over a line. This method is compared to the Nelder-Mead and SNOBFIT methods for falsification. The evaluation results show that line optimization enhances the performance of the falsification while having a simple implementation.

CHAPTER 6

Concluding Remarks and Future Work

This thesis has been focused on improving the falsification of cyber-physical systems by making it more efficient. This can be done by considering different aspects. We have enhanced the falsification by using new quantitative semantics and developing a new optimization method. The proposed methods are evaluated on benchmark problems.

The appended papers to this thesis are included in the order they were written. Initially the aim was to evaluate how different semantics might effect the falsification performance. There is no clear answer which semantics works the best for each SUT and given specifications. We show that this depends on the model and the specification. To gain better understanding of the performance of the quantitative semantics, a example of an autonomous driving model is analyzed. The evaluation results shows that for a specific quantitative semantics the objective function values do not change in a large part of the parameter space, but by using another semantics the objective value given by a quantitative semantics did change, thus giving a sense of direction to the optimization algorithm to exploit. This result from Paper A leads to the idea of Paper B where the use of multiple objective functions during the optimization procedure is evaluated.

Not only, the quantitative semantics affect the falsification performance, but also the optimizations affect the efficiency of falsification. Evaluation results on some benchmark problems showed that for some models and the given specifications, it is easy to be falsified on the corners or by just doing some random tests. Hence, a base-line optimization method is suggested in Paper C. In this method, a combination of two corners and random methods is used. On the other hand, optimization methods still are needed for the systems and the specifications that corners and random methods do not perform well. Hence, a new gradient-free optimization method is suggested in Paper C.

Now, we can give answers to the research questions that were formulated in Chapter 1:

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

Different semantics have been introduced in the literature that have specific features, strengths and weaknesses.

Paper A answers this question by evaluation of a simple adaptive cruise controller from an autonomous driving example. Evaluation results were done for two semantics, *Max* and *MARV*. In Paper A it is shown that for *Max* semantics results in constant objective values for some specific parts of the parameter space. It means that there is no useful information for an optimization algorithm to use. On the other hand, *MARV* has a clear direction to the falsification area for the same parameter space.

RQ2. *How do the combination of different semantics affect and improve the falsification of cyber-physical systems?*

Evaluation results from Paper A showed that using a single objective function may cause the optimization to search in the wrong direction or even having no clear direction to follow during the falsification process. Thus, a combination of using multiple objective functions during the optimization is suggested in Paper B. Multiple objective functions are used in an extended version of the Nelder-Mead optimization method. This is done in such a way that the optimizer avoids using the semantics that does not have a clear direction, to be used by the optimizer, to get closer to a falsification point.

It is shown that for the benchmark problems the use of multiple semantics improves the falsification performance.

RQ3 *How can different optimization methods effect the falsification process?*

Falsification, as used in this thesis, is done by using an optimization-based approach. However, many specifications can be falsified quickly by evaluating combinations of the min and max values in the allowed parameter ranges or by evaluating random parameter combinations. To get a base-line to compare optimization-based approaches with, we define a hybrid method that combines using extreme parameter values with a pure random strategy for selecting parameter values within the allowed parameter ranges. Our evaluation of standard benchmark examples shows that this approach performs surprisingly well. This approach and the evaluation is presented in Paper C. On the other hand, there are the specifications that needs optimization methods to be falsified. In Paper C we suggest a new simple gradient-free optimization method. The optimization is done on random lines in this new method. Evaluation of benchmark examples shows that this method enhance the falsification process while still having a simple implementation.

6.1 Future Work

For future work, we plan to continue the work on enhancing the falsification of CPSs. One interesting approach is to exploit the model more by estimating the gradients directly from the model. Another possible direction is to build analytical surrogate models from simulations and use those for the optimization phase. We also plan to develop the autonomous driving example further to handle more complex scenarios and control strategies.

References

[1] 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.

[2] G. Nicolescu and P. J. Mosterman, *Model-based design for embedded systems*. CRC Press, 2009.

[3] S. Abbaspour Asadollah, R. Inam, and H. Hansson, "A survey on testing for cyber physical system", in *Testing Software and Systems*, K. El-Fakih, G. Barlas, and N. Yevtushenko, Eds., Cham: Springer International Publishing, 2015, pp. 194–207.

[4] 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.

[5] G. Ernst, P. Arcaini, A. Donzé, G. Fainekos, L. Mathesen, G. Pedrielli, S. Yaghoubi, Y. Yamagata, and Z. Zhang, "ARCH-COMP 2019 Category Report: Falsification", in *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 61, pp. 129–140, 2019.

[6] J. L. Eddeland, S. Miremadi, and K. Åkesson, "Evaluating optimization solvers and robust semantics for simulation-based falsification", in

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems, 2020.

[7] 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.

[8] 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.

[9] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems", in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.

[10] 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.

[11] J. Fitzgerald, C. Gamble, P. G. Larsen, K. Pierce, and J. Woodcock, "Cyber-physical systems design: Formal foundations, methods and integrated tool chains", in *2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering*, 2015, pp. 40–46.

[12] T. Schattkowsky and W. Muller, "Model-based design of embedded systems", in *Seventh IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2004. Proceedings.*, IEEE, pp. 113–128, 2004.

[13] H. Shokry and M. Hinchey, "Model-based verification of embedded software", *Computer*, vol. 42, no. 4, pp. 53–59, 2009.

[14] A. Junghanns, J. Mauss, and M. Tatar, *Testweaver-a tool for simulation-based test of mechatronic designs-in: Proceedings of the 6th international modelica conference, 3.-4.3. 2008.*

[15] 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.

[16] T. Akazaki, S. Liu, Y. Yamagata, Y. Duan, and J. Hao, "Falsification of cyber-physical systems using deep reinforcement learning", in *Formal Methods*, K. Havelund, J. Peleska, B. Roscoe, and E. de Vink, Eds., Cham: Springer International Publishing, 2018, pp. 456–465.

[17] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems", in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 254–257.

[18] 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.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning", in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 1928–1937.

[20] 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, 408 –415, 2018, 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[21] 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.

[22] 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.

[23] T. Akazaki and I. Hasuo, "Time robustness in mtl and expressivity in hybrid system falsification", in *Computer Aided Verification*, D. Kroening and C. S. Păsăreanu, Eds., Cham: Springer International Publishing, 2015, pp. 356–374.

[24] 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.

[25] 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.

[26] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, "Stochastic local search for falsification of hybrid systems", in *Automated Technology for Verification and Analysis*, B. Finkbeiner, G. Pu, and L. Zhang, Eds., Cham: Springer International Publishing, 2015, pp. 500–517.

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

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

[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*, Y. Lakhnech and S. Yovine, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

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

[32] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications", in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[33] 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.

[34] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka, "On temporal logic and signal processing", in *Automated Technology for Verification and Analysis*, S. Chakraborty and M. Mukund, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 92–106.

[35] M. Wright, "Direct search methods: Once scorned, now respectable", English (US), in *Numerical analysis*, D. Griffiths and G. Watson, Eds. Addison-Wesley, 1996, pp. 191–208.

[36] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods", *SIAM Review*, vol. 45, pp. 385–482, 2003.

[37] 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.

Part II

Papers

PAPER A

Evaluating Two Semantics for Falsification using an Autonomous Driving Example

Zahra Ramezani, Nicholas Smallbone, Martin Fabian, Knut Åkesson

Proceedings of 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), pp. 386-391, July. 2018.
©2019 IEEE DOI: 10.1109/INDIN41052.2019.8972229

The layout has been revised.

Abstract

We consider the falsification of temporal logic properties as a method to test complex systems, such as autonomous systems. Since these systems are often safety-critical, it is important to assess whether they fulfill given specifications or not. An adaptive cruise controller for an autonomous car is considered where the closed-loop model has unknown parameters and an important problem is to find parameter combinations for which given specification are broken. We assume that the closed-loop system can be simulated with the known given parameters, no other information is available to the testing framework. The specification, such as, the ability to avoid collisions, is expressed using Signal Temporal Logic (STL). In general, systems consist of a large number of parameters, and it is not possible or feasible to explicitly enumerate all combinations of the parameters. Thus, an optimization-based approach is used to guide the search for parameters that might falsify the specification. However, a key challenge is how to select the objective function such that the falsification of the specification, if it can be falsified, can be falsified using as few simulations as possible. For falsification using optimization it is required to have a measure representing the distance to the falsification of the specification. The way the measure is defined results in different objective functions used during optimization. Different measures have been proposed in the literature and in this paper the properties of the *Max* Semantics (MAX) and the Mean Alternative Robustness Value ($MARV$) semantics are discussed. After evaluating these two semantics on an adaptive cruise control example, we discuss their strengths and weaknesses to better understand the properties of the two semantics.

1 INTRODUCTION

For autonomous systems in general and autonomous vehicles in particular, it is critical to use rigorous testing methods so that such vehicles will be significantly safer than they are with humans in the loop. Autonomous systems consist of perception, sensor-fusion, decision and control modules implemented in software that interact with the physical sensors and actuators of the system. As remarked in [1], the biggest challenge for autonomous vehicles is in creating an end-to-end design and deployment process that integrates the safety concerns. Formal verification and correct by construction techniques should certainly be used for those sub-systems where they can be applied. However, it is known [2] that for hybrid systems, i.e., systems consisting of both digital and analog components, the problem of deciding if a state is reachable or not, is undecidable in general. Thus, for any autonomous systems of reasonable complexity, testing will be an important part of the design process.

Model-based design is often used for the design of autonomous systems, and building high fidelity models of both software and hardware is a part of the design process. Since testing on physical hardware is both time-consuming and limited by the available physical hardware, it is an advantage to do as much testing as possible using only the models. One approach [3] is to use formal specifications of properties that the closed-loop system should satisfy combined with the use of simulation of the models to evaluate whether the desired properties are fulfilled or not. This can be combined with *falsification* techniques that search for counterexamples to given specifications of the closed-loop system.

Metric Interval Temporal Logic (MITL) [4] and Signal Temporal Logic (STL) [5] can be used to describe real-time properties of systems. Several quantitative semantics for these logics have been proposed to not only allow reasoning about the correctness of a signal with respect to a model but also give a real value that indicates how far a signal is from satisfying or violating a specification. During falsification, these values are used by an optimizer to find new input signals with a higher likelihood of violating the specification. The quantitative semantics chosen will influence the efficiency of the falsification procedure, and the efficiency of different quantitative semantics is problem-dependent.

To facilitate different quantitative semantics the concept of *Valued Booleans* (VBools) was introduced in [6]. Multiple quantitative semantics can be ex-

pressed using VBool, for example the *MAX* semantics, that is a widely used semantics for optimization based falsification. Also [7] introduced and investigated several alternative robustness measures, of which the Mean Alternative Robustness Value (*MARV*) will be considered here.

In this paper, we evaluate the feasibility of automated falsification techniques for one sub-system, an adaptive cruise controller, of an autonomous vehicle. The purpose of this paper is not to evaluate how automated falsification techniques can be used for fully autonomous vehicles but to improve our understanding of how different semantics can be used to facilitate the falsification process. Given formal specification and simulation traces, the semantic will result in a scalar that not only expresses if the specification is fulfilled or not but also to what extent. This additional information is used by an optimizer to adjust the input signals and parameters to the next simulation with the intent of finding a new set of input signals and parameters that are more likely to falsify the specification.

We aim with this paper to show how a rigorous method based on optimization can be used in the design process of autonomous vehicles and to give the reader an insight into how the quantitative semantics works for this particular problem.

The rest of the paper is organized as follows. Section 2 introduces the falsification of temporal properties. Section 3 introduces the adaptive cruise controller example. Section 4 evaluates the performance of the optimization algorithm when using *MAX* and *MARV*. Finally, Section 5 summarizes the contributions.

2 Falsification

Falsification of temporal logical properties is based on an optimization procedure where the objective function is determined by the definition of a robustness semantics for the temporal logic formalism. Breach [8] and S-TaLiRo [9] are two tools implemented on top of Matlab/Simulink that can do falsification assuming that the closed-loop system can be simulated. In this work, Breach is used for the simulation and hence STL [10] is used to model the specifications, but the discussion in this paper can be applied to MITL used in S-TaLiRo as well.

2.1 Signal Temporal Logic

The syntax of STL is defined as follows [11]:

$$\varphi ::= \mu \,|\, \neg\mu \,|\, \varphi \wedge \psi \,|\, \varphi \vee \psi \,|\, \Box_{[a,b]}\psi \,|\, \Diamond_{[a,b]}\psi \,|\, \varphi\, \mathcal{U}_{[a,b]}\psi,$$

where the predicate μ is $\mu \equiv \mu(x) > 0$; φ and ψ are STL formulas; $\Box_{[a,b]}$ denotes the *globally* operator between a and b; $\Diamond_{[a,b]}$ denotes the *finally* operator between a and b; and $\mathcal{U}_{[a,b]}$ denotes the *until* operator between a and b. The semantics of STL are defined by considering the discrete signal x at time instant k [11]:

$$\begin{aligned}
(x,k) &\models \mu && \Leftrightarrow\ \mu(x[k]) \\
(x,k) &\models \neg\mu && \Leftrightarrow\ \neg((x,k) \models \mu) \\
(x,k) &\models \varphi \wedge \psi && \Leftrightarrow\ (x,k) \models \varphi \wedge (x,k) \models \psi \\
(x,k) &\models \varphi \vee \psi && \Leftrightarrow\ (x,k) \models \varphi \vee (x,k) \models \psi \\
(x,k) &\models \Box_{[a,b]}\varphi && \Leftrightarrow\ \forall k' \in [k+a, k+b], (x,k') \models \varphi \\
(x,k) &\models \Diamond_{[a,b]}\varphi && \Leftrightarrow\ \exists k' \in [k+a, k+b], (x,k') \models \varphi \\
(x,k) &\models \varphi\, \mathcal{U}_{[a,b]}\psi && \Leftrightarrow\ \exists k' \in [k+a, k+b]\ \ (x,k') \models \psi \\
& && \quad \wedge \forall k'' \in [k,k'], (x,k'') \models \varphi
\end{aligned}$$

Instead of just checking the Boolean satisfaction of an STL formula, the notion of a robust semantics is defined to measure how far away a specification is from being satisfied. In the next part, these robust semantics will be introduced.

2.2 Valued Booleans and *MAX* semantics

A VBool is a combination of a Boolean value together with a *robustness* value, a non-negative real number that indicates how true or false the VBool is. In [6], VBools are used to define two semantics aimed at measuring the robustness of STL formulas in a testing setting; *MAX* evaluated in this paper, and the *Additive*.

In this work, the robustness value will be used as a measure of how convincingly a test passed, or how severely it failed, respectively. The comparison operator $\leq_{\mathrm{v}}$ corresponds to $\leq$ and takes the difference between its arguments

as its robustness. This is because, in order for the value of $x \leq y$ to change, one of the arguments has to change by at least $|x - y|$.

$$\leq_{\mathrm{v}} : \mathbb{R} \times \mathbb{R} \to \mathbb{V}$$
$$x \leq_{\mathrm{v}} y = \begin{cases} (\top, y - x) & \text{if } x \leq y \\ (\bot, x - y) & \text{otherwise,} \end{cases}$$

where $\top$ and $\bot$ denote true and false, respectively.

MAX is defined by the *MAX-and*, *MAX-or*, *MAX-always*, and *MAX-eventually* operators.

The *MAX-and* operator $\wedge_{MAX}$ is defined as:

$$\begin{aligned} (\top, x) \wedge_{MAX} (\top, y) &= (\top, min(x, y)) \\ (\top, x) \wedge_{MAX} (\bot, y) &= (\bot, y) \\ (\bot, x) \wedge_{MAX} (\top, y) &= (\bot, x) \\ (\bot, x) \wedge_{MAX} (\bot, y) &= (\bot, max(x, y)). \end{aligned} \tag{A.1}$$

The *MAX-always* operator is defined over [a, b] as:

$$\Box_{MAX,[\mathrm{a,b}]}\, \varphi = \bigwedge_{k=a}^{b}{}_{MAX} \varphi[k], \tag{A.2}$$

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

Other operators, like *MAX-or*, *MAX-eventually* and *MAX-until* can be expressed in terms of the above operators. These operators are not used in this paper. For implementation of *MAX* in Breach, a VBool is represented as a single real value, where a negative value represents that the VBool is $\bot$, and a positive value represents that it is $\top$. In both cases, the magnitude of the real value is the robustness value.

2.3 Mean Alternative Robustness Value (*MARV*) semantics

In this paper we restrict the comparison to timed always operators since this is a common temporal operator. For *MARV* [7] the discrete-time always operator is described by the following formula.

$$\square_{MARV,[a,b]}\varphi = \begin{cases} \frac{1}{b-a}\sum_{i=0}^{M-1} \rho(\varphi, t_i)\,(t_{i+1} - t_i) & \rho \geq 0 \\ \rho(\varphi, t_i) & \rho < 0 \end{cases} \tag{A.3}$$

where $\rho(\varphi, t_i)$ is a real-valued function that gives the objective value at each time instant, t_i, and M is the number of sampling times over the interval [a, b].

For positive robustness values, which represent the case when $\square_{MARV,[a,b]}\varphi$ is satisfied $MARV$ calculates the mean over the interval.

3 Use-Case

This study of the two semantics uses a simple example[1] involving two vehicles, an autonomous vehicle, called the *ego* vehicle, and a *lead* vehicle. The two vehicles travel on a road in the same direction (Fig. 1).

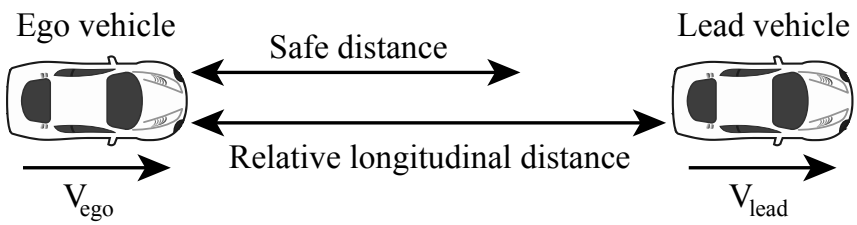

Figure 1: Definition of the distances between the ego vehicle and the lead vehicle.

The ego vehicle is equipped with adaptive cruise control (ACC). It has a sensor that measures the relative distance to, and the relative velocity of, the lead car. The ACC system operates in two modes: *speed* mode and *safety* mode. In the speed mode, the ego vehicle travels at a driver-set speed, and in the safety mode, the ego car maintains a safe distance from the lead car. The ACC system decides which mode to use based on real-time radar measurements; either it needs to keep a safe distance, or if the relative distance is safe, it can increase its speed to the driver-set speed.

[1] A demo example from the Matlab®/Simulink® toolbox.

3.1 Specification of Safe Longitudinal Distance

As mentioned earlier, safety guarantees are important in autonomous driving. However, the safe distance between two vehicles depends on many parameters, including road friction, braking force, and response time. These parameters are largely unknown: while some of these parameters might be estimated by the ego vehicles, less is known about, for example the maximal braking force of the lead vehicle. Various studies, [12], [13], have addressed the issue of the minimum safe longitudinal distance between two vehicles. In our experimental setup the controller in the ACC system is designed to keep the safe distance d_{safe} between the two vehicles by calculating the set-point distance using the following formula:

$$d_{safe} = d_{default} + t_{gap}\, v_{ego}, \tag{A.4}$$

where $d_{default}$ is the standstill default spacing, in this case set to 10 m, and t_{gap} denotes the time gap between the vehicles which is set to 1.4 s. The distance d_{safe} is used as a reference value for the ACC controller, but it cannot be used as a specification because when the ego car is in the safe mode, any deceleration by the lead car will make the longitudinal distance less than d_{safe}.

Note, the formula above is used to generate the set-point distance between the two vehicles, but it is not suitable as a specification because having a shorter distance between the vehicles does not necessarily imply that a collision will occur. It is certainly possible to formulate a specification that models that the two vehicles do not collide by specifying that the distance between the two vehicles should be positive. However, falsification may be easier if we strengthen the specification by exploiting physical insight. In our example, we will calculate the *minimal* distance d_{min} between two vehicles using the approach in [13] based on physical properties, such that if the vehicles are never closer than d_{min} to each other, collisions are guaranteed to be avoidable when the lead car brakes:

$$\begin{aligned} d_{min} = \Bigg[& v_{ego}\, t_r + \frac{1}{2} a_{max,acc}\, t_r^2 \\ & + \frac{(v_{ego} + a_{max,acc}\, t_r)^2}{2\, a_{min,brake}} - \frac{v_{lead}^2}{2\, a_{max,brake}} \Bigg]_+, \end{aligned} \tag{A.5}$$

where $[x]_+$ means the maximum of x and 0, and where the parameters are described in Table 1.

The specification used for falsification now expresses that at all times the relative distance between the cars must be greater than the safe distance d_{min}. With T as the simulation time, this is formulated as:

$$\square_{[0,T]}\big(\textit{relative longitudinal distance} > d_{min}\big). \tag{A.6}$$

The falsification process is significantly easier when we specify that the relative distance between the vehicles must be at least d_{min}, rather than just positive. This strengthening of the specification is justified by, as soon as, the relative distance between the vehicles is less than d_{min}, immediate braking by the lead vehicle might result in a collision. Thus the specification amounts to assuming worst-case behavior from the lead vehicle.

4 Evaluation Results

To evaluate the performance of the *MAX* and *MARV* semantics, falsification of the AD example is studied. The vehicle parameters used during the simulations of the closed-loop behavior are presented in Table 1.

Table 1: Parameters used in the example.

Parameters	Notations and Values
Velocity of lead car (m/s)	v_{lead}
Velocity of ego car (m/s)	v_{ego}
The driver-set velocity (m/s)	$v_{set} = 30$
Max acceleration of ego car (m/s^2)	$a_{max,acc} = 3$
Min acceleration of ego car to full stop (m/s^2)	$a_{min,brake}$ = -2.5
Max acceleration of lead car to full stop (m/s^2)	$a_{max,brake}$ = -3
Response time (sec)	$t_r = 0.1$

The simulation takes the acceleration of the lead vehicle a_{lead} as input and simulates the behavior of the closed-loop system. Before a simulation of the closed-loop system starts, the values of input parameters are selected by the falsification algorithm; in this example, the parameters are a_{lead0} and a_{lead1}. The simulation time is $T = 30$ s and the simulation starts with a_{lead0} chosen in the range $[0, 3]$ and a_{lead1} in the range $[-3, 0]$. For both the *MAX* and

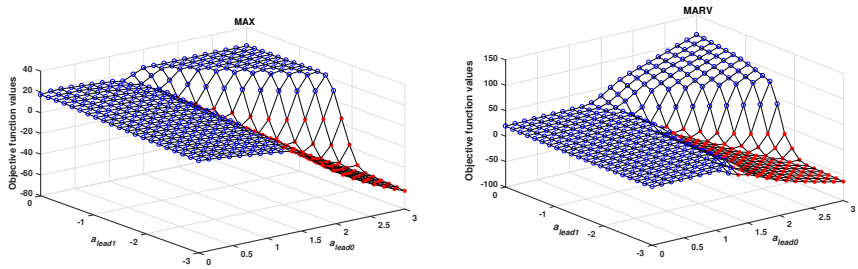

Figure 2: The objective values for combinations of the accelerations (a_{lead0}, a_{lead1}) for the two semantics *MAX* and *MARV*. Positive values ($\circ$) mean that the specification is satisfied, while negative values ($*$) mean that it is falsified. Note that the signs of the values should be the same for both semantics while the absolute values might be different.

MARV semantics the specification is given by formula (A.6). To illustrate the similarities and differences between the *MAX* and *MARV* semantics the objective value is calculated for different combinations of the parameters a_{lead0} and a_{lead1}. In this case a_{lead0} and a_{lead1} are divided into 20 equidistant points resulting in 400 parameter combinations, each requiring its own simulation.

The objective function values calculated for each of the 400 simulations for both *MAX* and *MARV* semantics are shown in Fig. 2. The main purpose of calculating an objective value is to guide the falsification process in the right direction by choosing the next set of parameters to be simulated such that the likelihood of falsifying the specification is increased. Note that in this work we do not assume that gradients can be derived analytically; instead, they have to be estimated by evaluating multiple parameter combinations. Ideally, a semantic should be such that when a parameter change brings the system closer to falsifying a specification, then the objective value of the specification should decrease.

By comparing the left and right graphs in Fig. 2 we notice that they have the same sign for every parameter combination. This is expected, since the sign indicates if the specification is fulfilled, which does not depend on which semantics is used. However, we observe that in the upper right corner (where close to $a_{lead0} = 3$ and $a_{lead1} = 0$) the two semantics result in different estimates of the gradients. In this region, for *MAX* the objective function values are the same for each point (the upper triangular side of the left graph).

In order to illustrate why, Fig. 3 presents for each choice of a_{lead0} and a_{lead1} the first time at which the objective value reaches its minimum when using *MAX*. We observe that for parameters close to $a_{lead0} = 3$ and $a_{lead1} = 0$, the simulation time at which the minimal objective value is reached is time 0. The reason for this can be seen in Fig. 4, which shows the relative and safe distance, and the velocities of the lead and ego vehicles, when $a_{lead0} = 3$ and $a_{lead1} = 0$. According to this figure, when the lead car continuously accelerates, the ego car increases its speed, too. But the ego car has a driver-set velocity limit (30 m/s) so that the relative distance between the vehicles always increases and the *minimum* objective function value occurs at the beginning of the simulation where the relative and safe distances are the closest. Thus, the *MAX* semantics consider the different simulations to be equally good/bad for parameters that are close to $a_{lead0} = 3$ and $a_{lead1} = 0$, resulting in no information that can be used by the optimization algorithm. On the other hand, as can been seen in the right graph of Fig. 2, these points have different values under *MARV*.

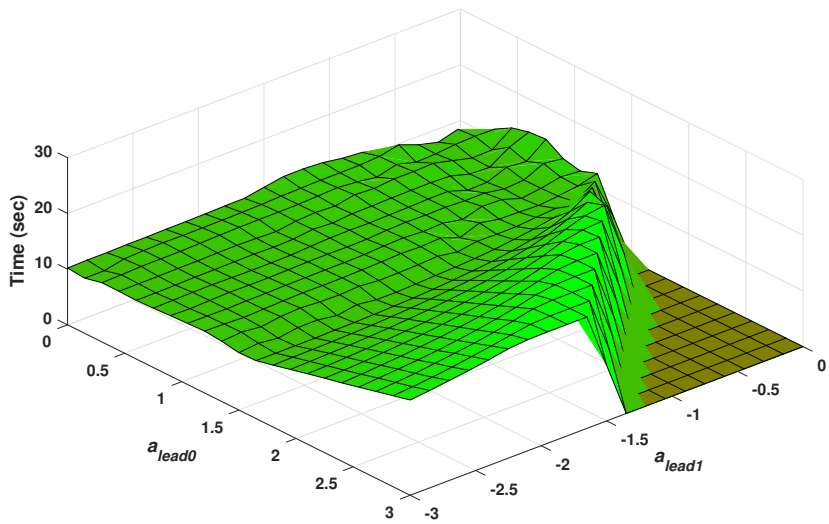

Figure 3: The simulation time when the minimum objective function value is reached for the first time for *MAX*. Note that a_{lead1} is here to the right.

In Fig. 2, where acceleration of the lead vehicle a_{lead0} is in the range $[0, 1.2]$,

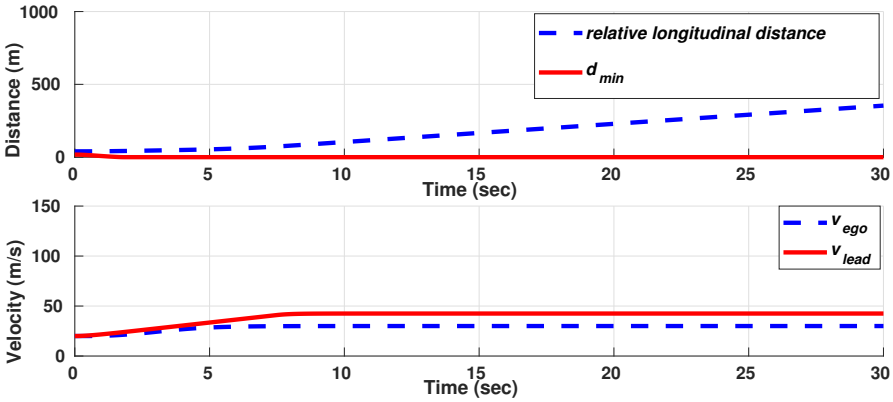

Figure 4: The relative and safe distance, and the vehicle velocities for $a_{lead0} = 3$ and $a_{lead1} = 0$. The minimum objective function value occurs at time zero, where the relative and safe distances are the closest.

and a_{lead1} in the range $[-3, 0]$, for all parameter combinations in these ranges, the ego vehicle behaves safely and keeps the safe distance from the lead vehicle. In order to show the behaviour of both vehicles in these ranges, the relative and safe distances and the velocities of the ego car v_{ego} and the lead car v_{lead} are shown for $a_{lead0} = 0$ and $a_{lead1} = -3$ in Fig. 5. As can be seen, the relative distance is greater than d_{min} for the whole of the simulation, it means when the lead car brakes the ego car starts braking too, and it can stop within safe distance from the lead car.

In Fig. 2, the safety formula (A.6) is violated where the objective values are negative. The relative and safe distances, and the velocities of both cars for the point $a_{lead0} = 3$ and $a_{lead1} = -3$, are shown in Fig. 6. As can be seen, when the lead car accelerates, the ego car increases its velocity to reach the driver-set velocity. Then, when the lead car brakes, because their distance is larger than d_{safe} (A.4), the controller is in speed mode and only after a delay does the ego car switch to safety mode and adjust its speed to maintain a safe distance from the lead car. As a result, not only does the relative distance between the vehicles become less than d_{min}, but the cars even crash. Note that while the cars crash at around 22 s in this scenario, in Fig. 6 the ego car does not stop until around 26 s. This is due to the simple model used in the example that does not model the actual collision.

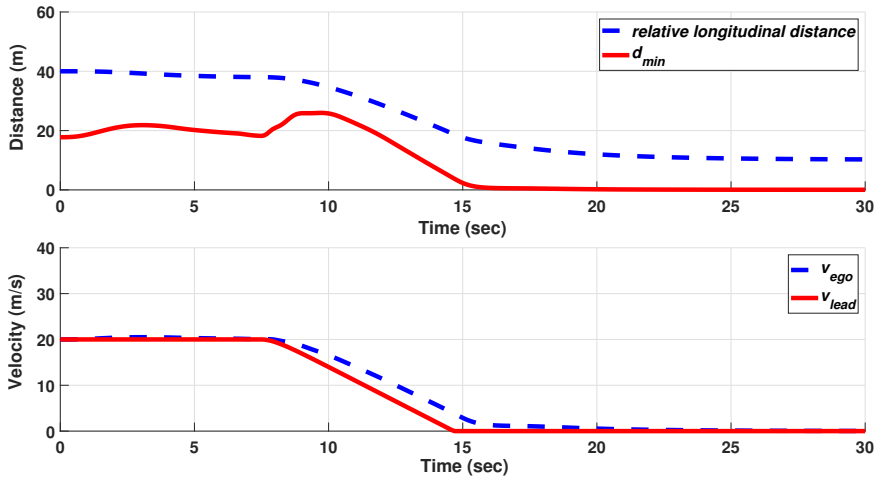

Figure 5: The relative and safe distance, and the vehicle velocities for $a_{lead0} = 0$ and $a_{lead1} = -3$.

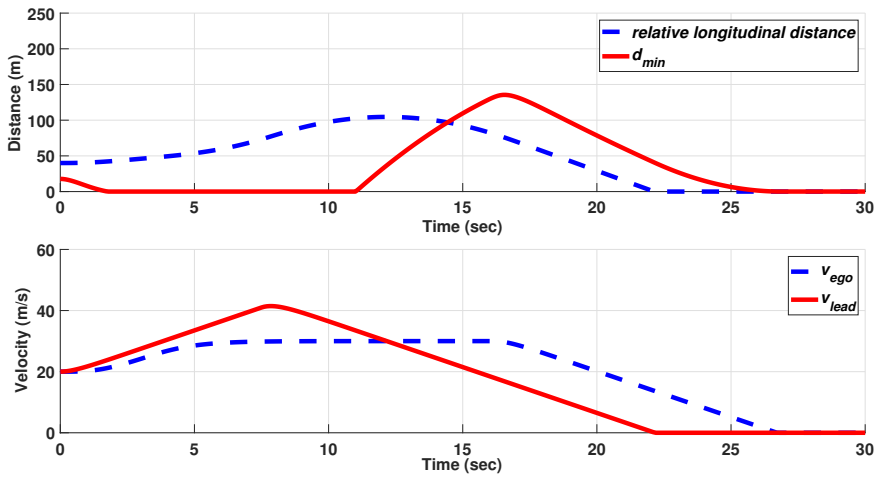

Figure 6: The relative and safe distance, and the vehicle velocities for $a_{lead0} = 3$ and $a_{lead1} = -3$. At around 14 s the relative distance between the vehicles becomes less than d_{min}, and a collision occurs at around 22 s.

By comparing the objective values from the two semantics *MAX* and *MARV*, we observe that since the *MAX-always* only considers the minimal value of the objective function it is possible to end up with objective values that do not differ between different simulations. Thus, the optimizer has no information in which direction to further explore the search. In this example we observed that in this case it might be beneficial to consider *MARV* since this semantic will result in higher objective values when the vehicles move further away from each other, thus providing information to the optimization algorithm that might guide the optimizer in the right direction to falsify the specification.

In this work, we have only considered the objective functions used by the optimization algorithm but not the optimization algorithms themselves. However, gradient-free optimization algorithms like Nelder-Mead or simulated annealing are typically used in the falsification process, and it is clear that the performance of these algorithms depends on having objective values that are not constant and that direct the search in a direction where the objective values decrease, increasing the chances of falsifying the specification.

5 Conclusion

In this paper, we showed how the efficiency of falsification might be affected by the semantics used to evaluate the specification. An adaptive cruise controller is used as an example and the objective is to test if the certain parameter combinations result in the possibility for two vehicles to collide. Using the example, we showed a situation where the *MAX* semantics will result in the same objective value for closely related parameter values, while *MARV* results in a non-constant objective value, which might guide the optimization algorithm in a direction that increases the chances of falsifying the specification. The objective of this paper was not to compare the efficiency of *MAX* and *MARV*, but rather to illustrate with an example of the importance of choosing a suitable semantics for the problem at hand. From our experience with industrial-scale systems we have observed that the simulation time is the most limiting factor, not the evaluation of the simulation results using different semantics. Thus, a future strategy might be to evaluate the simulation using multiple objective functions and use a high-level algorithm that during the optimization will take into account multiple objective values computed

using several different semantics.

Acknowledgements

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

References

[1] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge", *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017, ISSN: 1939-1390.

[2] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?", *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998, ISSN: 0022-0000.

[3] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "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.

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

[5] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals", in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[6] 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, 408 –415, 2018, 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[7] 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.

[8] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems", in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.

[9] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems", in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 254–257.

[10] 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 Berlin Heidelberg, 2010, pp. 92–106.

[11] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications", in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[12] G. Feng, W. Wang, J. Feng, H. Tan, and F. Li, "Modelling and simulation for safe following distance based on vehicle braking process", in *2010 IEEE 7th International Conference on E-Business Engineering*, 2010, pp. 385–388.

[13] 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.

PAPER B

Multiple Objective Functions for Falsification of Cyber-Physical Systems

Zahra Ramezani, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, Knut Åkesson

To appear in Proceedings 15th Workshop on Discrete Event Systems (WODES), November 2020

The layout has been revised.

Abstract

Cyber-physical systems are typically safety-critical, thus it is crucial to guarantee that they conform to given *specifications*, that are the properties that the system must fulfill. Optimization-based falsification is a model-based testing method to find counterexamples of the specifications. The main idea is to measure how far away a specification is from being broken, and to use an optimization procedure to guide the testing towards falsification. The efficiency of the falsification is affected by the objective function used to evaluate the test results; different objective functions are differently efficient for different types of problems. However, the efficiency of various objective functions is not easily determined beforehand. This paper evaluates the efficiency of using multiple objective functions in the falsification process. The hypothesis is that this will, in general, be more efficient, meaning that it falsifies a system in fewer iterations, than just applying a single objective function to a specific problem. Two objective functions are evaluated, *Max*, *Additive*, on a set of benchmark problems. The evaluation shows that using multiple objective functions can reduce the number of iterations necessary to falsify a property.

1 INTRODUCTION

Cyber-Physical Systems (CPSs) consist of computational parts described by state models, communicating with a physical environment described by differential equations. Using high fidelity models is typical in *model-based design* of CPSs. Testing and verifying the correctness of all physical and cyber components of CPSs are important and a big challenge [1]. An autonomous car is an example of a CPS where it is necessary with rigorous methods to assure the correctness of the system. Typically, *formal verification* and/or *testing* are used for this purpose.

For complex systems, testing is a necessary part of the design process since formal verification of systems with a combination of discrete and continuous

dynamics is an undecidable problem [2]. However, for both formal verification and testing, formal specifications of the properties that should be fulfilled are required in order to enable an automated approach. One approach, surveyed by [3], that can be used is formal specification of properties that the closed-loop system should satisfy, combined with simulation of models and where the given specifications are monitored and it is evaluated whether they are satisfied or not. This can be combined with *falsification* techniques that search for counterexamples to given specifications of the closed-loop system.

The falsification process can be based on an optimization procedure where the optimization is performed over an input parametrization expressing possible input signals. The aim is to find counterexamples, if possible, to specifications of the system under test. This is done in an iterative manner where an objective function measures the distance to the specification being falsified. The objective function is determined by the definition of quantitative semantics for temporal logic formalisms [4]. Metric Interval Temporal Logic (MITL) [5] and Signal Temporal Logic (STL) [6], are two variants of formalisms for which quantitative semantics can be defined. The main purpose of calculating an objective function value is to guide the testing process towards falsification by choosing the next set of parameters for the input signals to the system being simulated, such that the likelihood of falsifying the specification is increased.

For industrial systems typically only a *black-box* of the system under test is available, meaning that only input-output behavior of the system can be observed. In general, these systems are a mix of continuous dynamics, discrete event dynamics, and algorithms implemented using general-purpose programming languages. Parts of the system might also be implemented using physical hardware. Breach [7] and S-TaLiRo [8] are two Matlab/Simulink based toolboxes used for test monitoring and falsification. Both tools search for trajectories of minimal quantitative value to find counterexamples to MITL/STL specifications. [9] show how STL specifications can be automatically derived from Simulink blocks expressing the specifications, this is of practical value to engineers that are working with testing and falsification since it is not necessary to work with temporal logic specifications directly.

For both Breach and S-TaLiRo, the falsification process is guided by an optimization procedure. Due the system being a black-box, gradient-free optimization methods have to be used. Nelder-Mead [10] is a common gradient-

free optimization method that can be used by Breach to guide the falsification process. Although the optimization method does not use explicit gradients, the method will attempt to search in a direction that results in a smaller objective value, where the quantitative semantics are defined in such a way that a negative objective value means that the specification is falsified. In this work, we evaluate three quantitative semantics, *Max* and *Additive* to define the objective function for the falsification processes. In previous work, [11], Valued Booleans (VBools) were introduced as a way to express different quantitative semantics in a coherent way.

In [12], the *Max* and *MARV* semantics for defining objective functions were evaluated for an autonomous driving example. It was shown that, for certain areas of the parameter space, *Max* results in constant objective values, while *MARV* results in non-constant objective values. If the Nelder-Mead (NM) solver starts in an area where the objective values are constant (like for *Max*), it might eventually finish the optimization procedure without finding any falsifying point. This happens because there is no useful information for the optimization algorithm to guide the search to areas with parameters where the objective function has a lower value. As the simulation time is the most limiting factor; the more simulations that have to be run, the longer the falsification process takes.

The contribution of this paper is the introduction of a modified optimization approach that takes advantage of multiple objective functions for the purpose of falsifying specifications. The objective functions have in common that if the specification is satisfied the value of the objective function is positive and if the specification is falsified the value is negative. The motivation for this work is that evaluating multiple objective functions is often significantly less time-consuming than simulating or executing the system, and the objective values of multiple quantitative semantics can be computed using a single simulation of the system under test. The modified optimization approach then heuristically chooses which one of the parameter configurations to simulate next based on the variance (distance) of the respective objective function values. That is, for each iteration of the NM solver, the heuristic picks the point given by the quantitative semantic that has the largest variance (distance). This avoids using the semantic that has close to constant objective values. The approach is evaluated on a set of benchmark examples. The results show that using multiple objective functions can indeed falsify system properties in fewer

simulation runs, compared to using only a single objective function.

In the following, Section 1.1 introduces the quantitative semantics and different ways to define the objective functions used for the falsification process. Section 2 proposes the suggested multiple objective functions in this paper. Section 3 introduces the three benchmark examples. Section 4 evaluates the performance of the suggested optimization on benchmark examples. Finally, Section 5 summarizes the contributions.

1.1 Quantitative Semantics and Objective Functions

In this paper, Breach is used for falsification, hence STL is used to model the specifications. The syntax of STL is defined as follows [13]

$$\varphi ::= \mu \,|\, \neg\mu \,|\, \varphi \wedge \psi \,|\, \varphi \vee \psi \,|\, \square_{[a,b]}\psi \,|\, \Diamond_{[a,b]}\psi$$

where the predicate μ is $\mu \equiv \mu(s) > 0$ and s is a signal; φ and ψ are STL formulas; $\square_{[a,b]}$ denotes the *globally* operator between times a and b (with $a < b$); $\Diamond_{[a,b]}$ denotes the *finally* operator between a and b.

The satisfaction of the formula φ with respect to the discrete signal s at the discrete time instant k is defined as:

$$\begin{aligned}
(s,k) &\models \mu && \Leftrightarrow\ \mu(s[k]) \\
(s,k) &\models \neg\mu && \Leftrightarrow\ \neg((s,k) \models \mu) \\
(s,k) &\models \varphi \wedge \psi && \Leftrightarrow\ (s,k) \models \varphi \wedge (s,k) \models \psi \\
(s,k) &\models \varphi \vee \psi && \Leftrightarrow\ (s,k) \models \varphi \vee (s,k) \models \psi \\
(s,k) &\models \square_{[a,b]}\varphi && \Leftrightarrow\ \forall k' \in [k+a, k+b], (s,k') \models \varphi \\
(s,k) &\models \Diamond_{[a,b]}\varphi && \Leftrightarrow\ \exists k' \in [k+a, k+b], (s,k') \models \varphi
\end{aligned}$$

Instead of only checking the boolean satisfaction of an STL formula, the notion of a quantitative value, i.e. an objective value, will be defined in order to measure how far away a specification is from being falsified. A Valued Boolean (VBool) [11] (v, x) is a combination of a Boolean value v (true $\top$, or false $\bot$) together with a real number x that is a measure of how true or false the specification is. This value will be used as a measure of how convincingly a test passed, or how severely it failed, respectively. In the original VBool

definition, x is defined to always be non-negative. However, in this paper we use the convention that x is negative when v is false, and positive otherwise.

1.2 Quantitative Semantics

Using VBools, we define three quantitative semantics: *Max*, which is essentially the same as standard STL quantitative semantics; *Additive*; For these semantics we define the respective *and*, *or*, *always*, and *eventually* operators.

For conjunction, the semantics differ only in the two cases where the truth values are the same:

	Max	*Additive*
$(\top, x) \wedge (\top, y) =$	$(\top, \min(x, y))$	$\left(\top, \frac{1}{\frac{1}{x}+\frac{1}{y}}\right)$
$(\top, x) \wedge (\bot, y) =$	$(\bot, y)$	
$(\bot, x) \wedge (\top, y) =$	$(\bot, x)$	
$(\bot, x) \wedge (\bot, y) =$	$(\bot, \max(x, y))$	$(\bot, x + y)$

Using the de Morgan laws, the *or* operator can be defined in terms of *and*, as $(v_x, x) \vee (v_y, y) = \neg_{\mathrm{v}}(\neg_{\mathrm{v}}(v_x, x) \wedge \neg_{\mathrm{v}}(v_y, y))$, where VBool negation is defined as $\neg_{\mathrm{v}}(v_x, x) = (\neg v_x, -x)$.

For the *Max* semantics, the *always* operator over an interval $[a, b]$ is straightforwardly defined in terms of *and*, as $\Box_{[a,b]}\varphi = \bigwedge_{k=a}^{b} \varphi[k]$, where φ is a finite sequence of VBools defined for all the discrete time instants in $[a, b]$.

For the *Additive* semantics, though, *always* is a bit more elaborate: $\Box_{[a,b]}\varphi = \bigwedge_{k=a}^{b} \varphi[k] \# \delta t$, where δt is the simulation step size that makes the quantitative value independent of the simulation time, and $\#$ is $(\bot, x) \# \delta t = (\bot, x \cdot \delta t)$ and $(\top, x) \# \delta t = (\top, x/\delta t)$. Furthermore, the *eventually* operator is for all three semantics defined over an interval $[a, b]$ in terms of *always*, as $\Diamond_{[a,b]}\varphi = \neg(\Box_{[a,b]}(\neg_{\mathrm{v}}\varphi))$.

2 Falsification Using Multiple Objective Functions

This section presents the multiple objective functions for falsification of CPSs. Different combinations of objective functions and different strategies for switching between objective functions is discussed in this paper. The main optimization algorithm considered in this paper is an implementation of Nelder-Mead

which is also included in Breach. This algorithm is implemented as *fminsearch* [14] in Matlab. We propose a modified optimization algorithm, based on Nelder-Mead, that exploits multiple objective functions. The new algorithm is presented in Algorithm 1 and works in the following way. The presentation of the algorithms is based on two objective functions, in this case using the *Max* and *Additive* semantics, however the approach is generic and can be applied to an arbitrary number of objective functions.

1. The algorithms starts with p sample points. For each example, p is different and it refers to the number of inputs of each example multiple by 10. All these points are sorted from lowest to highest according to the values of different objective functions. Note, that objective values will be different for each used quantitative semantic, thus there will be one ordering for each semantic. By using a heuristic algorithm to choose which quantitative semantic, the minimum point is considered and it will generate n new points surround the first point before starting the optimization.

2. The first $n + 1$ points in the parameter space are needed by the NM style optimization algorithm to start the optimization process. Again, all these points must be sorted from lowest to highest according to the values of the different objective functions.

3. For each ordering of the objective values, the algorithm will suggest a new point in the parameter space for evaluation. For each quantitative semantic there will be an ordering of the points in the parameter space that is based on the objective value used for the specific semantic. By using a heuristic algorithm to choose which quantitative semantic, one new point will be selected for further evaluation, i.e. being simulated.

4. Now, one iteration in the optimization can execute, i.e. the execution of Step 4 to 7 in Algorithm 1.

5. When reaching to Step 8, once again calculate the objective values for *all* of the considered quantitative semantics and saved points, and create one ordering for each of the considered quantitative semantic. Choose $n + 1$ points with lowest objective value function of the semantic that wins the heuristic strategy.

Note, a new quantitative semantic can be selected in each iteration of the algorithm. In this paper, we have implemented two heuristic algorithms that are based on the variance of the $n+1$ lowest objective values and the distance of largest and lowest objective functions.

Strategy 1 (Variance). For each ordering of objective values, i.e. one ordering for each quantitative semantic, consider the $n+1$ points with lowest objective value, i.e. the points that according to the semantics that are closest to falsifying the specification. For these points, calculate the variance among the $n+1$ points using

$$\sigma_k^2 = \frac{\sum_{i=1}^{n+1} (f_i^k - \mu)^2}{n+1},$$

where f_i^k refers to the objective function value of each $n+1$ points that have lowest value for each semantic. μ is the mean of $n+1$ points.

Strategy 2 (Distance). The distance can be calculated using

$$Dis_k = \frac{f_{max}^k - f_{min}^k}{||x_{max}^k - x_{min}^k||},$$

where f_{max}^k and f_{min}^k refer to the maximum and minimum objective function values of the $n+1$ points that have the lowest objective value of each semantic. x_{max}^k and x_{min}^k refers to their points, respectively.

The heuristic will choose the point given by the quantitative semantic that corresponds to the largest variance in strategy 1; largest distance in strategy 2. These heuristics are thus selecting the quantitative semantic that has a clear sense of direction and avoids using semantic that has close to constant objective values resulting in small variance and distance. However, other heuristics could be used as well, but a more thorough evaluation of possible heuristics is future research.

3 Benchmark Problems

Three examples are considered here to show the performance of the multiple objective functions approach, that are also used in [9], below is a brief description of the examples.

Algorithm 2 Modified NM Using Multiple Objective Functions

1. Choose p random points. Start with p random points and order and re-label the vertices from lowest function value to highest function value:
$f^{Max}(x_1^1) \leq f^{Max}(x_2^1) \leq \cdots \leq f^{Max}(x_p^1)$,
$f^{Add}(x_1^2) \leq f^{Add}(x_2^2) \leq \cdots \leq f^{Add}(x_p^2)$,
Use a heuristic algorithm to choose which quantitative semantic to follow in this iteration of the optimization. Take the minimum point of semantic that wins the heuristic algorithm.
2. Let x_i denote the list of vertices in the current simplex, $i = 1, \ldots, n+1$. These points are generated from the minimum point of Step 1.
3. Order. For each objective function $i = 1, \ldots, j$, order and re-label the $n+1$ vertices from lowest function value to highest function value:
$f^{Max}(x_1^1) \leq f^{Max}(x_2^1) \leq \cdots \leq f^{Max}(x_{n+1}^1)$,
$f^{Add}(x_1^2) \leq f^{Add}(x_2^2) \leq \cdots \leq f^{Add}(x_{n+1}^2)$,
Use a heuristic algorithm to choose which quantitative semantic to follow in this iteration of the optimization, see Strategy 1 and 2 for examples.
4. Reflection. Compute the reflected point $x_r = \bar{x} + \rho(\bar{x} - x_{(n+1)})$, where $\bar{x}$ is the centroid of the n points with lowest objective function values, $\bar{x} = \sum \frac{x_i}{n},\ i = 1, \ldots, n$. The rest of the optimization will be executed with the chosen semantic, f refers to the objective function value of that semantic.
if $f(x_1) < f(x_r) < f(x_n)$ **then**
  Replace x_{n+1} with the point x_r and go to Step 8.
end if
5. Expansion.
if $f(x_r) < f(x_1)$ **then**
  Compute the expanded point x_e by $x_e = \bar{x} + \chi(x_r - \bar{x})$.
  if $f(x_e) < f(x_1)$ **then**
    Replace x_{n+1} with x_e and go to Step 8.
  else
    Replace x_{n+1} with x_r and go to Step 8.
  end if
end if
6. Contraction.
if $f(x_r) \geq f(x_n)$ **then**
  Perform a contraction between $\bar{x}$ and the best among x_{n+1} and x_r.
  if $f(x_n) \leq f(x_r) < f(x_{n+1})$ **then**
    Calculate $x_{oc} = \bar{x} + \tau(x_r - \bar{x})$ *Outside contract.*
    if $f(x_{oc}) \leq f(x_r)$ **then**
      Replace x_{n+1} with x_{oc} and go to Step 8.
    else
      Go to Step 7.
    end if
  end if
end if

if $f(x_r) \geq f(x_{n+1})$ **then**
  Calculate $x_{ic} = \bar{x} + \tau\,(x_{n+1} - \bar{x})$ *Inside contract.*
  if $f(x_{ic}) \geq f(x_{(n+1)})$ **then**
    Replace x_{n+1} with x_{ic} and go to Step 8.
  end if
end if
7. Shrink. Evaluate the n new vertices $x' = x_1 + \phi\,(x_i - x_1),\ i = 2, \ldots, n+1$.
Replace the vertices $x_2, ..., x_{n+1}$ with the new vertices $x'_2, ..., x'_{n+1}$.
8. Re-Order. Calculate the objective values for the new point for *all* the quantitative semantics and for each quantitative semantic and save it or them (If "Shrink" happens). Order and re-label the vertices of all m calculated points from lowest function value to highest function:
$f^{Max}(x_1^1) \leq f^{Max}(x_2^1) \leq \cdots \leq f^{Max}(x_m^1)$,
$f^{Add}(x_1^2) \leq f^{Add}(x_2^2) \leq \cdots \leq f^{Add}(x_m^2)$,
where k refers to the number of $n+1$ of Step 2 and plus the number of points that are reached at each the iterations (Steps 4-7).
9. Selecting semantics. Take $n+1$ of each semantic that has lowest objective function values from Step 8. Select the semantic according to the heuristic algorithm and continue with the chosen semantic, f refers to the objective function value of the chosen semantic. While the stopping condition is not reached go to Step 4, thus
if $f(x_{n+1}) - f(x_1) < \epsilon$ **then**
  Stop, where $\epsilon > 0$ is a small predetermined tolerance.
else
  Go to Step 4.
end if

3.1 Automatic Transmission (AT) Benchmark

The inputs to the model are the throttle and brake of a vehicle. The outputs of the model are the vehicle speed v, the engine speed ω, and the gear, see [15] for details.

3.2 Third Order Modulator

The third order $\Delta - \Sigma$ modulator is a model of a technique for analog to digital conversion. It has one input U, three states x_1, x_2, x_3, and three initial conditions $x_1^{init}, x_2^{init}, x_3^{init}$, see [16] for details.

3.3 Static Switched (SS) System

The static switched system is a model without any dynamics that is included as a simple case make falsification worse than only using single boolean objective function. The model has been inspired by [17].

Table 1: Specifications to falsify for the three benchmark models AT, ($\Delta - \Sigma$), and SS.

Spec.	Formula
φ_1^{AT}	$\Diamond_{[0,T]}(\omega \geq 2000)$
φ_2^{AT}	$\Box\Diamond_{[0,T]}(\omega \leq 3500 \vee \omega \geq 4500)$
φ_3^{AT}	$\Box_{[0,T]}(\neg(gear == 4))$
φ_4^{AT}	$\Diamond(\Box_{[0,T]}(gear == 3))$
φ_5^{AT}	$\bigwedge_{i=1,\ldots,4} \Box((\neg(gear == i) \wedge \Diamond_{[0,\epsilon]}(gear == i) \implies (\Box_{[\epsilon,T+\epsilon]}(gear == i)))$
φ_6^{AT}	$\Box_{[0,T]}(v \leq 85) \vee \Diamond(\omega \geq 4500)$
φ_7^{AT}	$\neg\Big((\Box_{[0,1]}gear == 1) \wedge (\Box_{[2,4]}gear == 2) \wedge (\Box_{[5,7]}gear == 3) \wedge (\Box_{[8,10]}gear == 3) \wedge (\Box_{[12,15]}gear == 2)\Big)$
φ_8^{AT}	$\Box_{[0,20]}\big((gear == 4 \wedge throttle > 45 \wedge throttle < 50) \implies \omega < \bar{\omega}\big)$
$\varphi^{\Delta-\Sigma}$	$\Box\left(\bigwedge_{i=1}^{3}(-1 \leq x_i \wedge x_i \leq 1)\right).$
φ^{SS}	$\Box(y \geq 0)$

4 Experimental Setup and Results

The experimental setup is described in more detail in [9]. The implementation starts by evaluating 100 random points for the AT example, 40 for third order $\Delta - \Sigma$ modulator example and 20 for the SS example before starting the optimization. After doing that if the falsified point is not found, the optimization solver starts from the point with lowest objective value. In Table 1, the STL specifications that should be falsified for all the models, and the benchmark models are presented.

The results of running Algorithm 1 on the benchmark problems are shown in Tables 2, 3, and 4, and the aggregated results are shown in Fig. 1. The tables are formatted as follows. The first column denotes the specification falsified. Each specification has one to three parameter values, these parameter values are shown in the second columns. The remaining columns show the different semantics including the *Max*, *Additive*, and their combination with two difference strategies. For each parameter value and semantic, two values are presented. The first value is the relative success rate of falsification, in percent. There are a total of 20 falsification run for each parameter value and objective functions, meaning that the success rate will be a multiple of 5%. The second value, inside parentheses, is the average number of needed simulations *per successful falsification*. Each falsification is set to have a maximum

Table 2: Results for the automatic transmission benchmark. For each parameter value and quantitative semantics, the first number indicates relative success ratio of falsification (%). The second number, in parentheses, indicates average number of simulations per successful falsification.

Spec.	Parameters \ Semantics	Max	Add	Mul Max-Add (Variance)	Mul Max-Add (Distance)
φ_1^{AT}	$T = 20$	**100** (138)	**100** (156)	**100** (93)	**100** (134)
	$T = 30$	85 (264)	95 (364)	95 (375)	**100** (309)
	$T = 40$	35 (315)	65 (556)	55 (561)	50 (483)
φ_2^{AT}	$T = 10$	**100** (33)	**100** (14)	**100** (20)	**100** (11)
φ_3^{AT}	$T = 4.5$	**100** (141)	90 (323)	**100** (223)	**100** (272)
	$T = 5$	**100** (65)	**100** (95)	**100** (44)	**100** (67)
φ_4^{AT}	$T = 1$	60 (505)	35 (357)	40 (367)	65 (402)
	$T = 2$	**100** (21)	**100** (19)	**100** (19)	**100** (14)
φ_5^{AT}	$T = 1$	95 (406)	75 (516)	95 (333)	**100** (330)
	$T = 2$	**100** (5)	**100** (4)	**100** (4)	**100** (6)
φ_6^{AT}	$T = 10$	50 (722)	45 (482)	55 (534)	55 (526)
	$T = 12$	**100** (236)	**100** (215)	**100** (186)	**100** (198)
φ_7^{AT}		20 (766)	75 (382)	65 (483)	90 (421)
φ_8^{AT}	$\bar{\omega} = 3000$	**100** (13)	**100** (11)	**100** (7)	**100** (10)
	$\bar{\omega} = 3500$	30 (439)	90 (375)	60 (372)	15 (323)

Table 3: Results for the Third Order $\Delta - \Sigma$ modulator. For each parameter value and quantitative semantics, the first number indicates relative success ratio of falsification (%). The second number, in parentheses, indicates average number of simulations per successful falsification.

Spec.	Parameters \ Semantics	Max	Add	Mul Max-Add (Variance)	Mul Max-Add (Distance)
$\varphi^{\Delta-\Sigma}$	$U \in [-0.35, 0.35]$	55 (331)	20 (515)	85 (361)	65 (445)
	$U \in [-0.40, 0.40]$	**100** (271)	75 (279)	**100** (228)	**100** (255)
	$U \in [-0.45, 0.45]$	**100** (73)	95 (314)	**100** (136)	**100** (141)

of 1000 simulations performed.

In addition, for each parameter value, the semantic with the highest (or tied highest) success rate has the success rate displayed in **bold** characters. For each parameter value if there are semantics with same success rate, the semantic with the lowest average number of simulations per successful simulation has that number displayed in **bold** characters (inside the parentheses).

Table 4: Results for the Static Switched System. For each parameter value and quantitative semantics, the first number indicates relative success ratio of falsification (%). The second number, in parentheses, indicates average number of simulations per successful falsification.

Spec.	Semantics / Parameters	Max	Add	Mul Max-Add (Variance)	Mul Max-Add (Distance)
φ^{SS}	$thresh = 0.7$	**100** (120)	**100** (248)	**100** (43)	**100** (89)
	$thresh = 0.8$	90 (201)	**100** (219)	**100** (114)	85 (356)
	$thresh = 0.9$	55 (511)	45 (519)	80 (334)	40 (365)

4.1 Results

As can be seen from Fig 1, the top two approaches are Multiple semantics. All two strategies of the multiple objective functions, perform better than when we only have a single objective function. They are more successful in falsifying with fewer simulations.

By looking at three tables of results, it can been seen that for automatic transmission, φ_1^{AT} $(T = 20)$, φ_3^{AT} $(T = 5)$, φ_5^{AT} $(T = 2)$, φ_6^{AT} $(T = 12)$, φ_8^{AT} $(\bar{\omega} = 3000)$, the Multi-Max-Add using variance strategy works better. While, for φ_1^{AT} $(T = 30)$, φ_2^{AT}, φ_4^{AT}, φ_5^{AT} $(T = 1)$, φ_6^{AT} $(T = 12)$, φ_7^{AT}, the Multi-Max-Add using distance works better. Only, for φ_3^{AT} $(T = 4.5)$, *Max* is better. *Additive* works for φ_1^{AT} $(T = 40)$, φ_8^{AT} $(\bar{\omega} = 3500)$.

For $\varphi^{\Delta-\Sigma}$ Benchmark, except $U \in [-0.45, 0.45]$ that *Max* performs better for that, multiple objective function using variance perform well. For all specifications of the Static Switched system, multiple objective function using variance give better results. One conclusion that can be given here is that the Max-Additive multiple objective function using both strategies work better than others, and for only a few of specifications, the single semantics works better. Only for the specifications φ_8^{AT} $(\bar{\omega} = 3500)$ and φ_5^{AT} $(T = 30)$ of the automatic transmission example, the *Additive* was successful in finding more falsification. For other semantics that *Max* or *Additive* work better still the single and multiple objective functions have same success ratio of falsification, only the number of simulations is different. As a result, the multiple objective functions using both strategies can guide the testing process towards falsification better than only single semantic is used, on the given benchmark set.

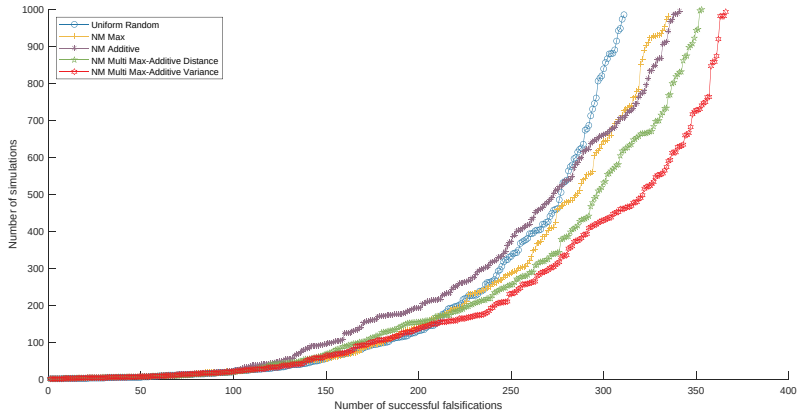

Figure 1: A cactus plot showing performance of using optimization based on multiple objective functions compared to using a single objective function. The plotted values tell how many successful falsifications (x-axis) were completed in less than a specific number of simulations (y-axis). The maximum number of simulations per falsification is 1000. Note that we also include the cactus plot for Uniform Random sampling, as a baseline approach.

5 Conclusion

In this paper, the use of multiple objective functions for falsification of CPSs was proposed. With a single objective function, the optimization may get the wrong or even no information for it to be able to guide the falsification process towards falsifying the specification. Since for CPSs, the simulation time is the most limiting factor, not the evaluation of the simulation results, multiple objective functions were suggested. Combinations of two different semantics *Max* and *Additive* were evaluated. A variance and distance based strategy were used to switch between the objective functions, such that the objective function with highest variance (distance) was picked for each iteration of the falsification. Three benchmark CPSs examples were considered to show the performance of the multiple objective functions.

The main conclusion drawn from the data gathered is that the proposed optimization algorithm perform better than when only a single objective function is used on the used benchmark examples. Also multiple objective functions are more successful in finding counterexamples in less number of simulation

runs. This so, since multiple objective functions can better guide the optimization algorithm towards falsification, increasing the chance of falsifying the specification.

For future work, it would be interesting to explore different heuristics for choosing between multiple objective functions as well as extending the number of benchmark problems to further evaluate the performance of the proposed approach.

Acknowledgements

This work was supported by the Swedish Research Council (VR) projectSyTeC VR 2016-06204 and from the Swedish Governmental Agency forInnovation Systems (VINNOVA) under project TESTRON 2015-04893.

References

[1] S. Abbaspour Asadollah, R. Inam, and H. Hansson, "A survey on testing for cyber physical system", in *Testing Software and Systems*, K. El-Fakih, G. Barlas, and N. Yevtushenko, Eds., Cham: Springer International Publishing, 2015, pp. 194–207.

[2] 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.

[3] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "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.

[4] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals", *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009, ISSN: 0304-3975.

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

[6] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals", in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[7] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems", in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.

[8] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems", in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 254–257.

[9] 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 System*, 2020, early access.

[10] 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.

[11] 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, 408 –415, 2018, 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[12] Z. Ramezani, N. Smallbone, M. Fabian, and K. Åkesson, "Evaluating two semantics for falsification using an autonomous driving example", in *2019 IEEE 17th International Conf. on Industrial Informatics*, vol. 1, 2019, pp. 386–391.

[13] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications", in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[14] J. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions", *SIAM J. on Optimization*, vol. 9, pp. 112–147, Dec. 1998.

[15] B. Hoxha, H. Abbas, and G. Fainekos, "Benchmarks for temporal logic requirements for automotive systems", *Proc. of Applied Verification for Continuous and Hybrid Systems*, 2014.

[16] 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.

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

PAPER C

Enhancing Temporal Logic Falsification with Line Optimization

Zahra Ramezani, Koen Claessen, Martin Fabian, Knut Åkesson

To be submitted to the possible journal publication

The layout has been revised.

Abstract

Because cyber-physical systems (CPS)s are complex and exhibit both continuous and discrete dynamics, it is difficult to guarantee that they satisfy given specifications, i.e., the properties that must be fulfilled by the system. Falsification of temporal logic properties is a testing approach that searches for counterexamples of given specification. Falsification can be done using random search methods or optimization methods. In this paper, a method based on combining random parameters together with considering extreme combinations of parameter values is suggested as another base-line. The evaluation result on benchmark problems shows that this method performs well on many of the benchmark problems. Optimization methods are needed for the examples that base-line methods do not perform well. The efficiency of the falsification is affected by the optimization methods used to search for inputs that might falsify the specifications. This paper suggests a new optimization method for falsification, line optimization, where optimization is done over a line. The evaluation results on benchmark problems show that using this method improves the falsification performance by reducing the number of simulations necessary to falsify a property.

1 Introduction

Cyber-physical systems (CPSs) bridge the cyber-world of communications and computing with the physical world. CPSs are widely adopted in many areas such as autonomous systems and smart grids. These systems exhibit both *continuous* and *discrete* dynamics. CPSs are often safety-critical systems, for example, autonomous cars and medical devices. It is difficult to verify that they satisfy the given specification must be fulfilled by the system. Verification and testing on models of CPSs are two common methods used for the validation and correctness of these systems. Testing is a necessary part of the design process of complex systems since formal verification of systems with a combination of discrete and continuous dynamics is an undecidable

problem [1].

CPSs are often developed in a model-based development (MBD) paradigm- [2]. For many industrial applications, there are no mathematical models to analyze, and it is only possible to simulate the system under test. thus formal verification is not a viable option. There are simulation-based approaches [3], [4] for testing of CPSs, like the falsification of temporal logic specifications.

Falsification of temporal logic properties is an approach to find the *counterexamples* to given specifications of CPSs. The falsification process is performed over the *quantitative semantics* of the specification. Quantitative semantics uses an objective function that models the distance to falsifying the specification. An objective function value is calculated to guide the falsification process towards a specification to be falsified. It can be done in this way that the next set of parameters, for the input signals to the system being simulated, be chosen such that the likelihood of falsifying the specification is increased, if it is possible. The objective function is determined by the definition of quantitative semantics for temporal logic formalisms [5] that Metric Interval Temporal Logic (MITL) [6] and Signal Temporal Logic (STL) [7] are commonly used.

Large industrial systems are a mix of continuous models, discrete event models, and source code, therefore only a *black-box* of the system under test is available. It means that only the input-output behavior of the system under test can be observed and there is no other information about the system available. S-TaLiRo [8] and Breach [9] are two MATLAB/Simulink toolboxes used for testing and falsification. One feature of these two simulation-based toolboxes is to facilitate the computation and the property investigation of large sets of trajectories that rely on an efficient numerical solver of ordinary differential equations. S-TaLiRo finds the counterexamples using of MITL properties, while Breach performs the falsification with using STL. In this paper, Breach is used since it is an open-source tool that can be easily extended and implemented.

In [10], the concept of valued booleans (VBools) was introduced as a way to express different quantitative semantics. Two quantitative semantics, *Max* and *Additive* were expressed in term of VBools to define the objective function for the falsification processes. In *Additive* semantics, the robustness of each clause in a conjunction can affect the final robustness of the full formula. The robustness is a value that assigned in the VBools. Disjunction and temporal

operators are defined in terms of conjunction for the *Additive* semantics.

Not only the different objective functions affect the outcome of falsification, but also the different optimization methods affect the falsification performance. Since the simulation-based approaches are used for falsification, hence the gradient-free optimization methods are needed. In the paper [11], different gradient-free optimizations were reviewed by a systematic comparison and evaluation of some problems. The computational results showed that there is no single optimization method whose performance dominates that of all others. Moreover, all the evaluated optimization methods in that paper provided the best solution possible for at least some of the test problems. Paper [12] evaluated the different semantics and optimization methods for falsification on some examples of ARCH benchmark problems [13]. The paper [13], is a friendly competition in the ARCH19 workshop for the falsification of temporal logic specifications over CPSs.

Two optimization methods were top in the paper [12] were Nelder-Mead (NM) [7] and SNOBFIT [14]. Both NM and SNOBFIT are gradient-free optimization methods [11], but, they have different properties. SNOBFIT [14] was one of the top optimization methods evaluated in that paper.

NM is an optimization method that is easy to implement and it can deal with functions with many variables. Moreover, NM needs only function evaluation that leads to being suitable in the minimization of non-differentiable functions. It is good method at finding the local optima. The NM starts with a simplex set of points. In each iteration, the objective function values are sorted from the lowest to highest objective value. This method attempts to replace the worst point, the point with the highest objective function value, by introducing a new vertex in a way that results in a new simplex, reflection, expansion, contractions. If shrink happens, n new points are generated by NM.

SNOBFIT is a method that can have the fast local search. It contains a parameter that controls the balance between local and global search of the optimization. The minimization in this method is done in stages, i.e., rather than only one point where the algorithm samples a specified number of points at a time. On the other hand, SNOBFIT is a complex method with a lot of MATLAB implemented codes. Hence, analyzing and understanding what is going on in a different situation by this method is hard.

Three evaluated semantics in the paper [12] were *Max*, *Additive*, and constant quantitative semantics. The later semantic works in this way that if the

specification is not falsified, a constant positive value, on the other hand, if it is falsified, a constant negative value is calculated. Therefore, the constant objective value does not indicate any information to optimization methods for falsification, i.e., no information about being far away from or close to falsify the specification. But, SNOBFIT using this constant objective value could perform even better than using NM with *Max* and *Additive* semantics. It was interesting and unexpected results that lead to end up with this question that how could SNOBFIT perform better than other optimizations? The answer to this question is that it seems when SNOBFIT does not get any information from the constant semantic, it tends to drive the new parameter values towards the extreme points. It was an inspiration for us to suggest a new optimization method that features the benefits of SNOBFIT but have low complexity and a small implementation in this paper.

Evaluation results on ARCH benchmark problems gave us this feedback that for some specifications and systems, it is easy to be falsified on the corners, i.e., the combination of extreme values of the allowed parameter ranges of the system under test. A similar condition can happen if we only consider some uniform random points for falsification. These two cases mean that the falsification process can be done without any optimization method. Since, to be able to use the advantage of both corners and random methods, a combination of the corner and random is suggested in this paper, called Hybrid Corners-Random. This method can work well for the specifications that if at least one of the corners or random points in the parameter space can be successful in the falsification process. This leads us to introduce three base-line algorithms. (i) Corners (ii) Uniform-Random, (iii) Hybrid Corners-Random. In our evaluation, the latter method surprisingly outperforms NM-based algorithms, irrespectively of the semantics used on the used benchmark set.

For specifications and examples that the Hybrid Corners-Random does not work, i.e., it is not successful in falsification process, it is reasonable to use an optimization-based method for falsification. In this paper, we propose a simple new optimization algorithm, called line optimization, for falsification that has a performance on par with, or better than, SNOBFIT but having a much simpler implementation. The algorithm is inspired by the crawling procedure in NM for local optimization, but also the ability to get out of local optima and continue the optimization from a new but related point. Evaluation results on the benchmark examples in this paper also will show

that the line optimization method performs better than NM and SNOBFIT for the falsification process.

The main contributions of this paper are:

- A Hybrid Corners-Random method that uses the combination of the corner and random methods is suggested.
- A new gradient-free optimization method is suggested, line optimization.
- The suggested methods in this paper are evaluated on ARCH benchmark problems and additional benchmark problems.

The paper is organized as follows: Section 2 introduces the quantitative semantics and different ways to define the objective functions used for the falsification process. Section 3 proposes the suggested Hybrid Corners-Random and line optimization methods. Section 4 introduces the evaluated benchmark problems in this paper. Section 5 evaluates the performance of the suggested optimization methods on benchmark problems. Finally, Section 6 summarizes the contributions.

2 Quantitative Semantics and Objective Functions

In this paper, STL is used to model the specifications. The syntax of STL is defined as follows [15]:

$$\varphi ::= \mu \,|\, \neg\mu \,|\, \varphi \wedge \psi \,|\, \varphi \vee \psi \,|\, \Box_{[a,b]}\psi \,|\, \Diamond_{[a,b]}\psi$$

where the predicate μ is $\mu \equiv \mu(s) > 0$ and s is a signal; φ and ψ are STL formulas; $\Box_{[a,b]}$ denotes the *globally* operator between times a and b (with $a < b$); $\Diamond_{[a,b]}$ denotes the *finally* operator between a and b.

The satisfaction of the formula φ with respect to the discrete signal s at

the discrete time instant k is defined as:

$$\begin{aligned}
(s,k) &\models \mu && \Leftrightarrow\ \mu(s[k]) \\
(s,k) &\models \neg\mu && \Leftrightarrow\ \neg((s,k) \models \mu) \\
(s,k) &\models \varphi \wedge \psi && \Leftrightarrow\ (s,k) \models \varphi \wedge (s,k) \models \psi \\
(s,k) &\models \varphi \vee \psi && \Leftrightarrow\ (s,k) \models \varphi \vee (s,k) \models \psi \\
(s,k) &\models \Box_{[a,b]}\varphi && \Leftrightarrow\ \forall k' \in [k+a, k+b], (s,k') \models \varphi \\
(s,k) &\models \Diamond_{[a,b]}\varphi && \Leftrightarrow\ \exists k' \in [k+a, k+b], (s,k') \models \varphi
\end{aligned}$$

Instead of only checking the boolean satisfaction of an STL formula, the notion of quantitative value, i.e., an objective value, will be defined to measure how far away a specification is from being falsified. A Valued Boolean (VBool) [10] (v, x) is a combination of a Boolean value v (true $\top$, or false $\bot$) together with a real number x that is a measure of how true or false the specification is. This value will be used as a measure of how convincingly a test passed, or how severely it failed, respectively. In the original VBool definition, x is defined to always be non-negative. However, in this paper, we use the convention that x is negative when v is false, and positive otherwise.

2.1 Quantitative Semantics

Using VBools, we define two quantitative semantics: *Max*, which is essentially the same as standard STL quantitative semantics; *Additive*; For these semantics we define the respective *and*, *or*, *always*, and *eventually* operators.

For conjunction, the semantics differ only in the two cases where the truth values are the same:

	Max	*Additive*
$(\top, x) \wedge (\top, y) =$	$(\top, \min(x, y))$	$\left(\top, \frac{1}{\frac{1}{x}+\frac{1}{y}}\right)$
$(\top, x) \wedge (\bot, y) =$	$(\bot, y)$	
$(\bot, x) \wedge (\top, y) =$	$(\bot, x)$	
$(\bot, x) \wedge (\bot, y) =$	$(\bot, \max(x, y))$	$(\bot, x + y)$

Using the de Morgan laws, the *or* operator can be defined in terms of *and*, as:

$$(v_x, x) \vee (v_y, y) = \neg_{\mathrm{v}}(\neg_{\mathrm{v}}(v_x, x) \wedge \neg_{\mathrm{v}}(v_y, y)),$$

where VBool negation is defined as $\neg_{\mathrm{v}}(v_x, x) = (\neg v_x, -x)$.

For the *Max* semantics, the *always* operator over an interval $[a, b]$ is straightforwardly defined in terms of *and*, as

$$\Box_{[a,b]}\varphi = \bigwedge_{k=a}^{b} \varphi\,[k]\,,$$

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

For the *Additive* semantics, though, *always* is a bit more elaborate:

$$\Box_{[a,b]}\varphi = \bigwedge_{k=a}^{b} \varphi\,[k]\ \#\,\delta t,$$

where δt is the simulation step size that makes the quantitative value independent of the simulation time, and $\#$ is:

$$(\bot, x)\,\#\,\delta t = (\bot, x \cdot \delta t)$$
$$(\top, x)\,\#\,\delta t = (\top, x/\delta t).$$

Furthermore, the *eventually* operator is for all three semantics defined over an interval $[a, b]$ in terms of *always*, as:

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

3 Optimization Methods

Falsification can be done with our without using an optimization method. In this paper, we introduce two optimization methods for falsification, one based on only using corner points and random points (Hybrid Corners-Random Falsification); and another one based on a gradient-free optimization algorithm (Line optimization) that combines the local search with the ability to explore new areas of the parameter space in case local optimization is unable to falsify the specification.

3.1 Hybrid Corners-Random Method

If any optimization method is not be used, two possible methods can be considered. First one is the corner points, i.e., the extreme points of the given input ranges of the system under test; second is a random approach. In this paper the uniform-random-based approach is considered.

We have observed from the corners points, that a successful strategy is to focus on these points, i.e., if the low simulation budget then we should start with the corners points. However, if a large simulation budget then random will eventually beat the corners cases. These cases can happen for the specifications that at least one of the corners or random methods are successful in the falsification process. Thus, to be able to use the advantages of both corners and random methods, a combination of these two methods can be used; Hybrid Corners-Random. This method is presented in Algorithm 1.

Algorithm 1 Hybrid Corners-Random Method for Falsification

```
1: nbr_of_simulations = 0.
2: Pick a corner point, x.
3: while nbr_of_simulations < max_nbr_simulations do
4:    Simulate system at x.
5:    nbr_of_simulations++;
6:    Evaluate spec at x.
7:    if the spec if is falsified then
8:       return x.
9:    else
10:      if nbr_of_simulations mod 2 = 0 OR corners exhausted then
11:         Pick a random point x.
12:      else
13:         Pick a corner point x.
14:      end if
15:   end if
16: end while
```

In this algorithm, it does not matter which semantic is used. It just evaluates the chosen point that it is falsified or not. This algorithm works in this way that in each iteration, a corner (random) point x is picked. Then, simulate the system at the chosen point x to evaluate the specification is falsified or not. If the specification is not falsified, then for the next iteration a random (corner) point x will be picked. This algorithm keeps working until

a maximum number of simulations, *max_nbr_simulations* is reached or the specification is falsified.

It should be mentioned here that the number of corners is different and each system has a specific maximum number of corners. Hence, if after some iterations the number of corners is finished, then the algorithm will continue with only the random points.

Although this method beats both corners and random methods, it is just a good method for the examples that at least one of the corners or random or both methods are successful in falsification. Hence, for the examples that they are not successful, an optimization method is needed. In the next section, the new line optimization method will be introduced.

3.2 Line Optimization

As was discussed in the previous section, always there are the examples that are not able to be falsified with a Hybrid Corners-Random method.

Two gradient-free optimization methods, NM and SNOBFIT are optimization methods that can be used for falsification. NM is an easy method that is good at finding a local optimum. On the other hand, SNOBFIT is a complex method that contains a parameter that controls the balance between local and global search components of the optimization. If SNOBFIT cannot find any information form the objective value, as was discussed for constant quantitative semantic, still it is successful in falsification. In this situation, SNOBFIT tries to guide the falsification search towards the extreme points. This was a primary motivation for us to introduce the line optimization method.

All optimization methods in this paper, aim to find the falsification process, i.e., the point has a negative objective value. This section introduces a gradient-free optimization method that aimed to guide the search towards the extreme points but also can go inside the input parameter ranges if that will be indicated by the objective functions.

This optimization method is presented in Algorithm 2, implemented for the falsification of CPSs in Breach.

Requirements

This method needs a system to be tested; the quantitative semantics that needed to assign the objective function values.

Algorithm 2 Line Optimization Algorithm

Initial point. Calculate $x^* = \frac{(l_b+u_b)}{2}$ and simulate the system at point, x^* and calculate the objective function value of the used semantic $f(x^*)$.
Pick a line. Pick a line and calculate two points x_L, x_R, if the spec is not falsified at x^*.
Alghorithm's loop:
while (nbr_of_simulations $\leq$ max_nbr_simulations) **do**
  if *if the spec is falsified* **then**
    Terminate the algorithm with the current evaluated point.
  else if (*counter* $>$ max_nbr_iterations) **then**
    Return the x^* and pick a new line.
    Generate two points x_R and x_L and simulate the system at these points.
    if the spec is falsified at any of x_R and x_L **then**
      Terminate the algorithm with the point that is falsified.
    end if
  else
    if The point x^* has the lowest objective value **then**
      if $||x^* - x_L|| \geq ||x_R - x^*||$ **then**
        Simulate the system at point $x_1 = (x_L + x^*)/2$, and calculate $f(x_1)$:
        if $f(x_1) < f(x^*)$ **then**
          Let (x_L, x^*, x_R) be (x_L, x_1, x^*).
        else
          Let (x_L, x^*, x_R) be (x_1, x^*, x_R).
        end if
      else
        Simulate the system at point $x_2 = (x^* + x_R)/2$, and calculate $f(x_2)$:
        if $f(x_2) < f(x^*)$ **then**
          Let (x_L, x^*, x_R) be (x^*, x_2, x_R).
        else
          Let (x_L, x^*, x_R) be (x_L, x^*, x_2).
        end if
      end if
    else if The point x^* does not have the lowest objective value: **then**
      if f_L is smallest objective value **then**
        Simulate the system at point $x_1 = (x_L + x^*)/2$, and calculate $f(x_1)$.
        Let (x_L, x^*, x_R) be (x_L, x_1, x^*).
      else if f_R is smallest objective value **then**
        Simulate the system at point $x_2 = (x_R + x^*)/2$, and calculate $f(x_2)$.
        Let (x_L, x^*, x_R) be (x^*, x_2, x_R).
      end if
    end if
  end if
end while

Output

The output of this algorithm is a point x, in n dimension, such as that:

- If the the falsification occurred, it returns $f(x) < 0$.
- If the maximum number of simulation is reached and the falsification does not occur, it returns the point that has the minimum objective value, but it is positive $f(x) \geq 0$.

Optimization Process

This method needs three points at the beginning and during the algorithm process as well, where these points will be updated and one new point in each iteration will be produced. For the falsification process of a system under test, each input parameter has a lower and upper bound i.e., the inputs are in R^n. Thus, all three needed points are in or on a lower the l_b and upper u_b the vectors that refer to the minimum and maximum of the input ranges in R^n dimension. The objective value function is the function used where $f : R^n \rightarrow R$.

The optimization process is started in step 1 of Algorithm 2 where it takes the middle point $x^* = \frac{l_b+u_b}{2}$, the point that aimed to be improved, i.e., find a point with negative objective value. The objective function value $f(x^*)$ of this point needs to be calculated. If $f(x^*) < 0$, then the algorithm terminates which means that the falsification process happens. Otherwise, a random line is needed to be picked.

Pick a random line that passes through x^*. Two new points are now defined, where they are generated from this line that cut the boundary of input ranges at least on one dimension. These two points are called x_L and x_R that at least one of the input parameters in on the upper (lower) and lower (upper) bound. More details about how the line can be chosen will be discussed later in this section. After calculating the objective function values for all these two points, if f_L or f_R is negative, then the algorithm terminates with its corresponding point. Otherwise, the optimization loop will be started from in step 3 of Algorithm 2, where we search for a new point to be able to improve x^*, if it is possible.

In this algorithm, we work with a chosen line in some iterations. Hence, a *counter* is used. The algorithm stays in the while loop with the chosen line

until the *counter* is reached to the *max_nbr_iterations*. In each iteration, if we can improve the point x^* and find a better point with lower objective function value, this *counter* resets to *zero*. Otherwise, it must be increased by 1. When this *counter* is reached to *max_nbr_iterations*, we get out from the current line and a new line is needed to be picked, (steps 7-12)

Three points x^*, x_L, and x_R can have different cases:

1. **The point x^* has the lowest objective value:** Two cases can happen in while loop of Algorithm 2 that depends on which interval (x_L, x^*) or (x^*, x_R) is larger, steps 15-29.

2. **The point x^* does not have the lowest objective value:** Compute a fourth new point, depending on which (f_L) or (f_R) has the lowest objective value, presented in while loop in steps 30-37 of Algorithm 2.

As was mentioned, only we allow being in one line until the *counter* is not reached to its maximum value. When the *counter* is reached to the maximum number of iterations, *max_nbr_iterations*, then we need to pick a new line i.e., considering the new points of x_L and x_R with their objective values. Here two different assumptions can be considered to choose x^*. We can force the algorithm to never returns the same x^* as was given as to the algorithm. This helps the algorithm to work better because it does not get stuck in a local minimum.

While the counter is greater than the *max_nbr_iterations* and we could not improve the x^*, then just instead of considering the same x^* to pick a new line, we will select the second-best point that has the lowest objective value generated in the while loop as x^*. This assumption can help to increase the chance to pick the better line and not wasting the iterations with working with an unsuitable point. On the other hand, if x^* is improved and a better point with less objective value is found during the optimization process, thus we work with the improved current x^* and a new line will be picked.

This algorithm works until the *max_nbr_simulations* is reached or a falsified point is found, i.e., a point with negative objective function value.

Note: The line can be a line that passes from x^* or it can be two lines that connect x^* to x_L and x^* to x_R. It depends on the chosen line that will be discussed in the next part.

How to choose a line?

There are different ways to pick a line. For example: considering a slope in a specific range, for instance between $[-1, 1]$, and try to randomly pick a slope and add this to x^* and continue from each side to at least cut one limitation inputs in the n dimension. Another way used in this paper will be discussed below.

One random point is considered inside the $[l_b, u_b]$ and only in one dimension this point on is on its corner point i.e., $l_b(i)$ or $u_b(i)$, where $i = 1, \cdots$, or n. This assumption allows us to be able to work with points that are at least on one border of the input ranges. Lets call this point x_R, now according to the distance between this point and x^*, the third point, x_L, will be generated. If the point x_R cuts at $l_b(i)$, then x_L will cut at $u_b(i)$. On the other hand, if the point x_R cut at $u_b(i)$, then x_L will cut at $l_b(i)$. By having these three points x^*, x_R and x_L, a line is generated. Therefore, according to the chosen line discussed above, three situations can occur:

- **Case 1:** The length of the line is fully in the box. One of x_L and x_R that is fully in the box such that we allow for an extra piece of the line length to stick out on either side of the box.

- **Case 2:** The line leaves the box $[l_b, u_b]$, i.e., at least one of the dimensions is not in the box anymore. We try to force the line to be on the close boundary.

- **Case 3:** The line leaves the box $[l_b, u_b]$, i.e., all dimensions are not in the box anymore. We force the point to be on the closet corner point.

To better understanding how we choose the line according to the above three situations, examples in Fig. 1 are given. In this figure, we assume that x_R is on the upper bound of its range and now we want to decide about the point x_L. It should be mentioned here that similar assumptions can be considered for x_R, too. For illustrative purposes, we will show the examples in two dimensions i.e., we have two input parameters that they have a lower and upper boundary in each dimension. We are allowed to take points within the grey input parameter ranges box.

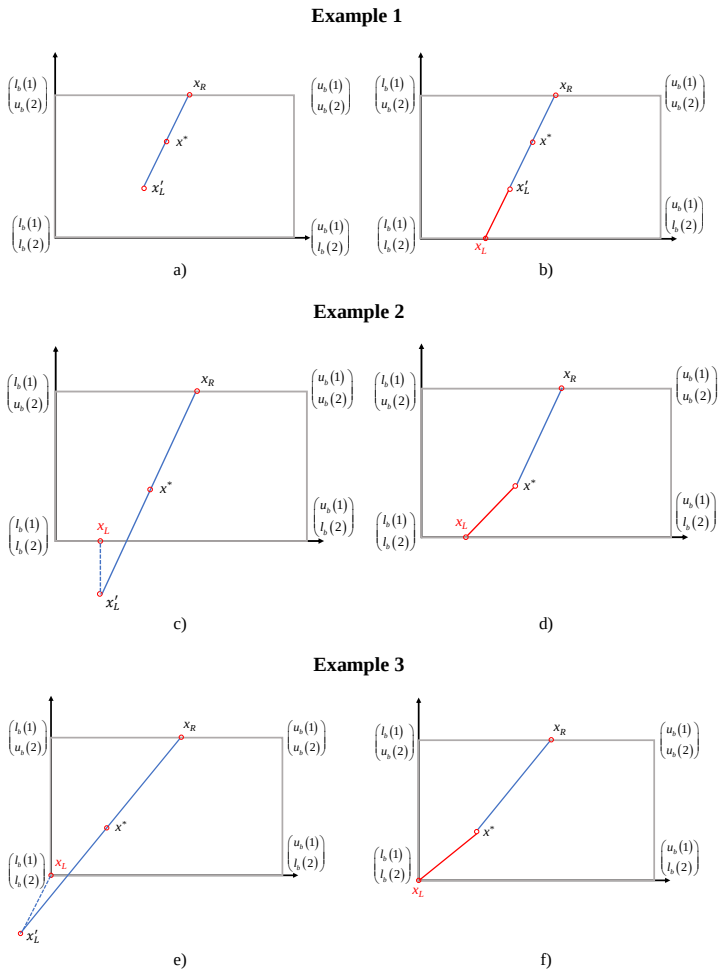

Figure 1: Examples in two dimension: Example 1 where x_L is inside the $[l_b, u_b]$; Example 2 where x_L outside the $[l_b, u_b]$ in one dimension; Example 3 where x_L is outside the $[l_b, u_b]$ in both dimensions.

Example for Case 1:

First, the line ends up inside the working ranges, i.e., in $[l_b, u_b]$. Example 1 in Fig. 1 shows this situation where the line ends up at point x'_L (graph a). Since that x_R is on the upper bound of the first dimension, then we will continue this line until to cut the lower bound of the first dimension. Thus, the x_L will be produced (graph b). In this situation, we work with only one line that passes from x^* and ends at two other points x_R and x_L.

Example for Case 2:

In example 2 of Fig. 1, the line ends up outside the working box, point x'_L, where only the second dimension is outside (graph c). Therefore, we force the point x_L to be on the border for the second dimension that results in the x_L in graph d. Similarly, if this situation happens for the first dimension, we will force it to be on the the closet boundary. In this situation, we have two lines: one the blue one that connects x^* to x_R; second, the red line that connects x^* to x_L.

Example for Case 3:

The third situation, in both dimension the line is outside the working box, x'_L in graph e of Fig. 1. Thus, we force the x_L to be equal to the extreme point $\begin{pmatrix} l_b(1) \\ l_b(2) \end{pmatrix}$ (graph f). In this situation, we have two lines: one the blue one that connects x^* to x_R; second the red line that connects x^* to x_L.

The assumptions are considered in the Case 2 and Case 3 where the line leaves the working input box helps to be able to towards the extreme points and also leads to being able to work with the points that at least in some dimensions we are on the border of the working input ranges box.

In the line optimization method, if point x^* is on a corner or close to that, we can point towards the middle of the box and not just be in a small area.

Admittedly, there are many ways to pick the random lines and possible different assumptions can be considered for the situations above in cases 1-3. We make no claim that the presented choosing line is the "best" way to use. How to choose a best line is one open question for future research.

In the next section, to evaluate the suggested method, some benchmark problems are considered will be explained in the next section.

4 Benchmark Problems

In this paper, we consider the simulation-based falsification for all benchmarks examples of ARCH workshop [13] and additional benchmark problems [16]. The additional benchmark problems are considered in this paper because there is a lot of specifications and examples in the ARCH examples that can be falsified easily. Although still, the additional benchmarks are easy to falsify, they are considered because we can have more examples to show the performance of the suggested optimization methods better. Table 1 demonstrates the STL specifications that should be falsified for all the example models. These examples briefly are introduced in the following.

4.1 ARCH Benchmark Problems

For some ARCH Benchmark problems, two input options are considered, called instance 1 and 2.

- *Instance 1: Arbitrary piece-wise continuous input signals.* This option considers the entire set of piecewise continuous input signals where the values for each individual dimensions are from a given range. Additional constraints are considered are the finite-number of discontinuity and finite variability for all continuous parts of inputs.

- *Instance 2: Constrained input signals.* This option fixes the format of the input signal precisely. An example input signal would be piecewise constant with k equally spaced control points, with ranges for each dimension of the input.

For more details about these two instances see paper [13].

Automatic Transmission (AT)

The controller of this model selects a gear 1 to 4 that depends on two inputs: *throttle, brake*; rotations per minute (ω); and car speed (v) [17]. Two different control inputs are considered for this example:

- Instance 1: $0 \leq throttle \leq 100$ and $0 \leq brake \leq 325$, where both can be active at the same time.

Table 1: Specifications to falsify for all benchmark examples

Spec.	Formula
φ_1^{AT}	$\square_{[0,20]}(v < 120)$
φ_2^{AT}	$\square_{[0,10]}(\omega < 4750)$
φ_3^{AT}	$\square_{[0,30]}\Big(\big(\neg g_1 \wedge \circ\, g_1\big) \implies \circ\, \square_{[0,2.5]} g_1\Big)$
φ_4^{AT}	$\square_{[0,30]}\Big(\big(\neg g_2 \wedge \circ\, g_2\big) \implies \circ\, \square_{[0,2.5]} g_2\Big)$
φ_5^{AT}	$\square_{[0,30]}\Big(\big(\neg g_3 \wedge \circ\, g_3\big) \implies \circ\, \square_{[0,2.5]} g_3\Big)$
φ_6^{AT}	$\square_{[0,30]}\Big(\big(\neg g_4 \wedge \circ\, g_4\big) \implies \circ\, \square_{[0,2.5]} g_4\Big)$
φ_7^{AT}	$\big(\square_{[0,30]}\omega < 3000\big) \implies \big(\square_{[0,4]} v < 35\big)$
φ_8^{AT}	$\big(\square_{[0,30]}\omega < 3000\big) \implies \big(\square_{[0,8]} v < 50\big)$
φ_9^{AT}	$\big(\square_{[0,30]}\omega < 3000\big) \implies \big(\square_{[0,20]} v < 65\big)$
φ_1^{AFC}	$\square_{[11,50]}\Big(((\theta < 8.8) \wedge (\lozenge_{[0,0.05]}(\theta > 40)) \vee (\theta > 40) \wedge (\lozenge_{[0,0.05]}(\theta < 8.8)) \implies (\square_{[1,5]}\lvert\mu\rvert < 0.008)\Big)$
φ_2^{AFC}	$\square_{[11,50]}\lvert\mu\rvert < 0.007$
φ_1^{NN}	$\square_{[1,37]}\Big(\lvert Pos - Ref\rvert > 0.005 + 0.03\lvert Ref\rvert \implies \lozenge_{[0,2]}\square_{[0,1]}\big(\neg 0.005 + 0.03\lvert Ref\rvert \leq \lvert Pos - Ref\rvert\big)\Big)$
φ_2^{NN}	$\square_{[1,37]}\Big(\lvert Pos - Ref\rvert > 0.005 + 0.04\lvert Ref\rvert \implies \lozenge_{[0,2]}\square_{[0,1]}\big(\neg 0.005 + 0.04\lvert Ref\rvert \leq \lvert Pos - Ref\rvert\big)\Big)$
φ_1^{WT}	$\square_{[30,630]}\theta \leq 14.2$
φ_2^{WT}	$\square_{[30,630]} 21000 \leq M_{g,d} \leq 47500$
φ_3^{WT}	$\square_{[30,630]}\Omega \leq 14.3$
φ_4^{WT}	$\square_{[30,630]}\lozenge_{[0,5]}\lvert\theta - \theta_d\rvert \leq 1.6$
φ_1^{CC}	$\square_{[0,100]} y_5 - y_4 \leq 40$
φ_2^{CC}	$\square_{[0,100]}\lozenge_{[0,30]} y_5 - y_4 \geq 15$
φ_3^{CC}	$\square_{[0,80]}\Big((\square_{[0,20]} y_2 - y_1 \leq 20) \vee (\lozenge_{[0,20]} y_5 - y_4 \geq 40)\Big)$
φ_4^{CC}	$\square_{[0,65]}\lozenge_{[0,30]}\square_{[0,20]} y_5 - y_4 \geq 8$
φ_5^{CC}	$\square_{[0,72]}\lozenge_{[0,8]}\Big((\square_{[0,5]} y_2 - y_1 \geq 9) \implies (\square_{[5,20]} y_5 - y_4 \geq 9)\Big)$
$\varphi^{F_{16}}$	$\square_{[0,15]} altitude > 0$
φ^{SC}	$\square_{[30,35]}\Big(87 \leq pressure \wedge pressure \leq 87.5\Big)$
$\varphi_1^{AT'}$	$\lozenge_{[0,T]}(\omega \geq 2000)$
$\varphi_2^{AT'}$	$\square\lozenge_{[0,T]}(\omega \leq 3500 \vee \omega \geq 4500)$
$\varphi_3^{AT'}$	$\square_{[0,T]}(\neg(gear == 4))$
$\varphi_4^{AT'}$	$\lozenge(\square_{[0,T]}(gear == 3))$
$\varphi_5^{AT'}$	$\bigwedge_{i=1,\ldots,4} \square((\neg(gear == i) \wedge \lozenge_{[0,\epsilon]}(gear == i) \implies (\square_{[\epsilon,T+\epsilon]}(gear == i)))$
$\varphi_6^{AT'}$	$\square_{[0,T]}(v \leq 85) \vee \lozenge(\omega \geq 4500)$
$\varphi_7^{AT'}$	$\neg\Big((\square_{[0,1]} gear == 1) \wedge (\square_{[2,4]} gear == 2) \wedge (\square_{[5,7]} gear == 3) \wedge (\square_{[8,10]} gear == 3) \wedge (\square_{[12,15]} gear == 2)\Big)$
$\varphi_8^{AT'}$	$\square_{[0,20]}\big((gear == 4 \wedge throttle > 45 \wedge throttle < 50) \implies \omega < \bar{\omega}\big)$
$\varphi^{\Delta-\Sigma}$	$\square\left(\bigwedge_{i=1}^{3}(-1 \leq x_i \wedge x_i \leq 1)\right).$
φ^{SS}	$\square(y \geq 0)$

- Instance 2: Constrained input signals permit discontinuities at most every 5 time units.

Fuel Control of an Automotive Power Train (AFC)

This system is modeled in [18]–[20]. The constrained input signal fixes that throttle θ to be piecewise constant with 10 uniform segments over a time horizon of 0 with two modes. The engine speed ω to be constant with $900 \leq \omega < 1100$.

Neural-Network Controller (NN)

This benchmark is based on the neural network of the MathWorks [1] controller. It is designed for a system that levitates a magnet above an electromagnet at a reference position. A reference value *Ref* for the position, where $1 \leq Ref$ and $Ref \leq 3$, is the only input of this model. The current position of the levitating magnet *Pos* is the output. Two different control inputs considered for this model are:

- Instance 1: The input specification requires discontinuities to be at least 3 time-units.
- Instance 2: An input signal with exactly three constant segments is required.

Wind Turbine (WT)

A simplified wind turbine model of paper [21] is considered. The input is: wind speed v; the outputs are: blade pitch angle θ, generator torque $M_{g,d}$, rotor speed Ω and demanded blade pitch angle θ_d. The wind speed is constrained by $8.0 \leq v \leq 16.0$.

Chasing Cars (CC)

The model is a simple model of an automatic chasing car [22]. CC model consists of five cars, where the first car is driven by inputs (*throttle* and *brake*),

[1] https://au.mathworks.com/help/deeplearning/ug/design-narma-l2-neural-controller-in-simulink.html

and the other four cars are driven by Hu et al.'s algorithm. The location of five cars y_1, y_2, y_3, y_4, y_5 is the system output. The inputs are:

- Instance 1: The input specifications allow any piecewise continuous signals.
- Instance 2: The input specifications constraints inputs to piecewise constant signals with control points for every 5 seconds, 20 segments.

Aircraft Ground Collision Avoidance System (F16)

The F16 aircraft and its inner-loop controller are modeled for Ground Collision. 16 continuous variables with piece-wise nonlinear differential equations are modeled [23]. The system is required to always avoid hitting the ground during its maneuver starting from all the initial conditions for roll, pitch, and yaw in the range $[0.2\pi, 0.2833\pi] \times [-0.5\pi, -0.54\pi] \times [0.25\pi, 0.375\pi]$.

Steam Condenser with Recurrent Neural Network Controller (SC)

It is a dynamic model of a steam condenser based on energy balance and cooling water mass balance controlled with a Recurrent Neural network in feedback [24]. The input can vary in the range $[3.99, 4.01]$, and the input signal should be piecewise constant with 20 evenly spaced segments.

4.2 Additional Benchmark Problems

Automatic Transmission (AT') Benchmark

The inputs to the model are the throttle and brake of a vehicle. The outputs of the model are the vehicle speed v, the engine speed ω, and the gear, see [25] for details. It should be mentioned here that this example has different specifications from the ARCH example presented in 4.1.

Third Order $\Delta - \Sigma$ Modulator

The third order $\Delta - \Sigma$ modulator is a model of a technique for analog to digital conversion. It has one input U, three states x_1, x_2, x_3, and three initial conditions $x_1^{init}, x_2^{init}, x_3^{init}$, see [26] for details.

Static Switched (SS) System

The static switched system is a model without any dynamics that is included as a simple case. The model has been inspired by [27].

5 Experimental Setup and Results

The two semantics *Max* and *Additive* are considered in this paper. The line optimization method will be compared with Nelder-Mead and SNOBFIT. Moreover, the corners and uniform-random-based methods are given to compare with the hybrid-corners-random. The results are shown in tables 2 to 14. Each falsification is set to have a maximum of 1000 simulations performed. There are a total of 20 falsifications run for each optimization method and objectives functions to account for the stochastic nature of most algorithms.

Using corners, random and hybrid-corners-random, it does not matter which semantic be used, thus they are evaluated by only one semantics.

Corners Implementation Setup

Each specification and example has a different number of corners. Thus, for some examples, the number of corners can be less than 1000, the maximum number of simulations, or more than 1000. Therefore, for those examples that it is less than 1000 search, the corners terminates after the maximum number of corners is reached.

Uniform-Random Implementation Setup

For each of 20 falsifications run, the uniform random points are generated from different seeds, i.e., 20 different seeds are considered.

Hybrid Corners-Random Implementation Setup

The Hybrid Corners-Random method starts with the corners point and the next point is the uniform-random point. It switches between the corners and uniform-random until the maximum number of simulations, 1000 here, is reached or a falsified point is found. The number of corners is limited, it depends on how many parameters the corner system under test has. If the

maximum number of corners is reached, the Hybrid Corners-Random continues the algorithm using only uniform-random points.

NM Implementation Setup

This algorithm is implemented as *fminsearch* [28] in Matlab. The implementation of all semantics for NM starts from 100 random sampling points for all examples. After evaluation of 100 random sampling points, if falsification does not happen, i.e., the point with negative objective value, then the NM method starts. This method uses a simplex of $n+1$ points for n-dimensional vectors. These points are generated from the minimum point, the point with the lowest objective value, of 100 random points. This algorithm is implemented as *fminsearch* [28] in Matlab. There are the options input argument in the implementation of NM which affects the algorithm used by fminsearch. The options input argument is a data structure that drives the behavior of fminsearch. It allows handling several options in a consistent and simple interface, without the problem of managing many input arguments.

The fminsearch function is sensitive to some options. We just describe some of them here to mention what assumption is considered in the implementation of NM.

First is the maximum number of iterations, *Max-Iter*. The default is $200 \times n$, where n is the number of variables. Thus, it is different for each example. Second is *Max-Fun-Evals*, the maximum number of evaluations of the function. The default is 150 in this paper.

Therefore, NM gets out of its loop, while the point with negative objective value is not found, if one of the following condition happens.

- The maximum iteration of *Max-Iter* is reached.
- The maximum of *Max-Fun-Evals* is reached.
- $f(x_{n+1}) - f(x_1) < \epsilon$, where $\epsilon > 0$ is a small predetermined tolerance.

Then, if one of the above conditions are fulfilled leads that NM starts 100 new random points and find a new point with the lowest objective value that leads to generating new $n+1$ new points.

Line Optimization Implementation Setup

The maximum fixed number of *counter* is a fixed number that can be chosen. This value in the algorithm 2 is considered be equal to 2 in this paper.

Results Tables Setups

The result tables are formatted as: the first column denotes the optimization method; the columns show the different semantics including the *Max*, *Additive*. For each optimization method and semantics, two values are presented. The first value is the relative success rate of falsification, in percent. There are a total of 20 falsifications runs for each parameter value and objectives functions, thus the success rate will be a multiple of 5%. The second value, inside parentheses, is the average number of needed simulations *per successful falsification*.

Table 2: Results for the Automatic Transmission (Instance 1).

Optimization Method	Semantics					
	φ_1^{AT}		φ_2^{AT}		φ_3^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (213)	**100** (227)	**100** (17)	**100** (8)	**100** (24)	**100** (19)
Line Optimization	**100** (34)	**100** (40)	**100** (9)	**100** (10)	**100** (37)	**100** (34)
SNOBFIT	**100** (40)	**100** (38)	**100** (6)	**100** (6)	**100** (8)	**100** (8)
Uniform-Random	0 (-)		**100** (10)		**100** (13)	
Corners	**100** (3)		**100** (2)		0 (-)	
Hybrid Corners-Random	**100** (5)		**100** (3)		**100** (27)	
	φ_4^{AT}		φ_5^{AT}		φ_6^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (14)	**100** (15)	**100** (13)	**100** (15)	**100** (57)	**100** (73)
Line Optimization	**100** (31)	**100** (36)	**100** (21)	**100** (17)	**100** (63)	**100** (112)
SNOBFIT	**100** (14)	**100** (30)	**100** (11)	**100** (9)	**100** (106)	**100** (87)
Uniform-Random	**100** (13)		**100** (11)		**100** (59)	
Corners	0 (-)		0 (-)		0 (-)	
Hybrid Corners-Random	**100** (25)		**100** (23)		**100** (117)	
	φ_7^{AT}		φ_8^{AT}		φ_9^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (43)	**100** (48)	**100** (57)	**100** (103)	**100** (43)	**100** (29)
Line Optimization	**100** (13)	**100** (24)	**100** (26)	**100** (28)	**100** (22)	**100** (12)
SNOBFIT	**100** (54)	**100** (38)	**100** (91)	**100** (70)	**100** (52)	**100** (47)
Uniform-Random	**100** (65)		**100** (96)		**100** (31)	
Corners	0 (-)		0 (-)		0 (-)	
Hybrid Corners-Random	**100** (130)		**100** (178)		**100** (61)	

Table 3: Results for the Automatic Transmission (Instance 2).

Optimization Method	Semantics					
	φ_1^{AT}		φ_2^{AT}		φ_3^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	0 (-)	0 (-)	80 (263)	90 (234)	**100** (5)	**100** (5)
Line Optimization	0 (-)	0 (-)	**100** (187)	**100** (96)	**100** (10)	**100** (13)
SNOBFIT	0 (-)	0 (-)	**100** (76)	**100** (59)	**100** (7)	**100** (5)
Uniform-Random	0 (-)		**100** (288)		**100** (5)	
Corners	0 (-)		**100** (2)		**100** (5)	
Hybrid Corners-Random	0 (-)		**100** (3)		**100** (7)	
	φ_4^{AT}		φ_5^{AT}		φ_6^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (1)	**100** (1)	**100** (1)	**100** (1)	**100** (2)	**100** (2)
Line Optimization	**100** (3)	**100** (3)	**100** (2)	**100** (2)	**100** (6)	**100** (6)
SNOBFIT	**100** (2)	**100** (1)	**100** (1)	**100** (1)	**100** (2)	**100** (2)
Uniform-Random	**100** (2)		**100** (1)		**100** (2)	
Corners	**100** (3)		**100** (3)		**100** (3)	
Hybrid Corners-Random	**100** (3)		**100** (2)		**100** (3)	
	φ_7^{AT}		φ_8^{AT}		φ_9^{AT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	0 (-)	5 (597)	0 (-)	0 (-)	0 (-)	0 (-)
Line Optimization	**100** (121)	**100** (122)	**100** (127)	**100** (214)	**100** (75)	**100** (41)
SNOBFIT	45 (357)	45 (304)	15 (621)	30 (369)	75 (194)	65 (243)
Uniform-Random	0 (-)		0 (-)		0 (-)	
Corners	0 (-)		0 (-)		0 (-)	
Hybrid Corners-Random	0 (-)		0 (-)		0 (-)	

Table 4: Results for the Fuel Control of an Automotive Power Train.

Optimization Method	Semantics			
	φ_1^{AFC}		φ_2^{AFC}	
	Max	Add	Max	Add
Nelder-Mead	**100** (148)	95 (318)	**100** (17)	**100** (13)
Line Optimization	**100** (6)	**100** (9)	**100** (2)	**100** (3)
SNOBFIT	**100** (33)	**100** (27)	**100** (8)	**100** (7)
Uniform-Random	**100** (364)		**100** (16)	
Corners	**100** (3)		**100** (3)	
Hybrid Corners-Random	**100** (5)		**100** (5)	

Table 5: Results for the Neural Network (Instance 1).

Optimization Method	Semantics			
	φ_1^{NN}		φ_2^{NN}	
	Max	Add	Max	Add
Nelder-Mead	**100** (71)	**100** (81)	15 (487)	10 (434)
Line Optimization	**100** (65)	**100** (62)	45 (400)	25 (289)
SNOBFIT	**100** (77)	**100** (87)	35 (379)	20 (440)
Uniform-Random	**100** (117)		0 (-)	
Corners	0 (-)		0 (-)	
Hybrid Corners-Random	**100** (125)		0 (-)	

Table 6: Results for the Neural Network (Instance 2).

Optimization Method	Semantics			
	φ_1^{NN}		φ_2^{NN}	
	Max	Add	Max	Add
Nelder-Mead	**100** (28)	**100** (19)	0 (-)	5 (164)
Line Optimization	**100** (25)	**100** (18)	80 (253)	90 (354)
SNOBFIT	**100** (18)	**100** (20)	75 (342)	85 (361)
Uniform-Random	**100** (29)		5 (506)	
Corners	**100** (36)		**100** (42)	
Hybrid Corners-Random	**100** (39)		**100** (83)	

Table 7: Results for the Wind Turbine.

Optimization Method	Semantics					
	φ_1^{WT}		φ_2^{WT}		φ_3^{WT}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (1)	**100** (1)	**100** (1)	**100** (1)	**100** (1)	**100** (1)
Line Optimization	**100** (3)	**100** (3)	**100** (2)	**100** (2)	**100** (2)	**100** (2)
SNOBFIT	**100** (1)	**100** (1)	**100** (1)	**100** (1)	**100** (1)	**100** (1)
Random	**100** (2)		**100** (1)		**100** (1)	
Corners	**100** (3)		**100** (3)		**100** (3)	
Hybrid Corners-Random	**100** (3)		**100** (2)		**100** (2)	

	φ_4^{WT}	
	Max	Add
Nelder-Mead	**100** (109)	**100** (79)
Line Optimization	**100** (66)	**100** (62)
SNOBFIT	**100** (62)	**100** (65)
Random	**100** (115)	
Corners	**100** (16)	
Hybrid Corners-Random	**100** (30)	

Table 8: Results for the Chasing Car (Instance 1).

Optimization Method	Semantics					
	φ_1^{CC}		φ_2^{CC}		φ_3^{CC}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (6)	**100** (11)	**100** (5)	**100** (5)	**100** (22)	**100** (21)
Line Optimization	**100** (8)	**100** (9)	**100** (8)	**100** (10)	**100** (14)	**100** (19)
SNOBFIT	**100** (4)	**100** (5)	**100** (5)	**100** (3)	**100** (5)	**100** (5)
Uniform-Random	**100** (8)		**100** (5)		**100** (17)	
Corners	**100** (3)		**100** (1)		**100** (3)	
Hybrid Corners-Random	**100** (5)		**100** (1)		**100** (5)	

	φ_4^{CC}		φ_5^{CC}	
	Max	Add	Max	Add
Nelder-Mead	0 (-)	0 (-)	**100** (45)	**100** (23)
Line Optimization	40 (381)	30 (604)	**100** (19)	**100** (37)
SNOBFIT	55 (382)	65 (375)	**100** (17)	**100** (16)
Uniform-Random	0 (-)		**100** (18)	
Corners	0 (-)		0 (-)	
Hybrid Corners-Random	0 (-)		**100** (36)	

Table 9: Results for the Chasing Car (Instance 2).

Optimization Method	Semantics					
	φ_1^{CC}		φ_2^{CC}		φ_3^{CC}	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (131)	**100** (142)	90 (245)	**100** (96)	**100** (22)	**100** (37)
Line Optimization	**100** (32)	**100** (26)	90 (304)	**100** (218)	**100** (13)	**100** (11)
SNOBFIT	**100** (32)	**100** (29)	90 (129)	**100** (125)	**100** (8)	**100** (12)
Uniform-Random	**100** (67)		**100** (157)		**100** (24)	
Corners	**100** (3)		**100** (1)		**100** (3)	
Hybrid Corners-Random	**100** (5)		**100** (1)		**100** (5)	
	φ_4^{CC}		φ_5^{CC}			
	Max	Add	Max	Add		
Nelder-Mead	0 (-)	0 (-)	**100** (152)	**100** (55)		
Line Optimization	5 (320)	5 (540)	**100** (67)	**100** (93)		
SNOBFIT	0 (-)	0 (-)	**100** (48)	**100** (38)		
Uniform-Random	0 (-)		**100** (96)			
Corners	0 (-)		**100** (21)			
Hybrid Corners-Random	0 (-)		**100** (38)			

Table 10: Results for Aircraft Ground Collision Avoidance system.

Optimization Method	Semantics	
	φ^{F16}	
	Max	Add
Nelder-Mead	25 (343)	20 (319)
Line Optimization	65 (530)	60 (399)
SNOBFIT	80 (217)	85 (271)
Uniform-Random	5 (759)	
Corners	0 (-)	
Hybrid Corners-Uniform-Random	5 (767)	

Table 11: Results for the Steam Condenser with Recurrent Neural Network Controller.

Optimization Method	Semantics	
	φ^{SC}	
	Max	Add
Nelder-Mead	0 (-)	0 (-)
Line Optimization	0 (-)	0 (-)
SNOBFIT	0 (-)	0 (-)
Uniform-Random	0 (-)	
Corners	0 (-)	
Hybrid Corners-Uniform-Random	0 (-)	

Table 12: Results for the Automatic Transmission (AT').

Optimization Method	Semantics					
	$\varphi_1^{AT'}(T=20)$		$\varphi_1^{AT'}(T=30)$		$\varphi_1^{AT'}(T=40)$	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (138)	**100** (156)	85 (264)	95 (365)	35 (315)	65 (557)
Line Optimization	**100** (63)	**100** (56)	**100** (172)	**100** (104)	**100** (241)	**100** (219)
SNOBFIT	**100** (27)	**100** (16)	**100** (44)	**100** (34)	**100** (80)	**100** (53)
Uniform-Random	**100** (168)		60 (396)		25 (267)	
Corners	**100** (1)		**100** (1)		**100** (1)	
Hybrid Corners-Uniform-Random	**100** (1)		**100** (1)		**100** (1)	
	$\varphi_2^{AT'}(T=10)$					
	Max	Add				
Nelder-Mead	**100** (33)	**100** (14)				
Line Optimization	**100** (18)	**100** (20)				
SNOBFIT	**100** (11)	**100** (10)				
Uniform-Random	**100** (20)					
Corners	**100** (2)					
Hybrid Corners-Uniform-Random	**100** (3)					
	$\varphi_3^{AT'}(T=4.5)$		$\varphi_3^{AT'}(T=5)$			
	Max	Add	Max	Add		
Nelder-Mead	**100** (141)	90 (323)	**100** (65)	**100** (96)		
Line Optimization	**100** (173)	**100** (151)	**100** (81)	**100** (61)		
SNOBFIT	**100** (89)	**100** (46)	**100** (41)	**100** (24)		
Uniform-Random	**100** (183)		**100** (79)			
Corners	**100** (11)		**100** (11)			
Hybrid Corners-Uniform-Random	**100** (21)		**100** (21)			
	$\varphi_4^{AT'}(T=1)$		$\varphi_4^{AT'}(T=2)$			
	Max	Add	Max	Add		
Nelder-Mead	60 (505)	35 (358)	**100** (21)	**100** (20)		
Line Optimization	60 (376)	85 (307)	**100** (30)	**100** (24)		
SNOBFIT	**100** (170)	60 (328)	**100** (17)	**100** (18)		
Uniform-Random	70 (407)		**100** (24)			
Corners	**100** (1)		**100** (1)			
Hybrid Corners-Uniform-Random	**100** (1)		**100** (1)			
	$\varphi_5^{AT'}(T=1)$		$\varphi_5^{AT'}(T=2)$			
	Max	Add	Max	Add		
Nelder-Mead	95 (407)	75 (516)	**100** (5)	**100** (4)		
Line Optimization	**100** (37)	**100** (127)	**100** (6)	**100** (7)		
SNOBFIT	**100** (54)	**100** (59)	**100** (3)	**100** (4)		
Uniform-Random	95 (212)		**100** (5)			
Corners	**100** (71)		**100** (22)			
Hybrid Corners-Uniform-Random	**100** (120)		**100** (10)			
	$\varphi_6^{AT'}(T=10)$		$\varphi_6^{AT'}(T=12)$			
	Max	Add	Max	Add		
Nelder-Mead	50 (722)	45 (483)	**100** (236)	**100** (215)		
Line Optimization	30 (543)	25 (317)	95 (173)	**100** (265)		
SNOBFIT	0 (-)	0 (-)	80 (481)	90 (344)		
Uniform-Random	0 (-)		90 (328)			
Corners	0 (-)		0 (-)			
Hybrid Corners-Uniform-Random	0 (-)		60 (297)			
	$\varphi_7^{AT'}$					
	Max	Add				
Nelder-Mead	20 (766)	75 (383)				
Line Optimization	25 (568)	75 (345)				
SNOBFIT	65 (546)	95 (295)				
Uniform-Random	45 (421)					
Corners	0 (-)					
Hybrid Corners-Uniform-Random	30 (621)					
	$\varphi_8^{AT'}(\bar{\omega}=3000)$		$\varphi_8^{AT'}(\bar{\omega}=3500)$			
	Max	Add	Max	Add		
Nelder-Mead	**100** (13)	**100** (12)	30 (439)	90 (375)		
Line Optimization	**100** (11)	**100** (9)	55 (324)	**100** (178)		
SNOBFIT	**100** (7)	**100** (9)	25 (462)	85 (341)		
Uniform-Random	**100** (9)		30 (389)			
Corners	**100** (3)		**100** (3)			
Hybrid Corners-Uniform-Random	**100** (5)		**100** (5)			

Table 13: Results for the Third Order $\Delta - \Sigma$ Modulator.

Optimization Method	Semantics					
	$\varphi_1^{\Delta-\Sigma}(U \in [-0.35, 0.35])$		$\varphi_1^{\Delta-\Sigma}(U \in [-0.40, 0.40])$		$\varphi_1^{\Delta-\Sigma}(U \in [-0.45, 0.45])$	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	65 (459)	25 (320)	**100** (301)	90 (399)	**100** (196)	**100** (140)
Line Optimization	**100** (249)	**100** (254)	**100** (52)	**100** (48)	**100** (39)	**100** (50)
SNOBFIT	**100** (180)	95 (160)	**100** (38)	**100** (47)	**100** (21)	**100** (28)
Uniform-Random	0 (-)		95 (313)		**100** (262)	
Corners	0 (-)		**100** (3)		**100** (6)	
Hybrid Corners-Uniform-Random	0 (-)		**100** (5)		**100** (11)	

Table 14: Results for the Static Switched System.

Optimization Method	Semantics					
	$\varphi_1^{SS}(thresh = 0.7)$		$\varphi_1^{SS}(thresh = 0.8)$		$\varphi_1^{SS}(thresh = 0.9)$	
	Max	Add	Max	Add	Max	Add
Nelder-Mead	**100** (62)	**100** (68)	**100** (80)	**100** (130)	80 (437)	95 (295)
Line Optimization	**100** (66)	**100** (47)	**100** (60)	**100** (119)	**100** (121)	**100** (192)
SNOBFIT	**100** (22)	**100** (22)	**100** (128)	**100** (63)	80 (304)	75 (208)
Uniform-Random	**100** (42)		**100** (84)		85 (261)	
Corners	**100** (2)		**100** (2)		**100** (2)	
Hybrid Corners-Uniform-Random	**100** (3)		**100** (3)		**100** (3)	

5.1 Hybrid Corners-Random vs. Corners and Random

First, the comparison among the corners, uniform-random-based, and the Hybrid Corners-Random methods are given. According to the tables 2-14 it can be seen that for many of the specifications, at least one of the corners and random can work well. Now, we go through all examples with details for better evaluation.

AT example instance 1, in Table 2, is one of the examples that for all of the specifications, one of the corners and random, or even both can falsify the specification with rate 100%. The specification φ_1^{AT} is only the specification that can be falsified with the corners. On the other hand, all specifications φ_3^{AT}, φ_4^{AT}, φ_5^{AT}, φ_6^{AT}, φ_7^{AT}, φ_8^{AT}, φ_9^{AT}, the uniform-random works well. Only, both methods can work well for φ_2^{AT}. Thus, if we only have only one corners or uniform-random, then we cannot get a falsified point for all specifications, thus the Hybrid Corners-Random methods can help to be a full success (100 %) in the falsification of all specifications in this example.

As can be seen in Table. 3 for instance 2 of AT example, the specifications φ_1^{AT}, φ_7^{AT}, φ_8^{AT}, φ_9^{AT} are the hard examples that none of the corners and uniform-random are successful in falsification. Thus, the Hybrid Corners-Random method does not help for these specifications to be falsified. This is

the situation that optimization methods are needed that will be discussed in the optimization part. On the other hand, the specifications φ_2^{AT}, φ_3^{AT}, φ_4^{AT}, φ_5^{AT} and φ_6^{AT} are falsified by both the corners and uniform-random. Thus, the Hybrid Corners-Random is also good for these specifications.

In the AFC example, the corners method is the best method for both specifications of this example especially in less simulation compare to the uniform-random, Table 4. The Hybrid Corners-Random helps here to improve the uniform-random and beats the uniform-random, then falsification is improved.

In the NN example, instance 1 for specification φ_1^{NN}, Hybrid Corners-Random improves the falsification and beats the corners method. In instance 2 of this specification both the corners and uniform-random work well, then Hybrid Corners-Random is well, too. On the other hand, for φ_2^{NN}, for instance 1, it is hard to be falsified with these methods, and optimization methods are needed. For instance 2 of this specification, the corners are successful thus the Hybrid Corners-Random works well in this example.

WT example is an example that is falsified easily by both the corners and uniform-random. Compare to the first three specifications, a little φ_4^{WT} is hardest and needed more simulations, but still, all three methods corners, uniform-random, and Hybrid Corners-Random are successfully falsify the specification .

In the CC example for both instances, all specifications (except φ_4^{CC} where an optimization method is needed) are possible to falsify easily. Also, φ_5^{CC} of instance 1, is the one example that the uniform-random beats the corners, thus the Hybrid Corners-Random improves the corners.

F16 example needs an optimization method. Although the uniform-random finds only one out of 20 times of testing, still it is not a good method and does not help the Hybrid Corners-Random to be more efficient. It should be motioned this note here that in this example, the number of corners is 8 and the most iterations of the Hybrid Corners-Random are done by picking random points.

SC is one example that it does not matter which semantic or optimization method is used because it is not possible to find falsification with none of them. In the paper [21], they tried to use Simulated Annealing (SA) global search in combination with an optimal control based local search on the infinite-dimensional input space to successfully falsify the specification.

In the AT', for all specifications of $\varphi_1^{AT'}$, $\varphi_2^{AT'}$, $\varphi_3^{AT'}$, $\varphi_4^{AT'}$, $\varphi_5^{AT'}$, and $\varphi_8^{AT'}$

it is possible to be falsified on the corners better than the uniform-random. As a result, the Hybrid Corners-Random is successful here. On the other hand, the uniform-random performs better than the corners in $\varphi_6^{AT'}(T = 12)$ and $\varphi_7^{AT'}$. Hybrid Corners-Random beats the corners here. Only the $\varphi_6^{AT'}(T = 10)$ needs an optimization method.

For modulator example, both corners and uniform-random methods work well for the specification $\varphi_1^{\Delta-\Sigma}(U \in [-0.40, 0.40])$ and $(U \in [-0.45, 0.45])$ (corners is much better). Hence, Hybrid Corners-Random is fully successful for them. On the other hand, for the specification $\varphi_1^{\Delta-\Sigma}(U \in [-0.35, 0.35])$, an optimization method is needed.

SS system is one of the examples that is falsified on the corners, thus the Hybrid Corners-Random approach performs well.

Conclusion

To be able to have a better comparison among these three methods a cactus plot of the results is shown in Fig 2. As it can be seen in this figure, while random is more successful in falsification, the corners method finds the falsified point for those examples that is falsified on the corners so fast in the less simulation. Thus, the Hybrid Corners-Random beats both methods and enhances the number of falsification.

It should be mentioned here that most of the available benchmark problems in this paper are so easy to be falsified using the corners or random methods. It is only for a subset of the specifications and models that optimization methods are needed compared to using corners or a random approach. This subset is discussed in the next part.

5.2 Line Optimization vs. Nelder-Mead and SNOBFIT

This part compares the results of the new line optimization method with the NM and the SNOBFIT for those specifications and examples that are not possible to be falsified by the corners and uniform-random. We have the specifications 7, 8, 9 of AT example in instance 2; the second specification of the NN example in instance 1; the fourth specification of both instance 1 and 2 of CC example; F16 example; the specifications 6 for both $T = 10, T = 12$ and 7 of AT' and the first specification of the Modulator example.

For some specifications and systems, there is a clear tendency that a specific

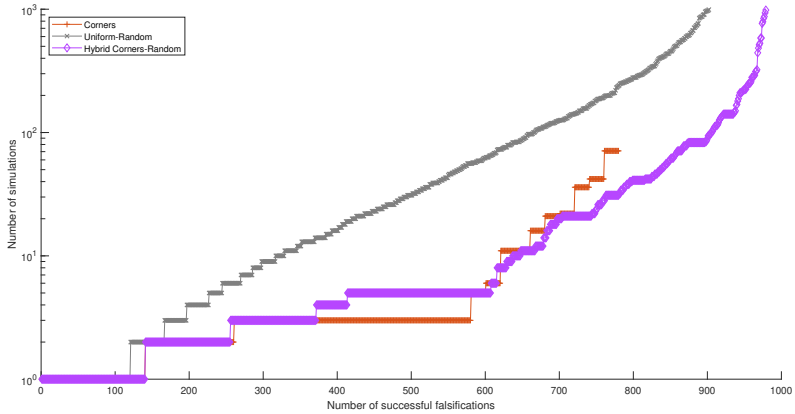

Figure 2: A cactus plot showing performance of corners, uniform-random and Hybrid Corners-Random. The plotted values show how many successful falsifications (x-axis) were completed in less than a specific number of simulations (y-axis). The maximum number of simulations per falsification is 1000.

optimization and semantics performs better.

As can be seen in Table. 3 AT example instance 2 is a good example to show the performance of the line optimization, where a significant enhancing in falsification success rate happens. It can be seen that an improvement from 0 in the NM to 100 percent in the line optimization happens for the specifications φ_7^{AT}, φ_8^{AT} and φ_9^{AT}. A huge difference between the line optimization and SNOBFIT methods can be seen in this example, too. The line optimization is falsified in 100% and it performs significantly better than the SNOBFIT.

In the NN example, for instances 1, specification φ_2^{NN}, enhancing falsification happens for both semantics, where the line optimization performers better than the NM and the SNOBFIT.

In the CC example instance 1, enhancing the falsification happens in specification φ_4^{CC} while NM is not successful in the falsification process, the SNOBFIT and the line optimization are successful in some runs where the SNOBFIT works slightly better. In instance 2 of the CC example, although only 5% improvement happens in CC, instance 2 using the line optimization, it gives this result that it is possible to falsify this specification.

In the F16 example, good improvement and performance of the line opti-

mization can be seen compared to the NM, but slightly the SNOBFIT gives better results.

In the AT', only in both specifications $\varphi_6^{AT'}(T = 10), (T = 12)$, this is the NM performs well totally better than two others. Compare to the SNOBFIT that is not successful in falsification, the line optimization performs better and it finds the falsification in some falsification run. $\varphi_7^{AT'}$ is the only specification of this example that the SNOBFIT works slightly better than the line optimization.

In $\Delta-\Sigma$ Modulator example, the good performance of the line optimization can be seen in Table 13 where the falsification in specification $\varphi_1^{\Delta-\Sigma}(U \in [-0.35, 0.35])$ is enhanced to 100% using the line optimization compare to NM and SNOBFIT for both semantics.

To better compare the optimization methods, aggregated results are shown in Figure 3. As can be seen in this figure and also according to the discussion above, the line optimization works better than NM. Since simulations are costly, the line optimization can find the falsified points fast and towards the falsification area using fewer simulations. Also, the line optimization works better than the SNOBFIT while line optimization has the simpler implementation. Compared to the semantics, the *Additive* works slightly better than *Max*.

5.3 Comparison of all optimization methods

The overall comparison of all algorithms is presented in Figure 4. The Hybrid Corners-Random beats not only the corners and uniform-random, but also beats both the NM using *Max* and *Additive*. It can be seen that the line optimization is more successful in falsification compared to the other evaluated optimization methods. It should be mentioned that this conclusion is made by evaluation on the available benchmark problems in this paper.

Usually in the industrial, there is this possibility to run the different optimization methods parallel on a computing cluster. It helps to identify the examples that are falsified on the corners or randomly fast. As a result, it is not need to consider and run any optimization method for those examples. It can be more efficient to get the falsifications results so fast.

Still, for better evaluation of the optimization methods specially the suggested line optimization in this paper, more examples and specifications are needed, where not be falsified easily using the corners and random points.

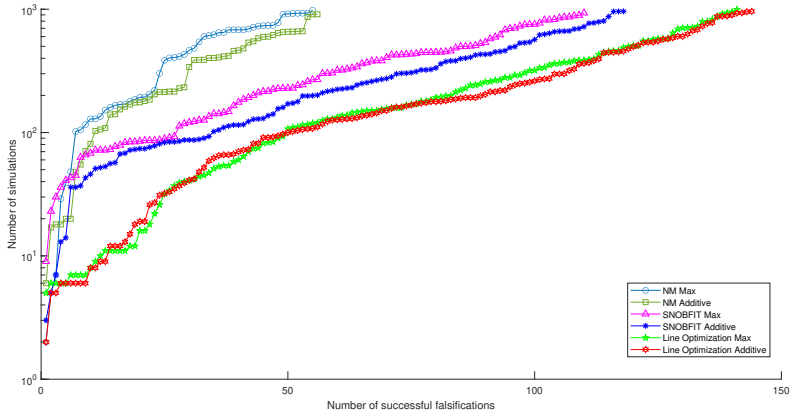

Figure 3: A cactus plot showing performance of each combination of quantitative semantics and optimization methods. The plotted values tell how many successful falsifications (x-axis) were completed in less than a specific number of simulations (y-axis). The maximum number of simulations per falsification is 1000.

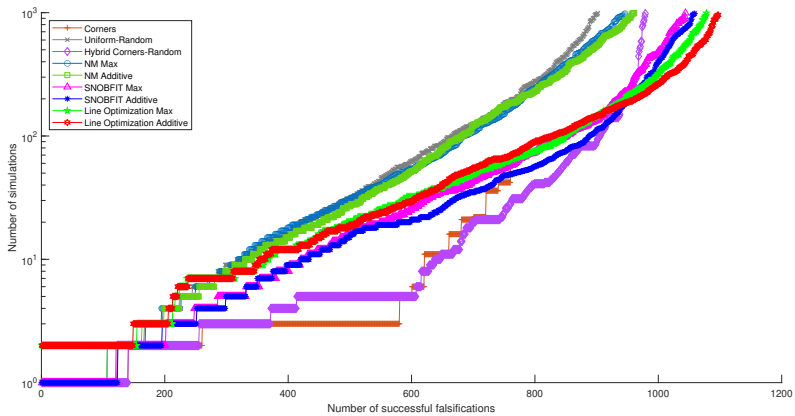

Figure 4: A cactus plot showing performance of each combination of quantitative semantics and all base-line and optimization methods. The plotted values tell how many successful falsifications (x-axis) were completed in less than a specific number of simulations (y-axis). The maximum number of simulations per falsification is 1000.

6 Conclusions

In this paper, two optimization methods were suggested to enhance the falsification of CPSs. The first method is the combination of the corners and uniform-random method, Hybrid Corners-Random. This method could help to get the falsification point so fast for those examples that can be falsified easily in few simulations on the corners or randomly. On the other hand, there are specifications and examples that none of the corners and uniform-random were efficient to be used, thus optimization methods that can falsify efficiently is needed.

The second proposed method is a gradient-free optimization method, called line optimization, that do the optimization over a line. This method is compared to the Nelder-Mead and the SNOBFIT methods. The proposed method is evaluated on standard benchmark problems and the line optimization method is able to, in general, reduce the number of simulations needed to falsify specifications.

A future work is to evaluate the suggested line optimization method on industrial applications.

References

[1] 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.

[2] P. Mosterman, "Model-based design of embedded systems", in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 2007, pp. 3–3.

[3] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Nickovic, and S. Sankaranarayanan, "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.

[4] 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.

[5] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals", *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009, ISSN: 0304-3975.

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

[7] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals", in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech and S. Yovine, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

[8] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A tool for temporal logic falsification for hybrid systems", in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 254–257.

[9] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems", in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 167–170.

[10] 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, 408 –415, 2018, 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[11] L. Rios and N. Sahinidis, "Derivative-free optimization: A review of algorithms and comparison of software implementations", *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[12] J. L. Eddeland, S. Miremadi, and K. Åkesson, "Evaluating optimization solvers and robust semantics for simulation-based falsification", in *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems*, 2020.

[13] G. Ernst, P. Arcaini, A. Donzé, G. Fainekos, L. Mathesen, G. Pedrielli, S. Yaghoubi, Y. Yamagata, and Z. Zhang, "ARCH-COMP 2019 Category Report: Falsification", in *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 61, pp. 129–140, 2019.

[14] 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.

[15] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications", in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[16] Z. Ramezani, J. L. Eddeland, K. Claessen, M. Fabian, and K. Åkesson, "Multiple objective functions for falsification of cyber-physical systems", in *Accepted to 2020 Workshop on Discrete Event Systems*, 2020.

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

[18] 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*, ser. HSCC '14, Berlin, Germany: Association for Computing Machinery, 2014, pp. 253–262.

[19] A. Dokhanchi, S. Yaghoubi, B. Hoxha, and G. Fainekos, "Arch-comp17 category report: Preliminary results on the falsification benchmarks", in *ARCH@CPSWeek*, 2017.

[20] A. Dokhanchi, S. Yaghoubi, B. Hoxha, G. Fainekos, G. Ernst, Z. Zhang, P. Arcaini, I. Hasuo, and S. Sedwards, "Arch-comp18 category report: Results on the falsification benchmarks", in *ARCH@ADHS*, 2018.

[21] 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.

[22] 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.

[23] G. Frehse, A. Abate, D. Adzkiya, L. Bu, M. Giacobbe, M. S. Mufid, and E. Zaffanella, "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.

[24] S. Yaghoubi and G. Fainekos, "Gray-box adversarial testing for control systems with machine learning component", *CoRR*, vol. abs/1812.11958, 2018.

[25] B. Hoxha, H. Abbas, and G. Fainekos, "Benchmarks for temporal logic requirements for automotive systems", *Proc. of Applied Verification for Continuous and Hybrid Systems*, 2014.

[26] 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.

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

[28] J. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions", *SIAM J. on Optimization*, vol. 9, pp. 112–147, Dec. 1998.